Technische Universität München
Max-Planck-Institut für Quantenoptik

Normalizer Circuits and Quantum Computation

Juan Bermejo-Vega

PHYSIK
DEPARTMENT

Vollständiger Abdruck der von der Fakultät für Physik der Technischen Universität München zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften (*Dr. rer. nat.*) genehmigten Dissertation.

Vorsitzender:	Univ-Prof. Dr. Alexander Holleitner
Prüfer der Dissertation:	1. Hon-Prof. Juan Ignacio Cirac, Ph.D.
	2. Univ-Prof. Dr. Robert König
	3. Univ.-Prof. Dr. Jens Eisert, Freie Universität Berlin (nur schriftliche Beurteilung)

Die Dissertation wurde am 26.11.2015 bei der Technischen Universität München eingereicht und durch die Fakultät für Physik am 01.02.2016 angenommen.

"This—" He indicated his sword again, seeing Bellis begin to understand. "—is a sword of possible strikes. A Possible Sword. It's a conductor for a very rare kind of energy. It's a node in a circuit, a possibility machine. This—" He patted the little pack strapped to his waist. "—is the power: a clockwork engine. These," the wires stitched into his armor, "draw the power up. And the sword completes the circuit. When I grip it, the engine's whole.

If the clockwork is running, my arm and the sword mine possibilities. For every factual attack there are a thousand possibilities, nigh-sword ghosts, and all of them strike down together."

Doul sheathed the blade and stared up into the trees' pitch-black canopy.

"Some of the most likely are very nearly real. Some are fainter than mirages, and their power to cut…is faint. There are countless nigh-blades, of all probabilities, all striking together."

China Miéville, *The Scar.*

Abstract

In this thesis, we introduce new models of quantum computation to study the potential and limitations of quantum computer algorithms. Our models are based on algebraic extensions of the qubit Clifford gates (CNOT, Hadamard and $\pi/4$-phase gates) and Gottesman's stabilizer formalism of quantum codes. We give two main kinds of technical contributions with applications in quantum algorithm design, classical simulations and for the description of generalized stabilizer states and codes.

Our first main contribution is a formalism of restricted quantum operations, which we name the *normalizer circuit formalism*, wherein the allowed gates are quantum Fourier transforms (QFTs), automorphism gates and quadratic phase gates associated to a set G, which is either an abelian group or an abelian hypergroup. These gates extend the qubit Clifford gates, which only have non-universal quantum computational power and can be efficiently simulated classically, to comprise additional powerful gates such as QFTs, which are central in Shor's celebrated factoring algorithm. Using our formalism, we show that normalizer circuit models with different choices of G encompass famous quantum algorithms, including Shor's and those that solve abelian Hidden Subgroup Problems (HSP). Exploiting self-developed classical-simulation techniques, we further characterize under which scenarios normalizer circuits succeed or fail to provide a quantum speed-up. In particular, we derive several no-go results for finding new quantum algorithms with the standard abelian Fourier sampling techniques. We also devise new quantum algorithms (with exponential speedups) for finding hidden commutative hyperstructures. These results offer new insights into the source of the quantum speed-up of the quantum algorithms for abelian and normal HSPs.

Our second main contribution is a framework for describing quantum many-body states, quantum codes and for the classical simulation of quantum circuits. Our framework comprises algebraic extensions of Gottesman's Pauli Stabilizer Formalism (PSF) [1], in which quantum states/codes are written as joint eigenspaces of stabilizer groups of commuting Pauli operators. We use our framework to obtain various generalizations of the seminal Gottesman-Knill theorem [2, 3], which asserts the classical simulability of Clifford operations. Specifically, we use group and hypergroup theoretic methods to manipulate novel types of stabilizer groups and hypergroups, from infinite continuous ones, to others that contain non-monomial non-unitary stabilizers and mimic reactions of physical particles. While the PSF is only valid for qubit and (low dimensional) qudit systems, our formalism can be applied both to discrete and continuous-variable systems, hybrid settings, and anyonic systems. These results enlarge the known families of quantum states/codes that can be efficiently described with classical methods.

This thesis also establishes the existence of a precise connection between the quantum algorithm of Shor and the stabilizer formalism, revealing a common mathematical structure in several quantum speed-ups and error-correcting codes. This connection permits a beautiful transfer of ideas between the fields of quantum algorithms and codes, which lies at the roots of our methods and results.

Zusammenfassung

In dieser Doktorarbeit führen wir neue *quantum-computing*-Modelle ein, um das Potential und die Einschränkungen von Quantenalgorithmen zu untersuchen. Unsere Modelle basieren auf algebraischen Erweiterungen der Clifford-Gatter für Qubits (CNOT-, Hadamard-, und $\pi/4$-Phasen-Gatter) und auf Gottesmans Stabilisator-Formalismus für Quantencodes. Wir legen hier zwei Arten von zentralen technischen Beiträgen und Ergebnissen vor und zeigen deren Anwendung für das Design von Quantenalgorithmen, die klassische Simulation von Quantensystemen und die Beschreibung verallgemeinerter Stabilisator-Zustände und -Codes auf.

Unser erstes Hauptergebnis ist ein Formalismus zur Beschreibung eingeschränkter Quantenoperationen, den wir als Normalisator Schaltkreis-Formalismus (*normalizer circuit formalism*) bezeichnen. Darin sind die folgenden Quantengatter erlaubt: die Quanten-Fouriertransformationen (QFTs), Automorphismen-Gatter und quadratische Phasengatter, die alle zu einer Menge G assoziiert sind, die entweder eine abelsche Gruppe oder eine abelsche Hypergruppe darstellt. Diese Gatter erweitern die Qubit-Clifford-Gatter, die kein universelles Quantencomputing erlauben und (mit einem klassischen Computer) effizient simuliert werden können, um mächtige zusätzliche Gatter wie z.B. die QFTs, die in Shors gefeiertem Algorithmus eine zentrale Rolle spielen. Mit unserem Formalismus zeigen wir, dass das Normalisator-Schaltkreis-Modell mit geeigneter Wahl von G wichtige Quantenalgorithmen, wie den Shor-Algorithmus und die Algorithmen zur Lösung abelscher *Hidden-Subgroup*-Probleme (HSP) umfasst. Weiterhin charakterisieren wir unter Verwendung selbstentwickelter klassischer Simulationstechniken die Szenarien, unter denen mit Normalisator-Schaltkreise eine Quantenbeschleunigung erreicht werden kann bzw. wann das nicht möglich ist. Insbesondere beweisen wir eine Reihe von No-go-Resultaten bezüglich der Möglichkeit, mit herkömmlichen abelschen Fourier-Sampling-Techniken neue Quantenalgorithmen (mit Quantenbeschleunigung) zu finden. Außerdem konstruieren wir neue Quantenalgorithmen zum Auffinden verborgener kommutativer Hyperstrukturen. Diese Ergebnisse ermöglichen neue Einsichten in die Ursache der exponentiellen Quantenbeschleunigung, die die Quantenalgorithmen zur Lösung des abelschen und normalen HSPs bieten.

Unser zweites Hauptergebnis ist ein Rahmen zur Beschreibung von Vielteilchen-Quantenzuständen und Quantencodes und zur klassischen Simulation von Quanten-Schaltkreise. Unser Rahmen umfasst algebraische Erweiterungen von Gottesmans Pauli-Stabilisator-Formalismus (PSF) [1] – in dem Quantenzustände/-codes als gemeinsame Eigenräume von Stabilisatorgruppen kommutierender Pauli-Operatoren geschrieben werden – und wir benutzen ihn, um verschiedene Verallgemeinerungen des fruchtbaren Gottesman-Knill Theorems [2, 3], das die effiziente klassische Simulierbarkeit von Clifford-Operationen beweist, abzuleiten. Genauer gesagt, verwenden wir gruppen- und hypergruppentheoretische Methoden um neue Typen von Stabilisatorgruppen und -hypergruppen zu behandeln, von unendlich-kontinuierlichen Gruppen zu solchen, die nicht-monomiale, nicht-unitäre Stabilisatoren enthalten und die Reaktionen physikalischer Teilchen nachbilden. Während der PSF nur für Qubit- (und niedrigdimensionale) Qudit-Systeme gültig ist, kann unser Formalismus sowohl auf diskrete wie kontinuierliche Systeme, auf hybride Fälle und auf anyonische Systeme angewendet werden. Diese Ergebnisse

vergrößern die bekannten Familien von Quantenzuständen/-codes, die mit klassischen Methoden effizient beschrieben werden können.

Diese Arbeit zeigt außerdem eine präzise Verbindung zwischen Shors Quantenalgorithmus und dem Stabilisatorformalismus und enthüllt eine mathematische Struktur, die zahlreichen Algorithmen mit Quantenbeschleunigung und fehlerkorrigierenden Codes gemeinsam ist. Diese Verbindung ermöglicht einen eleganten Transfer von Ideen zwischen den Quantenalgorithmen und Quantencodes und stellt die Grundlage unserer Methoden und Resultate dar.

Publications

Publications and preprints this thesis is based on:

1. Juan Bermejo-Vega and Kevin C. Zatloukal, *Abelian hypergroups and quantum computation*, preprint (2015), arXiv:1509.05806 [quant-ph].

2. Juan Bermejo-Vega, Cedric Yen-Yu Lin, Maarten Van den Nest. *The computational power of normalizer circuits over black-box groups*, preprint (2014), arXiv:1409.4800 [quant-ph].

3. Juan Bermejo-Vega, Cedric Yen-Yu Lin, Maarten Van den Nest. *Normalizer circuits and a Gottesman-Knill theorem for infinite-dimensional systems*, Quantum Information and Computation 2016, Vol 16., No 5&6 (2016), arXiv:1409.3208 [quant-ph].

4. Juan Bermejo-Vega, Maarten Van den Nest. *Classical simulations of Abelian-group normalizer circuits with intermediate measurements*, Quantum Information and Computation, Vol 14, No 3&4 (2014), arXiv:1201.4867 [quant-ph].

Other publications/preprints I contributed to:

1. Robert Raussendorf, Daniel E. Browne, Nicolas Delfosse, Cihan Okay, Juan Bermejo-Vega, *Contextuality and Wigner function negativity in qubit quantum computation*, preprint (2015), arXiv:1511.08506 [quant-ph].

Acknowledgements

My deep gratitude extends to my thesis advisors for their invaluable guidance. I am grateful to Juan Ignacio Cirac for his generous support and the outstanding research environment he provided. I thank Maarten Van den Nest for long hours of academic counseling, unforgettable times working together and teaching me his way to do science. I am grateful to Geza Giedke for his advice me during my last PhD years, fascinating research discussions, and for his help to put together this thesis.

I thank the Quantum Computing Control and Communication (QCCC), International PhD Program of Excellence and its director, Thomas Schulte-Herbrüggen, for providing a unique interdisciplinary environment to do a PhD.

This work benefited from fruitful interactions with great scientific minds. I thank my collaborators Maarten Van den Nest, Cedric Yen-Yu Lin, Kevin C. Zatloukal, Geza Giedke, Robert Raussendorf, Dan E. Browne, Nicolas Delfosse and Cihan Okay for the enthusiasm and brilliance they put in our projects together. I owe a big thanks to many quantum colleagues around the world with whom I had the great pleasure to discuss my research: chronologically, Earl T. Campbell, Mykhaylo (Mischa) Panchenko, Uri Vool, Pawel Wocjan, Raúl García-Patrón, Mari Carmen Bañuls, Liang Jiang, Steven M. Girvin, Barbara M. Terhal, Hussain Anwar, Tobias J. Osborne, Aram Harrow, Oliver Buerschaper, Martin Schwarz and Richard Jozsa.

My warmest thanks go to the members of Max Planck Institute of Quantum Optics and the MPQ Theory division, with whom I shared lovely years and experiences. I thank the MPQ Theory team as a whole for our great group atmosphere, our top-notch Wednesday Seminar series, our jolly annual retreats, the Wiesn Workshops and the heroic effort everyone made to follow my superluminal speech velocity. I thank Vanessa Paulisch and András Molnár for the Kicker matches, the excursions to the Honghong-Marat-Flex facilities and their help to print this thesis. Raúl García-Patrón, for insightful conversations on quantum time-travel, Turing machines and men's clothing in the Philosophenweg. Román Orús and our visiting fellow Ondiz Aizpuru, for helpful comments on tensor networks and tortilla de patatas. Miguel Aguado, for his clear vision on topological order, spicy food and tropes. Anna Hackenbroick, Eliška Greplová, Nayeli Azucena Rodríguez Briones, Thorsten Wahl, Alexander Müller-Hermes, Xiaotong Ni and Christine Muschik, for fun times at the office, the infamous Friedrichshafen shipwreck and math riddles. My many other colleagues of the Quantum Information subggroup, Oliver Buerschaper, Fernando Pastawski, Mari Carmen Bañuls, Hong Hao Tu, Gemma de las Cuevas, Johannes Kofler, Stefan Kühn, Yimin Ge, Henrik Dreyer and Nicola Pancotti, for our Quantum Coffee journal club series and inspiring discussions on science, pseudoscience and geopolitics. Hyung-Won Kim, Senaida Hernández Santana, Martí Perarnau, and Jordi Tura, for their memorable visits to the group. Veronika Lechner, Andrea Kluth, Lena Baumann and Karin Kügler, for countless hours of administrative help.

I am grateful to my colleagues from the MIT Center for Theoretical Physics for their hospitality during my stay in 2013, a visit that allowed me to meet great people, broadened my views of quantum computation and enriched this thesis. My sincere thanks go to Aram Har-

row, Eddie Farhi and Scott Morley for financial and administrative help to arrange the visit. Them, Cedric Yen-Yu Lin, Pawel Wocjan, Kristan Temme, Shelby Kimmel, Lina Necib, Thomas Vidick, Kamil Michnicki, Han Hsuan Lin and David Rosenbaum, for introducing me to MIT's vibrant research environment and graduate life.

During my PhD I had the privilege to meet great researchers and have enlightening discussions at numerous workshops and scientific visits. I thank Reinhard Werner and Ciara Morgan (Leibniz University, Hannover) and Robert Raussendorf (University of British Columbia, Vancouver) for inviting me to their groups to present my work.

A warm thank goes to my colleagues from the Max-Planck-Society's PhDnet (the codeword for our lovely doctoral society) with whom I had the pleasure to organize fantastic scientific events and activities. I am grateful to Ahmed Omran, Alexander Prehn, Matt Holbran and Axel Beyer, for the energy they put into organizing the 2014 MPQ Student Condensates, the 2014 Student Colloquia and the 2014 MPQ Summer Symposium. To them and to Rosa Glöckner, for our joint efforts that led to the 2014 MPQ PhD Satisfaction Survey. To the next generation of PhD reps, Julian Krauth, Dominik Ehberger, Vanessa Paulisch, Matthias Körber and Johannes Zeiher, who continued these efforts and organized many get-together with other MPI PhD researchers in Munich. To Eva-Regkina Symeonidou, Julia Hutenburg, Bjørt Kragesteen and Johanna Schulz for pushing forward the PhDnet Equal Opportunity group. To the directors of the MPQ (in particular, to Ignacio Cirac) for supporting the scientific activities of the doctoral researchers, which made life at the Max Planck Society a unique interdisciplinary experience.

I thank Abel Molina and Robert Raussendorf for piquing my research interest towards quantum computation early on when I was just a lil undergrad student. A special thank goes to Robert for welcoming me at his research group in 2008 during my undergrad exchange at UBC. They two convinced me that now is an exciting time to work on quantum computation.

Working on my thesis at MPQ Theory was not only an exciting time, but also a life-changing and enriching experience. Like the hobbit character of every epic, I could have not committed the extraordinary deed of completing this manuscript without overcoming some struggles and challenges, which made me grow as a scientist and as a person. As my main MPQ collaborators and quantum information colleagues departed from the group in 2014, I rose up to become an independent researcher. When the last stages of my thesis writing were slowed down by an episode of poor health, I endured it and taught myself to be patient. My deepest gratitude goes to Esther Román García, my Munich friends, my family, Kevin, Robert and Albóndiga, who accompanied me during these times; and to Maarten and Geza, who were always available remotely. A special thank goes to Felix Ehrentraut for being a long-time supporter, flatmate and best friend during my Munich years.

I am grateful to my university and high school teachers, who took an important part in my education and motivated me to pursue a career in quantum physics. In particular, I thank Pastora Vega Cruz, for being both a great teacher and caring family member.

I thank my parents, Juan José and Amparo, and my brother Andrés, who have always loved me and supported me unconditionally. I thank Esther for being my companion of daily adventures, and Albóndiga, for being our squire.

Funding from the Max Planck Institute of Quantum Optics, QCCC, SIQS, ALG-I and AQuS is gratefully acknowledged.

Contents

Chapter 0

Introduction

This chapter summarizes the technical contributions of this thesis, the methodology we use and points out key connections to prior work. Our aim is to present the main ideas and methods to a non-expert readership and locate them in their historical background. Some technical material (including some results, techniques and connections) are not included in our exposition and delegated to the technical chapters of the thesis. We refer the reader to chapters 1-6 for full statements of results and extended discussions of their technical significance.

0.1 Motivation

Quantum algorithms: the quest and the challenges

What are the potentials and limitations of quantum computers? Arguably, the most attractive feature of quantum computers is their ability to efficiently solve problems with no known classical solution, as demonstrated by Shor's 1994 groundbreaking discovery of an efficient quantum algorithm for factoring numbers [4]. To date, although a more than significant number of quantum algorithms has been discovered[1] [6–12, 5], there is still great demand for finding new ones and applications of them [11, 12]. Consequently, one of the greatest challenges of the field of quantum computing as per today is to understand for which precise problems quantum algorithms can be *exponentially* (or *super-polynomially*) faster than their classical counterparts.

One lesson gleaned from more than 20 years of quantum algorithm research, is that quantum computers can exponentially outperform classical ones at solving certain "structured" problems [13, 14]. Yet, our understanding of what these "structures" are remains limited and, even today, the search for quantum algorithms with exponential benefits remains more of an art than a science.

But what makes quantum algorithms with exponential advantages so hard to find? Although it is nearly impossible to give a mathematical answer to this question, a number of potential reasons have been pointed out in the literature. On the one hand, from the computer science perspective, a 2004 list[2] due to Shor [15], highlighted several major obstacles:

(i) the lack of an analogue *classical* theory for deciding which problems can be solved efficiently on a *classical* computer;

(ii) the modest number of quantum algorithmic techniques discovered so far;

[1] At the time of writing the "Quantum Algorithm Zoo" website [5] (one the best known online resources of this field) cites 262 papers on quantum algorithms.

[2] The list (i-iv) was made in 2004 [15], ten years after the factoring algorithm, but it is remarkably up to date.

(iii) the constrains to find interesting candidate problems to tackle with quantum computers, which should, ideally, "neither be in **P** nor **NP**-hard[3]" while most problems of interest in computer science are in either one of these two classes; and also because

(iv) candidate problems must remain unsolved after 60 years of classical algorithm research.

One the other hand, from the physical side, one needs to add the inherent difficulty of comprehending the *emergent complexity* of quantum systems, which poses a barrier not only to understand quantum speed-ups, but, more generally, quantum many-body phenomena. The same complexity that prevents us from simulating complex quantum dynamics on classical computers [17–19] is a double edge possibility sword, which makes possible (in principle) the existence of a quantum speed-up, but does not easily let us unravel the physical mechanisms that sustain them. In fact, the obstacle of complexity might be found (again and again) in other formidable unsolved problems in many-body physics, such as, e.g., deciding whether topological order is stable at non-zero temperature [20]; whether realizable self-correcting quantum memories [21, 22] exist; in identifying the physical ingredients that sustain high T_c superconductivity [23]; and last but not least, in the principles that guide the complex quantum many-body quantum dynamical evolution of a quantum computer.

0.2 This thesis in a nutshell

A Holy Grail of quantum computation would be to have a *theory of quantum speed-ups*[4] that would tell from basic principles which problems can be efficiently solved with a quantum computer. Ideally, such a theory would delineate the physical algorithmic mechanisms behind quantum speed-ups and be helpful in the design of new quantum algorithms. Although it is not a priori not clear whether one can even hope for such a theory within the state of the art of quantum physics and complexity theory (because of the obstacles surveyed above), in this thesis we make progress towards this ambitious goal by developing a theory for a *subclass* of quantum computational speed-ups.

In the rest of this preliminary chapter we outline how the above program will be implemented along the thesis. We begin with a discussion of why classical simulation methods and restricted gate models are central to our program, motivating the study of our first models of abelian-group normalizer circuits for gaining insight into quantum Fourier transforms (section 0.2.1). After looking at limitations of our first models, we discuss more powerful ones based on the notion of black-box groups, which we prove are useful to describe quantum algorithms and identifying no-go scenarios for finding them (section 0.2.2). Lastly, we describe our last normalizer-circuit formalism based on abelian hyper-structures, which we exploit to devise new quantum algorithms and infer insights into the working mechanisms of existing ones (section 0.2.3).

In parallel, we explain how algebraic generalizations of the stabilizer formalism [1] can be constructed and applied to address the main questions of this thesis.

In section 0.3 we discuss a few connections to previous work.

In section 0.4 we summarize the structure of the remaining chapters of this thesis.

[3]Strong evidence suggests that quantum computers cannot efficiently solve NP-complete problems [15, 16]. It is standard in quantum computing nowadays to assume that to be the case; and that $\mathbf{P} \neq \mathbf{NP}$, $\mathbf{BPP} \neq \mathbf{BQP}$. We take all these assumptions in this thesis. We remind the reader that **P** and **NP** are the classes of problems that can be *solved* and *verified* (respectively) in polynomial time on a deterministic classical computer, while **BPP** and **BQP** consist of problems that can be solved in polynomial time in probabilistic classical computers and quantum ones.

[4]Throughout this thesis, "quantum speed-up" will be synonymous of "superpolynomial quantum speed-up". We do not investigate quantum algorithms that yield polynomial advantages over classical computers.

0.2.1 Classical simulations, normalizer circuits and quantum Fourier transforms

A fruitful approach to understand the emergence and the structure of exponential quantum speed-ups is to study *restricted* models of quantum computation. Ideally, the latter should exhibit interesting quantum features and, at the same time, have less power than universal quantum computers (up to reasonable computational complexity assumptions). To date, several models studied in the literature seem to have these desirable properties, including Clifford circuits [1, 2, 24, 3], nearest-neighbor matchgates [25–28], Gaussian operations [29–32], the one-clean qubit (DQC1) model [33], and commuting circuits [34–37] (a more complete list is given in section 0.3).

The first result concerning restricted gate models in the history of quantum computation is the celebrated Gottesman-Knill theorem, which states that any quantum circuit built out of Clifford gates (Hadamards, CNOTs, $\pi/2$-phase gates) and Pauli measurements can be *efficiently* simulated on a classical computer [1, 2, 13]; thus, a quantum computer that works exclusively with these operations cannot achieve *exponential quantum speed-ups*.

The Gottesman-Knill theorem illustrates how subtle the frontier between classical and quantum computational power can be. For example, even though Clifford circuits can be simulated efficiently classically, replacing the $\pi/2$-phase gates by a $\pi/4$-phase gate immediately yields a quantum *universal* gate set [38, 39]. Another interesting feature is that, even though the computing power of Clifford circuits is not stronger than classical computation, their behavior is genuinely quantum: they can be used, for instance, to prepare highly entangled states (such as cluster states [40–42]), or to perform quantum teleportation [2]. Yet, in spite of the high degrees of entanglement that may be involved, the evolution of a physical system under Clifford operations can be tracked efficiently using a Heisenberg picture: the *stabilizer formalism*, backbone tool and basis of modern quantum error correction [22].

0.2.1.1 Normalizer circuits over abelian groups (setting in chapters 1-4)

The fact that the Gottesman-Knill theorem yields a powerful tool to identify non-trivial families of quantum circuits that cannot lead to a quantum speed-up, motivates us to adopt it as the starting point of this thesis. Unfortunately, for our purposes, the theorem presents the major downside that it can only be applied to study Clifford gate circuits, which have no known applications in quantum algorithm design[5]. To overcome this limitation, we dedicate the first part of this thesis (**chapters 1-4**) to the study of *new* restricted models of quantum circuits that contain more types of quantum gates.

More precisely, in **chapters 1-4** we introduce our first models of *normalizer circuits*, which have the most interesting feature of containing *quantum Fourier transforms* (QFT[6]), quantum gates that are essential in Shor's factoring algorithm [4] and are sometimes pointed out to be root of its exponential quantum speed-up. Specifically, we define a *normalizer circuit over an abelian group* G to be a quantum circuit consisting of three types of gates:

— Quantum Fourier transforms over G;
— Gates which compute automorphisms of G;
— Gates which compute quadratic functions on G.

We introduce the above normalizer circuit models in full detail and give examples in chapter 1, and in chapter 2 we develop classical group theoretic and algorithmic tools to investigate

[5] Note that Clifford circuits do provide a good setting to study which *quantum states* are universal resources in quantum computation via state injection [43–46, 45, 47, 48]; yet, this thesis is not concerned with universality but quantum speed-ups.

[6] Throughout the thesis, the acronym "QFT" will always stand for "quantum Fourier transform" and not for "quantum field theory".

them. In chapters 3-4 we present classical simulation results for normalizer circuits, which we summarize next.

0.2.1.2 Chapter 3: finite abelian group G

In chapter 3 we fix G to be a finite abelian group. When $G = \mathbb{Z}_2^n$ (the group of n-bit strings with addition modulo 2), normalizer circuits coincide precisely with the standard Clifford circuits. However, more exotic families of circuits can be obtained by simply modifying the parameter G. But there is more: for $G = \mathbb{Z}_{2^n}$, the associated normalizer circuit contain precisely the QFTs which are used in Shor's discrete-logarithm and factoring algorithms [4]; for other choices of G, normalizer circuits contain highly entangling gates and QFTs associated to arbitrary abelian groups, which are central subroutines in Kitaev's ubiquitous quantum phase estimation algorithm [49] and in quantum algorithms for solving so-called abelian **Hidden Subgroup Problems** (HSPs) [4, 50–53, 49, 54–58]: the latter comprise not only Shor's, but also Deutsch's [50], Simon's [51] quantum algorithms; furthermore, all famous quantum algorithms for breaking widely used public-key cryptosystems (namely, RSA [59], Diffie-Hellman's [60] and elliptic curve cryptopgraphy [61, 62]) belong to this class. Our motivation to investigate this abelian-group normalizer circuit model in **chapter 3** (see also **chapter 1**) is to gain insight into the question of *"When does the QFT serve as a resource for quantum computation?"* and, specifically, of *when does it lead exponential quantum speed-ups?*.

This chapter is based on [63] (joint work with Maarten Van den Nest).

Main results and techniques. Our first main result in **chapter 3** (cf. **theorem 3.7**) is a generalized Gottesman-Knill theorem, which states that normalizer circuits over a group G can be efficiently simulated in a classical computer if G is given to us in a canonically decomposed form. Specifically, when G is given as a product of cyclic group factors

$$G = \mathbb{Z}_{D_1} \times \cdots \times \mathbb{Z}_{D_m}$$

—-in which case normalizer gates over G act on a Hilbert space

$$\mathcal{H}_G = \mathbb{C}^{D_1} \otimes \cdots \otimes \mathbb{C}^{D_m}, \tag{1}$$

i.e. $\mathcal{H}_G$ is a collection of m finite systems of arbitrarily large dimensions $D_1, D_2, \ldots, D_m$—our result says that *any* quantum circuit built of normalizer gates over G can be classically simulated in time at most *polynomial* in the number of QFTs and gates present of the circuit, the number of factors m, and the logarithms $\log D_i$ of all local dimensions (hence, the simulation is efficient in the dimension of $\mathcal{H}_G$ even if $D_i = 2^n$ is exponentially large). The significance of this result is that it identifies many non-trivial families of quantum computations that *fail* to harness the power of QFTs in order to achieve achieve exponential quantum speed-ups.

Our second main contribution in **chapter 3** (cf. **theorems 3.2, 3.5, 3.4**) is a generalized **stabilizer formalism over finite abelian groups**, i.e., for systems of the form (1). For a given finite abelian group G, our formalism lets us describe rich families of quantum states and codes within $\mathcal{H}_G$, which we name *stabilizer states/codes* over G, as joint eigenspaces of stabilizer groups of *generalized Pauli operators* over G: for groups of the form $\mathbb{Z}_2^n$, we recover the standard definitions of qubit Pauli operator and qubit stabilizer state/code, hence, our formalism extends Gottesman's PSF. We show that our formalism can be used to efficiently track the evolution of abelian-group stabilizer states under arbitrarily-long normalizer circuits in a Heisenberg picture, by tracking a small-number of stabilizer group generators. Furthermore, we develop explicit analytic *normal forms* for the evolved states in terms of subgroup cosets and quadratic functions. This techniques are key to prove our main simulation result.

The main technical effort in chapter 3 goes into developing our generalized stabilizer formalism. Prior to our work, classical techniques to simulate Clifford circuits had been developed for *qudit* systems of constant dimension d (in our setting, this parameter grows unboundedly); most works further assumed d to be prime, in which case G is a vector space and exploited standard field-theoretic algorithms in the simulation (e.g. Gaussian elimination). For our simulations we develop different techniques that involve representation theory of abelian groups, computational group theory, and Smith normal forms.

0.2.1.3 Chapter 4: infinite abelian group G

In **chapter 4** we introduce new generalized families of normalizer circuits over *infinite* abelian groups that act on *infinite dimensional* systems. Specifically, we define normalizer circuits over groups of the form[7]

$$G = \mathbb{Z}^a \times \mathbb{T}^b \times \mathbb{Z}_{D_1} \times \cdots \times \mathbb{Z}_{D_c},$$

extending our prior setting by allowing new types of group factors, namely, integer lattices $\mathbb{Z}^a$ of arbitrary rank a, and hypertori $\mathbb{T}^b$ of arbitrary dimension b. The motivation for adding $\mathbb{Z}$ is that several number theoretical problems of interest in quantum computation are naturally connected to problems over the integers (e.g., factoring is related to hidden subgroup over $\mathbb{Z}$); we further add $\mathbb{T}$ because it is connected to $\mathbb{Z}$ via the Fourier transform over this group.

This chapter is based on [64] (joint work with Cedric Yen-Yu Lin and Maarten Van den Nest).

Main results and techniques. Our main contributions in this chapter are a generalized *stabilizer formalism* and a *Gottesman-Knill theorem* for infinite dimensional systems, which states that all normalizer circuits over infinite groups as above can be *simulated* classically in polynomial time. These results extend those of chapter 3 to infinite dimensions.

The simulation techniques in chapter 4 differ strongly from previous work on stabilizer simulations because they can handle continuous infinite groups G as well as and *continuous-infinite stabilizer groups*. The groups under consideration are notoriously difficult to manipulate because they are neither finite, nor finitely generated, nor countable; they are not vector spaces and do not have bases; and they are not compact. Remarkably, in this setting, generalized stabilizer groups *can no longer* be described with finite sets of generators. Instead, we develop a novel machinery for handling infinite stabilizer groups based on *linear map* encodings, *normal forms* for quadratic functions and group morphisms. Combining this technology with novel *ε-net techniques*, we devise the most powerful classical algorithm to date to sample the support of infinite dimensional stabilizer states. This leads to our simulation result a la Gottesman-Knill for infinite dimensional systems.

0.2.1.4 Discussion (chapters 3-4)

We end this subsection discussing potential applications of the techniques developed in chapters 3-4 outside the scope of this thesis.

First, we recall that Gottesman's original Paul Stabilizer Formalism and the Gottesman-Knill theorem for qubits and qudits has been used in a variety of settings. The PSF itself is a central tool in, e.g., measurement-based quantum computation (with qubits [42] and qudits [65, 66]), quantum error-correction (qubits [1, 2, 43], qudits [3, 67–71]), secret-sharing (qubits [72], qudits [73, 74]); in the study of topologically-ordered systems (qubits [75], qudits [76–78]) and universal resources for quantum computation via state injection (rebits [47], qubits [48],

[7] Our construction can be applied to define normalizer circuit models over arbitrary abelian groups, but we focus on these types for the reasons given in the main text.

qudits [44]), among others. The standard Gottesman-Knill is often applied in fault-tolerant quantum computation in order to simulate Pauli/Clifford noise channels [79, 80] and delay recovery operations [81], which indirectly reduces noise threshold requirements [82–86].

It is plausible that the techniques developed in chapters 3–4 could find applications in the fields mentioned above. An attractive feature of our work is that it leads to the first known stabilizer formalism and normalizer gate models for *hybrid* systems of asymmetric qudits $\mathcal{H}_{D_1} \otimes \cdots \otimes \mathcal{H}_{D_a}$, harmonic oscillators $\mathcal{H}_{\text{osc}}^{\otimes b}$ and quantum rotors $\mathcal{H}_{\text{rot}}^{\otimes c+d}$, which have Hilbert spaces labeled by groups of the form $\mathbb{Z}_{D_1} \times \ldots \times \mathbb{Z}_{D_a}$, $\mathbb{R}^b$ and $\mathbb{Z}^c \times \mathbb{T}^d$, respectively: qudits and harmonic oscillators have well-known applications in QIP over discrete and continuous-variables; the latter, quantum rotors, describe other QIP platforms like, e.g., *Josephson tunneling junctions*, which are the basic constituent of all superconducting-qubit designs for building quantum computers, and electromagnetic modes carrying angular momentum [87, 88]. Our normalizer gate models for these systems extend the well-known families of qudit Clifford gates and continuous-variable Gaussian unitaries [29–32] and define "superconducting" analogues the latter, which might find uses in QEC and QIP[8]. Our stabilizer formalism yields a framework for defining and analyzing stabilizer codes and states for all these platforms: in finite dimensions, our techniques are novel in that they can handle qudit dimensions that differ, or are not prime numbers, or can be large; for infinite dimensional systems labeled by $\mathbb{Z}$ and $\mathbb{T}$ groups, our methods could be applied to simulate charge/phase/flux noise or delay recovery operations in fault-tolerant quantum computing schemes based on "rotating-variable" superconducting codes (such as, e.g. Kitaev's 0-π codes [89, 90]).

0.2.2 The computational power of normalizer circuits over black-box groups

The models of normalizer circuit over abelian groups studied in chapters 3-4 let us identify scenarios where QFTs fail to achieve achieve exponential quantum speed-ups. In chapter 5 of this thesis we introduce models of quantum computation based on extended normalizer circuits that have non-trivial quantum power. We further use these models and classical simulation techniques in order to characterize the computational power of a large family of quantum algorithms.

Specifically, in chapter 5, we consider *black box normalizer circuits* that are associated to finite abelian groups that are *black box groups* $\mathbf{B}$ (as introduced by Babai and Szemerédi in [91]) and allow G to be of the form.

$$G = \left(\mathbb{Z}^a \times \mathbb{T}^b \times \mathbb{Z}_{D_1} \times \cdots \times \mathbb{Z}_{D_c}\right)_{\text{previous setting}} \times (\mathbf{B})_{\text{new setting}} . \tag{2}$$

Note that the difference between this and earlier settings is that the group $\mathbf{B}$ is no longer assumed to be given in a decomposed form $\mathbb{Z}_{D1} \times \cdots \times \mathbb{Z}_{D_d}$, which makes a distinction in terms of computational complexity: though every finite abelian group is isomorphic to some decomposed group $\mathbb{Z}_{D_1} \times \cdots \times \mathbb{Z}_{D_d}$, computing such a decomposition is at least as hard as factoring, which is polynomial-time reducible to the problem of decomposing *multiplicative* groups $\mathbb{Z}_N^\times$ of integers modulo N [92]. In chapter 5, abelian groups for which such a direct product product decomposition is a priori unknown are modeled as *black-box group* for which it is only known how elements can be *efficiently* represented (as bit-strings) and multiplied/added.

This chapter is based on [93] (joint work with Cedric Yen-Yu Lin and Maarten Van den Nest).

[8] Normalizer gates over groups of the form $\mathbb{Z}_d^a$ and $\mathbb{R}^b$, respectively, yield the standard qudit Clifford gates and Gaussian unitaries (see chapter 1, section 4.1 and appendix E). Within our formalism, we obtain more general models gates by either looking at systems whose Hilbert spaces $\mathcal{H}_G$ are labeled by different groups and/or by combining registers $\mathcal{H}_{G_1} \otimes \mathcal{H}_{G_2}$ into a larger "hybrid" system.

Main results and techniques. In contrast to our classical simulation result for decomposed abelian groups, we find that allowing black-box groups in our setting dramatically changes the computational power of normalizer circuits. In particular, we show that many of the most famous quantum algorithms are particular instances of normalizer circuits over black-box groups (2), thereby proving that normalizer circuits over black box groups can offer *exponential quantum speed-ups* and break widely used public-key cryptographic systems. Namely, in our generalized formalism, the following algorithms are examples of black-box normalizer circuits over a group G of form (2)—or have equivalent normalizer circuit versions:

- Shor's algorithm for computing discrete logarithms [4]: $G = \mathbb{Z}_{p-1} \times \mathbb{Z}_p^{\times}$;
- Shor's factoring algorithm [4]: $G = \mathbb{Z} \times \mathbb{Z}_N^{\times}$;
- The generalized Shor's algorithm for finding discrete-logarithms over an elliptic curve E [94–96]: $G = \mathbb{Z}^2 \times E$;
- Simon's algorithm [51] and other oracular abelian hidden subgroup problem algorithms [49, 52], are normalizer circuits over groups of the form $G \times \mathbf{B}$, where G and $\mathbf{B}$ are a group and a black-box group determined by the input of the HSP;
- Cheung-Mosca's algorithm for decomposing black-box finite abelian groups [97, 98] is a combination of several types of black-box normalizer circuits.

The above results establish a precise connection between Clifford circuits and Shor-like quantum algorithms and, furthermore, imply that black-box normalizer circuits are powerful enough to break important cryptosystems such as RSA [59], Diffie-Hellman's [60] and elliptic curve cryptopgraphy [61, 62]. In the rest of chapter 5, we further exploit the *abelian group stabilizer formalism* developed in earlier chapters to tightly *characterize* the computational power of normalizer circuits, as outlined next.

- We show that the problem of decomposing black-box groups is *complete* for the class of computational problems solvable by black-box normalizer circuits: once an oracle to solve that problem is provided, we show (**theorem 5.6**) that our simulation techniques from earlier chapters *render black-box normalizer circuits efficiently classically simulable*. For this result, we need to introduce a generalized version of the group decomposition problem considered by Cheung-Mosca [97, 98], for which we give an efficient quantum algorithm based on normalizer circuits; extending, along the way, the result of [97, 98]. These results demonstrate that the computational power of normalizer circuits is encapsulated precise in the classical hardness of decomposing black-box groups.
- We give a *no-go theorem* (**theorem 5.7**) for finding new quantum algorithms within the class of black-box normalizer circuits considered. This result has immediate implications for quantum algorithm design, for it imposes provable restrictions to quantum computing theorists for finding new quantum algorithms with the basic set of *Fourier sampling techniques over finite abelian groups*, which are covered by our normalizer circuit model: specifically, our result shows that any (potentially sophisticated) quantum algorithm based on such techniques can be emulated by smartly using our extended Cheung-Mosca quantum algorithm[9].
- Another consequence of theorem 5.7 is a *universality result for short normalizer circuits* that explains a curious quantum computing mystery: although many quantum algorithms

[9] Like all of our results, this no-go theorem is for quantum algorithms with superpolynomial speed-ups.

use quantum Fourier transforms, interestingly, many of them use only a small number of them and, in fact, two are often enough [4, 55–58, 50–53, 49, 54]. A corollary of theorem 5.7 is that all quantum algorithms based on normalizer circuits can be simulated by sequences of quantum circuits that contain two QFTs and use intermediate classical processing. Hence, normalizer circuits cannot gain any significant superpolynomial advantage from using more than two Fourier transforms.

0.2.2.1 Discussion (chapter 5)

The no-go theorem presented in chapter 5 is not only a useful tool to identify approaches for finding quantum algorithm that do not work, for, by carefully analyzing the conditions under which the theorem holds, one may guess promising avenues for finding new ones (cf. section 5.1.4). In fact, this and other insights from chapter 5 helped us to find the new quantum algorithms that we present in our next chapter 6.

0.2.3 Abelian hypergroups and quantum computation

In chapters 3-5 we showed that the abelian group normalizer circuit framework and the abelian group stabilizer formalism are helpful tools to understand the exponential quantum speed-ups of Shor's algorithm and the quantum algorithms for solving abelian hidden subgroup problems (HSP). In the final chapter of this thesis (**chapter 6**) we attempt to extend this approach in order to gain insight into quantum algorithms for *nonabelian groups* hidden subgroup problems, which have been object of intense research work [99–119, 9] in the last decades of quantum computation. The motivation of the HSP research program followed the breakthrough discoveries that solving the HSP over symmetric and dihedral groups would lead to revolutionary efficient algorithms for Graph Isomorphism [120] and certain latticed-based problems [121]. Despite much effort, no efficient quantum algorithm for dihedral or symmetric HSP has yet been found.

Specifically, our initial goal in chapter 6 is to gain understanding into a seminal *efficient* quantum algorithm of Hallgren, Russell, and Ta-Shma [99] for finding hidden *normal* subgroups, which, remarkably, works efficiently for *any* nonabelian group. Surprisingly, despite the fact that the HRT algorithm is also the basis of several sophisticated algorithms for nonabelian HSPs [122, 123], its efficiency remains poorly explained. Given the success of the normalizer circuit framework (chapters 3-5) at understanding abelian HSP quantum algorithms, we address the question of whether a more sophisticated stabilizer formalism can shed light into this question and lead to new applications of quantum computation. Our main results, summarized below, answer this question in the affirmative.

This chapter is based on [124] (joint work with Kevin C. Zatloukal).

Main results and techniques.

- Our first result in chapter 6 is a *connection* between the hidden normal subgroup problem (HNSP) and *abelian hypergroups*, which are algebraic objects that model collisions of physical particles and anti-particles and generalize abelian groups. Our result shows that in many natural cases the HNSP can be reduced to the commutative problem of finding subhypergroups of abelian hypergroups.

The above connection and the fact that abelian groups are particular instances of abelian hypergroups, motivates us to explore whether abelian hypergroups and the normalizer circuit framework can be combined to answer our initial main question. Our findings are presented next.

- **A hypergroup stabilizer formalism.** We present a generalized stabilizer formalism based on commuting *hypergroups* of generalized Pauli operators (whose multiplication mimics particle annihilation processes) as well as extended families of (Clifford-like) normalizer circuits over abelian hypergroups. Using our formalism, we devise classical algorithms for simulating hypergroup normalizer circuits and develop analytic normal forms for describing quantum many-body quantum states (namely, hypergroup coset states) and analyzing the convergence of quantum algorithms.

- **New quantum algorithms.** We devise the first provably efficient quantum algorithms for finding hidden subhypergroups of abelian hypergroups and, exploiting our hypergroup-HNSP connection, also new quantum algorithms for the latter problem. Our algorithms are based on hypergroup normalizer gates, which let us apply our hypergroup stabilizer methods in our analysis. We show that our algorithms provably work for hypergroups that arise from nilpotent, dihedral and symmetric groups, which are the most interesting groups from the nonabelian HSP perspective. In contrast, no efficient quantum algorithm for nilpotent, dihedral or symmetric HSPs is known.

0.2.3.1 Discussion (chapter 6)

Our HNSP quantum algorithms are different from the one of Hallgren et al. in that they exploit commutative structures that are related to those present in Shor's algorithm via a stabilizer formalism: this provides an important new insight into why the HNSP is much easier than the general nonabelian HSP. Furthermore, our quantum algorithm for finding abelian subhypergroups provide strong evidence that the abelian Hidden Subhypergroup Problem [125, 126] is a much easier problem for quantum computers than the nonabelian HSP (perhaps even more natural one because of its elegant connection to a stabilizer picture).

A main building block of the quantum algorithms in this chapter is a novel *adaptive/recursive quantum Fourier sampling* technique of independent interest. This technique overcomes the limitations of an earlier abelian HSHP quantum algorithm [125, 126] based on Shor-Kitaev's quantum phase estimation [49], which we prove to be inefficient on easy instances.

Beyond the scope of this thesis, abelian hypergroups have important applications in, e.g., convex optimization [127, 128], classical classical error correction [129] and conformal field theory [130]. In topological quantum computation [75], fusion-rule hypergroups are indispensable in the study of nonabelian anyons and topological order [131].

Our stabilizer formalism over abelian hypergroups provide the first generalization and alternative to Gottesman's Pauli Stabilizer Formalism where stabilizer operators are not necessarily *unitary*, nor *monomial*, nor *sparse* matrices. These techniques are likely to find applications in quantum error correction and classical simulations, e.g., for probing the classical simulability of protected gates over topological quantum field theories [132].

0.2.4 Summary of complexity theoretic results

In order to summarize our complexity theoretic results, we provide a Venn diagram (figure 1) that represents the known complexity classes associated to the different families of normalizer circuits investigated in this thesis and their relationships.

0.3 Relationship to previous works

We discuss some (non-technical) connections between our thesis and other works on restricted models of quantum computations and/or classical simulations. We refer to chapters 3.1.3, 4.1.2,

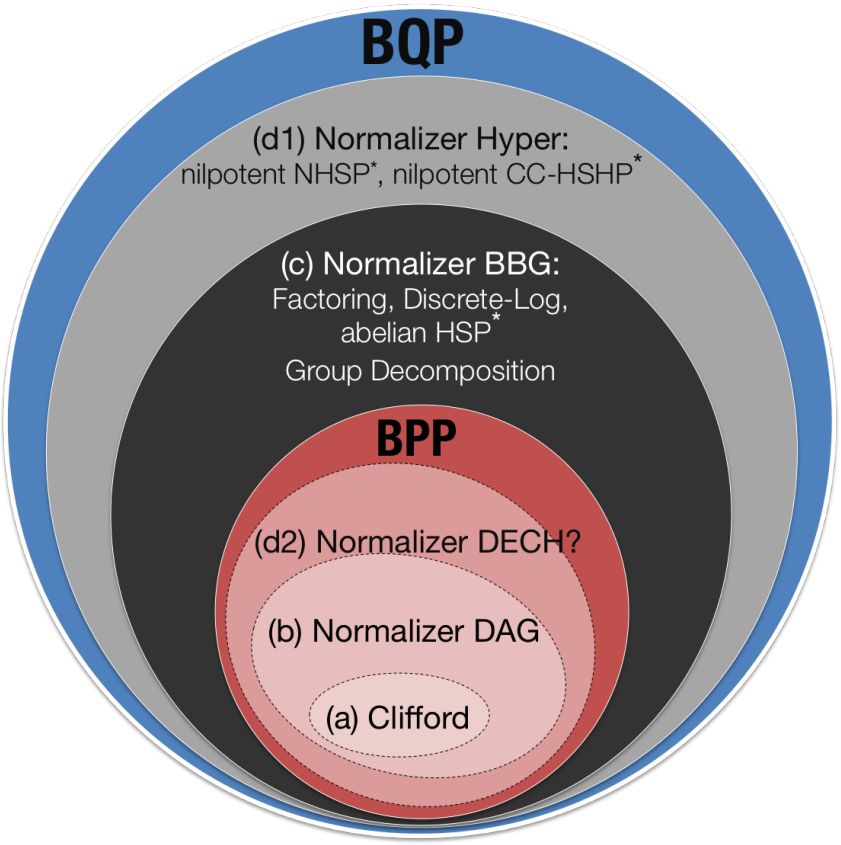

Figure 1: Circles in this Venn diagram represent complexity classes of computational problems: a class of problems X is plotted inside Y if Y is known to include X. BPP and BQP are the classes of problems that can be solved by quantum and (probabilistic) classical computers in polynomial time [13]. The quantum circuits corresponding to classes outside BPP have access to classical computers that carry out classical post-processing tasks. Class (a) Clifford, which contains the problems solvable via qubit Clifford circuits, forms a subset of BPP [2] (this containment is believed to be strict [133]). Class (b) Normalizer DAG represents the problems solvable via normalizer circuits over arbitrary decomposed abelian groups (chapters 3-4). Class (c) Normalizer BBG represents the problems solvable via black-box-group normalizer circuits, e.g., factoring and discrete-log (chapter 5). Class (d1) Normalizer Hyper—resp. class (d2) Normalizer DECH—contain problems solvable via normalizer circuits over efficiently—resp doubly-efficiently—computable hypergroups (chapter 6). The containment of (a) and (b) inside (d2) (marked with "?") is conjectured in chapter 6 and only proven for CSS-preserving operations. Problems marked by a superscript "*" are oracular and proven to be contained in their respective classes if certain subroutines to carry out some classical calculations are provided externally. We refer the reader to chapters 1-6 for details on the normalizer circuit models associated to each class.

5.1.3 and 6.1.3 for connections related to the main technical contributions of the thesis.

The normalizer circuit formalism presented in this thesis was developed by a sequence of various works. Normalizer circuits over *finite abelian groups* with terminal measurements were introduced by Van den Nest [134] and Bermejo-Vega-VdN [63], over infinite and black-box groups by BV-Lin-VdN [64, 93], and over abelian hypergroups by BV-Zatloukal [124].

Clifford circuits over qubits and qudits, which can be understood as normalizer circuits over groups of the form $\mathbb{Z}_2^m$ and $\mathbb{Z}_d^m$ (section 1.3.1), and Gaussian unitaries, which can be approximated by normalizer circuits over $\mathbb{R}^m$ to any degree of accuracy (chapter 4.1.2, appendix E), have been extensively investigated in the literature: see, e.g., [1–3, 135, 133, 136–139] and [29–32, 140–144] for Clifford and Gaussian references, respectively.

Certain generalizations of Clifford circuits that are not normalizer circuits have also been studied: [133, 43, 28, 138, 139] consider Clifford circuits supplemented with some non-Clifford ingredients; a different form of Clifford circuits based on projective normalizers of unitary groups was investigated in [145].

Aside from generalizations of Clifford circuits, many other classes of restricted quantum circuits have been studied in the literature (very often within the context of classical simulations). Some examples (by no means meant to be an exhaustive list) are nearest-neighbor matchgate circuits [25–28, 146–150], the one-clean qubit model [151–158], circuit models based on Gaussian and linear-optical operations [30–32, 144, 44, 159, 160], commuting circuits [34–37], low-entangling[10] circuits [162, 163] , low-depth circuits [164, 165], tree-like circuits [165–169], low-interference circuits [170, 171] and a few others [172, 173].

In this thesis, classical simulation techniques play an important role in the development of the complexity theoretic hardness results and quantum algorithms we present. In this way, our results relate to other projects where classical simulations methods helped to find new quantum algorithms [174, 37] and/or complexity theoretic hardness results [144, 36, 158].

Like Clifford circuits, the models we introduce are also unlikely to be universal: in chapters 3-5 we know they are not unless computational complexity classes that are believed to be distinct collapse; the universality of the model in chapter 6 was not fully investigated.

0.4 Reading guide

In chapter 1 we illustrate the many of the ideas developed in the thesis by introducing the simplest circuit families we investigate, namely, our models of normalizer circuits that arise from *abelian groups*—and from non-black-box ones. Therein, we give several examples of normalizer gates, and explain their connection with Clifford unitaries.

In chapter 2 we introduce classical group-theoretic and algorithmic techniques that will be essential in chapters 3-5, including a theory of matrix representations for group morphisms, normal forms for quadratic functions and algorithms for solving systems of linear equations over groups.

The remaining chapters contain the quantum contributions of the thesis. In chapters 3 and 4 we develop techniques for simulating normalizer circuits over decomposed abelian groups, and develop their associated stabilizer formalism. In chapter 5 we add black-box groups to these models, show how the resulting circuits can implement quantum algorithms and derive our first complexity-theoretic hardness results.

Finally, in chapter 5, we move above from the abelian-group setting allowing normalizer circuits to act on commutative hyper-structures. In this setting we investigate and devise

[10] 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 [161]).

new quantum algorithms for the normal HSP and the abelian HSHP, as well as a hypergroup stabilizer formalism.

Chapter 1

Normalizer circuits over abelian groups

Quantum Fourier transforms (QFT) lie among the most important quantum operations in quantum computation, being key components of many quintessential quantum algorithms [7] and often linked to the exponential speed-up of, e.g., Shor's factoring algorithm. In this chapter we introduce quantum circuit models that contain QFTs and resemble the circuits employed in Shor-like quantum algorithms. Specifically, we propose *normalizer circuits over abelian groups* [134, 63, 64] as high-dimensional generalizations of the well-known Clifford circuits [1–3] that contain group QFTs, automorphism gates and quadratic-phase gates.

Normalizer circuit models provide a framework that we exploit to develop the program of this thesis: in chapters 3-4 we show that the normalizer circuits in this chapter cannot provide quantum speed-ups despite the presence of QFTs in various settings; in chapters 5-6, we propose extended models of normalizer circuit model that lead to quantum algorithms. The purpose of this chapter is to introduce the simplest normalizer circuit models of the thesis (chapters 3-4) and convey their key quantum features before moving to more involved (albeit powerful) ones (chapters 5-6). To illustrate our definitions, we give several examples of normalizer gates and also present some other concepts that appear later in the thesis: namely, the notions of Clifford and Pauli operators.

Section 1.3 of this chapter is based on [63] (joint work with Maarten Van den Nest). Section 1.4 is based on [64] (joint work with Cedric Yen-Yu Lin and Maarten Van den Nest). Prior to us, normalizer circuits over finite abelian groups were considered in [134] by Van den Nest. Connections to prior work are surveyed at the end of section 1.1.

1.1 Introduction

Clifford gates are a winsome family of restricted quantum operations with a wide range o applications in quantum computation and information processing and, at the same time, a beautiful mathematical theory to describe them: i.e. the stabilizer formalism [1, 2]. By definition, an n-qubit *Clifford circuit* $\mathcal{C}$ is any unitary gate that leaves invariant the n-qubit Pauli group[1] under conjugation; equivalently, $\mathcal{C}$ is any circuit built of sequences of Hadamard gates, CNOTs, CZ gates and Phase gates $S = \mathrm{diag}(1, i)$ (acting on arbitrary qubits).

In this section we introduce *normalizer circuits* associated to an *abelian group* G, as group theoretic generalizations of the Clifford circuits containing abelian-group quantum Fourier transforms (QFTs), autormorphism and quadratic phase gates; the latter generalize the Hadamard, CNOT, CZ and S gates, respectively. Specifically, we will focus on groups of the form $G = \mathbb{Z}_{D_1} \times \ldots \times \mathbb{Z}_{D_a} \times \mathbb{Z}^b \times \mathbb{T}^c$, where $\mathbb{Z}_D = \{0, 1, \ldots, D-1\}$ is the additive group of integers modulo

[1] Ie. the group generated by the arbitrary n-fold tensor products of the Pauli matrices $\sigma_x, \sigma_y, \sigma_z$.

N, $\mathbb{Z}$ is the additive group of integers and $\mathbb{T}^c = \mathbb{T} \times \cdots \times \mathbb{T}$ is a b-dimensional hypertorus. Our motivation to consider these different types of group factors is twofold:

- On the one hand, our motivation to consider ***finite*** abelian groups of form $\mathbb{Z}_{D_1} \times \ldots \times \mathbb{Z}_{D_a}$ is that the 'standard" quantum Fourier transform $\mathcal{F}_{2^n}$ used by Shor in its factoring and discrete-log quantum algorithms can be understood as a QFT over $\mathbb{Z}_{2^n}$. The associated finite-group normalizer circuits will be investigated in **chapter 3**.

- On the other hand, our interest in the ***infinite*** integer group $\mathbb{Z}$ is that several number theoretical problems are naturally connected to problems over the integers, a crucial example being the factoring problem, which is reducible to a hidden subgroup problem over $\mathbb{Z}$ [55–58]. The motivation to consider hypertori $\mathbb{T}^m$ is that they are intrinsically connected to integer groups $\mathbb{Z}^m$ via a quantum Fourier transform. The associated normalizer circuits over the latter infinite groups will be investigated in **chapter 4**.

The families of normalizer circuits above are not the only ones we investigate along the thesis: in **chapters 5** and **6**, in order to develop some of our main quantum algorithm and complexity theoretic results, we will introduce more general (and powerful) models of normalizer gates that are related to ***black-box groups*** and ***abelian hypergroups***. Our later models, which are slightly more abstract, will be much easier to understand after going through the simpler examples in this chapter.

Also, in order to enrich the discussion in the introduction of the thesis (chapter 0), we consider in **appendix E** a model of normalizer circuits over ***real groups*** $\mathbb{R}^m$ and show that they realize the well-known families of (bosonic) *Gaussian unitaries*, the latter being central in *continuous*-variable quantum information processing [175, 29, 140, 31, 32, 141, 30, 142, 143]. Since Clifford gates are, in turn, a fundamental gate-set for QIP with *discrete*-variables, this side result motivates the search of potential applications of normalizer circuit models over more general commutative algebraic structures in quantum information processing. We leave this potential research avenue open to future investigations (cf. chapter 4 for extended discussion)

1.1.1 Chapter outline

We split the discussion of this section as follows. In section 1.2, we introduce normalizer gates over (fully) arbitrary abelian groups in a low level of detail. In section 1.3 we introduce our first quantum circuit model based on normalizer circuits over *finite* abelian groups $\mathbb{Z}_{D_1} \times \ldots \times \mathbb{Z}_{D_a}$. In section 1.4, we introduce the more involved infinite-dimensional normalizer circuit model over groups $\mathbb{Z}^a \times \mathbb{T}^b \times \mathbb{Z}_{D_1} \times \ldots \times \mathbb{Z}_{D_c}$.

Along the section we illustrate our definitions with several examples. Section 1.3.1 contains finite-dimensional ones, and explains the connection between normalizer circuits, the qubit and qudits Clifford gates used so-widely in quantum error correction [24, 3], and Shor's quantum Fourier transform [4]. Section 1.4.3 contains examples of infinite-dimensional normalizer gates.

1.2 Normalizer gates

In short, *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 (namely, a Haar measure) and a well-defined classical Fourier transform. Given these conditions, we define a *normalizer gate over* G to be any gate of the following three types:

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

(ii) **Group automorphism gates**. These implement group automorphisms $\alpha : G \to G$, at the quantum level $|g\rangle \to |\alpha(g)\rangle$, $g \in G$. When G is infinite, we require α to be continuous.

(iii) **Quadratic phase gates** are diagonal gates that multiply standard basis states with *quadratic* phases $|g\rangle \to \xi(g)|g\rangle$, where $|\xi(g)| = 1$. "Quadratic" means that $g \to \xi(g)$ is an "almost multiplicative" function with the property $\xi(g+h) = \xi(g)\xi(h)B(g,h)$, and $B(g,h)$ is a bi-character of G: i.e., a bi-multiplicative correcting term fulfilling

$$B(x+y,g) = B(x,g)B(y,g), \qquad B(g,x+y) = B(g,x)B(g,y), \qquad \text{for all } x,y,g \in G.$$

Again, when G is infinite, we require ξ, B to be continuous in all arguments.

Classical Fourier transforms. In the definition of QFT above (i), the *classical* Fourier transform over an abelian group G is defined canonically through the notion of *character functions* of G: a complex function χ_p on G is said to be a *character* if $\chi_p(x+y) = \chi_p(x)\chi_p(y)$ and $|\chi_p(x)| = 1$ holds for every $x, y \in G$; the set of all such functions is denoted $\widehat{G}$. Then, for all abelian groups G with reasonable topologies[2], the Fourier transform

$$\psi(x) \qquad \xrightarrow{\text{QFT over } G} \qquad \hat{\psi}(p) := \sum_{\chi_p \in \widehat{G}} \chi_p(g)\psi(g)$$

defines a unitary transformation (up to normalization), which we will regard as a valid quantum circuit element.

The properties of character functions will be reviewed in chapter 2. In the next sections, we give examples of QFTs for various groups.

The groups. Although normalizer circuits as above can be associated to almost[3] any abelian group, in this thesis we focus[4] on abelian groups of the form

$$G = \mathbb{Z}_{D_1} \times \ldots \times \mathbb{Z}_{D_a} \times \mathbb{Z}^b \times \mathbb{T}^c \tag{1.1}$$

where $\mathbb{Z}_D = \{0, 1, \ldots, D-1\}$ is the additive group of integers modulo N, $\mathbb{Z}$ is the additive group of integers and $\mathbb{T}^c = \mathbb{T} \times \cdots \times \mathbb{T}$ is a b-dimensional hypertorus. These particular groups are chosen for their connection with hidden subgroup problems (chapter 3-5). Throughout the paper, the elements of $\mathbb{T}$ (the circle group[5]) are represented as real numbers in $[0, 1)$ modulo 1 (these are angles measured in units of 2π).

It is important to note that the finite abelian groups $\mathbb{Z}_{D_1} \times \ldots \times \mathbb{Z}_{D_a}$ in (1.1) are fully arbitrary because of a well-known group-theoretic result.

[2] This holds for locally compact Hausdorff ones (i.e. the vast majority of groups used in quantum mechanics).

[3] Our circuit model is well-defined for any locally compact abelian group (cf. discussion in chapter 4, appendix E), though sometimes a renormalization factor is required for the map $|g\rangle \to |\alpha(g)\rangle$ to be unitary. Within this thesis, this re-scaling only plays a role in appendix E (cf. discussion).

[4] In appendix E we briefly study normalizer circuits over $\mathbb{R}^m$ and show that they coincide with the well-known family of (bosonic) continuous-variable Gaussian unitaries, widely used in the CV-QIP literature [29, 30, 142, 143].

[5] In our notation, the circle group $\mathbb{T}$ is a one-dimensional torus and $\mathbb{T}^2$ is the usual two-dimensional one.

Theorem 1.1 (**Fundamental Theorem of Finite Abelian Groups** [176]). Any finite abelian group G has a decomposition into a direct product of cyclic groups, i.e.

$$G = \mathbb{Z}_{D_1} \times \ldots \times \mathbb{Z}_{D_k} \tag{1.2}$$

for some positive integers $D_1, \cdots, D_k$. Here, the elements of (1.2) are m-tuples of the form $g = (g(1), \ldots, g(k))$ with $g(i) \in \mathbb{Z}_{D_i}$ and addition of two group elements is component-wise modulo D_i. The order (or cardinality) of G is denoted by $|G|$, and fulfills $|G| = D_1 D_2 \cdots D_k$.

On complexity. Although theorem 1.1 states that any finite abelian group can be expressed as a product of the type (1.2) via isomorphism, computing this decomposition is regarded as a difficult computational problem (at least as hard as factoring integers[6]). In this section and in chapters 3-4 a product decomposition (1.2) of G will always be explicitly given; however, this assumption will be removed in chapter 5.

The Hilbert space of a group. The Hilbert space $\mathcal{H}_G$ associated to any group of the form (1.1), inherits a natural tensor-product structure from the factors of G

$$\mathcal{H}_G = \mathcal{H}_{\mathbb{Z}_{D_1}} \otimes \cdots \otimes \mathcal{H}_{\mathbb{Z}_{D_a}} \otimes \mathcal{H}_{\mathbb{Z}}^{\otimes b} \otimes \mathcal{H}_{\mathbb{T}}^{\otimes c} \tag{1.3}$$

A normalizer circuit over G performs a quantum computation on the $m := a+b+c$ computational registers of $\mathcal{H}_G$. The former a registers form a finite-dimensional subspace of D_i-level systems $\mathcal{H}_{\mathbb{Z}_{D_1}} \otimes \cdots \otimes \mathcal{H}_{\mathbb{Z}_{D_m}}$, where $\mathcal{H}_{\mathbb{Z}_{D_i}} \cong \mathbb{C}^{D_i}$. The latter, form a subspace of $(b+c)$ infinite-dimensional *quantum rotors*[7], which may be regarded as quantum particles that can move in a circular orbit around a fixed axis, having angular position and integral 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 [178]). A normalizer computation over G will act on specific *designated basis* of $\mathcal{H}_G$: the first of these bases is the standard group-element basis $\mathcal{B}_G$ of product states labeled by elements of G

$$|g\rangle = |g(1)\rangle \otimes \cdots \otimes |g(m)\rangle \quad \text{for all} \quad g \in G. \tag{1.4}$$

The remaining bases can be obtained from (1.4) by performing single-register quantum Fourier transforms (QFT), which we introduce below.

1.3 Normalizer circuits over finite G

We introduce now our models of normalizer circuit models over finite abelian groups letting $G = \mathbb{Z}_{D_1} \times \ldots \times \mathbb{Z}_{D_m}$. These models will be investigated later in chapter 3. The latter act on the Hilbert spaces of form $\mathcal{H}_G = \mathcal{H}_{\mathbb{Z}_{D_1}} \otimes \cdots \otimes \mathcal{H}_{\mathbb{Z}_{D_m}}$ that are always *finite*-dimensional. Restricting to this case allows us to introduce our circuit models without technical complications that are only relevant in an infinite dimensional setting, such as in chapters 4-5.

In short, a *normalizer circuit over* G is a circuit composed of normalizer gates (i-ii-iii) acting on group-element states $|g\rangle$. The *size* of a normalizer circuit is the number of normalizer gates it contains. To complete the definition of this model, we define QFTs over G and give examples of automorphism gates and quadratic-phase gates.

[6] Decomposing $G = \mathbb{Z}_N^\times$ yields an efficient algorithm to compute the Euler Totient function and this knowledge can be used to factorize in polynomial time [92, chapter 10]. In turn, efficient *quantum* algorithms to decompose abelian groups exist, at least for "reasonably presented" (black-box) groups (cf. chapter 5).

[7] The rotors we consider are sometimes called *quantum fixed-axis rigid rotors* [177]

Input states: The allowed initial states of a normalizer circuit are group element states (1.4). At later steps the quantum state of the computation is of the form $\sum_{g\in G}\psi(g)|g\rangle$.

QFT over finite G: The QFT over $\mathbb{Z}_D$ implements a unitary change of basis on $\mathcal{H}_{\mathbb{Z}_D}$:

$$\mathcal{F}_{\mathbb{Z}_N} := \sum_{x,y\in\mathbb{Z}_N} \langle\tilde{y}\,|x\rangle|x\rangle\langle y|, \quad \text{with} \quad |\tilde{y}\rangle := \frac{1}{\sqrt{N}}\sum_{x\in\mathbb{Z}_N} \overline{e^{2\pi i\frac{xy}{N}}}|x\rangle \quad \text{for every } y\in\mathbb{Z}_N.$$

The global QFT over the entire group G acting on the entire space $\mathcal{H}_G$ is given by

$$\mathcal{F}_G = \mathcal{F}_{\mathbb{Z}_{D_1}}\otimes\cdots\otimes\mathcal{F}_{\mathbb{Z}_{D_m}} = \frac{1}{\sqrt{|G|}}\sum_{g,h\in G}\chi_g(h)|g\rangle\langle h|, \tag{1.5}$$

where χ_g are the *character functions* of the group G, which fulfill $\chi_g(x+y) = \chi_g(x)\chi_g(y) = \chi_g(x+y)$ for any $x,y,g\in G$ and are defined as follows:

$$\chi_g(h) = \exp\left(2\pi\mathrm{i}\sum_{i=1}^{m}\frac{g(i)h(i)}{D_i}\right). \tag{1.6}$$

A partial QFT is any operator obtained by replacing a subset of the gates $\mathcal{F}_{\mathbb{Z}_{D_i}}$ in this tensor product by identity operators. The unitarity of all QFTs above follows from well-known character orthogonality relationships[8].

Measurements: Throughout the thesis, measurements in the standard basis (1.20) at the end of normalizer circuit are always allowed, although in chapter 3 we will also allow measurements of any *generalized Pauli operator* $\sigma(a,g,h)$ over G, which we define to be of the form

$$\sigma(a,g,h) := \gamma^a Z(g)X(h), \qquad X(g) := \sum_{h\in G}|h+g\rangle\langle h|, \qquad Z(g) := \sum_{h\in G}\chi_g(h)|h\rangle\langle h|. \tag{1.7}$$

All operators $X(g)$, $Z(g)$ are unitary (the former just permute standard basis and the latter multiply by a complex phase). Generalized Pauli operators form a group $\mathcal{P}_G$ (cf. [134] or section 3.3), henceforth called the *Pauli group over* G.

Relationship to Clifford operations: A unitary operator U on $\mathcal{H}_G$ is called a *Clifford operator* over G if it maps the Pauli group $\mathcal{P}_G$ onto itself under conjugation $\sigma\to U\sigma U^\dagger$. The set of all Clifford operators forms a group, henceforth called the *Clifford group* $\mathcal{C}_G$. Formally, $\mathcal{C}_G$ is the (group theoretic) normalizer of the Pauli group in the full unitary group acting on $\mathcal{H}_G$.

It was proven in ([134] (see theorem 3.4, chapter 3) that every finite-G normalizer circuit is a Clifford operator, but it is currently not known whether all possible Clifford operators can be implemented via normalizer gates. Such a question is of considerable relevance, since the finding of a non-normalizer Clifford operation could lead to a new quantum gate. However, in section 3.3.3.2 we give supporting evidence (**lemma 3.5**) against the existence of such gates and further *conjecture* that any Clifford operator can be implemented as a poly-size normalizer circuit (**conjecture 3.1**). Further evidence is given below, where we explain how these notions are equivalent for regular Clifford circuits on qubits and qudits.

1.3.1 Examples with finite G

Here we give examples of Pauli and normalizer operations for several choices of finite abelian group G. We illustrate in particular how the definitions of the preceding section generalize existing notions of Pauli and Clifford operators for qubits and qudits.

[8] For any G, these relationships say that $\langle\chi_g,\chi_h\rangle = \frac{1}{|G|}\sum_{x\in G}\overline{\chi_g(x)}\chi_h(x)$.

1.3.1.1 Qubit Clifford circuits: $G = \mathbb{Z}_2^m$

Recall that qubit *Clifford circuits* [1, 2] are quantum circuits that normalize the qubit Pauli group and can be generated by sequences of CNOTs, CZ gates, Hadamard gates and Phase gates $S = \mathrm{diag}(1, i)$ (acting on arbitrary qubits). Below, we show that for $G = \mathbb{Z}_2^m$, qubit Clifford circuits become examples of normalizer gates for $G = \mathbb{Z}_2^m$; this was first observed in [134]. Note that, in this case, $\mathcal{H}_G = \mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2$ is a system of m qubits and its group-element basis $\{|x\rangle, x \in \mathbb{Z}_2^m\}$ is the standard basis labeled by m bit-strings.

1. **Hadamards**: Applying (1.5) one finds that the QFT over $\mathbb{Z}_2$ is simply the Hadamard gate H; the QFT over $\mathbb{Z}_2^m$ is $H^{\otimes m}$; and partial QFTs are obtained via combinations of single-qubit Hadamard action on qubit subsets.

2. **CNOT**, as a classical operation, implements the boolean map $(x_1, x_2) \to A(x_1, x_2) = (x_1, x_1 + x_2 \pmod 2)$ where A denotes an invertible 2×2 matrix over $\mathbb{Z}_2^2$, hence, a $\mathbb{Z}_2^2$ automorphism. It follows that $\mathrm{CNOT}|x_1, x_2\rangle = |A(x_1, x_2)\rangle$ is a $\mathbb{Z}_2^2$ automorphism gate.

3. **S and CZ**. Let A be an $m \times m$ matrix with entries in $\mathbb{Z}_2$ and let $a \in \mathbb{Z}_2^m$. Then, the following functions are quadratic[9] over $\mathbb{Z}^m$:

$$\xi_A : x \to (-1)^{x^T A x} \quad \text{and} \quad \xi_a : x \to \mathrm{i}^{(a^T x) \bmod 2}, \qquad x \in \mathbb{Z}_2^m. \tag{1.8}$$

   The fact that these functions are quadratic follows from the following equations[10]:

$$\xi_A(x + y) = \xi_A(x)\xi_A(y)(-1)^{x^T(A+A^T)y} \tag{1.9}$$
$$\xi_a(x + y) = \xi(x)\xi(y)(-1)^{q(x,y)} \quad \text{with } q(x, y) = (a^T x)(a^T y) \tag{1.10}$$

   As a particular case, we obtain that for $m = 1$ and $m = 2$ the functions $x \to i^x$ and $(x, y) \to (-1)^{xy}$ are quadratic. Finally, note the Clifford gates D, CZ are simply the quadratic functions associated to these gates, since:

$$D = \mathrm{diag}(1, i); \quad CZ = \mathrm{diag}(1, 1, 1, -1).$$

Moreover, all normalizer circuits over $\mathbb{Z}_2^m$ are also qubit Clifford circuits and, hence, these circuits families coincide. This follows from the fact that normalizer circuits leave the generalized Pauli group $\mathcal{P}_G$ invariant under conjugation and, moreover, for $G = \mathbb{Z}^m$, $\mathcal{P}_G$ becomes the standard qubit Pauli group: to see this, let σ_x and σ_z denote the standard Pauli matrices and let $g \in \mathbb{Z}_2^m$ be an m-bit string; then, applying definition (1.7) one finds that

$$X(g) = \sigma_x^{g(1)} \otimes \cdots \otimes \sigma_x^{g(m)}, \qquad Z(g) = \sigma_z^{g(1)} \otimes \cdots \otimes \sigma_z^{g(m)}, \qquad \text{for } g \in \mathbb{Z}_2^m \tag{1.11}$$

In short, $X(g)$ is a tensor product of σ_x-matrices and identities, and $Z(g)$ is a tensor product of σ_z-matrices and identities. Therefore, every Pauli operator (1.7) has the form $\sigma \propto U_1 \otimes \cdots \otimes U_m$ where each U_i is a single-qubit operator of the form $\sigma_x^u \sigma_z^v$ for some $u, v \in \mathbb{Z}_2$. This recovers the usual notion of a Pauli operator on m qubits [1, 2] .

[9] Note that the exponent in ξ_A is polynomial of degree 2 in x, whereas the exponent in ξ_a has degree 1. Hence, the notion of quadratic functions we use differs from the usual notion of "quadratic form" used in, e.g., [135, 136]

[10] Identity (1.10) can be proved by distinguishing between the 4 cases $a^T x, a^T y \in \{0, 1\}$.

1.3.1.2 Qudit Clifford circuits: $G = \mathbb{Z}_d^m$

In this case the Hilbert space $\mathcal{H}_G = \mathbb{C}^d \otimes \cdots \otimes \mathbb{C}^d$ is a system of m d-level systems (qudits) and Pauli operators have the form $\sigma \propto U_1 \otimes \cdots \otimes U_m$, where each U_i is a single-qudit operator of the form $X_d^u Z_d^v$ for some $u, v \in \mathbb{Z}_d$, where X_d and Z_d are the usual generalizations of σ_x and σ_z for d-level systems:

$$X_d = \sum_{x \in \mathbb{Z}_d} |x+1\rangle\langle x| \quad \text{and} \quad Z_d = \sum_{x \in \mathbb{Z}_d} \mathrm{e}^{2\pi \mathrm{i} x/d} |x\rangle\langle x| \tag{1.12}$$

Examples of normalizer gates over $\mathbb{Z}_d^m$ are the standard Clifford operations for qubits,

$$\mathrm{SUM}_d = \sum_{x \in \mathbb{Z}_d} |x, x+y\rangle\langle x, y|, \quad \mathrm{CZ}_d = \sum \omega_d^{xy} |x, y\rangle\langle x, y|, \quad \omega_d := \mathrm{e}^{2\pi \mathrm{i}/d} \tag{1.13}$$

$$\mathcal{F}_{\mathbb{Z}_d} = \frac{1}{\sqrt{d}} \sum \mathrm{e}^{2\pi \mathrm{i} xy/d} |x\rangle\langle y|, \qquad S_d = \sum \xi_d^{x(x+d)} |x\rangle\langle x|, \quad \xi_d := \mathrm{e}^{\pi \mathrm{i}/d}. \tag{1.14}$$

To show that SUM_d is a normalizer gate, note that $(x, y) \to (x, x+y)$ is indeed an automorphism of $\mathbb{Z}_d \times \mathbb{Z}_d$. The gates CZ_d and S_d are quadratic phase gates [134, section 11]. In addition, the "multiplication gate" $M_{d,a} = \sum |ax\rangle\langle x|$ is also a normalizer gate, for every $a \in \mathbb{Z}_d$ which is coprime to d. Indeed, for such a the map $x \to ax$ is known to be an automorphism of $\mathbb{Z}_d$. Furthermore, it is known that the *entire* Clifford group for qudits (for arbitrary d) is generated by the gates SUM_d, $\mathcal{F}_{\mathbb{Z}_d}$, S_d and M_a [136]; hence, for $G = \mathbb{Z}_2^m$ normalizer circuits become the qudit Clifford circuits.

Lastly, the diagonal gates associated to the functions below are quadratic phase gates [134]:

$$z \to \omega^{bz^2+cz} \quad \text{and } z \to \gamma^{bz(z+d)}; \quad \omega := \mathrm{e}^{2\pi \mathrm{i}/d}, \ \gamma := \omega^{1/2}. \tag{1.15}$$

1.3.1.3 Shor's discrete quantum Fourier transform: $G = \mathbb{Z}_{2^m}$

In our last example, we consider G to be the single cyclic group $G = \mathbb{Z}_{2^m}$. In this case, $\mathcal{H}_G$ is a 2^m-dimensional Hilbert space with standard basis $\{|0\rangle, \ldots, |2^m - 1\rangle\}$. Note that, in contrast with previous examples (e.g. $G = \mathbb{Z}_2^m$), the structure of $\mathbb{Z}_{2^m}$ does not naturally induce a factorization of the Hilbert space into m single-qubit systems. As a consequence, normalizer gates over $\mathbb{Z}_{2^m}$ act *globally* on $\mathcal{H}_G$.

Examples of normalizer gates are now given by $\mathcal{F}_{\mathbb{Z}_{2^m}}$, S_{2^m} and $M_{2^m, a}$, following the definitions of the previous example with $d = 2^m$. Crucially, here the gate $\mathcal{F}_{\mathbb{Z}_{2^m}}$ is the "standard" F_{2^m} QFT used in e.g Shor's algorithm and the phase estimation quantum algorithm [49].

1.4 Normalizer circuits over infinite G

We now extend the circuit model from previous section introducing normalizer gates over arbitrary infinite abelian groups $G = \mathbb{Z}_{D_1} \times \ldots \times \mathbb{Z}_{D_a} \times \mathbb{Z}^b \times \mathbb{T}^c$ with associated infinite-dimensional Hilbert spaces $\mathcal{H}_G = \mathcal{H}_{\mathbb{Z}_{D_1}} \otimes \cdots \otimes \mathcal{H}_{\mathbb{Z}_{D_m}} \otimes \mathcal{H}_{\mathbb{Z}}^{\otimes m} \otimes \mathcal{H}_{\mathbb{T}}^{\otimes m}$. We investigate these circuits in detail in chapter 4 and use them to understand Shor's factoring algorithm in chapter 5.

1.4.1 Infinite-dimensional aspects of infinite-group normalizer gates

We now introduce some idiosyncratic features of infinite-dimensional Hilbert spaces that, as explained next, will affect our treatment of infinite-dimensional quantum states, quantum Fourier transforms and allowed measurement bases. These aspects will be important to construct well-defined computational models based on infinite-group normalizer gates (section 1.4.2).

Infinite-dimensional quantum states

For $\mathbb{Z}$ (resp. $\mathbb{T}$), a quantum state in $\mathcal{H}_{\mathbb{Z}}$ (resp. $\mathcal{H}_{\mathbb{T}}$) is associated to any normalized *sequence* of complex numbers $\{\psi(x) : x \in \mathbb{Z}\}$ with $\sum |\psi(x)|^2 = 1$ (resp. normalized complex *function* $\{\phi(p) : p \in \mathbb{T}\}$ with $\int_{\mathbb{T}} \mathrm{d}p\, |\phi(p)|^2 = 1$):

$$|\psi\rangle = \sum_{x\in\mathbb{Z}} \psi(x)|x\rangle; \qquad |\phi\rangle = \int_{\mathbb{T}} \mathrm{d}p\ \phi(p)|p\rangle, \tag{1.16}$$

where $\mathrm{d}p$ denotes the standard Haar/Lebesgue measure on $\mathbb{T}$ and we introduced the *plane-wave states*

$$|p\rangle := \sum_{z\in\mathbb{Z}} \overline{e^{2\pi i z p}}|z\rangle \quad p \in \mathbb{T} = [0, 1). \tag{1.17}$$

Plane-wave states, as well as those $|\psi\rangle$ states whose squared sums are not finite, are *non-normalizable* unphysical states that do not belong to $\mathcal{H}_{\mathbb{Z}}$. Nonetheless, it will be convenient in our formalism to consider them.

Infinite-dimensional Quantum Fourier transforms

The QFT over $\mathbb{Z}$: Though non-normalizable, the plane-wave states $|p\rangle$ (1.17) labeled by torus elements define a dual "orthonormal basis"[11] of $\mathcal{H}_{\mathbb{Z}}$, in the sense that the map $\psi \to \hat{\psi}$:

$$|\psi\rangle \xrightarrow{\text{QFT over } \mathbb{Z}} |\hat{\psi}\rangle = \int_{\mathbb{T}} \mathrm{d}p\ \hat{\psi}(p)|p\rangle \quad \text{with} \quad \hat{\psi}(p) := \langle p|\psi\rangle = \sum_{x\in\mathbb{Z}} \mathrm{e}^{2\pi i p x}\psi(x), \tag{1.18}$$

is a well-defined *unitary* transformation and $\langle p|p'\rangle = \delta(p - p')$ is a normalized Dirac delta. The *QFT over* $\mathbb{Z}$ (denoted $\mathcal{F}_{\mathbb{Z}}$) is defined as the unitary transformation that implements the change of basis (1.18). It is crucial to note that, strictly speaking, the QFT over $\mathbb{Z}$ is an isomorphism from the Hilbert space $\mathcal{H}_{\mathbb{Z}}$ onto $\mathcal{H}_{\mathbb{T}}$, since it changes the underlying integral basis into a continuous one labeled by angles. Therefore, it is natural to identify $\mathcal{H}_{\mathbb{Z}} = \mathcal{H}_{\mathbb{T}}$ as spaces associated to two different canonical bases of quantum states of a single *physical* system (i.e., a *quantum rotor* with angular position and integral momentum [178]). Henceforth, we adopt this convention and use the index group $\mathbb{Z}$, $\mathbb{T}$ to denote in which basis we work.

We immediately observe, that, because the $\mathbb{Z}$ group-element and Fourier basis have different cardinality, their associated infinite-dimensional QFTs must have two unique exotic features with no finite-dimensional counterpart.

(a) **QFTs are not gates:** 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 onto each other. Instead, the QFT is a change of basis between two orthonormal basis, but it does not define a unitary rotation as in the standard (finite dimensional) circuit model[12]. This is in strong contrast, with the finite group case where the QFT could be implemented either as a change of basis or as a gate: e.g. the QFT over $\mathbb{Z}_N$ implemented the change of basis $|y\rangle \to |\hat{y}\rangle$.

[11] Although we use this terminology, the $|p\rangle$ states do not form a basis in the usual sense since they lie outside of $\mathcal{H}_{\mathbb{Z}}$. Rigorously, the $|p\rangle$ kets should be understood as Dirac-delta measures, or as Schwartz-Bruhat tempered distributions [179, 180]. The theory of rigged Hilbert spaces [181–184] (often used to study observables with continuous spectrum) establishes that the $|p\rangle$ kets can be "used as a basis" for all practical purposes.

[12] Mathematically, this Fourier transform is a unitary transformation between two different functional spaces, $L^2(\mathbb{Z})$ and $L^2(\mathbb{T})$. The latter two define one quantum mechanical system with two possible bases (of Dirac-delta measures) labeled by $\mathbb{Z}$ and $\mathbb{T}$. In the finite dimensional case, the picture is simpler because the QFT is a unitary transformation of $L^2(\mathbb{Z}_N)$ onto itself. (These facts are consequences of the Plancherel theorem for locally compact abelian groups [185, 186].)

(b) **QFT over $\mathbb{T}$.** A technical obstacle to construct a well-defined infinite-dimensional normalizer circuit model is that such models cannot be based on QFTs over $\mathbb{Z}$ only because they cannot be *concatenated* one after another: this happens because the QFT over $\mathbb{Z}$ changes the underlying group labeling the basis from $\mathbb{Z}$ to $\mathbb{T}$, and a QFT over $\mathbb{Z}$ is only a well-defined normalizer gate in the $\mathbb{Z}$ basis. To cope with this issue, we need to allow a normalizer circuit over $\mathbb{Z}$ to contain not only QFTs over $\mathbb{Z}$ but also a *distinct type* of ***QFT over*** $\mathbb{T}$ that re-expresses a state $|\phi\rangle = \int_{\mathbb{T}} \mathrm{d}p\ \psi(p)|p\rangle$ back in the integer basis of $\mathcal{H}_{\mathbb{Z}}$

$$|\phi\rangle \quad \xrightarrow{\text{QFT over } \mathbb{T}} \quad |\hat{\phi}\rangle = \sum_{x\in\mathbb{Z}} \hat{\phi}(x)|x\rangle \quad \text{with} \quad \hat{\phi}(x) := \overline{\langle x|\phi\rangle} = \int_{\mathbb{T}} \mathrm{d}p\ \mathrm{e}^{2\pi i p x}\phi(p). \tag{1.19}$$

We stress that, in our circuit model, the QFT over $\mathbb{Z}$ (resp. over $\mathbb{T}$) may only be applied if we work in the group-element basis labeled by $\mathbb{Z}$ elements (resp. $\mathbb{T}$ elements). Also, some readers may note, at this point, that the latter QFT over $\mathbb{T}$ implements the well-known classical *Fourier series* of a *periodic* real function as a quantum gate. Conversely, the QFT over $\mathbb{Z}$ is nothing but the quantum version of the (also well-known) discrete-time Fourier transform [187], which turns a discretized signal into a periodic function.

Designated bases

Above, we saw the action on QFTs on the Hilbert space $\mathcal{H}_{\mathbb{Z}} = \mathcal{H}_{\mathbb{T}}$ is two perform a change between *two* distinct natural bases. As a consequence of this feature, it follows that computational models based on normalizer gates over $\mathbb{Z}$ and $\mathbb{T}$ do not have a unique preferred "standard basis", as opposed to the finite-dimensional setting of section 1.3. Instead, we will let a normalizer circuit over an infinite group $G = \mathbb{Z}_{D_1} \times \ldots \times \mathbb{Z}_{D_a} \times \mathbb{Z}^b \times \mathbb{T}^c$ act on a *time-dependent designated basis*: the latter is a "standard basis" that is subject to change along the computation.

Definition 1.1 (**Designated basis**)**.** At every time step t in a normalizer circuit there is a *designated basis* $\mathcal{B}_{G_t}$ of the Hilbert space $\mathcal{H}_G$, which is the group-element basis of a group G_t picked from a family of size 2^{b+c} constructed below (1.20). The pair $(G_t, \mathcal{B}_{G_t})$ determines the allowed normalizer gates at time t as well as the basis in which measurements are performed.

Specifically, each *designated basis* $\mathcal{B}_{G'}$ is the group-element basis of a group G' of form

$$G' := \mathbb{Z}_{D1} \otimes \ldots \mathbb{Z}_{Da} \times G'_1 \times \cdots \times G'_{b+c}, \quad \text{where each} \quad G_i \in \{\mathbb{Z}, \mathbb{T}\}. \tag{1.20}$$

$$\mathcal{B}_{G'} := \{|g\rangle = |g(1)\rangle \otimes \cdots \otimes |g(m)\rangle; \quad g = (g(1), \ldots, g(m)) \in G'\}. \tag{1.21}$$

The notation $G_i = \mathbb{T}$ indicates that $|g(i)\rangle$ is a Fourier state of $\mathbb{Z}$ (1.17). The states $|g\rangle$ are product-states with respect to the tensor-product decomposition of $\mathcal{H}_G$. There are 2^b possible choices of groups in (1.20) (which are, in fact, related via Pontryagin duality[13]) and 2^b inequivalent group-element basis of the Hilbert space.

Example 1: The designated basis $\mathcal{B}_G$ is the group-element basis labeled by G elements:

$$|x(1)\rangle\otimes\cdots\otimes|x(a)\rangle\otimes|y(1)\rangle\otimes\cdots\otimes|y(b)\rangle\otimes|z(1)\rangle\otimes\cdots\otimes|z(c)\rangle, \quad x(i) \in \mathbb{Z}_{Di};\ (y,z) \in \mathbb{Z}^b\times\mathbb{T}^c. \tag{1.22}$$

[13] From a mathematical point of view, all groups (1.20) form a family (in fact, a category) which is generated by replacing the factors G_i of the group $\mathbb{Z}_{D_1} \times \ldots \times \mathbb{Z}_{D_a} \times \mathbb{Z}^b$ with their character groups G_i^* (cf. chapter 2), and identifying isomorphic groups. Pontryagin duality [188–190, 185, 186, 191, 192] 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.

Example 2: In turn, choosing the Fourier basis in the $(a+b)$-th space we obtain in turn

$$|x\rangle \otimes \left(|y'(1)\rangle \otimes \cdots \otimes |y'(b-1)\rangle \otimes |p\rangle\right) \otimes |z'\rangle \quad , (y', p, z') \in \left(\mathbb{Z}^{b-1} \times \mathbb{T}\right) \times \mathbb{T}^c, \tag{1.23}$$

which is labeled by the elements of $\mathbb{Z}_{D_1} \times \ldots \times \mathbb{Z}_{D_a} \times \mathbb{Z}^{b-1} \times \mathbb{T}^{c+1}$.

Update rules: The action of a QFT over $\mathbb{Z}$ or $\mathbb{T}$ in a normalizer circuit will be *precisely* to change the designated basis of the computation, as explained next:

1. Precisely, when we say that the *QFT over* $\mathbb{Z}$ (1.17) is applied to $|\psi\rangle$, we mean that the designated basis is changed from the $\mathbb{Z}$ group-element to its Fourier $\mathbb{T}$-element basis: here, the state does not actually change (no gate is physically applied), but the normalizer gates acting after the QFT will be associated with $\mathbb{T}$ (*not* $\mathbb{Z}$), and measurements will be performed in the $\mathbb{T}$ basis (cf. next section 1.4.2). Correspondingly, the wavefunction of the state $|\psi\rangle$ ought to be re-expressed in the Fourier basis (1.18).

2. Respectively, when we say that the *QFT over* $\mathbb{T}$ is applied to $|\psi\rangle$, we mean that the designated basis is changed from the $\mathbb{T}$-element basis to the $\mathbb{Z}$ group-element basis. Like in the previous case, we must re-express the state $|\psi\rangle$ in the new designated basis (1.19).

1.4.2 The full infinite-dimensional normalizer circuit model

We now present our infinite-group normalizer circuit model in precise terms. Below, we fix

$$G = \mathbb{Z}_{D_1} \times \ldots \times \mathbb{Z}_{D_a} \times \mathbb{Z}^{\otimes b} \times \mathbb{T}^c, \qquad \mathcal{H}_G = \mathcal{H}_{\mathbb{Z}_{D_1}} \otimes \cdots \otimes \mathcal{H}_{\mathbb{Z}_{D_a}} \otimes \mathcal{H}_{\mathbb{Z}}^{\otimes b} \otimes \mathcal{H}_{\mathbb{T}}^{\otimes c},$$

and let $m := a+b+c$ be the number of total registers of the computation. In this decomposition, the parameters a, b, c, D_i can be chosen arbitrarily.

Roughly speaking, a ***normalizer circuit over*** G of size T is a quantum circuit $\mathcal{C} = U_T \cdots U_1$ composed of T *normalizer gates* U_i as in section 1.3. However, in contrast with finite-group setting, now not all gates U_i need to be normalizer gates over the group G, but over any group G' (1.20) that labels one of the allowed designated basis of G. Specifically, a normalizer circuit over G is any quantum circuit generated by the following rules:

- **Input states:** The input states of a normalizer computation are elements of some designated group basis $\mathcal{B}_{G_0}$ at time zero. For instance, if we choose $G_0 = \mathbb{Z}_{D_1} \times \ldots \times \mathbb{Z}_{D_a} \times \mathbb{Z}^{\otimes b} \times \mathbb{T}^c$, in our notation, then the registers $\mathcal{H}_{\mathbb{Z}}^{\otimes b}$ and $\mathcal{H}_{\mathbb{T}}^{\otimes c}$ are fed, respectively, with standard-basis $|n\rangle$, $n \in \mathbb{Z}$ and Fourier-basis states $|p\rangle$, $p \in \mathbb{T}$.

- **Structure of the circuit:**

  - *At time* $t = 1$, the gate U_1 is applied, which is either an automorphism gate, quadratic phase gate over G_0 (see section 1.4.3) or a QFT. As earlier, we allow the application of *partial QFTs* on any subset of the individual registers $\mathcal{H}_{\mathbb{Z}_{N_i}}$, $\mathcal{H}_{\mathbb{Z}}$, $\mathcal{H}_{\mathbb{T}}$ and the full *QFT over* G is the combination of all partial QFTs acting on the smaller registers.
  - *At time* $t = 1$, after the action of U_1, the designated basis is changed from $\mathcal{B}_{G_0}$ to $\mathcal{B}_{G_1}$, for some group G_1 in the family (1.20), which may only differ from G_0 if a QFT was applied. Specifically: whenever if $G_0(i) = \mathbb{Z}$ (respectively, if $G_0(i) = \mathbb{T}$) and a QFT acts on the ith register, then the group G_1 is chosen so that $G_1(i) = \mathbb{T}$ (resp. $G_1(i) = \mathbb{Z}$); in all other case $G_1(i) = G_0(i)$ is left unchanged.

- *At time* $t = 2$, the gate U_1 is applied, which is, again, either an automorphism gate, quadratic phase gate or a QFT over G_1. The designated basis is changed from $\mathcal{B}_{G_1}$ to $\mathcal{B}_{G_2}$, for some group G_2, following the rules of the previous step.
- The gates $U_3, \ldots, U_t$ 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 (1.20)), for all $t = 3, \ldots, T$. Thus, after all gates have been applied, the designated basis is G_T.
- After the circuit, a measurement in the designated basis G_T is performed.

1.4.3 Examples of infinite-dimensional normalizer gates

Finally, we illustrate the above definitions giving examples of normalizer gates over $\mathbb{Z}^m$ and $\mathbb{T}^m$.

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

First, the formulas below show how the QFT over $\mathbb{Z}$ acts on quantum states:[14]

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

Second, the gates below are examples of automorphism and quadratic phase gates, 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 (lemma 2.8) and quadratic functions (theorem 2.1) that we introduced in chapter 2.

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

We now take a look at the effect of the quantum Fourier transform over $\mathbb{T}$ over some states.[15]

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

[14] The transformations we depict can be found in standard signal processing textbooks [187].
[15] These examples also illustrate that the QFT over $\mathbb{T}$ is the quantum version of the Fourier series [187].

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 $|x\rangle$ to $|-x\rangle$. This is a general phenomenon, which we shall observe throughout the thesis.

Examples of automorphism gates over $\mathbb{T}^m$ are the sum and sign-flip gates:

$$\mathrm{SUM}_{\mathbb{T},b} = \iint_{\mathbb{T}} \mathrm{d}p\mathrm{d}q\, |p, q + bp\rangle\langle p, q|, \qquad M_{\mathbb{T},s} = \int_{\mathbb{T}} \mathrm{d}p\, |sp\rangle\langle p|$$

where b is any integer and $s = \pm 1$ (the correctness of these formulas comes from lemma 2.8).

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[16]). In the case $m = 1$, this is equivalent to saying that any such gate is of the form

$$\int_{\mathbb{T}} \mathrm{d}p\, \exp\left(2\pi i b p\right) |p\rangle\langle p|$$

with b an arbitrary integer.

[16]This fact can be understood in the light of a later result, theorem 2.1 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.

Chapter 2

Classical group theoretic and algorithmic techniques

In this chapter we develop a series of *classical* **group-theoretic** and **algorithmic techniques**. These tools will provide a basic language in this thesis to attack quantum computing problems in chapters 3-5. The main contributions in this chapter are threefold:

I. **A theory of matrix representations for abelian-group homomorphisms**. We show that, similarly to real linear maps, homomorphisms between elementary abelian groups of form $\mathbb{R}^a \times \mathbb{Z}^b \times \mathbb{T}^c \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_d}$ admit concise classical descriptions in terms of matrix representations with well-behaved algebraic properties (lemmas 2.6, 2.7). We give a *normal form* that fully characterizes the structure of such matrices (lemma 2.8).

II. **Normal forms for quadratic and bi-character functions** over abelian-groups (theorem 2.1, lemma 2.10), based on matrix representations of group homomorphisms.

III. **Classical algorithms for group theoretic problems.** We give *efficient* classical algorithms for solving *linear systems of equations over abelian groups* $\alpha(x) = b$ where $\alpha : G \to H$ is an abelian-group homomorphism, $x \in G, b \in H$ (theorem 2.2): our algorithms decide the existence of and find *general solution* (definition 2.3) for any such system if a matrix representation A of α is provided; our technique is based on a reduction to mixed real-integer systems of equations (2.47) and the Smith normal form.

We highlight that in chapters 3-5 we will identify a rich variety of *quantum applications* for results I-II-III. For this reason, we regard the latter as main contributions of the thesis. To illustrate the versatility of these methods, we anticipate some of these applications:

- *Matrix representations* (result I) and *our normal form for quadratic functions* (result II) will be applied to define efficient *classical encodings* for abelian-group normalizer gates (s. 3.3.3, 4.2) and infinite-dimensional stabilizer states (s. 4.5.1); as well as to derive our classical simulation results (theorems 3.7, 4.1, 4.2) and our complexity theoretic hardness results (theorem 5.6, 5.7).

- *Our normal form for quadratic functions* can also be applied to characterize the wavefunctions of abelian-group[1] *stabilizer states* in combination with another normal form for the latter (theorem 3.4). These results partially[2] extend Gross' discrete Hudson theorem, which describes odd-dimensional pure qudit stabilizer states via quadratic forms [193, 194].

[1] This result is proven only for finite-dimensional stabilizer states but it can be easily extended to the infinite dimensional settings of chapter 4, appendix E.

[2] The theorem in [193, 194] also says that such states are precise those with a positive Wigner representation; this fact does not easily extend to even dimensions due to certain nonlocal features (cf. discussion in chapter 4).

- *Our group theoretic algorithms* (result III) are used ubiquitously in chapters 3-5, more importantly, to manipulate abelian-group stabilizer states, stabilizer groups and generalized abelian-group Pauli operators. As examples of key results where these techniques play a key role, the reader might look at theorems 3.2, 3.4, 3.5, 4.1, 4.3 and our extended Cheung-Mosca quantum algorithm 5.5 for the group decomposition problem.

Last, we point out that the concept of quadratic function explored by [134] and this thesis has a great theoretical value for understanding the algebraic structure of the stabilizer formalism. For instance, we saw in the examples of chapter 1 that this notion yields a one-line unified definition for all diagonal Clifford gates for qubits and qudits, as well as a (previously unknown) common group-theoretic operational interpretation for these gates (a generalization of this result for finite abelian groups is given by lemma 3.5). Furthermore, our later result (theorem 3.4) further shows that these functions yield a description of all phases of stabilizer states. It would be interesting to explore if quadratic functions have applications beyond this thesis in the area of fault-tolerant quantum computation, e.g., to understand better which quantum error correcting codes have transversal cubic (non-Clifford) diagonal gates [70]. We propose these questions to the reader as motivation for further research.

2.0.1 Relationship to previous work

The author makes no claim about the novelty of the methods in this chapter for solving non-quantum problems: it is quite possible that some of the results I-II-III might be known, e.g., by group theorists and/or computer scientists working on (classical) algorithms for algebraic problems, even if we did not find explicit proofs for them in the literature. The connections to existing classical works that we are aware of are pointed out throughout the chapter.

In our view, the value of the techniques in this chapter comes from their applications to solve problems in quantum information and computation. In this sense, we regard our results I-II-III as novel and our new and the techniques employed in their proofs of interest to the general quantum audience.

To the best of our knowledge, Van den Nest [134, section 6] and us [63], were the first to point out and exploit the notions of quadratic functions and abelian-group-homomorphism matrix representations in quantum computation theory. Quadratic functions over finite abelian groups were introduced in [134, 63], and over infinite groups in [64]. Prior to these works, quadratic *forms* (which are instances of quadratic functions) were used, e.g., in [135, 136] to study the qubit/qudit stabilizer formalism (see also section 3.5.1). VdN and us are also among the first to introduce classical algorithms for solving linear systems of equations. Prior to us, some quantum applications of classical algorithms for solving linear systems of equations were known (though the concept had not been introduced). Implicitly, methods for solving certain instances of these systems (of the type given in lemma 3.1.(e)) were employed in the classical post-processing of quantum algorithms for abelian hidden subgroup problems [118] and in the classical simulation algorithm of [134, theorems 3,4]. This technique was formalized in a group-theoretic language and generalized to the full extent of theorem 2.2 by us in [63, 64], as part of this thesis.

Our account in this chapter is based on [64] (joint work with Cedric Yen-Yu Lin and Maarten Van den Nest), which contains our most general algorithms for infinite-abelian-group problems.

2.0.2 Chapter outline

The proofs of the theorem in this chapter have been moved to appendices 1.1-1.6, in order to give more attention to the quantum contributions of the thesis. Section 2.1.1 surveys some

necessary notions of group and character theory. In section 2.2 we develop our theory of matrix representations of group homomorphisms. In section 2.3 we present normal forms for quadratic functions. In section 2.4 we study computational aspects of the abelian groups in this thesis, including the computational complexity of solving systems of linear equations over groups (s. 2.4.2), for which we give polynomial-time deterministic classical algorithms.

2.1 Introduction to abelian group theory

2.1.1 Definitions

Elementary abelian groups: A commutative group G is called *elementary* if it is of the form

$$G = \mathbb{Z}^a \times \mathbb{R}^b \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_c} \times \mathbb{T}^d \tag{2.1}$$

Below, we often let $F := \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_c}$ be the finite subgroup in the above decomposition. Though the main results in this thesis are not for normalizer circuits over real-number groups $\mathbb{R}^b$ (see appendix E), we consider these groups in this chapter to develop of our classical methods.

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 $G = G_\mathbb{Z} \times G_\mathbb{R} \times G_F \times G_\mathbb{T}$.

Characteristic: The group characteristic $\mathrm{char}(G)$ of a primitive group is a number defined as

$$\mathrm{char}(\mathbb{Z}) := 0, \quad \mathrm{char}(\mathbb{R}) := 0, \quad \mathrm{char}(\mathbb{Z}_N) := N, \quad \mathrm{char}(\mathbb{T}) := 1. \tag{2.2}$$

Group theoretically, $\mathrm{char}(G)$ can be equivalently defined as (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 $\mathbb{R}$).

Group element encodings: 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

$$x_i \equiv g_i \pmod{c_i} \quad \text{for every } i = 1, \ldots, m. \tag{2.3}$$

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 $0 \in G$.

2.1.2 Character functions and character duality

Definition 2.1 (Character [188, 195]). Let G be an elementary abelian group. A character of G is a complex function χ_μ on G that fulfills two properties:

(i) $\chi(g+h) = \chi(g)\chi(h)$, for every $g, h \in G$, **(ii)** $|\chi(g)| = 1$, for every $g \in G$.

Properties: For any two characters χ_1, χ_2 the function $\chi_1\chi_2$ is a new character. Furthermore, character functions form a new elementary abelian group under the functional point-wise product called the *character group* or *dual group* $\widehat{G}$. Finally, the character group of a direct product group $G = G_1 \times \cdots \times G_m$ is the product of character groups $\widehat{G} = \widehat{G}_1 \times \cdots \times \widehat{G}_m$.

Examples of groups and their character groups

Let $G = G_1 \times \cdots \times G_m$ be an elementary abelian group. Then $\widehat{G}$ is isomorphic to another elementary abelian group G^* obtained via the following map:

$$G = \mathbb{R}^a \times \mathbb{T}^b \times \mathbb{Z}^c \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_d} \quad \to \quad G^* := \left(\mathbb{R}^a \times \mathbb{Z}^b \times \mathbb{T}^c \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_d}\right) \cong \widehat{G}. \tag{2.4}$$

Thus, in particular, $\widehat{\mathbb{R}}$ is isomorphic to $\mathbb{R}$ itself and similarly $\widehat{\mathbb{Z}_{N_i}}$ is isomorphic to $\mathbb{Z}_{N_i}$ 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 (2.4) that the dual group of $\widehat{G}$ is isomorphic to G itself. This is a manifestation of the Pontryagin-Van Kampen duality [188–190] (cf. lemma 2.2).

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

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

$$\chi_x(y) := \exp(2\pi i x y), \quad \text{for every } x,\, y \in \mathbb{R}. \tag{2.5}$$

  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

$$\chi_x(y) := \exp\left(\frac{2\pi i}{N}\, xy\right), \quad \text{for every } x,\, y \in \mathbb{Z}_N. \tag{2.6}$$

  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

$$\chi_p(m) := \exp(2\pi i p m), \quad \text{for every } p \in \mathbb{T},\, m \in \mathbb{Z}, \tag{2.7}$$

  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

$$\chi_m(p) := \exp(2\pi i p m), \quad \text{for every } p \in \mathbb{T},\, m \in \mathbb{Z}; \tag{2.8}$$

  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 (2.4), we denote

$$G^* := \mathbb{R}^a \times \mathbb{Z}^b \times \mathbb{T}^c \times F. \tag{2.9}$$

Considering an arbitrary element

$$\mu = (r_1, \ldots, r_a, z_1, \ldots, z_b, t_1, \ldots, t_c, f_1, \ldots, f_d) \in G^*, \tag{2.10}$$

the associated character is given by the product

$$\chi_\mu := \chi_{r_1} \cdots \chi_{r_a}\ \chi_{z_1} \cdots \chi_{z_b}\ \chi_{t_1} \cdots \chi_{t_c}\ \chi_{f_1} \cdots \chi_{f_d} \tag{2.11}$$

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

$$\widehat{G} = \{\chi_\mu : \mu \in G^*\}. \tag{2.12}$$

2.1.3 Duality theory of abelian groups

Note that rule (2.4) immediately implies that $(G^*)^* = G$, i.e., the character group of G^* is $\{\chi_g : g \in G\}$, where χ_g is defined in full analogy with (2.11). Furthermore, these equations illustrate two fundamental features of elementary abelian groups and their character functions.

Lemma 2.1 (**The character group is elementary**). For every $\mu, \nu \in G^*, g \in G$ it holds that

$$\chi_{\mu+\nu}(g) = \chi_\mu(g)\chi_\nu(g). \tag{2.13}$$

As a consequence, the map $\mu \to \chi_\mu$ realizes the group isomorphism between G^* and $\widehat{G}$.

Lemma 2.2 (**Group-character duality**). For every $g \in G$ and $\mu \in G^*$ we have

$$\chi_\mu(g) = \chi_g(\mu). \tag{2.14}$$

This identity implies that the map $g \to \chi_g$ defines a group isomorphism between G and the character group of $\widehat{G}$, establishing a duality between groups and their associated characters[a]

[a] This is a manifestation of the Pontryagin-Van Kampen duality [188–190, 185, 186, 191, 192], which says that any locally compact abelian group G is isomorphic to $\widehat{\widehat{G}}$ via the map $g \to \widetilde{\chi}_g$ where $\widetilde{\chi}_g(\chi_\mu) = \chi_\mu(g)$.

Both lemmas 2.2 follow from inspection of the characters of $\mathbb{R}, \mathbb{Z}, \mathbb{T}$ and $\mathbb{Z}_N$ defined in (2.5)-(2.8). The lemmas also reflect the strong duality between G and G^*.

Finally, the definition of every character function $\chi_a(b)$ as given in (2.5)-(2.8), 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 (2.11) 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 2.3. Let $g \in G$ and $\mu \in G^*$. For every $x, y \in \mathbb{R}^m$ such that $x \equiv g \pmod{G}$ and $y \equiv \mu \pmod{G^*}$, we have

$$\chi_y(x) = \chi_\mu(g). \tag{2.15}$$

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

2.1.4 Duality of subgroups and morphisms

Character functions give rise to set-theoretical dualities among abelian group subgroups and morphisms, via the notions of *dual morphisms* and *subgroup annihilators*[3]. We review these concepts next.

Dual morphism: Let $\alpha : G \to H$ be a continuous group homomorphism between two elementary abelian groups G and H. Then, there exists a unique continuous group homomorphism $\alpha^* : H^* \to G^*$, which we call the **dual homomorphism** of α [188, prop. 30], defined as

$$\chi_{\alpha^*(\mu)}(g) = \chi_\mu(\alpha(g)). \tag{2.16}$$

Again, we have $\alpha^{**} = \alpha$ by duality.

[3] Annihilator subgroups have been called "orthogonal subgroups" in some quantum computing works [118, 134, 63]. We avoid using this term because it is hardly ever used in group theory and because "subgroup orthogonality" differs from the usual "orthogonality" of vector spaces.

Annihilator subgroup: Let G be an elementary abelian group and X be any subset of G. The *annihilator*[4] $X^\perp$ is the subset

$$X^\perp := \{\mu \in G^* : \chi_\mu(x) = 1 \text{ for every } x \in X\}. \tag{2.17}$$

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

$$Y^\perp := \{x \in G : \chi_\mu(x) = 1 \text{ for every } \mu \in Y\}. \tag{2.18}$$

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 related to each other and, in fact, identical sets *iff* X is a closed subgroup.

Lemma 2.4 (**Annihilator properties** [189])**.** Let X, Y and H, K be, respectively, two arbitrary *subsets* and two *closed subgroups* of an elementary abelian group G. Then the following holds.

For subsets:

(a) $X^\perp$ 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 $X^{\perp\perp} \subseteq Y^{\perp\perp}$.

For closed subgroups:

(a) $H^{\perp\perp} = H$.
(c) $H^\perp$ is isomorphic to $(G/H)^*$.
(d) $|H^\perp| = |G/H| = |G|/|H|$ if G is *finite*.
(e) $(H \cap K)^\perp = \langle H^\perp, K^\perp \rangle$.

2.1.5 Final note on notation: simplifying characters via the bullet group

In order to simplify calculations with characters in the next sections, it will be convenient to renormalize the elements of the group G^* with a map $\mu \to \mu^\bullet$ which is defined so that the following equation holds for any $g \in G$ and $\mu \in G^*$:

$$\chi_\mu(g) = \exp\left(2\pi\mathrm{i} \sum_{i=1}^{m} \mu_i^\bullet g_i\right). \tag{2.19}$$

For this reason, we introduce a new abelian group $G^\bullet$, called the bullet group of G, which is isomorphic to $G^\bullet = G_1^\bullet \times \cdots \times G_m^\bullet$ and $\widehat{G}$, defined as

$$\begin{aligned} \mathbb{Z}_N^\bullet &:= \left\{0, \frac{1}{N}, \frac{2}{N}, \ldots, \frac{N-1}{N} \bmod 1\right\}, \\ \mathbb{R}^\bullet := \mathbb{R}^* = \mathbb{R}; \quad \mathbb{Z}^\bullet &:= \mathbb{Z}^* = \mathbb{T}; \quad \mathbb{T}^\bullet := \mathbb{T}^* = \mathbb{Z}. \end{aligned} \tag{2.20}$$

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 manifestly isomorphic via the "bullet map"

$$\mu \in G^* \to \mu^\bullet := (\mu_1^\bullet, \ldots, \mu_m^\bullet) \in G^\bullet, \tag{2.21}$$

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 $\mathbb{T}$.

[4] As mentioned earlier, in the quantum computation literature (see e.g. [55, 118, 134]) the annihilator $H^\perp$ of a subgroup H is sometimes known as the *orthogonal subgroup* of H.

2.2 Homomorphisms and matrix representations

Given two elementary abelian groups H and G, a group homomorphism from G to H is a map $\alpha : H \to G$ that fulfills $\alpha(g+h) = \alpha(g) + \alpha(h)$ for every $g, h \in H$. (In other words, α is the group-theoretic analogue of a *linear* map.) An isomorphism from G to H is an invertible group homomorphism. An automorphism of G is an isomorphism of the form $\alpha : G \to G$, i.e. from a group onto itself. The set of all automorphisms of G forms a group, called the automorphism group.

Throughout this thesis, continuous group homomorphisms between abelian groups are to be described in terms of *matrix representations*. In this section we introduce and develop these techniques.

2.2.1 Normal form of a 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 2.1.1, 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 precisely, for $g \in G_\mathbb{Z}$ consider the map

$$(g, 0, 0, 0) \in G \to \alpha(g, 0, 0, 0) \in H \tag{2.22}$$

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

$$\alpha(g) = \begin{pmatrix} \alpha_{\mathbb{Z}\mathbb{Z}}(z) + \alpha_{\mathbb{Z}\mathbb{R}}(r) + \alpha_{\mathbb{Z}F}(f) + \alpha_{\mathbb{Z}\mathbb{T}}(t) \\ \alpha_{\mathbb{R}\mathbb{Z}}(z) + \alpha_{\mathbb{R}\mathbb{R}}(r) + \alpha_{\mathbb{R}F}(f) + \alpha_{\mathbb{R}\mathbb{T}}(t) \\ \alpha_{F\mathbb{Z}}(z) + \alpha_{F\mathbb{R}}(r) + \alpha_{FF}(f) + \alpha_{F\mathbb{T}}(t) \\ \alpha_{\mathbb{T}\mathbb{Z}}(z) + \alpha_{\mathbb{T}\mathbb{R}}(r) + \alpha_{\mathbb{T}F}(f) + \alpha_{\mathbb{T}\mathbb{T}}(t) \end{pmatrix} \leftrightarrow \begin{pmatrix} \alpha_{\mathbb{Z}\mathbb{Z}} & \alpha_{\mathbb{Z}\mathbb{R}} & \alpha_{\mathbb{Z}F} & \alpha_{\mathbb{Z}\mathbb{T}} \\ \alpha_{\mathbb{R}\mathbb{Z}} & \alpha_{\mathbb{R}\mathbb{R}} & \alpha_{\mathbb{R}F} & \alpha_{\mathbb{R}\mathbb{T}} \\ \alpha_{F\mathbb{Z}} & \alpha_{F\mathbb{R}} & \alpha_{FF} & \alpha_{F\mathbb{T}} \\ \alpha_{\mathbb{T}\mathbb{Z}} & \alpha_{\mathbb{T}\mathbb{R}} & \alpha_{\mathbb{T}F} & \alpha_{\mathbb{T}\mathbb{T}} \end{pmatrix} \begin{pmatrix} z \\ r \\ f \\ t \end{pmatrix} \tag{2.23}$$

α is therefore naturally identified with the 4×4 "matrix of maps" given in the r.h.s of (2.23).

The following lemma (see e.g. [196] for a proof) shows that homomorphisms between elementary abelian groups must have a particular block structure.

Lemma 2.5 (**Homomorphism normal form**)**.** Let $\alpha : G \to H$ be a continuous group homomorphism. Then α has the following block structure

$$\alpha \leftrightarrow \begin{pmatrix} \alpha_{\mathbb{Z}\mathbb{Z}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \alpha_{\mathbb{R}\mathbb{Z}} & \alpha_{\mathbb{R}\mathbb{R}} & \boldsymbol{0} & \boldsymbol{0} \\ \alpha_{F\mathbb{Z}} & \boldsymbol{0} & \alpha_{FF} & \boldsymbol{0} \\ \alpha_{\mathbb{T}\mathbb{Z}} & \alpha_{\mathbb{T}\mathbb{R}} & \alpha_{\mathbb{T}F} & \alpha_{\mathbb{T}\mathbb{T}} \end{pmatrix} \tag{2.24}$$

where $\boldsymbol{0}$ 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.

2.2.2 Matrix representations

Definition 2.2 (**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:

$$\alpha(g) \equiv Ax \pmod{H} \quad \text{for every } g \in G \text{ and } x \in \mathbb{R}^m \text{ satisfying } x \equiv g \pmod{G} \tag{2.25}$$

Conversely, a real $n \times m$ matrix A is said to define a group homomorphism if there exists a group homomorphism α satisfying (2.25).

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[5], which is an isomorphism from G^* to $G^\bullet$. Define the diagonal $m \times m$ matrix Υ with diagonal entries defined as

$$\Upsilon(i,i) = \begin{cases} 1/N_i & \text{if } G_i = \mathbb{Z}_{N_i} \text{ for some } N_i, \\ 1 & \text{otherwise.} \end{cases} \tag{2.26}$$

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

$$\mu^\bullet \equiv \Upsilon x \pmod{G^\bullet}. \tag{2.27}$$

Note that, with the definition of Υ, equation (2.19) implies

$$\chi_\mu(g) = \exp\left(2\pi\mathrm{i} \sum_{i=1}^{m} \mu^\bullet(i) g(i)\right) = \exp\left(2\pi\mathrm{i}\, \mu^T \Upsilon g\right). \tag{2.28}$$

Looking at equation (2.27) 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[6], i.e. the group isomorphism $\mu^\bullet \to \mu \pmod{G^*}$.

The next lemma (see appendix 1.1 for a proof) summarizes some useful properties of matrix representations.

Lemma 2.6 (**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 $\beta \circ \alpha$;

[5] Strictly speaking, definition 2.2 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.

[6] 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 lemma 2.8 we characterize which matrices are valid matrix representations. Also, in section 2.4.2 we discuss the problem of computing matrix representations of group automorphisms.

(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

$$e_i = (0, \ldots, 0, 1, 0, \ldots, 0) \tag{2.29}$$

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

The last property of lemma 2.7 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 1.1.

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 2.2.

Lemma 2.8 (**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:

$$c_j A(i,j) = 0 \mod d_i, \qquad d_i^* A^*(i,j) = 0 \mod c_j^*, \tag{2.30}$$

for every $i = 1, \ldots, n$, $j = 1, \ldots, m$, and being A^* the $m \times n$ matrix defined in lemma 2.6(b). Equivalently, A must be of the form

$$A := \begin{pmatrix} A_{\mathbb{Z}\mathbb{Z}} & 0 & 0 & 0 \\ A_{\mathbb{R}\mathbb{Z}} & A_{\mathbb{R}\mathbb{R}} & 0 & 0 \\ A_{F\mathbb{Z}} & 0 & A_{FF} & 0 \\ A_{\mathbb{T}\mathbb{Z}} & A_{\mathbb{T}\mathbb{R}} & A_{\mathbb{T}F} & A_{\mathbb{T}\mathbb{T}}. \end{pmatrix} \tag{2.31}$$

with the following restrictions:

1. $A_{\mathbb{Z}\mathbb{Z}}$ and $A_{\mathbb{T}\mathbb{T}}$ are arbitrary integer matrices.
2. $A_{\mathbb{R}\mathbb{Z}}$, $A_{\mathbb{R}\mathbb{R}}$ 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
$$A(i,j) = \alpha_{i,j} \frac{d_i}{\gcd(d_i, c_j)} \tag{2.32}$$
where $\alpha_{i,j}$ can be arbitrary integers[a].
4. $A_{\mathbb{T}\mathbb{Z}}$, $A_{\mathbb{T}\mathbb{R}}$ 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[b].

[a]Since $A_{F\mathbb{Z}}$, A_{FF} multiply integer tuples and output integer tuples modulo $F = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_c}$, 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).

[b]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 $[0, 1)$.

The result is proven in appendix 1.1.

2.3 Quadratic functions

In this section we study the properties of quadratic functions over arbitrary elementary groups of the form $G = \mathbb{R}^a \times \mathbb{T}^a \times \mathbb{Z}^b \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_c}$. 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 4.1 has quantum applications, since we will show in chapter 3 that quadratic functions can be used to give a powerful normal form for stabilizer states over elementary groups.

All results in this section are proven in appendix 1.6.

2.3.1 Definitions

Let G be an elementary abelian group. Recall from chapter 1 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

$$\xi(g + h) = \xi(g)\xi(h)B(g, h) \quad \text{for all } g, h \in G. \tag{2.33}$$

In this section, we call ξ a *B-representation* if the above equation holds. 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

$$B(g, h) = \xi(g + h)\overline{\xi(g)\xi(h)} = \xi(h + g)\overline{\xi(h)\xi(g)} = B(h, g). \tag{2.34}$$

2.3.2 Normal form of bicharacters

The next lemmas characterize bicharacter functions.

Lemma 2.9 (**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

$$B(g, h) = \chi_{\beta(g)}(h) \tag{2.35}$$

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

This result generalizes lemma 5(a) in [134].The next lemma gives a explicit characterization of symmetric bicharacter functions.

Lemma 2.10 (**Normal form of a symmetric bicharacter**)**.** Let B be a symmetric bicharacter of G in the form (2.35) 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 (2.26), and $M = \Upsilon A$. Then

(a) $B(g, h) = \exp(2\pi \mathrm{i}\, g^T M h)$ 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 $x \equiv g \pmod{G}$ and $y \equiv h \pmod{G}$, then

$$B(g,h) = \exp(2\pi \mathrm{i}\, x^T M y). \tag{2.36}$$

(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).

2.3.3 Normal form of quadratic functions

Our final goal is to characterize all quadratic functions. This is achieved in theorem 2.1. To show this result a few lemmas are needed.

Lemma 2.11. 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

$$\xi_1(g) = \chi_\mu(g)\xi_2(g), \quad \text{for every } g \in G. \tag{2.37}$$

Proof. This lemma can be proven using projective representation theory [197]. Here, we give a simple alternative proof. 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 $\chi_\mu = f$:

$$f(g+h) := \frac{\xi_1(g)}{\xi_2(g)} \frac{\xi_1(h)}{\xi_2(h)} \frac{B(g,h)}{B(g,h)} = f(g)f(h). \tag{2.38}$$

□

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 2.11 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 2.10).

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

Lemma 2.12. Let B be a bicharacter B of G. Consider a symmetric real matrix M such that $B(g,h) = \exp(2\pi \mathrm{i}\, g^T M h)$. Then the following function is quadratic and a B-representation:

$$Q(g) := \mathrm{e}^{\pi \mathrm{i} \left(g^T M g + C^T g\right)}, \tag{2.39}$$

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 main result of this section.

Theorem 2.1 (Normal form of a quadratic function). Let G be an elementary abelian group. Then a function $\xi : G \to U(1)$ is quadratic if and only if

$$\xi(g) = \mathrm{e}^{\pi \mathrm{i} \left(g^T M g + C^T g + 2v^T g\right)} \tag{2.40}$$

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.

The normal form in theorem 2.1 can be very useful to perform certain calculations within the space of quadratic functions, as illustrated by the following lemma.

Lemma 2.13. Let $\xi_{M,v}$ be the quadratic function (2.40) 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 (2.40) as $\xi_{M',v'}$, with

$$M' := A^T M A, \qquad v' := A^T v + v_{A,M}, \qquad v_{A,M} := A^T C_M - C_{A^T M A}, \tag{2.41}$$

where C_M is the vector C associated with M in (2.40).

2.4 Computational group theory

Computational aspects of finite abelian groups are now discussed; our discourse focuses on a selected catalog of computational problems relevant to this chapter and efficient classical algorithms to solve them. Since this section concerns only classical computational complexity, we will tend to omit the epithet *classical* all the way throughout it.

2.4.1 Basic group operations

We begin recalling that basic arithmetical computations within groups of the form

$$G = \mathbb{R}^a \times \mathbb{T}^b \times \cdots \times \mathbb{Z}^c \times F, \qquad F := \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_d}. \tag{2.42}$$

can be efficiently performed in a classical computer [198]. From now on, the *size* $\|N\|_{\mathbf{b}}$ of an integer N is the number of bits in its binary expansion (recall that $\|N_{\max}\|_{\mathbf{b}}$ is roughly $\log|N|$ times the absolute value of N). Throughout this thesis the elements of (2.42) will always be represented as $m := a + b + c + d$ vectors of fractions $g = (g(1), \ldots, g(m)) \in G$, which can be efficiently stored in a computer: when G is finite, $O(\text{polylog}\,|G|)$ bits of memory are enough[7], in general, $O(m\,\text{poly}\,\|N_{\max}\|_{\mathbf{b}})$ bits are enough where $N_{\max}$ denotes the largest numerator/denominator in g that we need to store. The bit-size scaling of these descriptions is efficient in the size of the input. Similarly, matrix representations as in lemma 2.8 can be efficiently described in terms of rational *matrices*, instead of vectors.

We discuss now how to perform some basic operations efficiently within any finite abelian group (2.42). First, given two fractions a and b with numerators and denominators of size at most l, common arithmetic operations can be computed in $\text{poly}(l)$ time with elementary algorithms: such as their sum, product, the quotient of a divided by b, and the remainder $a \bmod b$ [198]. Therefore, given $g, h \in G$, the sum $g + h$ can be obtained in $O(m\,\text{poly}\,\|N_{\max}\|_{\mathbf{b}})$ time by computing the m remainders $g(i) + h(i) \bmod \text{char}(G_i)$, where $\text{char}(G_i)$ is the characteristic function (2.2). Similarly, given an integer n, the element ng can be obtained in $\text{polylog}(m\|N_{\max}\|_{\mathbf{b}}, \|n\|_{\mathbf{b}})$ time by computing the remainders $ng(i) \bmod d_i$.

In connection with section 2.2, it follows from the properties just introduced that matrix representations can be stored using only a polynomial amount of memory, and, moreover, that given the matrix representation A we can efficiently compute $Ah \pmod G$. Specifically, given a matrix representation A of the homomorphism $\alpha : G_1 \times \cdots \times G_m \to H_1 \times \cdots \times H_n$, we need $\text{polylog}(mn, \|N_{\max}\|_{\mathbf{b}})$ space to store its columns a_i as tuples of integers, and $\text{polylog}(mn, \|N_{\max}\|_{\mathbf{b}})$ time to compute the function $\alpha(h)$.

[7] This follows from the inequalities $2^m \leq |G|$ and $N_i \leq |G|$.

2.4.2 Systems of linear equations over abelian groups

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

$$\alpha(x) \equiv Ax \equiv b \pmod{H}, \quad \text{where } x \in G, \tag{2.43}$$

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,

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \quad \mathbf{A} \in \mathbb{R}^{n\times m}, \mathbf{x} \in \mathbb{R}^m, \mathbf{b} \in \mathbb{R}^n, \tag{2.44}$$

as well as systems of linear equations over other types of vector spaces, such as $\mathbb{Z}_2^n$, e.g.

$$\mathbf{B}\mathbf{y} = \mathbf{c}, \quad \mathbf{B} \in \mathbb{Z}_2^{n\times m}, \mathbf{y} \in \mathbb{Z}_2^m, \mathbf{c} \in \mathbb{Z}_2^n. \tag{2.45}$$

In (2.44) 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 (2.45) is a linear map between $\mathbb{Z}_2$ vector spaces.

We dub systems (2.43) "linear" to highlight this resemblance. Yet the reader must beware that, in general, the groups G and H in problem (2.43) 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[8]), 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 (2.43) and that are not systems of linear equations over vectors spaces. For infinite groups, an example are systems of mixed real-integer linear equations [199, 200], which we introduce in equation (2.47) in this section. Furthermore, in the next chapters, we will encounter a wide range of computational problems directly related to simulating normalizer circuits that can be reduced to linear systems over abelian groups (cf. lemma 3.1 in chapter 3): hence, the techniques developed in this section will be useful throughout the thesis.

Input of the problem. In this thesis, we only consider systems of the form (2.43) 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 proofs.[9]

General solutions of system (2.44) Since A is a homomorphism, it follows that the set G_{sol} of all solutions of (2.43) is either empty or a coset of the kernel of A:

$$G_{\text{sol}} = x_0 + \ker A \tag{2.46}$$

The main purpose of this section is to devise efficient algorithms to solve system (2.43) when A, b are given as input, in the following sense: we say that we have *solved* system (2.43) if we manage to find a *general solution* of (2.43) as defined next.

[8] Note that $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ 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.

[9] Of course, not all group homomorphisms have rational matrix representations (cf. lemma 2.8). However, for the applications we are interested (cf. chapter 0-5) it is enough to study this subclass.

Definition 2.3 (General solution of system (2.43)). A *general solution* of a system of equations $Ax \equiv b \pmod{H}$ as in (2.43) is a pair $(x_0, \mathcal{E})$ 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 solvable systems of the form (2.43) *always* exist. This is shown in appendix 1.2.

A main contribution of this chapter is a deterministic classical algorithm that finds a general solution of any system of the form (2.43) in polynomial time. This is the content of the next theorem, which is one of our main technical results.

Theorem 2.2 (Classical algorithms for linear systems over groups (2.43)). Let A, b define a system of linear equations (over elementary abelian groups) of form (2.43), 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 H_j. Then there exist efficient, deterministic, exact classical algorithms to solve the following tasks in $O\left(\text{poly}(m, n, \|A\|_{\mathbf{b}}, \|b\|_{\mathbf{b}}, \log c_i, \log d_j)\right)$ time:

1. Deciding whether system (2.43) admits a solution.
2. Finding a general solution $(x_0, \mathcal{E})$ of (2.43).
3. Simplifying "discrete" solutions: given a finitely generated G and a solution $(x_0, \mathcal{E})$ where $\mathcal{E}$ acts on $\mathcal{X} = \mathbb{Z}^{\alpha+\beta}$[a]; find $\{Q, \mathcal{E}_{\text{iso}}\}$ such that (a) Q is an elementary group isomorphic to the quotient $\mathcal{X}/\ker\mathcal{E}$ and (b) $\mathcal{E}_{\text{iso}}$ is a matrix representation of the isomorphism $Q \xrightarrow{\mathcal{E}} \text{im}\mathcal{E}$.
4. If G is finite, counting the number of solutions of (2.43) and finding $x_1, \ldots, x_r \in G$ such that all solutions of the system are linear combinations of the form $x_0 + \sum k_i x_i$.

[a] Note that $\mathcal{X}$ needs to be of form $\mathcal{X} = \mathbb{Z}^{\alpha+\beta}$ for discrete G (lemma 2.8).

A rigorous proof of this theorem is given in appendix 1.3. Below, we sketch the key ideas behind our algorithms for task (1-2); our algorithms for tasks (3-4) crucially combine the former ones with fast classical methods to compute Smith normal forms [201].

In short, for tasks (1-2), we show that the problem of finding a general solution of a system of the form (2.43) reduces in polynomial time to the problem of finding a general solution of a so-called *system of mixed real-integer linear equations* [199].

$$A'x' + B'y' = c, \quad \text{where } x' \in \mathbb{Z}^a, y' \in \mathbb{R}^b, \tag{2.47}$$

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 (2.47) play an important role within the class of problems (2.43), 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 2.2 is to apply an existing (deterministic) algorithm by Bowman and Burdet [199] that computes a general solution to a system of the form (2.47). Although Bowman and Burdet did not prove the efficiency of their algorithm in [199], we show in appendix 1.4 that it can be implemented in polynomial-time, completing the proof of the theorem.

Application of theorem 2.2: computing inverses of group automorphisms

In section 2.2.2 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 2.2 can be adapted to solve this problem.

Lemma 2.14. Let $\alpha : G \to G$ 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 1.5.

Chapter 3

Classical simulations of normalizer circuits over finite abelian groups

In chapter 1 we introduced *normalizer circuits over finite abelian groups* as group theoretic generalizations of Clifford circuits composed of quantum Fourier transforms, automorphism gates and quadratic-phase gates (cf. chapters 1-1.3). An interesting feature of this circuit families is the presence of QFTs over finite abelian groups, which are central in Shor's factoring algorithm [4] and in quantum algorithms for abelian hidden subgroups problems [118, 119, 7].

Normalizer circuits over finite abelian groups were first studied by Van den Nest in [134], who proved that the action of such circuits on computational basis states and followed by computational basis measurements can be simulated classically efficiently. In this section, we generalize the result in [134] in several ways. Most importantly, we show that normalizer circuits supplemented with *intermediate measurements* of arbitrary (generalized) Pauli operators can also be simulated efficiently classically, even when the computation proceeds *adaptively*. This yields a generalization of the Gottesman-Knill theorem (valid for n-qubit Clifford operations [1, 2]) to quantum circuits described by arbitrary finite abelian groups. Moreover, our simulations are twofold: we present efficient classical algorithms to (a) sample the measurement probability distribution of any adaptive-normalizer computation, as well as (b) to compute the amplitudes of the state vector in every step of it.

Finally we develop a generalization of the *stabilizer formalism* [1, 2] relative to arbitrary finite abelian groups: for example we characterize how to update stabilizers under generalized Pauli measurements and provide a normal form of the amplitudes of generalized stabilizer states using quadratic functions and subgroup cosets.

The results in this chapter, together with [134]'s identify a large family of (arbitrarily long) quantum computations that *cannot* yield exponential speed-ups in spite of usage of the QFT. In chapter 4 we will show that many of our results can even be generalized to an infinite-dimensional setting. In the second part of this thesis (cf. chapter 5), the techniques developed in these first chapters will help us to identify and analyze more powerful models of normalizer gates that achieve exponential quantum speedups

This chapter is based on [63] (joint work with Maarten Van den Nest).

3.1 Introduction

In the circuit model considered in [134] the allowed operations are normalizer gates over a finite abelian group $G = \mathbb{Z}_{D_1} \times \ldots \times \mathbb{Z}_{D_m}$ (cf. chapter 1.3) supplemented with standard basis states and terminal measurements in the standard basis. The main result in [134] states that any such circuit is efficiently classically simulable for any group G. The main contribution of this

chapter is a generalization of the result of [134] where *intermediate measurements* are allowed at arbitrary times in the computation. This extension recovers a missing feature that was present in the original [1, 2] Gottesman-Knill theorem, which states that intermediate measurements of Pauli operators interspersed along a Clifford circuit can also be classically simulated, and even if operations are chosen adaptively.

Specifically, in this work, we define *adaptive normalizer circuits* over G to comprise the following three fundamental ingredients:

- **Normalizer gates over** G, i.e. QFTs, automorphism gates, quadratic phase gates.
- **Measurements** of generalized Pauli operators over G at any time of the computation.
- **Adaptiveness:** the choice of normalizer gate at any time may depend (in a polynomial-time computable way) on the outcomes obtained in all previous measurement rounds.

If G is chosen to be $\mathbb{Z}_2^n$, the corresponding class of adaptive normalizer circuits precisely corresponds to the class of adaptive Clifford circuits allowed in the original Gottesman-Knill theorem.

3.1.1 Main results

This chapter contains several results, summarized as follows:

I. **A Gottesman-Knill theorem for all finite abelian groups** (Theorem 3.7). Given any abelian group G, every poly-size adaptive normalizer circuit over G, acting on any standard basis input, can be efficiently simulated by a classical computer. That is, we show that the conditional probability distribution arising at each measurement (given the outcomes of the previous ones) can be sampled in classical polynomial time.

II. **A stabilizer formalism for finite abelian groups**. Generalizing the well-known stabilizer formalism for qubits, we develop a stabilizer formalism for arbitrary abelian groups. This framework is a key ingredient to efficiently track the evolution of quantum states under normalizer circuits. In particular, our results are:

   – We provide an analytic formula, as well as an efficient algorithm, to compute the dimension of any stabilizer code over a finite abelian group (Theorem 3.2).

   – We provide an analytic formula, as well as an efficient algorithm, to compute the update of any stabilizer group under Pauli measurements over arbitrary finite abelian groups (Theorem 3.5).

III. **A normal form for stabilizer states** (Theorem 3.4). We give an analytic formula to characterize the amplitudes of stabilizer states over abelian groups and show how to compute these amplitudes efficiently. It follows that all stabilizer states over abelian groups belong to the class of Computationally Tractable (CT) states, introduced in [170]. The interest in this property is that all CT states can be simulated classically in various contexts well beyond the setting of the present work—cf. [170] for a discussion.

In all the results above the term *efficient* is used as synonym of "in polynomial time in $\log|G|$" (where $|G|$ denotes the cardinality of the group G). All algorithms presented show good performance regarding computational errors: the sampling algorithm given in theorem 3.7 is *exact* (i.e. it samples the output probability of the adaptive normalizer circuit *exactly* in polynomial

time[1]), whereas the algorithms in theorem 3.4 yield *exponentially* accurate estimates of state amplitudes and normalization constants.

3.1.2 Discussion

Technical aspects of the results.

An important technical difference (and difficulty) of our setting compared to the Gottesman-Knill theorem qubit one is that in the context of arbitrary finite abelian groups (such as $G = \mathbb{Z}_{2^n}$) arithmetic is generally over large integers. This is in contrast to $\mathbb{Z}_2^n$ where arithmetic is simply over $\mathbb{Z}_2$ i.e. modulo 2. The difference is in fact twofold:

• First, $\mathbb{Z}_2$ is a **field**. As a result, it is possible to describe the "standard" stabilizer formalism for qubits with vector space techniques over $\mathbb{Z}_2$. In this context methods like Gaussian elimination have straightforward analogues, which can be exploited in the design of classical algorithms. General abelian groups are however no longer fields. This complicates both the analytic and algorithmic aspects of our abelian-group stabilizer formalism due to, for instance, the presence of zero divisors.

• Second, in $\mathbb{Z}_2$ arithmetic is with small numbers (namely 0s and 1s), whereas in general finite abelian groups arithmetic is with **large integers**. For example, this is the case with $G = \mathbb{Z}_{2^n}$. Of course, one must beware that some problems in number theory are widely believed to be *intractable* for classical computers: consider, for instance, the integer factorization problem or computing discrete logarithms. One of the main challenges in our scenario is to show that the "integer arithmetic" used in our classical simulation algorithms can be carried out efficiently. For this purpose, a significant technical portion of our work is dedicated to solving *systems of linear equations modulo a finite abelian group*, defined as follows: given a pair of finite abelian groups G_{sol} and G (both of which are given as a direct product of cyclic groups), and a homomorphism α between them, we look at systems of the form $\alpha(x) = b$ where $x \in G_{sol}$ and $b \in G$. We present polynomial-time deterministic classical algorithms for counting and finding solutions of these systems. These efficient algorithms lie at the core of our classical simulations of normalizer circuits.

The power of adaptiveness.

Another interesting feature in our work compared to the qubit setting, is that abelian-group normalizer circuits with intermediate measurements are more powerful than those with only terminal measurements for quantum state preparation: in section 3.7.2, we show that certain families of abelian group stabilizer states (namely, abelian-group coset states) can only be prepared if intermediate measurement is allowed[2]. Albeit, despite displaying superior QIP features, our main simulation result says that enhanced normalizer circuits with intermediate measurements can still not outperform classical computers.

[1] In our model, for simplicity we assume availability of a subroutine which allows to generate, with zero error, a uniformly random integer in the interval $[0, N]$ in polylog(N) time, for any integer N. Under this assumption, our classical sampling algorithm for simulating normalizer circuits also has perfect accuracy i.e. no additional errors are introduced.

[2] This is analogous to a known feature of stabilizer states in composite qudit dimensions, some of which cannot be prepared without measuring Paulis, *even though* mere terminal measurements are enough in the qubit setting (cf. discussion in section 3.7.2).

Applications.

Finally, we recall that the stabilizer formalism has been used in a variety of settings (both for qubits and d-level systems) beyond the context of the Gottesman-Knill theorem. This includes e.g. measurement-based quantum computation [42, 65, 66], quantum error-correction and fault-tolerance [3, 43, 67–71], secret-sharing [72–74], topological systems [75–78], quantum computation via state injection (rebits [47, 48, 44]) and other applications. The mathematical tools developed in the present work may therefore also have applications outside the realm of classical simulations of quantum circuits.

3.1.3 Relationship to previous work

In [134] it was proven that one can sample classically in poly-time the output distribution of any *non-adaptive* normalizer circuit followed by a terminal measurement in the standard basis. Our work extends this result in various ways, as outlined above in I-II-III. Main differences are the fact that here we consider adaptive normalizer circuits, and two different types of simulations: sampling output distributions and computation of amplitudes.

To our knowledge, ref. [134] and the present work are the first studies to investigate normalizer circuits over arbitrary finite abelian groups, including those of the form $G = \mathbb{Z}_d^m$ where d can be an *exponentially* large number, such as $d = 2^n$; they are also the first to consider normalizer operations that act on high-dimensional physical systems without a natural tensor product decomposition (such as $\mathbb{C}^p$ where $p > 2^n$ is an exponentially big prime number), or clusters of heterogeneous qudits (e.g. $\mathbb{C}^a \times \mathbb{C}^b \times \mathbb{C}^c$ when a, b, c are different, as opposed to $\mathbb{C}^{d^{\otimes n}}$).

Restricting to groups of the form $G = \mathbb{Z}_d^m$ where d is *constant*, our work recovers previous results regarding classical simulations of Clifford circuits for *qudits*. We emphasize that in this second scenario d is a *fixed* parameter that does not scale; this is in contrast with the cases studied in [134] and in the present paper. We briefly summarize prior work on qudits.

- Results when d is a **constant prime** number: if $d = 2$, the ability to *sample* classically efficiently follows from the Gottesman-Knill theorem [1, 2], whereas the computation of *amplitudes* from [135]; for prime values of d larger than 2, techniques given in [3] yield efficient sampling simulations also for adaptive Clifford circuits.

- Results when d is an **arbitrary constant**: techniques given in [136] can be used to simulate *non*-adaptive Clifford circuits followed by a terminal standard basis measurement (sampling output distributions and computation of amplitudes); tools developed in [137] can be used to sample in the adaptive case.

Finally, our work also connects to previous studies on the simulability of abelian quantum Fourier transforms (QFTs) [166–168, 134]. In [166] it was shown that the action of the approximate QFT over $\mathbb{Z}_{2^n}$ on product states, followed by a terminal measurement in a product basis can be classically simulated in *quasi*-polynomial $O(n^{\log n})$ time. This result was improved in [167] where fully efficient classical simulation algorithms were given for this setting and, more generally, for constant-depth circuits of bounded interaction range, interspersed with a constant amount of approximate QFTs. In [168] it was shown that the "semi-classical" QFT acting on a class of entangled input states can be efficiently classically simulated. Finally, Van den Nest [134] gave efficient classical algorithms for circuits of arbitrary size containing QFTs and normalizer gates.

A common ingredient in works [166–168] is that they all employ tensor contraction schemes in their simulations, which crucially depend on the geometric *structure* of the quantum circuit:

in particular, all these methods can only be efficient if the graph representing the gate structure of the circuit has a strong tree-like structure (measured by the tree-width [165]). Unlike the simulations given in [134] and in the present work, the circuits in [166–168] can only generate limited amounts of entanglement [169]. Also, the simulations in [134] and in the present work are fully *independent* of the structure of the circuit. This generality comes at the cost that we have to restrict our allowed gates, similarly to the original Gottesman-Knill theorem.

3.1.4 Chapter outline

We refer the reader to chapter 1-1.3 for an introduction to the normalizer circuit model of this chapter, its relationship to the standard Pauli and Clifford operations and definitions of character, quadratic and bicharacter functions. The rest of the chapter is organized as follows.

Sections 3.2 and 3.3 contain technical preliminaries. Section 3.2 presents a number of efficient classical algorithms to solve algebraic computational problems based on the classical techniques we developed in chapter 2. Section 3.3 gives a detailed account of the mathematical properties of Pauli, Clifford and (unitary) normalizer operations.

The remaining sections contain the main results of our work. In section 3.4, a theory of abelian-group stabilizer codes is developed. In section 3.5 we give normal forms for stabilizer states. Section 3.6.2 explains how intermediate (generalized) Pauli operator measurements can be implemented, and how they transform abelian-group stabilizer states. In section 3.7 we show how to simulate adaptive normalizer circuits classically and discuss the power of these operations for state preparation.

3.2 Preliminaries on finite abelian groups

Conventions and methodology: Throughout this section we fix the group G to be of the form

$$G = \mathbb{Z}_{d_1} \times \ldots \times \mathbb{Z}_{d_m}, \tag{3.1}$$

with parameters d_i, m chosen arbitrarily. The elements and *canonical generators*[3] of G are denoted by $g = (g(1), \ldots, g(m)), g(i) \in \mathbb{Z}_{d_i}$, resp., $e_i = (0, \ldots, 1_i, \ldots, 0)$. The m elements e_i generate G and for any $g \in G$ can be naturally written as $g = \sum g(i)e_i$. Throughout the section, we use the shorthand $(\mathrm{mod} G)$ as synonym of $(\mathrm{mod} d_1, \ldots, \mathrm{mod} d_m)$.

The classical simulation and stabilizer formalism methods in this chapter exploit several of the classical group theoretic techniques that we develop in chapter 2. In particular, we will apply the notions and main properties of annihilator subgroup $H^\perp$ (section 2.1.4), and character $\chi_g(h)$, quadratic $\xi(h)$ and bicharacter $B(g, h)$ functions (chapters 1, 2.1, 2.3) to devise analytic tools to describe stabilizer states and codes. Furthermore, the theory of matrix representations for group homomorphisms that we developed in chapter will be useful to characterize normalizer gates and various linear structures present in the problems we study.

Computational group theory: Computational aspects of finite abelian groups are now discussed; our discourse focuses on a selected catalog of computational problems relevant to this chapter and efficient classical algorithms to solve them.

In computational complexity theory a (classical or quantum) algorithm is said to be *efficient* if it solves a given computational problem of input-size n in (classical or quantum) poly (n) time: when one looks at problems related to finite abelian groups, this will be synonym of

[3] These elements play a similar role as the canonical basis vectors of vector spaces like $\mathbb{R}^m$ or $\mathbb{C}^m$ (though G is *not* a vector space).

"in polylog($|G_1|, |G_2|, \ldots, |G_n|$) time", being $G_1, \ldots, G_n$ the groups involved in a problem of interest. Since this section concerns only classical computational complexity, we will tend to omit the epithet *classical* all the way throughout it.

Periodically, and at crucial stages of this chapter, some advanced algebraic computational problems are bound to arise. The following lemma compiles a list of group theoretical problems that will be relevant to us and can be solved efficiently by classical computers.

Lemma 3.1 (**Algorithms for finite abelian groups**). Given H, K, two subgroups of G, and $\{h_i\}$, $\{k_j\}$, polynomial-size generating-sets of them, there exist efficient classical algorithms to solve the following problems deterministically.

(a) Decide whether $b \in G$ belongs to H; if so, find integers w_i such that $b = \sum w_i h_i$.
(b) Count the number of elements of H.
(c) Find a generating-set of the intersection $H \cap K$.
(d) Find a generating-set of the annihilator subgroup $H^\perp$ (cf. definition in chapter 2.1.4).
(e) Given the system of equations $\chi_{h_i}(g) = \gamma^{a_i}$, find elements $(g_0, g_1, \ldots, g_s)$ such that all solutions can be written as linear combinations of the form $g_0 + \sum v_i g_i$.
(h) Find a $r \times m$ matrix representation Ω of a homomorphism $\varpi : G \to \mathbb{Z}_d^r$ such that H coincides with the kernel of ϖ and r, d have polynomial bit-size.

The proof of the lemma is given in appendix 2.1, where we prove the following statement.

Lemma 3.2. Problems (a-e) in lemma 3.1 are polynomial-time reducible to either counting or finding solutions of systems of equations of the form $\alpha(x) = Ax = b$; where α is a group homomorphism between two (canonically-decomposed) finite abelian groups, $\mathbf{G}_{sol}$ and $\mathbf{G}$, to which x, b respectively belong and A is a matrix representation of α.

We recall that the system of equation in lemma 3.2 is a *linear system over groups* in the sense of (chapter 2.4.2), which can be solved efficiently with our classical algorithm in theorem 2.2: the latter may be applied to count solutions and/or output an element x_0 and a poly-size generating set of $\ker \alpha$ such that $X_{sol} = x_0 + \ker \alpha$ is the total number of solutions of the system.

3.3 Pauli operators and normalizer circuits over abelian groups

3.3.1 Definitions and terminology

We recall (section 1.3) that a generalized Pauli operator over G (hereafter often simply denoted *Pauli operator*) is any unitary operator of the form

$$\sigma(a, g, h) := \gamma^a Z(g) X(h), \quad X(g) := \sum_{h \in G} |h + g\rangle\langle h|, \quad Z(g) := \sum_{h \in G} \chi_g(h) |h\rangle\langle h| \tag{3.2}$$

where χ_g is a character, $\gamma := \mathrm{e}^{\mathrm{i}\pi/|G|}$ is a primitive root of unity, and $a \in \mathbb{Z}_{2|G|}$. Throughout this chapter, the triple (a, g, h) describing the Pauli operator is called the *label* of σ. It is important to observe that, although σ is a $|G| \times |G|$ matrix, its label (a, g, h) is an *efficient* description of itself comprising $O(\log |G|)$ bits; from now on, we will specify Pauli operators in terms of their labels, and refer to the latter as the *standard encoding* of these operators.

3.3.2 Manipulation of Pauli operators

First, note that every Pauli operator factorizes as a tensor product relative to the tensor decomposition of $\mathcal{H}_G$ i.e. σ can be written as $\sigma = U_1 \otimes \cdots \otimes U_m$ where U_i acts on $\mathbb{C}^{d_i}$. This

property simplifies several proofs; it can be verified straightforwardly by applying (3.2) and the definition (1.6) of the characters of G.

Basic manipulations of Pauli operators can be carried out transparently by translating them into transformations of their labels: we review now some of these rules. First, the Pauli matrices (3.2) obey the following commutation rules:

$$\begin{aligned} X(g)X(h) &= X(g+h) = X(h)X(g) \\ Z(g)Z(h) &= Z(g+h) = Z(h)Z(g) \\ Z(g)X(h) &= \chi_g(h)X(h)Z(g). \end{aligned} \tag{3.3}$$

Combinations of these rules straightforwardly lead to the next two lemmas.

Lemma 3.3 (**Products and powers of Pauli operators** [134])**.** Consider Pauli operators σ and τ and a positive integer n. Then $\sigma\tau$, σ^n and $\sigma^\dagger$ are also Pauli operators, the labels of which can be computed in $\mathrm{polylog}(|G|, n)$ time on input of n and the labels of σ and τ. Moreover, $\sigma^\dagger = \sigma^{2|G|-1}$.

Lemma 3.4 (**Commutativity**)**.** Consider two Pauli operators $\sigma(a_1, g_1, h_1) = \sigma_1$ and $\sigma(a_2, g_2, h_2) = \sigma_2$. Then the following statements are equivalent:

(i) σ_1 and σ_2 commute;
(ii) $\chi_{g_1}(h_2) = \chi_{g_2}(h_1)$;
(iii) $x := (g_1, h_1)$ and $y := (h_2, -g_2)$ annihilate each other as elements of $G \times G$: i.e. $\chi_x(y) = 1$.

Lemma 3.3 implies that the set of all Pauli operators $\mathcal{P}_G$ over G forms a (finite) group, called the Pauli group (over G).

3.3.3 Normalizer quantum circuits

Hitherto we have not considered technical aspects of normalizer circuits, such as how to describe normalizer circuits efficiently, or how to compute their action on Pauli operators; we address these questions in this section.

3.3.3.1 Describing normalizer operations

In this chapter we will be interested in classical simulations of *normalizer circuits*. To make meaningful statements about classical simulations one must first specify which *classical descriptions* of normalizer circuits are considered to be available. In the case of Pauli operators over G, we saw in the previous section that it is possible to describe them using few ($\mathrm{polylog}\,|G|$) memory resources, by choosing their labels (a, g, h) as standard encodings; this property holds for *all* normalizer gates and—hence—circuits [134]: all of them admit efficient classical descriptions. This is discussed next.

- First, a partial quantum Fourier transform is described by the set of systems $\mathcal{H}_{\mathbb{Z}_{d_i}}$ on which it acts non-trivially
- Second, an automorphism gate is described by the *matrix representation* of the associated automorphism (cf. chapter 2.2.2).
- Third, let ξ be an arbitrary quadratic function. Then, it follows from our normal form for quadratic functions (theorem 2.1) that there exists $n(g) \in \mathbb{Z}_{2|G|}$ such that $\xi(g) = \mathrm{e}^{\pi i n(g)/|G|}$

for every $g \in G$; furthermore, the $O(m^2)$ integers $n(e_i)$ and $n(e_i + e_j)$ comprise an efficient description of ξ and, thus, of the associated quadratic phase gate.[4]

Henceforth we will assume that all normalizer gates are specified in terms of the descriptions given above, which will be called their *standard encodings*. The standard encoding of each type of gate comprises polylog($|G|$) bits. The standard encoding of a normalizer circuits is the sequence of classical descriptions of its gates.

3.3.3.2 Normalizer vs Clifford

The following theorem from [134] states that every normalizer gate belongs to the Clifford group, and the action of any normalizer gate on a Pauli operator via conjugation can be described efficiently classically.

Theorem 3.1 (Normalizer gates are Clifford [134]). Every normalizer gate is a Clifford operator. Furthermore let U be a normalizer gate specified in terms of its standard classical encoding as above, and let σ be a Pauli operator specified in terms of its label; then the label of $U\sigma U^\dagger$ can be computed in polylog $|G|$ time.

Proof. We do not reproduce the original proof of this theorem since we present an infinite-dimensional generalization of it in chapter 4 (theorem 4.2). However, we illustrate here how the main types of normalizer gates $\mathcal{F}_G$, U_α, D_ξ act on Pauli operators $X(g)$, $Z(g)$ under conjugation:

$$\begin{array}{lcll} \mathcal{F}_G & : & X(g) \to Z(g); & Z(g) \to X(-g) \\ U_\alpha & : & X(g) \to X(\alpha(g)); & Z(g) \to Z(\alpha^{-*}(g)) \\ D_\xi & : & X(g) \to \xi(g)X(g)Z(\beta(g)); & Z(g) \to Z(g) \end{array} \tag{3.4}$$

Above, $\beta : G \to G$ denotes the homomorphism in lemma 2.9, α^* is the *dual group automorphism* of α (2.16); and α^{-*} denotes the inverse of α^*. □

It is unknown whether the entire Clifford group can be generated (up to global phase factors) by normalizer gates in full generality. However, it was proven in [136] (see also the examples in section 1.3.1) that this is indeed the case for groups of the form $G = \mathbb{Z}_d^m$ (i.e. m qudit systems); more strongly, every Clifford group element (over $\mathbb{Z}_d^m$) can be written as a product of at most polylog($|G|$) such operators. We *conjecture* that this feature holds true for Clifford operators over *arbitrary* finite abelian groups.

Conjecture 3.1. Let G be an arbitrary (canonically decomposed) finite abelian group. Then, up to a global phase, every Clifford operator over G can be written as a product of polylog $|G|$ normalizer gates.

Finally, in the following lemma we provide some partial support for this conjecture. We show that both automorphism gates and quadratic phase gates have a distinguished role within the Clifford group, characterized as follows:

Lemma 3.5. Up to a global phase, every Clifford operator which acts on the standard basis as a permutation has the form $X(g)U_\alpha$ for some $g \in G$ and some automorphism gate U_α. Every diagonal Clifford operator is, up to a global phase, a quadratic phase gate.

[4] Given these integers one can efficiently compute M and vector v as in (2.40) (cf. appendix 3.3.3): it follows that ξ can be efficiently computed given these numbers (see also [134] for an earlier proof of this fact).

Proof. The first statement was proved in [134]. We prove the second statement. Let $D = \sum \xi(g)|g\rangle\langle g|$ be a diagonal unitary operator (so that $|\xi(g)| = 1$ for all $g \in G$) in the Clifford group. Without loss of generality we may set $\xi(0) = 1$, which can always be ensured by choosing a suitable (irrelevant) overall phase. Then for every $h \in G$, D sends $X(h)$ to a Pauli operator under conjugation. This implies that there exists a complex phase $\gamma(h)$ and group elements $f_1(h), f_2(h) \in G$ such that

$$DX(h)D^\dagger = \gamma(h)X(f_1(h))Z(f_2(h)). \tag{3.5}$$

Since D is diagonal, it is easy to verify that we must have $f_1(h) = h$ for every $h \in G$. Now consider an arbitrary $g \in G$. Then

$$DX(h)D^\dagger|g\rangle = \overline{\xi}(g)\xi(g+h)|g+h\rangle; \tag{3.6}$$
$$\gamma(h)X(h)Z(f_2(h))|g\rangle = \gamma(h)\chi_g(f_2(h))|g+h\rangle. \tag{3.7}$$

Condition (3.5) implies that (3.6) is identical to (3.7) for every $g, h \in G$. Choosing $g = 0$ and using that $\xi(0) = 1$ and $\chi_0(x) = 1$ for every $x \in G$ it follows that $\gamma(h) = \xi(h)$. We thus find that

$$\xi(g+h) = \xi(g)\xi(h)\chi_g(f_2(h)). \tag{3.8}$$

The function $B(g,h) := \xi(g+h)\overline{\xi}(g)\overline{\xi}(h)$ is manifestly linear in g, since $B(g,h) = \chi_g(f_2(h))$. Furthermore by definition B is symmetric in g and h. Thus B is also linear in h. □

3.4 An abelian Group Stabilizer Formalism

In this section we develop further the stabilizer formalism for finite abelian groups as started in [134]. We provide new analytic and algorithmic tools to describe them and analyze their properties. Throughout this section we consider an arbitrary abelian group of the form $G = \mathbb{Z}_{d_1} \times \ldots \times \mathbb{Z}_{d_m}$.

3.4.1 Stabilizer states and codes

Let $\mathcal{S}$ be a subgroup of the Pauli group $\mathcal{P}_G$. Then $\mathcal{S}$ is said to be a stabilizer group (over G) if there exists a non-zero vector $|\psi\rangle \in \mathcal{H}_G$ which is invariant under all elements in $\mathcal{S}$ i.e. $\sigma|\psi\rangle = |\psi\rangle$ for every $\sigma \in \mathcal{S}$. The linear subspace $\mathcal{V} := \{|\psi\rangle : \sigma|\psi\rangle = |\psi\rangle \text{ for all } \sigma \in \mathcal{S}\}$ is called the stabilizer code associated with $\mathcal{S}$. If $\mathcal{V}$ is one-dimensional, its unique element (up to a multiplicative constant) is called the stabilizer state associated with $\mathcal{S}$. In this chapter we will mainly be interested in stabilizer states. Occasionally, however, it will be useful to consider the general setting of stabilizer codes (cf. e.g. theorem 3.2).

Note that every stabilizer group $\mathcal{S}$ is abelian. To see this, consider a state $|\psi\rangle \neq 0$ which is invariant under the action of all elements in $\mathcal{S}$ and consider two arbitrary $\sigma, \tau \in \mathcal{S}$. Then (3.3) implies that there exists a complex phase α such that $\sigma\tau = \alpha\tau\sigma$. It follows that $|\psi\rangle = \sigma\tau|\psi\rangle = \alpha\tau\sigma|\psi\rangle = \alpha|\psi\rangle$, where we have used that $\sigma|\psi\rangle = |\psi\rangle = \tau|\psi\rangle$. We thus find that $|\psi\rangle = \alpha|\psi\rangle$ so that $\alpha = 1$ (i.e. σ and τ commute).

On the other hand, not every abelian subgroup of the Pauli group is a stabilizer group. A simple counterexample is the group $\{I, -I\}$ where I is the identity operator acting on $\mathcal{H}_G$.

The support of a stabilizer code $\mathcal{V}$ is the set of all $g \in G$ for which $|g\rangle$ has a nonzero overlap with $\mathcal{V}$ i.e. there exists $|\psi\rangle \in \mathcal{V}$ such that $\langle g|\psi\rangle \neq 0$. The support of a stabilizer state $|\phi\rangle$ is simply the set of all $g \in G$ for which $\langle g|\phi\rangle \neq 0$.

3.4.2 Label groups

Let $\mathcal{S}$ be a stabilizer group over G. The diagonal subgroup $\mathcal{D}$ is the subgroup of $\mathcal{S}$ formed by its diagonal operators i.e. it consists of all operators in $\mathcal{S}$ of the form $\gamma^a Z(g)$. Second, we introduce two subgroups $\mathbb{H}$ and $\mathbb{D}$ of G called the label groups of $\mathcal{S}$:

$$\mathbb{H} = \{h \in G : \text{ there exists } \gamma^a Z(g)X(h) \in \mathcal{S}\}, \tag{3.9}$$

$$\mathbb{D} = \{g \in G : \text{ there exists } \gamma^a Z(g) \in \mathcal{D}\}, \tag{3.10}$$

Using (3.3) it is straightforward to verify that $\mathbb{D}$ is indeed a subgroup of G. To prove that $\mathbb{H}$ is a subgroup as well, one argues as follows. Let σ be a Pauli operator with label (a, g, h). We call g the "Z-component" and h the "X-component" of σ. Denote the X-component formally by $\varphi(\sigma) := h$. Then $\mathbb{H}$ is the image of $\mathcal{S}$ under the map φ. The commutation relations (3.3) yield

$$\varphi(\sigma\tau) = \varphi(\sigma) + \varphi(\tau) \quad \text{for all } \sigma, \tau \in \mathcal{S}. \tag{3.11}$$

This implies that φ is a homomorphism from $\mathcal{S}$ to G. It follows that $\mathbb{H}$ is a subgroup of G.

Lemma 3.6 (Label groups). Let $\mathcal{S}$ be a stabilizer group and assume that the labels of $k = \text{polylog}\,|G|$ generators of $\mathcal{S}$ are given as an input. Then the label groups of $\mathcal{S}$ fulfill:

(i) $\mathbb{H} \subseteq \mathbb{D}^\perp$, where $\mathbb{D}^\perp$ denotes the annihilator of $\mathbb{D}$ (section 2.1.4);
(ii) Generating sets of $\mathbb{H}$, $\mathbb{D}$ can be efficiently computed classically;
(iii) The labels of a generating set of $\mathcal{D}$ can be efficiently computed classically.

Proof. Property (i) is a straightforward consequence of the commutation relations given in lemma 3.4 and the definition of annihilator subgroup (2.17). To show property (ii), recall that the map φ defined above is a homomorphism from $\mathcal{S}$ to G with $\mathbb{H} = \text{Im}(\varphi)$. Suppose that $\mathcal{S}$ is generated by $\{\sigma_1, \ldots, \sigma_k\}$. Then $\mathbb{H}$ is generated by $\{\varphi(\sigma_1), \ldots, \sigma(\sigma_k)\}$: this yields an efficient method to compute generators of $\mathbb{H}$. To prove the second statement of (ii) as well as (iii) requires more work. The argument is a direct generalization of the proof of lemma 9 in [134] and the reader is referred to this work. □

3.4.3 Certificates

The main purpose of this section is to provide a criterion to verify when a stabilizer group gives rise to a one-dimensional stabilizer code i.e. a stabilizer state. This is accomplished in corollary 3.1. To arrive at this statement we first analyze how the dimension of a general stabilizer code is related the structure of its stabilizer group.

Theorem 3.2 (Structure Test). Let $\mathcal{S}$ be a stabilizer group with stabilizer code $\mathcal{V}$ and $\mathbb{D}^\perp$ be the annihilator of the label subgroup $\mathbb{D}$ (section 2.1.4). Then, there exists $g_0 \in G$ such that

$$(i)\ \text{supp}(\mathcal{V}) = g_0 + \mathbb{D}^\perp, \qquad (ii)\ \dim(\mathcal{V}) = \frac{|\mathbb{D}^\perp|}{|\mathbb{H}|}, \tag{3.12}$$

where $\mathbb{H}$, $\mathbb{D}$ are the label subgroups of $\mathcal{S}$. Furthermore, there exist efficient classical algorithms to compute a representative g_0 of the support, a generating set of $\mathbb{D}^\perp$ and the dimension $\dim(\mathcal{V})$.

Before proving theorem 3.2, we note that combining property (ii) together with lemma 3.6(i) immediately yield:

Corollary 3.1 (**Uniqueness Test**). Let $\mathcal{S}$ be a stabilizer group with stabilizer code $\mathcal{V}$. Then $\mathcal{V}$ is one-dimensional if and only if $\mathbb{H}$ and $\mathbb{D}$ annihilate each other: i.e. iff $\mathbb{H} = \mathbb{D}^{\perp}$.

Theorem 3.2(ii) also leads to an alternative formula for the dimension of a stabilizer code:

Corollary 3.2. The dimension of $\mathcal{V}$ equals $|G|/|\mathcal{S}|$.

The result in corollary 3.2 is well known for stabilizer codes over qubits [1, 13] (i.e. where $G = \mathbb{Z}_2^m$ so that $|G| = 2^m$) and qudits (where $G = \mathbb{Z}_d^m$) [1, 202].

Proof. [of corollary 3.2] Consider the map $\varphi : \mathcal{S} \to G$, defined in section 3.4.2, which is a group homomorphism with image $\mathbb{H}$. Furthermore the kernel of φ is precisely the diagonal subgroup $\mathcal{D}$ of G. Since $|\text{Im } \varphi| = |\mathcal{S}|/|\ker \varphi|$ it follows that $|\mathbb{H}| = |\mathcal{S}|/|\mathcal{D}|$. Finally we claim that $\mathcal{D}$ and $\mathbb{D}$ are isomorphic groups so that $|\mathcal{D}| = |\mathbb{D}|$. To prove this, consider the map $\delta : \mathcal{D} \to \mathbb{D}$ that sends $\sigma = \gamma^a Z(g)$ to $\delta(\sigma) = g$. Using (3.3) it follows that this map is a homomorphism; furthermore, it is a surjective one by definition of $\mathbb{D}$, and thus $\text{im}\delta = \mathbb{D}$. The kernel of δ is the set of all $\sigma \in \mathcal{S}$ having the form $\sigma = \gamma^a I$. But the only operator in $\mathcal{S}$ proportional to the identity is the identity itself, since otherwise $\mathcal{S}$ cannot have a common $+1$ eigenstate. This shows that the kernel of δ is trivial, so that $\mathcal{D}$ and $\mathbb{D}$ are isomorphic, as claimed. The resulting identity $|\mathbb{H}| = |\mathcal{S}|/|\mathbb{D}|$ together with $|\mathbb{D}^{\perp}| = |G|/|\mathbb{D}|$ (recall lemma 2.4) and theorem 3.2(ii) proves the result. □

We now prove theorem 3.2 using techniques developed in [203] where the properties of so-called M-spaces were studied. We briefly recall basic concepts and results.

A unitary operator acting on $\mathcal{H}_G$ is said to be *monomial* if it can be written as a product $U = DP$ where D is diagonal and P is a permutation matrix. A subspace $\mathcal{M}$ of $\mathcal{H}_G$ is called an *M-space* if there exists a group of monomial unitary operators $\mathcal{G}$ such that $|\varphi\rangle \in \mathcal{M}$ iff $U|\varphi\rangle = |\varphi\rangle$ for every $U \in \mathcal{G}$. The group $\mathcal{G}$ is called a stabilizer group of $\mathcal{M}$. If $\mathcal{M}$ is one-dimensional, its unique (up to a multiplicative factor) element $|\psi\rangle$ is called an M-state. The support of $\mathcal{M}$ is defined analogously to the support of a stabilizer code i.e. it is the set of all $g \in G$ such that $|g\rangle$ has a nontrivial overlap with $\mathcal{M}$. With this terminology, every stabilizer code is an instance of an M-space and every stabilizer state is an M-state. To see this, note that every Pauli operator $\sigma(a, g, h)$ is a monomial unitary operator. Indeed, σ can be written as a product $\sigma = DP$ where $D = \gamma^a Z(g)$ is diagonal and $P = X(h)$ is a permutation matrix.

We introduce some further terminology. Let $\mathcal{G}$ be an arbitrary monomial stabilizer group. For every $g \in G$, let $\mathcal{G}_g$ be the subset of $\mathcal{G}$ consisting of all $U \in \mathcal{G}$ satisfying $U|g\rangle \propto |g\rangle$ i.e. U acts trivially on g, up to an overall phase. This subset is easily seen to be a subgroup of $\mathcal{G}$. Also, we define the orbit $\mathcal{O}_g$ of g as:

$$\mathcal{O}_g = \{h : \exists U \in \mathcal{G} \text{ s.t. } U|g\rangle \propto |h\rangle\} \tag{3.13}$$

In the following result the support of any M-space is characterized in terms of the orbits $\mathcal{O}_g$ and the subgroups $\mathcal{G}_g$.

Theorem 3.3 (**Support of M-space** [203]). Consider an M-space $\mathcal{M}$ with monomial stabilizer group $\mathcal{G}$. Then the following statements hold:

(i) There exist orbits $\mathcal{O}_{g_1}, \ldots, \mathcal{O}_{g_{\mathbf{d}}}$ such that $\mathbf{d} = \dim(\mathcal{M})$ and

$$\text{supp}(\mathcal{M}) = \mathcal{O}_{g_1} \cup \cdots \cup \mathcal{O}_{g_{\mathbf{d}}}. \tag{3.14}$$

(ii) Consider $g \in G$ and an arbitrary set of generators $\{V_1, \ldots, V_r\}$ of $\mathcal{G}_g$. Then $g \in \mathrm{supp}(\mathcal{M})$ if and only if $V_i|g\rangle = |g\rangle$ for every i.

Using this result, we can now prove theorem 3.2.

Proof. **[of theorem 3.2]** We apply theorem 3.3 to the Pauli stabilizer group $\mathcal{S}$. In this case, the group $\mathcal{S}_g$ and the orbit $\mathcal{O}_g$ fulfill

$$\mathcal{O}_g = g + \mathbb{H}, \qquad \mathcal{S}_g = \mathcal{D}. \tag{3.15}$$

To demonstrate the first identity in (3.15), we use (3.2) which implies $\sigma(a,x,y)|g\rangle \propto |g+y\rangle$ for every $\sigma(a,x,y) \in \mathcal{S}$. To show the second identity, first note that $D|g\rangle \propto |g\rangle$ for every diagonal operator $D \in \mathcal{D}$, showing that $\mathcal{D} \subseteq \mathcal{S}_g$. Conversely, if $\sigma \in \mathcal{S}_g$ has label (a,x,y) then $\sigma|g\rangle \propto |g+y\rangle$. Since $\sigma \in \mathcal{S}_g$ the state $|g\rangle$ is an eigenvector of σ; this can only be true if $y = 0$, showing that $\sigma \in \mathcal{D}$.

Using lemma 3.6, we can efficiently compute the labels of a generating set $\{\sigma_1, \ldots, \sigma_r\}$ of $\mathcal{S}_g = \mathcal{D}$, where $\sigma_i = \gamma^{a_i} Z(g_i)$ for some $a_i \in \mathbb{Z}_{2|G|}$ and $g_i \in G$. Owing to theorem 3.3(ii), any $g \in G$ belongs to the support of $\mathcal{V}$ if and only if $\sigma_i|g\rangle = |g\rangle$ for every $i = 1, \ldots, r$. Equivalently, g satisfies

$$\gamma^{a_i} \chi_{g_i}(g) = 1 \quad \text{for all } i = 1, \ldots, r. \tag{3.16}$$

Since the elements g_i generate the label group $\mathbb{D}$, the solutions of the system are easily seen—use the multiplicativity of characters and (2.17)—to form a coset of the form $\mathrm{supp}(\mathcal{V}) = g_0 + \mathbb{D}^\perp$ for some particular solution g_0. Moreover, the classical algorithm in lemma 3.2.(e), returns a valid g_0 and a generating set of $\mathbb{D}^\perp$, showing (i).

Further, we combine (i) with theorem 3.3(i) to get a short proof of (ii): the equation

$$\mathrm{supp}(\mathcal{V}) = \mathcal{O}_{g_1} \cup \cdots \cup \mathcal{O}_{g_{\mathbf{d}}} = (g_1 + \mathbb{H}) \cup \cdots \cup (g_{\mathbf{d}} + \mathbb{H}) = g_0 + \mathbb{D}^\perp$$

implies, computing the cardinalities of the sets involved, that $\mathbf{d}|\mathbb{H}| = \dim \mathcal{V}|\mathbb{H}| = |\mathbb{D}^\perp|$.

Finally, the ability to compute g_0 and to find generators of $\mathbb{D}^\perp$ efficiently classically follows by applying theorem 2.2 to a linear system described by a $r \times m$ matrix Ω that defines a homomorphism from G to $\mathbb{Z}_{|G|}^r$, with $r \in O(\mathrm{polylog}\,|G|)$. Furthermore, we can compute $\dim \mathcal{V}$ directly using formula (ii) together with lemma 3.6 and the algorithms of lemma 3.1. □

3.5 Normal form of a stabilizer state

We now apply the stabilizer formalism of section 3.4 to develop an analytic characterization of the amplitudes of arbitrary stabilizer states over finite abelian groups. In addition, we show that the wavefunction of any stabilizer state can always be efficiently computed and sampled, which we use later to study the classical simulability of normalizer circuits.

Theorem 3.4 (Normal form of stabilizer states). Every stabilizer state $|\phi\rangle$ over a finite abelian group G with stabilizer group $\mathcal{S}$ has the form

$$|\phi\rangle = \alpha \frac{1}{\sqrt{|\mathbb{H}|}} \sum_{h \in \mathbb{H}} \xi(h)|s+h\rangle. \tag{3.17}$$

Here α is a global phase, $\mathbb{H}$ is the label group (3.9), $s \in G$, and relative phases are described by a quadratic function ξ on the group $\mathbb{H}$. Furthermore, if a generating set $\{\sigma_1, \ldots, \sigma_r\}$ of $\mathcal{S}$ is specified, the following tasks can be carried out efficiently:

(a) Compute s;

(b) Given $g \in G$, determine if $g \in s + \mathbb{H}$;

(c) Given $h \in \mathbb{H}$, compute $\xi(h)$ up to n bits in $\mathrm{poly}(n, \log|G|)$ time;
(d) Compute $\sqrt{|\mathbb{H}|}$.

Proof. Corollary 3.1 implies that $\mathbb{D}^{\perp} = \mathbb{H}$. Using this identity together with theorem 3.2(i), we find that $\mathrm{supp}(|\phi\rangle) = s + \mathbb{H}$ for some $s \in G$. By definition of $\mathbb{H}$, for every $h \in \mathbb{H}$ there exists some element $\sigma(a, g, h) \in \mathcal{S}$. Using that $\sigma(a, g, h)|\phi\rangle = |\phi\rangle$ we then have

$$\langle s + h|\phi\rangle = \langle s + h|\sigma(a, g, h)|\phi\rangle = \gamma^a \chi_{s+h}(g)\langle s|\phi\rangle \tag{3.18}$$

This implies that $|\langle s+h|\phi\rangle| = |\langle s|\phi\rangle|$ for all $h \in \mathbb{H}$. Together with the property that $\mathrm{supp}(|\phi\rangle) = s + \mathbb{H}$, it follows that $|\phi\rangle$ can be written as

$$|\phi\rangle = \frac{1}{\sqrt{|\mathbb{H}|}} \sum_{h \in \mathbb{H}} \xi(h)|s + h\rangle \tag{3.19}$$

for some complex phases $\xi(h)$. By suitably choosing an (irrelevant) global phase, w.l.o.g. we can assume that $\xi(0) = 1$.

We now show that the function $h \in H \to \xi(h)$ is quadratic. Using (3.18, 3.19) we derive

$$\xi(h) = \sqrt{|\mathbb{H}|}\langle s + h|\phi\rangle = \sqrt{|\mathbb{H}|}\gamma^a \chi_{s+h}(g)\langle s|\phi\rangle = \gamma^a \chi_{s+h}(g)\xi(0) = \gamma^a \chi_{s+h}(g). \tag{3.20}$$

Since $\xi(h)$ by definition only depends on h, the quantity $\gamma^a \chi_{s+h}(g)$ only depends on h as well: i.e. it is independent of a and g. Now select $h_1, h_2 \in \mathbb{H}$ and two associated stabilizer operators $\sigma_1 = \sigma(a_1, g_1, h_1)$, $\sigma_2 = \sigma(a_2, g_2, h_2) \in \mathcal{S}$. Then

$$\begin{aligned}
\xi(h_1 + h_2) &= \sqrt{|\mathbb{H}|}\langle s + h_1 + h_2|\phi\rangle && (3.21)\\
&= \sqrt{|\mathbb{H}|}\langle s + h_1 + h_2|\sigma_1\sigma_2|\phi\rangle && (3.22)\\
&= \sqrt{|\mathbb{H}|}\gamma^{a_1}\chi_{s+h_1+h_2}(g_1)\,\langle s + h_2|\sigma_2|\phi\rangle && (3.23)\\
&= [\gamma^{a_1}\chi_{s+h_1}(g_1)]\;[\gamma^{a_2}\chi_{s+h_2}(g_2)]\;\chi_{g_1}(h_2)\,\xi(0) && (3.24)\\
&= \xi(h_1)\xi(h_2)\chi_{g_1}(h_2) && (3.25)
\end{aligned}$$

In (3.22) we used that $\sigma_1\sigma_2|\phi\rangle = |\phi\rangle$; in (3.23-3.24) we used the definitions of Pauli operators and the fact that $\sqrt{|\mathbb{H}|}\langle s|\phi\rangle = \xi(0)$; finally in (3.25) we used identity (3.20) and the fact that $\xi(0) = 1$. Now define $B(h_1, h_2) = \xi(h_1 + h_2)\overline{\xi}(h_1)\overline{\xi}(h_2)$. We claim that B is a bicharacter function of $\mathbb{H}$. To see this, note that the derivation above shows that $B(h_1, h_2) = \chi_{g_1}(h_2)$ for any $\sigma(a_1, g_1, h_1) \in \mathcal{S}$. Linearity in the second argument h_2 is immediate. Furthermore, by definition B is a symmetric function i.e. $B(h_1, h_2) = B(h_2, h_1)$. This shows that B is bicharacter, as desired.

We now address (a)-(d). As for (a) recall that $s + \mathbb{H}$ is the support of a stabilizer state $|\phi\rangle$; theorem 3.2 then provides an efficient method to compute a suitable representative s. Note also that a generating set of $\mathbb{H}$ can be computed efficiently owing to lemma 3.6. Statement (b) follows from lemma 3.1(a). Statement (d) follows from lemma 3.1(b). Finally we prove (c), by showing that the following procedure to compute $\xi(h)$ is efficient, given any $h \in \mathbb{H}$:

(i) determine some element $\sigma \in \mathcal{S}$ such that $\sigma|s\rangle \propto |s + h\rangle$;
(ii) compute $\langle s + h|\sigma|s\rangle = \xi(h)$.

To achieve (i), it suffices to determine an arbitrary stabilizer element of the form $\sigma = \sigma(a, g, h) \in \mathcal{S}$. Assume that generators $\sigma_1 = \sigma(a_1, g_1, h_1), \ldots, \sigma_r = \sigma(a_r, g_r, h_r)$ are given to us. We can then use algorithm (a) in lemma 3.1 to find integers w_i such that $h = \sum w_i h_i$, for which $\sigma = \prod \sigma_i^{w_i}$ is

an operator of form $\sigma(a, g, h)$ for some values of a, g—use (3.3). Moreover, given the w_i the label (a, g, h) of σ can be computed efficiently; this accomplishes (i). Finally, it is straightforward that (ii) can be carried out efficiently: using formula $\xi(h) = \gamma^a \chi_{s+h}(g)$ and standard algorithms to compute elementary functions [198]. □

3.5.1 Reproduction of existing normal forms

Theorem 3.4 generalizes result from [135, 136, 193] where analogous characterizations were given for qubits and qudits, although those works do not consider the notion of quadratic functions used here (furthermore their methods are completely different from ours). For example, in ref. [135] it was shown that every Pauli stabilizer state for qubits (corresponding to the group $\mathbb{Z}_2^m$) can be written as

$$|\phi\rangle \propto \frac{1}{\sqrt{|S|}} \sum_{x \in S} (-1)^{q(x)} i^{l(x)} |x + s\rangle. \tag{3.26}$$

Here S is a linear subspace of $\mathbb{Z}_2^m$, $q(x) = x^T A x \mod 2$ is a quadratic form over $\mathbb{Z}_2$, and $l(x)$ mod 2 is a linear form. This characterization indeed conforms with theorem 3.4: the set S is a subgroup of $\mathbb{Z}_2^m$ and the function

$$x \in \mathbb{Z}_2^m \to \xi(x) := (-1)^{q(x)} i^{l(x)} \tag{3.27}$$

is quadratic (chapter 1.3.1).

Application: computational tractability of stabilizer states

Theorem 3.4 also implies that every stabilizer state belongs to the family of Computationally Tractable states (CT states) [170]. A state $|\psi\rangle = \sum \psi_g |g\rangle \in \mathcal{H}_G$ is said to be CT (relative to its classical description) if the following properties are satisfied:

(a) there exists an efficient randomized classical algorithm to sample the distribution $\{|\psi_g|^2\}$;

(b) given $g \in G$, the coefficient ψ_g can be computed efficiently with exponential precision.

CT states form a basic component in a general class of quantum computations that can be simulated efficiently classically using probabilistic simulation methods. For example consider a quantum circuit $\mathcal{C}$ acting on a CT state and followed by a final standard basis measurement on one of the qubits. Then, regardless of which CT state is considered, such computation can be efficiently simulated classically when $\mathcal{C}$ is e.g. an arbitrary Clifford circuit, matchgate circuit, constant-depth circuit or sparse unitary. See [170] for an extensive discussion of classical simulations with CT states.

Here we show that every stabilizer state $|\psi\rangle \in \mathcal{H}_G$ over a finite abelian group G is CT. To be precise, we prove that such states are CT *up to a global phase*. That is, instead of (b) we prove a slightly weaker statement which takes into account the fact that any stabilizer state specified in terms of its stabilizer is only determined up to an overall phase. Formally, we consider the property

(b') there exists an efficient classical algorithm that, on input of $g \in G$, computes a coefficient ψ'_g, where the collection of coefficients $\{\psi'_g : g \in G\}$ is such that $|\psi\rangle = \alpha \sum \psi'_g |g\rangle$ for some complex phase α.

Corollary 3.3. Let $|\psi\rangle$ be a stabilizer state over an abelian group G, specified in terms of a generating set of polylog $|G|$ stabilizers. Then $|\psi\rangle$ is CT in the sense (a)-(b').

Proof. Property (a) was proved in [134]. To prove (b'), note that theorem 3.4 implies there exists a global phase α such that

$$\langle g|\psi\rangle = \begin{cases} \alpha \cdot \frac{1}{\sqrt{|\mathbb{H}|}} \cdot \xi(h) & \text{if } g = s + h \text{ for some } h \in \mathbb{H} \\ 0 & \text{if } g \notin \mathbb{H} + s. \end{cases} \tag{3.28}$$

Using theorem 3.4(b) it can be efficiently determined whether g belongs to $\mathbb{H} + s$. If not, then $\langle g|\psi\rangle = 0$. If yes, then compute $h = g - s$; then $\xi(h)$ can be computed owing to theorem 3.4(c). Finally, $\sqrt{|\mathbb{H}|}$ can be computed owing to theorem 3.4(d). □

3.6 Pauli measurements in the stabilizer formalism

The rest of this chapter is investigates normalizer circuits that contain intermediate measurements of generalized Pauli operators. In this section we show how generalized Pauli measurements can be implemented and neatly described within our stabilizer formalism over abelian groups (cf. section 3.4). The main result of this section (**theorem 3.5**) is an update-rule for describing the output state after measuring any generalized Pauli operator on an abelian-group stabilizer state. We use these tools to probe the classical simulability of adaptive normalizer circuits in section 3.7.

3.6.1 Definition

Associated with every Pauli operator σ (3.2) we will consider a quantum measurement in the eigenbasis[5] of σ. Consider the spectral decomposition $\sigma = \sum \lambda P_\lambda$ where $\lambda \in \mathbb{C}$ are the distinct eigenvalues of σ and P_λ is the projector on the eigenspace associated with eigenvalue λ. Given a state $|\psi\rangle \in \mathcal{H}_G$, the measurement associated with σ is now defined as follows: the possible outcomes of the measurement are labeled by the eigenvalues $\{\lambda\}$ where each λ occurs with probability $\|P_\lambda|\psi\rangle\|^2$; furthermore, if the outcome λ occurs, the state after the measurement equals to $P_\lambda|\psi\rangle$ up to normalization.

Consider a group G of the form (1), with associated physical system $\mathcal{H}^G = \mathbb{C}^{d_1} \otimes \cdots \otimes \mathbb{C}^{d_m}$. We remark that a measurement of the i-th system $\mathbb{C}^{d_i}$ in the standard basis $\{|0\rangle, \ldots, |d_i - 1\rangle\}$ can be realized as a measurement of a suitable Pauli operator, for every i ranging from 1 to m. To keep notation simple, we demonstrate this statement for the special case $G = \mathbb{Z}_d^m$, yet the argument generalizes straightforwardly to arbitrary G. Denote by $e_i \in G$ the group element which has $1 \in \mathbb{Z}_d$ in its i-th component and zeroes elsewhere. Then definition (3.2) implies that the Pauli operator $Z(e_i)$ acts as Z_d on the i-th qudit and as the identity elsewhere, where Z_d was defined in (1.12). Note that Z_d has d distinct eigenvalues, each having a rank-one eigenprojector $|x\rangle\langle x|$ with $x \in \mathbb{Z}_d$. It follows straightforwardly that measurement of $Z(e_i)$ corresponds to measurement of the i-th qudit in the standard basis.

3.6.2 Implementation

It is easily verified that every Pauli operator σ can be realized as a poly-size (unitary) quantum circuit [134]. Therefore, measurement of σ can be implemented efficiently on a quantum computer using standard phase estimation methods [13]. Here we provide an alternate method. In particular we show that every Pauli measurement can be implemented using *only* normalizer circuits and measurements in the standard basis, which will be a useful ingredient in our proof of theorem 3.7. To this end, we will use the following result from [204].

[5] Recall that σ is not hermitian but still unitary, hence, diagonalizable.

Lemma 3.7 ([204]). For any dimension d and for integers j and k such that $j, k \in \mathbb{Z}_d$, there exists a poly-size normalizer circuit $\mathcal{C}$ over the group $\mathbb{Z}_d$ that transforms $Z(j)X(k)$ into a diagonal Pauli operator of the form $\gamma^a Z(\gcd(j,k))$. Furthermore, there are efficient classical algorithms to compute a description of $\mathcal{C}$.

Corollary 3.4. Consider a Pauli operator σ over an arbitrary finite abelian group G. Then there exists a poly-size normalizer circuit $\mathcal{C}$ over G such that $\mathcal{C}\sigma\mathcal{C}^\dagger = \gamma^a Z(g)$. Furthermore, there are efficient classical algorithms to compute a description of $\mathcal{C}$ as well as γ^a, a and g.

Proof. To compute $\mathcal{C}$ note that every Pauli operator over G has the form $\sigma \propto U_1 \otimes \cdots \otimes U_m$ where U_i is a Pauli operator over $\mathbb{Z}_{d_i}$ and apply lemma 3.7 to each factor. The rest follows by applying theorem 3.1 to compute the label of $\mathcal{C}\sigma\mathcal{C}^\dagger$ and, in the case of γ^a, by using standard algorithms to compute scalar exponentials. □

Lemma 3.7 and corollary 3.4 reduce the problem of measuring general Pauli operators to that of implementing measurements of $Z(g)$. Indeed, given an arbitrary σ to be measured, we can always compute a poly-size normalizer circuit that transforms it into a diagonal operator $\gamma^a Z(g)$, using corollary 3.4. Then, the measurement of σ is equivalent to the procedure (a) apply $\mathcal{C}$; (b) measure $\gamma^a Z(g)$; (c) apply $\mathcal{C}^\dagger$. Finally, Pauli operators that are proportional to each other define the *same* quantum measurement, up to a simple relabeling of the outcomes. Therefore it suffices to focus on the problem of measuring an operator of the form $Z(g)$.

Note now that, by definition, the eigenvalues of $Z(g)$ have the form $\chi_g(h)$. Define the following function ω from G to $\mathbb{Z}_d$, where $d = \mathrm{lcm}(d_1, \ldots, d_m)$:

$$\omega(h) = \sum_i \frac{d}{d_i}\, g(i)h(i) \mod d. \tag{3.29}$$

With this definition one has $\chi_g(h) = \mathrm{e}^{2\pi \mathrm{i}\omega(h)/d}$. Given any $y \in \mathbb{Z}_d$, the eigenspace of $Z(g)$ belonging to the eigenvalue $\lambda = \mathrm{e}^{2\pi \mathrm{i}y/d}$ is spanned by all standard basis states $|h\rangle$ with $\omega(h) = y$.

Next, note that ω is a group homomorphism from G to $\mathbb{Z}_d$ due to lemma 2.8. As a result, the controlled operation $f(h,a) = (h, a + \omega(h))$ is a group automorphism of $G \times \mathbb{Z}_d$ and it can be implemented by a normalizer gate $U_f|h,a\rangle = |h, a+\omega(h)\rangle$.

The gate U_f can now be used to measure $Z(g)$, with a routine inspired by the coset-state preparation method used in the standard quantum algorithm to solve the abelian hidden subgroup problem [118, 119]: first, add an auxiliary d-dimensional system $\mathbb{C}^d$ in the state $|0\rangle$ to $\mathcal{H}^G$, the latter being in some arbitrary state $|\psi\rangle$; second, apply the global interaction U_f; third, measure the ancilla in the standard basis. The global evolution of the system along this process is

$$|\psi\rangle|0\rangle = \sum_{h\in G} \psi(h)|h\rangle|0\rangle \xrightarrow{U_f} \sum_{h\in G} \psi(h)|h\rangle|\omega(h)\rangle \xrightarrow{\text{Measure}} \frac{1}{\sqrt{p_y}} \left(\sum_{h:\omega(h)=y} \psi(h)|h\rangle|y\rangle \right)$$

The measurement yields an outcome $y \in \mathbb{Z}_d$ with probability $p_y = \sum_{h:\omega(h)=y} |\psi(h)|^2$. The latter precisely coincides with $\|P_\lambda|\psi\rangle\|^2$ where P_λ is the eigenprojector associated with the eigenvalue $\lambda = \mathrm{e}^{2\pi \mathrm{i}y/d}$ and, therefore, we have implemented the desired measurement.

In figure 3.1 we show a poly-size quantum circuit that implements the measurement of the Pauli operator $\sigma = \mathcal{C}Z(g)\mathcal{C}^\dagger$ in the way just described. In the picture, the $m+1$ horizontal lines represent the m physical subsystems that form $\mathcal{H}^G = \mathbb{C}^{d_1} \otimes \cdots \otimes \mathbb{C}^{d_m}$ and the ancillary system $\mathbb{C}^d$; the numbers $c_i := d/d_i\, g(i)$ are chosen to compute the function (3.29) in the ancillary system. For merely pictorial reasons, the depicted measurement acts on a standard-basis state. We now make two remarks. First, the state of the ancilla could be reset (with Pauli gates) to its

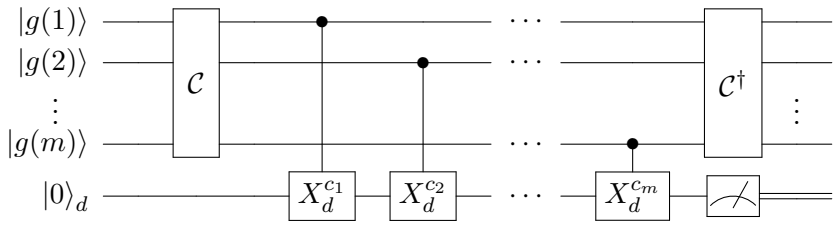

Figure 3.1: Quantum circuit implementing measurement of operator $\sigma = \mathcal{C}Z(g)\mathcal{C}^\dagger$

original value once the measurement outcome $\omega(x)$ is recorded; this could be used to implement a series of measurements using only one ancilla. Second, the value $\omega(x)$ can be used to compute $\lambda = \chi_g(x) = \mathrm{e}^{2\pi \mathrm{i}\omega(x)/d}$.

Finally we mention that a procedure given in [137] to implement measurements of qudit Pauli operators (as presented in section 1.3.1.2) can be recovered from ours by choosing $G = \mathbb{Z}_d^m$.

3.6.3 Measurement update rule

In this section we show that Pauli measurements transform stabilizer states into new stabilizer states. We give an analytic formula to update their description. Moreover we show that the update can be carried out efficiently.

Theorem 3.5 (**Measurement update-rule**). Consider a stabilizer state $|\phi\rangle$ over G with stabilizer group $\mathcal{S}$ and let σ be a Pauli operator. Perform a measurement of σ on $|\phi\rangle$, let the measurement outcome be labeled by an eigenvalue λ of σ, and let $|\phi_m\rangle$ denote the post-measurement state. Then the following statements hold:

(i) The state $|\phi_m\rangle$ is a stabilizer state, with stabilizer group

$$\mathcal{S}_m = \langle\, \overline{\lambda}\sigma,\, C_{\mathcal{S}}(\sigma)\rangle. \tag{3.30}$$

Here $C_{\mathcal{S}}(\sigma)$ denotes the centralizer of σ inside $\mathcal{S}$, i.e. the group containing all elements of $\mathcal{S}$ that commute with σ.

(ii) The labels of a generating set of $\mathcal{S}_m$ can be computed efficiently classically, given the labels of a generating set of $\mathcal{S}$.

Proof. First we show that $|\phi_m\rangle$ is stabilized by $\mathcal{S}_m$. To see this, first note that $|\phi_m\rangle$ is trivially stabilized by $\overline{\lambda}\sigma$. Furthermore, σ commutes with every $\tau \in C_{\mathcal{S}}(\sigma)$. It follows that the projector P onto the λ-eigenspace of σ commutes with τ as well (this can easily be shown by considering σ and τ in their joint eigenbasis). Hence $\tau P|\phi\rangle = P\tau|\phi\rangle = P|\phi\rangle$. Using that $|\phi_m\rangle \propto P|\phi\rangle$, we find that $\tau|\phi_m\rangle = |\phi_m\rangle$ for every $\tau \in C_{\mathcal{S}}(\sigma)$.

Second, we prove that $|\phi_m\rangle$ is the unique state stabilized by the group $\mathcal{S}_m$. Without loss of generality we restrict to the case $\sigma = Z(g_m)$. This is sufficient since, first, every Pauli operator can be transformed into an operator of the form $\alpha Z(g)$ with a suitable normalizer circuit $\mathcal{C}$ (cf. the discussion in section 3.6.2); and, second, for any normalizer circuit $\mathcal{C}$, the quantum state $|\phi_m\rangle$ is a stabilizer state with stabilizer group $\mathcal{S}_m$ if and only if $\mathcal{C}|\psi_m\rangle$ is a stabilizer state with stabilizer group $\mathcal{C}\mathcal{S}_m\mathcal{C}^\dagger$.

Working with the assumption $\sigma = Z(g_m)$, we write the label subgroups $\mathbb{H}_m$ and $\mathbb{D}_m$ of $\mathcal{S}_m$ in terms of the label groups $\mathbb{H}$ and $\mathbb{D}$ of $\mathcal{S}$. We have $\mathcal{S}_m = \langle\, \overline{\lambda}\sigma,\, C_{\mathcal{S}}(\sigma)\,\rangle$ where $\sigma = Z(g_m)$ for

some $g_m \in G$. This implies that only the labels of $C_{\mathcal{S}}(\sigma)$ contribute to $\mathbb{H}_m$. The centralizer $C_{\mathcal{S}}(\sigma)$ can be written as $C_{\mathcal{S}}(\sigma) = \mathcal{S} \cap C_{\mathcal{P}}(\sigma)$, where $C_{\mathcal{P}}(\sigma)$ is the subgroup of all Pauli operators that commute with σ. Thence, using lemma 3.4 we see that $C_{\mathcal{P}}(\sigma)$ consists of all $\gamma^a Z(g)X(h)$ with labels $h \in \langle g_m \rangle^\perp$. Hence,

$$\mathbb{H}_m = \mathbb{H} \cap \langle g_m \rangle^\perp \tag{3.31}$$

Due to the commutativity of $Z(g_m)$ and $C_{\mathcal{S}}(\sigma)$, any element in $\mathcal{S}_m$ can be reordered as $\tau\, Z(g_m)^i$, with $\tau \in C_{\mathcal{S}}(\sigma)$. Therefore, the diagonal group of $\mathcal{S}_m$ can be written as $\mathcal{D}_m = \langle \mathcal{D}', Z(g_m) \rangle$ where $\mathcal{D}'$ is the diagonal subgroup of $C_{\mathcal{S}}(\sigma)$. We now claim that $\mathcal{D}' = \mathcal{D}$ where $\mathcal{D}$ is the diagonal subgroup of $\mathcal{S}$. To see this, first note that trivially $\mathcal{D}' \subseteq \mathcal{D}$ since $C_{\mathcal{S}}(\sigma)$ is a subgroup of $\mathcal{S}$. Conversely, $\mathcal{D} \subseteq \mathcal{D}'$: as every diagonal element of $\mathcal{S}$ commutes with $Z(g_m)$, we have $\mathcal{D} \subseteq C_{\mathcal{S}}(\sigma)$; but this implies $\mathcal{D} \subseteq \mathcal{D}'$.

Putting everything together, we thus find $\mathcal{D}_m = \langle \mathcal{D}, Z(g_m) \rangle$. It follows:

$$\mathbb{D}_m = \langle \mathbb{D}, \langle g_m \rangle \rangle \quad \Longrightarrow \quad \mathbb{D}_m^\perp = \langle \mathbb{D}, \langle g_m \rangle \rangle^\perp = \mathbb{D}^\perp \cap \langle g_m \rangle^\perp, \tag{3.32}$$

where we used lemma 2.4. Since $\mathcal{S}$ uniquely stabilizes $|\phi\rangle$, we have $\mathbb{H} = \mathbb{D}^\perp$ owing to corollary 3.1. With (3.32) and (3.31) this implies that $\mathbb{H}_m = \mathbb{D}_m^\perp$. Again using corollary 3.1, it follows that $\mathcal{S}_m$ uniquely stabilizes $|\phi_m\rangle$.

To complete the proof of the theorem, we give an efficient classical algorithm to find a generating set of the centralizer $C_{\mathcal{S}}(\sigma)$; our approach is to reduce this task to a certain problem over the group $G \times G$ that can be efficiently solved using lemma 3.1. Let $K \subset G \times G$ be the group of tuples (g, h) such that there exists a stabilizer operator $\sigma(a, g, h) \in C_{\mathcal{S}}(\sigma)$; we prove that $C_{\mathcal{S}}(\sigma)$ is isomorphic to K via the map $\kappa : \sigma(a, g, h) \to (g, h)$, and that κ is efficiently classically invertible: this reduces the problem to finding a generating set of K and applying the map κ^{-1} to all its elements.

First, it is straightforward to verify that κ is an isomorphism. Equations (3.3) imply that the map is indeed linear. Surjectivity is granted by definition. Invertibility follows then from the fact that only elements of the type $\gamma^a I \in C_{\mathcal{S}}(\sigma)$, for some a, belong to $\ker \kappa$ (where I denotes the identity): the latter are invalid stabilizer operators unless $\gamma^a = 1$.

Second, we show how to compute κ^{-1}. Let the operator to measure σ be of the (general) form $\sigma = \gamma^{a_m} Z(x)X(y)$, and let (a', g', h') be the label of an arbitrary stabilizer $\tau \in \mathcal{S}$. Given a set of (mutually commuting) generators $\sigma_1, \ldots, \sigma_r$ of $\mathcal{S}$, with corresponding labels (a_i, g_i, h_i), the element τ can be written in terms of them as

$$\tau = \prod \sigma_i^{v_i} = \gamma^{a'} X\left(\sum v_i g_i\right) Z\left(\sum v_i h_i\right) \tag{3.33}$$

for some integers v_i. From this equation it follows that $K \subset \langle (g_1, h_1), \ldots, (g_r, h_r) \rangle$, which leads us to the following algorithm to compute κ^{-1}: given $(g, h) \in K$, use the algorithm of lemma 3.1(a) to compute r integers w_i such that $(g, h) = \sum w_i (g_i, h_i)$; due to (3.3), the stabilizer operator defined as $\varsigma = \prod \sigma_i^{w_i}$ (whose label can be efficiently computed) is proportional to $X(g)Z(h)$; it follows that $\kappa(\varsigma) = (g, h)$ and, hence, ς equals $\kappa^{-1}(g, h)$.

Finally, combining (3.33) with formula (iii) in lemma 3.4 we obtain

$$K = \langle y, -x \rangle^\perp \cap \langle (g_1, h_1), \ldots, (g_r, h_r) \rangle. \tag{3.34}$$

Using eq. (3.34) together with algorithms (c-d) of lemma 3.1, we can efficiently compute $s = \operatorname{polylog} |G|$ elements $(x_1, y_1), \ldots, (x_s, y_s)$ that generate K; applying κ^{-1} to these, we end up with a set of stabilizer operators $\kappa^{-1}(x_i, y_i)$ that generates $C_{\mathcal{S}}(\sigma)$. □

3.7 Classical simulation of adaptive normalizer Circuits

In this section we prove our main result, i.e., a classical simulation theorem for adaptive normalizer circuits a la Gottesman-Knill (**theorem 3.7**). We conclude the chapter discussing some significant differences between the power of adaptiveness for quantum state preparation in our formalism compared to previous qubit and prime qudit works (**section 3.7.2**).

3.7.1 Simulation result

Recall that in [134] the following classical simulation result was shown:

Theorem 3.6. Let $G = \mathbb{Z}_{d_1} \times \cdots \times \mathbb{Z}_{d_m}$ be a finite abelian group. Consider a polynomial size unitary normalizer circuit over G acting on a standard basis input state. Both circuit and input are specified in terms of their standard encodings as described above. The circuit is followed by a measurement in the standard basis. Then there exists an efficient classical algorithm to sample the corresponding output distribution.

In the theorem, the *standard encoding* of a normalizer circuit is defined as in section 3.3 in this chapter; the *standard encoding* of a standard basis input state $|g\rangle$ is simply the tuple g, i.e. a collection of m integers. Recall also that "efficient" is synonymous to "in polynomial time in $\log|G|$".

Proof of theorem 3.6. For the sake of completeness, we will prove the result using the techniques of this chapter and refer the reader to [134] for the original proof. Let $\mathcal{C}$ denote the normalizer circuit. Without loss of generality we assume that the input state is $|0\rangle$. Indeed, any standard basis state $|g\rangle$ can be written as $|g\rangle = X(g)|0\rangle$. The Pauli operator $X(g)$ can be realized as a polynomial-size normalizer circuit because of corollary 3.4 and because diagonal $Z(g)$ gates can be implemented with standard phase kickback tricks [134, 205]. Hence, $X(g)$ can be absorbed in the overall adaptive normalizer circuit.

Next, let $e_i \in G$ denote the i-th "canonical basis vector" and note that the state $|0\rangle$ is a stabilizer state with stabilizer generators $Z(e_1), \ldots, Z(e_m)$: clearly, all $Z(e_i)$s stabilize $|0\rangle$; furthermore, the label subgroups associated to the stabilizer code $\mathcal{S} := \langle Z(e_1), \ldots, Z(e_m)\rangle$ fulfill[6] $\mathbb{D} = G = \{0\}^\perp = \mathbb{H}^\perp$, hence, $|0\rangle$ is uniquely stabilized because of corollary 3.1. It follows that the state $|\psi\rangle = \mathcal{C}|0\rangle$ is a stabilizer state uniquely stabilized by $\mathcal{C}\mathcal{S}\mathcal{C}^\dagger$ with generators $\{\mathcal{C}Z(e_i)\mathcal{C}^\dagger\}_{i=0}^m$, the labels of which can be efficiently computed due to theorem 3.1. Finally, since simulating a terminal measurement in the standard basis is equivalent to sampling the probability distribution $|\psi(x)|^2$, the claim follows from the fact that stabilizer states described in terms of poly-size sets of stabilizer generators are computationally tractable (corollary 3.3). □

The main classical simulation result of this chapter (theorem 3.7 below) is a generalization of the above result. Rather than unitary normalizer circuits, the family of quantum circuits considered here is that of the *adaptive normalizer circuits*. A polynomial-size adaptive normalizer circuit consists of polylog $|G|$ elementary steps, each of which is either a unitary normalizer gate U or a Pauli measurement M. Furthermore, the choice of which U or M to apply in any given step may depend, in a (classical) polynomial-time computable way, on the collection of outcomes obtained in all previous measurements. The notion of adaptive normalizer circuits is thus a direct generalization of the adaptive Clifford circuits considered in the original Gottesman-Knill theorem [1, 2]. Note that, compared to theorem 3.6, two elements are added. First, measurements are no longer restricted to be standard basis measurements but arbitrary Pauli measurements. Second, the circuits are adaptive.

[6] This identity follows from lemma 2.4(a-b), which yields $G = (G^\perp)^\perp = \{0\}^\perp$.

Before stating our classical simulation result, we make precise what is meant by an efficient classical simulation of an adaptive normalizer circuit. First, recall that the outcomes of any Pauli measurement are labeled by the eigenvalues of the associated Pauli operator. Since $\sigma^{2|G|} = I$ (recall lemma 3.3) it follows that each Pauli operator eigenvalue is a $2|G|$-th root of unity i.e. it has the form $\lambda = \mathrm{e}^{\pi \mathrm{i} k/|G|}$ for some $k \in \{0, \ldots, 2|G|-1\}$. This implies that any Pauli measurement gives rise to a probability distribution over the set of $2|G|$-th roots of unity; we denote the latter set by $S_{2|G|}$. Now consider an adaptive normalizer circuit $\mathcal{C}$. Let $P_i(\lambda|\lambda_1 \cdots \lambda_{i-1})$ denote the conditional probability of obtaining the outcome $\lambda \in S_{2|G|}$ in the i-th measurement, given that in previous measurements the outcomes $\lambda_1 \cdots \lambda_{i-1} \in S_{2|G|}$ were measured. We now say that $\mathcal{C}$ can be simulated efficiently classically if *for every* i the i-th conditional probability distribution $P_i(\lambda|\lambda_1 \cdots \lambda_{i-1})$ can be sampled efficiently on a classical computer, given the description of all gates and measurement operators in the circuit.

Theorem 3.7 (**Classical simulation of adaptive normalizer circuits**)**.** Consider a polynomial size *adaptive* normalizer circuit over G, specified as a list of normalizer gates in their standard encoding, which acts on an arbitrary standard basis input state. Then any such circuit can be efficiently simulated classically.

Proof. Let $\mathcal{C}$ denote the adaptive normalizer circuit. Without loss of generality we assume that the input state is $|0\rangle$. Indeed, any standard basis state $|g\rangle$ can be written as $|g\rangle = X(g)|0\rangle$; the Pauli operator $X(g)$ can be realized as a polynomial-size normalizer circuit [134] and can thus be absorbed in the overall adaptive normalizer circuit. Letting $e_i \in G$ denote the i-th "canonical basis vector", the state $|0\rangle$ is a stabilizer state with stabilizer generators $Z(e_1), \ldots, Z(e_m)$ [134]. We now recall the following facts, proved above:

(a) Given any normalizer gate U and any stabilizer state $|\psi\rangle$ specified in terms of a generating set of polylog $|G|$ generators, the state $U|\psi\rangle$ is again a stabilizer state; moreover a set of generators can be determined efficiently (see theorem 3.1).

(b) Given any Pauli operator σ and any stabilizer state $|\psi\rangle$ specified in terms of a generating set of polylog$(|G|)$ generators, the state $|\psi_\lambda\rangle$ obtained after measurement of σ, for any outcome λ, is again a stabilizer state; moreover a set of generators can be determined efficiently (cf. theorem 3.5).

Furthermore, the measurement probability distribution can be sampled efficiently in polynomial time on a classical computer. The latter is argued as follows. First, it follows from the discussion in section 3.6.2 that the simulation of any Pauli measurement, on some input stabilizer state $|\psi\rangle$, reduces to simulating a unitary normalizer circuit (the description of which can be computed efficiently) followed by a *standard basis* measurements (acting on the same input $|\psi\rangle$ and a suitable ancillary stabilizer state $|0\rangle$). Second, normalizer circuits acting on stabilizer state inputs and followed by standard basis measurements on stabilizer states can be simulated efficiently: this was shown in the proof of theorem 3.6 for an input stabilizer state with stabilizer generators $Z(e_1), \ldots, Z(e_m)$; the argument, however, carries over immediately to the general case.

The proof of the result is now straightforward. Given any tuple $\lambda_1, \ldots, \lambda_{i-1}$, a generating set of stabilizers can be computed efficiently for the state of the quantum register obtained immediately before the i-th measurement, given that the previous measurement outcomes were $\lambda_1 \cdots \lambda_{i-1}$. Furthermore, given this stabilizer description, the distribution $P_i(\lambda|\lambda_1 \cdots \lambda_{i-1})$ can be sampled efficiently on a classical computer, as argued in (b). □

3.7.2 The role of adaptiveness

To conclude this section we comment on an interesting difference between normalizer circuits and the "standard" qubit Clifford circuits, concerning the role of adaptiveness as a **tool for state preparation**. For qubits, adaptiveness adds no new state preparation power to the unitary Clifford operations. Indeed for any n-qubit stabilizer state $|\psi\rangle$ there exists a (poly-size) *unitary* Clifford circuit $\mathcal{C}$ such that $|\psi\rangle = \gamma\mathcal{C}|0\rangle^{\otimes n}$, for some global phase γ [135]. In contrast, over general abelian groups G this feature is no longer true. The associated adaptive normalizer circuits allow to prepare a *strictly* larger class of stabilizer states compared to unitary normalizer circuits alone.

To demonstrate this claim, we provide a simple example of a stabilizer state over $G = \mathbb{Z}_4$ that *cannot* be prepared from standard basis input states via unitary normalizer transformations over G, even in exponential time. However, the same state can be prepared efficiently *deterministically* if one considers adaptive normalizer schemes. We consider

$$|\psi\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |2\rangle\right) \tag{3.35}$$

Suppose that there existed a unitary Clifford operator $U \in \mathcal{C}^G$ which generates $|\psi\rangle$ from $|0\rangle$. Since the stabilizer group of $|0\rangle$ is generated by $Z(1)$, the stabilizer group of $|\psi\rangle$ would be generated by $UZ_dU^\dagger$. However it was shown in [136] that the stabilizer group of $|\psi\rangle$ cannot be generated by one single Pauli operator (i.e. at least two generators are needed), thus leading to a contradiction.

On the other hand, we now provide an efficient adaptive normalizer scheme to prepare, not only the example $|\psi\rangle$, but in fact any coset state [118, 119, 7] of any finite abelian group G. This refers to any state of the form

$$|H + x\rangle := \frac{1}{\sqrt{|H|}} \sum_{h\in H} |h + x\rangle, \tag{3.36}$$

where H is a subgroup of G and $x \in G$. Note that $|\psi\rangle$ is a coset state of the group $G = \mathbb{Z}_4$ with $H := \langle 2\rangle$ and $x := 0$.

Our algorithm to efficiently prepare general coset states $|H+x\rangle$ receives the element x and a polynomial number of generators of H. Here, we use the classical algorithm in lemma 3.1.(h) to efficiently compute the matrix representation of a group homomorphism $\varpi : G \to \mathbb{Z}_d^s$ such that $\ker\varpi = H$, where s and d have polynomial bit-size. Given ϖ, we define a group automorphism α of the group $G \times \mathbb{Z}_d^s$ by $\alpha(g, h) := (g, h + \varpi(g))$. We now consider the following procedure[7]:

$$|0\rangle|0\rangle \xrightarrow{\mathcal{F}_G\otimes I} \sum_{h\in G} |h\rangle|0\rangle \xrightarrow{U_\alpha} \sum_{h\in G} |h\rangle|\varpi(h)\rangle \xrightarrow{M} \frac{1}{\sqrt{|H|}} \sum_{h\in H} |g + h\rangle|b\rangle = |g + H\rangle|b\rangle, \tag{3.37}$$

where $\mathcal{F}_G$ denotes the QFT over G, the unitary U_α is the automorphism gate sending $|g, h\rangle$ to $|\alpha(g, h)\rangle$, and M is a measurement of the second register in the standard basis. If the measurement outcome is b, then the post-measurement state is $|g + H\rangle|b\rangle$ where g is a solution of the equation $\varpi(g) = b$. It can be verified that each coset state of H (and thus also the desired coset state $|x + H\rangle$) occurs equally likely, i.e. with probability $p = |H|/|G|$: in general, p can be exponentially small. However, if we apply adaptive operations, we can always prepare $|x + H\rangle$ with probability 1, as follows. First, given the measurement outcome b we efficiently compute

[7] Observe that ϖ can be considered as a function that *hides* the subgroup H in the sense of the hidden subgroup problem (HSP) [118, 119, 7]. That is, for every $g, g' \in G$ we have $\varpi(g) = \varpi(g')$ iff $g - g' \in H$. Procedure (3.37) is essentially the routine used in the quantum algorithm for HSP to prepare random coset states.

an element $g' \in G$ satisfying $\varpi(g') = b$ using theorem 2.2. Then we apply a "correcting" Pauli operation $X(x - g')$ to the first register state, yielding $X(x - g')|g + H\rangle = |x + (g - g') + H\rangle = |x + H\rangle$ as desired (we implicitly used $g - g' \in H$).

Chapter 4

Normalizer circuits and a Gottesman-Knill theorem for infinite-dimensional systems

In chapter 3 we studied models of normalizer circuits over finite abelian groups that act on arbitrary finite-dimensional systems $\mathcal{H}_{D_1} \otimes \cdots \otimes \mathcal{H}_{D_n}$. The latter constituted group theoretic generalized Clifford operations that implemented quantum Fourier transforms, group automorphism gates and quadratic phase gates over a group of the form $G = \mathbb{Z}_{D_1} \times \cdots \times \mathbb{Z}_{D_n}$. In this chapter, we extend the normalizer-circuit formalism to *infinite dimensions*, by allowing normalizer gates to act on systems of the form $\mathcal{H}_{\mathbb{Z}}^{\otimes a}$: each factor $\mathcal{H}_{\mathbb{Z}}$ has a standard basis labeled by *integers* $\mathbb{Z}$, and a Fourier basis labeled by *angles*, elements of the *circle group* $\mathbb{T}$. As discussed in chapter 1, in this setting, normalizer circuits become hybrid quantum circuits acting both on continuous- and discrete-variable systems. Here, we show that infinite-dimensional normalizer circuits can be efficiently simulated classically with a generalized ***stabilizer formalism*** for Hilbert spaces associated with groups of the form $\mathbb{Z}^a \times \mathbb{T}^b \times \mathbb{Z}_{D_1} \times \cdots \times \mathbb{Z}_{D_n}$. We develop new techniques to track stabilizer-groups based on the *normal forms* for group automorphisms and quadratic functions we developed in chapter 2: we use the latter classical techniques to reduce the problem of simulating these extended normalizer circuits to that of finding general solutions of systems of mixed real-integer linear equations [199] and exploit this fact to devise a robust simulation algorithm: the latter remains efficient even in pathological cases where stabilizer groups become *infinite*, *uncountable* and *non-compact*. The techniques developed in this chapter might find applications in the study of fault-tolerant quantum computation with superconducting qubits [89, 90].

This chapter is based on [64] (joint work with Cedric Yen-Yu Lin and Maarten Van den Nest).

4.1 Introduction

In this chapter we investigate our generalized infinite-dimensional normalizer circuit model (chapter 1) where normalizer gates were associated to abelian groups G that can be *infinite*. Specifically, our interest is to focus on groups of the form $G = F \times \mathbb{Z}^a$, where $F = \mathbb{Z}_{d_1} \times \ldots \times \mathbb{Z}_{d_n}$ is a finite abelian group (decomposed into cyclic groups) as in the normalizer circuit setting considered in chapter 3, and where $\mathbb{Z}$ denotes the additive group of integers—the latter being an infinite group. The motivation for adding $\mathbb{Z}$ is that several number theoretical problems are naturally connected to problems over the integers, a crucial example being the *factoring*

problem, which is reducible to a hidden subgroup problem over $\mathbb{Z}$ [55–58]. The main result of this paper is a proof that all normalizer circuits over infinite groups G can be simulated classically in polynomial time, thereby extending the classical simulation results obtained in chapter 3 for normalizer circuits over finite abelian groups.

Similarly to the finite group case, normalizer circuits over an infinite group G are composed of automorphism gates, quadratic phase gates and quantum Fourier transforms (chapter 1, 1.4). However, as discussed in chapter 1-1.4.1, several issues that are not present in finite dimensions arise in extending normalizer circuits to infinite groups G:

(i) First, because the physical system associated with G has standard basis vectors $|g\rangle$ labeled by elements in G, the Hilbert space of the computation is infinite-dimensional.

(ii) Second, infinite-dimensional quantum Fourier transforms (QFT) perform changes of basis between the group element basis $\{|g\rangle\}$ and a *Fourier basis*, but are no longer gates: this was true in chapter 3 because Fourier basis elements were labeled by elements of the character group $\widehat{G}$ that was isomorphic to G; however, the infinite-yet-discrete group $\mathbb{Z}$ has a *continuous-variable* Fourier basis labeled by elements of the circle group $\mathbb{T}$[1].

In chapter 1.4.1, we saw that (i-ii) have important consequences for the treatment of normalizer gates over G. In particular, in order to construct a closed normalizer formalism over groups $F\times\mathbb{Z}^a$ in this chapter, we will need to consider continuous ones $F\times\mathbb{Z}^a\times\mathbb{T}^b$, let the computational basis change along the computation via the action of QFTs (the latter changed the underlying group G indexing the basis) and allow initial state preparations and measurements in all group-element and Fourier bases associated to the Hilbert space $\mathcal{H}_G$.

4.1.1 Main results

To achieve an efficient classical simulation of normalizer circuits over $F\times\mathbb{Z}^a\times\mathbb{T}^b$ (**theorem 4.1**), we develop new ***stabilizer formalism*** techniques which extend the stabilizer formalism for finite abelian groups of chapter 3 (which, in turn, extended the well-known stabilizer formalism for qubit/qudit systems [1, 2, 24, 135, 136, 133, 137]).

In generalizing the stabilizer formalism to describe infinite-dimensional normalizer circuits, several complications arise from the presence of non-finite nor-finitely-generated associated groups. One immediate consequence is that the associated Pauli stabilizer groups are *no longer finitely generated* either, which is a *unique* infinite-dimensional feature. This fact requires the development of new simulation techniques, since in our (and all known) finite-dimensional stabilizer formalisms, a central common feature is that quantum states can be *efficiently* described as eigenstates of stabilizer groups of Pauli operators that are finite and fully determined by *small lists* of group generators. 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, from which final measurement statistics can be efficiently reproduced classically by suitably manipulating the stabilizer generators of the final state.

For the above reasons, in this chapter we abandon the standard method to track stabilizer groups. Instead, we devise a new simulation method based on the existence of certain concise ***normal forms*** for quadratic functions and homomorphisms on G, which we presented as independent (classical) results in chapter 2. Thereby, we demonstrate that these purely group-theoretic contributions of the thesis have interesting applications in quantum information, far beyond those discussed in chapter 3: therein, we exploited matrix representations in simulations

[1]We recall that $\mathbb{T}=[0,1)$ (resp. $\mathbb{T}^n$) are the group of angles (given in 2π units) with the addition and the n-dimensional hypertorus: as discussed in chapter 2.1.2, $\mathbb{T}^n$ is isomorphic to the character group of $\mathbb{Z}^n$ for any n.

and used quadratic functions to develop normal forms for *stabilizer states*[2]; in this chapter, we find new uses of these classical tools, e.g., to develop *efficient classical encodings* for infinite-dimensional stabilizer states that cannot be described with any other available method.

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 ***linear equations over infinite groups***, which we showed how to solve in chapter 2. To find this reduction, we make crucial use of the afore-mentioned normal forms and our infinite-group stabilizer formalism.

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, accuracy 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.

4.1.2 Relationship to previous work

In the particular case when G is finite, our results completely generalize the results in [134] and some of the results of chapter 3 (ref. [63]): here, we fully characterize the support of stabilizer states in infinite dimensions, but, for simplicity, we will no longer allow adaptive Pauli measurements in the middle of a normalizer computation.[3]

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 [175, 29, 140, 31, 32, 141], 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 [175, 29, 140, 31, 32, 141, 30, 142, 143], being a key ingredient in current schemes for CV quantum error correction [140, 206] and CV measurement-based quantum computation with CV cluster states [207–209, 206]. A CV version of the Gottesman-Knill theorem [31, 32] for simulations of Gaussian unitaries (acting on Gaussian states) has been derived in this framework.

We stress that, although our infinite-group stabilizer formalism in this chapter and the CV stabilizer formalism share some similarities, they are physically and mathematically *inequivalent* and should not be confused with each other. The results in this chapter are for Hilbert spaces of the form $\mathcal{H}_{\mathbb{Z}}^{\otimes a} \otimes \mathcal{H}_{\mathbb{T}}^{\otimes b} \otimes \mathcal{H}_{\mathbb{Z}_{N_1}} \otimes \cdots \otimes \mathcal{H}_{\mathbb{Z}_{N_c}}$ with a basis $|g\rangle$ labeled by the elements of $\mathbb{T}^a \times \mathbb{Z}^b \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_c}$: 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**[4]. In the CV formalism [31], 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 framework (see sections 1.4.2, 4.3 and [31] for examples) are simply inequivalent.

Furthermore, dealing with continuous-variable stabilizer groups as in [140, 31, 141] is sometimes simpler, from the simulation point of view, because the group $\mathbb{R}^m$ is also a finite-

[2] This is due to our normal form for stabilizer states in theorem 3.4; we mention that this result can also be easily extended to infinite dimensional stabilizer states as considered later in this chapter (section 4.4).

[3] We believe it should be possible to fully extend the simulation techniques in chapter 3 to infinite dimensions.

[4] 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 [178]).

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*. 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 chapter 3, where the differences between prime-qudit stabilizer codes [1–3] (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. Last, we mention that a similarity with the $\mathbb{R}^m$ case is that our groups are not *compact*.

Finally, we mention some related work on the classical simulability of Clifford circuits based on different techniques other than stabilizer groups: see [138] for simulations of qubit non-adaptive Clifford circuits in the Schrödinger picture based on the stabilizer-state normal form of [135]; see [44, 159] for phase-space simulations of odd-dimensional qudit Clifford operation exploiting a local hidden variable theory based on the discrete Wigner function of [210, 193, 194]; see [47] and [48] for phase-space simulations of restricted types of Clifford operations based on sampling rebit and qubit Wigner functions.

It should be insightful at this point to discuss briefly whether the latter results may extend to our set-up. In this regard, it seems plausible to the authors that efficient simulation schemes for normalizer circuits analog to those in [138] might exist and may even benefit from the techniques developed in the present work (specifically, our normal forms, as well as those given in chapter 3). Within certain limitations, it might also be possible to extend the results in [44, 159, 47] and [48] to our setting. One fundamental limitation is that local hidden variable models for the full-fledged stabilizer formalism on *qubits* (which we generalize here) cannot exist due to the existence of certain stabilizer-type Bell inequalities [211–213]. Consequently, in order to find a hypothetical non-negative quasi-probability representation of normalizer circuits with properties analogue to those of the standard discrete Wigner function as in [193, 194, 47] (which leads to local HVMs), one would necessarily need to specialize to restricted normalizer-circuit models[5] with, e.g., fewer types of gates, input states or measurements; this, in fact, is part of the approach followed in [47, 48]. The case for [48] is more subtle, since the classical simulation method therein is based on more general *non-contextual* hidden variable models: it is presently a subject of ongoing research whether the standard (qubit) Gottesman-Knill can be recovered by sampling non-contextual HVMs; this program deals with some interesting challenges related to the phenomenon of state-independent quantum contextuality with Pauli observables (we refer the reader to [48] for discussion).

Currently, there are no good candidate Wigner functions for extending the results of [44, 159] or [47] to systems of the form $\mathcal{H}_{\mathbb{Z}}^{\otimes a}$: the proposed ones (see [88, 87, 214] and references therein) associate negative Wigner values to Fourier basis states (which are allowed inputs in our formalism and also in [44, 159, 47]) that we introduce in section 1.4.1; for one qubit, these are the usual $|+\rangle$, $|-\rangle$ states. The existence of a non-negative Wigner representation for this individual case has not been ruled out by Bell inequalities or contextuality arguments, up to our best knowledge.

[5]Note that this might not be true for quasi-probability representations that do not lead to non-local HVMs. The locality of the hidden variable models given in [210, 194, 44, 47] comes both from the positivity of the Wigner function and an additional factorizability property (cf. [194] and [47] page 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.

4.1.3 Discussion and outlook

Finally, we discuss open questions suggested by the work in this chapter as well as a few potential avenues for future research.

First, we anticipate that the techniques developed in this chapter will play a key role in the following chapter 5, where we will draw a rigorous connection between the normalizer circuit framework developed here and a large family of quantum algorithms, including Shor's factoring algorithm. (In particular, normalizer circuits over infinite groups of the form $\mathbb{Z}$ will be essential to understand this celebrated quantum algorithm.)

Connections with quantum error correction. We also point out that, 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-"rotating"-variable error correcting codes for quantum computing architectures based on superconducting qubits. Consider, for instance, the so-called 0-π *qubits* [89, 90]. 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 [89, 90]). Hence, we can interpret them as instances of generalized *stabilizer codes*[6] over the groups $\mathbb{Z}$ and $\mathbb{T}$. The authors believe that it should be possible to apply the simulation techniques in this paper (e.g., our generalized Gottesman-Knill theorem) in 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 [1, 2] 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, [82, 83, 81, 79]).

Also in relation with quantum error correction, it would interesting to improve the stabilizer formalism in this chapter in order to describe *adaptive Pauli measurements*; this would fully extend our simulation results from chapter 3.

Connection with bosonic Gaussian unitaries. In relation to works on continuous-variable and hybrid quantum information processing, it would be interesting to investigate normalizer circuits over more general types of infinite groups. After completion of this work, we became aware that *normalizer circuits over* $\mathbb{R}^m$ groups and Gaussian unitaries can be used to efficiently approximate each other to any accuracy (this result is presented in appendix E, theorem 5.1), hence, define identical gate models for all practical purposes. Because of the importance of the Clifford and Gaussian formalisms in discrete- and continuous-variable QIP, a natural next question (that we leave to future investigations) would be to analyze the potential QIP applications of normalizer gates over hybrid systems $\mathcal{H}_{\mathbb{R}}^{\otimes a} \otimes \mathcal{H}_{\mathbb{Z}}^{\otimes b} \otimes \mathcal{H}_{\mathbb{T}}^{\otimes c} \otimes \mathcal{H}_{\mathbb{Z}_{N_1}} \otimes \cdots \otimes \mathcal{H}_{\mathbb{Z}_{N_d}}$. We highlight that results developed in this chapter can be extended to study such hybrid system context with only tiny modifications (cf. discussion in appendix E).

Connection with duality theory. 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**[7] [188–190, 185, 186, 191, 192]. From a mathematical point of view, it is possible to associate a family of normalizer gates to 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

[6] In this chapter we only discuss stabilizer states but it is easy to adapt our techniques in chapter 3.4 to study codes.

[7] Aspects of this duality and a generalization of it feature also in the circuit models studied in chapters 5-6.

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 $\mathbb{Z}^a \times \mathbb{T}^b \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_c}$, which can be naturally associated to known quantum mechanical systems. As aforementioned, we believe that the results presented in this paper can easily be extended to all groups of the form $\mathbb{Z}^a \times \mathbb{T}^b \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_c} \times \mathbb{R}^d$, which we call "*elementary*", and form a well-studied class of groups known as "*compactly generated abelian Lie groups*" [189]. 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 [190].

4.1.4 Chapter outline

We refer the reader to chapter 1 for a detailed introduction to the **normalizer circuit model** over finite and infinite abelian groups: specifically, see section 1.4 for specific details on the infinite-group case, including a discussion of the most technical infinite-dimensional aspects of these circuits compared to the finite-group setting (s. 1.4.1), a full description of the circuit model (s. 1.4.2) and examples 1.4.3.

In section 4.2 we state the **main result** (theorem 4.1) of this chapter. 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. Finally, in 4.5, we prove our main result.

We refer the reader to chapter 2 for a description of the **classical techniques** that we exploit in the classical simulations of this chapter: namely, see sections 2.2, 2.3 for details on our matrix representations and normal forms for group homomorphisms and quadratic functions, and section 2.4.2 for our classical algorithms for solving linear equations over groups.

4.2 Main result

In our main result (theorem 4.1 below) we show that any polynomial-size normalizer circuit (see c. 1.4.2 for definitions and details on our computational model) over any group of form

$$G = \mathbb{Z}^{\otimes a} \times \mathbb{T}^b \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_c}, \quad \text{with} \quad \mathcal{H}_G = \mathcal{H}_{\mathbb{Z}}^{\otimes a} \otimes \mathcal{H}_{\mathbb{T}}^{\otimes b} \otimes \mathcal{H}_{\mathbb{Z}_{N_1}} \otimes \cdots \otimes \mathcal{H}_{\mathbb{Z}_{N_c}} \tag{4.1}$$

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, 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*.

We specify next how an instance of the computational problem is presented to us.

First, we introduce **standard encodings** that we use to describe normalizer gates. Our encodings are *efficient*, in the sense 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 (4.1) 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 4.1). This polynomial (as opposed to exponential) scaling in m is crucial in our setting, since normalizer gates may act non-trivially on all m registers of the Hilbert space (4.1)—this is an important difference between our computational model (based on normalizer gates) and the standard quantum circuit model [13], where a quantum circuit is always given as a sequence of one- and two-qubit gates.

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

(ii) An automorphism gate U_α is described by what we call a *matrix representation* A of the automorphism α (definition 2.2): an $m \times m$ real matrix A that specifies the action of the map α.

(iii) A quadratic phase gate D_ξ is described by an $m \times m$ real matrix M and an m-dimensional real vector v. The pair (M, v) specifies the action of the quadratic function ξ associated to D_ξ. Here we exploit the normal form for quadratic functions that we gave in theorem 2.1.

In this chapter, 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[8].

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

The existence and efficiency of our standard encodings is guaranteed by the (classical) **theory of matrix representations** and **quadratic functions** that we developed in chapter 2: specifically, because of our existence lemma 2.7 for group-homomorphism matrix representations; our normal form that characterizes the structure of these matrices (lemma 2.8); and our analytic normal forms for bicharacter functions (lemmas 2.9, 2.10) and quadratic functions (theorem 2.1). Because of their quantum applications, these earlier classical results are also main contributions of this thesis.

Lastly, we mention that, in this chapter, 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 (4.1) implies that there exists an infinite number of normalizer gates (namely, of automorphism and quadratic gates, which follows from the our analysis in sections 2.2 and 2.3). This is in contrast with the settings considered in chapter 3, where both the group (4.1) 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).

We state now our main result.

Theorem 4.1 (**Main result**). Let $\mathcal{C}$ be any normalizer circuit over any group $G = \mathbb{Z}^a \times \mathbb{T}^b \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_c}$ as defined in section 1.4.2. Let $\mathcal{C}$ act on a input state $|g\rangle, g \in G_0$ in the designated standard basis $\mathcal{B}_{G_0}$ at time zero, and be followed by a final measurement in the designated basis $\mathcal{B}_{G_T}$ at time T. Then the output probability distribution can be sampled classically efficiently using an infinite-dimensional stabilizer formalism and epsilon-net methods.

We remind the reader that in theorem 4.1 both standard and Fourier basis states of $\mathcal{H}_\mathbb{Z}$ are allowed inputs (cf. section 1.4.2).

In theorem 4.1, the state $|g\rangle$ is described by the group element g, which is encoded as a tuple of m rational[9] numbers of varying size (see table 4.1, row 1). The memory needed to store

[8] Some automorphisms and quadratic functions can only be represented by matrices with irrational entries (cf. the normal forms in sections 2.2,2.3). 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 the stabilizer formalism in this paper 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.)

[9] In this work we do not use floating point arithmetic.

the normalizer gates comprising $\mathcal{C}$ is summarized in table 4.1, row 2. By "*classically efficiently*" it is meant that there exists a classical algorithm (**theorem 4.3**) 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 $\mathcal{C}$) and of all other variables listed in the "bits needed" column of table 4.1), and *polylogarithmically* in the parameters $\frac{1}{\varepsilon}$, Δ that specify the number of points in a (Δ, ε)-*net* (which we introduce below) and their geometrical arrangement.

Input element	Description needs to	Bits needed
Input state $\|g\rangle$	Specify element $g(i)$ of infinite group $\mathbb{Z}$, $\mathbb{T}$	variable
	Specify element $g(j)$ of finite group $\mathbb{Z}_{N_j}$	$\log N_j$
Normalizer circuit $\mathcal{C}$	Specify quantum Fourier transform $\mathcal{F}_{G_i}$	$\log m$
	Specify automorphism gate U_α via A	$m^2 \|\|A\|\|_{\mathbf{b}}$
	Specify quadratic phase gate D_ξ via M, v	$m^2 \|\|M\|\|_{\mathbf{b}} + m\|\|v\|\|_{\mathbf{b}}$

Table 4.1: The input-size in theorem 4.1. Above, $m = a + b + c + d$ denotes the number of primitive factors of G. $\|X\|_{\mathbf{b}}$ denotes the number of bits used to store one single coefficient of X, which is always assumed to be a rational matrix/vector. Formulas in column 3 are written in Big Theta Θ notation and do not include constant factors.

Sampling techniques

We finish this section by saying a few words about the (Δ, ϵ)-net methods used in the proof of theorem 4.1. These techniques are fully developed in sections 4.5 and 4.5.3.

We shall show later (lemma 4.3) that the final quantum state $|\psi\rangle$ generated by a normalizer circuit is always a uniform quantum superposition in any designated basis $\mathcal{B}_{G_t}$, (1.21) at any time step t: i.e., given the most general form of state $|\psi\rangle$, which is

$$|\psi\rangle = \int_X \mathrm{d}g\, \psi(g)|g\rangle, \tag{4.2}$$

where $\mathrm{d}g$ denotes the Haar measure[10] on G_t, $\psi(g)$ are the amplitudes of a normalized wavefunction and X is the support of ψ (i.e., the set of $x \in G_t$ such that $\psi(x) \neq 0$); we show that $|\psi(x)| = |\psi(y)|$ for all $x, y \in X$ at any time step. As a result the final distribution of measurement outcomes of a normalizer circuit is always a flat distribution over some set X.

Moreover, we show in section 4.5.3 that X is always isomorphic to a group of the form $K \times \mathbb{Z}^{\mathbf{r}}$ where K is compact, and that an isomorphism can be efficiently computed: as a result, we see that, although X is not compact, the non-compact component of X inherits a simple Euclidean geometry from $\mathbb{R}^{\mathbf{r}}$. 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

$$\mathcal{N}_{\Delta,\varepsilon} = \mathcal{N}_\varepsilon \oplus \mathcal{P}_\Delta, \tag{4.3}$$

where $\mathcal{N}_\varepsilon$ is an ε-net (definition 4.1) 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 4.2). The algorithm in theorem 4.1 can efficiently construct and sample uniformly from such sets for any ε and $\Delta := \Delta_1, \ldots, \Delta_{\mathbf{r}}$ of our choice: its worst-case running-time is $O(\mathrm{polylog}\, \frac{1}{\varepsilon}, \mathrm{polylog}\, \Delta_i)$, as a function of these parameters. We refer the reader to section 4.5 and theorem 4.3 for more details.

[10] If X is a discrete set, this Haar integral is simply a sum over all elements in G_t.

Treatment of finite-squeezing errors

It follows from the facts that we have just discussed 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* [215]. In a physical implementation (cf. chapter 5), these states can be replaced by physical finitely-squeezed states, whose amplitudes will decay towards the infinite ends[11] of the support set X. This leads to finite-squeezing errors, compared to the ideal scenario.

In this chapter, we consider normalizer circuits to work perfectly in the ideal infinite-squeezing scenario. Our simulation algorithm in theorem 4.1 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.

In the following chapter 5, we will make use of this simulation algorithm to study quantum algorithms based on normalizer circuits. We will also study how information can be represented with finitely-squeezed states in a computation.

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 in this paper), 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 the paper, we sometimes 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 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. From now on, $G^* = \mathbb{T}^a \times \mathbb{Z}^b \times F$ will always denote the uniquely-define dual elementary group that is isomorphic to the character group $\widehat{G}$ of G (cf. chapter 2.1.2 for details about group characters).

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 1.4.1 . 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):

$$X_G(g)\psi(h) := \psi(h - g), \quad \text{for every } g, h \in G, \tag{4.4}$$

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:

$$X_G(g)|h\rangle = |g + h\rangle, \quad \text{for every } g, h \in G. \tag{4.5}$$

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

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 $Z_G(\mu)$:

$$Z_G(\mu)|g\rangle := \chi_\mu(g)|g\rangle, \quad \text{for every } g \in G,\ \mu \in G^*. \tag{4.6}$$

We define a *generalized Pauli operator* of G to be any unitary operator of the form

$$\sigma := \gamma Z_G(\mu) X_G(g) \tag{4.7}$$

where γ is a complex number with unit modulus. We will call the pair (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 chapter 3, 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 $\mu \in G^*$:

$$\begin{aligned} X_G(g)X_G(h) &= X_G(g+h) = X_G(h)X_G(g) \\ Z_G(\mu)Z_G(\nu) &= Z_G(\mu+\nu) = Z_G(\nu)Z_G(\mu) \\ Z_G(\mu)X_G(g) &= \chi_\mu(g) X_G(g) Z_G(\mu) \end{aligned} \tag{4.8}$$

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 [1, 2]. More generally, as we know from [134] and chapter 3, normalizer gates over any finite abelian group G also 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 chapter 1.4, the QFT over G (or any partial QFT) will change the group that labels the designated basis of $\mathcal{H}_G$ 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 4.2), we provide two properties of QFTs.

Lemma 4.1 (**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:

$$Z_{G^*}(g) = \mathcal{F}_G X_G(g) \mathcal{F}_G^\dagger. \tag{4.9}$$

Proof. We show this by direct evaluation of the operator $X_G(h)$ on the Fourier basis states. Using the definitions introduced in section 2.1.2 we can write the vectors in the Fourier basis of G (chapter 1.4) in terms of character functions (definition 2.1): letting $|\mu\rangle$ be the state

$$|\mu\rangle = \int_G \mathrm{d}h\, \overline{\chi_\mu(h)} |h\rangle = \int_G \mathrm{d}h\, \chi_{-\mu}(h)|h\rangle, \tag{4.10}$$

then the Fourier basis of G is just the set $\{|\mu\rangle,\, \mu \in G^*\}$. Now it is easy to derive

$$\begin{aligned} X_G(g)|\mu\rangle &= X_G(g)\left(\int_G \mathrm{d}h \overline{\chi_\mu(h)}|h\rangle\right) = \int_G \mathrm{d}h \overline{\chi_\mu(h)}|g+h\rangle = \int_G \mathrm{d}h' \overline{\chi_\mu(h'-g)}|h'\rangle \\ &= \overline{\chi_\mu(-g)}\left(\int_G \mathrm{d}h' \chi_\mu(h')|h'\rangle\right) = \chi_g(\mu)|\mu\rangle = Z_{G^*}(g)|\mu\rangle. \end{aligned} \tag{4.11}$$

In the derivation we use lemmas 2.2, 2.1 and equation (4.6) 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[12].

Theorem 4.2 (**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 1.4.2. Then the conjugation map $\sigma \to U\sigma U^\dagger$ sends Pauli operators of G to Pauli operators of G', hence, U is a generalized Clifford operator.

Our result generalizes Van den Nest's theorem for finite abelian group normalizer gates (theorem 3.1), which we reviewed in chapter 3.

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 (4.8). 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 $U_\alpha :\ |h\rangle \to |\alpha(h)\rangle$ then

$$U_\alpha X_G(g) U_\alpha^\dagger |h\rangle = |\alpha(\alpha^{-1}(h) + g)\rangle = |h + \alpha(g)\rangle = X_G(\alpha(g))|h\rangle, \tag{4.12}$$

$$U_\alpha Z_G(\mu) U_\alpha^\dagger |h\rangle = \chi_\mu(\alpha^{-1}(h))|h\rangle = \chi_{\alpha^{*-1}(\mu)}(h)|h\rangle = Z_G\left(\alpha^{*-1}(\mu)\right)|h\rangle, \tag{4.13}$$

  where α^* is the dual group automorphism (2.16).

- If U is a quadratic phase gate D_ξ associated with a quadratic function ξ then

$$\begin{aligned} D_\xi X_G(g) D_\xi^\dagger |h\rangle &= \xi(g+h)\overline{\xi(h)}|g+h\rangle = \xi(g)B(g,h)|g+h\rangle \\ &= \xi(g)\chi_{\beta(g)}(h)|g+h\rangle = \xi(g)X(g)Z(\beta(g))|h\rangle, \end{aligned} \tag{4.14}$$

  where, in the second line, β is the group homomorphism in the bi-character normal form of lemma 2.9. Moreover $D_\xi Z_G(\mu) D_\xi^\dagger = Z_G(\mu)$ since diagonal gates commute.

[12] 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" [216, Gottesman] to Clifford algebras.

- If U is the Fourier transform $\mathcal{F}_G$ on the $\mathcal{H}_G$ then

$$\mathcal{F}_G X_G(g)\mathcal{F}_G^\dagger = Z_{G^*}(g), \qquad \mathcal{F}_G Z_G(\mu)\mathcal{F}_G^\dagger = X_{G^*}(-\mu). \tag{4.15}$$

The first identity is the content of lemma 4.1. The second is proved in a similar way:

$$\begin{aligned} Z_G(\mu)|\nu\rangle &= Z_G(\mu)\left(\int_G \mathrm{d}h \chi_{-\nu}(h)|h\rangle\right) = \int_G \mathrm{d}h \chi_{-\nu}(h)\chi_\mu(h)|h\rangle = \int_G \mathrm{d}h \chi_{-(\nu-\mu)}(h)|h\rangle \\ &= |\nu-\mu\rangle = X_{G^*}(-\mu)|\nu\rangle, \end{aligned} \tag{4.16}$$

where we apply (4.10), lemma 2.1 and (4.5,4.6). 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 $G = \mathbb{Z}^a \times \mathbb{T}^b \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_c}$. As explained in section 4.1, our techniques generalize methods given in chapter 3 (which apply to groups of the form $F = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_c}$) and are closely related to the (more general) monomial stabilizer formalism [203].

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 (chapter 3, [1, 2, 24, 3]).

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 $\mathcal{S}_G^\times$ [179, 180]. 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 $|p\rangle$ (1.17) (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

$$\int_{\mathbb{T}} dp|p\rangle \quad \xrightarrow{\text{QFT over } \mathbb{T}} \quad |0\rangle, 0 \in \mathbb{Z}. \tag{4.17}$$

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 4.1). 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 4.2. 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

$$\{\overline{\chi_\mu}(g)Z_G(\mu) : \mu \in G^*\}. \tag{4.18}$$

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^* [189, 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 (4.8)).

□

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:

$$\begin{aligned}
\mathbb{L} &:= \{(\mu, g) \in G^* \times G \ : \ \mathcal{S} \text{ contains a Pauli operator of the form } \gamma Z(\mu)X(g)\};\\
\mathbb{H} &:= \{g \in G \ : \ \mathcal{S} \text{ contains a Pauli operator of the form } \gamma Z(\mu)X(g)\};\\
\mathbb{D} &:= \{\mu \in G^* \ : \ \mathcal{S} \text{ contains a Pauli operator of the form } \gamma Z(\mu)\}
\end{aligned} \tag{4.19}$$

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

$$\mathbb{H} \subseteq \mathbb{D}^\perp \quad \text{(or, equivalently, } \mathbb{D} \subseteq \mathbb{H}^\perp\text{)}, \tag{4.20}$$

which follows from the commutativity of the elements in $\mathcal{S}$ and the definition of annihilator (recall sections 4.3.1 and 2.1.4).

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 structure of this wavefunction.

Lemma 4.3. Every stabilizer state $|\psi\rangle$ over G is a uniform quantum superposition over some subset of the form $s + \mathbb{H}$, where $\mathbb{H} \subset G$ is the label subgroup defined as in (4.19). Equivalently, any stabilizer state $|\psi\rangle$ can be writte in the form

$$|\psi\rangle = \int_{\mathbb{H}} \mathrm{d}h\, \psi(h)|s + h\rangle \tag{4.21}$$

where $\mathrm{d}h$ is the Haar measure over $\mathbb{H}$ and all amplitudes have equal absolute value $|\psi(h)| = 1$. We call the $s + \mathbb{H}$ the *support* of $|\psi\rangle$.

This lemma generalizes corollary 1 in [134].

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

$$U|\psi\rangle = \gamma \int_X \mathrm{d}g\, \chi_\mu(g + h)\psi(g)|g + h\rangle = \int_X \mathrm{d}g\, \psi(g)|g\rangle = |\psi\rangle. \tag{4.22}$$

Recall the definition of $\mathbb{H}$ in (4.19). Comparing the two integrals in (4.22), 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,

$$\textit{for every } g \in X \textit{ it holds } |\psi(g)| = |\psi(g+h)| \textit{ for every } h \in \mathbb{H}. \tag{4.23}$$

Now let $Y \subset X$ denote the subset of points $y \in X$ for which $\psi(y) \neq 0$. Eq. (4.23) implies that Y is a disjoint union of cosets of $\mathbb{H}$, i.e.

$$Y = \bigcup_{\iota \in I} s_\iota + \mathbb{H}, \tag{4.24}$$

where I is a (potentially uncountable) index set, and that $|\psi\rangle$ is of the form

$$|\psi\rangle = \int_Y \mathrm{d}y\, \psi(y)|y\rangle = \int_I \mathrm{d}\iota\, \alpha(\iota)|\phi_\iota\rangle, \tag{4.25}$$

where the states $|\phi_\iota\rangle$ are *non-zero* linearly-independent uniform superpositions over the cosets $x_\iota + \mathbb{H}$:

$$|\phi_\iota\rangle = \int_{\mathbb{H}} \mathrm{d}h\, \phi_\iota(h)|s_\iota + h\rangle \tag{4.26}$$

and $|\phi_\iota(h)| = 1$ for every h. Putting together (4.25) and (4.26) 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 (4.25, 4.26) that I has exactly one element and $Y = s + \mathbb{H}$; as a result, $|\psi\rangle$ is a uniform superposition of the form (4.26). This proves the lemma. □

Lemma 4.4. An element $x \in G$ belongs to the support $s + \mathbb{H}$ of a stabilizer state $|\psi\rangle$ iff

$$D|x\rangle = |x\rangle \quad \text{for all } D \in \mathcal{D}. \tag{4.27}$$

Equivalently, using that $D = \gamma_\mu Z_G(\mu)$ for some $\mu \in \mathbb{D}$ and that γ_μ is determined by μ, we get

$$\mathrm{supp}(|\psi\rangle) = \{x \in G \,:\, \chi_\mu(x) = \overline{\gamma_\mu} \text{ for all } \mu \in \mathbb{D}\}. \tag{4.28}$$

Lemma 4.4 was proven for finite groups in [134] and by us in chapter 3, partially exploiting the monomial stabilizer formalism (MSF) developed in [203]. Since the MSF framework has not been generalized to infinite dimensional Hilbert spaces, the techniques in [203, 134] and chapter 3 can no longer be applied in our present setting[13]. Our proof works in infinite dimensions and even in the case when the Pauli operators (4.4,4.6) have unnormalizable eigenstates.

Proof. Write $|\psi\rangle$ as in (4.21) 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

$$D|\psi\rangle = |\psi\rangle \quad \Longleftrightarrow \quad \int_X \mathrm{d}x\, (\gamma_\mu \chi_\mu(x))\, \psi(x)|x\rangle = \int_X \mathrm{d}x\, \psi(x)|x\rangle, \tag{4.29}$$

[13] The authors believe that the MSF formalism in [203] 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 (4.4,4.6)—seems to be notoriously harder.

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

$$|\phi\rangle := \left(\int_Y \mathrm{d}U\, U\right)|x\rangle = \left(\int_{\mathbb{H}} \mathrm{d}h\, U_h\right)|x\rangle = \int_{\mathbb{H}} \mathrm{d}h\, \xi(\nu_h, h)\chi_{\nu_h}(x+h)|x+h\rangle \mathrm{d}h. \tag{4.30}$$

The last equality in (4.30) 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

$$D|\phi\rangle = D\left(\int_Y \mathrm{d}U\, U\right)|x\rangle = \left(\int_Y \mathrm{d}U\, U\right) D|x\rangle = \left(\int_Y \mathrm{d}U\, U\right)|x\rangle, \tag{4.31}$$

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

$$U_x|\phi\rangle = U_x\left(\int_{\mathbb{H}} \mathrm{d}h\, U_h\right)|x\rangle = \left(\int_{\mathbb{H}} \mathrm{d}h\, U_x U_h\right)|x\rangle = \left(\int_{\mathbb{H}} \mathrm{d}h\, U_{x+h}\right) D_{x,h}|x\rangle \tag{4.32}$$

$$= \left(\int_{\mathbb{H}} \mathrm{d}h'\, U_{h'}\right)|x\rangle = |\phi\rangle \tag{4.33}$$

Hence, using property (a) above, it follows that any arbitrary stabilizer V stabilizes $|\phi\rangle$ as well. □

Corollary 4.1. The sets $\mathbb{H}$ and $\mathrm{supp}(|\psi\rangle) = s + \mathbb{H}$ are closed.

Proof. It follows from (4.28) that $\mathrm{supp}(|\psi\rangle)$ is of the form $x_0 + \mathbb{D}^\perp$. Putting this together with (4.21) in lemma 4.3 it follows that $\mathbb{H} = \mathbb{D}^\perp$. Since any annihilator is closed (lemma 2.4), $\mathbb{H}$ is closed. Since the group operation of G is a continuous map[14], $s + \mathbb{H}$ is closed too. □

[14] This is a fundamental property of topological groups. Consult e.g. [189, 190] for details.

4.5 Proof of theorem 4.1

In this section we prove our main result (theorem 4.1). As anticipated, we divide the proof in three parts. In section 4.5.1, 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 4.5.2 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 [199] and solved with the classical algorithms presented in section 2.4. Finally, in section 4.5.3, we show how to simulate the final measurement of a normalizer computation by developing **net techniques** (based, again, on techniques of section 2.4) to sample from the support of the final state.

4.5.1 Tracking normalizer evolutions with stabilizer groups

As in the celebrated Gottesman-Knill theorem [1, 2] and its existing generalizations (cf. chapter 3), 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 4.2) and that normalizer gates are Clifford operations (lemma 4.5), 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 is the approach we used in chapter 3 to simulate normalizer circuits over finite abelian groups.

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[15], 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 4.2); 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 chapter 3, the group G associated to a normalizer circuits was a parameter that does not change during the computation. In section 4.3.2 we discussed that our setting is now different, as Fourier transforms can change the group that labels the designated basis (theorem 4.2, eq. 4.15); this reflects that groups (4.1) are not autodual.

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

[15] As discussed in section "Relationship to previous work", there are a few simulation methods [138, 44, 159] for Clifford circuits that are not based on stabilizer-groups, but they are more limited than stabilizer-group methods: the Schrödinger-picture simulation in [138] is for non-adaptive qubit Clifford circuits; the Wigner-function simulation in [44, 159] is for odd-dimensional qudit Clifford circuits (cf. also section 4.1).

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 4.2 can be understood as transformations of this group. We show next that the transformations induced 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 4.2). We can keep track of this effect by storing the $2m \times 2m$ identity matrix I_{2m} (the matrix clearly defines a group automorphism of Γ). Moreover, (4.15) 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

$$(\mu(i), g(i)) \in G_i^* \times G_i \quad \longrightarrow \quad (g(i), -\mu(i)) \in G_i \times G_i^*. \tag{4.34}$$

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 $G = \mathbb{T}^a \times \mathbb{Z}^b \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_c}$ with $m = a + b + c$, then all elements of the triple (Λ, M, v) will have $O(\text{poly}\, m)$ 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 4.5, 4.7) 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 (4.19), 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

$$\mathbb{L}(0) = \{(\mu, 0) : \mu \in G(0)^*\}. \tag{4.35}$$

A key observation is that this group can be written as the image of a continuous group homomorphism

$$\Lambda_0 : (\mu, g) \in \Gamma(0) \to (\mu, 0) \in \Gamma(0); \tag{4.36}$$

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):

$$\Lambda_0 = \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} \tag{4.37}$$

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 4.5 (**Evolution of Pauli labels**)**.** There exists a group homomorphism Λ_t from $\Gamma(0)$ to $\Gamma(t)$ satisfying

$$\mathbb{L}(t) = \operatorname{im} \Lambda_t. \tag{4.38}$$

Moreover, a matrix representation of Λ_t can be computed in classical polynomial time, using $O(\text{poly}(m, t))$ basic arithmetic operations.

Proof. We show this by induction. As discussed above, at $t = 0$ we choose Λ_0 as in (4.37). 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 4.2. 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 (4.12)-(4.13) imply

$$\Lambda_{t+1} = \begin{pmatrix} A^{*^{-1}} & 0 \\ 0 & A \end{pmatrix} \Lambda_t. \tag{4.39}$$

  The matrix $A^{*^{-1}}$ can be computed efficiently due to lemmas 2.14 and 2.6.(b).

- Quadratic phase gate D_ξ: suppose that ξ is a B-representation for some bicharacter B (recall section 2.3). Let M be a matrix representation of the homomorphism β that appears in lemma 2.9. Then (4.14) implies

$$\Lambda_{t+1} = \begin{pmatrix} I & M \\ 0 & I \end{pmatrix} \Lambda_t. \tag{4.40}$$

- Partial Fourier transform $\mathcal{F}_{G_i}$: recalling (4.15), we simply have

$$\Lambda(t+1) = S_i \Lambda(t), \tag{4.41}$$

  with

$$S_i = \left(\begin{array}{ccc|ccc} \mathbf{1} & & & \mathbf{0} & & \\ & 0 & & & 1 & \\ & & \mathbf{1} & & & \mathbf{0} \\ \hline \mathbf{0} & & & \mathbf{1} & & \\ & -1 & & & 0 & \\ & & \mathbf{0} & & & \mathbf{1} \end{array}\right) \tag{4.42}$$

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

$$\sigma := \gamma Z(\mu)X(g) \quad \text{and } \tau := \beta Z(\mu)X(g) \tag{4.43}$$

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

$$\mathcal{S} = \{\gamma(\mu, g)Z(\mu)X(g) : \ (\mu, g) \in \mathbb{L}\}. \tag{4.44}$$

Lemma 4.6. 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 $\gamma((\mu_1, g_1) + (\mu_2, g_2))$ of their product operator $\sigma_2\sigma_1$, we obtain

$$\gamma((\mu_1, g_1) + (\mu_2, g_2)) = \gamma(\mu_1, g_1)\gamma(\mu_2, g_2)\overline{\chi_{\mu_2}(g_1)}, \tag{4.45}$$

which implies that γ is quadratic. □

Although it does not follow from lemma 4.6, in our setting, the quadratic function γ will always be *continuous*. As a result, we can apply the normal form given in theorem 2.1 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 4.2) and normalizer gates continuously transform Pauli operators under conjugation; this is rigorously shown using induction in the proof of theorem 4.7.

We will use that these phases of Pauli operators are described by quadratic functions on $\mathbb{L}(t)$ (recall lemma 4.6). In particular, theorem 2.1 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 4.7 (**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 2.1). Moreover, M_t and v_t can be efficiently computed classically with $O(\text{poly}(m, n))$ basic arithmetic operations.

Proof. The proof is similar to the proof of lemma 4.5. 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 4.5 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 (4.12, 4.13) we have

$$\xi_t(\mu, g)Z_G(\mu)X_G(g) \xrightarrow{U_\alpha} \xi_t(\mu, g)Z_G(\mu')X_G(g') \tag{4.46}$$

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 2.14 and 2.6.(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

$$\xi_{t+1}(\mu', g') := \xi_t(\mathbf{A}^{-1}(\mu', g')) = \xi_t(\mu, g). \tag{4.47}$$

The function is manifestly quadratic. By applying lemma 2.13 we obtain

$$M_{t+1} = \mathbf{A}^{-T} M_t \mathbf{A}^{-1}, \qquad v_{t+1} = \mathbf{A}^{-T} v_t + v_{\mathbf{A}^{-1}, M_t}, \tag{4.48}$$

where $v_{\mathbf{A}^{-1}, M_t}$ is defined as $v_{A,M}$ in lemma 2.13.

- **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 $\mathcal{H}_{G_i}$ (4.15), we locally exchange the operators using (4.8), 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

$$\xi_{t+1}(\mu', g') := \xi_t(\mu, g)\, \chi_{(0,\ldots,\mu(i),\ldots,0)}((0,\ldots,g(i),\ldots,0)). \tag{4.49}$$

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

$$M_F := \left(\begin{array}{cc|cc} & & 0 & \\ & \mathbf{0} & & \Upsilon_G(i,i) \\ & & & \mathbf{0} \\ \hline \mathbf{0} & & & \\ & \Upsilon_G(i,i) & & \mathbf{0} \\ & \mathbf{0} & & \end{array}\right), \tag{4.50}$$

where $\Upsilon_G(i, i)$ is the ith diagonal element of Υ_G (2.19). Applying lemma 2.13 we obtain

$$M_{t+1} = \mathbf{A}^{-T}(M_t + M_F)\,\mathbf{A}^{-1}, \qquad v_{t+1} = \mathbf{A}^{-T} v_t + v_{\mathbf{A}^{-1}, M_t + M_F}. \tag{4.51}$$

- **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 (2.9). We know from lemma 2.10 that $M_Q = \Upsilon_G M_\beta$. Using (4.14) 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

$$M' := M_t + \begin{pmatrix} 0 & 0 \\ 0 & M_Q \end{pmatrix} - \begin{pmatrix} 0 & 0 \\ 0 & 2M_Q \end{pmatrix}, \qquad v' := v + \begin{pmatrix} 0 \\ v_Q \end{pmatrix} \tag{4.52}$$

Using lemma 2.13 and $\mathbf{A} = \begin{pmatrix} I & M_\beta \\ 0 & I \end{pmatrix}$ (from the proof of lemma 4.5) we arrive at:

$$M_{t+1} = \mathbf{A}^{-T} M' \mathbf{A}^{-1}, \qquad v' = \mathbf{A}^{-T} v' + v'_{\mathbf{A}^{-1}, M'} \tag{4.53}$$

□

Combining lemmas 4.5 and 4.7, 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 $\mathcal{S}$. The latter is described by the triple (Λ, M, v), the map from $\Gamma(0)$ to Γ is described by Λ, etc.

4.5.2 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 (4.19) and $x \in G$ is any element satisfying conditions (4.27). 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 (4.27). In the case of finite groups G, treated in chapter 3, the approach consisted of first computing a (finite) set of generators $\{D_1, \ldots, D_r\}$ of $\mathcal{D}$. Note that $x \in G$ satisfies (4.27) if and only if $D_i|x\rangle = |x\rangle$ for all i. This gives rise to a finite number of equations. In chapter 3 we 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.

4.5.2.1 Computing $\mathcal{D}$

We want to solve the system of equations (4.28). Our approach will be to reduce this problem to a system of linear equations over a group of the form (2.43) and apply the algorithm in theorem 2.2 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

$$\Lambda = \begin{pmatrix} \Lambda_1 \\ \Lambda_2 \end{pmatrix} \tag{4.54}$$

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 $\Lambda_2 u \equiv 0 \bmod G$ defines a linear system of constrains over a group as in (2.43). This means (because of our algorithm in theorem 2.2) that we can compute in polynomial time a *general solution* of it and, in particular, a matrix representation $\mathcal{E}_\mathbb{D}$ of a group homomorphism $\mathcal{E}_\mathbb{D} : \mathbb{R}^a \times \mathbb{Z}^b \to G^*$ whose image is precisely $\mathbb{D}$, i.e.:

$$\mathbb{D} = \{\mathcal{E}_\mathbb{D} w : \ w \in \mathbb{R}^a \times \mathbb{Z}^b\}.$$

In particular, this means that we can efficiently compute a classical description $\mathcal{E}_\mathbb{D}$ of $\mathbb{D}$.

4.5.2.2 Computing the support $x_0 + \mathbb{H}$

Recalling the support equations (4.28) 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

$$\xi_{M,v}(\mu, 0)\chi_\mu(x_0) = 1, \quad \text{for all } \mu \in \mathbb{D}. \tag{4.55}$$

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 $\mathcal{E}_{\mathbb{D},\text{pad}} := \begin{pmatrix} \mathcal{E}_{\mathbb{D}} \\ 0 \end{pmatrix}$. 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

$$\xi_{M,v}(\mathcal{E}_{\mathbb{D}} w, 0) = \xi_{M',v'}(w) \qquad \text{with } M' := \mathcal{E}^T_{\mathbb{D},\text{pad}} M \mathcal{E}_{\mathbb{D},\text{pad}},\ v' := \mathcal{E}^T_{\mathbb{D},\text{pad}} v. \tag{4.56}$$

- $\chi_{\mathcal{E}_{\mathbb{D}} w}(x_0)$, 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 (4.55) if and only if the quadratic function $\xi_{M',v'}$ is a character and coincides with χ_ϖ. Using lemma 2.10 and theorem 2.1, we can write these two conditions equivalently as:

$$w_1^T M' w_2 = 0 \pmod{\mathbb{Z}}, \quad \text{for all } w_1,\, w_2 \in \mathbb{R}^a \times \mathbb{Z}^b \tag{4.57}$$

$$\mathcal{E}^*_{\mathbb{D}}(x_0) = \mathcal{E}^T_{\mathbb{D},\text{pad}} v \pmod{\mathbb{R}^a \times \mathbb{T}^b}. \tag{4.58}$$

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. Furthermore, and crucially, the second equation is again a *system of linear equations over groups* (section 2.4.2), a general solution $(x_0, \mathcal{E})$ of which can be efficiently computed with our classical algorithm in theorem 2.2.

4.5.3 Sampling the support of a state

In the last section we showed how to efficiently compute a classical description of the (uniform) support of the final state $\text{supp}(|\psi\rangle)$, re-expressing it as the set of solutions $G_{\text{sol}} = x_0 + \text{im}\,\mathcal{E}$ of a linear system over groups (4.58) and using our classical algorithm (theorem 2.2) to find a general solution $(x_0, \mathcal{E})$. In this section, we complete our classical simulation algorithm by devising a classical subroutine to uniformly sample from the solution space G_{sol} of any linear system $\alpha(x) = b$ with variables x in a group

$$G = \mathbb{T}^a \times \mathbb{Z}^b \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_c}. \tag{4.59}$$

More generally, our main result (**theorem 4.3**) is an algorithm to uniformly sample elements from any coset of form $x_0 + \text{im}\,\mathcal{E}$, provided that $(x_0, \mathcal{E})$ is given to us as an input: here, $\mathcal{E}$ denotes an arbitrary matrix representation of a group homomorphism from $\mathbb{R}^\alpha \times \mathbb{Z}^\beta$ to G with known α, β. Throughout the section, we let $H := \text{im}\mathcal{E}$ be the image of $\mathcal{E}$ and denote $m := a+b+c$. Because stabilizer states have uniform support (lemma 4.3), this yields our final classical algorithm to simulate final measurement statistics.

Input of the problem and assumptions: For our algorithm to work, H needs to be a *closed* subgroup (in the topological sense), which is enough since the subgroup $\mathbb{H}$ that defines the

support of a stabilizer state (and we aim to sample) is always closed (corollary 4.1). Below, we use the word "subgroup" as a synonym of "closed subgroup".

A simple heuristic: The coset structure of $\mathrm{supp}(|\psi\rangle)=G_{\mathrm{sol}}$ immediately suggests us a *simple heuristic* to sample this set, which will be the first step towards our algorithm:

(a) Choose a random element $v \in \mathbb{R}^\alpha \times \mathbb{Z}^\beta$ using some efficient classical procedure. This step should be easy since this group is a simple product of a conventional real Euclidean space $\mathbb{R}^\alpha$ and an integer lattice $\mathbb{Z}^\beta$.

(b) Apply the map $v \to x_0 + \mathcal{E}(v)$ to obtain a probability distribution on G_{sol}.

Unfortunately, this straightforward strategy has important caveats and does not yet yield an algorithm to sample G_{sol}. 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 $\mathbb{R}^\alpha \times \mathbb{Z}^\beta$ by the application of a *non-injective* map $\mathcal{E} : \mathbb{R}^\alpha \times \mathbb{Z}^\beta \to G$; this step is prone to create a wild number of *collisions* among samples, about which the heuristic gives no information.

Norms

In the rest of this section we show how to tackle problems (a-b) with a strategy that uses epsilon-net methods. To this end, our first step is to introduce a suitable notion of 2-norm for any group of form $G := \mathbb{Z}^a \times \mathbb{R}^b \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_c} \times \mathbb{T}^d$ analogous to the standard 2-norm $\|\cdot\|_2$ of a real Euclidean space (we denote the group 2-norm simply by $\|\cdot\|_G$): given $g = (g_{\mathbb{Z}}, g_{\mathbb{R}}, g_F, g_{\mathbb{T}}) \in G$,

$$\|g\|_G := \left\|(g_{\mathbb{Z}}, g_{\mathbb{R}}, g_F^{\circlearrowleft}, g_{\mathbb{T}}^{\circlearrowleft})\right\|_2 \tag{4.60}$$

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:

$$\text{if} \quad \|g\|_2 \leq \tfrac{1}{2} \quad \text{then} \quad \|g\|_G = \|g\|_2 \leq \tfrac{1}{2}; \tag{4.61}$$

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 (4.59) 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 chapter 3 —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 from a continuous subgroup H exactly. Instead, we will sample from 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 4.1 (**ε-net**). An ε-net[a] $\mathcal{N}$ 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 $\|h - n\|_G \leq \varepsilon$.

[a]Our definition of ε-net is based on the ones used in [217, 218, 37, 219]. We adopt an additional non-standard convention, that $\mathcal{N}$ must be a subgroup, because it is convenient for our purposes.

The second issue in our setting is the unboundedness of certain subgroups of G *by themselves*. 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" from $\mathcal{N}$ 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 approximately) it is enough to sample uniformly from 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 from H in the sense described above. For this, we need to review some structural properties of the subgroups of groups of the form (4.59)

It is known that any arbitrary closed subgroup H of an elementary group G of the form (4.59) is isomorphic to an elementary group also of the form (4.59) (see [189] 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

$$\mathcal{N}_\varepsilon := \mathcal{N}_{\varepsilon,\text{comp}} \oplus \mathcal{N}_{\varepsilon,\text{free}}. \tag{4.62}$$

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* [220]: i.e. a set $\{b_1, \ldots, b_{\mathbf{r}}\}$ 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:

$$\mathcal{N}_{\varepsilon,\text{free}} = \left\{ n = \sum_{i=1}^{\mathbf{r}} n_i b_i, \ \text{ for some } n_i \in \mathbb{Z} \right\}. \tag{4.63}$$

In view of equation (4.59) we introduce a more general notion of nets that is adequate for sampling from this type of set.

Definition 4.2. Let $\mathcal{N}_\varepsilon$ be an ε-net of H and let $\{b_1, \ldots, b_{\mathbf{r}}\}$ be a prescribed basis of $\mathcal{N}_{\varepsilon,\text{free}}$. Then, we call a **$(\boldsymbol{\Delta}, \boldsymbol{\varepsilon})$-*net*** any finite subset $\mathcal{N}_{\Delta,\varepsilon}$ of $\mathcal{N}_\varepsilon$ of the form

$$\mathcal{N}_{\Delta,\varepsilon} = \mathcal{N}_{\varepsilon,\text{comp}} \oplus \mathcal{P}_\Delta, \tag{4.64}$$

where $\mathcal{P}_\Delta$ denotes the parallelotope contained in $\mathcal{N}_{\varepsilon,\text{free}}$ with vertices $\pm\Delta_1 b_1, \ldots, \pm\Delta_{\mathbf{r}} b_{\mathbf{r}}$,

$$\mathcal{P}_\Delta := \left\{ n = \sum_{i=1}^{\mathbf{r}} n_i b_i, \ \text{ where } n_i \in \{0, \pm 1, \pm 2, \ldots, \pm\Delta_i\} \right\}. \tag{4.65}$$

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 from H in the sense we want. Moreover, the next theorem (a main contribution of this chapter) states that there exist classical algorithms to sample the subgroup H through (Δ, ε)-nets *efficiently*.

Theorem 4.3. Let H be an arbitrary closed subgroup of an elementary group $G = \mathbb{T}^a \times \mathbb{Z}^b \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_c}$. 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 $O(\mathrm{poly}\Big(m, \alpha, \beta, \log N_i, \|\mathcal{E}\|_{\mathbf{b}}, \log \frac{1}{\varepsilon}, \log \Delta_i\Big))$ time and bits of memory.

Again, $\|\mathcal{E}\|_{\mathbf{b}}$ 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 4.3: 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 $\mathbb{R}^\alpha \times \mathbb{Z}^\beta$, 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 $\mathcal{E}_{\mathbb{TR}}$:

$$\|\mathcal{E}_{\mathbb{TR}}\|_{\mathrm{op}}^2 \leq \alpha a |\mathcal{E}_{\mathbb{TR}}|^2 \leq \alpha a |\mathcal{E}|^2. \tag{4.66}$$

The first inequality in (4.66) follows from Schur's bound on the maximal singular value of a real matrix. This bound implies that, if two elements $\chi := (x, z) \in \mathcal{L}$ and $\chi' := (x', z) \in \mathcal{L}$ are $\varepsilon_1\sqrt{\alpha}/2$-close to each other, then

$$\|\mathcal{E}\chi - \mathcal{E}\chi'\|_2 \leq \|\mathcal{E}_{\mathbb{TR}}\|_{\mathrm{op}} \|x - x'\|_2 \leq \tfrac{1}{2}\alpha\sqrt{a}|\mathcal{E}|\varepsilon_1 \tag{4.67}$$

(In the first inequality, we apply the normal form in lemma 2.8.) 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 (4.61); it follows that $\mathcal{N}$ is an ε-net of H 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 $Q \cong \mathcal{L}/\ker\mathcal{E}$ and a matrix representation of the group isomorphism[16] $\mathcal{E}_{\mathrm{iso}} : Q \to \mathcal{N}$. 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. The first steps of our subroutine are described next.

1. **Set precision.** Choose ε_1 so that $\mathcal{N}$ is an ε-net using the above bounds.

2. **Put $\mathcal{L}$ in standard form.** Note that $\mathcal{L}$ is isomorphic to the discrete finite-generated abelian group $\mathcal{L}' := \mathbb{Z}^{\alpha+\beta}$ via an isomorphism $\phi : \mathcal{L}' \to \mathcal{L}$ with matrix representation $\varepsilon_1 I_\alpha \oplus I_\beta$. In order to apply the algorithms in theorem 2.2, we "absorb" the ε_1 parameter into the map ϕ and replace $\mathcal{L}$ with $\mathcal{L}'$, and $\mathcal{E}$ with the map $\mathcal{E}' := \mathcal{E}(\varepsilon_1 I_\alpha \oplus I_\beta)$.

[16] Q and $\mathcal{N}$ are isomorphic due to the first isomorphism theorem [189].

3. **Take quotient.** Apply algorithm 3 in theorem 2.2 to compute an efficient decomposition $Q' = \mathbb{Z}_{\sigma_1} \times \ldots \times \mathbb{Z}_{\sigma_{\mathbf{a}}} \times \mathbb{Z}^{\mathbf{b}} \cong$ of the quotient $\mathcal{L}'/\ker \mathcal{E}' \cong Q'$ and a new matrix representation $\mathcal{E}'_{\text{iso}}$ of the isomorphism $\mathcal{E}'_{\text{iso}} : Q' \to \mathcal{N}$. Our earlier result says that this step can be implemented in time at most polynomial in the variables m, α, β, $\log N_i$, $\|\mathcal{E}\|_{\mathbf{b}}$, and $\log \frac{1}{\varepsilon}$.

The above steps procedure outputs a new classical representation $(x_0, \mathcal{E}'_{\text{iso}})$ of the support $\text{supp}(|\psi\rangle) = x_0 + \mathcal{E}'_{\text{iso}}$ with the remarkable property that now $\mathcal{E}'_{\text{iso}}$ is invertible., Moreover, since the matrix $\mathcal{E}'_{\text{iso}}$ acts isomorphically on Q', it follows that the epsilon-net subgroup $\mathcal{N}$ is a direct sum of cyclic subgroups generated by the columns of $\mathcal{E}_{\text{iso}}$:

$$\mathcal{N} = \langle f_1 \rangle \oplus \cdots \oplus \langle f_{\mathbf{a}} \rangle \oplus \langle b_1 \rangle \oplus \cdots \oplus \langle b_{\mathbf{b}} \rangle, \tag{4.68}$$

where f_i, b_j stand for the (i)th and the $(\mathbf{a}+j)$th column of $\mathcal{E}_{\text{iso}}$; via isomorphism, it must also hold that the elements f_is (resp. b_j) generate the compact subgroup $\mathcal{N}_{\text{comp}}$ (resp. form a $\mathbb{Z}$-basis of $\mathcal{N}_{\text{free}}$). Finally, taking $\{f_i\}$ (resp. $\{b_j\}$) as default generating-set (resp. default basis) of $\mathcal{N}_{\text{comp}}$ and $\mathcal{N}_{\text{free}}$, we select a parallelotope $\mathcal{P}_\Delta$ of the form (4.65) with some desired $\Delta = (\Delta_1, \ldots, \Delta_{\mathbf{b}})$. This procedures specifies a net $\mathcal{N}_\Delta = \mathcal{N}_{\text{comp}} \oplus \mathcal{P}_\Delta$ that can be efficiently represented with $O(\text{poly}\Big(m, \alpha, \beta, \log N_i, \|\mathcal{E}'_{\text{iso}}\|_{\mathbf{b}}, \log \frac{1}{\varepsilon} \log \Delta_i\Big))$ bits of memory (by keeping track of the generating-sets of $\mathcal{N}$ and the numbers Δ_i). Moreover, we can efficiently sample from $\mathcal{N}_\Delta$ uniformly and collision-freely by generating random strings of the form

$$\sum_{i=1}^{\mathbf{a}} x_i f_i + \sum_{j=1}^{\mathbf{b}} y_j b_j, \tag{4.69}$$

where $x_i \in \mathbb{Z}_{\sigma_i}$ and $y_j \in \{0, \pm 1, \ldots, \pm\Delta_j\}$. This completes the proof.

Chapter 5

The computational power of normalizer circuits over black box groups

In this chapter we present a precise connection between Clifford circuits, Shor's factoring algorithm and several other famous quantum algorithms with exponential quantum speed-ups for solving abelian hidden subgroup problems. We show that all these different forms of quantum computation belong to a common new restricted model of quantum operations that we call *black-box normalizer circuits*. To define these, we extend the model of normalizer circuits of chapters 3-4 where normalizer gates could be quantum Fourier transforms, group automorphism and quadratic phase gates associated with a (finite or infinite) abelian group G. While earlier G was always given in an explicitly decomposed form, in this chapter we remove this assumption and allow G to be a black-box group [91]. In contrast with standard normalizer circuits, which we showed to be efficiently classically simulable, we find that black-box normalizer circuits are powerful enough to factorize and solve classically-hard problems in the black-box setting. We further set upper limits to their computational power by showing that decomposing finite abelian groups is complete for the associated complexity class. In particular, solving this problem renders black-box normalizer circuits efficiently classically simulable by exploiting the generalized stabilizer formalism of chapters 3-4. Lastly, we employ our connection to draw a few practical implications for quantum algorithm design: namely, we give a no-go theorem for finding new quantum algorithms with black-box normalizer circuits, a universality result for low-depth normalizer circuits, and identify two other complete problems.

This chapter is based on [93] (joint work with Cedric Yen-Yu Lin and Maarten Van den Nest).

5.1 Introduction

In this chapter, we introduce ***black-box normalizer circuits***, a *new restricted family* of quantum operations, and characterize their computational power. Our new model extends the classes of normalizer circuits over abelian groups of chapters 3-4 as explained next. In previous chapters, normalizer circuits acted in high and infinite dimensional systems associated with an abelian group G: in our construction, we associated G with a Hilbert space $\mathcal{H}_G$ with a standard basis $\{|g\rangle\}_{g\in G}$ labeled by G elements. Furthermore, previously, the group G was assumed to be given in an explicit factorized form, which endows the Hilbert space of the computation with a tensor-product structure:

$$G = \mathbb{Z}^a \times \mathbb{T}^b \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_c} \quad \longleftrightarrow \quad \mathcal{H}_G = \mathcal{H}_{\mathbb{Z}}^{\otimes(a+b)} \otimes \mathcal{H}_{\mathbb{Z}_{N_1}} \otimes \cdots \otimes \mathcal{H}_{\mathbb{Z}_{N_c}}; \tag{5.1}$$

above, $\mathbb{Z}$ is the group of *integers*, $\mathbb{Z}_N$ the group of integers modulo N, and $\mathbb{T}$ is the *circle group*, consisting of angles from 0 to 1 (in units of 2π) with the addition modulo 1. The Hilbert space $\mathcal{H}_{\mathbb{Z}}$ has a standard basis labeled by integers ($\mathbb{Z}$ basis) and a Fourier-basis labeled by angles ($\mathbb{T}$ basis). A *normalizer circuit over* G was a circuit built of three types of normalizer gates: Quantum Fourier transforms over G, group automorphism gates and quadratic phase gates.

With these definitions, we saw (section 1.3.1) that n-qubit Clifford circuits are examples of normalizer circuits over the group $\mathbb{Z}_2^n$.

In chapters 3-4 we showed that, despite containing arbitrary numbers of QFTs, which play an important role in Shor's algorithms [4], and entangling gates (automorphism, quadratic phase gates), normalizer circuits can be *efficiently simulated* by classical computers. For this, we exploited an extended *stabilizer formalism* over groups to track the evolution of normalizer circuits, thereby generalizing the celebrated Gottesman-Knill theorem [1, 2].

The key new element in the present chapter are normalizer circuits that can be associated with abelian ***black-box groups*** [91], which we may simply call "black-box normalizer circuits". A group $\mathbf{B}$ (always abelian in this work) is a black-box group if it is *finite*, its elements are uniquely encoded by strings of some length n and the group operations are performed by a black-box (the *group black box*) in one time-step. We define *black-box normalizer circuits* to be a normalizer circuits associated with groups of the form $G = G_{\text{prev}} \times \mathbf{B}$, with G_{prev} is of form (5.1).

The ***key new feature*** in this chapter is that the black-box group $\mathbf{B}$ is *not* given to us in a factorized form. This is a subtle yet tremendously important difference: although such a decomposition *always* exists for any finite abelian group (chapter 1, theorem 1.1), 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[1] is $\mathbb{Z}_N^\times$, the multiplicative group of integers modulo N, which plays an important role in Shor's factoring algorithm [4]. With some abuse of notation, we call any such group also a "black-box group"[2].

5.1.1 Main 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 (chapters 3- 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*** over all known classical algorithms. Namely, the following algorithms are examples of black-box normalizer circuits.

[1] Knowing $\mathbf{B} \cong \mathbb{Z}_{d_1} \times \ldots \times \mathbb{Z}_{d_m}$ implies that the order of the group is $|G| = d_1 d_2 \cdots d_m$. Hardness results for computing orders [91, 221] 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 [92].

[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.

- **Discrete logarithm.** Shor's discrete-log quantum algorithm [4] is a normalizer circuit over $\mathbb{Z}_{p-1}^2 \times \mathbb{Z}_p^\times$ (theorem 5.1, section 5.3.1).
- **Factoring.** We show that a hybrid infinite-finite dimensional version of Shor's factoring algorithm [4] 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 5.2, 5.3). 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 5.4). Last, we further *conjecture* that only normalizer circuits over infinite groups can factorize (conjecture 5.1).
- **Elliptic curves.** The generalized Shor's algorithm for computing discrete logarithms over an elliptic curve [94–96] can be implemented with black-box normalizer circuits (section 3.2); 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 [97, 98] is a combination of several types of black-box normalizer circuits. Furthermore, we present a new *extended* quantum algorithm building upon Cheung-Mosca's that finds even more information about the structure of the group and is also normalizer-circuit based (section 5.3.4).
- **Hidden subgroup problem.** Deutsch's [50], Simon's [51] and, in fact, all quantum algorithms that solve abelian hidden subgroup problems [52, 53, 49, 54–58], 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 5.3.3). 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 5.3.3). 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 [60], RSA [59] and elliptic curve cryptography [61, 62].

2. **Group decomposition is *as hard as* simulating normalizer circuits.** Another main contribution of this work 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 (chapters 3- 4), this result suggests that the computational power of normalizer circuits originates *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 5.6), 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 work, we provide an 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 our extended Cheung-Mosca algorithm and classical post-processing. This implies (theorem 5.7) that new *exponential* speed-ups cannot be found without changing our setting (we discuss how the setting might be changed in the discussion, section 5.1.4). 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 [4, 94–98, 50–53, 49, 54–58] (section 5.3): 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 5.6): these are the (afore-mentioned) 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 chapters 2, 3, 4).

5.1.2 The link between Clifford circuits and Shor's algorithm

The results in this chapter together with those previously obtained in chapters 3-4 (see also [134]) 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 [1, 2, 24, 3, 135, 133, 136, 222, 138, 137, 139]. However, from the point of view of this chapter, both turn out to be *intimately related* in that they both 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, has also some sensible consequences for the theory of quantum computing. One that follows from theorem 5.6, 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" [1, 2]. 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

[3]Note that throughout this chapter we always work at a high-level of abstraction (algorithmically speaking),

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 this chapter, 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).

As a side remark, we regard it a memorable curiosity that replacing decomposed groups with black-box groups not only renders the simulation methods in chapters 3-4 inefficient (this is, in fact, something to be expected, due to the existence of hard computational problems related to black-box groups), but it is also precisely this modification that suddenly bridges the gap between Clifford/normalizer circuits, Shor's algorithms, Simon's and so on.

Finally, it is mathematically elegant to note that all normalizer circuits we have studied are related through the so-called **Pontryagin-Van Kampen duality** [188–190, 185, 186, 191, 192], which states that all locally-compact abelian (LCA) groups are dual to their character groups. The role of this duality in the normalizer circuit model was discussed in chapter 4.

5.1.3 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 investigated before this thesis.

The hidden subgroup problem (HSP) has played a central role in the history of quantum algorithms and has been extensively studied before our thesis. The abelian HSP, 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 [50–53, 49, 54–58]. Its best-known generalization, the non-abelian HSP (which we investigate in chapter 6), has also been heavily investigated due to its relationship to the graph isomorphism problem and certain shortest-vector-lattice problems [100, 223, 101–117] (see also the reviews [118, 119, 9] and references therein).

The notion of black-box group, which is a key concept in our setting, was first considered by Babai and Szemerédi in [91] and has since been extensively studied in classical complexity theory [224–227, 221]. In general, black-box groups may not be abelian and do not need to have uniquely represented elements [91]; 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 [97, 98, 228] and the non-abelian group setting [229, 105, 113, 230, 231, 112, 232, 233]. Except for a few exceptions (cf. [229, 228]) most quantum results have been obtained for uniquely-encoded black-box groups.

---

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 our classical simulation is at most polynomial in the input-size).

5.1.4 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 the work in this chapter.

In this work, 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 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 interesting 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 our methods in chapters 4-5 can be extended, e.g., to efficiently simulate normalizer circuits over groups of the form $\mathbb{R}^a \times \mathbb{Z}^b \times \mathbb{T}^c \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_d} \times \mathbf{B}$, with additional $\mathbb{R}$ factors, once we know how to decompose $\mathbf{B}$ (see also our discussion in chapter 4). 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 [234, 235], which may, informally, be understood as "infinite black-box groups"[4] We expect, in fact, that known quantum algorithms for finding hidden periods and hidden lattices within real vector spaces [236–239] and/or or infrastructures [234, 235] (e.g., Hallgren's algorithm for solving Pell's equation [236, 237]) 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 chapter[5]. Some open questions in this quantum algorithm subfield have been discussed in [235].

A second 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 [100, 223, 101–117]. This direction will be explored in the last chapter of this thesis (chapter 6) where we develop a possible nonabelian model of normalizer circuits and use it to devise new efficient quantum algorithms for the so called normal Hidden Subgroup Problem [223].

A third possible direction to investigate would be whether different models of normalizer circuits could be constructed over ***algebraic structures that are not groups***.

Our results in chapter 6 will also make significant progress in this direction: therein, we consider (in general) normalizer circuit models over so-called *abelian hypergroups*, which generalize abelian groups, and used them to develop the first provably-efficient quantum algorithms for a hypergroup extension of the hidden subgroup problem.

Furthermore, one could, for instance, consider sets with *less algebraic structure* than groups, like semi-groups. In this regard, we highlight that a quantum algorithm for finding discrete logarithms over finite semigroups was recently given in [240]. 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 [241].

Lastly, we mention some open questions suggested by the work of this chapter.

In this work, we have not investigated the computational complexity of black-box normalizer

[4] 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. [234, 235] 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.

[5] No such treatment is needed in this work, since we study quantum algorithms for finding hidden structures in *discrete* groups.

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 non-adaptive Clifford circuits with standard basis inputs and measurements is $\oplus\mathbf{L}$ [133], which is believed to be a strict subclass[6] of $\mathbf{P}$. The second is that finite-dimensional normalizer circuits are unable of implementing classical boolean functions coherently in various settings (see [134] and lemma 3.5 in chapter 3).

Finally, one may study whether considering more general types of inputs, measurements 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 [43, 139]. Concerning measurements, the authors believe that allowing, e.g. adaptive Pauli operator measurements (in the sense of chapter 3) 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 we showed (chapter 3) that adaptive normalizer circuits over finite abelian groups are also efficiently classically simulable with stabilizer techniques. 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 [40, 42]. The possibility of obtaining intermediate hardness results if non-adaptive yet also non-Pauli measurements are allowed (in the lines of [144] or [139, theorem 7]) remains also open.

5.1.5 Chapter outline

In section 5.2 we introduce our normalizer-circuit models over black-box groups. In section 5.3 we show how the quantum algorithms in result 1 above are examples of black-box normalizer circuits[7]. In section 5.4 we give our first completeness result and our no-go theorem (results 2-3). In section 5.5 we present our universality result. Finally, in section 5.6 we study additional complete problems (result 5).

5.2 Black-box groups and black-box normalizer circuits

In this section we introduce abelian black-box groups and present models of black-box group normalizer circuits; the latter generalize our earlier models in chapters 1, 3-4.

5.2.1 Decomposed groups and black-box groups

The most general groups we consider in this chapter are abelian groups of the form

$$G = \mathbb{Z}^a \times \mathbb{T}^b \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_c} \times \mathbf{B}. \tag{5.2}$$

where the parameters a, b, $N_1, \cdots, N_c$ are arbitrary integers of unbounded size and $\mathbf{B}$ is an arbitrary (finite) *abelian* black-box group. Following the nomenclature of chapters 3-4, $\mathbb{Z}$ denotes the (infinite, discrete) group of integers under addition, $\mathbb{T}$ is the (infinite, continuous) group of angles in the interval $[0, 1)$ under addition modulo 1 and $\mathbb{Z}_{N_i}$ is the (finite) group of integers modulo N_i.

Note that the key difference with earlier chapters is the presence of a black-box group $\mathbf{B}$. In terms of computational complexity, there is an stark separation between decomposed abelian groups and black-box groups, which is discussed next.

[6] This is the class of problems solvable by classical poly-size circuits of NOT and CNOT gates [133].

[7] For the sake of conciseness, our results about quantum algorithms to compute discrete-logarithms over elliptic-curves (which require a brief introduction to the latter abstract groups) are given in appendix 3.2.

Decomposed abelian groups: A finite abelian group G is *decomposed* if it is of the form

$$G = \mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2} \times \cdots \times \mathbb{Z}_{N_k} \tag{5.3}$$

and the positive integers $k, N_1, \cdots, N_k$ are given to us as a description of G. By the fundamental theorem of finite abelian groups (chapter 0, theorem 1.1), any finite group can be put in the form (5.3) via isomorphism. Yet finding such a decomposition for a group G may be difficult in practice. As a key example, there is currently not known efficient classical algorithm to decompose the multiplicative group[8] $\mathbb{Z}_N^\times$ of integers modulo N into cyclic subgroups. In fact, the latter problem has long been believed to be classically hard even in the simple case $N = pq$ for p, q prime: in this case, $\mathbb{Z}_{pq}^\times \cong \mathbb{Z}_{p-1} \times \mathbb{Z}_{q-1}$ and decomposing $\mathbb{Z}_{pq}^\times$ becomes at least as hard as *factoring* pq and, hence, breaking the ubiquitous RSA cryptosystem [59]. More generally, decomposing $\mathbb{Z}_N^\times$ is known to be polynomial time equivalent to factoring [92]. In the quantum case, however, Cheung and Mosca gave an algorithm [97, 98] to decompose any finite abelian group.

Black box groups: In equation (5.2), the factors $\mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_c}$ 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 $\mathbf{B}$.

In this chapter, we define a *black-box group* $\mathbf{B}$ [91] 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-bock, 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 black box 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 work will be assumed to be *abelian*. Although we only consider finite abelian black-box groups, we stress now, that the only (albeit subtle) difference between these groups and the explicitly decomposed finite abelian groups in chapter 3 is that, for black-box groups, we assume no knowledge of a decomposition (5.3). 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 $\mathrm{poly}(n)$-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 [198]; the same algorithm tells us whether a given integer $z \in \mathbb{Z}$ belongs to $\mathbb{Z}_N^\times$ (i.e., whether z is coprime to N); on the other hand, decomposing $\mathbb{Z}_N^\times$ is as hard as factoring [92]. Last, note that a generating set of $\mathbb{Z}_N^\times$ can be efficiently found by uniformly sampling random integers from $\{1, \ldots, N\}$ because $|\mathbb{Z}_N^\times|/|\mathbb{Z}_N| \in \Omega(1/\log\log N)$ [242] and due to the following lemma.

Lemma 5.1. For any uniquely-encoded black-box group $\mathbf{B}$ with encoding length n, if it holds that $|\mathbf{B}|/2^n \in \Omega(1/\mathrm{poly}(n))$ (i.e., if the encoding used does not incur into superpolynomial overhead), then a generating set of $\mathbf{B}$ can be found in probabilistic polynomial-time by sampling

[8] Recall that $\mathbb{Z}_N^\times$ is formed of integers relatively prime to N multiplied modulo N.

bit-strings in $\{0,1\}^n$ uniformly at random and rejecting those not that are not elements of $\mathbf{B}$.

Proof. Because $|\mathbf{B}|/2^n \in \Omega(1/\text{poly}(n))$, we obtain an uniformly-random element of $\mathbf{B}$ after $T \in O(\text{poly}\, N)$ trials with $\Omega(1-c^T)$ probability for some constant $c \in (0,1)$ (via the Chernoff-Hoeffding bound [243]). Furthermore, by uniformly sampling $t \in \Theta(\log |G|)$ elements $g_1, \ldots, g_t$ from any finite group G, we obtain a generating-set with probability exponentially close to 1. To see this, note that if $G_i := \langle g_1, \ldots, g_i \rangle$ is a proper subgroup of G, then $g_{i+1} \in G$ belongs to G_i with a small probability $|G_i|/|G| \leq 1/2$. Further if $g_{i+1} \notin G_i$, then $|G_{i+1}|/|G_i| \geq 2$. Hence, the cardinality $|G_t|$ converges exponentially fast to $|G|$ in t. □

5.2.2 Black box normalizer circuits

We now define families of *normalizer circuits over any group* G of form (5.2) with Hilbert space

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

which we view as the physical system of $m := a + b + c + 1$ computational registers. Because, mathematically speaking, $\mathbf{B}$ is equivalent to some group $\mathbb{Z}_{D_1} \times \ldots \times \mathbb{Z}_{D_m}$ (via some isomorphism), the definitions of *normalizer gates* over G in this chapter will be identical to previous ones (chapter 1). Consequently, in this chapter we adopt the model of normalizer circuit over infinite groups from chapters 1-4 with only a few minor *modifications*—cf. (a-b-c-d) below—, which are needed to take some aspects of our new setting into consideration. In short, modifications below are related to, mainly, two facts: (i) now less information is given to us about the structure of the black-box group $\mathbf{B}$; (ii) in this chapter, precision errors need to be handled with more care than before since we aim at characterizing the power of black-box normalizer circuit model rigorously in terms of computational complexity classes.

(a) Black-box designated bases: As in chapter 1.4, all *designated bases* (ditto for input states and final measurements[9]) of a normalizer computation on a Hilbert space $\mathcal{H}_G$ are labeled by a group that is either fixed along the computation (when G is a finite group) or may change via the action of infinite-group QFTs. Each designated basis $\mathcal{B}_{G'}$ is parametrized by a group G' from the following family:

$$G' = G'_1 \times \cdots \times G'_{a+b} \times \mathbb{Z}_{N_1} \otimes \ldots \mathbb{Z}_{N_c} \times \mathbf{B} \quad \text{where } G'_i \in \{\mathbb{Z}, \mathbb{T}\}, \tag{5.4}$$

$$\mathcal{B}_{G'} := \{|g\rangle := |g(1)\rangle \otimes \ldots |g(m)\rangle, \quad g = (g(1), \ldots, g(m)) \in G'\}. \tag{5.5}$$

where we simply adapted earlier formulas to consider $\mathbf{B}$. Note that, in the case of the (finite) black box group $\mathbf{B}$, the Hilbert space $\mathcal{H}_{\mathbf{B}}$ has a (unique) standard basis $\{|\mathbf{b}\rangle\}$ where $\mathbf{b}$ ranges over all elements of $\mathbf{B}$. (It follows that $\mathcal{H}_{\mathbf{B}}$ is $|\mathbf{B}|$-dimensional.) Hence, the existence of multiple designated bases is, again, an infinite-dimensional feature as in chapter 1.4 and *does not* come from the black-box. In fact, though multiple designated bases will play a role in some of the quantum algorithms we consider (see, e.g., the factoring algorithm in section 5.3.2) they will now show up e.g. in Shor's discrete log quantum algorithm (section 5.3.1).

(b) Black-box normalizer gates: As in chapter 1, a normalizer circuit over G is a sequence of automorphism gates, quadratic phase functions and QFTs over any group G'. However, in this chapter we will *not allow* QFTs to act on the black box subspace $\mathcal{H}_{\mathbf{B}}$ of the total system $\mathcal{H}_G$. This restriction is natural because, although $\mathcal{H}_{\mathbf{B}}$ has a mathematically well-defined Fourier basis, it is not currently known how to implement its associated QFT without decomposing the black-box group first[10].

[9] As in chapter 1.4, we only consider terminal measurements performed at the end of the computation.

[10] To our best knowledge, all existing QFT-based quantum algorithms exploit only those abelian-group QFTs that act on (explicitly) factorized systems of the form $\mathcal{H}_{\mathbb{Z}_{N_1}} \otimes \cdots \otimes \mathcal{H}_{\mathbb{Z}_{N_c}}$ for this reason.

(c) Classical encodings for black-box normalizer gates

Because of the presence of black box groups in $G = \mathbb{Z}^a \times \mathbb{T}^b \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_c} \times \mathbf{B}$, it is no longer possible in this chapter to use the classical encodings of chapters 3-4 to represent automorphism and quadratic-phase gates since the latter crucially exploited that G was given in a fully decomposed form[11]. (This issue does not affect QFTs since we allow only decomposed-group ones.)

For the above reason, throughout this chapter, we assume that any *automorphism gate* U_α and *quadratic phase gate* D_ξ over G can be given to us in a more general format, namely, as a black-box quantum gates (i.e., an *oracle*) that can be implemented either at unit cost or by a poly-size quantum algorithm (that is explicitly given to us). Furthermore, we make some additional simplifying assumptions about the associated classical functions α, ξ, which we will assume *efficiently computable rational functions*, which have the following restrictions.

1. **Rational.**[12] An automorphism/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 $\xi(g) = \exp(2\pi \mathrm{i}\, q(g))$ 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 *group black box* 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 $q_i(g) = q(g)$.

   In certain cases (see section 5.3) 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 group black box (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 $i + n_{\mathrm{out}}$. 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[13]. 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 imple-

[11] Note that, by construction, one cannot use, our earlier matrix representations encoding to represent group automorphisms over a non-decomposed $\mathbf{B}$; similar issues affect our prior encodings for quadratic-phase gates.

[12] We expect this assumption not to be essential, but it simplifies our proofs by allowing us to use exact arithmetic operations. Our stabilizer formalism in chapter 4 can still be applied if the functions α, ξ are not rational, and we expect some version of the simulation result (theorem 5.6) to hold even when transcendental numbers are involved (taking carefully into account precision errors). It is a good question to explore whether an exact simulation result may hold for algebraic numbers [244].

[13] 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 chapter 2). The number n_{out} parametrizes the precision needed to compute the function. This assumption might be weakened if a treatment for precision errors is incorporated in the model.

mentable as well[14], by poly(m, i, n_{out})-size quantum circuits comprising at most poly(m, i, n_{out}) quantum queries of the group black box. 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 the next section 5.3, 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 [198]. 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 chapter 1; for decomposed finite group $\mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_c}$. It was also shown in [134] that all normalizer gates over such groups can be efficiently implemented.

(d) Precision requirements

Finally, recall that in the model of quantum circuits we use, 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 $|p\rangle = \sum \overline{e^{2\pi i z p}}|z\rangle$; 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 work (namely, the order-finding algorithm in theorem 5.2), 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 finite 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 $V_{i,d} = \text{span}\{\left|\frac{i}{d} + \Delta\right\rangle \text{ with } |\Delta| < \frac{1}{2d}\}$. 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 $\{V_{i,d}\}_i$ can be reliably distinguished.

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

[14] Recall that, in finite dimensions, the gate cost of implementing a classical function α as a quantum gate is at most the classical cost [13] and that computing q efficiently is enough to implement ξ using phase kick-back tricks [95]. We expect these results to extend to infinite dimensional systems of the form $\mathcal{H}_{\mathbb{Z}}$.

Definition 5.1 (**Efficient use of precision**). A *quantum algorithm* that uses d-approximate Fourier states to solve a computational problem with input size n is said to use an *efficient* amount of precision if and only if $\log d$ 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 $\log(m_{\max}) \in O(\operatorname{poly} n)$ do not play a role in the computation.[a]

[a] 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 chapter (cf. theorem 5.3).

5.3 Quantum algorithms

5.3.1 The discrete logarithm problem over $\mathbb{Z}_p^\times$

In this section we consider the discrete-logarithm problem studied by Shor [4]. 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 [4, 7] for this problem and prove our first result.

Theorem 5.1 (**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 5.1 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 [60], 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 5.1 (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 5.3.4): 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*.

5.3.2 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 $\mathbf{B}$ [91] 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 factor large composite numbers, and thus break the widely used RSA public-key cryptosystem [59].

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[15] $a^r = 1$). 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 5.2 (Order finding over $\mathbf{B}$). Let $\mathbf{B}$ be a finite abelian black-box group and $\mathcal{H}_{\mathbf{B}}$ its associated Hilbert space. 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 $c\text{-}V_a$ the unitary that performs V_a on $\mathcal{H}_{\mathbf{B}}$ controlled on the value of an ancillary register $\mathcal{H}_{\mathbb{Z}}$:

$$|m, x\rangle \quad \xrightarrow{c-V_a} \quad |m, a^m x\rangle, \qquad \text{for any } m \text{ in } \mathbb{Z}.$$

Assume that we can query an oracle that implements $c\text{-}V_a$ 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 5.2, 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 5.2.2).

Proof. 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 $\text{Out}_a = \{\frac{k}{|a|} : k \in \mathbb{Z}\}$ 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 [4] 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 5.2 (Hybrid order finding with infinite precision).

Input. A black box (finite abelian) group $\mathbf{B}$, and an element $a \in \mathbf{B}$.

Output. The order $r := |a|$ of a in $\mathbf{B}$, i.e. the least positive integer r such that $a^r = 1$.

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 $\mathbb{T} \times \mathbf{B}$.

[15] Since $\mathbf{B}$ is finite, the order $|a|$ is a well-defined number.

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:
$$\sum_{m\in\mathbb{Z}} |m\rangle|1\rangle \quad \xrightarrow{c\text{-}V_a} \quad \sum_{m\in\mathbb{Z}} |m, a^m\rangle. \tag{5.6}$$
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[a] group automorphism, since
$$\begin{aligned}\alpha\left((m,x)(n,y)\right) = \alpha(m+n, xy) &= (m+n, (a^{m+n})(xy)) \\ &= (m+n, (a^m x)(a^n y)) = (m, a^m x)(n, a^n y) \\ &= \alpha(m,x)\alpha(n,y).\end{aligned} \tag{5.7}$$

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
$$\left(\sum_{j\in\mathbb{Z}} |s + jr\rangle\right) |a^s\rangle. \tag{5.8}$$
After discarding $\mathcal{H}_{\mathbf{B}}$ 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 labeled by the circle group $\mathbb{T}$. The periodic state $\sum |s + jr\rangle$ in the dual $\mathbb{T}$ basis reads [187]
$$\sum_{k=0}^{r-1} \mathrm{e}^{2\pi\mathrm{i}\frac{sk}{r}} \left|\frac{k}{r}\right\rangle \tag{5.9}$$

6. Measure $\mathcal{H}_{\mathbb{Z}}$ in the Fourier basis (the basis labeled by $\mathbb{T}$). Since we assume that the initial state of the computation can be as close to $|0\rangle$ as we wish, the wavefunction of the final state (5.9) 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.

7. **Classical postprocessing:** Repeat steps 1-7 a few times and use a (classical) continued-fraction expansion algorithm [13, 205] to extract the order r from the randomly sampled multiples $\{k_i/r\}_i$. 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.

[a] This is vacuously true: since the group $G := \mathbb{Z} \times \mathbf{B}$ is discrete, *any* function $f : G \to G$ is continuous.

Manifestly, there is a strong similarity between algorithm 5.2 and Shor's factoring algorithm: the quantum Fourier transforms $\mathcal{F}_{\mathbb{T}}$ in our algorithm $\mathcal{F}_{\mathbb{Z}}$ plays the role of the discrete Fourier transform $\mathcal{F}_{2^n}$, and c-V_a acts as the modular exponentiation gate [4]. In fact, one can regard algorithm 5.2 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 5.2 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 $|p\rangle = \sum_{m\in\mathbb{Z}} \mathrm{e}^{-2\pi \mathrm{i} p m}|m\rangle$ (which are unphysical and cannot be prepared with finite computational resources) and arbitrarily precise measurements. We finish the proof of theorem 5.2 by giving an improved algorithm that does not rely on unphysical requirements.

Algorithm 5.3 (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 $\left|\widetilde{0}\right\rangle = \frac{1}{\sqrt{2M+1}} \sum_{-M}^{+M} |m\rangle$, i.e. a square pulse of length $2M+1$ in the integer basis, centered at 0. This step simulates steps 1-2 in algorithm 5.2.

3-4 Repeat steps 3-4 of algorithm 5.2. 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

$$|\psi\rangle = \frac{1}{\sqrt{L}} \sum_{-L_a}^{L_b} |s + jr\rangle, \tag{5.10}$$

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 (5.10). We compute $\hat{\psi}$ using formula (1.18):

$$\begin{aligned}\hat{\psi}(p) &= \sum_{x \in \mathbb{Z}} \mathrm{e}^{2\pi i p x} \psi(x) = \frac{1}{\sqrt{L}} \sum_{-L_a}^{L_b} \mathrm{e}^{2\pi i p(s+jr)} = \frac{1}{\sqrt{L}} \left(\mathrm{e}^{2\pi i p s}\right) \frac{\mathrm{e}^{2\pi i p r(L_b+1)} - \mathrm{e}^{-2\pi i p r L_a}}{\mathrm{e}^{2\pi i p r} - 1} \\ &= \frac{\mathrm{e}^{2\pi i p \left(s + \frac{L_b - L_a}{2}\right)}}{\sqrt{L}} \frac{\sin(\pi L p r)}{\sin(\pi p r)} = \frac{\mathrm{e}^{2\pi i p \left(s + \frac{L_b - L_a}{2}\right)}}{\sqrt{L}} D_{L,r}(p)\end{aligned} \tag{5.11}$$

(to derive the equation, we apply the summation formula of the geometric series and re-express the result in terms of the Dirichlet kernel [185]

$$D_{L,r}(p) = \frac{\sin(\pi L p r)}{\sin(\pi p r)}. \tag{5.12}$$

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 (5.9). 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 [185]. 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.

$$\Pr(|p - \tfrac{k}{r}| \leq \tfrac{\Delta}{2}) = \frac{1}{L} \int_{-\frac{\Delta}{2}}^{+\frac{\Delta}{2}} \frac{\sin^2(\pi L p r)}{\sin^2(\pi p r)} \, \mathrm{d}p \geq \frac{\Delta}{L} \frac{\sin^2\left(\frac{\pi}{2}\right)}{\sin^2\left(\frac{\pi}{2L}\right)} \geq \frac{4}{\pi^2 r}, \tag{5.13}$$

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 (5.13) 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 (5.13), it follows that $\Theta(\frac{1}{M})$ 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 5.2.2).

7 **Classical postprocessing:** We now set M (the length of the initial comb state) to be large enough so that $\frac{\Delta}{2} = \frac{1}{2Lr} \leq \frac{1}{2r^2}$; since $r \leq |\mathbf{B}|$, taking $\log M \in O(\text{poly}\, n)$, where n denotes the encoding length of $\mathbf{B}$, 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 5.2 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 [4]: 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 5.2.

Theorem 5.3 (Shor's algorithm as a normalizer circuit). Shor's order-finding algorithm [4] provides an efficient discretized implementation of our hybrid normalizer algorithm 5.3.

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 [4] 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 5.3 if we discretize the computation. Recall that Shor's algorithm implements a quantum phase estimation [49] 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[16] 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

$$\sum_{m=0}^{D-1} |m\rangle|1\rangle = \sum_{-M}^{M} |m\rangle|1\rangle. \tag{5.14}$$

   So far, we have simulated step 1 in algorithm 5.3 by constructing the same periodic state. These first two steps are also clearly analogous to steps 1-2 in algorithm 5.2.

3-4 Apply the modular exponentiation gate U_{me}, which is the following unitary [4]

$$U_{\text{me}}|m, x\rangle = |m, a^m x\rangle, \tag{5.15}$$

   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 (5.10), with $L \leq D$.

[16] This choice is not essential, neither in Shor's algorithm nor in algorithm 5.3, but it simplifies the proof.

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 (5.11) 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 $|\psi_D\rangle$.

$$\hat{\psi}(p) = \sum_{x=-M}^{x=+M} \mathrm{e}^{2\pi \mathrm{i} p x} \psi(x) \qquad \longleftrightarrow \qquad \hat{\psi}_D(k) = \sum_{x=-M}^{x=+M} \mathrm{e}^{2\pi \mathrm{i} \frac{kx}{D}} \psi_D(x) \tag{5.16}$$

The correspondence (equation 5.16) 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

$$\sum_{k=0}^{D-1} \hat{\psi}\left(\tfrac{k}{D}\right) |k\rangle. \tag{5.17}$$

which is, indeed, a discretized version of (5.11).

7-8 The last steps of Shor's algorithm are identical to 7-8 in algorithm 5.3, with the only difference being that the wavefunction (5.17) is now a discretization of (5.11). 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 (5.12). Indeed, one can show (see, e.g. [7]) with an argument similar to (5.13) 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. [4] 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 the view of the authors, this is a **key indispensable ingredient** of our proof.

We begin our discussion pointing out obstacles for finding quantum factoring algorithm based on *modular exponentiation* gates (controlled V_a rotations), showing that implementing the latter by normalizer circuits over *finite* groups $\mathbb{Z}_M \times \mathbf{B}$ is not possible without solving a computational problem at least as hard as factoring.

Theorem 5.4. Let $\mathcal{H}_{\mathbb{Z}_M}$ = 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 $\mathcal{H} = \mathcal{H}_{\mathbb{Z}_M} \times \mathcal{H}_{\mathbf{B}}$ 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 $\|\mathcal{C} - U_{\mathrm{me}}\|_{\mathrm{op}} \leq 1 - 2^{-1/2}$.

We prove the theorem in appendix 3.1. We highlight that a similar result was proven in [134, 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 5.4 is also true: if $|a|$ divides M, then an argument similar to (3.1) shows that $(m, x) \to (m, a^m x)$

is a group automorphism of $\mathbb{Z}_M \times \mathbf{B}$, and the gate U_{me} automatically becomes a normalizer automorphism gate.

The main implication of theorem 5.4 is that, although finite-group normalizer circuits over $\mathbb{Z}_N \times \mathbf{B}$ can easily implement the quantum Fourier transforms needed for Shor's factoring algorithm, they *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 $\lambda\varphi(N)$ [92].

The above no-go result highlights a deep reason why normalizer gates over $\mathbb{Z} \times \mathbf{B}$ (where we may view $\mathbb{Z}$ as the limit of $\mathbb{Z}_M$ when $M \to \infty$) are needed in theorem 5.3 for implementing a modular exponentiation gate. We further conjecture that the obstacles displayed above are a general feature of finite-group normalizer gates, and that no finite-dimensional black-box normalizer circuit can implement an efficient factoring algorithm.

Conjecture 5.1. 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 [4] (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 5.4. On the other hand, the reason why this does works for the group $\mathbb{Z}$ seems to be, in the view of the authors, intimately related to the fact that the order-finding problem can be naturally cast as an instance of the abelian ***hidden subgroup problem*** over $\mathbb{Z}$ (see also section 5.3.3). 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 [92].

5.3.2.1 Elliptic curves

We finish our discussion of Shor's algorithms for discrete-log and factoring by highlighting that the techniques in sections 5.3.1-5.3.2 can be combined to show that existing generalized quantum algorithms for computing discrete-logarithms [94–96] over *elliptic curves*[17] can also be implemented with black-box normalizer circuits (over the infinite group $\mathbb{Z}^2 \times E$): here, the black-box group is the group of points E of an elliptic curve; despite the latter groups being relatively more abstract that those in sections above, they are finite, abelian and efficient (unique) encodings and fast multiplication algorithms for them are known (hence, they can be modeled as black-box groups).

This last result, which we give in **appendix 3.2** implies that normalizer circuits can also render *elliptic curve cryptography (ECC)* vulnerable, as discussed in the chapter introduction.

5.3.3 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

[17] This last result extends easily even to arbitrary black-box groups.

abelian groups [55–58]. Most quantum algorithms discovered in the early days of quantum computation solve problems that can be recast as abelian HSPs, including Deutsch's problem [50], Simon's [51], order finding and discrete logarithms [4], finding hidden linear functions [52], testing shift-equivalence of polynomials [53], and Kitaev's abelian stabilizer problem [49, 54].

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[18] 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 [7]. 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 [55, 56, 97]. The algorithm is efficient given that the group G is explicitly given[19] in the form $G = \mathbb{Z}_{d_1} \times \ldots \times \mathbb{Z}_{d_m}$ [97, 98, 58].

Algorithm 5.4 (Abelian HSP).

Input. An explicitly decomposed finite abelian group $G = \mathbb{Z}_{d_1} \times \ldots \times \mathbb{Z}_{d_m}$, 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
$$\frac{1}{\sqrt{|G|}} \sum_{g\in G} |g, f(g)\rangle \tag{5.18}$$
3. Apply the QFT over G to the first register.
4. Measure the first register in the standard basis.

[18] 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 5.3.2, where we studied the order finding problem (which is a HSP over $\mathbb{Z}$).

[19] 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).

5. 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 [118] for details on this classical part).

We now claim that the quantum part of algorithm 5.4 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 5.5 (**The abelian HSP algorithm is a normalizer circuit.**). In any abelian HSP, 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 is a group homomorphism, and the HSP-solving quantum circuit is a normalizer circuit over $G \times \mathcal{O}$.

The proof is the content of the next two sections.

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:

$$x \cdot y = \tilde{f}\left(f^{-1}(x) + f^{-1}(y)\right). \tag{5.19}$$

In (5.19) 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 [176]:

$$f^{-1}(x) + f^{-1}(y) = (g_x + H) + (g_y + H) = (g_x + g_y) + H. \tag{5.20}$$

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:

$$x \cdot e = e \cdot x = \tilde{f}\left(f^{-1}(x) + f^{-1}(e)\right) = \tilde{f}\left(f^{-1}(x) + H\right) = x \tag{5.21}$$

The group $\mathcal{O}$ is manifestly finite and abelian—the latter property is due to the fact that the addition (5.20) 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

$$f(g + h) = \tilde{f}\,(g + h + H) = \tilde{f}\,((g + H) + (h + H)) = \tilde{f}\left(f^{-1}\,(x) + f^{-1}\,(y)\right) \tag{5.22}$$

$$= x \cdot y = f(g) \cdot f(h). \tag{5.23}$$

It follows from the first isomorphism theorem in group theory [176] 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 5.4 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 $\sum_{g \in G} |g, e\rangle$.

Next, we look now at step 2 of the computation:

$$\frac{1}{\sqrt{|G|}} \sum_{g \in G} |g, e\rangle \quad \longrightarrow \quad \frac{1}{\sqrt{|G|}} \sum_{g \in G} |g, f(g)\rangle. \tag{5.24}$$

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 (5.22), 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:

$$U_\alpha \sum_g |g, e\rangle = \sum_g |\alpha(g, e)\rangle = \sum_g |g, f(g) \cdot e\rangle = \sum_g |g, f(g)\rangle. \tag{5.25}$$

Finally, note that in the last two steps of the algorithm we measure the register $\mathcal{H}_G$ in the standard basis and post-process the information classically like in a normalizer computation. Hence, we have shown that every step in the quantum algorithm 5.4 can be implemented by a normalizer gate over $G \times \mathcal{O}$. This finishes the proof of theorem 5.5.

The oracular group $\mathcal{O}$ is not a black-box group (but almost)

We ought to stress, at this point, that although theorem 5.5 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 5.2.1), since it is not clear how to compute the group operation (5.19), 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 5.4 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 5.4-5.6 and appendix 3.4.

A connection to a result by Mosca and Ekert

Prior to our work, it was observed by Mosca and Ekert [57, 97] 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 $\tilde{f} \circ \beta$.

5.3.4 Decomposing finite abelian groups

As mentioned earlier, there is a quantum algorithm for decomposing abelian groups, due to Cheung and Mosca [97, 98]. 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 our ***extended Cheung-Mosca's algorithm***.

The group decomposition problem

In this work, 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:

$$\Big(\beta_1,\ldots,\beta_\ell\Big) = \Big(\alpha_1,\ldots\alpha_k\Big)\,A, \qquad \Big(\alpha_1,\ldots\alpha_k\Big) = \Big(\beta_1,\ldots,\beta_\ell\Big)\,B. \tag{5.26}$$

This last equation should be read in multiplicative notation (as in e.g. [246]), 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 [97, 98]. 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 $G = \mathbb{Z}_{c_1}\times\ldots\times\mathbb{Z}_{c_k}$. 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 (5.26) implies

$$\begin{aligned}\alpha_1^{x_1}\cdots\alpha_k^{x_k} &= \Big(\alpha_1,\ldots\alpha_k\Big)\,x = \Big(\beta_1,\ldots,\beta_\ell\Big)\,(Bx), \qquad \text{for any } x\in\mathbb{Z}^k,\\ \beta_1^{y_1}\cdots\beta_1^{y_\ell} &= \Big(\beta_1,\ldots,\beta_\ell\Big)\,y = \Big(\alpha_1,\ldots\alpha_k\Big)\,(Ay), \qquad \text{for any } y\in\mathbb{Z}^\ell.\end{aligned}$$

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 5.6, 5.7:

Lemma 5.2 (**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:

$$\beta_i = (\beta_1,\ldots,\beta_{\ell+1})e_i = \Big(\beta'_1,\ldots,\beta'_m\Big)\,(B'e_i) = \Big(\beta'_1,\ldots,\beta'_m\Big)\,b'_i = \beta_1'^{b'_i(1)}\cdots\beta_m'^{b'_i(m)}. \tag{5.27}$$

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 $G = \mathbb{Z}_{c'_1}\times\ldots\times\mathbb{Z}_{c'_m}$. 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

$$\beta_{\ell+1} = (\beta_1,\ldots,\beta_\ell)x = \Big(\beta'_1,\ldots,\beta'_m\Big)\,(b'_1,\ldots,b'_\ell)x = \Big(\beta'_1,\ldots,\beta'_m\Big)\,b'_{\ell+1}, \tag{5.28}$$

which implies $(b'_1, \ldots, b'_\ell)x \equiv b'_{\ell+1} \bmod (c'_1, \ldots, c'_m)$. By finding such an x, we can solve the multivariate discrete problem, since $\beta_1^{x_1} \cdots \beta_\ell^{x_\ell} = \beta_1^{\prime b'_{\ell+1}(1)} \cdots \beta_m^{\prime b'_{\ell+1}(m)} = \beta_{\ell+1} = \beta$, due to (5.27). Finally, note that we can find x efficiently with existing our deterministic classical algorithms for Group Membership in finite abelian groups (lemma 3.1). □

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 5.3.1). 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 $\mathbb{Z}_{c_1} \times \ldots \times \mathbb{Z}_{c_\ell}$ 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. The first 3 steps of our algorithm mimic Cheung and Mosca's[20]. Our novel contribution here is step 4, which computes the B matrix in (5.26).

Algorithm 5.5 (Extended Cheung-Mosca's algorithm).

Input. A list of generators $\alpha = (\alpha_1, \ldots, \alpha_k)$ of an abelian black-box group $\mathbf{B}$.

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 $d = \operatorname{lcm}(d_1, \ldots, d_k)$.

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 5.3.3).

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 generators [98, 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 $e_i + \ker f \subset \mathbb{Z}_d^k$.

[20] Cheung-Mosca's original presentation first applied Shor's algorithm to decompose B into Sylow p-subgroups and subsequently performed the group decomposition on these subgroups [98]. It is discussed in [98] that the first of these steps is not necessary. In our presentation we bypass this step.

(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 5.2). However, we will use a more subtle efficient *classical* approach to obtain X, by reducing the problem to a **system of linear equations over abelian groups** as in chapter 2. 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 $\mathbb{Z}^{m+\ell}$:

$$\begin{pmatrix} A & -H \end{pmatrix} \begin{pmatrix} y_i \\ z_i \end{pmatrix} = e_i \mod d, \qquad (y_i, z_i) \in \mathbb{Z}_d^m \times \mathbb{Z}^\ell, \tag{5.29}$$

which can be solved in classical polynomial time using the algorithms from chapter 2. Then, the matrix X can 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 [200, 199]:

$$\alpha = \alpha X = \alpha(AA^\#)X = (\alpha A)(A^\# X) = (\beta_1, \ldots, \beta_\ell)(A^\# X). \tag{5.30}$$

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 [200]. Since integral pseudo-inverses can be computed efficiently using the Smith normal form (see appendix 1.4), we finally set $B := A^\# X$.

5.4 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, normalizer circuits associated with abelian groups that are *explicitly decomposed*, can be efficiently simulated classically, by exploiting the generalized stabilizer formalism of chapters chapters 3-4.

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 chapters 3-4 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 5.6 (**Simulation of black-box normalizer circuits**). Black-box normalizer circuits can be efficiently simulated classically using the stabilizer formalism over abelian groups

of chapters 3-4 if a subroutine for solving the group-decomposition problem is provided as an oracle.

The proof of this theorem is the subject of section 3.3 in the appendix.

Since normalizer circuits can solve the group decomposition problem (section 5.3.4), we obtain that this problem is complete for the associated normalizer-circuit complexity class, which we now define.

Definition 5.2 (**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 (section 5.2.2), with polynomial-sized classical computation interspersed between. In other words, if N is an oracle that given an efficient (poly-size) black-box normalizer circuit as input, samples from its output distribution, then

$$\textbf{Black-Box Normalizer} = BPP^N. \tag{5.31}$$

Corollary 5.1 (**Group decomposition is complete**)**.** Group decomposition is a complete problem for the complexity class **Black-Box Normalizer** under classical polynomial-time Turing reductions.

We stress that theorem 5.6 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 5.6 can be restated as a *no-go theorem* for finding new quantum algorithms based on black-box normalizer circuits.

Theorem 5.7 (**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 work, 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 our 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* speedups 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 5.6 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 5.3.3 and refer to appendix 3.4 for a proof). We discuss some implications of this fact in the next sections.

5.5 Universality of short quantum circuits

Since all problems in **Black-Box Normalizer** are solvable by our extended Cheung-Mosca quantum algorithm (supplemented with classical processing), the structure of said quantum algorithm allows us to state the following:

Theorem 5.8 (**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 5.1) and from the structure of the extended Cheung-Mosca quantum algorithm with this problem, which has precisely this structure. □

Similarly to theorem 5.6, theorem 5.8 can be extended to the general abelian HSP setting. For details, we refer the reader to appendix 3.4.

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 5.8 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[21]. In this sense, the abelian HSP quantum algorithm uses an *asymptotically optimal* (constant) number of quantum Fourier transforms. Furthermore, the normalizer-gate depth of this algorithm is optimal in general.

5.6 Other Complete problems

We end this chapter by giving two other complete problems for the complexity class **Black Box Normalizer**.

Theorem 5.9 (**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 $G = \mathbb{Z}^a \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_b}$ 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 $\mathbb{Z}_d^a \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_b}$ □

The latter result can be extended to show that any abelian hidden subgroup problem is polynomial-time equivalent to decomposing groups of form $\mathcal{O}$ (cf. appendix 3.4).

[21] 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 [247].

Theorem 5.10 (**System of linear equations over groups**)**.** Let α be a group homomorphism from a group $G = \mathbb{Z}^a \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_b}$ onto a black-box group $\mathbf{B}$. An instance of a linear system of equations over G and $\mathbf{B}$ is given by a homomorphism 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 $\mathbf{B} = \bigoplus\langle\beta_i\rangle \cong H = \mathbb{Z}_{c_1} \times \ldots \times \mathbb{Z}_{c_\ell}$ with black-box normalizer circuits. Second, we recycle the "de-black-boxing" idea from the proof of theorem 5.6 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 5.3.4). 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 chapter 2. □

Chapter 6

Abelian hypergroups and quantum computation

Motivated by a connection, described here for the first time, between the hidden normal subgroup problem (HNSP) and abelian hypergroups (algebraic objects that model collisions of physical particles), we develop a stabilizer formalism using abelian hypergroups and an associated classical simulation theorem (a la Gottesman-Knill). Using these tools, we develop the first provably efficient quantum algorithm for finding hidden subhypergroups of nilpotent abelian hypergroups and, via the aforementioned connection, a new, hypergroup-based algorithm for the HNSP on nilpotent groups. We also give efficient methods for manipulating non-unitary, non-monomial stabilizers and an adaptive Fourier sampling technique of general interest.

This chapter is based on [124] (joint work with Kevin C. Zatloukal).

6.1 Introduction

Ever since Shor's groundbreaking discovery of an efficient quantum algorithm for factoring [4], researchers have striven to understand the source of its quantum speed up and find new applications for quantum computers. An era of breakthroughs followed, in which researchers found that factoring and discrete log are instances of the so-called *Hidden Subgroup Problem* (HSP), a more general problem about *finite groups*[1]; developed efficient quantum algorithms for the abelian group HSP [49, 54–57, 97, 98, 58]; and discovered that solving the nonabelian group HSP over symmetric and dihedral groups would lead to a revolutionary algorithm for Graph Isomorphism [120] and break lattice-based cryptography [121].

Motivated by these breakthroughs, there has been a great deal of research work over the last decade aimed at finding efficient quantum algorithms for nonabelian HSPs, leading to many successes [99–119, 9], though efficient quantum algorithms for dihedral and symmetric HSP have still not been found.

Thus far, the foundation of nearly all known quantum algorithms for nonabelian HSPs has been the seminal work of Hallgren, Russell, and Ta-Shma [99], which showed that hidden *normal* subgroups can be found efficiently for *any* nonabelian group. For example, the algorithms for (near) Hamiltonian groups [122] work because all subgroups of such groups are (nearly) normal. Likewise, the sophisticated algorithm of Ivanyos et al. for 2-nilpotent groups [123] cleverly reduces the problem of finding a hidden non-normal subgroup to two problems of finding hidden normal subgroups.

[1] In the HSP, the task is to find a subgroup H of a finite group G by evaluating a function $f : G \to X$, which is given to us and is promised to *hide* H in the sense that $f(x) = f(y)$ iff $x = yh$ for some $h \in H$.

Surprisingly, given the importance of the nonabelian HSP program in the history of quantum computing, the success of the quantum algorithm for the hidden *normal* subgroup problem (HNSP) [99] remains poorly explained. The initial motivation for this work was to improve our understanding of the quantum algorithm for the HNSP up to the same level as those for abelian HSPs.

Our approach is inspired by the connection between Shor's algorithm, Gottesman's *Pauli stabilizer formalism* (PSF) [1], and the Gottesman-Knill theorem [1–3] of chapter 5. In short, there we showed that all most-famous quantum algorithms for abelian Hidden Subgroup Problems are generalized types of Clifford operations over groups; this connection, combined with the generalized Group Stabilizer Formalism (GSF) for simulating normalizer circuits (chapters 3-4), let us derive a sharp *no-go theorem* for finding new quantum algorithms with the standard abelian group Fourier sampling techniques.

Given the success of our previous techniques at understanding abelian HSP quantum algorithms, our aim in this chapter is to gain a deeper understanding of the algorithm for HNSPs on nonabelian groups using a more sophisticated stabilizer formalism. Furthermore, because the PSF (and generalizations) have seminal applications in fault tolerance [3, 43], measurement based quantum computation [42], and condensed matter theory [75], we expect a new stabilizer formalism to find new uses outside of quantum algorithm analysis.

6.1.1 Main results

While it would be natural to generalize the abelian group stabilizer formalisms into a nonabelian group stabilizer formalism, we find that the proper way to understand the quantum algorithm for the HNSP is not to generalize the "abelian" property but rather the "group" property. In particular, we will work with *abelian hypergroups*. These are generalizations of groups and can be thought of as collections of particles (and anti-particles) with a "collision" operation that creates new particles. A group is a special case of a hypergroup where each collision produces exactly one resulting particle.

Our first result is a formal connection between the HNSP and abelian hypergroups which will be helpful to understand why quantum computers can solve this problem:

I. **Connecting the HNSP to the abelian HSHP.** We demonstrate (**section 6.3**) that, in many natural cases, the HNSP can be reduced to a problem on abelian hypergroups, called the hidden *subhypergroup* problem (HSHP) [125, 126]. This occurs because all of the information about the normal subgroups of a nonabelian group is captured in its hypergroup of conjugacy classes. Even in a nonabelian group, there is a multiplication operation on conjugacy classes that remains abelian. Our results show that, in many natural cases, finding hidden normal subgroups remains a problem about an abelian algebraic structure even when the group is nonabelian.

Our next results show that the tools that proved successful for understanding quantum algorithms for abelian group HSPs (as well as many other problems) can be generalized to the setting of abelian hypergroups:

II. **A hypergroup stabilizer formalism.** We extend the PSF [1–3] and our earlier abelian-group extension (chapters 3-4) into a stabilizer formalism that uses commuting *hypergroups* (instead of *groups*) of generalized Pauli operators. The latter are no longer unitary nor monomial but still exhibit rich Pauli-like features that let us manipulate them with (new) hypergroup techniques and are normalized by associated Clifford-like gates. We also provide a normal form for hypergroup stabilizer states (**theorem 6.4**) that are CSS-like [248–250] in our setting.

III. **A hypergroup Gottesman-Knill theorem.** We introduce models of *normalizer circuits over abelian hypergroups*, which contain hypergroup quantum Fourier transforms (QFTs) and other entangling gates. These models provide a major generalization of the (finite[2]) abelian group normalizer circuit models of chapters 3-5. We show (**theorem 6.3**) that the dynamical evolution of such circuits can be tracked in our hypergroup stabilizer picture and, furthermore, that for large hypergroup families (including products $\mathcal{T}^m$ of constant size hypergroups), many hypergroup normalizer circuits can be efficiently simulated classically (**theorem 6.5**).[3]

We complete our analysis of the HNSP, which we reduced to the abelian HSHP (result I.), showing that our normalizer circuit model encompasses an earlier HSHP quantum algorithm based on a variant of Shor-Kitaev's quantum phase estimation, which was proposed by but not fully analyzed by Amini-Kalantar-Roozbehani in [125, 126]. Using our stabilizer formalism, we prove the latter to be inefficient on easy instances, and, thereby, point out the abelian HSHP as the *first* known commutative hidden substructure problem in quantum computing that *cannot* be solved by standard phase estimation[4]. In spite of this no-go result, we also show, in our last main contribution, that in the interesting cases from the nonabelian HSP perspective, the abelian HSHP can actually be solved with a novel *adaptive/recursive* quantum Fourier sampling approach:

IV. **New quantum algorithms.** We present the first provably efficient quantum algorithm for finding hidden subhypergroups of *nilpotent*[5] abelian hypergroups, provided we have efficient circuits for the required QFTs. This algorithm also leads, via the connection above (result I.), to a new efficient quantum algorithm for the HNSP over nilpotent groups that directly exploits the abelian hypergroup structure and is fundamentally different from the algorithm of Hallgren et al. [99].

Our correctness proofs for these last quantum algorithms can further be extended to crucial non-nilpotent groups[6] (and their associated class hypergroups) such as the dihedral and symmetric groups.[7] In contrast, no efficient quantum algorithm for the nilpotent, dihedral and symmetric HSPs is known. This provides strong evidence that abelian HSHP is a much *easier* problem for quantum computers than nonabelian HSP, and, because of its Shor-like connection with a stabilizer formalism, perhaps even a more *natural* one.

6.1.2 Applications

Though lesser known than nonabelian groups, abelian hypergroups have a wide range of applications in convex optimization (cf. association schemes [127, 128]), classical cryptography, coding theory [129] and conformal field theory [130]. In topological quantum computation [75], fusion-rule hypergroups [131] are indispensable in the study of nonabelian anyons [131]. Our stabilizer formalism over the latter hypergroups likely has applications for quantum error correction and for the simulation of protected gates over topological quantum field theories [132].

[2] For simplicity, we do not consider infinite groups nor infinite dimensional Hilbert spaces in this chapter.

[3] Here, we rely on computability assumptions (section 6.6.1) that are always fulfilled in chapter 3.

[4] Note that, in our context and in any HSP setting, phase estimation is used to extract information from a *fixed* unitary oracle U. This should not be confused with the settings where U is not fixed and phase estimation becomes a BQP-complete problem [251].

[5] These are conjugacy class hypergroups associated to *nilpotent groups* [176]. The latter form a *large* group class that includes abelian groups, Pauli/Heisenberg groups over $\mathbb{Z}_{p^r}$ with prime p, dihedral groups D_{2N} with $N = 2^n$, groups of prime-power order and their direct products.

[6] We give another algorithm that works for all groups under some additional mild assumptions.

[7] For dihedral groups/hypergroups we give a quantum algorithm; for symmetric ones, a *classical* one already does the job because symmetric groups/hypergroups have few normal subgroups/subhypergroups.

The stabilizer formalism and classical simulation techniques presented in this chapter are unique in that they are the first and only available methods to manipulate stabilizer operators that neither *unitary*, nor *monomial*, nor *sparse* that we are aware of [203]. Furthermore, our stabilizer formalism yields the first known families of qubit/qudit stabilizer operators for any arbitrary finite dimension d that are not the standard Weyl-Heisenberg operators [3], with associated normalizer gates that are *not* the standard qudit Clifford gates. Additionally, our methods allow great flexibility to construct new codes because the stabilizer families can be chosen over any hypergroup of interest.

6.1.3 Relationship to prior work

Because finite groups are particular examples of finite abelian groups (see section 6.2), the hypergroup normalizer circuits extend (finite) abelian group models of [134], chapters 3-6. Though we will not consider infinite hypergroups, many of our results should extend to locally compact abelian hypergroups (cf. [252, 126] and the discussion in chapter 4 on locally compact abelian groups).

Our efficient classical simulation result (**theorem 6.5**) is not a full generalization of the Gottesman-Knill theorem [1, 2], but of its CSS-preserving variant [47] and without intermediate measurements; in terms of gates and compared to [134], chapter 3-4, this means that we can only simulate hypergroup normalizer circuits built of automorphism gates, global QFTs and generalized Pauli gates. In this chapter, we dedicate most effort to cope with the highly nontrivial difficulty that our Pauli operators are *non-monomial* and *non-unitary*, which renders *all* existing stabilizer formalism techniques [1–3, 134, 63, 64, 93, 133, 135, 136, 138, 137, 203, 253] inapplicable (including those developed thus far within the thesis). To tackle this issue, we develop new simulation techniques based on hypergroup methods, up to a fairly mature state, though further improvement remains possible (see **section 6.6** for a discussion and a related conjecture).

The quantum algorithm results of this chapter solve a question left open in chapter 5 (cf. discussion) where we gave our no-go theorem for finding new quantum algorithms based on black-box group normalizer circuits. Therein, we raised the question of whether normalizer circuits over *different algebraic structures* could be found and be used to bypass our no-go theorem. Our quantum algorithms for abelian HSHPs answer their question in the affirmative: the circuits we use to solve that problem are instances of normalizer circuits over nonabelian groups/hypergroups (**section 6.7**).

The hidden subhypergroup problem (HSHP) we discuss was first considered by Amini, Kalantar and Roozbehani in [125, 126], yet (to the best of our knowledge) no provably efficient quantum algorithm for this problem has been given before. We show that an earlier quantum algorithm proposed in [126] for solving the problem using a variant of Shor-Kitaev's quantum phase estimation [49] is inefficient on easy instances (section 6.7). Interestingly, this means that abelian HSHP is the first known *commutative* hidden substructure problem that cannot be solved by standard phase estimation. Instead, our quantum algorithm is based on a novel *adaptive/recursive Fourier sampling* quantum approach.

For any non-abelian group G, the simulation results we present lead to efficient classical algorithms for simulating quantum Fourier transforms over G (specifically, as employed in weak Fourier sampling routines) acting on coset states $|aH\rangle$, $a \in G$, $H \subset G$ such that aH is invariant under conjugation[8]. In this sense, our work connects with [254], where efficient classical algorithms were given for simulating weak and strong quantum Fourier sampling on arbitrary coset states of semi-direct products group $\mathbb{Z}_p \ltimes A$, where p is prime and A is an abelian group given

[8]This happens, e.g., if $aH = N$ for normal N or if the subgroup H contains the derived subgroup $[G, G]$.

in a canonical form $\mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_m}$.

Finally, we mention it is not contradictory that some hypergroup normalizer circuits are efficiently classically simulable while others lead to valuable quantum algorithms. Analogously to earlier chapters (see discussion in chapter 5), this difference arises from the existence of hypergroups with weaker and stronger computability properties, which now play the role of "decomposed abelian groups" and "abelian black-box groups" (see section 6.6.1 and appendix 4.3).

6.1.4 Chapter outline

We give a non-technical introduction to the theory of hypergroups in **section 6.2**. We then re-introduce the hidden normal subgroup problem (HNSP) and prove its connection to the hidden subhypergroup problem (HSHP) in **section 6.3**. We present our models of hypergroup normalizer circuits, our hypergroup stabilizer formalism, and our simulation results in **sections 6.4-6.6** and describe these on some examples. Finally, we use these tools to develop new quantum algorithms for abelian HSHP and HNSP in **section 6.7**.

While our motivation for developing our hypergroup stabilizer formalism was to understand more about the HNSP, we note that the results of sections 6.4–6.6 are more general, as they apply to arbitrary hypergroups. We expect that these tools will have applications outside of the analysis of quantum algorithms such as to the development of new error correcting codes.

6.2 Abelian hypergroups and hypergroup duality

This section is an introduction for quantum computer scientists to the beautiful theory of *finite abelian hypergroups*[9], whose origin dates back to works by Dunkl [258], Jewett [259], Spector [260] in the 70s. Our account is based on [255–257, 261, 262, 130, 263, 252] and borrows most notation and terminology from [130, 261–263]. Throughout the chapter, hypergroups and groups are assumed to be *finite* unless said otherwise.

In brief, abelian hypergroups are algebraic structures that generalize abelian groups, although in a *different* way than nonabelian groups. Despite being less known than the latter, abelian hypergroups have a wide number of applications in multiple fields, including combinatorics, convex optimization [127, 128]; cryptography, classical error correction [129]; classical information theory [130]; and conformal field theory [130]. In topological quantum computation [75], certain hypergroups known by the names of "fusion theories or categories" [130, 131] are invaluable in the study of topological order and nonabelian anyons [131].

On top of their versatility, abelian hypergroups also admit a simple and intuitive *physical* definition, which we give now before going into the full mathematical details of their theory. In simple terms, a *finite abelian hypergroup* $\mathcal{T}$ is a set of particle types $\{x_0, x_1, \ldots, x_n\}$ that can collide. When x_i collides with x_j a particle x_k is created with probability n_{ij}^k. Furthermore, a non reactive *vacuum* particle x_0 will be created with non-zero probability by such process iff x_i is the antiparticle of x_k (which always exists).

6.2.1 Definition

We now turn the intuitive definition of hypergroup above into a precise mathematical one.

[9] The hypergroups we consider are frequently called "finite commutative hypergroups" in mathematics. We call them "abelian" because of the focus of this work on abelian and nonabelian HSPs. In some of our references [255–257], the hypergroups in this work are called "*reversible abelian hypergroups*".

A finite abelian hypergroup $\mathcal{T} = \{x_0, x_1, \ldots, x_n\}$ is a basis of a commutative complex C^* algebra $\mathcal{A}(\mathcal{T}) = \mathbb{C}\mathcal{T}$, called the *hypergroup algebra* of $\mathcal{T}$, with a particular structure. $\mathcal{A}(\mathcal{T})$ is endowed with an associative commutative *hyper*operation

$$x_i x_j = \sum_{k=0}^{n} n_{ij}^k x_k \quad \forall x_i, x_j \in \mathcal{T}, \tag{6.1}$$

which returns a superposition of outcomes in $\mathcal{T}$ (we write "$x_k \in x_i x_j$" when x_k is a possible outcome of $x_i x_j$, with $n_{ij}^k \neq 0$); a multiplicative identity $x_0 = 1$; and an involution $x_i \to \overline{x}_i$. Note that commutativity and the presence of the involution imply that $n_{\overline{a},\overline{b}}^{\overline{c}} = n_{ba}^c = n_{ab}^c$ holds for any $a, b, c \in \mathcal{T}$.

Furthermore, the "structure constants" $n_{ij}^k \geq 0$ are *real* numbers with three properties:

(i) **Anti-element property.** For every x_i and any x_j, the identity $x_0 = 1$ can be an outcome of $x_i x_j$ if and only if $x_j = \overline{x}_i$. We call $\overline{x}_i$ the *anti-element* of x_i.

(ii) **Normalization property.** For all values of $k = 0, \ldots, n$ we have $\sum_{k=0}^{n} n_{i,j}^k = 1$; in other words, n_{ij}^k is a probability distribution (of outcomes) over k.

(iii) **Reversibility.**[10] For every $x, y, z \in \mathcal{T}$, it holds that $z \in xy$ if and only if $y \in \overline{x}z$. Moreover, if the *weight of* x is defined as $w_x := 1/n_{x\overline{x}}^0$, the following identity holds:

$$\frac{n_{xy}^z}{w_z} = \frac{n_{\overline{x}z}^y}{w_y} = \frac{n_{z\overline{y}}^x}{w_x} \tag{6.2}$$

As a simple example, any finite abelian group G is an abelian hypergroup. The elements of G define the basis of the group algebra $\mathbb{C}G$ and the involution is $\overline{x} := x^{-1}$. In the case of a group, though, for any $i, j \in \mathbb{Z}_{n+1}$, there is only a single nonzero $n_{i,j}^k$ since $x_i x_j = x_k$ for some k; though hypergroups have a more complicated multiplication than groups, they preserve the property that the product of x and $\overline{x}$ includes the identity.

Hypergroups in this work. Though nonabelian hypergroups exist[11], this chapter focuses on *abelian* ones because they fulfill certain useful dualities (see below). In sections 6.3 and 6.7, we further focus on specific abelian hypergroups that arise from *finite groups* (section 6.2.3).

6.2.2 Glossary

We now give a glossary of hypergroup theoretic concepts for future reference. In all definitions below $\mathcal{T}$ is fixed to be an arbitrary *abelian* finite hypergroup.

Weight functions. Every subset $X \subset \mathcal{T}$ has a *weight* $\varpi_X := \sum_{x \in X} w_x$, with w_x as in (6.2).

Subhypergroup. A *subhypergroup* $\mathcal{N}$ is a subset of $\mathcal{T}$ that is also a hypergroup with the same identity, involution, structure constants and weights.

Quotient hypergroup. For any subhypergroup $\mathcal{N}$ the *quotient hypergroup* $\mathcal{T}/\mathcal{N}$ is an abelian hypergroup whose elements are the cosets $a\mathcal{N} := \{x \in \mathcal{T} : x \in ab \text{ for some } b \in \mathcal{N}\}$. Its hyperoperation is defined [256, 262] by, first, identifying each $a\mathcal{N}$ with an element of the $\mathcal{A}(\mathcal{T})$

[10] This last property (iii) and (6.2) can both be derived from the previous axioms [256].
[11] In fact, every nonabelian group G is also a kind of nonabelian hypergroup.

algebra[12] via $a\mathcal{N} := \sum_{x\in a\mathcal{N}} w_x x/\varpi_{a\mathcal{N}}$. Then, $\mathcal{T}/\mathcal{N}$ inherits a hyperoperation with structure constants $r^{c\mathcal{N}}_{a\mathcal{N},b\mathcal{N}} = \sum_{d\in c\mathcal{N}} n^d_{ab}$ and weights $w_{a\mathcal{N}} = 1/(\sum_{b\in\mathcal{N}} n^b_{a,\overline{a}}) = \varpi_{a\mathcal{N}}/\varpi_{\mathcal{N}}$.

Morphisms. A map between two hypergroups $f : \mathcal{T} \to \mathcal{T}'$ is a *homomorphism* if $f(ab) = f(a)f(b) = \sum_c n^c_{ab} f(c)$ and $f(\overline{a}) = \overline{f(a)}$. An invertible homomorphism is an *isomorphism*. An isomorphism from $\mathcal{T}$ to $\mathcal{T}$ is an *automorphism*. As with groups, isomorphic hypergroups have identical hypergroup-theoretic properties (weights, subhypergroups, etc.).

Character hypergroup $\mathcal{T}^*$. A complex function $\mathcal{X}_\mu : \mathcal{T} \to \mathbb{C}$ is a *character of* $\mathcal{T}$ if it is not identically zero and satisfies the identity[13]

$$\mathcal{X}_\mu(ab) = \mathcal{X}_\mu(a)\mathcal{X}_\mu(b) = \sum_c n^c_{ab}\mathcal{X}(c) \quad \text{and} \quad \mathcal{X}_\mu(\overline{a}) = \overline{\mathcal{X}_\mu}(a) \quad \text{for all } a, b \in \mathcal{T}. \tag{6.3}$$

For any abelian hypergroup $\mathcal{T}$, its set $\mathcal{T}^*$ of character functions defines an abelian *signed* hypergroup with the point-wise functional product as hyperoperation, the trivial character $\mathcal{X}_1(a) = 1$ as identity and the complex conjugate map $\mathcal{X}_\mu \to \overline{\mathcal{X}_\mu}$ as involution: here, "signed" means that $\mathcal{T}^*$ fulfills (i-ii-iii) but may have some negative structure constants $m^\gamma_{\mu\nu}$, which represent negative probabilities. If all constants $m^\gamma_{\mu\nu}$ are non-negative, $\mathcal{T}^*$ is a hypergroup called the *character hypergroup of* $\mathcal{T}$, and $\mathcal{T}$ is said to be *strong* [252]. Throughout the chapter, we assume all hypergroups to be strong (without notice) so that the associated character hypergroups $\mathcal{T}^*$ define new "dual theories" of particle collisions[14].

Weight-order duality. The hypergroups $\mathcal{T}$ and $\mathcal{T}^*$ have the same cardinalities and weights:

$$\varpi_{\mathcal{T}} = \sum_{a\in\mathcal{T}} w_a = \sum_{\mathcal{X}_\mu\in\mathcal{T}^*} w_{\mathcal{X}_\mu} = \varpi_{\mathcal{T}^*}. \tag{6.4}$$

Abelian hypergroup duality. The hypergroup $\mathcal{T}^{**}$ of characters of $\mathcal{T}^*$ is *isomorphic* to the original hypergroup $\mathcal{T}$. This isomorphism is constructed canonically by sending $a \in \mathcal{T}$ to a character

$$\widetilde{\mathcal{X}_a}(\mathcal{X}_\mu) = \overline{\mathcal{X}_\mu}(a). \tag{6.5}$$

In particular, this shows that the hypergroups $\mathcal{T}$, $\mathcal{T}^*$ have the same number of elements.

Remark 6.1 (**Notation**)**.** Throughout the text, we identify dual characters $\widetilde{\mathcal{X}}_a \in \mathcal{T}^{**}$ with elements $a \in \mathcal{T}$ via the isomorphism (6.5). We write the hyperoperation of $\mathcal{T}^*$ compactly as $\mathcal{X}_\mu\mathcal{X}_\nu = \sum_\gamma m^\gamma_{\mu\nu}\mathcal{X}_\gamma$ and, occasionally, use the expression $\mathcal{X}_{\overline{\mu}}$ as a shorthand for $\overline{\mathcal{X}_\mu}$.

The notions of character and duality lead to a family of related concepts that are extremely valuable in hypergroup theory and in the present work:

Annihilators. The *annihilator* $\mathcal{N}^\perp$ of a subhypergroup $\mathcal{N} \subset \mathcal{T}$ is a subhypergroup of $\mathcal{T}^*$

$$\mathcal{N}^\perp := \{\mathcal{X}_\mu \in \mathcal{T}^* : \mathcal{X}_\mu(a) = 1 \text{ for all } a \in \mathcal{N}\}. \tag{6.6}$$

[12] See [255] for a set theoretic definition

[13] If characters are linearly extended to act on the hypergroup algebra $\mathcal{A}(\mathcal{T})$, condition (6.3) becomes $\mathcal{X}_\mu(ab) = \mathcal{X}_\mu(a)\mathcal{X}_\mu(b)$, $\forall a, b \in \mathcal{A}(\mathcal{T})$; in other words, the characters of $\mathcal{T}$ are also the characters of $\mathcal{A}(\mathcal{T})$.

[14] Many of the hypergroup concepts and properties presented in this section as well as our results in sections 6.4, 6.6 can be effortlessly extended to the setting where $\mathcal{T}$ is an abelian signed hypergroup, in which case $\mathcal{T}^*$ is also an abelian signed hypergroup [256, 257, 252, 264, 265]). Though it seems plausible, we have not investigated whether our results in section 6.5 can be extended to signed hypergroups. In the remaining sections, we focus on class and character hypergroups that arise from finite groups, which are always strong.

Subhypergroup duality. A stronger form of hypergroup duality relates the notions of annihilator, subhypergroup and quotient: the annihilator $\mathcal{N}^\perp$ is *isomorphic* to the characters $(\mathcal{T}/\mathcal{N})^*$ of $\mathcal{T}/\mathcal{N}$; moreover, the character hypergroup $\mathcal{N}^*$ is isomorphic to $\mathcal{T}^*/\mathcal{N}^\perp$.

Character orthogonality. Character functions are orthogonal with the inner product

$$\langle \mathcal{X}_\mu, \mathcal{X}_\nu \rangle = \sum_{a \in \mathcal{T}} \frac{w_{\mathcal{X}_\nu} w_a}{\varpi_{\mathcal{T}}} \overline{\mathcal{X}_\mu}(a) \mathcal{X}_\nu(a) = \delta_{\mu,\nu}. \tag{6.7}$$

Moreover, due to hypergroup and subhypergroup duality, for any subhypergroup $\mathcal{N} \subset \mathcal{T}$, any two cosets $a\mathcal{N}, b\mathcal{N} \in \mathcal{T}/\mathcal{N}$ and any $\mathcal{X}_\mu, \mathcal{X}_\nu \in \mathcal{N}^\perp$, the following generalized orthogonality relationships are always fulfilled

$$\sum_{a \in \mathcal{N}} \frac{w_{\mathcal{X}_\nu \mathcal{N}^\perp} w_a}{\varpi_{\mathcal{N}}} \overline{\mathcal{X}_\mu}(a) \mathcal{X}_\nu(a) = \delta_{\mathcal{X}_\mu \mathcal{N}^\perp, \mathcal{X}_\nu \mathcal{N}^\perp}, \qquad \sum_{\mathcal{X}_\mu \in \mathcal{N}^\perp} \frac{w_{b\mathcal{N}} w_{\mathcal{X}_\mu}}{\varpi_{\mathcal{N}^\perp}} \overline{\mathcal{X}_\mu}(a) \mathcal{X}_\mu(b) = \delta_{a\mathcal{N}, b\mathcal{N}}, \tag{6.8}$$

6.2.3 Examples from group theory

We now introduce two examples of hypergroups that play a central role in our work (namely, in sections 6.3 and 6.7). For an arbitrary finite group, these are the hypergroups of conjugacy classes and of characters, which are dual to each other in the sense of (6.5). The existence of these hypergroups linked to arbitrary groups lets us apply our hypergroup normalizer circuit and stabilizer formalisms (sections 6.4–6.5) to nonabelian groups.

6.2.3.1 The hypergroup of conjugacy classes of G

Let G be any finite group. For any $g \in G$, we let $C_g := \{g^a \mid a \in G\}$, where $g^a := a^{-1}ga$ denotes the conjugacy class of g. We let $\overline{G}$ be the set of distinct conjugacy classes of G.

Let $C = \{g_1, g_2, \dots\}$ and $D = \{h_1, h_2, \dots\}$ be two conjugacy classes. Then, for any product, $g_i h_j$, its conjugate $(g_i h_j)^a = g_i^a h_j^a$ is a product of conjugates, so it can be written as $g_k h_\ell$ for some k and ℓ. Furthermore, if there are M distinct products $g_{i_1} h_{j_1}, \dots, g_{i_M} h_{j_M}$ producing some element x, then the distinct products $g_{i_1}^a h_{j_1}^a, \dots, g_{i_M}^a h_{j_M}^a$ all produce x^a. Thus, for each conjugacy class E arising in the product of elements of C and D, we get a well defined number of "how many times" that class arises, which we denote $M_{C,D}^E$.

We will denote by $\mathcal{A}(\overline{G})$ the complex vector space with the distinct conjugacy classes as a basis, which make into a $\mathbb{C}$-algebra by defining the product $CD := \sum_{E \in \overline{G}} M_{C,D}^E E$.

We take the map $C_g \mapsto C_{g^{-1}}$, extended to all of $\mathcal{A}(\overline{G})$ by linearity, as our involution.

It is easy to see that C_e arises in a product $C_g C_h$ iff C_h contains g^{-1}, which occurs iff $C_h = C_{g^{-1}}$. Thus, we can see that the first of the two required properties holds for the product with structure constants $M_{C,D}^E$.

To get the normalization property to hold, though, we must make a minor change. For each $C_g \in \overline{G}$, define c_g to be the vector $(1/|C_g|)\, C_g$. Then we will take $\{c_g \mid C_g \in \overline{G}\}$ to be a new basis. The structure constants become $m_{C,D}^E := M_{C,D}^E |E|/|C||D|$. Since the total number of products of elements formed multiplying C by D is $|C||D|$, we can see that $\sum_{E \in \overline{G}} M_{C,D}^E |E| = |C||D|$, which means that these new structure constants, $m_{C,D}^E$, are properly normalized. Thus, conjugacy classes define a hypergroup, up to this normalization, which we call the **class hypergroup** $\overline{G}$.

Finally, we note that this hypergroup is abelian, even if the underlying group is not abelian. To see this, we calculate $gh = hh^{-1}gh = hg^h = (hg)^h$ (since $h^h = h$), which shows that gh and hg are in the same conjugacy class. Hence, if we are multiplying conjugacy classes instead of elements, we do not distinguish between gh and hg, and we get an abelian structure.

6.2.3.2 The hypergroup of characters

Let $\widehat{G}$ denote the set of irreducible characters of the finite group of G, $\mathcal{A}(\widehat{G})$ the complex vector space with basis $\widehat{G}$, and χ_μ the character of the irreducible representation μ. As we explain next, $\widehat{G}$ has a natural hypergroup structure.

First, the involution of $\widehat{G}$ will be the linear extension of the map $\chi_\mu \mapsto \overline{\chi}_\mu$, for $\chi_\mu \in \widehat{G}$, the image also being an irreducible character.

Second, for any two characters, χ_μ and χ_σ, the pointwise product $\chi_\mu\chi_\sigma$ is also a character, though it is not necessarily irreducible. However, as is well known, any representation can be written as a linear combination of irreducible characters: $\chi_\mu\chi_\sigma = \sum_{\tau\in\widehat{G}} N^\tau_{\mu,\sigma}\chi_\tau$ for some non-negative integers $N^\tau_{\mu,\sigma}$. Using this as our product, $\mathcal{A}(\widehat{G})$ becomes a C^*-algebra, where the identity element is the trivial irreducible representation, χ_1, given by $\chi_1(g) \equiv 1$.

Third, the coefficient $N^\tau_{\mu,\sigma}$, as is also well known from representation theory, is given by $\langle\chi_\mu\chi_\sigma, \chi_\tau\rangle$, where $\langle\cdot,\cdot\rangle$ is the inner product $\langle\chi_\mu, \chi_\sigma\rangle = |G|^{-1}\sum_{g\in G}\chi_\mu(g)\overline{\chi_\sigma(g)}$. From this, we can see that $N^1_{\mu,\sigma} = \langle\chi_\mu\chi_\sigma, \chi_1\rangle = \langle\chi_\mu, \overline{\chi_\sigma}\,\chi_1\rangle = \langle\chi_\mu, \overline{\chi_\sigma}\rangle$. Since χ_μ and $\overline{\chi_\sigma}$ are both irreducible, this is 1 if $\chi_\sigma = \overline{\chi_\mu}$ and 0 otherwise. Hence, we can see that the structure constants $N^\tau_{\mu,\sigma}$ have the first required property.

Finally, we will normalize the characters, as in section 6.2.3.1, in order to have (ii). For this, we define $\widehat{G}$, the **character hypergroup of** G, to be the hypergroup with elements

$$\mathcal{X}_\mu := \frac{\chi_\mu}{d_\mu} \tag{6.9}$$

where d_μ is the dimension of the irrep μ. The structure constants now become $n^\tau_{\mu,\sigma} := N^\tau_{\mu,\sigma}d_\tau/d_\mu d_\sigma$. Since $\chi_\mu\chi_\sigma$ is actually the character of the representation $\mu\otimes\sigma$, which splits into a direct sum of irreducible representations (as described above), we must have $\sum_{\tau\in\widehat{G}} N^\tau_{\mu,\sigma}d_\tau = d_\mu d_\sigma$ as the latter is the dimension of the tensor product. This implies that (ii) is fulfilled and that $\widehat{G}$ (now suitably normalized) is indeed a hypergroup.

Finally, we note that, in this case, our product is manifestly abelian since $\chi_\mu\chi_\sigma$ denotes the element-wise product of these functions, which takes place in the abelian group $\mathbb{C}$.

6.2.3.3 The relationship between $\overline{G}$ and $\widehat{G}$

Crucially, the characters of the hypergroup $\overline{G}$ turn out to be the normalized characters $\mathcal{X}_\mu = \chi_\mu/d_\mu$ of G and, due to duality (6.13), conjugacy classes are the characters of $\widehat{G}$. Classes and characters have weights $w_{C_g} = |C_g|$ and $w_{\chi_\mu} = d_\mu^2$, respectively, where d_μ is the dimension of the irrep μ. This fantastic connection between groups and hypergroups lets one easily derive many well known results in nonabelian group character theory [176, 266] using the properties of section 6.2.1, including the usual character orthogonality relationships and the famous $|G| = \sum_{\mathcal{X}_\mu\in\widehat{G}} d_\mu^2$ identity: the latter can be derived from (6.4), which leads to $\varpi_{\overline{G}} = \sum_{C_g\in\overline{G}}|C_g| = |\overline{G}| = \sum_{\mathcal{X}_\mu\in\widehat{G}} d_\mu^2$ and also implies, $\varpi_{\overline{G}} = |\overline{G}|$ and $\varpi_{\widehat{G}} = |\widehat{G}|$.

6.3 Understanding the Hidden Normal Subgroup Problem

In this section, we demonstrate a formal connection between the hidden *normal* subgroup problem (HNSP) and a problem on abelian hypergroups, defined below, which we call the CC-HSHP. Specifically, we show that, in many cases, we can efficiently reduce the HNSP to the CC-HSHP, classically. This reduction tells us that, even though the HNSP is defined in terms of nonabelian groups, it can be translated into a problem about an algebraic structure that is abelian, albeit one that is more complex than a group (a hypergroup).

In the remainder of the chapter, we will see the effects of moving from nonabelian groups to abelian hypergroups. While the switch from groups to hypergroups creates some new difficulties, we also gain a great deal by working with an abelian structure. In particular, we will see that the mathematical structure of abelian hypergroups leads to a beautiful stabilizer formalism and to new quantum algorithms. Here, we explain how abelian hypergroups arise specifically when looking for hidden normal subgroups before moving to the more general setting.

In the first subsection, we formally define the two problems mentioned above, the HNSP and the CC-HSHP. Afterwards, we show how to reduce the former to the latter.

6.3.1 The HNSP and the CC-HSHP

In the HNSP, we are given an oracle $f : G \to \{0,1\}^*$, assigning labels to group elements, that is promised to *hide* some normal subgroup $N \lhd G$. The latter means that we have $f(x) = f(x')$ for $x, x' \in G$ if and only if $x' = xn$ for some $n \in N$. An algorithm solves the HNSP if it can use this oracle and other quantum computation in order to determine the subgroup N with high probability.

The algorithm of Hallgren et al. for the HNSP finds the hidden subgroup N using exclusively information provided by characters of the group. They showed that this works only for normal subgroups as it cannot distinguish a non-normal subgroup $H \leq G$ from a conjugate subgroup $H^a \neq H$.

If we are only examining the characters of the group G, then it stands to reason that we can get the same information from the hypergroup of characters $\widehat{G}$ or, equivalently, from the hypergroup of conjugacy classes $\overline{G}$ since these two hypergroups contain the same information.[15] Hence, we may expect that the HNSP on G is related to some problem on the abelian hypergroup $\overline{G}$.

A natural question for abelian hypergroups like these is the hidden subhypergroup problem [125]. For our abelian hypergroup of conjugacy classes, we will refer to this problem as the conjugacy class hidden subhypergroup problem or CC-HSHP. Here, we are given an oracle $f : \overline{G} \to \{0,1\}^*$, assigning labels to conjugacy classes, that hides some subhypergroup, and we are asked to determine that subhypergroup via oracle queries and quantum computation. We will see next how this is related to the HNSP.

6.3.2 Reducing the HNSP to the CC-HSHP

Since a normal subgroup N is (the union of) a set of conjugacy classes that is closed under multiplication and taking inverses, it also defines a subhypergroup of $\overline{G}$, which we denote by $\overline{N}_G$.[16] Hence, any subgroup that can be found as the solution of the HNSP can also be found as the solution of the CC-HSHP. Indeed, as we will see next, in many cases, we can directly reduce the HNSP to the CC-HSHP.

In order to perform this reduction, we need to provide a CC-HSHP oracle. Our proofs will show how to translate an oracle for the HNSP into an oracle for the CC-HSHP. These translations assume that we can perform certain computations with conjugacy classes, described in detail in appendix 4.4.2, which we refer to as "computing efficiently with conjugacy classes". (While formally an assumption, we know of no group for which these calculations cannot be performed efficiently.)

Theorem 6.1 (HNSP $\leq$ CC-HSHP, I)**.** Let G be a group. Suppose that we are given a hiding

[15] After all, each can be recovered from the other as its dual hypergroup.

[16] We distinguish this from N, which is a set of group elements, because $\overline{N}_G$ is a set of conjugacy classes.

function $f : G \to \{0,1\}^*$ that is also a class function[a]. If we can compute efficiently with conjugacy classes, then we can efficiently reduce this HNSP to the CC-HSHP.

[a]This means that f is constant on conjugacy classes. This will occur iff G/N is abelian, where N is the hidden subgroup.

Proof. The assumptions about computing efficiently with conjugacy classes imply that, given a conjugacy class C_g, we can efficiently find an element $x \in C_g$ and apply f to get a label. (Since f is a class function, the label is the same for any $x' \in C_g$.) Let N be the hidden subgroup. Since f hides N and N is normal, we can see that $f(xn) = f((xn)^a) = f(x^a n^a) = f(x^a) = f(x^a n')$ for any $n, n' \in N$. This shows that f is constant on $C_g \overline{N}_G$, which corresponds to a coset of the subhypergroup $\overline{N}_G \leq \overline{G}$. It follows immediately that f has distinct values on distinct cosets of $\overline{N}_G$, so we can see that f is a hiding function for this subhypergroup corresponding (uniquely) to N. □

Theorem 6.2 (HNSP $\leq$ CC-HSHP, II)**.** Let G be a group. Suppose that we are given a hiding function $f : G \to H$ that is also a homomorphism. If we can efficiently compute with conjugacy classes of G and H, then we can efficiently reduce this HNSP to the CC-HSHP.

Proof. Consider any element $x \in G$. For any conjugate x^a, for some $a \in G$, we see that $f(x^a) = f(a^{-1}xa) = f(a^{-1})f(x)f(a)$ since f is a homomorphism. Furthermore, since $a^{-1}a = e$, we see that $f(a^{-1})f(a) = f(e) = e$, which shows that $f(a^{-1}) = f(a)^{-1}$. Putting these together, we have $f(x^a) = f(a)^{-1}f(x)f(a) = f(x)^{f(a)}$. This means that the function $\tilde{f}$ taking x to the conjugacy class label of $f(x)$ is a class function, which we can compute efficiently by assumption.[17] Thus, by the same proof as in previous theorem, we can reduce this to the CC-HSHP. □

This latter theorem applies to many of the important examples of HSPs. This includes the oracles used for factoring, discrete logarithm over cyclic groups and elliptic curves, and abelian group decomposition (cf. chapter 5).

While all of these examples are abelian groups, it is true in general that, for any normal subgroup of any group, there is always some hiding function that is a group homomorphism.[18]

From these proofs, we can see that the essential difference between the HNSP and the CC-HSHP is the slightly differing requirements for their oracles. We have seen that, whenever we can convert an oracle for the former into one for the latter, we can reduce the HNSP to the CC-HSHP.[19] Above, we showed this can be done in the case that the two sets of requirements are actually the same (**theorem 6.1**) and the case where the labels produced by the oracle are not opaque but rather come with enough information to compute with their conjugacy classes (**theorem 6.2**).

Apart from this, it is worth reflecting on which of the types of oracle is the most sensible for the problem of finding hidden normal subgroups. With this in mind, we note that the oracle in the HNSP is not specific to normal subgroups: the same type of oracle can hide non-normal subgroups as well — we are simply promised that, in these cases, the hidden subgroup happens to be normal. In contrast, the oracle in the CC-HSHP can *only* hide normal subgroups because it is required to be constant on conjugacy classes. Hence, even though we came upon the oracle definition from the CC-HSHP by looking at hypergroups, it is arguable that this is actually a *better* definition of hiding function for normal subgroups. Our proofs above demonstrate that,

[17]Also note that, since e is the only element in its conjugacy class, $\tilde{f}$ hides the same subgroup as f.

[18]If $H \trianglelefteq G$ is the hidden subgroup, then one example is the canonical oracle $G \to G/H$ given by $x \mapsto xH$.

[19]This also assumes the relatively minor assumption that we can compute with conjugacy classes.

whenever we are given an oracle of this type, we can reduce finding the hidden normal subgroup to the CC-HSHP.

We will return to the HNSP in section 6.7. There, we will show that the CC-HSHP can be efficiently solved on a quantum computer under reasonable assumptions. This, together with the theorems above, show that the HNSP is easy because the CC-HSHP is easy, which gives an explanation for why the HNSP is easy in terms of the presence of an *abelian* algebraic structure.

Before we can do that, however, we need to first develop some tools for analyzing quantum algorithms using abelian hypergroups. These tools will be of independent interest.

6.4 Normalizer circuits over abelian hypergroups

In section 6.3, we described our motivating example (the hidden normal subgroup problem) for considering how abelian hypergroups can be used to understand quantum computation. There, the abelian hypergroups arose from nonabelian groups. However, there are a vast number of interesting hypergroups with applications in physics and mathematics [130], including many of the ones used in topological quantum computation [75, 75], that do *not* arise from groups. So in the next three sections, we will work with a general abelian hypergroup $\mathcal{T}$, which could come from any of these settings.

Our plan in these next few sections is to extend the abelian-group normalizer circuit model that we applied to successfully understand quantum algorithms in chapters 3-5. We start, in this section, by defining a model of normalizer circuits over hypergroups that we will analyze. Definitions are given in section 6.4.1. In section 6.4.2, we go through a few examples of what these models consist of for different hypergroups. In later sections, we develop a stabilizer formalism and a Gottesman-Knill-type theorem that applies to these circuits.

6.4.1 Circuit model

Fix $\mathcal{T}$ to be an arbitrary *finite abelian hypergroup*. We now define a circuit model, which we call *normalizer circuits* over $\mathcal{T}$. The gates of these circuits are called *normalizer gates*.

The Hilbert space: Normalizer gates over $\mathcal{T}$ act on a Hilbert space $\mathcal{H}_{\mathcal{T}}$ with two orthonormal bases, $\mathcal{B}_{\mathcal{T}} = \{|a\rangle, a \in \mathcal{T}\}$ and $\mathcal{B}_{\mathcal{T}^*} = \{|\mathcal{X}_\mu\rangle, \mathcal{X}_\mu \in \mathcal{T}^*\}$, labeled by elements and characters of $\mathcal{T}$,[20] that are related via the *quantum Fourier transform* (QFT) of $\mathcal{T}$:

$$\mathcal{F}_{\mathcal{T}}|a\rangle = \sum_{\mathcal{X}_\mu \in \mathcal{T}^*} \sqrt{\frac{w_{\mathcal{X}_\mu} w_a}{\varpi_{\mathcal{T}^*}}} \mathcal{X}_\mu(a)|\mathcal{X}_\mu\rangle, \qquad \mathcal{F}_{\mathcal{T}}^\dagger|\mathcal{X}_\mu\rangle = \sum_{a \in \mathcal{T}} \sqrt{\frac{w_a w_{\overline{\mathcal{X}_\mu}}}{\varpi_{\mathcal{T}}}} \overline{\mathcal{X}_\mu}(a)|a\rangle. \tag{6.10}$$

Character orthogonality (6.7) implies that (6.10) is a unitary transformation.

Registers: Because in many settings it is important to split a quantum computation in multiple registers, we let $\mathcal{T}$ and $\mathcal{H}_{\mathcal{T}}$ have a general direct product and tensor product form

$$\mathcal{T} = \mathcal{T}_1 \times \cdots \times \mathcal{T}_m \qquad \longleftrightarrow \qquad \mathcal{H}_{\mathcal{T}} \cong \mathcal{H}_{\mathcal{T}_1} \otimes \cdots \otimes \mathcal{H}_{\mathcal{T}_m}. \tag{6.11}$$

In this case, the QFT over $\mathcal{T}$ is the tensor product of the QFTs over the $\mathcal{T}_i$'s:

$$\mathcal{F}_{\mathcal{T}} = \mathcal{F}_{\mathcal{T}_1} \otimes \cdots \otimes \mathcal{F}_{\mathcal{T}_m}. \tag{6.12}$$

Input states: Each register $\mathcal{H}_{\mathcal{T}_i}$ is initialized to be in either an *element state* $|x_i\rangle, x_i \in \mathcal{T}_i$ or in

[20] Note that duality (6.5) implies $\dim \mathcal{H}_{\mathcal{T}} = \dim \mathcal{H}_{\mathcal{T}^*}$.

a *character state* $|\mathcal{X}_\mu\rangle, \mathcal{X}_\mu \in \mathcal{T}_i^*$.

Gates: The allowed *normalizer gates* at step t of a normalizer circuit depend on a parameter $\mathcal{T}(t)$, which is a hypergroup, related to $\mathcal{T}$, of the form

$$\mathcal{T}(t) = \mathcal{T}(t)_1 \times \cdots \times \mathcal{T}(t)_m \quad \text{with} \quad \mathcal{T}(t)_i \in \{\mathcal{T}_i, \mathcal{T}_i^*\}. \tag{6.13}$$

The role of $\mathcal{T}(t)$ is to indicate whether the operations carried out by circuit at time t will be on the element or character basis. At step 0, $\mathcal{T}(0)$ is chosen so that $\mathcal{T}_i(0) \in \{\mathcal{T}_i, \mathcal{T}_i^*\}$ indicates whether $\mathcal{H}_{\mathcal{T}_i}$ begins on an element or character state. At any steps $t > 0$, $\mathcal{T}(t)$ depends on the gates that have been applied at earlier steps, following rules given below.

Normalizer gates at time t can be of four types:

1. **Pauli gates.** Pauli gates of type X[21] implement the $\mathcal{T}(t)$ hyperoperation $X_{\mathcal{T}(t)}(a)|b\rangle = |ab\rangle$ for invertible elements $a \in \mathcal{T}(t)$. Pauli gates of type Z multiply by phases $Z_{\mathcal{T}(t)}(\mathcal{X}_\mu)|b\rangle = \mathcal{X}_\mu(b)|b\rangle$ which correspond to invertible characters in $\mathcal{T}(t)^*$.

2. **Automorphism Gates.** Let $\alpha : \mathcal{T}(t) \to \mathcal{T}(t)$ be an automorphism of the hypergroup $\mathcal{T}(t)$. Then the automorphism gate U_α taking $|g\rangle \mapsto |\alpha(g)\rangle$ is a valid normalizer gate.

3. **Quadratic Phase Gates** A complex function $\xi : \mathcal{T}(t) \to U(1)$ is called "quadratic" if the map $B : \mathcal{T}(t) \times \mathcal{T}(t) \to U(1)$ defined by $\xi(gh) = \xi(g)\xi(h)B(g,h)$ is a bi-character, i.e., a character of the hypergroup in either argument. A quadratic phase gate is a diagonal map D_ξ taking $|g\rangle \mapsto \xi(g)|g\rangle$ for some quadratic function ξ.

4. **Quantum Fourier Transforms**. A *global QFT* implements the gate $\mathcal{F}_{\mathcal{T}(t)}$ over $\mathcal{T}(t)$ (6.10). *Partial QFTs* implement the gates $\mathcal{F}_{\mathcal{T}(t)_i}$ on single registers $\mathcal{H}_{\mathcal{T}(t)_i}$ (while the other registers remain unchanged).

Update rule: Because QFTs change the hypergroup that labels the standard basis (6.10), the rules above do not specify which normalizer gates should be applied on the second step. For this reason, in our gate model, we *update* the value of $\mathcal{T}(t+1)$ at time $t+1$ so that $\mathcal{T}(t+1)_i = \mathcal{T}(t)_i^*$ if a QFT acts on $\mathcal{H}_{\mathcal{T}(t)_i}$ and $\mathcal{T}(t+1)_i = \mathcal{T}(t)_i$ otherwise.

Measurements: At the final step T, every register $\mathcal{H}_{\mathcal{T}_i}$ is measured in either the element or the character basis depending on the configuration of the QFTs in the circuit: specifically, $\mathcal{H}_{\mathcal{T}_i}$ is measured in basis $\mathcal{B}_{\mathcal{T}_i}$ labeled by elements of $\mathcal{T}_i$ when $\mathcal{T}(T)_i = \mathcal{T}_i$, and in the character basis $\mathcal{B}_{\mathcal{T}_i^*}$ when $\mathcal{T}(T)_i = \mathcal{T}_i^*$. In the end, the final string of measurement outcomes identifies an element of the hypergroup $\mathcal{T}(T)$.

6.4.2 Examples from group theory

We now give examples of normalizer gates over conjugacy class and character hypergroups with the aim to illustrate our definitions and, furthermore, show how our results can be applied to define models of *normalizer circuits over nonabelian groups*.

[21] In previous chapters we did not consider X-type Pauli gates to be normalizer gates explicitly but we proved that they can be implemented via normalizer circuits comprising 3 gates (lemma 4.1). This property is shared with hypergroup normalizer circuits, as shown in (6.26), theorem 6.3 below. For the sake of generality, we add them to the basic set of normalizer gates in this final chapter.

Example 1: Clifford and abelian-group normalizer circuits

For an abelian group G, all conjugacy classes contain a single group element. Consequently, the class hypergroup $\overline{G}$ is always a *group* and it is equal to G. In this scenario, our gate model coincides with the finite abelian-group normalizer-circuit model of earlier chapters, which contain numerous examples of normalizer gates, including the standard Clifford circuits for qubits/qudits [2, 3] and circuits that contain abelian-group QFTs. We refer the reader to chapter 1 for examples of normalizer circuits that are efficiently classically simulable, and to chapter 5 for hard instances that can implement famous quantum algorithms such as Shor's [4].

Example 2: normalizer circuits over nonabelian groups

We now apply our circuit formalism to introduce (new) models of normalizer gates over any finite nonabelian group G. For this, we associate a Hilbert space $\mathcal{H}_G$ to G with basis $\{|g\rangle, g \in G\}$ and restrict the computation to act on its (nontrivial) subspace $\mathcal{I}_{\overline{G}}$ of *conjugation invariant* wavefunctions.[22]

As is well-known from representation theory [266], the Dirac delta measures δ_{C_g} over conjugacy classes $C_g \in \overline{G}$ and the character functions χ_μ of the irreducible representations $\mu \in \mathrm{Irr}(G)$ form two dual orthonormal bases of $\mathcal{I}_{\overline{G}}$. In our circuit model, recalling the definitions of class hypergroup $\overline{G}$ and character hypergroup $\widehat{G}$ (see section 6.2.3), this means that $\mathcal{I}_{\overline{G}}$ can be viewed as the Hilbert space of the conjugacy class hypergroup $\mathcal{H}_{\overline{G}}$ with a *conjugacy-class basis* $\mathcal{B}_{\overline{G}} = \{|C_g\rangle, C_g \in G\}$ and a *character basis* $\mathcal{B}_{\widehat{G}} = \{|\mathcal{X}_\mu\rangle, \mathcal{X}_\mu \in \widehat{G}\}$ if we define these bases within $\mathcal{H}_G$ as the vectors

$$|C_g\rangle = \frac{1}{\sqrt{|C_g|}} \sum_{aga^{-1} \in C_g} \left|aga^{-1}\right\rangle \quad \text{and} \quad |\mathcal{X}_\mu\rangle = \sqrt{\frac{d_\mu^2}{|G|}} \sum_{g \in G} \overline{\mathcal{X}_\mu}(g)|g\rangle. \tag{6.14}$$

With these identifications, we can now define a *normalizer circuit over* G to be a normalizer circuit over the hypergroup $\overline{G}$: the latter acts on the conjugation invariant subspace, admits conjugacy class and character state inputs, and applies QFTs, group automorphisms, and quadratic phase functions associated to $\overline{G}$ and $\widehat{G}$. Furthermore, if we have a direct product $G = G_1 \times \cdots \times G_m$, then $\overline{G} = \overline{G}_1 \times \cdots \times \overline{G}_m$, $\widehat{G} = \widehat{G}_1 \times \cdots \times \widehat{G}_m$ and $\mathcal{H}_{\overline{G}} = \mathcal{H}_{\overline{G}_1} \otimes \cdots \otimes \mathcal{H}_{\overline{G}_m}$. In this setting, normalizer gates such as partial QFTs and entangling gates over different registers are allowed.

It is straightforward to check, using the identities $w_{C_g} = |C_g|$, $w_{\mathcal{X}_\mu} = d_\mu^2$ and $\varpi_{\overline{G}} = \varpi_{\widehat{G}} = |G|$ (section 6.2.3), that the QFT defined with the bases from (6.14) is actually the identity map. Even so, the QFT performs a useful purpose in these circuits as it changes the basis used for subsequent gates, $\mathcal{T}(t+1)$. In particular, the QFT can change the final basis to the character basis, which means that the final measurement is performed in the character basis rather than the element basis.

As we show in appendix 4.4, we can perform a final measurement in the character basis provided that we have an efficient QFT circuit for the group G. The same techniques also allow us to prepare initial states and perform all the gate types (Pauli gates, automorphisms, and quadratic phases) in the character basis efficiently.

Performing the gate types in the conjugacy class basis is straightforward if we make some modest assumptions about our ability to compute with conjugacy classes of the group. For example, we need a way to map an element $g \in G$ to a label of its conjugacy class C_g. These details are discussed in appendix 4.4.2, where we explain why these assumptions are easily satisfied for typical classes of groups.

[22] That is, wave functions $\psi(x)$ such that $\psi(x^g) = \psi(x)$ for all $x, g \in G$.

Example 3: quaternionic circuits

Lastly, we give concrete examples of normalizer gates over nonabelian groups for systems of the form Q_8^n where Q_8 is the quaternion group with presentation

$$Q_8 = \langle -1, i, j, k | (-1)^2 = 1, i^2 = j^2 = k^2 = ijk = -1 \rangle \tag{6.15}$$

Note that Q_8 is nonabelian and that, although it has eight elements $\pm 1, \pm i, \pm j, \pm k$, it has only five conjugacy classes $\{1\}$, $\{-1\}$, $\{\pm i\}$, $\{\pm j\}$, $\{\pm k\}$. Hence, although the Hilbert space $\mathcal{H}_{Q_8}^{\otimes n} = \{|g\rangle : g \in Q_8\}$ is 8^n-dimensional, our quantum computation based on normalizer gates will never leave the 5^n-dimensional conjugation invariant subspace $\mathcal{H}_{\overline{Q_8}}^{\otimes n}$, which can be viewed as a system of 5-dimensional qudits. Using the group character table [177], it is easy to write down the conjugacy class and character basis states of $\mathcal{H}_{\overline{Q_8}}$:

- **Conjugacy-class states:**

$$|C_1\rangle = |1\rangle, \quad |C_{-1}\rangle = |-1\rangle, \quad |C_i\rangle = \tfrac{|i\rangle + |-i\rangle}{\sqrt{2}}, \quad |C_j\rangle = \tfrac{|j\rangle + |-j\rangle}{\sqrt{2}}, \quad |C_k\rangle = \tfrac{|k\rangle + |-k\rangle}{\sqrt{2}}$$

- **Character states:**

$$\begin{aligned}
|\mathcal{X}_1\rangle &= \tfrac{1}{\sqrt{8}} \left(|C_1\rangle + |C_{-1}\rangle + \sqrt{2}|C_i\rangle + \sqrt{2}|C_j\rangle + \sqrt{2}|C_k\rangle \right), \\
|\mathcal{X}_i\rangle &= \tfrac{1}{\sqrt{8}} \left(|C_1\rangle + |C_{-1}\rangle + \sqrt{2}|C_i\rangle - \sqrt{2}|C_j\rangle - \sqrt{2}|C_k\rangle \right), \\
|\mathcal{X}_j\rangle &= \tfrac{1}{\sqrt{8}} \left(|C_1\rangle + |C_{-1}\rangle - \sqrt{2}|C_i\rangle + \sqrt{2}|C_j\rangle - \sqrt{2}|C_k\rangle \right), \\
|\mathcal{X}_k\rangle &= \tfrac{1}{\sqrt{8}} \left(|C_1\rangle + |C_{-1}\rangle - \sqrt{2}|C_i\rangle - \sqrt{2}|C_j\rangle + \sqrt{2}|C_k\rangle \right), \\
|\mathcal{X}_2\rangle &= \tfrac{2}{\sqrt{8}} \left(|C_1\rangle - |C_{-1}\rangle \right),
\end{aligned}$$

We now give a list of nontrivial normalizer gates (not intended to be exhaustive), which we obtain directly from the definitions in section 6.4 applying basic properties of the quaternion group [176, 177]. For the sake of conciseness, the elementary group-theoretic derivations are omitted.

- **Quantum Fourier transform.** For one qudit, the QFT implements the change of basis between the conjugacy-class and character bases written above. For n-qudits, the total QFT implements this change of bases on all qudits. Partial QFTs, instead, implement the QFT on single qudits.

- **Pauli gates:** $X_{\overline{Q_8}}(-1)|C_x\rangle = |-C_x\rangle, \quad Z_{\overline{Q_8}}(\mathcal{X}_\ell)|C_x\rangle = \mathcal{X}_\ell(C_x)|C_x\rangle \quad \text{for } \ell = i, j, k.$

- **Automorphism gates:** All automorphisms of the class-hypergroup can be obtained by composing functions α_{xy} that swap pairs of conjugacy classes C_x, C_y with $x, y \in \{i, j, k\}$. The corresponding *swap gates* $U_{\alpha_{xy}}|C_z\rangle = |\alpha_{xy}(C_z)\rangle$ are instances of one-qudit quaternionic automorphism gates.

- **Quadratic phase gate.** Next, we give examples of non-linear quadratic phase gates. For one qudit, quadratic phase gates D_{ξ_i}, D_{ξ_j}, D_{ξ_k} defined as

$$D_{\xi_x}|C_y\rangle = |C_y\rangle, \text{ if } y \in \langle x \rangle = \{\pm 1, \pm x\} \quad \text{and} \quad D_{\xi_x}|C_z\rangle = i|C_z\rangle \text{ otherwise,}$$

  provide quaternionic analogues of the one-qubit $P = \mathrm{diag}(1, i)$ Clifford gate.

  For two qudits, there is also a "*quaternionic controlled-Z gate*" D_ξ, which implements a quadratic function $\xi(C_x, C_y) = f_{C_x}(C_y)$, with f_{C_x} being a linear character specified by the following rules: $f_{C_{\pm 1}} = \mathcal{X}_1$ and $f_{C_x} = \mathcal{X}_x$ for $x = i, j, k$. We refer the reader to appendix D.2 for a proof that the above functions are quadratic.

Most of the above gates act on a single copy of $\mathcal{H}_{Q_8}$ and, thus, cannot generate entanglement. Entangling normalizer gates can be found by considering two copies of $\mathcal{H}_{Q_8}$. The allowed normalizer gates are now those associated to the group $Q_8 \times Q_8$.

We give next three examples of two-qudit automorphism gates U_{α_i}, U_{α_j}, U_{α_k}, that can generate **quantum entanglement** and provide quaternionic analogues of the qubit CNOT [2] and the qudit CSUM gates [3]. The three are defined as

$$U_{\alpha_x}|C_1, C_2\rangle = |\alpha_x(C_1, C_2)\rangle = |C_1, f_x(C_1)C_2\rangle \tag{6.16}$$

where $f_x(C_y) = C_1$ if C_y is contained in the subgroup $\langle x\rangle = \{\pm 1, \pm x\}$ generated by x and $f_x(C_y) = C_{-1}$ otherwise[23]. The action of any of these gates on the product state $|\mathcal{X}_1\rangle|C_1\rangle$ generates an entangled state; we show this explicitly for U_{α_i}:

$$U_{\alpha_i}|\mathcal{X}_1\rangle|C_1\rangle = \tfrac{1}{2}\left(\tfrac{|C_1\rangle+|C_{-1}\rangle}{\sqrt{2}} + |C_i\rangle\right)|C_1\rangle + \tfrac{1}{2}\left(|C_j\rangle + |C_k\rangle\right)|C_{-1}\rangle.$$

Quaternionic quadratic-phase gates can also generate **highly entangled states**. For instance, the action of D_ξ on a product state $|\mathcal{X}_1\rangle|\mathcal{X}_1\rangle$ creates an entangled bi-partite state with Schmidt rank 4, which is close to the maximal value of 5 achievable for a state in $\mathcal{H}_{\overline{Q_8}} \otimes \mathcal{H}_{\overline{Q_8}}$:

$$D_\xi|\mathcal{X}_1\rangle|\mathcal{X}_1\rangle = \tfrac{1}{4}\left(\left(\tfrac{|C_1\rangle+|C_{-1}\rangle}{\sqrt{2}}\right)|\mathcal{X}_1\rangle + |C_i\rangle|\mathcal{X}_i\rangle + |C_j\rangle|\mathcal{X}_j\rangle + |C_k\rangle|\mathcal{X}_k\rangle\right). \tag{6.17}$$

A quaternionic analogue of the ($d = 4$) qudit *cluster state* [65] displaying multi-partite entanglement can prepared by repeatedly applying D_ξ to all pairs of neighboring qudits on a lattice, chosen to be initially in the state $|\mathcal{X}_1\rangle$.

As this example shows, while normalizer circuits have fairly simple algebraic properties, they can produce states that are very complicated and often highly entangled. Thus, as in the abelian case, it comes as a surprise that these circuits can often be classically simulated efficiently, as we will see in section 6.

6.5 A Hypergroup Stabilizer Formalism

In this section we develop a stabilizer formalism based on abelian hypergroups that extends Gottesman's PSF [1–3] and the abelian group extension of chapters 3-5. We apply our formalism to the description of new types of quantum many-body states, including hypergroup coset states and those that appear at intermediate steps of quantum computations by normalizer circuits over hypergroups.

This section is organized as follows. In section 6.5.1, we introduce new types of Pauli operators based on hypergroups that have richer properties than those of chapters 3-5: most remarkably, they can be non-monomial and non-unitary matrices. In section (section 6.5.2) we show that commuting *stabilizer hypergroups* built of the latter Paulis can be used to describe interesting families of quantum states, which we call *hypergroup stabilizer states*, as well as track the dynamical evolution of hypergroup normalizer circuits (**theorem 6.3**). In section 6.5.3, we give a powerful *normal form* (**theorem 6.4**) for hypergroup stabilizer states that are CSS-like [248, 249]. The latter will be an invaluable tool in this chapter, which we later use to describe hypergroup coset states (equation 6.31, corollary 6.1) and analyze the quantum algorithms of section 6.7. The techniques in this section will also be the basis of the classical simulation methods developed in section 6.6.

[23] The function f_x defines a group homomorphism from $\overline{Q}_8$ into its center $Z(Q_8)$. Using this fact, it is easy to show that α_x is a group automorphism.

The fact that our hypergroup stabilizer formalism is based on non-monomial, non-unitary stabilizers introduces nontrivial technical difficulties that are discussed in detail in sections 6.5.1-6.5.2. The techniques we develop to cope with the issues are unique in the stabilizer formalism literature since both the original PSF and all of its previously known extensions [1–3, 134, 63, 64, 93, 133, 135, 136, 138, 137, 203, 253] were tailored to handle unitary monomial stabilizer matrices. For this reason, we regard them as a main contribution of this chapter.

Like in previous section, we develop our stabilizer formalism over arbitrary abelian hypergroups. Throughout the section, we fix $\mathcal{T} = \mathcal{T}_1 \times \cdots \times \mathcal{T}_m$ to be an arbitrary finite abelian hypergroup with Hilbert space $\mathcal{H}_\mathcal{T} \cong \mathcal{H}_{\mathcal{T}_1} \otimes \cdots \otimes \mathcal{H}_{\mathcal{T}_m}$.

6.5.1 Hypergroup Pauli operators

We introduce generalized Pauli operators over $\mathcal{T}$ acting on $\mathcal{H}_\mathcal{T}$ with analogous properties to the qubit and abelian-group Pauli matrices (chapters 3-4). In our formalism Pauli operators perform operations associated to the abelian hypergroups of section 6.4. First, *Pauli operators over a hypergroup* $\mathcal{T}$ (6.18) implement multiplications by hypergroup characters as well as the hypergroup operation of $\mathcal{T}$. For the character group $\mathcal{T}^*$, *Pauli operators over* $\mathcal{T}^*$ are defined analogously (6.19). More precisely, for all $x, y \in \mathcal{T}$, $\mathcal{X}_\mu, \mathcal{X}_\nu \in \mathcal{T}^*$, we define

$$Z_\mathcal{T}(\mathcal{X}_\mu)|x\rangle := \mathcal{X}_\mu(x)|x\rangle, \qquad X_\mathcal{T}(x)|y\rangle := \sum_{z\in\mathcal{T}} \sqrt{\tfrac{w_y}{w_z}}\, n^z_{x,y}|z\rangle, \tag{6.18}$$

$$Z_{\mathcal{T}^*}(x)|\mathcal{X}_\mu\rangle := \mathcal{X}_\mu(x)|\mathcal{X}_\mu\rangle, \qquad X_{\mathcal{T}^*}(\mathcal{X}_\mu)|\mathcal{X}_\nu\rangle := \sum_{\mathcal{X}_\gamma\in\mathcal{T}^*} \sqrt{\tfrac{w_\nu}{w_\gamma}}\, m^\gamma_{\mu,\nu}|\mathcal{X}_\gamma\rangle. \tag{6.19}$$

With this definition, Pauli operators over a product $\mathcal{T} = \mathcal{T}_1 \times \cdots \times \mathcal{T}_m$ inherit a tensor product form $X_\mathcal{T}(a) := X_{\mathcal{T}_1}(a_1) \otimes \cdots \otimes X_{\mathcal{T}_m}(a_m)$ and $Z_\mathcal{T}(\mathcal{X}_\mu) = Z_{\mathcal{T}_1}(\mathcal{X}_{\mu_1}) \otimes \cdots \otimes Z_{\mathcal{T}_m}(\mathcal{X}_{\mu_m})$. Any operator that can be written as a product of operators of type $X_\mathcal{T}(a)$ and $Z_\mathcal{T}(\chi_\mu)$ will be called a generalized *hypergroup Pauli operator*.

We state a few *main properties* of hypergroup Pauli operators. First, it follows from (6.18) that the Pauli operators $Z_\mathcal{T}(\mathcal{X}_\mu)$ commute and form a hypergroup isomorphic to $\mathcal{T}^*$:

$$Z_\mathcal{T}(\mathcal{X}_\mu)Z_\mathcal{T}(\mathcal{X}_\nu) = Z_\mathcal{T}(\mathcal{X}_\nu)Z_\mathcal{T}(\mathcal{X}_\mu) = \sum_\gamma m^\gamma_{\mu\nu} Z_\mathcal{T}(\mathcal{X}_\gamma), \quad \text{for any } \mathcal{X}_\mu, \mathcal{X}_\nu \in \mathcal{T}^*. \tag{6.20}$$

Although it is not obvious from the definitions, we show later (theorem 6.3, eq. 6.26) that the X-Paulis $X_\mathcal{T}(a)$ are also pair-wise commuting normal operators, which are diagonal in the the character basis $\{|\mathcal{X}_\mu\rangle, \mathcal{X}_\mu \in \mathcal{T}^*\}$, and form a faithful representation of the conjugacy-class hypergroup. Precisely, for any $a, b \in \mathcal{T}$, $\mathcal{X}_\mu \in \mathcal{T}^*$, we have

$$X_\mathcal{T}(a)|\mathcal{X}_\mu\rangle = \mathcal{X}_\mu(a)|\mathcal{X}_\mu\rangle, \qquad X_\mathcal{T}(a)X_\mathcal{T}(b) = X_\mathcal{T}(b)X_\mathcal{T}(a) = \sum_c n^c_{ab} X_\mathcal{T}(c). \tag{6.21}$$

The following lemma characterizes when Pauli operators of different type commute.

Lemma 6.1 (Commutativity). The operators $X_\mathcal{T}(a)$, $Z_\mathcal{T}(\mathcal{X}_\mu)$ commute iff $\mathcal{X}_\mu(a) = 1$.

Proof. For any state $|b\rangle$ in the basis $\mathcal{B}_\mathcal{T}$, compare $Z_\mathcal{T}(\mathcal{X}_\mu)X_\mathcal{T}(a)|b\rangle = \sum_c \sqrt{\frac{w_b}{w_c}} n^c_{ab}\mathcal{X}_\mu(c)|c\rangle$ with $X_\mathcal{T}(a)Z_\mathcal{T}(\mathcal{X}_\mu)|b\rangle = \sum_c \sqrt{\frac{w_b}{w_c}} n^c_{ab}\mathcal{X}_\mu(b)|c\rangle$. Then, the "if" holds because $\mathcal{X}_\mu(a) = 1$ implies $\mathcal{X}_\mu(b) = \mathcal{X}_\mu(c)$ for any $b \in \mathcal{T}$, $c \in ab$ [252, proposition 2.4.15], and the two expressions coincide. Conversely, letting $b = e$ in these two expressions yields the "only if". □

Note that the above properties are always fulfilled by qubit [2], qudit [3] and group Pauli operators (chapter 3). In this sense, hypergroup Pauli operators provide a generalization of these concepts. Yet, as we will see in the next section, there are some remarkable properties of group Pauli operators that are fully shared by their hypergroup counterparts.

6.5.2 Hypergroup stabilizer states

We will now extend the notion of stabilizer state from groups to hypergroups.

Definition 6.1 (**Stabilizer hypergroup and stabilizer state**)**.** A *stabilizer hypergroup* $\mathcal{S}^\lambda$ is a hypergroup of commuting hypergroup Pauli operators over $\mathcal{T}$ (6.18-6.19) with an associated *stabilizer function* λ that selects an eigenvalue $\lambda(U)$ for every U in $\mathcal{S}^\lambda$.

Let $\{\mathcal{S}_i^{\lambda_i}\}_{i=1}^r$ be a *collection* of r mutually commuting stabilizer hypergroups. Then, a quantum state $|\psi\rangle$ is called a *stabilizer state* stabilized by $\{\mathcal{S}_i^{\lambda_i}\}_{i=1}^r$ if, up to normalization and global phases, it is the unique non-zero solution to the system of spectral equations

$$U|\psi\rangle = \lambda_i(U)|\psi\rangle, \quad \text{for all } U \in \mathcal{S}_i^{\lambda_i},\ i = 1, \dots, r. \tag{6.22}$$

By definition, every function λ_i is further constrained to be a *character* of $\mathcal{S}_i^{\lambda_i}$. This is necessary for the system (6.22) to admit nontrivial solutions.[a]

[a] Let $UV = \sum_W s_{UV}^W W$ for any $U, V \in \mathcal{S}_i^{\lambda_i}$ with structure constants $s_{U,V}^W$. Then (6.22) implies $UV|\psi\rangle = \lambda(U)\lambda(V)|\psi\rangle = \sum_W s_{UV}^W \lambda(W)|\psi\rangle$. Since $|\psi\rangle \neq 0$, every non-zero λ must be a character (by definition).

In this work we focus on *pure* stabilizer states and do not discuss mixed ones.

Though, in the PSF and in the group GSF (chapters 3-5), stabilizer groups can always be described efficiently in terms of generators or matrix representations of morphisms, the existence of efficient descriptions is hard to prove in the hypergroup setting. The stabilizer hypergroups $\mathcal{S}^\lambda$ in this chapter (see section 6.5.3) have efficient polylog $|\mathcal{T}|$-size classical descriptions (where $|\mathcal{T}|$ is the dimension of the Hilbert space $\mathcal{H}_\mathcal{T}$) if poly-size descriptions for the subhypergroups of $\mathcal{T}$ and $\mathcal{T}^*$ are promised to exist. We highlight that this latter condition is fulfilled for many hypergroups of interest, including conjugacy class and character hypergroups, and anyonic fusion rule theories[24]. The stabilizer hypergroups obtained from all those cases provide a powerful means to describe quantum many-body states that are uniquely defined via an equation of the form (6.22).

The definition of stabilizer hypergroup and state generalizes the standard notions used in the PSF and the Group Stabilizer Formalism. At the same time, our hypergroup Paulis also have novel interesting properties that are explained next.

Comparison 6.1 (**Relationship to group stabilizer states.**)**.** All qubit, qudit, and abelian-group stabilizer states (chapter 3) are instances of hypergroup stabilizer states over an abelian hypergroup $\overline{G}$, where G chosen to be a group of the form $\mathbb{Z}_2^m$, $\mathbb{Z}_d^m$, and $\mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_m}$ (respectively) with $\lambda(U) = +1$. Hypergroup Pauli operators and stabilizer hypergroups over $\overline{G}$ (6.18) become standard Pauli operators and stabilizer groups over G (in the notation of chapter 3).

[24] All examples mentioned belong to a class of so-called *resonance* hypergroups $\mathcal{T}$ that have integral weights, w_a, and fulfill a Lagrange theorem [262], which says that $\varpi_\mathcal{N}$ for any subhypergroup $\mathcal{N} \subset \mathcal{T}$ always divides $\varpi_\mathcal{T}$. As in the finite group case (lemma 5.1), this implies that a random set $\{a_1, \dots, a_m\} \subset \mathcal{T}$ with $m \in \Theta(\log \varpi_\mathcal{T})$ generates $\mathcal{T}$ via hyperoperations with high probability $\Omega(1 - c^m)$ form some constant $c \in (0, 1)$.

Comparison 6.2 (**Subtleties of hypergroup Pauli operators**). Interestingly, in spite of having some Pauli-like mathematical properties (section 6.5.1), hypergroup Pauli operators are *not as simple* as group Pauli operators (chapter 3): namely, they are not necessarily *unitary* and no longer *monomial* (1-sparse) matrices because the hyperoperations in (6.18) are non-invertible and return multiple outcomes. The absence of these two properties is reflected in definition 6.1. In our formalism, stabilizer *hypergroups* $\mathcal{S}^{\lambda}$ of commuting Paulis do not necessarily form *groups*. Stabilizer states are no longer restricted to be +1-eigenstates of hypergroup Pauli operators $U \in \mathcal{S}^{\lambda}$, as in chapter 3, for Pauli operators can now have zero eigenvalues[a]. This allows us to include more states in the formalism.

[a]This follows from (6.18) because nonabelian group and hypergroup characters can take zero values [261, 126].

Comparison 6.3 (**Non-commutativity up to a phase**). Hypergroup Paulis do not satisfy an identity of the form $Z_{\mathcal{T}}(\mathcal{X}_{\mu})X_{\mathcal{T}}(a) = \mathcal{X}_{\mu}(a)X_{\mathcal{T}}(a)Z_{\mathcal{T}}(\mathcal{X}_{\mu})$ in general, although this is the case when $\mathcal{T}$ is a group (chapter 3). In our setting, this is fulfilled only in some special cases, e.g., when either a or $\mathcal{X}_{\mu}$ is an invertible hypergroup element (theorem 6.3, eq. 6.23) or when $\mathcal{X}_{\mu}(a)$=1 (where $Z_{\mathcal{T}}(\mathcal{X}_{\mu})$ and $X_{\mathcal{T}}(a)$ *commute* due to lemma 6.1).

Comparison 6.4 (**Multiple stabilizer hypergroups**). The reason why we use multiple stabilizer hypergroups $\{\mathcal{S}_i^{\lambda_i}\}$ instead of merging them into a single commutative algebra is that finding a basis with hypergroup structure for the latter object is not a simple task[a]. On the other hand, we can easily keep track and exploit the available hypergroup structures by simply storing a poly-size list of pairwise commuting stabilizer hypergroups.[b]

[a]We have not investigated this problem nor whether a hypergroup basis can always be found.

[b]Throughout the chapter r will always be poly-sized and all examples we give (section 6.5.3) have $r \leq 2$.

The next two results imply that any intermediate quantum state of a hypergroup normalizer circuit (section 6.4) computation is a hypergroup stabilizer state and, hence, can be characterized concisely as a joint-eigenstate of some commuting hypergroup Pauli operators. This observation motivates our further development of these concepts.

Claim 6.1 (**Standard basis states**). Conjugacy-class and character states (6.10) are instances of hypergroup stabilizer states stabilized by *single* stabilizer hypergroups.

Theorem 6.3 (**Evolution of stabilizer states**). Normalizer gates map hypergroup Pauli operators to new hypergroup Pauli operators under conjugation and, therefore, transform hypergroup stabilizer states into stabilizer states. It follows from this and the previous claim that the quantum state of a normalizer circuit is always a stabilizer state.

Note that theorem 6.3 extends Van den Nest's theorem 3.1 for abelian group stabilizer states.

In order to prove claim 6.1, theorem 6.3, and many of the main results in the next sections, we will develop a new kind of hypergroup stabilizer formalism techniques that can cope with the ***non-monomiality*** and the ***non-unitarity*** of hypergroup Pauli operators. A central part of the chapter will be dedicated exclusively to this end. We stress the necessity to develop such techniques since currently available stabilizer-formalism methods—including the PSF [1–3, 133, 135, 136, 138, 137], the Group Stabilizer Formalism (chapters 3-5), the general Monomial Stabilizer Formalism of Van den Nest [203], and the recent XS stabilizer formalism [253]—can not be applied in our setting as they *critically exploit the monomiality/unitarity of stabilizer operators* for central tasks such as simulating Clifford/Normalizer operations, analyzing stabilizer state and code properties (e.g., code dimension, code support), and giving normal forms for stabilizer states[25]. The lack of these beneficial properties requires a change of paradigm in

[25]The role of monomiality and unitarity in the PSF has been extensively discussed in [203].

our setting.

To prove claim 6.1, we show a stronger result (**theorem 6.4** below), which gives a *normal form* for hypergroup stabilizer states; we outline the proof of theorem 6.3 below, with details referred to appendix 4.1.

Proof of theorem 6.3, part I. We show that normalizer gates transform X- and Z-type Pauli operators over $\mathcal{T}$ into new Pauli operators (which may involve products of X, Z Paulis) under conjugation. This result extends to arbitrary products of these operators.

Specifically, for any invertible element $s \in \mathcal{T}$, invertible character $\mathcal{X}_\varsigma \in \mathcal{T}^*$, automorphism α, quadratic function ξ, we can calculate this action for the normalizer gates $Z_\mathcal{T}(\mathcal{X}_\mu)X_\mathcal{T}(a)$, U_α, D_ξ, and the hypergroup QFT:

$$X_\mathcal{T}(a) \xrightarrow{Z_\mathcal{T}(\mathcal{X}_\varsigma)X_\mathcal{T}(s)} \mathcal{X}_\varsigma(a)X_\mathcal{T}(a), \qquad Z_\mathcal{T}(\mathcal{X}_\mu) \xrightarrow{Z_\mathcal{T}(\mathcal{X}_\varsigma)X_\mathcal{T}(s)} \mathcal{X}_\mu(\overline{s})Z_\mathcal{T}(\mathcal{X}_\mu), \tag{6.23}$$

$$X_\mathcal{T}(a) \xrightarrow{U_\alpha} X_\mathcal{T}(\alpha(a)), \qquad Z_\mathcal{T}(\mathcal{X}_\mu) \xrightarrow{U_\alpha} Z_\mathcal{T}(\mathcal{X}_{\alpha^{-*}(\mu)}), \tag{6.24}$$

$$X_\mathcal{T}(a) \xrightarrow{D_\xi} \xi(a)X_\mathcal{T}(a)Z_\mathcal{T}(\beta(a)), \qquad Z_\mathcal{T}(\mathcal{X}_\mu) \xrightarrow{D_\xi} Z_\mathcal{T}(\mathcal{X}_\mu), \tag{6.25}$$

$$X_\mathcal{T}(a) \xrightarrow{\text{QFT}} Z_{\mathcal{T}^*}(a), \qquad Z_\mathcal{T}(\mathcal{X}_\mu) \xrightarrow{\text{QFT}} X_{\mathcal{T}^*}(\overline{\mathcal{X}_\mu}). \tag{6.26}$$

When $\mathcal{H}_\mathcal{T} = \mathcal{H}_{\mathcal{T}_1} \otimes \cdots \otimes \mathcal{H}_{\mathcal{T}_m}$, the partial QFT over $\mathcal{T}_i$ simply transforms the ith tensor factor of the Pauli operators according to (6.26). In (6.25), β is a homomorphism from $\mathcal{T}$ to the subhypergroup of invertible characters $\mathcal{T}^*_{\text{inv}}$ that depends on ξ; in (6.24), α^{-*} is the inverse of the *dual automorphism* α^* [256]:

Definition 6.2 (Dual automorphism [256]). The *dual automorphism* of α, denoted α^*, is the automorphism of $\mathcal{T}^*$ that takes $\mathcal{X}_\mu$ to the character $\mathcal{X}_{\alpha^*(\mu)} := \mathcal{X}_\mu \circ \alpha$ for fixed $\mathcal{X}_\mu$.[a]

[a] This is a morphism because $\mathcal{X}_{\alpha^*(\mu)}\mathcal{X}_{\alpha^*(\nu)} = (\mathcal{X}_\mu \circ \alpha)(\mathcal{X}_\nu \circ \alpha) = \sum_\gamma m^\gamma_{\mu\nu}(\mathcal{X}_\gamma \circ \alpha) = \sum_\gamma m^\gamma_{\mu\nu}\mathcal{X}_{\alpha^*(\gamma)}$.

Proving (6.23-6.26) involves bulky yet beautifully structured hypergroup calculations that are carried out in appendix 4.1. □

It is worth noting that normalizer gates transform Pauli operators over $\mathcal{T}$ into Pauli operators over $\mathcal{T}$ if they are not QFTs and into Pauli operators over $\mathcal{T}^*$ otherwise[26].

6.5.3 A normal form for stabilizer states and examples

In this final subsection we give examples and a normal form for a class of hypergroup states that generalize the well-known notion of Calderbank-Shor-Steane (CSS) stabilizer states [248–250]:

Definition 6.3 (CSS stabilizer state). A hypergroup stabilizer state $|\psi\rangle$ over $\mathcal{T}$ is said of CSS type if it is uniquely stabilized by two mutually commuting stabilizer hypergroups $\mathcal{S}_Z^{\lambda_z}$, $\mathcal{S}_X^{\lambda_x}$ consisting only of Z and X Pauli operators respectively.

The standard definition of CSS state is recovered by setting $\mathcal{T} = \mathbb{Z}_2^n$ (chapter 1). For the sake of brevity, we assume $\mathcal{S}_Z^{\lambda_z}$ and $\mathcal{S}_X^{\lambda_x}$ to be maximal mutually commuting hypergroups in this section[27]. With these requirements, lemma 6.1 imposes that $\mathcal{S}_Z^{\lambda_z}$, $\mathcal{S}_X^{\lambda_x}$ must be of the form

$$\begin{aligned} \mathcal{S}_X^{\lambda_x} &= \{X_\mathcal{T}(a), a \in \mathcal{N}\}, & \lambda_x(X_\mathcal{T}(a)) &= \mathcal{X}_\varsigma(a) \\ \mathcal{S}_Z^{\lambda_z} &= \{Z_\mathcal{T}(\mathcal{X}_\mu), \mathcal{X}_\mu \in \mathcal{N}^\perp\}, & \lambda_z(Z_\mathcal{T}(\mathcal{X}_\mu)) &= \mathcal{X}_\mu(s), \end{aligned} \tag{6.27}$$

[26] Recall from section 6.4 that the hypergroup label $\mathcal{T}$ keeps track of the basis in which basis (6.10) measurements are performed and indicates which Pauli operators are diagonal in each basis.

[27] This maximality assumption is not necessary in our derivation but it shortens the proofs.

where $s \in \mathcal{T}$, $\mathcal{X}_\varsigma \in \mathcal{T}^*$, $\mathcal{N} \leq \mathcal{T}$ is a subhypergroup, and $\mathcal{N}^\perp$ is the *annihilator* of $\mathcal{N}$ (6.6).

We are now ready to prove the main technical result of this section, theorem 6.4, which characterizes the set of hypergroup stabilizer states of CSS type and leads to specific examples.

Theorem 6.4 (**Normal forms for CSS-type hypergroup stabilizer states**).
(a) The quantum states stabilized by $\{\mathcal{S}_X^{\lambda_x}, \mathcal{S}_Z^{\lambda_z}\}$ from (6.27) are those in the subspace

$$\mathcal{V}_\mathcal{S} := \text{span}\left\{ \sum_{x \in s\mathcal{N}} \psi_y(x)|x\rangle,\ y \in s\mathcal{N} \right\} = \text{span}\left\{ \sum_{\mathcal{X}_\mu \in \mathcal{X}_\varsigma \mathcal{N}^\perp} \widehat{\psi}_\nu(\mu)|\mathcal{X}_\mu\rangle,\ \mathcal{X}_\nu \in \mathcal{X}_\varsigma \mathcal{N}^\perp \right\},$$

where ψ_y and $\widehat{\psi}_\nu$ are functions supported on $s\mathcal{N}$ and $\mathcal{X}_\varsigma\mathcal{N}^\perp$, respectively, and defined by

$$\psi_y(x) := \sqrt{w_x} \left(\sum_{b \in \mathcal{N}} n^b_{x,\overline{y}} \mathcal{X}_{\overline{\varsigma}}(b) \right) \quad \text{and} \quad \widehat{\psi}_\nu(\mu) := \sqrt{w_{\mathcal{X}_\mu}} \left(\sum_{\beta \in \mathcal{N}^\perp} m^\beta_{\mu,\overline{\nu}} \mathcal{X}_\beta(s) \right). \tag{6.28}$$

A state $|\psi\rangle$ is *uniquely* stabilized by $\{\mathcal{S}_X^{\lambda_x}, \mathcal{S}_Z^{\lambda_z}\}$ iff $\dim(\mathcal{V}_\mathcal{S}) = 1$. **(b)** Furthermore, if either s or $\mathcal{X}_\varsigma$ is invertible, then $\{\mathcal{S}_X^{\lambda_x}, \mathcal{S}_Z^{\lambda_z}\}$ stabilizes a *unique* state of form[a]

$$|\psi\rangle = \sum_{x \in s\mathcal{N}} \sqrt{\frac{w_x w_{\mathcal{X}_{\overline{\varsigma}}\mathcal{N}^\perp}}{\varpi_{s\mathcal{N}}}}\, \mathcal{X}_{\overline{\varsigma}}(x)|x\rangle, \qquad \mathcal{F}_\mathcal{T}|\psi\rangle = \sum_{\mathcal{X}_\mu \in \mathcal{X}_\varsigma \mathcal{N}^\perp} \sqrt{\frac{w_{\mathcal{X}_\mu} w_{s\mathcal{N}}}{\varpi_{\varsigma\mathcal{N}^\perp}}}\, \mathcal{X}_\mu(s)|\mathcal{X}_\mu\rangle. \tag{6.29}$$

[a] Cf. section 6.2.2 for definitions of $w_{\mathcal{X}_{\overline{\varsigma}}\mathcal{N}^\perp}, w_{s\mathcal{N}}, \varpi_{s\mathcal{N}}, \varpi_{\varsigma\mathcal{N}^\perp}$.

Theorem 6.4 is proven at the end of the section after mentioning a few main applications.

We highlight that, despite our focus on stabilizer *states*, theorem 6.4 can be easily extended to study hypergroup stabilizer *codes*. For instance, in case **(b)**, we could choose a smaller stabilizer hypergroup $\mathcal{S}_X^{\lambda_x} = \{X_\mathcal{T}(a), a \in \mathcal{K}\}$ with $\mathcal{K} \subsetneq \mathcal{N}$ to obtain a stabilizer code $\mathcal{V}_\mathcal{S}$, whose dimension is easy to compute with our techniques[28]. Similarly, one could shrink $\mathcal{S}_Z^{\lambda_z}$ or both stabilizer hypergroups at the same time.

Open question The hypergroup CSS code construction we outlined clearly mimics the standard qubit one [248–250, 13]. Interestingly, there could also be hypergroup CSS codes with no qubit/qudit analogue if there exist groups (or even hypergroups $\mathcal{T}$ that do not arise from groups) for which $\mathcal{V}_\mathcal{S}$ in theorem 6.4**(a)** can be degenerate. Though this is never the case for abelian groups because the Pauli operators in (6.27) generate a maximal linearly independent set of commuting operators (with rank one common eigenprojectors), it can happen in our setting.[29] We leave open the question of whether such codes exist.

Examples and applications of theorem 6.4

Theorem 6.4 is an important technical contribution of this work that will be used three times within the scope of the chapter: firstly, in the examples below, to construct efficient classical descriptions for new kinds of complex many body states; secondly, in section 6.6, to devise classical algorithms for simulating hypergroup normalizer circuits (theorems 6.5); and finally, in section 6.7, in the development of an efficient quantum algorithm for hidden subhypergroup problems (theorems 6.8–6.9).

[28] With minor modifications of our proof of theorem 6.4, we get that the dimension is the number of cosets of $\mathcal{K}$ inside $s\mathcal{N}$, if $\mathcal{X}_\varsigma$ is invertible, and the number of cosets of $\mathcal{N}^\perp$ inside $\mathcal{X}_\varsigma\mathcal{K}^\perp$, if s is invertible.

[29] Products of hypergroup Pauli operators in (6.27) can be linearly dependent (choose $\mathcal{N} = \{\pm 1, C_i\}$ for the quaternion group, section 6.4) and their cardinality $|\mathcal{N}||\mathcal{N}^\perp|$ may not match the dimension of $\mathcal{H}_{\overline{G}}$ [264].

We now give examples of CSS hypergroup stabilizer states of the simpler form (6.29).

Example 1: standard basis states We show now that conjugacy-class states and character states (6.10) are instances of hypergroup CSS states (of type (b)), as anticipated above (claim 1). Consider, first, an arbitrary $|a\rangle$ with $a \in \mathcal{T}$. Eq. (6.20) implies that $|\psi\rangle$ is stabilized by $\mathcal{S}_Z^{\lambda_z} = \{Z_{\mathcal{T}}(\mathcal{X}_\mu), X_\mu \in \mathcal{T}^*\}$ with maximal $\mathcal{N}^\perp = \mathcal{T}^*$ and $\lambda_z(Z_{\mathcal{T}}(\mathcal{X}_\mu)) = \mathcal{X}_\mu(a)$. Letting $\mathcal{S}_X^{\lambda_x}$, $\mathcal{N}$, and λ_x be trivial, the state can written in the form given in theorem 6.4.(b) (note that λ_x is an invertible character) and, hence, $|a\rangle$ is a uniquely stabilized CSS state. An almost identical argument, using (6.21) instead, shows that any character state $|\mathcal{X}_\nu\rangle$ is uniquely stabilized by $\mathcal{S}_X^{\lambda_x} = \{X_{\mathcal{T}}(a), a \in \mathcal{T}\}$ with $\lambda_x(X_{\mathcal{T}}(a)) = \mathcal{X}_\nu(a)$. (Note that in both cases equation (6.29) reproduces (6.10) consistently.)

Example 2: hypergroup coset states The states in example 1 are always product states. Yet theorem 6.4 also implies that highly entangled states such as the abelian group coset states that appear in the abelian HSP quantum algorithms [7],

$$|x + H\rangle = \sum_{h \in H} \frac{1}{\sqrt{|H|}} |x + h\rangle, \quad H \text{ is a subgroup of a finite abelian group } G, \tag{6.30}$$

as well as the abelian hypergroup coset states used in the quantum algorithm [125],

$$|s\mathcal{N}\rangle = \sum_{x \in s\mathcal{N}} \sqrt{\frac{w_x}{\varpi_{s\mathcal{N}}}} |x\rangle, \quad \mathcal{N} \text{ is a subhypergroup of a finite abelian hypergroup } \mathcal{T}, \tag{6.31}$$

are all instances of CSS hypergroup states of type (a) with trivial $\mathcal{X}_\varsigma$, uniquely stabilized by $\mathcal{S}_X^{\lambda_x} = \{X_{\mathcal{T}}(a), a \in \mathcal{N}\}$, $\mathcal{S}_Z^{\lambda_z} = \{Z_{\mathcal{T}}(\mathcal{X}_\mu), \mathcal{X}_\mu \in \mathcal{N}^\perp\}$, $\lambda_z(Z_{\mathcal{T}}(\mathcal{X}_\mu)) = \mathcal{X}_\mu(s)$, and trivial, invertible λ_x.

In the special case of abelian group coset states (6.30), theorem 6.4 recovers a result by Van den Nest [134] who identified the latter with generalized abelian group stabilizer states (see also theorem 3.4). Theorem 6.4 extends the latter result demonstrating the existence of complex many-body states—hypergroup coset states (6.31) and more (6.28-6.29)—that can be described within the present Hypergroup Stabilizer Formalism but not within the standard PSF nor the GSF (chapters 3-5), or even (to the best of our knowledge[30]) or within other generalizations of the PSF such as the Monomial Stabilizer Formalism [203] and the X-S Stabilizer Formalism [253].

Proof of theorem 6.4

We finish this section by proving theorem 6.4 and giving a method for preparing coset states as a corollary (corollary 6.1). As announced in the previous section, the proof of this result relies on new technical ideas based on hypergroup methods, which are needed to handle the *non-unitary, non-monomial* stabilizers of theorem 6.4.

Proof of theorem 6.4. First note that the properties discussed in section 6.5.1 show that both stabilizer hypergroups $\mathcal{S}_X^{\lambda_x}$, $\mathcal{S}_X^{\lambda_z}$ are well-defined. To prove the theorem, we will use some basic hypergroup theoretic results described in the following lemma.

[30] The authors are not aware of any method to express hypergroup coset states in terms of monomial unitary stabilizers as in [1–3, 134, 63, 203, 253]. We doubt such a description could exist and, at the same time, be easy to track under the action of hypergroup normalizer circuits as in theorem 6.3.

Lemma 6.2 ([252, 2.4.15,2.4.16])**.** For any subhypergroup $\mathcal{N}$, the hypergroup isomorphisms $\mathcal{N}^* \cong \mathcal{T}^*/\mathcal{N}^\perp$ and $(\mathcal{T}/\mathcal{N})^* \cong \mathcal{N}^\perp$ (cf. section 6.2.2), can be canonically realized as follows.

(i) All characters of $\mathcal{N}$ are obtained via *restriction* of characters of $\mathcal{T}$, and two characters $\mathcal{X}_\alpha, \mathcal{X}_\beta \in \mathcal{T}^*$ act *equally* on $\mathcal{N}$ if and only if $\mathcal{X}_\alpha \in \mathcal{X}_\beta \mathcal{N}^\perp$.

(ii) All quotient characters are obtained by letting characters in $\mathcal{N}^\perp$ act on cosets $x\mathcal{N}$, and this map is well-defined because the former act *constantly* on the latter.

Next, we identify necessary and sufficient conditions for a state $|\psi\rangle$ to be uniquely stabilized by $\{\mathcal{S}_X^{\lambda_x}, \mathcal{S}_Z^{\lambda_z}\}$. First, condition (6.22) says that $|\psi\rangle$ is stabilized by $\mathcal{S}_Z^{\lambda_z}$ iff

$$Z_\mathcal{T}(\mathcal{X}_\mu)|\psi\rangle = \mathcal{X}_\mu(s)|\psi\rangle = \lambda_z(Z_\mathcal{T}(\mathcal{X}_\nu))|\psi\rangle \quad \text{for every } \mathcal{X}_\mu \in \mathcal{N}^\perp. \tag{6.32}$$

Due to lemma 6.2(ii), this holds iff the wavefunction ψ is supported on a subset of the coset $s\mathcal{N}$. On the other hand, we show that $|\psi\rangle$ is further stabilized by $\mathcal{S}_X^{\lambda_x}$ iff ψ belongs to the image of the following operator:

$$P_X := \varpi_\mathcal{N}^{-1} \sum_{b\in\mathcal{N}} w_{\mathcal{X}_\varsigma \mathcal{N}^\perp} w_b \mathcal{X}_\varsigma(b) X_\mathcal{T}(b). \tag{6.33}$$

The "only if" follows from the fact that $X_\mathcal{T}(b)|\psi\rangle = \mathcal{X}_\varsigma(b)$ and the orthogonality relationship (6.8). The "if" follows from the calculation

$$X_\mathcal{T}(a)P_X = \sum_{b,c\in\mathcal{N}} \frac{w_{\mathcal{X}_\varsigma \mathcal{N}^\perp} w_b}{\varpi_\mathcal{N}} n_{ab}^c \mathcal{X}_\varsigma(b) X_\mathcal{T}(c) = \sum_{c\in\mathcal{N}} \frac{w_{\mathcal{X}_\varsigma \mathcal{N}^\perp} w_c}{\varpi_\mathcal{N}} \left(\sum_{b\in\mathcal{N}} n_{\bar{a}c}^b \mathcal{X}_\varsigma(b) \right) X_\mathcal{T}(c) \tag{6.34}$$

$$= \sum_{c\in\mathcal{N}} \frac{w_{\mathcal{X}_\varsigma \mathcal{N}^\perp} w_c}{\varpi_\mathcal{N}} \mathcal{X}_\varsigma(\bar{a})\mathcal{X}_\varsigma(c) X_\mathcal{T}(c) = \mathcal{X}_\varsigma(a) P_X, \tag{6.35}$$

which implies with (6.8) that P_X is a *projector*, and consequently, we get $X_\mathcal{T}(a)|\psi\rangle = X_\mathcal{T}(a)P_X|\psi\rangle = \mathcal{X}_\varsigma(a)P_X|\psi\rangle = \mathcal{X}_\varsigma(a)|\psi\rangle$ as desired.

As a result, we obtain that the stabilized states of $\{\mathcal{S}_X^{\lambda_x}, \mathcal{S}_Z^{\lambda_z}\}$ are the quantum states in the vector space $\mathcal{V}_\mathcal{S} := \text{span}\{P_X|y\rangle : y \in s\mathcal{N}\}$, where

$$P_X|y\rangle = \sum_{\substack{b\in\mathcal{N} \\ x\in s\mathcal{N}}} \frac{w_{\mathcal{X}_\varsigma \mathcal{N}^\perp} w_b}{\varpi_\mathcal{N}} \sqrt{\frac{w_y}{w_x}} n_{b,y}^x \mathcal{X}_\varsigma(b)|x\rangle \propto \sum_{x\in s\mathcal{N}} \sqrt{w_x} \left(\sum_{b\in\mathcal{N}} n_{x,\bar{y}}^b \mathcal{X}_\varsigma(b) \right) |x\rangle, \tag{6.36}$$

and that $|\psi\rangle$ is uniquely stabilized iff this space is one dimensional. The proof for the RHS of (6.28) is the same: due to duality, we can apply a QFT (6.26) and reach this equality by repeating the whole proof from the beginning with exchanged roles for $\mathcal{T}$ and $\mathcal{T}^*$. This proves case **(a)**.

Finally, we prove **(b)**. First, in the simplest case, $s = e$, we can see that $\psi_y(x) = \sqrt{w_x}\mathcal{X}_\varsigma(x\bar{y}) = \sqrt{w_x}\mathcal{X}_\varsigma(x)\mathcal{X}_\varsigma(\bar{y}) = \psi_1(x)\mathcal{X}_\varsigma(\bar{y})$ by (6.3), since $x, y \in \mathcal{N}$ and $\mathcal{X}_\varsigma$ is a character of $\mathcal{N}$. When $x, y \in s\mathcal{N}$, for $s \neq e$, we can, in general, have $n_{x,\bar{y}}^z \neq 0$ for $z \notin \mathcal{N}$, so we cannot apply (6.3). However, when s is invertible (so $s\bar{s} = 1$), we can get the same result as in the simplest case by a simple change of variables.

For any $x, y \in s\mathcal{N}$, we define $x' := \bar{s}x$ and $y' := \bar{s}y$. Since $x'\bar{y}' = \bar{s}x\bar{y}s = s\bar{s}x\bar{y} = xy$, we have $n_{x'\bar{y}'}^b = n_{x\bar{y}}^b$ and, taking $b = 1$ and $y = x$, we have $w_{x'} = w_x$ from the definition of w_x. As these are the only appearances of x and y in $\psi_y(x)$, this shows that $\psi_y(x) = \psi_{y'}(x')$. This combined with the previous easy case (for $x', y' \in \mathcal{N}$), shows that all ψ_y's are proportional to the non-zero function $\psi_1(x)$, which shows that the space is one-dimensional and contains $|\psi\rangle$. Finally, it is

easily checked, in the case that $\mathcal{X}_\varsigma$ is invertible, that the normalization constant in (6.29, LHS) is $(w_{\mathcal{X}_\varsigma\mathcal{N}^\perp}/\varpi_{s\mathcal{N}})^{-1/2}$; otherwise, it follows from (6.8). As in case **(a)**, duality lets us repeat the argument to get (6.29, RHS). □

As a final remark, we highlight that theorem 6.4 introduces many new states that we are not aware to be preparable by standard or character basis inputs and normalizer gates (the ingredients of the computational model in section 6.4), in general. However, we point out that the CSS states of type theorem 6.4.(b) can always be prepared by measuring Pauli operators.

Corollary 6.1 (**Coset state preparations**). Let $\mathcal{C}$ be a circuit takes the standard basis state $|\mathcal{X}_1\rangle$ as input, performs $\mathcal{F}_\mathcal{T}^\dagger$ (an inverse QFT) or $\mathcal{F}_{\mathcal{T}^*}$, and then performs a syndrome measurement of the Pauli operators in a stabilizer hypergroup $\mathcal{S}_Z^{\lambda_z}$ of form (6.27)[a]. Then $\mathcal{C}$ prepares a coset state. Specifically, it prepares $|s\mathcal{N}\rangle$ with probability $\varpi_{s\mathcal{N}}/\varpi_\mathcal{T}$. Furthermore, if $|x_0\rangle$ is given, $\mathcal{F}_\mathcal{T}$ or $\mathcal{F}_{\mathcal{T}^*}^\dagger$ is applied, and $\mathcal{S}_X^{\lambda_x}$ (6.27) is measured, then the outcome is a coset state $\left|\mathcal{X}_\varsigma\mathcal{N}^\perp\right\rangle$ with probability $\varpi_{\varsigma\mathcal{N}^\perp}/\varpi_{\mathcal{T}^*}$.

[a] This is the canonical measurement defined by the common eigenprojectors that may be implemented, e.g., by measuring a poly-size set generating set of $\mathcal{S}_Z^{\lambda_z}$, which exists if there is one for $\mathcal{N}^\perp$ (section 6.5).

All states of form (6.29) can further be prepared from a coset state by applying Pauli gates.

Proof. If we prove the first case, the second holds due to hypergroup duality. Lemma 6.2(ii) implies that measuring $\mathcal{S}_Z^{\lambda_z}$ is equivalent to performing a projective measurement with projectors $\{P_{s\mathcal{N}} = |s\mathcal{N}\rangle\langle s\mathcal{N}|\}$. The claim follows by rewriting $\mathcal{F}_\mathcal{T}^\dagger|\mathcal{X}_1\rangle = \mathcal{F}_{\mathcal{T}^*}|\mathcal{X}_1\rangle = \sum_{a\in\mathcal{T}}\sqrt{\frac{w_a}{\varpi_\mathcal{T}}}|a\rangle = \sum_{s\mathcal{N}\in\mathcal{T}/\mathcal{N}}\sqrt{\frac{\varpi_{s\mathcal{N}}}{\varpi_\mathcal{T}}}|s\mathcal{N}\rangle$. □

6.6 Classical simulation of hypergroup normalizer circuits

In chapter 3, we gave efficient classical algorithms for simulating normalizer circuits over finite abelian groups, which recovered the standard Gottesman-Knill theorem [1, 2]. Together with the original GK theorem, these results demonstrate the existence of quantum computations that fail to give exponential quantum speed-ups despite usage of several powerful ingredients, such as maximally entangled states [41], quantum Fourier transforms and abelian-group coset states. Our next theorem extends a variant of the original Gottesman-Knill to normalizer circuits over arbitrary abelian hypergroups.

Theorem 6.5 (**Simulation**). Let $\mathcal{C}$ be a normalizer circuit over a finite abelian hypergroup $\mathcal{T}$ containing *global* QFTs, automorphism gates, and Pauli gates (but no quadratic phase gates) followed by a final measurement in the standard basis (cf. section 6.4). Then, given certain computability assumptions about $\mathcal{T}$ and its characters (section 6.6.1), there exists an efficient classical algorithm for sampling the measurement outcomes of $\mathcal{C}$.

The proof of the theorem is given at the end of the section. Our simulation result greatly expands the number of known families of quantum circuits that can be classically simulated and it also adds yet more evidence to support the idea that, for the HSP, quantum efficiency may go hand-in-hand with classical simulability.

We highlight that our theorem generalizes the so-called CSS-preserving Gottesman-Knill theorem [47] without intermediate measurements, where the only normalizer gates allowed are those that send CSS states to CSS states (definition 6.3). Our result also extends the CSS-preserving non-adaptive case of the theorems in chapter 3. Yet, theorem 6.5 does not fully extend

the ones in (chapter 3-4, [1, 2, 134, 63]), which altogether cover simulations of partial QFTs, quadratic phase gates, and intermediate Pauli-operators measurements interspersed along the circuit. Simulating these extended cases is much harder in our nonabelian setting because of the *non-unitarity and non-monomiality* of hypergroup Pauli operators (cf. discussion in section 6.5), which do not let us apply any existing techniques for manipulating stabilizer codes [1–3, 134, 63, 64, 93, 203, 133, 135, 136, 138, 137]; instead, the simulation method we give is based on the new hypergroup stabilizer techniques of section 6.5.

We stress that CSS normalizer operations can be highly nontrivial, as the quantum algorithms for abelian HSP we investigated in chapter 5 are normalizer circuits with CSS structure. Hence, theorem 6.5 could be used to simulate, e.g., Shor's discrete-log quantum algorithm gate-by-gate if the information about the hidden subgroups was not manifestly hidden and groups were presented in a factorized form (cf. chapter 5 for an extended discussion). This means, for instance, that the entanglement present in a CSS-preserving circuit can be quite substantial.

Lastly, we conjecture that our simulation result can be extended to all normalizer gates despite the non-monomiality/non-unitarity issues we discuss.

Conjecture 6.1 (**Conjecture**)**.** There exist nontrivial families of abelian hypergroups for which the normalizer circuits of theorem 6.5 can still be efficiently classically simulated if they are supplemented with partial QFTs and quadratic phase gates acting at arbitrary circuit locations, and even if operations are chosen adaptively depending on the outcome of intermediate measurements of hypergroup Pauli operators.

In conjecture 6.1, to qualify as "nontrivial", an abelian hypergroup family should not just consist of abelian groups and the weights of the elements of these hypergroups should be allowed to grow asymptotically with the number of bits needed to represent them (in order for the hypergroups not to be excessively "group-like"). The measurement of a hypergroup Pauli operator is defined via its eigenvalue decomposition[31] as in section 3.6. We consider examples of these measurements in theorem 6.7.

Discussion: extensions of theorem 6.5 In the light of our conjecture, we mention a few simpler extensions of theorem 6.5 that we are aware of.

First, note that an efficient classical simulation is still possible if the main circuit is followed by another one $\mathcal{C}'$ that contains any monomial normalizer gate (including quadratic-phase gates), which is then followed by a measurement in the standard basis[32]: such circuits can prepare more types of entangled stabilizer states like (6.17) and the quaternionic cluster state (section 6.4.2).

Second, the theorem can be easily extended to allow arbitrary CSS state/stabilizer state inputs with one further minimal assumption, namely, that their corresponding wavefunctions can be sampled both in the hypergroup element basis $\mathcal{B}_{\mathcal{T}}$ and in the character basis $\mathcal{B}_{\mathcal{T}^*}$, which lets us, in particular, simulate QFTs acting on the state (see section 6.6.1, condition (ii) and section 6.6.2). Furthermore, if this holds for the simple CSS states of theorem 6.4, then theorem 6.5 can be extended to circuits that can, e.g., prepare coset states as in corollary 6.1 and/or accept coset states as inputs.

[31] Note that the following operators are manifestly diagonalizable: any $Z(\mathcal{X}_\mu), \mathcal{X}_\mu \in \mathcal{T}^*$ (6.18); any $X(a), a \in \mathcal{T}$ (6.26); any product of commuting diagonalizable Paulis such as those in (6.27) and in theorem 6.4; any operator that is unitary equivalent to any of the latter. Unlike abelian-group Pauli operators, which are unitary (chapter 3), it is not a priori obvious whether any *product* of hypergroup Paulis always remains diagonalizable (this question was not investigated in this thesis); only diagonalizable products define measurements in conjecture 6.1.

[32] One can simply absorb those gates in the measurements [170].

6.6.1 Computability assumptions on hypergroups

In theorem 6.5, we must restrict ourselves to hypergroups with sufficient structure to let us efficiently compute within them and their character hypergroups. Note that assumptions of this kind are typically made in the HSP literature: for instance, in order for the HRT quantum algorithm for the HNSP [99] to be efficient one needs to be given the ability to intersect characters kernels. The assumptions needed for theorem 6.5 are listed next, followed by examples of hypergroups that meet them.

First, in theorem 6.5 we assume that the hypergroup $\mathcal{T}$ as well as its dual $\mathcal{T}^*$ are *efficiently computable*: we say that a hypergroup $\mathcal{T}$ is efficiently computable[33] if its elements can be uniquely represented with $n = O(\text{polylog}|\mathcal{T}|)$ bits and there are $O(\text{poly}(n))$-time classical subroutines to perform the hypergroup multiplication, i.e., given two elements x_i, $x_j \in \mathcal{T}$ and an index k, we can efficiently compute the coefficient n_{ij}^k for any i, j, k.

In theorem 6.5, we further need to assume that the involved hypergroups are what we call *doubly efficiently computable*: a hypergroup $\mathcal{T}$ is doubly efficiently computable if both $\mathcal{T}$ and $\mathcal{T}^*$ are efficiently computable and, furthermore, if the structure of their associated character tables is sufficiently well-known that we are able to efficiently perform the following tasks classically:

(i) **Computable characters.** For any $a \in \mathcal{T}$, any character function $\mathcal{X}_\mu(a)$ can be efficiently computed classically[34].

(ii) **Simulable input states.** Quantum Fourier transforms of allowed input states can be efficiently *sampled* classically, or equivalently, the distributions $\{p_a\}$ and $\{q_\mu\}$, with $p_a := \frac{w_a w_{\mathcal{X}_\mu}}{\varpi_{\mathcal{T}}}|\mathcal{X}_\mu(a)|$ for fixed $\mathcal{X}_\mu \in \mathcal{T}^*$ and $q_\mu := \frac{w_a w_{\mathcal{X}_\mu}}{\varpi_{\mathcal{T}}}|\mathcal{X}_\mu(a)|$ for fixed $a \in \mathcal{T}$, can be efficiently sampled.

(iii) **Computable dual morphisms.** For any efficiently computable hypergroup automorphism $\alpha : \mathcal{T} \to \mathcal{T}$, its inverse α^{-1} and its dual automorphism (definition 6.2) $\alpha^* : \mathcal{T} \to \mathcal{T}^* : \chi \to f_\chi^*$, can both be efficiently determined and computed. Duals of computable hypergroup homomorphisms $f : \mathcal{T} \to \mathcal{T}'$ can also be computed[35].

Examples and remarks Both computability notions presented are preserved by taking direct products $\mathcal{T}_1 \times \mathcal{T}_2$. Furthermore, the notion of doubly efficiently computable hypergroup is preserved under taking duals $\mathcal{T} \leftrightarrow \mathcal{T}^*$.

Any hypergroup of the form $\mathcal{T}_1 \times \cdots \times \mathcal{T}_m$ is doubly efficiently computable if homomorphisms are restricted to be of a product form $f_1 \times \cdots \times f_m$, where m is constant, or if they act nontrivially only in a constant number of sites. As a result, *normalizer circuits* over hypergroups of the from $\mathcal{T}_1 \times \cdots \times \mathcal{T}_m$ with constant-size $\mathcal{T}_i$ will always turn out to be efficiently simulable if they contain at most k-local entangling gates, for any constant k (theorem 6.5). The examples given in section 6.4.2 over the quaternions were of this form.

As another example, for any finite abelian group G, all problems in (i-ii-iii) can be solved in $O(\text{polylog}\,|G|)$ time given that G is explicitly given in the form $G = \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_m}$. Condition (ii) holds for any abelian group stabilizer state with known stabilizer group. These results are invariant of the bit-size of any N_i (chapter 3).

For arbitrary finite abelian hypergroups finding simple bounds like those in chapter 3 is likely to be an "impossible" problem, since the question cannot even be addressed without classifying all conjugacy class and character hypergroups of all finite groups, whereas classifying the latter is

[33] Efficiently computable hypergroups generalize the black-box groups [91] explored in chapter 5.

[34] For simplicity, we will assume that this can be done with perfect precision in this chapter. Our results readily extend if characters can be computed within an arbitrarily small error.

[35] Applying def. 6.2 to a homomorphism $f : \mathcal{T} \to \mathcal{T}'$ one gets a dual morphism $f^* : \mathcal{T}' \to \mathcal{T}$ [256, 1.6.(ii)].

regarded as a (so-called) "wild" problem [267, 268]. It is easier to prove polynomial-time bounds for particular hypergroup/group families that fulfill the minimal computability requirements (i-ii-ii), like in the two examples given above.

Finally, we highlight that some efficiently computable hypergroups are provably not *doubly* efficiently computable unless efficient classical algorithms for believed-to-be-hard problems like computing discrete-logarithms exist (see appendix 4.3). In the abelian group case, the associated normalizer circuits can realize Shor's [4] algorithms and lead to exponential quantum speed-ups (chapter 5). In section 6.7, we will develop new quantum algorithms based on normalizer circuits over such "black-box" hypergroups.

6.6.2 Proof of theorem 6.5

We finish this section proving theorem 6.5 by giving an explicit classical algorithm for sampling the outcome distribution after measuring the final state of the computation $\mathcal{C}|\psi_0\rangle$, being $|\psi_0\rangle$ the input state. Our algorithm is efficient given that the hypergroup $\mathcal{T}$ is doubly efficiently computable (section 6.6.1). The key technique that we exploit in our simulation is a normal form for CSS normalizer circuits.

Lemma 6.3 (**Normal form**). Let $\mathcal{C}$ be a normalizer circuit over a $\mathcal{T}$ as in theorem 6.5. Then, $\mathcal{C}$ can be put in a layered normal form $\mathcal{C} = MF$, where F is either trivial or a QFT and M is a monomial circuit[a] of automorphism gates and Pauli gates. Furthermore, classical descriptions of M and F can be computed classically efficiently if $\mathcal{T}$ is doubly efficiently computable.

[a] That is, a circuit whose transformation in matrix form has one entry per row and column.

Proof of lemma 6.3. First, note that the Pauli gates in the circuit (which are of the form $X_{\mathcal{T}}(C)$, $Z_{\mathcal{T}}(\mathcal{X})$ in the conjugacy-class basis and of the form $Z_{\mathcal{T}^*}(C)$, $X_{\mathcal{T}^*}(\mathcal{X})$ in the character basis) can be conjugated with all the other normalizer gates using the update rules in theorem 6.3(6.24,6.25). As a result, if U is a Pauli gate at an intermediate circuit position $\mathcal{C} = \mathcal{C}_2 U \mathcal{C}_1$, it can be removed from its location by adding a new Pauli-correction term $U' = \mathcal{C}_2 U \mathcal{C}_2^\dagger$ at the beginning of the circuit. By doing this, we put $\mathcal{C}$ in an intermediate two-layered normal form $\mathcal{C} = P\mathcal{C}'$, where P is a circuit of Pauli gates and $\mathcal{C}'$ collects all QFTs and all automorphism gates that were present in $\mathcal{C}$, in the same temporal order.

We finish the proof of the lemma by showing that $\mathcal{C}'$ can be put in a normal form AS where S is either the identity gate or a QFT, and A is a circuit of automorphism gates. Once we have that, we can combine P and A into a single layer M and obtain $\mathcal{C} = MS$. Because every gate in M is either a permutation or a diagonal unitary, M is manifestly monomial.

To this end, we use that group automorphism gates can be conjugated with Fourier transforms in an elegant way, by replacing them with dual automorphism gates. Specifically, we have $\mathcal{F}_{\mathcal{T}} U_\alpha = U_{\alpha^*}^{-1} \mathcal{F}_{\mathcal{T}} = U_{\alpha^{-*}} \mathcal{F}_{\mathcal{T}}$ and $\mathcal{F}_{\mathcal{T}^*} U_{\alpha^{-*}} = U_\alpha \mathcal{F}_{\mathcal{T}^*}$, where α^{-*} is the dual of α^{-1} (definition 6.2). This follows by simple calculation, using (6.10) and the fact that automorphisms cannot change the weights of elements. Furthermore, we in fact have

$$U_\alpha |\mathcal{X}_\mu\rangle = \sum_{a\in\mathcal{T}} \sqrt{\frac{w_a w_{\overline{\mathcal{X}_\mu}}}{\varpi_{\mathcal{T}}}}\, \overline{\mathcal{X}_\mu}(a) |\alpha(a)\rangle \overset{b:=\alpha(a)}{=} \sum_{b\in\mathcal{T}} \sqrt{\frac{w_b w_{\overline{\mathcal{X}_{\alpha^{-*}(\mu)}}}}{\varpi_{\mathcal{T}}}}\, \overline{\mathcal{X}_{\alpha^{-*}(\mu)}}(b) |b\rangle = U_{\alpha^{-*}} |\mathcal{X}_\mu\rangle.$$

Hence, it follows that $U_\alpha = U_{\alpha^{-*}}$ as gates, which means that can implement the latter since we can implement the former by assumption.

Applying these rules, we can move the QFTs before the automorphisms. Furthermore, since the effect of each QFT is just to change the designated of the circuit, any product of QFTs is

equivalent to a single QFT gate, F, that changes the basis once (or not at all). By this process, we get $\mathcal{C}' = AF$, where A is a product of automorphism gates.

Finally, the assumption that both $\mathcal{T}$ and $\mathcal{T}^*$ are efficiently computable (and, hence, so is any hypergroup of form in (6.13)) means that we can efficiently compute the conjugations to determine the gates in P using (6.24, 6.26). The assumption that $\mathcal{T}$ is doubly efficiently computable also implies that we can find α^{-*} efficiently for any α, so that we can compute the automorphisms in A efficiently as well. Finally, we efficiently obtain a classical description for M (resp. F) by listing the Pauli and automorphism gates (resp. QFTs) it contains. □

To prove the theorem, we first apply lemma 6.3 to put $\mathcal{C}$ in normal form, MF. Next, we note that M acts as $M|a\rangle = \gamma_a|\pi(a)\rangle$, where γ_a has unit modulus and π is some permutation on the elements of the basis for the image of F. Thus, measuring in the final basis, after applying M, is equivalent to measuring in the basis after F and then applying π. Because $\mathcal{T}$ is doubly efficiently computable (as defined in section 6.6.1), we can, first, compute M and F by lemma 6.3; second, simulate a measurement after applying F by assumption (ii); and, finally, compute the action of π on the obtained samples: for the latter step, we can infer a poly-size boolean circuit implementing π from the automorphism and Pauli X-gates in M—via assumption (iii) and via the circuit that implements the hypergroup multiplication. Q.E.D.

Remark As mentioned in the discussion after theorem 6.5, one can straightforwardly extend this simulation method to any input $|\psi_0\rangle$ if measurements on $|\psi_0\rangle$ on the hypergroup element and character bases are easy to simulate (because we only use that information about the state).

6.7 Quantum algorithms for HNSP and abelian HSHP

In this last section, we apply the hypergroup methods of previous sections to the development new quantum algorithms for the HNSP and for the CC-HSHP.

We give three quantum algorithms for the HNSP of increasing generality (and complexity). These algorithms are interesting because they are fundamentally different from the one of Hallgren et al. [99], as they exploit the hypergroup structure of the HNSP: to solve the problem, we turn it into a CC-HSHP (using theorems 6.1-6.2, section 6.3) and solve the resulting abelian HSHP instead. Our results show that, in the cases considered here, the HNSP is easy because the CC-HSHP is easy, which gives an explanation for why the HNSP is easy in terms of the presence of an *abelian* algebraic structure.

Our quantum algorithms for the CC-HSHP are interesting in their own right (beyond their use in solving the HNSP) as no provably efficient algorithms were previously known.

As we will see shortly, all of the quantum algorithms we consider here fit within our normalizer circuit model. This means that they can be analyzed using the stabilizer formalism of section 6.5. The results of that section, especially theorem 6.4, will be critical to all of our analysis below.

We begin in section 6.7.1 by looking at a previously proposed algorithm for the HSHP [126]. We show that, in one subclass of cases, not only can we reduce the HNSP to the CC-HSHP (as we saw in section 6.3), but in fact, the algorithm of [99] for the HNSP is identical, under a simple vector space isomorphism, to the proposed algorithm of [126] applied to the CC-HSHP. This demonstrates an even deeper connection between the HNSP and the CC-HSHP than that demonstrated by the reductions of section 6.3.

In section 6.7.2, we use our stabilizer formalism to analyze the proposed algorithm of [126]. We describe instances where it does and does not work correctly. This analysis leads us to a new algorithm, described in the same section, which works correctly for all groups.

In section 6.7.3, we develop new quantum algorithms, taking advantage of the unique structure of abelian hypergroups. The resulting algorithms work for all nilpotent (hyper)groups (along with some non-nilpotent groups) and requires fewer assumptions than those of section 6.7.2. As with the algorithms of section 6.7.2, the stabilizer formalism remains key to our analysis.

Finally, in section 6.7.4, we mention a few further results and some open problems.

6.7.1 A comparison of two simple algorithms for the HNSP

To illustrate ideas we use later and introduce the quantum algorithm proposed in [126] for the CC-HSHP, we begin by discussing it and comparing it to the standard algorithm for the HNSP [99] in the case when the oracle $f : G \to \{0,1\}^*$ is a class function. This happens if and only if G/N is abelian, where N is the hidden normal subgroup. We show that, for this case, the two algorithms actually *coincide*: the HNSP becomes an instance of the CC-HSHP and the same algorithm solves both problems.

As we saw in section 6.3, such an f is easily transformed into an oracle $\overline{f} : \overline{G} \to \{0,1\}^*$ for the CC-HSHP since each coset xN is in a conjugacy class by itself. This allows us to turn algorithms for the CC-HSHP into algorithms for the HNSP (and vice versa). We now compare two algorithms designed for this common problem via two different perspectives.

The quantum algorithm of Hallgren, Russell, and Ta-Shma for HNSP [99], henceforth referred to as "HRT", operates as follow:

1. Initialize a workspace register in a quantum state $|\chi_1\rangle$.
2. Apply an inverse QFT in order to obtain a superposition $\sum_{g\in G} |g\rangle$.
3. Evaluate the oracle on an ancillary register to obtain $\sum_g |g, f(g)\rangle$. Measure the second register to project the state of the first onto a coset state $|xN\rangle$, for some x drawn uniformly at random.
4. Apply a QFT to $|xN\rangle$ and measure the label μ of an irreducible representation.
5. Repeat the experiment T times and record the outcomes $\mu_1, \ldots, \mu_T$.
6. Determine the subgroup $\bigcap_i \ker \chi_{\mu_i}$. With exponentially high probability $1 - O(\frac{1}{2^T})$, the subgroup $\bigcap_i \ker \chi_{\mu_i}$ is the hidden subgroup N.

The quantum part of this algorithm (steps 1–5) can be implemented efficiently if we have an efficient implementation of the QFT. However, the complexity of the classical post-processing (step 6) is unknown, in general.

The quantum algorithm of Amini, Kalantar, and Roozbehani applies to the hidden subhypergroup problem (HSHP). When applied to the conjugacy class hypergroup, $\overline{G}$, we refer to this algorithm as "AKR". It takes as input an oracle $\overline{f} : \overline{G} \to \{0,1\}^*$ and operates as follows:

1. Initialize a workspace register in a quantum state $|\mathcal{X}_1\rangle$.
2. Apply an inverse QFT in order to obtain a superposition $|G|^{-1/2} \sum_{C_x\in\overline{G}} \sqrt{w_{C_x}} |C_x\rangle$.
3. Evaluate the oracle on an ancillary register to obtain $|G|^{-1/2} \sum_{C_x} \sqrt{w_{C_x}} \left|C_x, \overline{f}(C_x)\right\rangle$. Measure the second register to project the state of the first onto a hypergroup coset state $\left|C_x\overline{N}_G\right\rangle$, for some x drawn uniformly at random.[36]

[36] Note that we are working in the Hilbert space with basis $\{|C_x\rangle \mid C_x \in \overline{G}\}$ and the state $\left|C_x\overline{N}_G\right\rangle$ is a superposition of those conjugacy classes that make up $C_x\overline{N}_G$.

4. Apply a QFT to the state $\left|C_x\overline{N}_G\right\rangle$ and measure the label $\mathcal{X}_\mu$ of a character.
5. Repeat the experiment T times and record the outcomes $\mathcal{X}_{\mu_1}, \ldots, \mathcal{X}_{\mu_T}$.
6. Determine the subhypergroup of classes in the kernels of all the $\mathcal{X}_{\mu_i}$'s.

Like HRT, steps 1–5 can be implemented efficiently while the complexity of step 6 is unknown, in general.

As the reader can see, the two algorithms perform the same steps. First, they apply an inverse Fourier transform to prepare a superposition over the entire basis on which they operate. Next, they apply their respective oracles to an additional register, measure, and throw away its value. Finally, they apply a Fourier transform and measure in the new basis. For AKR, this is measuring a character label, while for HRT, this is measuring the label of an irrep. It is critical to note that HRT does not use the value of the matrix index register, which is part of the output of the QFT for the group.

When f is a class function (so that G/N is abelian), we have $C_x\overline{N}_G = xN$ since each coset xN is in its own conjugacy class. Hence, we can see that the two algorithms are in matching states after steps 1–3. In particular, the state of HRT after step 3 is conjugation invariant, and since the QFT preserves conjugation invariance, so is the state of HRT after step 4. In fact, it is easy to check that the QFT of G applied to the conjugation invariant subspace is exactly the QFT of $\overline{G}$. More precisely, we have the following result, which is also easy to check.

Proposition 6.1 (HRT = AKR). If $f : G \to \{0,1\}^*$ is a class function, then HRT operates entirely within the conjugation invariant subspace $\mathcal{H}_{\overline{G}}$. Furthermore, within this subspace, HRT is identical to AKR.

The first part of the following lemma simplifies our analysis of the algorithm. (And the second part will be useful to us later on.)

Lemma 6.4 ([255])**.** For any normal subgroups N, K such that $N \trianglelefteq K \trianglelefteq G$, the hypergroup $\overline{G}/\overline{N}_G$ is isomorphic to $\overline{G/N}$ (the class hypergroup of G/N) and $\overline{K}_G/\overline{N}_G$ is a subhypergroup of $\overline{G}/\overline{N}_G$ isomorphic to $\overline{K/N}_{G/N}$ (which is a subhypergroup of $\overline{G/N}$).

Since G/N is abelian, the quotient hypergroup $\overline{G}/\overline{N}_G \cong \overline{G/N}$ is actually a group. Hence, it follows immediately from the standard results on Fourier sampling of abelian groups [49] that both algorithms are correct in this case as they are actually just performing Fourier sampling of an abelian group.

Theorem 6.6 (CC-HSHP is easy, I). Let G be a group. Suppose that we are given a function $f : \overline{G} \to \{0,1\}^*$ hiding the subhypergroup corresponding to a normal subgroup $N \lhd G$ such that G/N is abelian, and suppose that we can efficiently compute the QFT for $\overline{G}$. Then there is an efficient quantum algorithm for the CC-HSHP.

As a corollary of theorem 6.6, we can see that there is an efficient hypergroup-based algorithm for solving HNSP in the case when the oracle $f : G \to \{0,1\}^*$ is a class function.

Corollary 6.2 (HNSP is easy, I). Let G be a group. Suppose that we are given a hiding function $f : G \to \{0,1\}^*$ that is also a class function. If we can efficiently compute the QFT for G and compute efficiently with conjugacy classes of G, then we can efficiently solve this HNSP.

Proof. This follows since we can efficiently reduce the HNSP to a CC-HSHP by theorem 6.1, we can use the QFT of G to implement the QFT of $\overline{G}$ as described in appendix 4.4, and we can solve this CC-HSHP efficiently by theorem 6.6. □

6.7.2 Analysis of the algorithm of Amini et al.

In this subsection, we analyze the AKR algorithm in more detail. Our analysis will benefit from our hypergroup stabilizer formalism tools (section 6.5), namely, our normal forms in theorem 6.4, which will let us characterize the outcome probability distribution of the quantum algorithm.[37] This is possible due to the following connection with our normalizer circuit framework.

Theorem 6.7 (AKR is normalizer). For any finite group G, the AKR quantum algorithm for the CC-HSHP over $\overline{G}$ is a normalizer circuit with intermediate Pauli measurements. Furthermore, all of its intermediate quantum states are CSS stabilizer states of form (6.29).[a]

[a] Although we do not discuss the full AKR algorithm, this theorem also holds for all abelian hypergroups.

Proof. Steps 1-3 implement a coset state preparation scheme as in corollary 6.1, using the oracle as a black box to perform a syndrome measurement of type $\mathcal{S}_Z^\lambda$ (cf. the proofs of corollary 6.1, theorem 6.4 for details). Step 4 takes the QFT of a coset state of form (6.29). □

Using this connection, we can apply the tools developed in the previous sections to compute the probability of measuring a given character label $\mathcal{X}_\mu$ at step 5.

The mixed state of AKR after the oracle is called and its value discarded is given by

$$\rho = \sum_{C_x\overline{N}_G \in \overline{G}/\overline{N}_G} \frac{w_{C_x\overline{N}_G}}{\varpi_{\overline{G}/\overline{N}_G}} \left|C_x\overline{N}_G\right\rangle\!\left\langle C_x\overline{N}_G\right|$$

since the probability of measuring each coset $C_x\overline{N}_G$ is proportional to its weight.[38] Each coset state $\left|C_x\overline{N}_G\right\rangle$ is a stabilized by $X_{\overline{G}}(C_y)$ for any $C_y \in \overline{N}_G$ and any $Z_{\widehat{G}}(\mathcal{X}_\mu)$ for any $\mathcal{X}_\mu \in \overline{N}_G^\perp$, which means that it is a CSS stabilizer state of the form shown in equation (6.27) with $s = C_x$ and $\mathcal{X}_\varsigma = \mathcal{X}_1$, the trivial character. Thus, we can read off the Fourier transform of this state directly from theorem 6.4.

Our application of theorem 6.4 is greatly simplified by the fact that $\mathcal{X}_{\overline{\varsigma}}$ is the trivial character $\mathcal{X}_1$. This means that $w_{\mathcal{X}_{\overline{\varsigma}}\overline{N}_G^\perp} = 1$ since the weight of a character is the dimension of the underlying representation μ.[39] This means that the state $|\psi\rangle$ for $s = C_x$ and $\mathcal{N} = \overline{N}_G$ is proportional to $\sum_{C_y \in C_x\overline{N}_G} \sqrt{w_{C_y}}|C_y\rangle$, which is proportional to $\left|C_x\overline{N}_G\right\rangle$, and thus, we have $|\psi\rangle = \left|C_x\overline{N}_G\right\rangle$ since both states are normalized. Thus, theorem 6.4 tells us

$$\mathcal{F}_{\overline{G}}\left|C_x\overline{N}_G\right\rangle = \sum_{\mathcal{X}_\mu \in \overline{N}_G^\perp} \sqrt{\frac{w_\mu w_{C_x\overline{N}_G}}{\varpi_{\overline{N}_G^\perp}}}\, \mathcal{X}_\mu(C_x)|\mathcal{X}_\mu\rangle,$$

[37] This is a new result. No formula for this probability was given in [125, 126].

[38] The initial superposition has probability of measuring C_x proportional to w_{C_x}, so the probability of measuring the coset $C_x\overline{N}_G$, which contains all the elements with the same value from the oracle as C_x, is proportional sum of all of their weights, $\varpi_{C_x\overline{N}_G}$. These sums are proportional to the weights $w_{C_x\overline{N}_G}$, so we get the form in the equation above after normalizing the probabilities to sum to 1.

[39] Note that $\mathcal{X}_{\overline{\varsigma}}\overline{N}_G^\perp$ is an element of $\widehat{G}/\overline{N}_G^\perp \cong \overline{N}_G^*$ (section 6.2.2), so this is the weight of $\mathcal{X}_{\overline{\varsigma}}$ viewed as a character of $\overline{N}_G$, where it remains trivial.

and hence, the Fourier transform of the mixed state of AKR is

$$\begin{aligned}\mathcal{F}_{\overline{G}}\rho\mathcal{F}_{\overline{G}}^{\dagger} &= \sum_{C_x\overline{N}_G\in\overline{G}/\overline{N}_G}\frac{w_{C_x\overline{N}_G}}{\varpi_{\overline{G}/\overline{N}_G}}\left(\sqrt{w_{C_x\overline{N}_G}}\sum_{\mu\in\overline{N}_G^{\perp}}\sqrt{\frac{w_\mu}{\varpi_{\overline{N}_G^{\perp}}}}\mathcal{X}_\mu(C_x)|\mathcal{X}_\mu\rangle\right)\times \\ &\qquad\left(\sqrt{w_{C_x\overline{N}_G}}\sum_{\nu\in\overline{N}_G^{\perp}}\sqrt{\frac{w_\nu}{\varpi_{\overline{N}_G^{\perp}}}}\mathcal{X}_\nu(C_x)\langle\mathcal{X}_\mu|\right) \\ &= \sum_{\mu,\nu\in\overline{N}_G^{\perp}}\sqrt{w_\mu w_\nu}\left(\sum_{C_x\overline{N}_G\in\overline{G}/\overline{N}_G}\frac{w_{C_x\overline{N}_G}^2}{\varpi_{\overline{G}/\overline{N}_G}^2}\mathcal{X}_\mu(C_x)\overline{\mathcal{X}_\nu}(C_x)\right)|\mathcal{X}_\mu\rangle\langle\mathcal{X}_\nu|,\end{aligned}$$

where, in the last step, we have used the fact that $\overline{N}_G^{\perp}\cong\left(\overline{G}/\overline{N}_G\right)^*$. Finally, using lemma 6.4, we conclude that the probability of measuring the outcome $\mathcal{X}_\mu$ is

$$\Pr(\mathcal{X}_\mu)=w_\mu\sum_{C_x\overline{N}_G\in\overline{G}/\overline{N}_G}\frac{w_{C_x\overline{N}_G}^2}{\varpi_{\overline{G}/\overline{N}_G}^2}\mathcal{X}_\mu(C_x)\overline{\mathcal{X}_\mu}(C_x)=w_\mu\sum_{C_{xN}\in\overline{G/N}}\frac{w_{C_{xN}}^2}{\varpi_{\overline{G/N}}^2}\mathcal{X}_\mu(C_x)\overline{\mathcal{X}_\mu}(C_x). \tag{6.37}$$

This formula is the key to our analysis of AKR in this section and the next.

6.7.2.1 Non-convergence of AKR for simple instances

We begin examining these probabilities by looking at an example.

Example 6.1 (AKR Counterexample). The Heisenberg group over $\mathbb{Z}_p$ (with p prime) is the set $\mathbb{Z}_p^3$ with multiplication defined by $(x,y,z)\cdot(x',y',z')=(x+x',y+y',z+z'+xy')$. For a nice review of its representation theory, see the article of Bacon [269].

The center of this group is the normal subgroup $Z(G)=\{(0,0,z)\,|\,z\in\mathbb{Z}_p\}$. Hence, any element of $Z(G)$ is in a conjugacy class of its own. For any $(x,y,z)\notin Z(G)$ (i.e., with $(x,y)\neq(0,0)$), it is easy to check that its conjugacy class is the coset $(x,y,z)Z(G)$. Hence, the weight of the former classes are 1 and those of the latter are $|Z(G)|=p$.

Let us consider the case when the hidden subgroup is trivial $N=\{e\}$. For any $(a,b)\neq(0,0)$ there is a 1-dimensional irrep of the Heisenberg group, which we denote $\mathcal{X}_{a,b}$ given by $\mathcal{X}_{a,b}(x,y,z)=\omega_p^{ax+by}$, where ω_p is a p-th root of unity. We can apply equation (6.37) to compute the probability of measuring this irrep in AKR. To do this, we first note that this probability would be the inner product of $\mathcal{X}_\mu$ with itself except that weights have been squared. Since there are only two sizes of conjugacy classes, it is not difficult to rewrite this expression in terms of that inner product as follows.

$$\begin{aligned}\Pr(\mathcal{X}_{a,b}) &= \sum_z\frac{1}{p^6}|\mathcal{X}_{a,b}(0,0,z)|^2+\sum_{(x,y)\neq(0,0)}\frac{p^2}{p^6}|\mathcal{X}_{a,b}(x,y,0)|^2 \\ &= \frac{1}{p^6}\sum_z 1+\frac{p^2}{p^6}\sum_{(x,y)\neq(0,0)}1 \\ &= \frac{1}{p^6}p+\frac{p^2}{p^6}(p^2-1) \\ &= \frac{p^4-p^2+p}{p^6}\end{aligned}$$

where we have used the fact that $|\mathcal{X}_{a,b}(x,y,z)| = 1$ for all $x, y, z \in \mathbb{Z}_p$.

Finally, the probability of measuring any of the 1-dimensional irreps is

$$\sum_{(a,b)\neq(0,0)} \Pr(\mathcal{X}_{a,b}) = (p^2-1)\frac{p^4-p^2+p}{p^6} = \frac{p^6-2p^4+p^3+p^2-p}{p^6} = 1 - O\left(\frac{1}{p^2}\right).$$

This means that if p is exponentially large, we are unlikely to ever see an irrep other than the $\mathcal{X}_{a,b}$'s, and since $Z(G)$ is in the kernel of all such irreps, the intersection of the kernels of polynomially many irreps will include $Z(G)$ with high probability.

Since the AKR algorithm returns the intersection of the kernels of the sampled irreps as its guess of the hidden subgroup, this demonstrates that the AKR algorithm will fail to find the hidden subgroup with high probability for the Heisenberg group with $N = \{e\}$. Indeed, this shows that AKR will fail to distinguish between $N = Z(G)$ and $N = \{e\}$, despite the fact that the former is exponentially larger than the latter.

The probability distribution over irreps established by AKR in this example (and many others) favors the small dimensional irreps, whereas the distribution established by HRT favors the large dimensional irreps. In this example, that fact prevents AKR from ever seeing the irreps needed to uniquely determine N. (Paradoxically, when finding non-normal hidden subgroups, it is often the small irreps that are most useful and HRT that struggles to find them.[40])

6.7.2.2 An application of AKR: a 2nd hypergroup algorithm for the HNSP

While AKR may fail to uniquely determine N from its samples, the following lemma tells us that it will, with high probability, learn something about N.

Lemma 6.5. If $N \neq G$, then the intersection K of the kernels of the irreps sampled by AKR is a strict subgroup of G — i.e., we will have $N \leq K \lneq G$ — with high probability.

Proof. The intersection of the kernels will be smaller than G provided that at least one of the samples has a kernel smaller than G. In other words, the intersection will be a strict subgroup provided that at least one of the irreps sampled is not the trivial irrep.

We can use eq. (6.37) to calculate the probability of sampling the trivial irrep. Since $\mathcal{X}_1(C_x) = 1$ for any C_x, the probability for the trivial irrep is just $\sum_{C_{xN}\in\overline{G/N}} w_{C_{xN}}^2/\varpi_{\overline{G/N}}^2$. If we let G' denote the group G/N, then we can also write this as $\sum_{C_{x'}\in\overline{G'}} w_{C_{x'}}^2/\varpi_{\overline{G'}}^2$. Now, our job is to determine how close this can be to 1.

Let $C_1, \ldots, C_m$ be the conjugacy classes of G', and define $s_i = w_{C_i}/\varpi_{G'} = |C_i|/|G'|$. The s_i's satisfy $\sum_i s_i = 1$ (since every element of G' is in some conjugacy class). Since the size of a conjugacy class divides the size of the group, we also know that $s_i \leq 1/2$ for every i. Forgetting everything but these constraints, we can upper bound the probability of sampling the trivial irrep by the solution of the optimization problem

$$\text{maximize} \quad \sum_i s_i^2, \qquad \text{subject to} \quad \sum_i s_i = 1, \qquad s_i \leq \tfrac{1}{2} \qquad \text{for } i = 1 \ldots m.$$

The latter problem can be solved by ordinary methods of calculus. In particular, it is easy to check that the objective is increased if s_i and s_j are replaced by $s_i + \epsilon$ and $s_j - \epsilon$ provided that $s_i > s_j$. (I.e., the derivative in this direction is positive.) Hence, the objective will be

[40] While the AKR distribution would be better, AKR does not apply to finding non-normal hidden subgroups since the hypergroup structure is no longer present.

maximized when the two largest s_i's have value $1/2$ and all other s_i's are 0. At that point, the objective is $(1/2)^2 + (1/2)^2 = 1/2$.

This tells us that the probability of sampling the $\mathcal{X}_1$ is at most $1/2$ on each trial. Hence, the probability that all the samples are the trivial irrep is exponentially small. □

This lemma tells us that AKR will find, with high probability, a strict subgroup $K \lneq G$ such that $N \leq K$. This means that we can reduce the problem of finding N hidden in G to the problem of finding N hidden in K, which is a strictly smaller group. Provided that we understand the representation theory of K as well and, in particular, have a QFT for it, then we can recursively solve this problem. In more detail, we have:

Theorem 6.8 (**CC-HSHP is Easy, II**)**.** Let G be a group and N a normal subgroup. Suppose that, for each normal subgroup K satisfying $N \leq K \leq G$, we have a function $\overline{f}_K : \overline{K} \to \overline{H_K}$ that hides $\overline{N}_K$.[a] If we can efficiently compute the QFT for each such $\overline{K}$, then there is an efficient quantum algorithm for the CC-HSHP.

[a] Alternatively, we may assume that, for any $K \lhd G$, we have a function $\overline{f}_K$ that hides $\overline{(N \cap K)}_K$.

As a corollary, we can see that there is an efficient hypergroup-based algorithm for solving HNSP in the case when the oracle $f : G \to H$ is a group homomorphism.

Corollary 6.3 (**HNSP is easy, II**)**.** Let G be a group. Suppose that we are given a hiding function $f : G \to H$ for $N \lhd G$ that is a group homomorphism. If we can efficiently compute the QFT for any normal subgroup K satisfying $N \leq K \leq G$ and compute efficiently with conjugacy classes for any K and H_K, where H_K is the image of $f|_K$, then we can efficiently solve this HNSP.

Proof. The restriction of f to K, $f|_K : K \to H_K$, is itself a group homomorphism, so by the assumptions of the corollary and theorem 6.2, we can efficiently compute from it a CC-HSHP oracle $\overline{f}_K : \overline{K} \to \overline{H_K}$. Hence, the result follows from theorem 6.8 and the results of appendix 4.4 on implementing the QFT of a hypergroup with the QFT of the group. □

We finish this subsection comparing the quantum and classical requirements of our quantum algorithms in theorem 6.8, corollary 6.3 to those of HRT's.

(i) Post-processing requirements: Similarly to HRT's, the quantum algorithms in theorem 6.8, corollary 6.3 are information theoretic and are only fully-efficient if certain subroutines are given (e.g., as oracles) to carry out classical post-processing tasks. Specifically, the HRT quantum algorithm (step 6 above), relies on a subroutine to compute kernels of group irreps. In our case, it is easy to show that the *same* kernel-intersection subroutine is necessary and sufficient in order to compute a description of $\overline{K}$ (resp. K) in every iteration; hence, our algorithms and HRT's have *identical* post-processing requirements[41].

(ii) Reduction requirements (only for corollary 6.3) : Our HNSP quantum algorithm in corollary 6.3 requires the ability to *efficiently* convert an oracle $f : G \to X$ for the HNSP into an oracle $\overline{f} : \overline{G} \to X'$ for the CC-HSHP by some procedure. E.g. we know that such a procedure exists if f is a group homomorphism and we can compute with conjugacy classes (section 6.3).

[41] The proof of our claim is straightforward. Our algorithms compute kernels of measured irreps of G when restricted to a normal subgroup in every iteration. An algorithm that intersects kernels of irreps can be used to find $\ker(\nu|_K)$ since the latter equals $\ker \nu \cap K$ and K itself can be written as a irrep-kernel intersection by induction. Conversely, given a list of irreps, one can find the intersection of their kernels by repeatedly computing the kernel of the next irrep restricted to the intersection of the kernels of those previous.

It is important to note that any HNSP oracle can always be converted into an HSHP. Indeed, all one needs to convert f into $\overline{f}$ is to make f *worse* by *forgetting* how to *distinguish* two conjugate cosets xN, x^aN. Of course, this can be done classically in (at most) exponential time; an open question (which we leave open to future research) is whether a *polynomial-time* reduction between these problems always exists.

(iii) Quantum-circuitry requirements: The quantum steps of our algorithms above rely on

(iii.a) (In both algorithms) our ability to implement QFTs over normal subgroups of G;

(iii.b) (In the CC-HSHP case) our ability to construct the intermediate oracle $\overline{f}$ for any K.

Requirement (iii.a) is not needed in the original HRT algorithm [223] but appears in our setting because we use recursion. Requirement (iii.b) may be prohibitive for groups, yet, when fulfilled, it gives us a general quantum algorithm that works *for any group* G.

We highlight that Requirements (iii.a-iii.b) are not fundamental. In next section, we give improved quantum algorithms for *specific* groups and hypergroups (including nilpotent ones), exploiting additional algebraic structure to bypass these two assumptions. Yet, our quantum algorithms in theorem 6.8, corollary 6.1 are *conceptually* simpler and group-independent at the cost of having these two extra assumptions.

6.7.3 Efficient quantum algorithm for the nilpotent group HNSP and CC-HSHP

In this section, we present our most sophisticated hypergroup-based quantum algorithms for the HNSP and the CC-HSHP. Unlike those in previous sections, our new algorithm does not require any extra assumptions about subgroups (such as hiding functions for them or the ability to compute efficiently with their conjugacy classes). Remarkably, our algorithm for the HNSP is fundamentally different from HRT's, showing that this central problem can be solved efficiently via a hypergroup approach if the hidden-subgroup oracle is a group homomorphism of a nilpotent group.

Our approach takes advantage of unique properties of hypergroups. In particular, as we are looking to reduce our problem on G to a subproblem, we note that, for any normal subgroup $K \leq G$, there are actually two smaller hypergroups associated to it. The first is the hypergroup $\overline{K}$ that we get by looking at K as a group separate from G. The second is the subhypergroup $\overline{K}_G$, where $\overline{K}_G$ contains the conjugacy classes $C_x \in \overline{G}$ such that $x \in K$. Above, when we recursively solved a problem in K, we were using the hypergroup $\overline{K}$. However, as we will see in this section, it is also possible to solve the subproblem on the subhypergroup $\overline{K}_G \leq \overline{G}$.

The subhypergroup $\overline{K}_G$ has two advantages over $\overline{K}$. The first is that a CC-HSHP oracle for $\overline{G}$ is also a CC-HSHP oracle for $\overline{K}_G$: since the conjugacy classes of the two hypergroups are the same, the condition that the oracle is constant on conjugacy classes of $\overline{G}$ means that the same is true for $\overline{K}_G$. The second advantage of this subhypergroup is given in the following lemma.

Lemma 6.6 (Subhypergroup QFT). Let $K \triangleleft G$. If we can efficiently compute $\mathcal{F}_{\overline{G}}$, the QFT over $\overline{G}$, then we can also efficiently compute $\mathcal{F}_{\overline{K}_G}$, i.e., the QFT over $\overline{K}_G$.

Proof. In fact, the QFT for $\overline{K}_G$ is implemented by the same QFT as for $\overline{G}$ provided that we choose the *appropriate basis* for the dual of $\overline{K}_G$.

To describe this basis, we first note that every character on $\overline{G}$ is a character on $\overline{K}_G$ simply by restricting its domain to $\overline{K}_G$. This map of $\overline{G} \to \overline{K}_G$ is in fact surjective with kernel $\overline{K}_G^{\perp}$

[252]. This means that the characters of $\overline{K}_G$ are in 1-to-1 correspondence with the cosets of $\overline{K}_G^\perp$ in $\widehat{G}$ (i.e., $\overline{K}_G^* \simeq \widehat{G}/\overline{K}_G^\perp$), so our basis for characters of $\overline{K}_G$ should be a basis of cosets of $\overline{K}_G^\perp$ in $\widehat{G}$. As described in the example of section 6.5.3, the cosets of $\overline{K}_G^\perp$ are of the form $\left|\mathcal{X}_\nu \overline{K}_G^\perp\right\rangle = \sum_{\mathcal{X}_\mu \in \mathcal{X}_\nu \overline{K}_G^\perp} \sqrt{w_\mu / \varpi_{\mathcal{X}_\nu \overline{K}_G^\perp}} |\mathcal{X}_\mu\rangle$. Note that $\varpi_{\mathcal{X}_\mu \overline{K}_G^\perp} = w_{\mathcal{X}_\mu \mathcal{K}^\perp} \varpi_{\overline{K}_G^\perp} = \varpi_{\mathcal{X}_\mu \overline{K}_G^\perp} \varpi_{(\overline{G}/K)^*} = \varpi_{\mathcal{X}_\mu \overline{K}_G^\perp} \varpi_{\overline{G}/K} = \varpi_{\mathcal{X}_\mu \overline{K}_G^\perp} \varpi_{\overline{G}} / \varpi_{\overline{K}_G}$, using the definitions in section 6.2. This means that we can write the state instead as

$$\left|\mathcal{X}_\nu \overline{K}_G^\perp\right\rangle = \sum_{\mathcal{X}_\mu \in \mathcal{X}_\nu \overline{K}_G^\perp} \sqrt{\frac{w_\mu \, \varpi_{\overline{K}_G}}{w_{\mathcal{X}_\mu \overline{K}_G^\perp} \varpi_G}} |\mathcal{X}_\mu\rangle,$$

which is the definition we will use below.

With that basis chosen, we can now calculate the Fourier transform of a conjugacy class state $|C_x\rangle$ with $C_x \in \overline{K}_G$. The key fact we will use below is that $\mathcal{X}_\mu(C_x) = \mathcal{X}_\nu(C_x)$ whenever $\mathcal{X}_\mu \in \mathcal{X}_\nu \overline{K}_G^\perp$ since they differ only by multiplication with a character that is identity on C_x (lemma 6.2.(i)). Hence, we can define $\mathcal{X}_\nu \overline{K}_G^\perp(C_x)$ to be this common value.

$$\begin{aligned}
\mathcal{F}_{\overline{G}} |C_x\rangle &= \sqrt{\frac{w_{C_x}}{\varpi_G}} \sum_{\mathcal{X}_\mu \in \widehat{G}} \sqrt{w_\mu} \mathcal{X}_\mu(C_x) |\mathcal{X}_\mu\rangle \\
&= \sqrt{\frac{w_{C_x}}{\varpi_G}} \sum_{\mathcal{X}_\nu \overline{K}_G^\perp \in \widehat{G}/\overline{K}_G^\perp} \mathcal{X}_\nu \overline{K}_G^\perp(C_x) \sum_{\mu \in \mathcal{X}_\nu \overline{K}_G^\perp} \sqrt{w_\mu} |\mathcal{X}_\mu\rangle \\
&= \sqrt{\frac{w_{C_x}}{\varpi_G}} \sum_{\mathcal{X}_\nu \overline{K}_G^\perp \in \widehat{G}/\overline{K}_G^\perp} \mathcal{X}_\nu \overline{K}_G^\perp(C_x) \sqrt{\frac{w_{\mathcal{X}_\nu \overline{K}_G^\perp} \varpi_G}{\varpi_{\overline{K}_G}}} \left|\mathcal{X}_\nu \overline{K}_G^\perp\right\rangle \\
&= \sqrt{\frac{w_{C_x}}{\varpi_{\overline{K}_G}}} \sum_{\mathcal{X}_\nu \overline{K}_G^\perp \in \widehat{G}/\overline{K}_G^\perp} \sqrt{w_{\mathcal{X}_\nu \overline{K}_G^\perp}} \mathcal{X}_\nu \overline{K}_G^\perp(C_x) \left|\mathcal{X}_\nu \overline{K}_G^\perp\right\rangle
\end{aligned}$$

Because $\overline{K}_G^* \cong \widehat{G}/\overline{K}_G^\perp$ and $\varpi_{\overline{K}_G} = \varpi_{\overline{K}_G^*}$, this last line is, by definition, the QFT for $\overline{K}_G$, so we have seen that the QFT for $\overline{G}$ implements this QFT as well. □

The lemma shows that the assumption that we have an efficient QFT for the whole hypergroup $\overline{G}$ is sufficient to allow us to recurse on a subproblem on $\overline{K}_G$ without having to assume the existence of another efficient QFT specifically for the subproblem, as occurred in our last algorithm.

Our next task is to analyze the AKR algorithm applied to this hypergroup and, in particular, determine the probability distribution that we will see on character cosets when we measure. Recall that the algorithm starts by preparing a weighted superposition over $\overline{K}_G$ (which is a uniform distribution over K) and invoking the oracle. The result is

$$\rho = \sum_{C_x \overline{N}_G \in (\overline{N}_G / \overline{K}_G)} \frac{w_{C_x \overline{N}_G}}{\varpi_{(\overline{K}_G / \overline{K}_G)}} \left|C_x \overline{N}_G\right\rangle \left\langle C_x \overline{N}_G\right|.$$

As with AKR, we can find the Fourier transform of this state directly from theorem 6.4. Following the same argument as before, we see that the $|\psi\rangle$ from part (b) with $\mathcal{X}_\varsigma = \mathcal{X}_1$, the trivial character, and $s = C_x$ is precisely the state $\left|C_x \overline{N}_G\right\rangle$. Note that, since we are no longer working in $\overline{G}$ but the subhypergroup $\overline{K}_G$, we do a QFT over $\overline{K}_G$ (lemma 6.6) and the subhypergroup $\mathcal{N}^\perp = \overline{N}_G^\perp$ in theorem 6.4 belongs to $\overline{K}_G^*$.

$$\mathcal{F}_{\overline{K}_G} \left|C_x \overline{N}_G\right\rangle = \sum_{\mathcal{X}_\mu \in \overline{N}_G^\perp \leq \overline{K}_G^*} \sqrt{\frac{w_{\mathcal{X}_\mu} w_{C_x \overline{N}_G}}{\varpi_{\overline{N}_G^\perp}}} \mathcal{X}_\mu(C_x) |\mathcal{X}_\mu\rangle,$$

By a similar calculation to before, we have

$$\mathcal{F}_{\overline{K}_G}\rho\mathcal{F}^\dagger_{\overline{K}_G} = \sum_{\mathcal{X}_\mu,\mathcal{X}_\nu\in\overline{N}_G^\perp}\sqrt{w_{\mathcal{X}_\mu}w_{\mathcal{X}_\nu}}\sum_{C_x\overline{N}_G\in(\overline{K}_G/\overline{N}_G)}\frac{w^2_{C_x\overline{N}_G}}{\varpi^2_{\overline{N}_G^\perp}}\mathcal{X}_\mu(C_x)\mathcal{X}_{\bar{\nu}}(C_x)|\mathcal{X}_\mu\rangle\langle\mathcal{X}_\nu|.$$

Finally, using $\overline{N}_G^\perp \cong (\overline{K}_G/\overline{N}_G)^* \cong (\overline{K/N}_{G/N})^*$ (lemma 6.4), we conclude that the probability of measuring $\mathcal{X}_\mu \in \overline{N}_G^\perp$ is

$$\begin{aligned}\Pr(\mathcal{X}_\mu) &= w_{\mathcal{X}_\mu}\sum_{C_x\overline{N}_G\in(\overline{K}_G/\overline{N}_G)}\frac{w^2_{C_x\overline{N}_G}}{\varpi^2_{(\overline{K}_G/\overline{N}_G)}}\mathcal{X}_\mu(C_x)\overline{\mathcal{X}_\mu}(C_x)\\ &= w_{\mathcal{X}_\mu}\sum_{C_{xN}\in(\overline{K/N}_{G/N})}\frac{w^2_{C_{xN}}}{\varpi^2_{\overline{K/N}_{G/N}}}\mathcal{X}_\mu(C_{xN})\overline{\mathcal{X}_\mu}(C_{xN}),\end{aligned} \tag{6.38}$$

which is analogous to what we saw in equation (6.37).

With this in hand, we can now prove the following result for *p-qroups* [176], i.e., groups whose order is a power of a prime number p.

Lemma 6.7 (HSHP over *p*-groups). Let G be a *p-group* (where p is prime). Suppose that we are given a hiding function $\overline{f} : \overline{G} \to \{0,1\}^*$. If we can efficiently compute the QFT for $\overline{G}$ and we can efficiently compute the kernels of irreps when restricted to subgroups, then there is an efficient quantum algorithm for the CC-HSHP.

Proof. We follow a similar approach to before, applying AKR to subhypergroups $\overline{K}_G$ (starting with $K = G$) until we measure an irrep ν with kernel smaller than K and then recursing on $\overline{J}_G \leq \overline{K}_G$, where $J = \ker(\nu|_K)$. By assumption, we can compute $\ker(\nu|_K)$ efficiently. If we fail to find such a ν in polynomially many samples, then we can conclude that K is the hidden subgroup with high probability.

By lemma 6.6 and the notes above it, our assumptions imply that we have an oracle and an efficient QFT for each subproblem. To implement AKR, we also need the ability to prepare a uniform superposition over K. This was implemented earlier using the inverse Fourier transform. In this case, that would require us to prepare a complicated coset state in $\widehat{G}$. However, we can instead just prepare the superposition directly by the result of Watrous [270].[42]

Finally, it remains to prove that we have a good probability (e.g., at least 1/2) of measuring a nontrivial irrep or, equivalently, that the probability of measuring the trivial irrep is not too large (e.g., at most 1/2). As before, this amounts to putting a bound on $\sum_{C_{xN}\in(\overline{K/N}_{G/N})}(w_{C_{xN}}/\varpi_{\overline{K/N}_{G/N}})^2$, this time by equation (6.38). By the same argument as before, this will hold if we can show that $w_{C_x\overline{N}_G}/\varpi_{\overline{K/N}}$ is bounded by a constant less than 1. Earlier, we showed this by using the fact that the size of a conjugacy class divides the size of the group. Unfortunately, that does not help us here because we are comparing $w_{C_x\overline{N}_G} = |C_{xN}|$ not to the size of G/N but to the size of the subgroup K/N. These two need not be related by even a constant factor. In extreme cases, K/N may contain only one group element that is not in C_{xN} [271].

Instead, we will use properties of p groups. Since G is a p-group, so is K/N, and the size of K/N must be p^k for some k. Since the size of a conjugacy class divides the size of a group (this time G/N), the size of C_{xN} in K/N must be p^j for some j. Now, we must have $j \leq k$ since

[42] This result holds for black-box groups, so we only need to assume that we know the kernel of each irrep not that we can intersect the kernels of arbitrary irreps.

$C_{xN} \subset \overline{K/N}_{G/N}$. However, we cannot have $j = k$ unless $N = K$, which we have assumed is not the case, since K/N must contain at least two classes (one identity and one non-identity). Hence, we can conclude that $w_{C_{xN}} = |C_{xN}|$ is smaller than $\varpi_{\overline{K/N}_{G/N}} = |K/N|$ by at least a factor of $p \geq 2$. We conclude as before that the probability of measuring the trivial irrep is at most $1/2$. □

Theorem 6.9 (**CC-HSHP Is Easy, III**)**.** Let G be a nilpotent group. Suppose that we are given a hiding function $\overline{f} : \overline{G} \to \{0,1\}^*$. If we can efficiently compute the QFT for G and we can efficiently compute the kernels of irreps when restricted to subgroups, then there is an efficient quantum algorithm for the CC-HSHP.

Proof. A nilpotent group is a direct product of p-groups for different primes [272]. This means that any subgroup must be a direct product of subgroups, one in each of the p-groups.[43] Hence, it suffices to solve the CC-HSHP in each of these p-groups. □

Finally, we can apply this result to the HNSP if we are given an oracle that can be converted into a CC-HSHP oracle.

Corollary 6.4 (**HNSP is easy, III**)**.** Let G be a nilpotent group. Suppose that we are given a hiding function $f : G \to H$ that is a homomorphism. If we can efficiently implement the QFT for G, compute with conjugacy classes of G and H, and compute kernels of irreps when restricted to subgroups, then there is an efficient quantum algorithm for the HNSP.

We note that the class of nilpotent groups includes the Heisenberg group, which, as we saw in Example 6.1, is a case where the original AKR algorithm does not find the hidden normal subgroup with polynomially many samples. Our last algorithm, however, solves the problem efficiently in this case.

Minimal requirements: We highlight that our final quantum algorithms (theorem 6.9, corollary 6.4) work under a minimal amount of assumptions compared to those in last section:

- Our final quantum algorithm for CC-HSHP (theorem 6.9) is provably efficient given **(a)** a *single* circuit to implement the QFT over G, due to lemma 6.6, and **(b)** an HRT kernel-intersection subroutine—requirement (i) in last section). In particular, this algorithm does not need the additional requirements (iii.a-iii.b) of theorem 6.8, hence, runs efficiently under the same assumptions as HRT's.
- Our final quantum algorithm for HNSP (corollary 6.4) is provably efficient given requirement **(ii)** (the HNSP oracle is efficiently convertible into an HNSP oracle); and, again, **(a)** an *single* circuit to implement the QFT over G, due to lemma 6.6, and **(b)** an HRT kernel-intersection subroutine. Requirements (iii.a-iii.b) are no longer needed.

Conclusion about the HNSP: Altogether, the results in this section show that one can solve the HNSP by reducing it to CC-HSHP in many settings under reasonable assumptions, the most significant one being (in the view of the authors) the need to find an oracle conversion protocol (which we handled restricting to group-homomorphism hiding functions). In such scenarios, we show that the fact that the HNSP can be solved efficiently depends crucially on the fact that the hypergroups in the CC-HSHP is abelian. Thus, the results of the last two sections give us

[43] This is, for example, an immediate consequence of Goursat's Lemma [273] (since a quotient of a p-group and a quotient of a q-group, with $q \neq p$, can only be isomorphic if they are both trivial groups).

an explanation for why the HNSP is easy (in a wide range of settings) because of the presence of an *abelian* algebraic structure, the abelian hypergroup of conjugacy classes.

Remarkably, though both HRT's quantum algorithm and ours rely on equivalent assumptions in the classical post-processing step, the quantum parts of the two algorithms are *fundamentally different*. Hence our algorithm demonstrates a new way in which this important problem can be solved efficiently by quantum computers.

Finally, we mention that because of the very special mathematical structure that is common to both the HRT and AKR oracles—see requirement (ii) and section 6.7.2—we are optimistic about the possibility of extending our results beyond the homomorphism-oracle setting[44].

6.7.4 Further results and open problems

We finish this section with some discussion on whether this last result can be extended further. The most natural next step beyond nilpotent groups would be to show that the algorithm works for super-solvable groups. We start with a positive example in that direction.

Example 6.2 (**Dihedral Groups**)**.** As we saw above, the algorithm will work correctly provided that the probability of measuring the trivial irrep is not too close to 1. By equation (6.38), this is given by $\sum_{C_{xN} \in (\overline{K/N}_{G/N})} (w_{C_{xN}} / \varpi_{\overline{K/N}_{G/N}})^2$.

Without loss of generality, we may assume $N = \{e\}$ by instead looking at the group G/N. For a dihedral group, such a quotient is either dihedral or abelian. Since abelian groups are (trivially) nilpotent, we know the algorithm works in that case already.

By our earlier arguments, the probability $\sum_{C \in (\overline{K})_G} (w_C / \varpi_{\overline{K}})^2$ is bounded by a constant below one provided that the fractional weights $w_C / \varpi_{\overline{K}}$ are bounded by a constant below one. In other words, our only worry is that there is a normal subgroup K containing a conjugacy class that is nearly as large as K.

Let us consider the dihedral group of order $2n$, generated by a rotation a of order n and a reflection r of over 2. Most of the normal subgroups are contained in the cyclic subgroup $\langle a \rangle$. These are subgroups of the form $\langle a^d \rangle$ with d dividing n. Every conjugacy class in this subgroup contains either 1 or 2 elements (since $r^{-1} a^j r = a^{-j}$ and hence $r^{-1} a^{-j} r = a^j$). Since a nontrivial normal subgroup cannot consist of one conjugacy class, the worst case would be when K has 3 elements and contains a conjugacy class with 2 elements. In that case, the probability of measuring the trivial irrep could only be as large as $2/3$, which still a constant (independent of n) less than one[a].

If n is odd, then any normal subgroup K containing r is the whole group, and the largest conjugacy class contains every $a^j r$ for $j \in \mathbb{Z}_n$, which is half the elements, so we get a bound of $1/2$ in that case. If n is even, then there are two more normal subgroups, one containing $a^{2j} r$ for each j and one containing $a^{2j+1} r$, but both also contain all rotations of the form a^{2j}, so at least half of the elements in these subgroups are contained in 1–2 element conjugacy classes, and once again we get a bound of $1/2$.

All together, this shows that the probability of measuring the trivial irrep is at most $2/3$ for the dihedral groups, so the algorithm will succeed with high probability.

[a] Note that if $w_{C_x} / \varpi_{\overline{K}_G} \leq c$, then character orthogonality (6.8) lets us bound the probability of measuring a character $\mathcal{X}_\mu \in \mathcal{T}^*$ (6.38) since $\Pr(\mathcal{X}_\mu) / w_{\mathcal{X}_\mu} = \sum_{C_x \in \overline{K}_G} (w_{C_x} / \varpi_{\overline{K}_G})^2 |\mathcal{X}_\mu(C_x)|^2 \leq c(\sum_{C_x \in \overline{K}_G} w_{C_x} / \varpi_{\overline{K}_G} |\mathcal{X}_\mu(C_x)|^2) = c / w_{\mathcal{X}_\mu \overline{K}_G^\perp}$. Specifically, for any invertible character we get $\Pr(\mathcal{X}_\mu) \leq c$.

[44] In fact, similarly to the abelian HSP setting (cf. chapter 5.3.3, theorem 5.5) it can easily be shown that hiding-subgroup promise of the HRT oracle $f : G \to X$ induces a group structure on X—an isomorphism onto G/N—such that $f : G \to X$ is a group homomorphism. A potential approach to extend our results would be to search for a method to exploit this hidden group-homomorphism structure.

On the other hand, we also have a negative example.

Example 6.3 (**Super-solvable group** [271])**.** We will consider the group of simple affine transformations over $\mathbb{Z}_p$. These are transformations of the form $x \mapsto ax + b$ for some $a \in \mathbb{Z}_p^\times$ and $b \in \mathbb{Z}_p$, which we denote by (a, b). These form a group under composition. In particular, applying (a, b) and then (c, d) gives $acx + bc + d$, which shows that $(c, d) \cdot (a, b) = (ac, bc + d)$. A simple calculation shows the formula for the commutator

$$[(a, b), (c, d)] = (1, c^{-1}(1 - a^{-1})d - a^{-1}(1 - c^{-1})b).$$

This implies that the commutator subgroup $[G, G]$ is contained in the set $\{(1, b) \mid b \in \mathbb{Z}_p\}$. On the other hand, taking $a = 1$, $c = 2^{-1}$, and $d = 0$ in this formula gives the result $(1, b)$, so $[G, G]$ must contain all the elements of this set. If we mod out $[G, G]$, then we are left with the abelian group $\mathbb{Z}_p^\times$. We have proven that the group is super-solvable.

On the other hand, for any element $(1, d)$, taking $a = 2^{-1}, c = 1$ and $b = 0$ in the formula above gives the result $(1, -d)$, which is not the identity $(1, 0)$. This means that the group has a trivial center, and thus, it cannot be nilpotent.

Another simple calculation shows that conjugating $(1, b)$ by $(c, 0)$ gives us $(1, c^{-1}b)$. Hence, the conjugacy class of $(1, b)$ with $b \neq 0$ contains every $(1, b')$ with $b' \neq 0$. This is all of the subgroup $[G, G]$ except for the identity element $(1, 0)$. Hence, once we have $K = [G, G]$, we can see by equation (6.38) with $N = \{e\}$ that the algorithm will get the trivial irrep with high probability, so we can see that the algorithm will fail to find N in this case.

Put together, these results show that our last algorithm works for some non-nilpotent, super-solvable groups (like the dihedral groups[45]), but not all super-solvable groups since it fails on the affine linear group. Determining exactly which super-solvable groups the algorithm does succeed on is an open problem.

[45] It is super-solvable since it is a semi-direct product of abelian groups, and it is easy to check that it is not nilpotent unless n is a power of 2.

Appendices

Appendix A

Supplement to chapter 2

A.1 Supplementary material for section 2.2

Proof of lemma 2.6

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, $\beta\circ\alpha(g+h) = \beta(Ag+Ah+\mathrm{zero}_H) = BAg+BAh+\mathrm{zero}_J \pmod J$, 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

$$\chi_\mu(\alpha(g)) = \exp\left(2\pi\mathrm{i}\sum_{ij}\mu^\bullet(i)A(i,j)g(j)\right) = \exp\left(2\pi\mathrm{i}(A^T\mu^\bullet)\cdot g\right) \text{ for every } g\in G. \tag{A.1}$$

Let f be the function $f(g) := \exp(2\pi\mathrm{i}(A^T\mu^\bullet)\cdot g)$. Then it follows from (A.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,

$$\alpha^*(\mu) = \Upsilon_G^{-1}(A^T\mu^\bullet) \pmod{G^*} = \Upsilon_G^{-1}A^T\Upsilon_H\mu \pmod{G^*}. \tag{A.2}$$

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 2.7

We will show that each of the homomorphisms α_{XY} as considered in lemma 2.5 has a matrix representation, say A_{XY}. Then it will follow from (2.24) in lemma 2.5 that

$$A := \begin{pmatrix} A_{\mathbb{Z}\mathbb{Z}} & 0 & 0 & 0 \\ A_{\mathbb{R}\mathbb{Z}} & A_{\mathbb{R}\mathbb{R}} & 0 & 0 \\ A_{F\mathbb{Z}} & 0 & A_{FF} & 0 \\ A_{\mathbb{T}\mathbb{Z}} & A_{\mathbb{T}\mathbb{R}} & A_{\mathbb{T}F} & A_{\mathbb{T}\mathbb{T}}. \end{pmatrix}, \tag{A.3}$$

as in (2.31), 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 2.2. Thus, all homomorphisms α_{XY} with Y of the types $\mathbb{Z}^a$ or F have matrix representations; by duality and lemma 2.6(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

$$\alpha_{\mathbb{RR}}(rx) = r\alpha_{\mathbb{RR}}(x) \tag{A.4}$$

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 (A.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 (A.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 lemma 2.8

It suffices to show (a), that any matrix representation A of α must be an element of Rep and fulfill the consistency conditions (2.30); (b), that these consistency conditions imply that A is of the form (2.31) 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 2.2 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 (2.30).

Next, we will now prove (b).

First, the block form (2.31) almost follows from (2.24) in lemma 2.5: we only have to show, in addition, that the zero matrix is the only valid matrix representation for any trivial group homomorphism $\alpha_{XY} = \boldsymbol{0}$ in (2.24). 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.

Second, we prove propositions 1-4. In proposition 1, $A_{\mathbb{ZZ}}$ must be integral 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 2.6(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 $A_{XY}(e_i) \pmod{Y}$ 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 (2.32), due to basic properties of linear congruences (see e.g. lemma 11 in [63] for a similar derivation.) 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 $A_{\mathbb{T}F}$ has rational coefficients of 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 (2.30) is fulfilled. We prove the remaining cases in a

single step, by letting $G' = G_F \times G_{\mathbb{T}}$, $H' = H_F \times H_{\mathbb{T}}$ and $A' = \begin{pmatrix} A_{FF} & 0 \\ A_{\mathbb{T}F} & A_{\mathbb{T}\mathbb{T}} \end{pmatrix}$ and showing that $A' : G' \to H'$ is a homomorphism given (2.30). To this end, we let a'_i denote the ith column of A', m' be the total number of columns; with this notation, we evaluate the action of A' on $g, h, g+h \in G'$ *without* taking remainders to be

$$A'g + A'h = \sum [g(i) + h(i)]a'_i, \qquad A'(g+h) = \sum (g+h)(i)a'_i. \tag{A.5}$$

Recalling associativity of H and G, the latter expression shows that $A'h$ defines a function from G' to $\mathbb{Z}^{m'}$, and, thus, $A'h \pmod{H'}$ is a function from G' to H'. Last, it holds for every i that $g(i)+h(i) = q_i c_i + (g+h)(i)$ for some integers q_i , since (by definition of the group G) $(g+h)(i)$ is the remainder obtained when $g(i) + h(i)$ is divided by c_i (q_i is the quotient). It follows, subtracting modularly, that $A'(g) + A'(h) - A'(g+h) = \sum_i q_i c_i a'_i = 0 \pmod{H'}$ for every g, h, using (2.30); it follows that A' (hence A_{FF}, $A_{\mathbb{T}F}$, $A_{\mathbb{T}\mathbb{T}}$) are group homomorphisms. □

A.2 Existence of general-solutions of systems of the form (2.43)

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 A.1 (See theorem 21.19 in [189] or section 7.3.3 in [190])**.** The class of elementary abelian groups[a] is closed with respect to forming closed subgroups, quotients by these, and finite products.[b]

[a] Beware that in [189] the class of elementary groups is referred as "the category CGAL", which stands for Compactly Generated Abelian Lie groups.

[b] In fact, as mentioned in [189], corollary 21.20 elementary LCA groups constitute the smallest subclass of LCA containing $\mathbb{R}$ and fulfilling all these properties.

In our setting, the kernel of a continuous group homomorphism $A : G \to H$ as in (2.43) is always closed: this follows from the fact that the singleton $\{0\} \subset H$ is closed (because elementary abelian groups are Hausdorff [189]), which implies that $\ker A = A^{-1}(\{0\})$ is closed (due to continuity of A). Hence, it follows from lemma A.1 that $\ker A$ is topologically isomorphic to some elementary abelian group $H' := \mathbb{R}^a \times \mathbb{T}^b \times \mathbb{Z}^c \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_c}$; 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 [189]. 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 (2.43) constitutes a general solution of (2.43), proving the statement.

A.3 Proof of theorem 2.2

In this appendix, we prove theorem giving efficient classical algorithms for tasks (1-4).

A.3.1 Algorithms for tasks (1-2)

We show how to decide the existence of and find general solutions of the system $Ax = b \pmod{H}$. Our first step is to show that systems of linear equations over groups (2.43) can be reduced to systems of mixed real-integer linear equations (2.47). This is proven next.

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* 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 $c_1e_1, \ldots, c_me_m$. Observe that the group G—the solution space in system (2.43)—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 $\mathbf{q} : \mathbf{X} \to G = \mathbf{X}/\mathbf{V}$:

$$\mathbf{q}(\mathbf{x}) := (\mathbf{x}(1) \bmod c_1, \ldots, \mathbf{x}(m) \bmod c_m) = \mathbf{x} \pmod{G}; \tag{A.6}$$

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 2.6.(a), since the $m \times m$ identity matrix I_m is a matrix representation of $\mathbf{q}$.) We now consider the relaxed[1] system of equations

$$\alpha(\mathbf{x}) = b \pmod{H}, \quad \text{where } \mathbf{x} \in \mathbf{X} = \mathbb{Z}^a \times \mathbb{R}^b. \tag{A.7}$$

Note that the problem of solving (2.43) reduces to solving (A.7), 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 (A.7); then[2]

$$G_{\text{sol}} = \mathbf{q}(\mathbf{X}_{\text{sol}}) \quad \Longrightarrow \quad G_{\text{sol}} = \mathbf{q}(\mathbf{x}_0) + \mathbf{q}(\ker \alpha) = \mathbf{x}_0 + \ker \alpha \pmod{G}, \tag{A.8}$$

Hence, our original system (2.43) admits solutions iff (A.7) also does, and the former can be obtained from the latter via the homomorphism $\mathbf{q}$. We further show next that (A.7) is equivalent to a system of form (2.47). 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 $(\bmod H)$ is equivalent to the modular constraints $\bmod d_1, \ldots, \bmod d_n$, 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

$$A_{\mathbb{Z}}\mathbf{x}_{\mathbb{Z}} + A_{\mathbb{R}}\mathbf{x}_{\mathbb{R}} + D\mathbf{y} = c, \quad \text{where } D = \operatorname{diag}(d_1, \ldots, d_n),\ \mathbf{y} \in \mathbb{Z}^n. \tag{A.9}$$

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 (A.9) is a system of mixed-integer linear equations as in (2.47). Also, system (2.47) 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 (A.9) can be rewritten as

$$\mathbf{Ag} := \begin{pmatrix} A & D \end{pmatrix} \mathbf{g} = c, \quad \text{where } \mathbf{g} \in \mathbf{G} \tag{A.10}$$

and c represents an element of $\mathbf{Y}$.

Mind that (A.9) (or equivalently (A.10)) admits solutions if and only if both of (A.7) and (2.43) admit solutions. Indeed, the solutions of (A.9) and (A.7) are—again—related 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

[1] Notice that the new system is less constrained, as we look for solutions in a larger space than beforehand.

[2] 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 $G_{\text{sol}} = \mathbf{q}(\mathbf{q}^{-1}(G_{\text{sol}})) \subset \mathbf{q}(\mathbf{X}_{\text{sol}})$.

(A.9) that $\pi(\mathbf{G}_{\text{sol}}) = \mathbf{X}_{\text{sol}}$ and, consequently, $\mathbf{q} \circ \pi(\mathbf{G}_{\text{sol}}) = G_{\text{sol}}$; these relationships show that either all systems admit solutions or none of them do.

In the second step of the proof, we use existing algorithms to find a general solution $(\mathbf{g}_0, \mathbf{P})$ of system (A.10) and show how to use this information to compute a general solution of our original problem (2.43).

First, we recall that algorithms presented in [199] can be used to: (a) *check* whether a system of the form (2.47,A.10) admits a solution; (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[3] of $\mathbf{A} = \begin{pmatrix} A & D \end{pmatrix}$ (for details see theorem 1 in [199]).

Assume now that (A.9) 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 (2.43) can be computed by making use of the map $\mathbf{q} \circ \pi$. We also discuss the overall worst-case 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 (2.43) (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 (A.9) and the worst-time scaling of the algorithms in [199] is also polynomial in the mentioned variables. (We discussed the complexity of these methods in section 2.4.2.)

- Second, a particular solution x_0 of (2.43) 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

$$\text{if} \quad \mathbf{P} = \begin{pmatrix} \mathbf{P}_{\mathbf{XX}} & \mathbf{P}_{\mathbf{X}\mathbb{Z}} \\ \mathbf{P}_{\mathbb{Z}\mathbf{X}} & \mathbf{P}_{\mathbb{Z}\mathbb{Z}} \end{pmatrix} \quad \text{then} \quad P := \begin{pmatrix} \mathbf{P}_{\mathbf{XX}} & \mathbf{P}_{\mathbf{X}\mathbb{Z}} \end{pmatrix} \tag{A.11}$$

  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 (2.43), with worst-time scaling as a polynomial in the variables m, n, $\|A\|_{\mathbf{b}}$, $\|b\|_{\mathbf{b}}$, $\log c_i$, $\log d_j$. This proves theorem 2.2.

A.3.2 Algorithm for problem (3-4)

First, we show that problem 4 can be solved by via the algorithm for problems 1-2-3. First, we can use algorithms 1-2 to decide if a general solution exists and (in the affirmative case) find $(x_0, \mathcal{E})$ such that $G_{\text{sol}} = x_0 + \text{im}\mathcal{E}$. Moreover, we can run our algorithm 3 to find (a) an elementary group Q isomorphic to im$\mathcal{E}$, which must necessarily be finite and of form $Q = \mathbb{Z}_{D_1} \times \ldots \times \mathbb{Z}_{D_m}$, so that the total number of solutions is $|G_{\text{sol}}| = |\text{im}\mathcal{E}| = |Q| = D_1 \ldots D_m$, which is efficiently

[3] 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).

computable; and (b) a matrix representation of the isomorphism $\mathcal{E}_{\text{iso}} : Q \to \text{im}\mathcal{E}$, the columns of which form a generating set of $\text{im}\mathcal{E}$; hence, we can simply set the sought elements $x_1, \ldots, x_r$ to be the columns of $\mathcal{E}_{\text{iso}}$.

The above reduction shows that we can finish our proof if we give an efficient classical algorithm for problem 4. We address this question next.

Recall that, in problem 4, we are given $G = \mathbb{Z}^a \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_b}$, $\mathcal{X} = \mathbb{Z}^{\alpha+\beta}$ and $(x_0, \mathcal{E})$ as an input. Our task is to devise an algorithm to compute a primitive-group decomposition of the quotient $Q = \mathcal{X}/\ker\mathcal{E}$ and a matrix representation of the isomorphism $\mathcal{E}_{\text{iso}} : Q \to \text{im}\mathcal{E}$. To this end, we first apply our algorithm in theorem 2.2 to obtain a $(\alpha+\beta) \times \gamma$ matrix representation A of a group homomorphism[4] $A : \mathbb{Z}^\gamma \to \mathcal{X}$ such that $\text{im}\, A = \ker\mathcal{E}$ (where $\gamma = \alpha + \beta + m$). We can represent these maps in a diagram:

$$\mathbb{Z}^\gamma \xrightarrow{A} \mathbb{Z}^{\alpha+\beta} \xrightarrow{\mathcal{E}} \text{im}\mathcal{E} \tag{A.12}$$

The worst-case time complexity needed to compute A is polynomial in the variables m, α, β, $\log N_i$, $\|\mathcal{E}\|_{\mathbf{b}}$. Next, we compute two integer invertible matrices U, V such that $A = USV$ and S is in Smith normal form (SNF). This can be done in $O\left(\text{poly}\left(m, \alpha, \beta, \log N_i, \|\mathcal{E}\|_{\mathbf{b}}), \log\frac{1}{\varepsilon}\right)\right)$ time with existing algorithms to compute the SNF of an integer matrix (see e.g. [201] for a review). Each matrix V, S, U is the matrix of representation of some new group homomorphism, as illustrated in the following diagram.

$$\begin{array}{ccccc} \mathbb{Z}^\gamma & \xrightarrow{A} & \mathbb{Z}^{\alpha+\beta} & \xrightarrow{\mathcal{E}} & \text{im}\mathcal{E} \\ \downarrow{\scriptstyle V} & & \uparrow{\scriptstyle U} & & \\ \mathbb{Z}^\gamma & \xrightarrow{S} & \mathbb{Z}^{\alpha+\beta} & & \end{array} \tag{A.13}$$

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, $\text{im}\, S = \text{im}\, U^{-1}AV^{-1} = \text{im}\, U^{-1}A = U^{-1}(\text{im}\, A) = U^{-1}(\ker\mathcal{E})$, which shows that $\ker\mathcal{E}$ is isomorphic to $\text{im}\, S$ via the isomorphism U^{-1}. These facts together with lemma 2.6.(a) show that $\mathcal{E}_{\text{iso}} := \mathcal{E}U$ is a matrix representation of a *group isomorphism* from the group $Q := \mathcal{X}/\text{im}\, S$ into $\text{im}\mathcal{E}$.

Finally, we show that Q can be written explicitly as a direct product of primitive groups of type $\mathbb{Z}$ and $\mathbb{Z}_d$. We make crucial use of the fact that S is Smith normal form, i.e.

$$S = \left(\begin{array}{cccc|c} s_1 & & & & \\ & s_2 & & & \\ & & \ddots & & 0 \\ & & & s_{(\alpha+\beta)} & \end{array}\right) = \left(\begin{array}{cccc|c} I_{\mathbf{a}} & & & & \\ & \sigma_1 & & & \\ & & \ddots & & 0 \\ & & & \sigma_{\mathbf{b}} & \\ & & & & 0 \end{array}\right), \tag{A.14}$$

where the coefficients σ_i are strictly positive. It follows readily that $\text{im}\, S = \mathbb{Z}^{\mathbf{a}} \times \sigma_1\mathbb{Z} \times \cdots \sigma_{\mathbf{b}}\mathbb{Z} \times \{0\}^{\mathbf{c}}$, and therefore

$$Q = \mathbb{Z}^{a+b}/\text{im}\, S = \{0\}^{\mathbf{a}} \times \mathbb{Z}_{\sigma_1} \times \ldots \times \mathbb{Z}_{\sigma_{\mathbf{b}}} \times \mathbb{Z}^{\mathbf{c}}. \tag{A.15}$$

A.4 Efficiency of Bowman-Burdet's algorithm

In this appendix we briefly discuss the time performance of Bowman-Burdet's algorithm [199] and argue that, using current algorithms to compute certain matrix normal forms (namely,

[4] Lemma 2.8 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 $\mathbb{Z}$.

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 (2.47), 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 [199], section 3) involves different types of steps, of which the most time consuming are (see equations 8-10 in [199]):

1. the calculation of a constant number of certain types of generalized inverses introduced by Hurt and Waid [200];
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 [200], equations 2.3-2.4. The worst-case 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. [201] for a slightly outdated review). Furthermore, S, U and V can be computed in polynomial time (we refer the reader to [201] again).

The analysis above shows that Bowman-Burdet's algorithm runs in worst-time polynomial in the variables $\|A\|_{\mathbf{b}}$, $\|B\|_{\mathbf{b}}$, $\|C\|_{\mathbf{b}}$, a, b, c, which is enough for our purposes.

A.5 Proof of lemma 2.14

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,

$$(\alpha + \beta)(g) = \alpha(g) + \beta(g) = Ag + Bg = (A + B)g \pmod{G}, \tag{A.16}$$

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 lemma 2.8 is that the group X_j is always an elementary abelian group, namely

$$\begin{aligned}
G_j = \mathbb{Z} \quad &\Rightarrow \quad X_j = G = \mathbb{Z}^a \times \mathbb{R}^b \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_c} \times \mathbb{T}^d; \\
G_j = \mathbb{R} \quad &\Rightarrow \quad X_j = \{0\}^a \times \mathbb{R}^b \times \{0\}^c \times \mathbb{R}^d; \\
G_j = \mathbb{Z}_{N_j} \quad &\Rightarrow \quad X_j = \{0\}^a \times \{0\}^b \times (\eta_{1,j}\mathbb{Z} \times \cdots \times \eta_{c,j}\mathbb{Z}) \times \left(\tfrac{1}{N_j}\mathbb{Z}\right)^d; \\
G_j = \mathbb{T} \quad &\Rightarrow \quad X_j = \{0\}^{m_z} \times \{0\}^{m_r} \times \{0\}^{m_f} \times \mathbb{Z}^{m_t};
\end{aligned} \tag{A.17}$$

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

$$\alpha \circ \beta = \mathrm{id} \quad \Longrightarrow \quad \beta = (\alpha^{-1} \circ \alpha) \circ \beta = \alpha^{-1} \circ (\alpha \circ \beta) = \alpha^{-1}. \tag{A.18}$$

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

$$AX\left(\sum_j g(j)e_j\right) = \sum_j g(j)Ax_j = \sum_j g(j)e_j \pmod G, \text{ for every } g \in G, \tag{A.19}$$

where x_j denotes the jth column of X. Since (A.19) holds, in particular, when all but one number $g(j)$ are zero, it can be re-expressed as an equivalent system of equations:

$$g(j)Ax_j = g(j)e_j \pmod G, \text{ for any } g(j) \in G_j, \text{ for } j = 1, \ldots, m. \tag{A.20}$$

Finally, we will reduce each individual equation in (A.20) to a linear system of equations of the form (2.43). This will let us apply the algorithm in theorem 2.2 to compute every individual column x_j of X.

We begin by finding some simpler equivalent form for (A.20) 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 (A.20) holds iff $Ax_j = e_j \pmod G$.

(b) If $G_j = \mathbb{R}$ or $G_j = \mathbb{T}$ we show that (A.20) is equivalent to $Ax_j = e_j \pmod{X_j}$. Clearly, (A.20) implies $g(j)Ax_j = g(j)e_j + zero$ where $zero = 0 \pmod G$ 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 $\frac{g(j)}{d}Ax_j = \frac{g(j)}{d}e_j + zero'$. These two equations combined show that $zero = d\, zero'$ 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 $zero = 0$ and, consequently, (A.20) 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 (A.17).

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 (2.43). First notice that for any two homomorphisms β, β' with matrix representations X, Y, it follows from (A.16) and lemma 2.6.(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,

$$A(X + Y)g = (AX + AY)g \pmod G, \text{for every } g \in G. \tag{A.21}$$

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 (A.21). 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 (A.21) 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 $A(x_j + y_j) = Ax_j + Ay_j \pmod{X_j}$ 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 (2.43).

A.6 Supplementary material for section 2.3

Proof of lemma 2.9

The lemma is a particular case of proposition 1.1 in [274]. We reproduce a shortened proof of the result in [274] (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 $f_g : h \to B(g,h)$ 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 $g \to B(g,h)$ 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 $\beta : G \to G^*$ is a group homomorphism. It remains to show that β is continuous; for this we refer to the proof in [274], where the author analyzes how neighborhoods are transformed under this map.

Proof of lemma 2.10

We obtain (a) by combining (2.28) with the normal form (2.35): 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 2.9, property (a) and lemma 2.3.

To prove (d) we bring together (a) and the relationship $B(h,g) = B(g,h)$, and derive

$$g^T M h = g^T M^T h \mod \mathbb{Z}, \quad \text{for every } g,\, h \in G. \tag{A.22}$$

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 (A.22) 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

$$st(M(i,j) - M(j,i)) \in \mathbb{Z} \tag{A.23}$$

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 (A.22) it follows that

$$M(j,i)g(i)h(j) = M(i,j)g(i)h(j) + \Delta_{i,j}g(i)h(j) \mod \mathbb{Z}, \tag{A.24}$$

As $g(i)$ and $h(j)$ are integers the factor $\Delta_{i,j}g(i)h(j)$ gets canceled 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 (2.20), 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 2.11

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 $\chi_\mu = f$:

$$f(g+h) := \frac{\xi_1(g)}{\xi_2(g)} \frac{\xi_1(h)}{\xi_2(h)} \frac{B(g,h)}{B(g,h)} = f(g)f(h). \tag{A.25}$$

Proof of lemma 2.12

Define the function $q : G \to \mathbb{R}$ as

$$q(g) := g^T M g + C^T g. \tag{A.26}$$

We prove that $q(g)$ is a quadratic form modulo $2\mathbb{Z}$ with associated bilinear form $b_q(g,h) := 2g^T M h$; or, in other words, that the following equality holds for every $g, h \in G$:

$$q(g+h) = q(g) + q(h) + 2g^T M h \pmod{2\mathbb{Z}}. \tag{A.27}$$

Assuming that (A.27) 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:

$$Q(g+h) = Q(g)Q(h)\exp(2\pi i\, g^T M h) \tag{A.28}$$

We prove (A.27) by direct evaluation of the statement. First we define $q_M(g) := g^T M g$ and $q_C(g) := C^T g$, 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 $G = G_1 \times \ldots G_m$ with G_i primitive, we define $c := (c_1, \ldots, c_m)$ 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: $\lambda \circ c = (\lambda_1 c_1, \ldots, \lambda_m c_m)$. 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 $M = M^T$):

$$\begin{aligned} q_M(g \oplus h) &= q_M(g) + q_M(h) + 2g^T M h \\ &\quad + 2g^T M(\lambda \circ c) + 2h^T M(\lambda \circ c) + (\lambda \circ c)^T M(\lambda \circ c), \end{aligned} \tag{A.29}$$

$$q_C(g \oplus h) = q_C(g) + q_C(h) + \sum_i M(i,i)\lambda(i)c_i^2. \tag{A.30}$$

Consider an $x \in \mathbb{R}^m$ for which there exists $g \in G$ such that $x \equiv g \bmod G$. Then $x^T M(\lambda \circ c)$ with $x \in G$ must be an integer. Indeed, we have

$$1 = B(g,0) = \exp(2\pi i x^T M(\lambda \circ c)), \tag{A.31}$$

where in the second identity we used lemma 2.10 together with the property $\lambda \circ c \equiv 0 \bmod G$. This shows that $x^T M(\lambda \circ c)$ is an integer. It follows that the fourth and fifth terms on the right hand side of eq. (A.29) must be equal to an even integer and thus cancel modulo $2\mathbb{Z}$. Combining results we end up with the expression

$$q(g \oplus h) = q(g) + q(h) + 2g^T M h + \Delta \pmod{2\mathbb{Z}}, \tag{A.32}$$

where

$$\Delta := (\lambda \circ c)^T M(\lambda \circ c) + \sum_i M(i,i)\lambda(i)c_i^2. \tag{A.33}$$

We finish our proof by showing that Δ is an even integer too, which proves (A.27).

First, we note that, due to the symmetry of M, we can expand $(\lambda \circ c)^T M(\lambda \circ c)$ as

$$(\lambda \circ c)^T M(\lambda \circ c) = \sum_{i,j\,:\,i<j} 2M(i,j)\lambda(i)\lambda(j)c_i c_j + \sum_i M(i,i)\lambda(i)^2 c_i^2. \tag{A.34}$$

Revisiting (A.31) and choosing $x = e_i$ and $\lambda = e_j$ for all different values of i, j, we obtain the following consistency equation for M

$$c_j M(i,j) = c_i M(i,j) = 0 \pmod{\mathbb{Z}} \tag{A.35}$$

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 (A.34) by taking modulo $2\mathbb{Z}$, yielding

$$\Delta = \sum_i M(i,i)\lambda(i)^2 c_i^2 + \sum_i M(i,i)\lambda(i)c_i^2 \pmod{2\mathbb{Z}} \tag{A.36}$$

$$= \sum_i M(i,i)c_i^2 \lambda(i)(\lambda(i)+1) = 0 \pmod{2\mathbb{Z}}, \tag{A.37}$$

where in the last equality we used the fact that $\lambda(i)(\lambda(i)+1)$ is necessarily even.

Proof of lemma 2.13

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 2.1 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 2.6(a) and lemma 2.10) 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 2.11. By construction, both $\xi_{M,v} \circ \alpha$ and $Q_{M'}$ are $B_{M'}$-representations of the bicharacter $B_{M'}$. As a result, lemma 2.11 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:

$$\chi_{v'}(g) = \exp\left(\pi i\,(A^T C_M - C_{A^T M A})\,g\right) \exp\left(2\pi i\,(A^T v)\cdot g\right). \tag{A.38}$$

It can be checked that the function $\exp\left(2\pi i\,(A^T v)\,g\right)$ is a character, using that it is the composition of a character $\exp(2\pi v^T g)$ (theorem 2.1) and a continuous group homomorphism α. Since $\chi_{v'}$ is also a character, the function $\exp\left(\pi i\,(A^T C_M - C_{A^T M A})\,g\right)$ 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 $G^\bullet$ [5]; 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 $\xi_{M,v} \circ \alpha(g) = Q_{M'}(g) f(g) = Q_{M'}(g)\chi_{v'}(g) = \xi_{M',v'}(g)$.

[5] This statement can also be proven (more laboriously) by explicit evaluation, using arguments similar to those in the proof of lemma 2.12.

Appendix B

Supplement to chapter 3

B.1 Proof of lemma 3.2

In the following, we define A_H, A_K to be integer matrices whose columns are the elements of the sets $\{h_1, \ldots, h_r\}$ and $\{k_1, \ldots, k_s\}$; the latter generate respectively H and K. Also, we denote by d the least common multiplier of $d_1, \ldots, d_m$. One can use lemma 2.8 to check that the matrices A_H, A_K and $[A_H|A_K]$ define group homomorphisms from $\mathbb{Z}_d^t$ to G, if the value of t is respectively chosen to be r, s and $r+s$.

We will show how to turn the problems (a-c) into system of the form $Ax = b \pmod{\mathbf{G}}$ such that $\mathbf{G}$ equals the original group G; $\mathbf{G}_{sol}$ is chosen to be $\mathbb{Z}_d^t$, for some t; and A is an integer matrix that defines a group homomorphism from $\mathbf{G}$ to $\mathbf{G}_{sol}$:

(a) b belongs to H if and only if b can be obtained as a linear combination of elements of H, i.e., if and only if $A_H x = b \pmod{G}$ has at least one solution $x \in \mathbb{Z}_d^r$. Moreover, if one finds a particular solution w, this element fulfills $b = A_H w = \sum w(i) h_i \pmod{G}$.

(b) The order of H is the number of distinct linear combinations of columns of A_H, which coincides with the order of the *image* of the group homomorphism $A_H : \mathbb{Z}_d^r \to G$. With this knowledge, it suffices to count the number of solutions of $A_H x = 0 \pmod{G}$, which equals $|\ker A_H|$. Then, one can compute $|H| = |\mathrm{im} A_H| = d^r / |\ker A_H|$, where the latter identity comes from the first isomorphism theorem ($\mathrm{im} A_H \cong \mathbb{Z}_d^r / \ker A_H$).

(c) g belongs to $H \cap K$ iff it can be simultaneously written as $h = \sum x(i) h_i = \sum y(i) k_i$ for some $(x, y) \in \mathbb{Z}_d^r \times \mathbb{Z}_d^s$; or, equivalently, iff there exist an element (x, y) of the kernel of $[A_H|A_K] : \mathbb{Z}_d^r \times \mathbb{Z}_d^s \to G$ such that $h = A_H x = -A_K y \pmod{G}$. Thus, given a generating-set $\{(x_i, y_i)\}$ of $\ker [A_H|A_K]$, the elements $g_i := A_H x_i \pmod{G}$ generate $H \cap K$, and, owing to, the problem reduces to finding solutions of $[A_H|A_K] \binom{x}{y} = 0 \pmod{G}$.

(d-e) Note that problem (d) reduces to (e) by setting all a_i to be 0—this yields the system (2.17), whose solutions are the elements of the annihilator subgroup. Therefore, it will be enough to prove the (e)th case. Moreover, since the equations $\chi_{h_i}(g) = \gamma^{a_i}$ can be fulfilled for some $g \in G$ only if all γ^{a_i} are $|G|$th-roots of the unit, this systems can only have solutions if all a_i are even numbers. As we can determine it efficiently whether these numbers are even, we assume from now on that it is the case.

Now define a tuple of integers b coefficient-wise as $b(i) := a_i/2$; use the later to re-write $\gamma^{a_i} = \exp(2\pi \mathrm{i}\, b(i)/|G|)$. Also, denote by H the group generated by the elements h_i. By letting $|G|$ multiply numerators and denominators of all fractions in (1.6), the system of complex exponentials

$\chi_{h_i}(g) = \gamma^{a_i}$ can be turned into an equivalent system of congruences $\sum_j (|G|/d_j)\, h_i(j) g(j) = b(i) \mod |G|$. Finally, by defining a matrix Ω with coefficients $\Omega(i,j) := (|G|/d_j)\, h_i(j)$ the system can be written as

$$\Omega g = b \pmod{\mathbf{G}}, \tag{B.1}$$

where b belongs to $\mathbf{G} = \mathbb{Z}^r_{|G|}$, being r the number of generators h_i, and we look for solutions inside $\mathbf{G}_{sol} = G$. Moreover, the coefficients of Ω fulfill $d_j \Omega(i,j) = 0 \mod |G|$; hence, condition (2.30) is met and Ω defines a homomorphism.

(h) Note that the group homomorphism $\omega(g) := \Omega(g) \pmod{\mathbb{Z}^r_{|G|}}$ fulfills $\ker \omega = H^\perp$. Therefore, if we substitute H with $H' := H^\perp$ in the procedure above, given $s = \operatorname{polylog}|G|$ generators of H', we would obtain an $s \times m$ integer matrix Ω' that defines a second group homomorphism $\varpi : G \to \mathbb{Z}^s_{|G|}$ such that

$$\varpi(g) = \Omega'(g) \pmod{G} \qquad \text{and} \qquad \ker \varpi = H'^\perp = H. \tag{B.2}$$

As a result, our classical algorithm for problem (d) can be efficiently adapted to solve (h).

(Remarks:) Finally, note that Ω can be computed in $O(\operatorname{polylog}|G|)$ using standard algorithms to multiply and divide integers (chapter 2.4). It is now routine to check, using the concepts developed thus far, that both $\log|\mathbf{G}_{sol}|$ and $\log|\mathbf{G}|$ are $O(\operatorname{polylog}|G|)$; as a result, the input-size of the new problem, as well as the memory needed to store Ω and b, are all $O(\operatorname{polylog}|G|)$.

Finally, notice that r, s and $r+s$ are $O(\operatorname{polylog}|G|)$ due to the initial assumption that the generating-sets are poly-size, and that d is $O(d_1 d_2 \cdots d_m) = O(|\mathbf{G}|)$; as a consequence, $\log|\mathbf{G}_{sol}|$, $\log|\mathbf{G}|$ are also $O(\operatorname{polylog}|G|)$; and, thus, we need $O(\operatorname{polylog}|G|)$ memory to store the matrix A. It follows that the input-size of the new problem is $O(\operatorname{polylog}|G|)$ and, therefore, we have reduced all problems (a-c) to systems of linear equations over finite abelian groups in polynomial time.

Appendix C

Supplement to chapter 5

C.1 Proof of theorem 5.4

To prove the result we can assume that we know a group isomorphism $\varphi : \mathbf{B} \to G$ that decomposes the black-box group as a product of cyclic factors $G = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_d}$. 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 $b \in \mathbf{B}$. 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 $\mathbb{Z}_M \times \mathbf{B}$: automorphism (resp. quadratic phase) gates get mapped to automorphism (resp. quadratic phase) gates and vice-versa; isomorphic groups have isomorphic character groups [188], 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 $\mathbb{Z}_M \times G$. The advantage now is that we can use results from chapter 3, [134]. In fact, the rest of the proof will be similar to the proof of theorem 2 in [134].

The action of U_{me} in the group-element basis reads $U_{\text{me}}|m, g\rangle = |m, m\alpha + g\rangle$, in additive notation. Define a function $F(m, g) = (m, m\alpha + g)$. We now assume that the order M of 1 as an element of $\mathbb{Z}_M$ is not divisible by $|a|$ and that there exists a normalizer circuit $\mathcal{C}$ such that $\|\mathcal{C} - U_{\text{me}}\| < \delta$ with $\delta = 1 - 1/\sqrt{2}$ and try to arrive to a contradiction. This property implies that $\|\mathcal{C}|m, g\rangle - U_{\text{me}}|m, g\rangle\| < \delta$ for any standard basis state, and consequently

$$|\langle F(m, g)|\mathcal{C}|m, g\rangle| > 1 - \delta = \tfrac{1}{\sqrt{2}} \tag{C.1}$$

We now from, e.g., theorem 3.4 that $\mathcal{C}|m, g\rangle$ is a uniform superposition over some subset $x + K$ of $\mathbb{Z}_M \times G$, being K a subgroup. If K 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 $\langle m', g'|\mathcal{C}|m, g\rangle \leq \frac{1}{\sqrt{2}}$ for any basis state $|m', g'\rangle$ in contradiction with (C.1), so that we can assume $K = \{0\}$ and that $\mathcal{C}|m, g\rangle$ is a standard basis state. Then (C.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 [134], 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 if $F(m, g)$ is affine then M need to be 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 check that $b = 0$, so that $F(m, g)$ must be an automorphism. Because of M is the order of $(1, 0)$ in $\mathbb{Z}_M \times G$, it follows that $(0, 0) = F((0, 0)) = F(M(1, 0)) = MF((1, 0)) = M(1, \alpha)$ modulo $\mathbb{Z}_M \times G$, which holds only if M is a multiple of the order of α.

C.2 Quantum algorithm for discrete logarithms over elliptic curves

In this appendix, we show that a quantum algorithm given by Proos and Zalka [94] to compute *discrete logarithms over elliptic curves* can be implemented with black-box normalizer circuits. This generalizes our result from section 5.3.1, where we saw that black-box normalizer circuits can compute discrete logarithm in $\mathbb{Z}_p^\times$ and break the Diffie-Hellman key exchange protocol: specifically, we showed that Shor's algorithm for this problem decomposes naturally in terms of normalizer gates over $\mathbb{Z}_{p-1}^2 \times \mathbb{Z}_p^\times$, where $\mathbb{Z}_p^\times$ is treated as a black-box group. Unlike the previous setting, our implementation of Proos and Zalka algorithm requires either normalizer gates over an *infinite* group $\mathbb{Z}^2 \times E$ (similarly to Shor's factoring algorithm, section 5.3.2) or an order-finding oracle.

Basic notions

To begin with, 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 [94], over fields with characteristic different than 2 and 3. Our discussion applies equally to the (more general) cases considered in [95, 96], although the definition of the elliptic curve group operation becomes more cumbersome in such settings[1]. For more details on the subject, in general, we refer the reader to [7, 275].

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

$$C : y^2 = x^3 + \alpha x + \beta \tag{C.2}$$

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 assumed to be 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 $Q = (x_Q, y_Q) \in E$, the element $P + Q$ is defined via the following rule:

$$P + Q = \begin{cases} O & \text{if } P = (x_P, y_P) = (x_Q, -y_Q) = -Q, \\ -R & \text{otherwise (read below).} \end{cases} \tag{C.3}$$

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) [7].

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

[1] Correspondingly, the complexity of performing group multiplications in [95, 96] is greater.

$xa = b$ 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 [7].

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 [94]; 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 5.3.1.

Our **main contribution** in this appendix is a proof 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 5.3.1 and 5.3.2 and will be sketched in less detail.

Algorithm C.1 (Proos-Zalka's [94]).

Input. An elliptic curve with associated group E (the group operation is defined as per (C.3)), 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 (C.2), 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. Fourier transforms are applied to the ancillas to create the superposition $\sum_{(x,y)\in A} |x, y, O\rangle$.

3. The following transformation is applied unitarily:

$$\sum_{(x,y)\in A} |x, y, O\rangle \quad \xrightarrow{c\text{-}U} \quad \sum_{(x,y)\in A} |x, y, xa + yb\rangle. \tag{C.4}$$

4. Fourier transforms are applied again over the ancillas and then measured, 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 C.1 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 5.3.2). In view of our results in sections 5.3.1 and 5.3.2, one can easily come up with two approaches to implement algorithm 5.1 using normalizer gates.

(a) The first approach would be to use our normalizer version of Shor's algorithm (theorem 5.2) 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 C.1 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 5.2. The algorithm becomes a normalizer circuit over $\mathbb{Z} \times \mathbb{Z} \times E$: as in algorithm 5.2, 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 5.2. Proos-Zalka's original algorithm could, again, be interpreted as a discretization of the resulting normalizer circuit.

C.3 Proof of theorem 5.6

In this section we will prove theorem 5.6. The proof uses results of Section 5.3.4; the reader may wish to review that section before proceeding with this proof.

A key ingredient of our proof will be the main simulation result of chapter 4 (**theorem 4.1**). We recall that this theorem can be applied given the following conditions: (i) G is given in an explicitly decomposed form; (ii) any group automorphism gate is specified as a rational matrix A, as in the normal form of theorem 2.8; (iii) any quadratic phase gate is specified as (M, v), where rational M is a matrix and v is a rational vector, as in the normal form of theorem 2.1; (iv) (partial) quantum Fourier transforms are specified by the elementary subgroups it acts on. Note that, in the black-box normalizer-circuit setting, only condition (iv) is granted by assumption.

Hence, given a black-box normalizer circuit acting on a black-box group $\mathbf{G} = \mathbb{Z}^a \times \mathbb{T}^b \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_c} \times \mathbf{B}$, there are two things we need to do to "de-black-box" it, so that the circuit can be classically simulated via theorem 4.1:

1. Decompose the black-box portion of $\mathbf{G}$, $\mathbf{B}$: i.e., find $\mathbb{Z}_{\mathbf{B}} := \mathbb{Z}_{N_{c+1}} \times \cdots \times \mathbb{Z}_{N_{c+d}}$, isomorphic to $\mathbf{B}$, and matrix representations of the isomorphisms φ, φ^{-1} that relate these groups.

2. Calculate *normal forms* for each of the normalizer gates in the computation.

Since we are given access to an oracle for Group Decomposition, we do not to show step 1. In the rest of the paper we show how to tackle task 2.

C.3.1 Switching from black-box encoding to decomposed group encoding.

Throughout the rest of the appendix, we fix $G = G_1 \times \cdots \times G_m$ be the decomposed group

$$G = \mathbb{Z}^a \times \mathbb{T}^b \times \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_c} \times \mathbb{Z}_{N_{c+1}} \times \cdots \times \mathbb{Z}_{N_{c+d}}$$

where G_i corresponds to the ith primitive factor in the above equation, $m := a + b + c + d$ and $\mathbb{Z}_{\mathbf{B}} := \mathbb{Z}_{N_{c+1}} \times \cdots \times \mathbb{Z}_{N_{c+d}}$ is the group given in step 1. above. We recall now that our classical algorithms to decompose our black box group $\mathbf{B}$ output a set of linearly independent generators $b_1, \cdots, b_{k'}$ of $\mathbf{B}$ such that $\mathbf{B} = \langle \beta_1 \rangle \oplus \cdots \oplus \langle \beta_\ell \rangle$, as well as the order $N_i' = N_{c+i}$ of β_i.

In our proof below, we will need to be able to convert elements back and forth from the original black-box encoding and this decomposed group encoding. To change between encodings, we need show how to perform the following tasks:

(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 other words, 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 5.2; from lemma 5.2 we see that Task (b) can be solved efficiently with a polynomial number of calls to the Group Decomposition oracle.

C.3.2 Step (i): Group automorphism gates

Recall that, by assumption, we have access to a group automorphism oracle $\alpha : \mathbf{G} \to \mathbf{G}$ which, by the change of encoding of section C.3.1, can be efficiently turned into a classical rational automorphism $f : G \to G$. Furthermore, f can be efficiently evaluated by using the oracle α and switching 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 (see previous subsection for details).

Our next step is to find a matrix representation A for f. We will assume (for the efficiency of this algorithm) that the size and precision of the coefficients are upper bounded by a known parameter 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.[2]

We now show how to obtain the matrix representation A from by evaluating $f : \mathbb{Q}^{a+b+c+d} \to \mathbb{Q}^{a+b+c+d}$ (which we view as a function sending rational inputs to rational outputs). Because this function is a group automorphism we know that it further fulfills

$$f(x) \equiv f(x') \bmod G \quad \text{if } x \equiv x' \bmod G, \tag{C.5}$$

where two vectors are equal modulo G if each pair of corresponding entries are equal modulo $\mathrm{char}(G_i)$. Recalling that any matrix representation A for f has a specific block-structure of characterized by lemma 2.8(2.31), we show how find a matrix representation A for f coefficient-by-coefficient.

Let $c_i = \mathrm{char}(G_i)$ be the characteristic of the ith primitive-group factor G_i of G as in lemma 2.8. Then, for most entries of A this is trivial: note that we have

$$A_{i,j} \equiv f(e_j)_i \bmod c_i. \tag{C.6}$$

Hence by evaluating f on the unit vectors e_i, we can determine $A_{i,j}$ modulo c_i. Thus we can evaluate $A_{\mathbb{T}F}$ 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 $c_i = \mathrm{char}(\mathbb{T}) = 1$ in this case). We can instead evaluate $f(e_j/\Delta)$ for some large integer Δ:

$$A_{i,j}/\Delta \equiv f(e_j/\Delta)_i \bmod c_i \tag{C.7}$$

which allows us to determine $A_{i,j}$ modulo Δc_i for our choice of α. Choosing $\Delta > 2D$ then allows us to determine $A_{i,j}$ exactly for the case of $A_{\mathbb{T}\mathbb{T}}$.

C.3.3 Step (ii): quadratic phase gates

Next, we consider a quadratic phase gate ξ, implemented as a classical circuit family $q : G \to \mathbb{Q}$ such that

$$\xi(g) = \mathrm{e}^{2\pi \mathrm{i} q(g)} \quad \forall g \in G. \tag{C.8}$$

For simplicity, we assume that we have changed from the original encoding $\mathbf{G}$ to G using the same technique as in previous section and treat the elements of G as a vector in $\mathbb{Q}^{a+b+c+d}$. Our next goal is to write the quadratic function $\xi(g)$ in the normal form given by theorem 2.1, i.e. find M, v as in theorem such that

$$\xi(g) = \mathrm{e}^{\pi \mathrm{i} \left(g^T M g + C^T g + 2 v^T g\right)}. \tag{C.9}$$

[2]Note that D can be inferred from the precision bound n_{out} (s. 5.2.2) of an automorphism gate: because the output of α can only be n_{out} bits larger than its input, it follows that the size of the denominator/numerator of every matrix element increases at most by $D = 2^{n_{\mathrm{out}}}$. A similar argument will hold for quadratic phase gates.

Here, q, M, C, and v, are rational by the assumptions and we do not need to find C, which is determined by M. Furthermore, due to theorem 2.1, lemmas 2.10-2.8, M, v have some additional structural features, namely: 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, up to a permutation, M has the following upper triangular structure

$$M := \begin{pmatrix} M_{\mathbb{T}\mathbb{Z}} & M_{\mathbb{T}F} & M_{\mathbb{T}\mathbb{T}} \\ M_{F\bullet\mathbb{Z}} & M_{F\bullet F} & 0 \\ M_{\mathbb{Z}\mathbb{Z}} & 0 & 0 \end{pmatrix} \tag{C.10}$$

where (i) $M_{\mathbb{Z}\mathbb{Z}}$ and $M_{\mathbb{T}\mathbb{T}}$ are arbitrary integer matrices; (ii) $M_{F\bullet\mathbb{Z}}$ and $M_{\mathbb{T}F}$ have rational entries, 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 $\mathbb{Z}_{N_i}$; (iii) $M_{F\bullet F}$ is a rational matrix with coefficients of the form $M(i,j) = \frac{\alpha_{i,j}}{\gcd(N_i,N_j)}$ where $\alpha_{i,j}$ are arbitrary integers, and N_i is the order of the i-th cyclic subgroup $\mathbb{Z}_{N_i}$; (iv) $M_{\mathbb{T}\mathbb{Z}}$ 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. $M^T_{\mathbb{Z}\mathbb{Z}} = M_{\mathbb{T}\mathbb{T}}$, $M^T_{F\bullet\mathbb{Z}} = M_{\mathbb{T}F}$, $M^T_{F\bullet F} = M_{F\bullet F}$, and $M^T_{\mathbb{T}\mathbb{Z}} = M_{\mathbb{T}\mathbb{Z}}$.

We now show how to compute M and v, To this end, we 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

$$\xi(x+y) = \xi(x)\xi(y)\mathrm{e}^{2\pi\mathrm{i}\, x^T M y} \tag{C.11}$$

for any $x, y \in G$, and therefore

$$x^T M y \equiv q(x+y) - q(x) - q(y) \bmod \mathbb{Z}. \tag{C.12}$$

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.

$$M_{i,j} = e_i^T M e_j \equiv q(e_i + e_j) - q(e_i) - q(e_j) \bmod \mathbb{Z}. \tag{C.13}$$

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 $x = \Delta^{-1}e_i$, and $y = e_j$, such that the coefficient $M(i,j)$ is in the submatrix $M_{\mathbb{Z}\mathbb{Z}}$ and $1/\Delta$ is an element of the circle group with $\Delta > 2D$, where D is the precision bound. We obtain an analogous equation

$$\left(\frac{e_i^T}{\Delta} M e_j\right) \equiv \frac{M_{i,j}}{\Delta} \equiv q(\Delta^{-1}e_i + e_j) - q(\Delta^{-1}e_i) - q(e_j) \bmod \mathbb{Z}, \tag{C.14}$$

which allows us to determine $M_{i,j}$: since the number $M_{i,j}/\Delta$ 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 $y = \Delta^{-1}e_j$.

Once we determine all the entries of M in this manner, we get immediately the vector C as well (since $C(i) = c_i M(i,i)$) (theorem 2.1). It is then straightforward to calculate the vector v. Thus we can efficiently find the normal form of $\xi(g)$ through polynomially many uses of the classical function q.

C.4 Extending theorem 5.6 to the abelian HSP setting

In this appendix, we briefly discuss that theorem 5.6 (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 5.3.3. 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 5.5) states that the quantum algorithm abelian HSP is a normalizer circuits over a group of the form $\mathbb{Z}_{d_1} \times \ldots \times \mathbb{Z}_{d_m} \times \mathcal{O}$, where $\mathcal{O}$ is a group associated with the abelian HSP oracle f via the formula (5.19). 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 5.3.3 that one can use the hidden subgroup problem oracle to perform certain multiplications implicitly.

We show next that theorem 5.6 can be re-casted in the HSP setting as "*the ability to decompose the oracular group $\mathcal{O}$ renders normalizer circuits over $\mathbb{Z}_{d_1} \times \ldots \times \mathbb{Z}_{d_m} \times \mathcal{O}$ 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 $G \times \mathbb{Z}_{c_1} \times \cdots \times \mathbb{Z}_{c_m}$, 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 5.6 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 5.5) has precisely the same form as the HSP oracle. Using the HSP oracle as a subroutine in algorithm 5.5 (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 5.9 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 5.9.

Appendix D

Supplement to chapter 6

D.1 Proof of theorem 6.3, part II

In this appendix, we derive equations (6.23-6.26) finishing the proof of theorem 6.3. We treat the different types of normalizer gates separately below.

1. **Automorphism Gates.** It follows from the definition in section 6.2 that any hypergroup automorphism α fulfills $n^{\alpha(c)}_{\alpha(a),\alpha(b)} = n^c_{a,b}$ and $w_{\alpha(a)} = w_a$ for all $a, b, c \in \mathcal{T}$. Combining these properties with (6.18) we derive (6.24).

2. **Quadratic phase gates.** The RHS of (6.25) follows because D_ξ is diagonal, hence, commutes with $Z_\mathcal{T}(\mathcal{X}_\mu)$.

   The LHS can be derived by explicitly evaluating the action of $D_\xi X_\mathcal{T}(a) D_\xi^\dagger$ on basis states in $\mathcal{B}_\mathcal{T} = \{|b\rangle, b \in \mathcal{T}\}$ using that, for any $c, c' \in ab$ and any quadratic function ξ with associated bicharacter B, the following identities holds:

   $$\text{(i)} \quad \xi(c) = \xi(c') = \xi(ab), \quad \text{(ii)} \quad \xi(c) = \xi(a)\xi(b)B(a,b), \quad \text{(iii)} \quad B(a,b) = \mathcal{X}_{\beta(a)}(b)$$

   for some homomorphism β from $\mathcal{T}$ onto $\mathcal{T}^*_{\text{inv}}$. Above, (i) follows from the triangle inequality and given properties, namely, $\xi(ab) = \sum_{c\in ab} n^c_{ab}\xi(c)$, $|\xi(c)| = 1$, and $\sum_c n^c_{ab}{=}1$; (ii) follows from (i) and the definition of quadratic function; and last, the normal form for bicharacters (iii) can be obtained by extrapolating the group-setting argument given in [134], lemma 5.

3. **Quantum Fourier transforms.** We derive (6.26) by explicitly computing the action of Pauli operators on the states $\mathcal{F}_\mathcal{T}^\dagger|\mathcal{X}_\mu\rangle$ (which form a basis) using (6.10):

   $$X_\mathcal{T}(a)\mathcal{F}_\mathcal{T}^\dagger|\mathcal{X}_\mu\rangle = \sum_{b\in\mathcal{T}} \sqrt{\tfrac{w_b w_{\overline{\mathcal{X}_\mu}}}{\varpi_\mathcal{T}}}\overline{\mathcal{X}_\mu}(b) X_\mathcal{T}(a)|b\rangle \tag{D.1}$$

   $$\overset{(6.18)}{=} \sum_{b\in\mathcal{T}} \sqrt{\tfrac{w_b w_{\overline{\mathcal{X}_\mu}}}{\varpi_\mathcal{T}}}\overline{\mathcal{X}_\mu}(b) \left(\sum_c \sqrt{\tfrac{w_b}{w_c}}\, n^c_{a,b}|c\rangle\right)$$

   $$\overset{(6.2)}{=} \sum_{c\in\mathcal{T}} \sqrt{\tfrac{w_c w_{\overline{\mathcal{X}_\mu}}}{\varpi_\mathcal{T}}} \left(\sum_b n^b_{\overline{a},c}\overline{\mathcal{X}_\mu}(b)\right)|c\rangle = \sum_{c\in\mathcal{T}} \sqrt{\tfrac{w_c w_{\overline{\mathcal{X}_\mu}}}{\varpi_\mathcal{T}}}\overline{\mathcal{X}_\mu}(\overline{a})\overline{\mathcal{X}_\mu}(c)|c\rangle$$

   $$= \mathcal{X}_\mu(a)\mathcal{F}_\mathcal{T}^\dagger|\mathcal{X}_\mu\rangle = \mathcal{F}_\mathcal{T}^\dagger Z_{\mathcal{T}^*}(a)|\mathcal{X}_\mu\rangle \tag{D.2}$$

$$
\begin{aligned}
Z_{\mathcal{T}}(\mathcal{X}_\mu)\mathcal{F}_{\mathcal{T}}^\dagger|\mathcal{X}_\nu\rangle &\overset{(6.18)}{=} \sum_{b\in\mathcal{T}} \sqrt{\frac{w_b w_{\overline{\mathcal{X}_\nu}}}{\varpi_{\mathcal{T}}}}\, \mathcal{X}_\mu(b)\overline{\mathcal{X}_\nu}(b)|b\rangle \\
&= \sum_{b\in\mathcal{T}} \sqrt{\frac{w_b w_{\overline{\mathcal{X}_\nu}}}{\varpi_{\mathcal{T}}}} \left(\sum_{\overline{\mathcal{X}_\gamma}\in\mathcal{T}^*} m_{\mu\overline{\nu}}^{\overline{\gamma}} \mathcal{X}_{\overline{\gamma}}(b) \right) |b\rangle \qquad \text{(D.3)} \\
&= \sum_{\overline{\mathcal{X}_\gamma}\in\mathcal{T}^*} \sqrt{\frac{w_{\overline{\mathcal{X}_\nu}}}{w_{\mathcal{X}_{\overline{\gamma}}}}} m_{\mu\overline{\nu}}^{\overline{\gamma}} \left(\sum_{b\in\mathcal{T}} \sqrt{\frac{w_b w_{\mathcal{X}_{\overline{\gamma}}}}{\varpi_{\mathcal{T}}}}\, \mathcal{X}_{\overline{\gamma}}(b)|b\rangle \right) \\
&= \mathcal{F}_{\mathcal{T}}^\dagger \sum_{\overline{\mathcal{X}_\gamma}\in\mathcal{T}^*} \sqrt{\frac{w_{\overline{\mathcal{X}_\nu}}}{w_{\mathcal{X}_{\overline{\gamma}}}}} m_{\mu\overline{\nu}}^{\overline{\gamma}} |\mathcal{X}_\gamma\rangle = \mathcal{F}_{\mathcal{T}}^\dagger X_{\mathcal{T}^*}(\overline{\mathcal{X}_\mu})|\mathcal{X}_\nu\rangle, \qquad \text{(D.4)}
\end{aligned}
$$

where we used $w_{\mathcal{X}_{\overline{\gamma}}} = w_{\mathcal{X}_\gamma}$ and $m_{\mu\overline{\nu}}^{\overline{\gamma}} = m_{\overline{\mu}\nu}^{\gamma}$ from section 6.2. The analogous statement for partial QFTs follows straightforwardly using that character hypergroup of $\mathcal{T}_1 \times \cdots \times \mathcal{T}_m$ is $\mathcal{T}_1^* \times \cdots \times \mathcal{T}_m^*$ [252] and the tensor-product structure of Pauli operators (section 6.5.1).

4. **Pauli gates.** We can now use (6.26) to get $Z_{\mathcal{T}}(\mathcal{X}_\varsigma)X_{\mathcal{T}}(a)Z_{\mathcal{T}}(\mathcal{X}_\varsigma)^\dagger = \mathcal{X}_\varsigma(a)X_{\mathcal{T}}(a)$ and $Z_{\mathcal{T}}(\mathcal{X}_\varsigma)Z_{\mathcal{T}}(\mathcal{X}_\mu)Z_{\mathcal{T}}(\mathcal{X}_\varsigma)^\dagger = Z_{\mathcal{T}}(\mathcal{X}_\mu)$ since invertible characters are quadratic functions with trivial B and β. Moreover, we can apply (6.26) and repeat the argument in the character basis, obtaining $X_{\mathcal{T}}(s)X_{\mathcal{T}}(a)X_{\mathcal{T}}(s)^\dagger = X_{\mathcal{T}}(a)$, $X_{\mathcal{T}}(s)Z_{\mathcal{T}}(\mathcal{X}_\mu)X_{\mathcal{T}}(s)^\dagger = \mathcal{X}_\mu(\bar{s})Z_{\mathcal{T}}(\mathcal{X}_\mu)$. Equation (6.23) is derived combining these expressions.

D.2 Quadratic functions

We prove that the functions ξ_i, ξ_j, ξ_k and ξ defined in section 6.4.2 are quadratic. The quadraticity of ξ_x, with $x = i, j, k$, follows from the fact that the function can be obtained by composing the quotient map $\overline{Q}_8 \to \overline{Q}_8/\{\pm 1, \pm x\} \cong \mathbb{Z}_2$, with the isomorphism $\overline{Q}_8/\langle x\rangle \to \mathbb{Z}_2$ and the map $\mathbb{Z}_2 \to \mathbb{C} : a \to i^a$; since the latter is a quadratic function of $\mathbb{Z}_2$ [134], it follows easily that ξ_x is a quadratic function of $\overline{Q}_8$. Note that in this derivation we implicitly use that $\{\pm 1, \pm x\}$ is a subhypergroup of $\overline{Q}_8$ [255], that the quotient $\overline{Q}_8/S$ is an abelian hypergroup for any subhypergroup S, and that the quotient map $\overline{Q}_8 \to \overline{Q}_8/S$ is a hypergroup homomorphism [255].

To show that $\xi : \overline{Q}_8 \times \overline{Q}_8 \to \mathbb{C}$ is quadratic, we use the fact, prove below, that the function $B(C_x, C_y) := f_{C_x}(C_y)$ is a symmetric bi-character of $\overline{Q}_8$. Given that property as a promise and recalling that $\xi((C_x, C_y)) = B(C_x, C_y)$ (by definition), we can see that

$$
\begin{aligned}
\xi\left((C_a, C_b)\cdot(C_c, C_d)\right) &= B(C_aC_c, C_bC_d) = B(C_a, C_bC_d)B(C_c, C_bC_d)) \\
&= B(C_a, C_b)B(C_a, C_d)B(C_c, C_b)B(C_c, C_d) \\
&= \xi\left((C_a, C_b)\right)\xi\left((C_c, C_d)\right) B(C_a, C_d)B(C_c, C_b) \\
&= \xi\left((C_a, C_b)\right)\xi\left((C_c, C_d)\right) B'\left((C_a, C_b), (C_c, C_d)\right), \qquad \text{(D.5)}
\end{aligned}
$$

where we define $B'\left((C_a, C_b), (C_c, C_d)\right) = B(C_a, C_d)B(C_c, C_b)$. The latter is easily seen to be a bi-character of $\overline{Q}_8 \times \overline{Q}_8$, so that ξ is indeed quadratic.

It remains to show that $B(C_x, C_y)$ is a symmetric bi-character. To see this, note, that both the quotient hypergroup $\overline{Q}_8/\{\pm 1\}$ and the subhypergroup of linear characters $\widehat{Q_{8\ell}}$ of Q_8 are isomorphic to the Klein four group $\mathbb{Z}_2 \times \mathbb{Z}_2$. Observe next that the map $C_x \to f_{C_x}$ is a homomorphism $\overline{Q}_8 \to \widehat{Q_{8\ell}}$, as it can be obtained composing the quotient map $\overline{Q}_8 \to \overline{Q}_8/\{\pm 1\}$ with a chain of isomorphisms $\overline{Q}_8/\{\pm 1\} \to \mathbb{Z}_2 \times \mathbb{Z}_2 \to \widehat{Q_{8\ell}}$. This latter fact implies that $B(C_x, C_y) = f_{C_x}(C_y)$ is a character in both arguments, hence, a bi-character. Finally, it is routine to check that $B(C_x, C_y) = B(C_y, C_x)$ by explicit evaluation, which completes the proof.

D.3 Efficient vs. doubly efficient computable hypergroups

We give an example of an efficient computable abelian hypergroup that cannot be doubly efficiently computable unless we are given the ability to compute discrete logarithms over $\mathbb{Z}_p^\times$. In chapter 5, we discussed that this problem is believed to be hard for classical computers (being the basis of the Diffie-Hellman public-key cryptosystem [60]), yet it can be solved via Shor's discrete-log quantum algorithm [4]. This problem reduces to the so-called hidden subgroup problem over $\mathbb{Z}_{p-1}^2$ [204] for a certain hiding function f, which defines a group homomorphism from $\mathbb{Z}_{p-1}^2$ to $\mathbb{Z}_p^\times$ (chapter 5.3.1).

Considering now the group $\mathcal{T} = \mathbb{Z}_{p-1}^2 \times \mathbb{Z}_p^\times$, which is manifestly efficiently computable following our definition, we can define an efficiently computable group automorphism $\alpha : \mathcal{T} \to \mathcal{T} : (m, x) \to (m, f(m)x)$. We show that $\mathcal{T}$ cannot be doubly efficiently computable unless the initial hidden subgroup problem and, hence, the discrete logarithm problem, can be solved in probabilistic polynomial time (which, up to date, is not possible).

First, we show that, if we are able to compute[1] α^*, we must also be able to compute $f^* : \widehat{\mathbb{Z}}_p^\times \to \widehat{\mathbb{Z}}_{p-1}^2$ (the dual of f, which is defined analogously to α^*), since for any $\mathcal{X}_{\mu,\nu} := \mathcal{X}_\mu \otimes \mathcal{X}_\nu$ we have

$$\mathcal{X}_{\alpha^*(\mu,\nu)}(m, x) = (\mathcal{X}_\mu \otimes \mathcal{X}_\nu)(m, f(m)x) = \mathcal{X}_\mu(m)\mathcal{X}_\nu(f(m))\mathcal{X}_\nu(x) = \mathcal{X}_\mu(m)\mathcal{X}_{f^*(\nu)}(m)\mathcal{X}_\nu(x),$$

consequently, $\mathcal{X}_{\alpha^*(\mu,\nu)} = \left(\mathcal{X}_\mu \cdot \mathcal{X}_{f^*(\nu)}\right) \otimes \mathcal{X}_\nu$. Hence, if we can evaluate α^* on any character $\mathcal{X}_1 \otimes \mathcal{X}_\mu$, then we can determine $\mathcal{X}_{f^*(\nu)}$, the value of f^* on ν, for any $\mathcal{X}_\nu$. If we now evaluate $f^*(\mu_i)$ on all elements of a $O(\log p)$-sized randomly-obtained generating set $\{\mathcal{X}_{\mu_i}\}$ of $\widehat{\mathbb{Z}}_p^\times$ and use our classical algorithms (theorem 2.2) to solve the system of equations $\{[f^*(\mu_i)](x) = \mathcal{X}_{\mu_i}(f(x)) = 1, x \in \mathbb{Z}_{p-1}^2\}$, whose solutions are those x for which $f(x) = e$, we have found (in these x's) generators of the hidden subgroup. This finishes the reduction.

D.4 Implementing normalizer circuits over $\overline{G}$

In this section, we present more details on how to efficiently implement normalizer circuits over the hypergroup $\overline{G}$ when we are working in the Hilbert space $\mathcal{H}_G = \{|g\rangle \,|\, g \in G\}$ labeled by elements of the group. As described in section 6.4.2, normalizer circuits over $\overline{G}$ can be thought of as operating entirely within the subspace $\mathcal{I}_G \leq \mathcal{H}_G$ of conjugation invariant wavefunctions; however, we will describe these operations in this section in terms of how they operate on the entire Hilbert space. In section D.4.1, we discuss operations applied in the character basis, and in section D.4.2, we discuss the same in the conjugacy class basis.

D.4.1 Working in the character basis

Normalizer circuits allow of the following operations to be performed in the character class basis: preparation of initial states; Pauli, automorphism, and quadratic phase gates; and measurement of final states. It should be easy to understand how each of these could be implemented efficiently if we worked in a basis $\{|\mu\rangle \,|\, \mu \in \mathrm{Irr}(G)\}$ of irrep labels. However, the Hilbert space $\mathcal{H}_G$ is only naturally labeled by group elements, which is why, in section 6.4.2, we defined the character basis states $\{|\mathcal{X}_\mu\rangle \,|\, \mathcal{X}_\mu \in \widehat{G}\}$ in the element basis.

Below, we will describe how to implement an isometry $|\mathcal{X}_\mu\rangle \overset{\tau}{\mapsto} |\mu\rangle$ and its inverse, using a readily available choice for the basis $\{|\mu\rangle \,|\, \mu \in \mathrm{Irr}(G)\}$. It should then be clear that we can

[1] We are implicitly assuming that there are efficient unique classical encodings for representing the characters of $\mathbb{Z}_p^\times$, which is a strong yet *weaker* assumption that $\mathbb{Z}_p^\times$ being doubly efficiently computable.

implement each of the above gates by applying τ, performing the operation in the irrep label basis, and then applying τ^{-1}. To prepare an initial state $|\mathcal{X}_\mu\rangle$, we prepare $|\mu\rangle$ in the irrep label basis and then apply τ^{-1}. Finally, to measure in the character basis, we apply τ and then read the irrep label.

Our definition of the irrep label basis $\{|\mu\rangle \,|\, \mu \in \mathrm{Irr}(G)\}$ comes from the definition of the QFT over the group G. Recall that the QFT over any finite group G [7], denoted $\mathcal{F}_G$, is a unitary gate that sends an element state $|g\rangle$, for any $g \in G$, to a weighted superposition $|G|^{-1/2} \sum_{\mu \in \mathrm{Irr}(G)} d_\mu |\mu, \mu(g)\rangle$, where $|\mu\rangle$ is a state that labels the irrep μ and $|\mu(g)\rangle$ is a d_μ^2 dimensional state defined via

$$|\mu(g)\rangle = \Big(\mu(g) \otimes I_{d_\mu}\Big) \sum_{i=1}^{d_\mu} \frac{|i,i\rangle}{\sqrt{d_\mu}} = \sum_{i,j=1}^{d_\mu} \frac{[\mu(g)]_{i,j}}{\sqrt{d_\mu}} |i,j\rangle. \tag{D.6}$$

This transformation $\mathcal{F}_G$ has been extensively studied in the HSP literature and efficient quantum implementations over many groups are currently known (including the symmetric group, wreath products of polynomial-sized groups and metabelian groups [7]).

To see how we can use this, let's look at what $\mathcal{F}_G$ does to a character class state. In (6.14), we defined the state $|C_x\rangle$, when living inside the Hilbert space $\mathcal{H}_G$, to be a uniform superposition over the elements in the class C_x. If we apply $\mathcal{F}_G$ to this state, the result is

$$\begin{aligned}
\mathcal{F}_G|C_x\rangle &= \frac{1}{\sqrt{|C_x|}} \sum_{g \in C_x} \mathcal{F}_G|g\rangle \\
&= \frac{1}{\sqrt{|C_x|}} \sum_{g \in C_x} \frac{1}{\sqrt{|G|}} \sum_{\mu \in \mathrm{Irr}(G)} d_\mu |\mu\rangle \otimes \sum_{i,j=1}^{d_\mu} \frac{[\mu(g)]_{i,j}}{\sqrt{d_\mu}} |i,j\rangle \\
&= \frac{1}{\sqrt{|C_x||G|}} \sum_{\mu \in \mathrm{Irr}(G)} d_\mu |\mu\rangle \otimes \sum_{i,j=1}^{d_\mu} \frac{[\sum_{g \in C_x} \mu(g)]_{i,j}}{\sqrt{d_\mu}} |i,j\rangle.
\end{aligned}$$

To simplify further, we need to better understand the sum in the numerator on the right.

The sum $\sum_{g \in C_x} \mu(g)$ is more easily analyzed if we write it as $(|C_x|/|G|) \sum_{h \in G} \mu(x^h)$: by standard results on orbits of group actions [273], each $\mu(g)$, for $g \in C_x$, arises the same number of times in the sum $\sum_{h \in G} \mu(x^h)$, which hence must be $|G|/|C_x|$ times for each, so we have $(|C_x|/|G|) \sum_{h \in G} \mu(x^h) = \sum_{g \in C_x} \mu(g)$. The sum $(1/|G|) \sum_{h \in G} \mu(x^h)$ may be familiar, as it is well known to be $\frac{1}{d_\mu} \chi_\mu(x) I$ [276].[2]

Putting these parts together, we can see that

$$\begin{aligned}
\mathcal{F}_G|C_x\rangle &= \frac{1}{\sqrt{|C_x||G|}} \sum_{\mu \in \mathrm{Irr}(G)} d_\mu |\mu\rangle \otimes \sum_{i=1}^{d_\mu} \frac{|C_x| \chi_\mu(x)}{d_\mu \sqrt{d_\mu}} |i,i\rangle \\
&= \sqrt{\frac{|C_x|}{|G|}} \sum_{\mu \in \mathrm{Irr}(G)} d_\mu \frac{\chi_\mu(x)}{d_\mu} \left(\frac{1}{\sqrt{d_\mu}} \sum_{i=1}^{d_\mu} |\mu, i, i\rangle \right),
\end{aligned}$$

which is rewritten in our usual hypergroup notation as

$$\mathcal{F}_G|C_x\rangle = \sum_{\mathcal{X}_\mu \in \widehat{G}} \sqrt{\frac{w_{C_x} w_{\mathcal{X}_\mu}}{w_{\overline{G}}}} \mathcal{X}_\mu(C_x) \left(\frac{1}{\sqrt{d_\mu}} \sum_{i=1}^{d_\mu} |\mu, i, i\rangle \right). \tag{D.7}$$

[2] This is a simple application of Schur's lemma. This sum is a G-invariant map $\mathcal{H}_G \to \mathcal{H}_G$, so it must be a constant times the identity. The constant is easily found by taking the trace of the sum.

This precisely mirrors the definition of $\mathcal{F}_{\overline{G}}$ with $|\mathcal{X}_\mu\rangle$ replaced by $d_\mu^{-1/2} \sum_{i=1}^{d_\mu} |\mu, i, i\rangle$. We will denote the latter state below by $|\mu_{\text{diag}}\rangle$. Thus, it follows by (6.7) that $\mathcal{F}_G|\mathcal{X}_\mu\rangle = |\mu_{\text{diag}}\rangle$.

To implement the operation τ, we apply $\mathcal{F}_G$ and then *carefully* discard the matrix index registers.[3] By the above discussion, we can see that this maps $|\mathcal{X}_\mu\rangle$ to the state $|\mu\rangle$, so this implements the operation τ correctly for any conjugation invariant state.

To implement the operation τ^{-1}, we do the above in reverse. Starting with a state $|\mu\rangle$, we adjoin matrix index registers, prepare a uniform superposition over $|1\rangle, \ldots, |d_\mu\rangle$ in the first index register using the inverse Fourier transform over the abelian group $\mathbb{Z}_{d_\mu}$, and then copy the first index register to the second[4] to get the state $|\mu_{\text{diag}}\rangle$. Finally, we apply $\mathcal{F}_G^\dagger$ to get the state $|\mathcal{X}_\mu\rangle$ per the calculations above.

As discussed earlier, the operations τ and τ^{-1} are all that we need in order to implement each of the required operations of normalizer circuits over $\overline{G}$ in the character basis.

D.4.2 Working in the character class basis

Most of the time, gates applied in the conjugacy class basis arise from operations on the whole group. For example, automorphisms of conjugacy classes often arise from automorphisms of the group. Likewise, Pauli Z operators in the conjugacy class basis are applications of characters, which are defined on the whole group, and Pauli X operators can also be implemented using multiplication in the group. Hence, it remains only discuss how to prepare initial states and measure in the conjugacy class basis.

For this, we need to assume that we can perform certain operations on conjugacy classes, as described in the following definition.

Definition D.1. Let $C_1, \ldots, C_m$ be the conjugacy classes of G. Consider the following operations for working with conjugacy classes:

- Given a conjugacy class label i, produce the size of this class, $|C_i|$.
- Given an $x \in G$, produce the pair (i, j), where $x = x_j$ in the class $C_i = \{x_1, \ldots, x_t\}$.
- Given a pair (i, j), produce the element x_j from C_i.

If each of these operations can be performed efficiently, then we say that we can *compute efficiently with conjugacy classes* of G.

We note that this assumption is trivial for abelian groups since each element is in its own conjugacy class. For some common examples of nonabelian groups, such as the dihedral and Heisenberg groups (and their higher nilpotent generalizations), elements are normally encoded in this manner already, so no additional assumption is actually required. For other common examples like the symmetric group, while elements are not always encoded directly in this manner, it is easy to see how the above calculations can be performed efficiently. In general, while we must formally make this assumption, we are not aware of any group for which these calculations cannot be performed efficiently.

If we can compute efficiently with conjugacy classes of G, then we can prepare initial states as follows. Starting with the conjugacy class label i in a register, we first compute the size $|C_i|$ into a new register. Next, we adjoin another new register and invoke the inverse Fourier

[3] In full detail, we do the following. First, apply the map $|i, j\rangle \mapsto |i, j - i\rangle$, which gives $|i, 0\rangle$ when applied to $|i, i\rangle$. Next, write down d_μ in a new register and then invoke the Fourier transform over $\mathbb{Z}_{d_\mu}$ on the first index register. The result of this will always be $|0\rangle$, so after uncomputing d_μ, we are left with the state $|0, 0\rangle$ in the index registers regardless of the value of μ. At that point, they are unentangled and can be safely discarded.

[4] Or rather, apply the map $|i, j\rangle \mapsto |i, i + j\rangle$, which gives $|i, i\rangle$ when applied to $|i, 0\rangle$.

transform over the abelian group $\mathbb{Z}_{|C_i|}$. After uncomputing the size $|C_i|$, we are left with the superposition $M^{-1/2}\sum_{j=1}^{M}|i,j\rangle$, where $M = |C_i|$. Finally, we apply the operation that turns pairs into group elements to get $M^{-1/2}\sum_{j=1}^{M}|x_j\rangle$, where $x_1, \ldots, x_M$ are the elements of C_i, which is the desired initial state.

To perform a measurement in the conjugacy class basis, we can do the reverse of how we prepared the initial states in order to produce a conjugacy class label $|i\rangle$ in a register. Alternatively, we can simply measure in the group element basis and then, afterward, compute the conjugacy class of this element. These two approaches will give identical measurement probabilities.

Finally, we note that the two operations just described are the equivalent of the operations τ and τ^{-1} from section D.4.1 for the conjugacy class basis.[5] As a result, if we do have gates that can be easily implemented on conjugacy classes but do not extend easily to the whole group, then we can implement these gates in the same manner as in the character basis: apply τ to convert into a basis of conjugacy class labels $\{|i\rangle \,|\, C_i \in \overline{G}\}$, apply the gate in this basis, and then apply τ^{-1} move back to the conjugacy class basis in $\mathcal{H}_G$.

Thus, we can see that the ability to compute efficiently with conjugacy classes of G allows us to fully implement normalizer circuits operations applied in the conjugacy class basis. If we also have an efficient QFT for G, then as we saw in the previous section, we can implement normalizer circuits operations applied in the character basis as well. Together, these two assumptions allow us to fully implement normalizer circuits over $\overline{G}$ when working in the Hilbert space $\mathcal{H}_G$.

[5] Indeed, the separation of a group element label into a conjugacy class label and an index label is analogous to how, in the space of irreducible representations, we separate each basis element into an irrep label and matrix index labels. It is frequently assumed that we can separate the latter into different registers whenever convenient, so our assumption that we can do the same for conjugacy classes is only affording the same convenience for the hypergroup $\overline{G}$ that is often assumed for $\widehat{G}$.

Appendix E

Normalizer circuits over $\mathbb{R}$ generate all bosonic Gaussian unitaries

To complement our discussion in sections 0.2.1-4.1, we show in this appendix that normalizer circuits over real groups of the form $\mathbb{R}^m$ (which were not considered in this thesis) coincide with the well-known definitions of (bosonic) Gaussian unitaries, which are central in continuous-variable quantum information processing [175, 29, 140, 31, 32, 141, 30, 142, 143]. This appendix is based on unpublished joint work with Geza Giedke [277].

To begin with, we expand on an earlier comment in section 4.1, where we mentioned that the infinite-group normalizer circuit model (chapter 1.4) is well-defined for *any* abelian group that has a locally compact Hausdorff topology and and, in particular, for groups of the form $\mathbb{Z}^a \times \mathbb{T}^b \times \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_c}$ with additional $\mathbb{R}^m$ factors. Here, we discuss how to define normalizer gates and Pauli operators over $\mathbb{R}^m$ groups. In fact, this turns out to be slightly easier than for the groups in chapter 4, since $\mathbb{R}^m$ groups are always isomorphic to their own character groups and have several other benign algebraic features: namely, they are vector spaces, have a well-defined inner product and, because $\mathbb{R}$ is a field of zero characteristic, do not contain zero divisors. In fact, because $\mathbb{R}^m = (\mathbb{R}^m)^* \cong \widehat{\mathbb{R}^m}$ (section 2.1.2) all designated bases (1.20) and Pauli operators (4.5-4.6) are labeled by the same index group $\mathbb{R}^m$; hence, the distinction between G and G^* can be dropped from our formalism, similarly to the the finite abelian group setting of chapter 3).

We now relate normalizer circuits over $\mathbb{R}^m$ to unitary gates that act on harmonic oscillators. Note, first, that the Hilbert space of the computation $\mathcal{H}_{\mathbb{R}^m} = \mathcal{H}_{\mathbb{R}}^{\otimes m}$ has a group element basis $\{|x\rangle_X, x \in \mathbb{R}^m\}$ and a character basis $\{|p\rangle_P, p \in \mathbb{R}^m\}$ that are related through the quantum Fourier transform over $\mathbb{R}^m$ as

$$|p\rangle_P = \int_{\mathbb{R}^m} \mathrm{d}x \, \overline{\mathrm{e}^{2\pi \mathrm{i} p x}} |x\rangle_X. \tag{E.1}$$

Unlike for $\mathbb{Z}^n$ integer groups, there exists a natural 1-to-1 mapping between the elements and characters of $\mathbb{R}^m$ that lets us implement the QFT over $\mathbb{R}^m$ as the unitary gate $\mathcal{F}_{\mathbb{R}^m}$ that implements the map $|x\rangle_X \to \int_{\mathbb{R}^m} \mathrm{d}x \, \mathrm{e}^{2\pi \mathrm{i} p x} |p\rangle_X$. Realize that, w.l.o.g, we can identify $\mathcal{H}_{\mathbb{R}^m}$ with the Hilbert space $\mathcal{H}_{\mathrm{osc}}^{\otimes m}$ of m harmonic oscillators and the states $|x\rangle_X$ (respectively $|p\rangle_P$) with the joint eigenstates of all position operators $\hat{X}_i$ (respectively, all momentum ones $\hat{P}_j$) of the m-mode harmonic-oscillator. With this identification, it follows from [215, 2.80,2.83] that the generalized Pauli operators $Z_{\mathbb{R}^m}(p)$, $X_{\mathbb{R}^m}(x)$ over $\mathbb{R}^m$ coincide with tensor-products of so-called position and momentum shift operators in the Gaussian formalism:

$$X_{\mathbb{R}^m}(x) = \bigotimes_{i=1}^m X(x_i) := \bigotimes_{i=1}^m \exp\left(\mathrm{i}\, x_i \hat{P}_i\right), \quad Z_{\mathbb{R}^m}(p) = \bigotimes_{i=1}^m Z(p_i) := \bigotimes_{i=1}^m \exp\left(\mathrm{i}\, p_i \hat{X}_i\right). \tag{E.2}$$

Pauli operators are, hence, examples of normalizer circuits[1] over $\mathbb{R}^m$ that are *Gaussian* unitaries, i.e., gates that can be generated by Hamiltonians that are (at most) quadratic polynomials of position and momentum operators. Furthermore, it follows from [215, 2.87,2.89] that single-*mode* (in our notation, *mode* is synonym of register) partial quantum Fourier transforms over $\mathbb{R}^m$ (E.1) are instances of so-called single-mode Gaussian *phaseshifters*

$$U(\theta) := \exp\left(\theta \hat{a}^\dagger \hat{a}\right) = \exp\left(\theta \frac{\hat{X}^2+\hat{P}^2}{2}\right), \tag{E.3}$$

with angle $\theta = 3\pi/4$, (here and below we fix physical units so that $1 = \hbar = \omega$, where ω is the oscillator frequency); it follows that any QFT over $\mathbb{R}^m$ and, in particular, the global m-modes QFT $\mathcal{F}_{\mathbb{R}^m} = \mathcal{F}_{\mathbb{R}} \otimes \cdots \otimes \mathcal{F}_{\mathbb{R}} = U(3\pi/4)^{\otimes m}$ is Gaussian. Our next result shows that *all* other normalizer gates can also be implemented via Gaussian unitaries and vice-versa.

Theorem E.1 (Bosonic Gaussian = Normalizer over $\mathbb{R}^m$). Let $\mathcal{H}_{\mathbb{R}^m} = \mathcal{H}_{\text{osc}}^{\otimes m}$ be the Hilbert space of m harmonic oscillators. Then, any m mode Gaussian unitary U can be approximated up to error ε by circuit $\tilde{U}$ of $O(\text{polylog}\,\frac{1}{\varepsilon})$ two-mode normalizer gates over $\mathbb{R}^m$. Moreover, any normalizer circuit V over $\mathbb{R}^m$ can be approximated up to error ε by a circuit $\tilde{V}$ of $O(\text{polylog}\,\frac{1}{\varepsilon})$ two-mode Gaussian unitaries[a].

[a] These error bounds are the standard ones that come from quantum gate synthesis algorithms [278]

To prove the result, we give an explicit classical algorithm that outputs classical descriptions for $\tilde{U}$, $\tilde{V}$. This classical algorithm is *efficient* if certain maps that describe the action of U V on Pauli operators–see (E.4) below—can be efficiently computed.

Proof. First, note that both normalizer circuits and Gaussian unitaries send Pauli operators (over $\mathbb{R}^m$) to Pauli operators under conjugation: for the former, this follows from theorem 4.2 (the proof of which holds for any locally compact abelian G); for the later, it follows from the fact that Gaussian unitaries send shift operators to shift operators [215]. We prove our result by showing that *any* unitary $\mathcal{C}$ that sends Paulis to Paulis can be efficiently implemented by circuits U, V built of Gaussian unitaries and normalizer gates, respectively. The results follows by choosing the circuit $\mathcal{C}$ to be either Gaussian or normalizer.

Our next step is to write the action of $\mathcal{C}$ on Pauli operators as

$$\mathcal{C} Z(p) X(x) \mathcal{C}^\dagger = \gamma^{\mathcal{C}}(p,x) Z(\alpha_Z^{\mathcal{C}}(p,x)) X(\alpha_X^{\mathcal{C}}(p,x)) \tag{E.4}$$

for some functions $\gamma^{\mathcal{C}}, \alpha^{\mathcal{C}}$. To prove our claim we will assume that $\gamma^{\mathcal{C}}, \alpha^{\mathcal{C}}$ can be computed efficiently for any target $\mathcal{C}$ that we may want to implement. This choice will lead to efficient algorithms to decompose $\mathcal{C}$ in terms of simpler Gaussian and normalizer gates. We highlight that if we drop this assumption then our algorithm becomes inefficient but still outputs poly-size approximations for $\mathcal{C}$: the latter claim, though weaker, is already enough to show that normalizer circuits over $\mathbb{R}^m$ and Gaussian unitaries define the same families of unitary gates.

Now, note that, because $\mathcal{C}\sigma_1\sigma_2\mathcal{C}^\dagger = (\mathcal{C}\sigma_1\mathcal{C}^\dagger)(\mathcal{C}\sigma_2\mathcal{C}^\dagger)$ for any two Pauli operators $\sigma_1 := Z(p_1)X(x_1), \sigma_2 := Z(p_2)X(x_2)$, the map $\alpha^{\mathcal{C}}$ needs to be a group automorphism of $\mathbb{R}^m \times \mathbb{R}^m$ and, necessarily, a continuous one, because U is continuous. Hence, $\alpha^{\mathcal{C}}$ has a matrix representation $A^{\mathcal{C}}$ (lemma 2.7) whose entries $A^{\mathcal{C}}(e_i, e_j)$, where we denote $e_i := (0, \ldots, 0, 1_i, 0, \ldots, 0)$, can be efficiently inferred by evaluating $\alpha^{\mathcal{C}}$. Furthermore, because of the identity

$$\mathcal{C}\sigma_1\sigma_2\mathcal{C}^\dagger = \mathcal{C}\sigma_2\sigma_1 e^{2\pi i[(p_1,x_1),(p_2,x_2)]}\,\mathcal{C}^\dagger = (\mathcal{C}\sigma_2\mathcal{C}^\dagger)(\mathcal{C}\sigma_1\mathcal{C}^\dagger) e^{2\pi i[A^{\mathcal{C}}(p_1,x_1),A^{\mathcal{C}}(p_2,x_2)]} \tag{E.5}$$

[1] Recall that Pauli X gates can be implemented by a normalizer circuit with 3 normalizer gates due to lemma 4.1 (the proof of which applies to any locally compact abelian group).

for any $x_1, p_1, x_2, p_2 \in \mathbb{R}^m$, it follows that $A^{\mathcal{C}}$ must be a *symplectic* matrix, i.e., it must preserve the symplectic product defined as $[(p_1, x_1), (p_2, x_2)] = p_1 \cdot x_2 - x_1 \cdot p_2$.

We will now show that $\mathcal{C}$ can be efficiently approximated both by Gaussian-unitary and normalizer-gate circuits. For this, we use the following lemma, which says that if we can find (Gaussian-gate or normalizer-gate) circuits that act like $\mathcal{C}$ on Pauli operator labels (i.e., on *phase space* in the CV QIP jargon) then we are done.

Lemma E.1 (**Phase space actions**)**.** Let U be any unitary that sends Pauli operators to Pauli operators and whose action on (phase space) Pauli operator labels (E.4) is identical to that of $\mathcal{C}$: i.e., such that $\alpha^U = \alpha^{\mathcal{C}}$ but γ^U might differ from $\gamma^{\mathcal{C}}$. Then, U coincides with $\mathcal{C}$ up to a correction term $W := Z_{\mathbb{R}^m}(b)X_{\mathbb{R}^m}(a)$ that is a Pauli operator over $\mathbb{R}^m$. Moreover, if α^U, γ^U and their inverses are given to us as oracles, then W can be efficiently determined classically.

Note that in lemma E.1 an oracle could be any arbitrary poly-size circuit that approximates α^U, γ^U. In particular, a valid oracle could be given by a circuit $\tilde{U}$ of $O(\text{polylog}\,\frac{1}{\varepsilon})$ (not necessarily nearest neighbor) k-mode gates (with constant k) that fulfill (E.4) and implements U up to error ε, as studied below: from such a description, one can efficiently infer classical boolean circuits that compute these maps.

Lemma E.1 is important because it tells us that if we can find an efficient circuit U that implements the action of $\mathcal{C}$ in phase space then we can also implement $\mathcal{C}$ efficiently by performing a Pauli correction. Since the latter can be implemented with either Gaussian unitaries or normalizer gates, we can reduce our original problem to finding good Gaussian and normalizer approximations of $\mathcal{C}$ in phase space.

Proof of lemma E.1. For $\mathcal{C}' := \mathcal{C}U^\dagger$ and any $\alpha, \beta \in \mathbb{R}^m$, we consider the stabilizer groups

$$\mathcal{S}_Z = \{\overline{\mathrm{e}^{2\pi\mathrm{i}\alpha\cdot p}} Z_{\mathbb{R}^m}(p), p \in \mathbb{R}^m\}, \qquad \mathcal{S}_X = \{\mathrm{e}^{2\pi\mathrm{i}\beta\cdot x} X_{\mathbb{R}^m}(x), x \in \mathbb{R}^m\},$$

which are easily seen to uniquely stabilize, respectively, the states $|\alpha\rangle_X$ and $|\beta\rangle_P$ (cf. proof of lemma 4.2) . By assumption, the gate $\mathcal{C}'$ sends Paulis to Paulis (via conjugation) with $\alpha_{\mathcal{C}'} = \mathrm{id}$, being the identity. Hence, $\mathcal{C}'$ transforms the stabilizer groups as $\mathcal{S}_Z \to \mathcal{S}_{Z'}, \mathcal{S}_X \to \mathcal{S}_{X'}$ where

$$\mathcal{S}'_Z = \{\gamma^{\mathcal{C}'}(p,0)\overline{\mathrm{e}^{2\pi\mathrm{i}\alpha\cdot p}} Z_{\mathbb{R}^m}(p), p \in \mathbb{R}^m\}, \qquad \mathcal{S}_X = \{\gamma^{\mathcal{C}'}(0,x)\mathrm{e}^{2\pi\mathrm{i}\beta\cdot x} X_{\mathbb{R}^m}(x), x \in \mathbb{R}^m\}.$$

Moreover, using that $Z_{\mathbb{R}^m}(p_1 + p_2) = Z_{\mathbb{R}^m}(p_1)Z_{\mathbb{R}^m}(p_2)$, $X_{\mathbb{R}^m}(x_1 + x_2) = X_{\mathbb{R}^m}(x_1)X_{\mathbb{R}^m}(x_2)$ it follows that $\gamma^{\mathcal{C}'}$ restricts to a character of $\mathbb{R}^m$ on the subgroups $\mathbb{R}^m \times \{0\}$ and $\{0\} \times \mathbb{R}^m$. Therefore, there exist $\alpha', \beta' \in \mathbb{R}^m$ such that $\gamma^{\mathcal{C}'}(p,0) = \overline{\mathrm{e}^{2\pi\mathrm{i}\alpha'\cdot p}}$ and $\gamma^{\mathcal{C}'}(0,x) = \mathrm{e}^{2\pi\mathrm{i}\beta'\cdot x}$. From these equations, we derive that the action of $\mathcal{C}'$ is

$$\mathcal{C}'|\alpha\rangle_X = \varphi(\alpha)|\alpha + \alpha'\rangle_X, \qquad \mathcal{C}'|\beta\rangle_P = \vartheta(\beta)|\beta + \beta'\rangle_P, \tag{E.6}$$

where the additional terms $\varphi(\alpha)$, $\vartheta(\beta)$ are introduced since stabilizer states are only well-defined up to an (arbitrary) phase. Finally, we show that the complex function φ (resp. ϑ) is proportional (as a vector), to the character function $\chi_{(-\beta')}$ (resp. $\chi_{\alpha'}$) of $\mathbb{R}^m$: for φ this follows from the identity

$$\mathcal{C}'|\beta\rangle_P \overset{(E.1)}{=} \int_{\mathbb{R}^m} \mathrm{d}\alpha\, \overline{\mathrm{e}^{2\pi\mathrm{i}\beta\cdot\alpha}}\varphi(\alpha)|\alpha + \alpha'\rangle_X \overset{(E.6,E.1)}{=} \vartheta(\beta)\int_{\mathbb{R}^m} \mathrm{d}\alpha''\, \overline{\mathrm{e}^{2\pi\mathrm{i}(\beta+\beta')\cdot\alpha''}}|\alpha''\rangle;$$

the proof for φ is analogous. These last equations fully determine $\mathcal{C}'$ as a unitary gate to be of form $\mathcal{C}' \propto W = Z_{\mathbb{R}^m}(b)X_{\mathbb{R}^m}(a)$ with $a := \alpha', b := -\beta'$, up to a (neglectable) global phase.

Finally, note that W can be efficiently identified because that the vectors a, b can be inferred by evaluating the function $\gamma_{\mathcal{C}'} = (\gamma_{\mathcal{C}} \circ \alpha^{U^{-1}})(\overline{\gamma_U})$ on basis vectors e_i, which can be done efficiently by computing the oracles that we are given. □

Next, we recall that for any (necessarily invertible) symplectic matrix $A^{\mathcal{C}}$, there exist efficient classical algorithms [279, 215] that can be used to find a circuit of *Gaussian* unitaries U such that $\alpha^U = \alpha^{\mathcal{C}}$; here γ^U might differ from $\gamma^{\mathcal{C}}$. Hence, adding a Pauli correction, U provides an exact Gaussian implementation of $\mathcal{C}$. Moreover, this result still holds if we restrict our Gaussian gates to belong to simple gate sets: first, U can be written as a circuit of $O(m)$ Gaussian single-mode squeezers, which are gates of the form

$$S(r) := \exp\left(\mathrm{i}r\left(\hat{X}\hat{P} + \hat{P}\hat{X}\right)\right) \tag{E.7}$$

where r is a real parameter and two global Gaussian passive[2] transformations[3]; furthermore, techniques from [281] let us approximate the latter global operations to any error ε by Gaussian circuits comprising $O(\text{polylog}\,\frac{1}{\varepsilon})$ (non nearest-neighbor) two-mode beamsplitters

$$U_{BS} = \exp\left(i\frac{\pi}{4}\left(\hat{X}_1\hat{X}_2 + \hat{P}_1\hat{P}_2\right)\right) \tag{E.8}$$

and single-mode QFTs (ie. phaseshifters of the form $U(3\pi/4)$). Combining these two facts, we obtain a poly-size circuit of two-local[4] Gaussian gates $\tilde{U}$ approximating U.

We complete our proof showing that all gates in the Gaussian circuit $\tilde{U}$ are normalizer gates over $\mathbb{R}^m$ up to a Pauli correction. Since, we have already discussed that all QFTs and Pauli gates are normalizer, we just need to show this for $S(r)$ and U_{BS}. Moreover, due to lemma E.1, it is enough to show that the action in phase space of these gates coincides with that of some normalizer gates. Writing our phase space points as $(p_1, p_2, x_1, x_1) \in \mathbb{R}^{2m}$, the symplectic matrices associated to these gates are [143]

$$A^{S(r)} = \begin{pmatrix} \mathrm{e}^{-r} & 0 \\ 0 & \mathrm{e}^{r} \end{pmatrix}, \qquad A^{U_{BS}} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & 1 \end{pmatrix}. \tag{E.9}$$

It follows that, up to a Pauli, $S(r)$, U_{BS} can be implemented as the one-mode and two-mode normalizer automorphism gates

$$U_{\alpha_{S(r)}}|x\rangle = \frac{1}{\sqrt{\mathrm{e}^r}}|\mathrm{e}^r x\rangle, \quad x \in \mathbb{R}, \qquad U_{\alpha_{BS}}|x_1, x_2\rangle = \left|\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}(x_1, x_2)\right\rangle, \quad (x_1, x_2) \in \mathbb{R}^2$$

which implement the classical maps $\alpha_{S(r)} : x \to \mathrm{e}^r x$ and $(x_1, x_2) \to \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}(x_1, x_2)$. These are easily seen to be continuous group automorphisms of $\mathbb{R}$ and $\mathbb{R}^2$ whose actions in phase space are also described by the matrices (E.9): the proof of the latter fact is analogous to the one of eq. (4.39) in lemma 4.5.

Discussion: renormalization factor for automorphism gates

Finally, we mention that there is a subtle (albeit negligible) difference between the automorphism gates over $\mathbb{R}^m$ compared to those in chapters 1-4: unlike earlier, for the $\mathbb{R}^m$, a normalization factor is needed in the definition of automorphism gate in order that they are always

[2] Passive unitaries are those that preserve the energy eigenspaces of the global m-mode oscillator hamiltonian.

[3] In the standard approaches [279, 215], this step is implemented by computing the so-called Euler decomposition of a $2m \times 2m$ symplectic matrix [280] using efficient classical algorithms for the Singular Value Decomposition.

[4] Here we used non nearest neighbor interactions. Of course, these could be decomposed into linear-size circuits of two-mode nearest neighbor ones using two-mode swap operations (which are both Gaussian and normalizer).

unitary. This is exemplified in our formulas for $U_{\alpha_{S(r)}}$ and $U_{\alpha_{BS}}$ above. In general, for any automorphism gate U_α, a re-scaling by a factor of $1/\sqrt{|\det A|}$ is needed because of the well-known change-of-variable formula used in integration by substitution

$$\int_{\mathbb{R}^m} \mathrm{d}x|\psi^2(x)| = \int_{\mathbb{R}^m} \mathrm{d}y|\det A|\,|\psi^2(\alpha(y))|$$

where $\det A$ is the determinant of a matrix representation of α.

More generally, for an arbitrary locally compact abelian group G, the re-scaling factor needed for an automorphism gate U_α (with respect to earlier chapters) is $1/\sqrt{\mathrm{mod}\alpha}$, where $\mathrm{mod}\,\alpha$ denotes the so-called *module function* of G [189]. The latter equals $|\det A|$ for $G = \mathbb{R}^m$ and is defined via the analogous integration-by-substitution formula:

$$\int_G \mathrm{d}g|\psi^2(g)| = \int_G \mathrm{d}h\,\mathrm{mod}\alpha\,|\psi^2(\alpha(h))|.$$

The existence of this function follows from properties of the Haar measure of an LCA group and guarantees that re-scaled automorphism gates do not increase volumes locally, at the level of group-element labels, hence, preserve inner products at the quantum-state level: it follows that they are always *unitary gates*. On the other hand, this re-normalization was not needed for the groups $\mathbb{Z}_{D_1} \times \ldots \times \mathbb{Z}_{D_a} \times \mathbb{Z}^b \times \mathbb{T}^b$ considered earlier because their associated module function is always trivial (intuitively, because continuous invertible endomorphisms cannot locally increase/shrink volumes on toruses and discrete groups).

Finally, we highlight that the presence of these normalization factors is meaningless from a stabilizer formalism perspective. This is because $\mathrm{mod}\,\alpha$ is a group homomorphism $\mathrm{Aut}(G) \to \mathbb{R}^+$, which readily implies $\mathrm{mod}\,\alpha^{-1} = 1/\,\mathrm{mod}\,\alpha$ and $U_{\alpha^{-1}} = U_\alpha^\dagger$. As a result, these factors always get canceled when automorphism gates act by conjugation on the Heisenberg picture.

□

Bibliography

[1] D. Gottesman, *Stabilizer Codes and Quantum Error Correction*. PhD thesis, California Institute of Technology, 1997. quant-ph/9705052v1.

[2] 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. quant-ph/9807006v1.

[3] 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. quant-ph/9802007v1.

[4] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Sci. Statist. Comput. 26* (1997), quant-ph/9508027.

[5] S. Jordan, "Quantum Algorithm Zoo." `http://math.nist.gov/quantum/zoo/`.

[6] M. Mosca, "Quantum algorithms," in *Encyclopedia of Complexity and Systems Science*. Springer, 2009. arXiv:0808.0369 [quant-ph].

[7] A. M. Childs and W. van Dam, "Quantum algorithms for algebraic problems," *Rev. Mod. Phys.* **82** (2010), arXiv:0812.0380v1 [quant-ph].

[8] D. Bacon and W. van Dam, "Recent progress in quantum algorithms," *Commun. ACM* **53** no. 2, (2010).

[9] 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].

[10] J. Smith and M. Mosca, "Algorithms for quantum computers," in *Handbook of Natural Computing*. Springer, 2012.

[11] A. M. Childs, "Quantum algorithms: Equation solving by simulation," *Nat Phys* **5** no. 12, (12, 2009). Preprint at http://www.cs.umd.edu/ amchilds/papers/linear.pdf.

[12] S. Aaronson, "Read the fine print," *Nat Phys* **11** no. 4, (04, 2015). Preprint at http://www.scottaaronson.com/papers/qml.pdf.

[13] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[14] S. Aaronson and A. Ambainis, "The need for structure in quantum speedups," *Theory of Computing* **10** no. 6, (2014), arXiv:0911.0996v3 [quant-ph].

[15] P. W. Shor, "Progress in quantum algorithms," *Quantum Information Processing* **3** no. 1-5, (2004).

[16] S. Aaronson, "NP-complete problems and physical reality," *SIGACT News* **36** (2005), quant-ph/0502072.

[17] Y. Manin, "Computable and uncomputable," *Sovetskoye Radio* (1980). (in Russian).

[18] Y. I. Manin, "Classical computing, quantum computing, and shor's factoring algorithm," *Séminaire Bourbaki* **41** (1998-1999).

[19] R. Feynman, "Simulating physics with computers," *International Journal of Theoretical Physics* **21** no. 6-7, (1982).

[20] B. Yoshida, "Feasibility of self-correcting quantum memory and thermal stability of topological order," *Annals of Physics* **326** no. 10, (2011), arXiv:1103.1885 [quant-ph].

[21] B. J. Brown, D. Loss, J. K. Pachos, C. N. Self, and J. R. Wootton, "Quantum Memories at Finite Temperature," *arXiv preprint* (2014), arXiv:1411.6643 [quant-ph].

[22] B. M. Terhal, "Quantum error correction for quantum memories," *Rev. Mod. Phys.* **87** (Apr, 2015), arXiv:1302.3428 [quant-ph].

[23] G. Crabtree, L. Greene, and P. Johnson, "Celebrating 100 years of superconductivity: special issue on the iron-based superconductors," *Reports on Progress in Physics* **74** no. 12, (2011).

[24] E. Knill, "Non-binary unitary error bases and quantum codes," tech. rep., Los Alamos National Laboratory, 1996. quant-ph/9608048.

[25] L. G. Valiant, "Quantum circuits that can be simulated classically in polynomial time," *SIAM J. Comput.* **31** no. 4, (2002).

[26] E. Knill, "Fermionic Linear Optics and Matchgates," 2001. arXiv:quant-ph/0108033.

[27] B. M. Terhal and D. P. DiVincenzo, "Classical simulation of noninteracting-fermion quantum circuits," *Phys. Rev. A* **65** (2002), quant-ph/0108010.

[28] 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].

[29] S. Lloyd and J.-J. E. Slotine, "Analog quantum error correction," *Phys. Rev. Lett.* **80** (1998), quant-ph/9711021.

[30] S. Lloyd and S. L. Braunstein, "Quantum computation over continuous variables," *Phys. Rev. Lett.* **82** (1999), quant-ph/9810082.

[31] S. D. Bartlett, B. C. Sanders, S. L. Braunstein, and K. Nemoto, "Efficient classical simulation of continuous variable quantum information processes," *Phys. Rev. Lett.* **88** (2002), quant-ph/0109047.

[32] S. D. Bartlett and B. C. Sanders, "Efficient classical simulation of optical quantum information circuits," *Phys. Rev. Lett.* **89** (2002), quant-ph/0204065.

[33] E. Knill and R. Laflamme, "Power of one bit of quantum information," *Phys. Rev. Lett.* **81** (1998), quant-ph/9802037.

[34] D. J. Shepherd, *Quantum Complexity: restrictions on algorithms and architectures*. PhD thesis, 2010. 1005.1425 [quant-ph].

[35] 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].

[36] 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].

[37] X. Ni and M. Van den Nest, "Commuting quantum circuits: efficiently classical simulations versus hardness results," *Quantum Info. Comput.* **13** no. 1&2, (2013), arXiv:1204.4570 [quant-ph].

[38] P. O. Boykin, T. Mor, M. Pulver, V. Roychowdhury, and F. Vatan, "On universal and fault-tolerant quantum computing: A novel basis and a new constructive proof of universality for shor's basis," in *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, FOCS '99. IEEE Computer Society, 1999. quant-ph/9906054v1.

[39] P. O. Boykin, T. Mor, M. Pulver, V. Roychowdhury, and F. Vatan, "A new universal and fault-tolerant quantum basis," *Information Processing Letters* **75** no. 3, (2000).

[40] H. J. Briegel and R. Raussendorf, "Persistent entanglement in arrays of interacting particles," *Phys. Rev. Lett.* **86** (2001), quant-ph/0004051v2.

[41] M. Hein, W. Dür, J. Eisert, R. Raussendorf, M. Van den Nest, and H. J. Briegel, "Entanglement in graph states and its applications," in *Quantum Computers, Algorithms and Chaos*, International School of Physics Enrico Fermi. IOS Press, 2006. quant-ph/0602096.

[42] R. Raussendorf and H. J. Briegel, "A one-way quantum computer," *Phys. Rev. Lett.* **86** (2001).

[43] S. Bravyi and A. Kitaev, "Universal quantum computation with ideal Clifford gates and noisy ancillas," *Phys. Rev. A* **71** (2005), quant-ph/0403025.

[44] V. Veitch, C. Ferrie, David, 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].

[45] M. Howard, J. Wallman, V. Veitch, and J. Emerson, "Contextuality supplies the 'magic' for quantum computation," *Nature* **510** no. 7505, (June, 2014), arXiv:1401.4174 [quant-ph].

[46] V. Veitch, S. A. H. Mousavian, D. Gottesman, and J. Emerson, "The resource theory of stabilizer quantum computation," *New Journal of Physics* **16** no. 1, (2014), arXiv:1307.7171 [quant-ph].

[47] N. Delfosse, P. Allard Guerin, J. Bian, and R. Raussendorf, "Wigner function negativity and contextuality in quantum computation on rebits," *Phys. Rev. X* **5** (Apr, 2015), arXiv:1409.5170 [quant-ph].

[48] R. Raussendorf, N. Delfosse, D. E. Browne, C. Okay, and J. Bermejo-Vega, "Contextuality and wigner function negativity in qubit quantum computation," *arXiv preprint* (2015).

[49] A. Y. Kitaev, "Quantum measurements and the Abelian stabilizer problem," *arXiv preprint* (1995). arXiv:quant-ph/9511026v1.

[50] 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).

[51] D. R. Simon, "On the power of quantum computation," *SIAM Journal on Computing* **26** (1994). http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.51.5477&rep=rep1&type=pdf.

[52] D. Boneh and R. Lipton, "Quantum cryptanalysis of hidden linear functions," in *Advances in Cryptology — CRYPT0' 95*, D. Coppersmith, ed., vol. 963 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1995.

[53] D. Grigoriev, "Testing shift-equivalence of polynomials by deterministic, probabilistic and quantum machines," *Theor. Comput. Sci.* **180** no. 1-2, (1997).

[54] A. Y. Kitaev, "Quantum computations: algorithms and error correction," *Russian Mathematical Surveys* **52** no. 6, (1997).

[55] 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. quant-ph/9704027.

[56] P. Høyer, "Conjugated operators in quantum algorithms," *Phys. Rev. A* **59** (1999).

[57] 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. quant-ph/9903071.

[58] I. Damgård, "QIP note: on the quantum Fourier transform and applications." http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.116.2654&rep=rep1&type=pdf.

[59] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM* **21** (1978).

[60] W. Diffie and M. Hellman, "New directions in cryptography," *Information Theory, IEEE Transactions on* **22** no. 6, (1976).

[61] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*. CRC Press, 1st ed., 1996.

[62] J. A. Buchmann, *Introduction to Cryptography*. Springer, 1st ed., 2000.

[63] J. Bermejo-Vega and M. Van Den Nest, "Classical simulations of Abelian-group normalizer circuits with intermediate measurements," *Quantum Info. Comput.* **14** no. 3-4, (2014), arXiv:1210.3637 [quant-ph].

[64] J. Bermejo-Vega, C. Y.-Y. Lin, and M. Van den Nest, "Normalizer circuits and a Gottesman-Knill theorem for infinite-dimensional systems," *Quantum Information and Computation* **16** no. 5-6, (2016), arXiv:1409.3208 [quant-ph].

[65] D. L. Zhou, B. Zeng, Z. Xu, and C. P. Sun, "Quantum computation based on d -level cluster state," *Phys. Rev. A* **68** (2003), quant-ph/0304054.

[66] D. Schlingemann, "Cluster states, algorithms and graphs," *Quantum Info. Comput.* **4** no. 4, (2004), quant-ph/0305170.

[67] H. Anwar, E. T. Campbell, and D. E. Browne, "Qutrit magic state distillation," *New Journal of Physics* **14** no. 6, (2012), arXiv:1202.2326 [quant-ph].

[68] E. T. Campbell, H. Anwar, and D. E. Browne, "Magic-state distillation in all prime dimensions using quantum reed-muller codes," *Phys. Rev. X* **2** (2012), arXiv:1205.3104 [quant-ph].

[69] H. Anwar, B. J. Brown, E. T. Campbell, and D. E. Browne, "Fast decoders for qudit topological codes," *New Journal of Physics* **16** no. 6, (2014), arXiv:1311.4895 [quant-ph].

[70] E. T. Campbell, "Enhanced fault-tolerant quantum computing in d-level systems," *Phys. Rev. Lett.* **113** (Dec, 2014), arXiv:1406.3055 [quant-ph].

[71] F. H. E. Watson, E. T. Campbell, H. Anwar, and D. E. Browne, "Qudit color codes and gauge color codes in all spatial dimensions," *Phys. Rev. A* **92** (Aug, 2015), arXiv:1503.08800 [quant-ph].

[72] M. Hillery, V. Bužek, and A. Berthiaume, "Quantum secret sharing," *Phys. Rev. A* **59** (1999), quant-ph/9806063.

[73] R. Cleve, D. Gottesman, and H.-K. Lo, "How to share a quantum secret," *Phys. Rev. Lett.* **83** (1999), quant-ph/9901025.

[74] D. Gottesman, "Theory of quantum secret sharing," *Phys. Rev. A* **61** (2000).

[75] A. Y. Kitaev, "Fault-tolerant quantum computation by anyons," *Annals of Physics* **303** no. 1, (2003), quant-ph/9707021.

[76] H. Bombín and M. A. Martín-Delgado, "Homological error correction: Classical and quantum codes," *Journal of Mathematical Physics* **48** (2007), quant-ph/0605094.

[77] S. S. Bullock and G. K. Brennen, "Qudit surface codes and gauge theory with finite cyclic groups," *Journal of Physics A: Mathematical and Theoretical* **40** no. 13, (2007), quant-ph/0609070.

[78] G. Duclos-Cianci and D. Poulin, "Kitaev's $\mathcal{F}_d$-code threshold estimates," *Phys. Rev. A* **87** (Jun, 2013), arXiv:1302.3638 [quant-ph].

[79] 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].

[80] M. Gutiérrez, L. Svec, A. Vargo, and K. R. Brown, "Approximation of realistic errors by clifford channels and pauli measurements," *Phys. Rev. A* **87** (Mar, 2013).

[81] D. P. DiVincenzo and P. Aliferis, "Effective fault-tolerant quantum computation with slow measurements," *Phys. Rev. Lett.* **98** (2007), quant-ph/0607047.

[82] A. M. Steane, "Overhead and noise threshold of fault-tolerant quantum error correction," *Phys. Rev. A* **68** (2003), quant-ph/0207119.

[83] E. Knill, "Quantum computing with realistically noisy devices," *Nature* **434** (2005), arXiv:quant-ph/0410199.

[84] A. W. Cross, *Fault-tolerant Quantum Computer Architectures Using Hierarchies of Quantum Error-correcting Codes*. PhD thesis, Cambridge, MA, USA, 2008. AAI0820521.

[85] P. Aliferis, D. Gottesman, and J. Preskill, "Quantum accuracy threshold for concatenated distance-3 codes," *Quantum Info. Comput.* **6** no. 2, (Mar., 2006).

[86] A. W. Cross, D. P. Divincenzo, and B. M. Terhal, "A comparative code study for quantum fault tolerance," *Quantum Info. Comput.* **9** no. 7, (July, 2009).

[87] 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].

[88] I. Rigas, L. L. Sánchez-Soto, A. B. Klimov, J. Řeháček, and Z. Hradil, "Non-negative Wigner functions for orbital angular momentum states," *Phys. Rev. A* **81** (2010), arXiv:0909.1887 [quant-ph].

[89] A. Kitaev, "Protected qubit based on a superconducting current mirror," *arXiv preprint* (2006), cond-mat/0609441.

[90] P. Brooks, A. Kitaev, and J. Preskill, "Protected gates for superconducting qubits," *Phys. Rev. A* **87** (2013), arXiv:1302.4122 [quant-ph].

[91] L. Babai and E. Szemerédi, "On the complexity of matrix group problems I," in *Proceedings of the 25th Annual Symposium on Foundations of Computer Science, 1984*, SFCS '84. IEEE Computer Society, 1984.

[92] V. Shoup, *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, 2nd ed., 2008.

[93] J. Bermejo-Vega, C. Yen-Yu Lin, and M. Van den Nest, "The computational power of normalizer circuits over black-box groups," *arXiv preprint* (2014), arXiv:1409.4800 [quant-ph].

[94] J. Proos and C. Zalka, "Shor's discrete logarithm quantum algorithm for elliptic curves," *Quantum Info. Comput.* **3** no. 4, (2003), quant-ph/0301141.

[95] P. Kaye, "Optimized quantum implementation of elliptic curve arithmetic over binary fields," *Quantum Info. Comput.* **5** no. 6, (2005), quant-ph/0407095.

[96] 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].

[97] M. Mosca, *Quantum computer algorithms*. PhD thesis, University of Oxford, 1999.

[98] K. K. H. Cheung and M. Mosca, "Decomposing finite Abelian groups," *Quantum Info. Comput.* **1** no. 3, (2001), cs/0101004.

[99] S. Hallgren, A. Russell, and A. Ta-Shma, "Normal subgroup reconstruction and quantum computation using group representations," in *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, STOC '00. ACM, New York, NY, USA, 2000. http://www.cs.tau.ac.il/~amnon/Papers/HRT.stoc00.pdf.

[100] 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.

[101] 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.

[102] O. Regev, "A subexponential time algorithm for the dihedral hidden subgroup problem with polynomial space," arXiv:quant-ph/0406151.

[103] G. Kuperberg, "Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem," in *Proceedings of TQC13*. 2013. arXiv:1112.3333 [quant-ph].

[104] M. Roetteler and T. Beth, "Polynomial-time solution to the hidden subgroup problem for a class of non-abelian groups," arXiv:quant-ph/9812070.

[105] G. Ivanyos, F. Magniez, and M. Santha, "Efficient quantum algorithms for some instances of the non-abelian hidden subgroup problem," arXiv:quant-ph/0102014.

[106] 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.

[107] Y. Inui and F. Le Gall, "Efficient quantum algorithms for the hidden subgroup problem over a class of semi-direct product groups," *Quantum Info. Comput.* **7** no. 5/6, (2007), arXiv:0412033 [quant-ph].

[108] 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].

[109] 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.

[110] 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.

[111] 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.

[112] 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].

[113] 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.

[114] D. Gavinsky, "Quantum solution to the hidden subgroup problem for poly-near-hamiltonian-groups," *Quantum Info. Comput.* **4** (2004).

[115] 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.

[116] A. Denney, C. Moore, and A. Russell, "Finding conjugate stabilizer subgroups in psl(2;q) and related groups," *Quantum Info. Comput.* **10** no. 3, (2010), arXiv:0809.2445 [quant-ph].

[117] 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].

[118] C. Lomont, "The hidden subgroup problem - review and open problems," *arXiv* (2004), arXiv:quant-ph/0411037v1.

[119] A. Childs, *Lecture Notes on Quantum Algorithms.* University of Waterloo, 2011. Published online.

[120] M. Ettinger and P. Høyer, "A quantum observable for the graph isomorphism problem," tech. rep., Los Alamos National Laboratory, 1999. quant-ph/9901029. preprint number LA-UR-99-179.

[121] O. Regev, "Quantum computation and lattice problems," *SIAM J. Comput.* **33** no. 3, (Mar., 2004), cs/0304005.

[122] D. Gavinsky, "Quantum solution to the hidden subgroup problem for poly-near-hamiltonian groups," *Quantum Info. Comput.* **4** no. 3, (May, 2004). http://users.math.cas.cz/~gavinsky/papers/HSP-Near-Ham.pdf.

[123] G. Ivanyos, L. Sanselme, and M. Santha, "An efficient quantum algorithm for the hidden subgroup problem in nil-2 groups," in *Proceedings of the 8th Latin American Conference on Theoretical Informatics*, LATIN'08. Springer-Verlag, 2008. arXiv:0707.1260 [quant-ph].

[124] J. Bermejo-Vega and K. C. Zatloukal, "Abelian Hypergroups and Quantum Computation," *arXiv preprint* (2015), arXiv:1509.05806 [quant-ph].

[125] M. Amini, M. Kalantar, and M. M. Roozbehani, "Hidden sub-hypergroup problem," *arXiv preprint* (2006), quant-ph/0609220.

[126] M. Amini, H. Myrnouri, M. Roozbahani, and M. Kalantar, *Fourier Transform on Group-Like Structures and Applications.* INTECH Open Access Publisher, 2011.

[127] E. de Klerk and D. Pasechnik, *On Semidefinite Programming Relaxations of Association Schemes With Application to Combinatorial Optimization Problems*. No. 2009-54. 2009. https://pure.uvt.nl/portal/files/1102147/2009-54.pdf.

[128] M. Anjos and J. Lasserre, *Handbook on Semidefinite, Conic and Polynomial Optimization*. International Series in Operations Research & Management Science. Springer, 2011. https://books.google.de/books?id=CgxAx7Ti4-kC.

[129] P. Corsini and V. Leoreanu, *Applications of Hyperstructure Theory*. Advances in Mathematics (Kluwer Academic Publishers), V. 5. Springer, 2003. https://books.google.de/books?id=99Ve3Xp2ajwC.

[130] N. Wildberger, "Finite commutative hypergroups and applications from group theory to conformal field theory," in *Proc. of Applications of Hypergroups and Related Measure Algebras 1993*. AMS, 1995.

[131] A. Kitaev, "Anyons in an exactly solved model and beyond," *Annals of Physics* **321** no. 1, (2006), cond-mat/0506438.

[132] M. E. Beverland, O. Buerschaper, R. König, F. Pastawski, J. Preskill, and S. Sijher, "Protected gates for topological quantum field theories," *arXiv preprint* (2014), arXiv:1409.3898.

[133] S. Aaronson and D. Gottesman, "Improved simulation of stabilizer circuits," *Phys. Rev. A* **70** (2004), quant-ph/0406196.

[134] M. Van den Nest, "Efficient classical simulations of quantum Fourier transforms and normalizer circuits over Abelian groups," *Quantum Info. Comput.* **13** no. 11-12, (2013), arXiv:1201.4867v1 [quant-ph].

[135] J. Dehaene and B. De Moor, "Clifford group, stabilizer states, and linear and quadratic operations over GF(2)," *Phys. Rev. A* **68** (2003), quant-ph/0304125v1.

[136] E. Hostens, J. Dehaene, and B. De Moor, "Stabilizer states and Clifford operations for systems of arbitrary dimensions and modular arithmetic," *Phys. Rev. A* **71** (2005), quant-ph/0408190v2.

[137] N. de Beaudrap, "A linearized stabilizer formalism for systems of finite dimension," *Quantum Info. Comput.* **13** no. 1-2, (2013), arXiv:1102.3354v3 [quant-ph].

[138] M. Van den Nest, "Classical simulation of quantum computation, the Gottesman-Knill theorem, and slightly beyond„" *Quantum Info. Comput.* **10** no. 3, (2010), arXiv:0811.0898 [quant-ph].

[139] R. Jozsa and M. Van Den Nest, "Classical simulation complexity of extended clifford circuits," *Quantum Info. Comput.* **14** no. 7&8, (2014), arXiv:1305.6190 [quant-ph].

[140] D. Gottesman, A. Kitaev, and J. Preskill, "Encoding a qubit in an oscillator," *Phys. Rev. A* **64** (2001), quant-ph/0008040.

[141] R. L. Barnes, "Stabilizer codes for continuous-variable quantum error correction," *arXiv preprint* (2004), quant-ph/0405064.

[142] S. L. Braunstein and P. van Loock, "Quantum information with continuous variables," *Rev. Mod. Phys.* **77** (2005), quant-ph/0410100.

[143] C. Weedbrook, S. Pirandola, R. García-Patrón, N. J. Cerf, T. C. Ralph, J. H. Shapiro, and S. Lloyd, "Gaussian quantum information," *Rev. Mod. Phys.* **84** (2012), arXiv:1110.3234 [quant-ph].

[144] 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].

[145] S. Clark, R. Jozsa, and N. Linden, "Generalized clifford groups and simulation of associated quantum circuits," *Quantum Info. Comput.* **8** no. 1, (2008), quant-ph/0701103.

[146] S. Bravyi, "Lagrangian representation for fermionic linear optics," *Quantum Info. Comput.* **5** no. 3, (2005), quant-ph/0404180.

[147] 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).

[148] 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].

[149] S. Bravyi and R. König, "Classical simulation of dissipative fermionic linear optics," *Quantum Info. Comput.* **12** no. 11-12, (2012), arXiv:1112.2184 [quant-ph].

[150] 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].

[151] A. Ambainis, L. J. Schulman, and U. Vazirani, "Computing with highly mixed states," *J. ACM* **53** no. 3, (2006), quant-ph/0003136.

[152] D. Poulin, R. Laflamme, G. J. Milburn, and J. P. Paz, "Testing integrability with a single bit of quantum information," *Phys. Rev. A* **68** (2003), quant-ph/0303042.

[153] D. Poulin, R. Blume-Kohout, R. Laflamme, and H. Ollivier, "Exponential speedup with a single bit of quantum information: Measuring the average fidelity decay," *Phys. Rev. Lett.* **92** (2004), quant-ph/0310038.

[154] D. Shepherd, "Computation with Unitaries and One Pure Qubit," 2006. arXiv:quant-ph/0608132v2.

[155] P. W. Shor and S. P. Jordan, "Estimating Jones polynomials is a complete problem for one clean qubit," *Quantum Info. Comput.* **8** no. 8, (2008), arXiv:0707.2831 [quant-ph].

[156] S. P. Jordan and P. Wocjan, "Estimating Jones and Homfly polynomials with one clean qubit," *Quantum Info. Comput.* **9** no. 3, (2009), arXiv:0807.4688 [quant-ph].

[157] 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].

[158] T. Morimae, K. Fujii, and J. F. Fitzsimons, "Hardness of classically simulating the one-clean-qubit model," *Phys. Rev. Lett.* **112** (2014), arXiv:1312.2496 [quant-ph].

[159] A. Mari and J. Eisert, "Positive wigner functions render classical simulation of quantum computation efficient," *Phys. Rev. Lett.* **109** (2012), arXiv:1208.3660 [quant-ph].

[160] 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].

[161] M. Van Den Nest, "Universal quantum computation with little entanglement," *Phys. Rev. Lett.* **110** (2012), arXiv:1204.3107 [quant-ph].

[162] 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), quant-ph/0201143.

[163] G. Vidal, "Efficient classical simulation of slightly entangled quantum computations," *Phys. Rev. Lett.* **91** (2003).

[164] B. M. Terhal and D. P. DiVincenzo, "Adaptive Quantum Computation, Constant Depth Quantum Circuits and Arthur-Merlin Games," *Quantum Info. Comput.* **4** no. 2, (2014), quant-ph/0205133.

[165] I. Markov and Y. Shi, "Simulating quantum computation by contracting tensor networks," *SIAM Journal on Computing* **38** no. 3, (2008), quant-ph/0511069.

[166] D. Aharonov, Z. Landau, and J. Makowsky, "The quantum FFT can be classically simulated," *arXiv* (2006), quant-ph/0611156v2.

[167] N. Yoran and A. J. Short, "Efficient classical simulation of the approximate quantum Fourier transform," *Phys. Rev. A* **76** (2007), quant-ph/0611241v1.

[168] D. E. Browne, "Efficient classical simulation of the quantum fourier transform," *New Journal of Physics* **9** no. 5, (2007), quant-ph/0612021.

[169] N. Yoran, "Efficiently contractable quantum circuits cannot produce much entanglement," 2008.

[170] M. Van den Nest, "Simulating quantum computers with probabilistic methods," *Quantum Info. Comput.* **11** no. 9-10, (2011), arXiv:0911.1624v3 [quant-ph].

[171] D. Stahlke, "Quantum interference as a resource for quantum speedup," *Phys. Rev. A* **90** (2014), arXiv:1305.2186 [quant-ph].

[172] S. P. Jordan, "Permutational quantum computing," *Quantum Info. Comput.* **10** no. 5, (2010), arXiv:0906.2508 [quant-ph].

[173] M. Schwarz and M. Van den Nest, "Simulating quantum circuits with sparse output distributions," *Electronic Colloquium on Computational Complexity* (2013), arXiv:1310.6749 [quant-ph].

[174] M. Van den Nest, W. Dür, R. Raussendorf, and H. J. Briegel, "Quantum algorithms for spin models and simulable gate sets for quantum computation," *Phys. Rev. A* **80** (Nov, 2009), arXiv:0805.1214 [quant-ph].

[175] S. L. Braunstein, "Error correction for continuous quantum variables," *Phys. Rev. Lett.* **80** (1998), quant-ph/9711049.

[176] J. F. Humphreys, *A course in group theory*. Oxford University Press, 1996.

[177] M. Atiyah, "Characters and cohomology of finite groups," *Publications Mathématiques de l'Institut des Hautes Études Scientifiques* **9** no. 1, (1961).

[178] G. Aruldhas, *Quantum Mechanics: 500 Problems with Solutions*. Prentice-Hall of India, 2010.

[179] F. Bruhat, "Distributions sur un groupe localement compact et applications à l etude des représentations des groupes p-adiques," *Bull. Soc. Math. France* **89** (1961).

[180] M. S. Osborne, "On the Schwartz-Bruhat space and the Paley-Wiener theorem for locally compact Abelian groups," *J. Funct. Anal.* **19** (1975).

[181] R. de la Madrid, "The role of the rigged hilbert space in quantum mechanics," *European Journal of Physics* **26** no. 2, (2005), quant-ph/0502053.

[182] 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.

[183] M. Gadella and F. Gómez, "A unified mathematical formalism for the Dirac formulation of quantum mechanics," *Foundations of Physics* **32** no. 6, (2002).

[184] M. Gadella, F. Gómez, and S. Wickramasekara, "Riggings of locally compact Abelian groups.," *J. Geom. Symmetry Phys.* **11** (2008).

[185] W. Rudin, *Fourier analysis on groups*. No. 12 in Interscience Tracts in Pure and Applied Mathematics. Interscience Publishers, 1962.

[186] K. H. Hofmann and S. A. Morris, *The Structure of Compact Groups*. No. 25 in de Gruyter Studies in Mathematics. Walter de Gruyter, 2006.

[187] A. V. Oppenheim, A. S. Willsky, and S. H. Nawab, *Signals &Amp; Systems (2Nd Ed.)*. Prentice-Hall, 1996.

[188] S. A. Morris, *Pontryagin Duality and the Structure of Locally Compact Abelian Groups*. Cambridge University Press, 1977.

[189] M. Stroppel, *Locally Compact Groups*. EMS Textbooks in Mathematics. European Mathematical Society, 2006.

[190] D. Dikranjan, "Introduction to topological groups." 2010.

[191] D. L. Armacost, *The structure of locally compact abelian groups*. M. Dekker New York, 1981.

[192] 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.

[193] D. Gross, "Hudson's theorem for finite-dimensional quantum systems," *Journal of Mathematical Physics* **47** no. 12, (2006), quant-ph/0602001.

[194] D. Gross, *Computational power of quantum many-body states and some results on discrete phase spaces*. PhD thesis, 2008.

[195] C. J. Moreno, *Advanced Analytic Number Theory: L-Functions*, vol. 15 of *Mathematical Surveys and Monographs*. American Mathematical Society, 2005.

[196] A. Prasad and M. K. Vemuri, "Decomposition of phase space and classification of Heisenberg groups," arXiv:0806.4064 [quant-ph].

[197] 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.

[198] R. P. Brent and P. Zimmermann, *Modern Computer Arithmetic*. Cambridge University Press, 2010.

[199] 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).

[200] M. F. Hurt and C. Waid, "A generalized inverse which gives all the integral solutions to a system of linear equations,".

[201] A. Storjohann, *Algorithms for Matrix Canonical Forms*. PhD thesis, University of Waterloo, 2000.

[202] V. Gheorghiu, "Standard form of qudit stabilizer groups," arXiv:1101.1519v1 [quant-ph].

[203] M. Van 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].

[204] M. A. Nielsen, M. J. Bremner, J. L. Dodd, A. M. Childs, and C. M. Dawson, "Universal simulation of hamiltonian dynamics for quantum systems with finite-dimensional state spaces," *Phys. Rev. A* **66** (2002), quant-ph/0109064v2.

[205] P. R. Kaye, R. Laflamme, and M. Mosca, *An Introduction to Quantum Computing*. Oxford University Press, 2007.

[206] N. C. Menicucci, "Fault-tolerant measurement-based quantum computing with continuous-variable cluster states," *Phys. Rev. Lett.* **112** (2014), arXiv:1310.7596 [quant-ph].

[207] J. Zhang and S. L. Braunstein, "Continuous-variable gaussian analog of cluster states," *Phys. Rev. A* **73** (2006).

[208] 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," *Phys. Rev. Lett.* **97** (2006), quant-ph/0605198. http://link.aps.org/doi/10.1103/PhysRevLett.97.110501.

[209] M. Gu, C. Weedbrook, N. C. Menicucci, T. C. Ralph, and P. van Loock, "Quantum computing with continuous-variable clusters," *Phys. Rev. A* **79** (2009), arXiv:0903.3233 [quant-ph].

[210] K. S. Gibbons, M. J. Hoffman, and W. K. Wootters, "Discrete phase space based on finite fields," quant-ph/0401155.

[211] N. D. Mermin, "Extreme quantum entanglement in a superposition of macroscopically distinct states," *Phys. Rev. Lett.* **65** (1990).

[212] V. Scarani, A. Acín, E. Schenck, and M. Aspelmeyer, "Nonlocality of cluster states of qubits," *Phys. Rev. A* **71** (2005), quant-ph/0405119.

[213] O. Gühne, G. Tóth, P. Hyllus, and H. J. Briegel, "Bell inequalities for graph states," *Phys. Rev. Lett.* **95** (2005), quant-ph/0410059.

[214] 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].

[215] P. Kok and B. W. Lovett, *Introduction to Optical Quantum Information Processing*. Cambridge University Press.

[216] 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].

[217] P. Hayden, D. Leung, P. W. Shor, and A. Winter, "Randomizing quantum states: Constructions and applications," *Communications in Mathematical Physics* **250** no. 2, (2004), quant-ph/0307104.

[218] 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].

[219] M. M. Deza and E. Deza, *Encyclopedia of Distances*. Springer, 2 ed., 2013.

[220] R. A. Mollin, *Advanced Number Theory with Applications*. Discrete Mathematics and its Applications. CRC Press, 2010.

[221] 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.

[222] S. Anders and H. J. Briegel, "Fast simulation of stabilizer circuits using a graph-state representation," *Phys. Rev. A* **73** (2006), quant-ph/0504117.

[223] 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).

[224] V. Arvind and N. Vinodchandran, "Solvable black-box group problems are low for pp," *Theoretical Computer Science* **180** (1997).

[225] 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.

[226] L. Babai, "Bounded round interactive proofs in finite groups," *SIAM J. Discret. Math.* **5** no. 1, (Feb., 1992).

[227] L. Babai, "Randomization in group algorithms: conceptual questions," in *Groups and Computation II*, vol. 28 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.* 1997.

[228] 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.

[229] J. Watrous, "Quantum algorithms for solvable groups," in *Proceedings of the 33rd ACM Symposium on Theory of Computing*. 2001.

[230] 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.

[231] 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.

[232] F. Le Gall, "An efficient quantum algorithm for some instances of the group isomorphism problem," in *Proceedings of STACS*. 2010. arXiv:1001.0608 [quant-ph].

[233] K. C. Zatloukal, "Classical and quantum algorithms for testing equivalence of group extensions," arXiv:1305.1327 [quant-ph].

[234] P. Sarvepalli and P. Wocjan, "Quantum algorithms for one-dimensional infrastructures," *Quantum Info. Comput.* **14** no. 1-2, (2014), arXiv:1106.6347 [quant-ph].

[235] F. Fontein and P. Wocjan, "Quantum Algorithm for Computing the Period Lattice of an Infrastructure," *arXiv* (2011), arXiv:1111.1348 [quant-ph].

[236] S. Hallgren, "Polynomial-time quantum algorithms for pell's equation and the principal ideal problem," *J. ACM* **54** no. 1, (2007).

[237] 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), quant-ph/0302134.

[238] 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.

[239] 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.

[240] A. M. Childs and G. Ivanyos, "Quantum computation of discrete logarithms in semigroups," arXiv:1310.6238 [quant-ph].

[241] U. B. Wim van Dam (HP, MSRI and G. S. (HP), "Efficient Quantum Algorithms for Estimating Gauss Sums," *arXiv* (2002), quant-ph/0207131.

[242] J. B. Rosser and L. Schoenfeld, "Approximate formulas for some functions of prime numbers," *Illinois J. Math.* **6** no. 1, (03, 1962).

[243] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association* **58** no. 301, (1963).

[244] H. Cohen, *A Course in Computational Algebraic Number Theory*. Springer, 1993.

[245] E. Knill, "On shor's quantum factor finding algorithm: Increasing the probability of success and tradeoffs involving the fourier transform modulus," 1995.

[246] H. Cohen, *Advanced Topics in Computational Number Theory*. Graduate Texts in Mathematics. Springer.

[247] D. Nagaj, P. Wocjan, and Y. Zhang, "Fast amplification of qma," *Quantum Info. Comput.* **9** no. 11, (2009), arXiv:0904.1549 [quant-ph].

[248] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Phys. Rev. A* **54** (Aug, 1996).

[249] A. Steane, "Multiple-particle interference and quantum error correction," *Proceedings: Mathematical, Physical and Engineering Sciences* **452** no. 1954, (1996), quant-ph/9601029.

[250] A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane, "Quantum error correction and orthogonal geometry," *Phys. Rev. Lett.* **78** (Jan, 1997), quant-ph/9608006.

[251] P. Wocjan and S. Zhang, "Several natural BQP-Complete problems," arXiv:quant-ph/0606179.

[252] W. R. Bloom and H. Heyer, *Harmonic analysis of probability measures on hypergroups / Walter R. Bloom, Herbert Heyer*. W. de Gruyter Berlin ; New York, 1995.

[253] X. Ni, O. Buerschaper, and M. Van den Nest, "A non-commuting stabilizer formalism," *Journal of Mathematical Physics* **56** no. 5, (2015), arXiv:1404.5327 [quant-ph].

[254] J. Bermejo-Vega, "Classical simulations of non-abelian quantum Fourier transforms.," Master's thesis, Technical University of Munich, 2011. `http://www2.mpq.mpg.de/Theorygroup/CIRAC/wiki/images/8/8e/BermejoVega_diploma_2011.pdf`.

[255] R. Roth, "Character and conjugacy class hypergroups of a finite group," *Annali di Matematica Pura ed Applicata* **105** no. 1, (1975).

[256] J. McMullen, "An algebraic theory of hypergroups," *Bulletin of the Australian Mathematical Society* **20** (1, 1979).

[257] J. McMullen and J. Price, "Duality for finite abelian hypergroups over splitting fields," *Bulletin of the Australian Mathematical Society* **20** (Jan, 1979).

[258] C. F. Dunkl, "The measure algebra of a locally compact hypergroup," *Transactions of the American Mathematical Society* **179** (1973).

[259] R. I. Jewett, "Spaces with an abstract convolution of measures," *Advances in Mathematics* **18** no. 1, (1975).

[260] R. Spector, "Mesures invariantes sur les hypergroupes," *Trans. Am. Math. Soc.* **239** (1978).

[261] N. J. Wildberger, "Duality and entropy for finite commutative hypergroups and fusion rule algebras," *Journal of the London Mathematical Society* **56** (1997).

[262] N. Wildberger, "Lagrange's theorem and integrality for finite commutative hypergroups with applications to strongly regular graphs," *Journal of Algebra* **182** no. 1, (1996).

[263] N. Wildberger, *Algebraic structures associated to group actions and sums of hermitian matrices*. Textos de matemática. Departamento de Matemática da Universidade de Coimbra, 2001. https://www.researchgate.net/profile/Norman_Wildberger/publication/251560903_Algebraic_Structures_associated_to_group_actions_and_sums_of_Hermitian_matrices/.

[264] R. Ichihara, *Order Structures of Hypergroup Extensions with respect to Subhypergroups and their Quotients*. Phd thesis, Chiba University, 2010.

[265] S. Yamanaka, *Extension problem and duality of conditional entropy associated with a commutative hypergroup*. Phd thesis, Osaka Prefecture University, 2013.

[266] I. Isaacs, *Character Theory of Finite Groups*. Dover books on advanced mathematics. Dover.

[267] "In what sense is the classification of all finite groups "impossible"?" Discussion at MathOverflow. http://mathoverflow.net/q/180355.

[268] J. Figueroa-O'Farrill, "When is a classification problem "wild"?." http://mathoverflow.net/q/10481.

[269] D. Bacon, "How a Clebsch-Gordon transform helps to solve the Heisenberg hidden subgroup problem," *Quantum Info. Comput.* **8** no. 5, (2008), arXiv:quant-ph/0612107.

[270] J. Watrous, "Quantum algorithms for solvable gropus," in *Proceedings of the 33rd ACM Symposium on Theory of Computing*. 2001. arXiv:quant-ph/0011023.

[271] "When does a normal subgroup contain pricesely one non-identity conjugacy class?." http://math.stackexchange.com/questions/176242/when-does-a-normal-subgroup-contain-precisely-one-non-identity-conjugacy-class. Accessed: May 2015.

[272] D. S. Dummit and R. M. Foote, *Abstract Algebra*. John Wiley & Sons, Inc., 2004.

[273] S. Lang, *Algebra*. Springer, 2002.

[274] A. Kleppner, "Multipliers on abelian groups," *Mathematische Annalen* **158** no. 1, (1965).

[275] D. Lorenzini, *An Invitation to Arithmetic Geometry*. Graduate studies in mathematics. American Mathematical Society, 1997.

[276] J.-P. Serre, *Linear Representations of Finite Groups*. Springer, 1996.

[277] In preparation.

[278] C. M. Dawson and M. A. Nielsen, "The Solovay-Kitaev algorithm," *Quantum Info. Comput.* **6** no. 1, (Jan., 2006), quant-ph/0505030.

[279] S. L. Braunstein, "Squeezing as an irreducible resource," *Phys. Rev. A* **71** (May, 2005), quant-ph/9904002.

[280] Arvind, B. Dutta, N. Mukunda, and R. Simon, "The real symplectic groups in quantum mechanics and optics," *Pramana* **45** no. 6, (1995), quant-ph/9509002v3.

[281] A. Bouland and S. Aaronson, "Generation of universal linear optics by any beam splitter," *Phys. Rev. A* **89** (Jun, 2014), arXiv:1310.6718 [quant-ph].