New methods for Quantum Compiling

by

Vadym Kliuchnikov

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science - Quantum Information

Waterloo, Ontario, Canada, 2014

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The efficiency of compiling high-level quantum algorithms into instruction sets native to quantum computers defines the moment in the future when we will be able to solve interesting and important problems on quantum computers. In my work I focus on the new methods for compiling single qubit operations that appear in many quantum algorithms into single qubit operations natively supported by several popular architectures. In addition, I study several questions related to synthesis and optimization of multiqubit operations.

When studying the single qubit case, I consider two native instruction sets. The first one is Clifford+T; it is supported by conventional quantum computers implementing fault tolerance protocols based on concatenated and surface codes, and by topological quantum computers based on Ising anyons. The second instruction set is the one supported by topological quantum computers based on Fibonacci anyons. I show that in both cases one can use the number theoretic structure of the problem and methods of computational algebraic number theory to achieve improvements over the previous state of the art by factors ranging from 10 to 1000 for instances of the problem interesting in practice. This order of improvement might make certain interesting quantum computations possible several years earlier.

The work related to multiqubit operations is on exact synthesis and optimization of Clifford+T and Clifford circuits. I show an exact synthesis algorithm for unitaries generated by Clifford+T circuits requiring exponentially less number of gates than previous state of the art. For Clifford circuits two directions are studied: the algorithm for finding optimal circuits acting on a small number of qubits and heuristics for larger circuits optimization. The techniques developed allows one to reduce the size of encoding and decoding circuits for quantum error correcting codes by 40-50% and also finds their applications in randomized benchmarking protocols.

Acknowledgements

I would like to thank to my advisor Michele Mosca and my co-advisor Dmitri Maslov for their guidance and for creating motivating research environment. Many enlightening discussions with the members of quantum circuits group at Institute for Quantum Computing were very important part of it. I always found very helpful to share thoughts and ideas with Adam Paetznick, Matthew Amy and Martin Roetteler. I am grateful to them for their questions and feedback. During the years of my PhD program I had a pleasure to collaborate with and to learn from Alex Bocharov, David Gosset, Krysta Svore, and Nathan Wiebe. I hope to do much more together with them.

I was happy to be a member of Institute for Quantum Computing. I am grateful to all people who participated in creating it and to all people who made it such a pleasant experience to work there.

Also I would like to thank to many great scientists and researchers I worked with in Kyiv before the start of my PhD program at University of Waterloo. In particular, to Andrey A. Dorogotsev for helping me to develop as a researcher and mathematician and to Helen V. Gomonay for introducing me to quantum computing and inspiring to read papers on topological quantum computing.

I would like to thank to my old friends and to new friends I met at Waterloo for their support during the time I worked on PhD program. Finally, I would like to thank to my family for supporting me in every small step forward I needed to make in my life.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Quantum computing is a computing paradigm harnessing the inherent complexity of physical systems described by the laws of Quantum Mechanics. The number of variables needed to describe the results of experiments involving such physical systems scales exponentially with the number of subsystems. There are examples of interesting and important problems that can be solved asymptotically more efficiently on a quantum computer than on a classical one.

A quantum compiler is a program or a set of programs for a classical computer that addresses the following problem: given a high level description of a quantum algorithm (the procedure for solving a certain problem on a quantum computer) find a way to execute it on a specific quantum system. For the algorithm to be executed on a specific quantum system, the quantum compiler must produce a description of a procedure that is executed by a classical hardware controlling the quantum system. In addition, the procedure should be as efficient as possible. We use the term "quantum compiling" to denote a set of problems arising when creating a quantum compiler.

Building a quantum computer is a hard problem and the resources available for quantum computation will be very limited for a long period of time. It is important to harness as much computational power as possible taking into account the limited number of qubits (the limited size of a quantum computer's registers and memory) and computation time available. The efficiency of a quantum compiler is one of the factors that defines how early in the future we will be able to benefit from having a quantum computer. Our goal is to demonstrate different methods and ideas that lead to improvements in the amount of resources needed by factors between two and thousands depending on the quantum compiling problem type and size considered.

To describe the set of problems we are solving and identify resources we are minimizing we examine typical examples of quantum computer architectures. The design of modern quantum computer architectures is directly related to the main challenges of building a quantum computer. To show the context in which quantum compiling problems appear we consider examples of interesting quantum algorithms, and a protocol useful for benchmarking a quantum computer.

1.1 Quantum computer architectures

One of the main challenges on the way of building a quantum computer is a precise and accurate control of quantum systems. On one hand, a quantum system used for a computation must be isolated from the environment to be protected from noise. On the other hand, interactions with some external physical system are needed for the system's control. Naively one may assume that solving bigger instances of problems on a quantum computer requires more accurate control of a quantum system used for the computation, and that progress in the field of quantum computing will always be limited by this factor. Indeed, larger instances of the problem require more elementary operations and the overall imprecision of the computation is estimated as a sum of imprecisions of elementary steps. Fortunately, it is possible to overcome this issue either by running a fault-tolerance protocol, or by using quantum systems with topological degrees of freedom. Next we look at the examples of quantum computer architectures dictated by both solutions to the issue.

1.1.1 Quantum computer running fault-tolerance protocol

Quantum Threshold Theorems are one of the important steps towards practical quantum computing [65, 3]. The main implication of the theorems is that if one can achieve a level of accuracy above a certain threshold (or, in other words, error rate below a certain value), then arbitrary precision quantum computation can be performed with only poly-logarithmic overhead. These result are based on using the theory of Quantum Error Correcting Codes. Recently it was shown that the same could be achieved with only a constant overhead using certain families of codes [30, 33]. Usually the proof of a quantum threshold theorem is constructive — it describes a way how to perform an arbitrary precision computation given a quantum computer with some fixed error rate below the threshold. In other words, the proof describes a fault-tolerance protocol.

The fault tolerance protocol describes a way to get the basic operations needed for a quantum computation with a sufficiently high level of accuracy. It introduces redun-

dancy by encoding one logical qubit using several physical qubits and includes methods for initializing logical qubits, performing gates and getting measurement results with much lower noise levels. Measurement and state preparation in this case are usually limited to measurement in the computational basis and preparation of $|0\rangle$ state. The set of gates that can be executed fault-tolerantly is limited to a discrete set; the Clifford+T gate set is one of the most common gate sets provided by fault tolerant protocols. An example of a quantum computer architecture running fault tolerance protocol is shown on Figure 1.1; more details on this architecture can be found at [40].

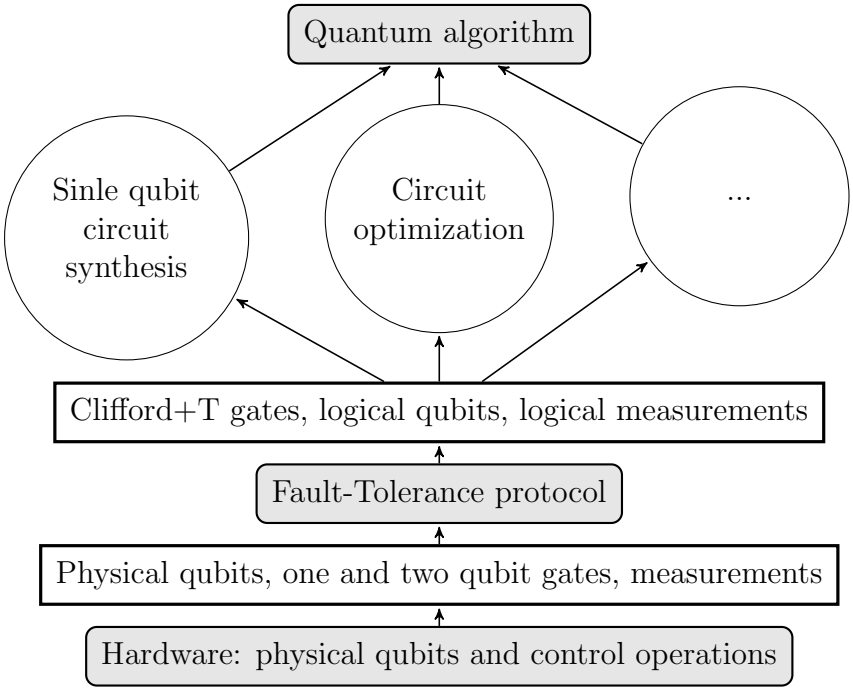

Figure 1.1: Simplified layered architecture of a quantum computer running fault-tolerance protocol. White rectangles show functionalities provided by lower levels to upper levels. White circles show quantum compiling problems that must be solved to execute quantum algorithm using services provided by lower levels.

The Pauli and Clifford groups are crucial in the theory of Quantum Error Correction and Fault Tolerance. In addition, unitaries from these groups are usually easy to implement fault-tolerantly. We discuss this in more detail after the definitions of the Pauli and Clifford groups.

The single qubit Pauli group is generated by:

$$\mathrm{X} := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \mathrm{Z} := \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, e^{i\phi}\mathrm{I} := \begin{pmatrix} e^{i\phi} & 0 \\ 0 & e^{i\phi} \end{pmatrix}.$$

Any element of the single qubit Pauli group can be written as $e^{i\phi}\mathrm{X}^{b(1)}\mathrm{Z}^{b(2)}$ where $b(j)$ are zero or one. For example, the Pauli matrix Y is equal to $i\mathrm{XZ}$. The Pauli group acting on n qubits is the group of the following tensor products

$$e^{i\phi}\mathrm{X}^{b(1)}\mathrm{Z}^{b(2)} \otimes \mathrm{X}^{b(3)}Z^{b(4)} \otimes \ldots \otimes \mathrm{X}^{b(2n-1)}\mathrm{Z}^{b(2n)}, \text{ for } b(j) \in \{0,1\}, \phi \in [0, 2\pi)$$

equivalently it is a group generated by all possible global phase operators $e^{i\phi}I$, and the Pauli matrices X and Z acting on k-th qubit for $k = 1, \ldots, n$.

The Clifford group on n qubits is a group of unitaries that maps elements of the n qubit Pauli group to themselves by conjugation. Equivalently this is a group generated by the Hadamard(H), Controlled-NOT(CNOT), Phase(P) gates and $e^{i\phi}I$ where

$$\mathrm{H} := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \mathrm{P} := \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}.$$

The Controlled-NOT with a control on the first qubit and target on the second qubit is represented by the unitary matrix:

$$\mathrm{CNOT}_{1,2} := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \frac{(\mathrm{I}+\mathrm{Z})}{2} \otimes \mathrm{I} + \frac{(\mathrm{I}-\mathrm{Z})}{2} \otimes \mathrm{X}$$

Similarly one can define $\mathrm{CNOT}_{k,j}$ with control on k-th qubit and target on j-th qubit.

Definition 1.1.1. *The Clifford gate set on n qubits is the set of Hadamard, Phase and single qubit Pauli gates acting on each qubit together with $\mathrm{CNOT}_{k,j}$ for k, j from $1, \ldots, n$ and $k \neq j$.*

The details of logical Clifford gate implementations depend on the quantum error correcting code used in the fault-tolerance protocol. The most simple situation is when most of the Clifford gates can be implemented transversally (see Figure 1.2). The procedures for implementing Clifford gates when using the surface code are also relatively simple [23].

A quantum computer that supports only initialization of its registers to the states of computational basis, Clifford gates and measurements can be simulated classically due to

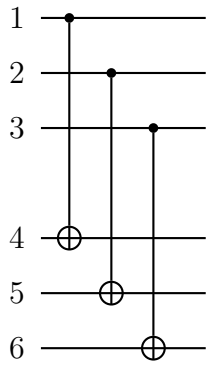

Figure 1.2: Example of transversal implementation of a logical CNOT gate. Qubits 1-3 encode the first logical qubit and qubits 4-6 the second.

the Gottesman-Knill theorem[60]. In other words, the mentioned set of operations is not universal for quantum computation. This issue is solved by adding one gate that is outside of the Clifford group to the gate set. The most common single qubit gate used for this purpose is a T gate:

$$\mathrm{T} := \begin{pmatrix} 1 & 0 \\ 0 & \zeta_8 \end{pmatrix}, \zeta_8 := e^{i\pi/4}.$$

We discuss the question of gate set universality in more details in Section 2.

Definition 1.1.2. *The Clifford+*T *gate set on n qubits is the set of* T *gates acting on each qubit together with the gates from Clifford gate set on n qubits.*

In many common fault-tolerance protocols, such as those based on a surface code, the T gate can only be implemented using magic state distillation [23]. The distillation protocol requires the use of extra qubits, includes measurement, state preparations and requires much longer time to be executed in comparison to the time needed for Clifford gates. For this reason the aim of many methods described in this work is to minimize the number of T gates. To be concise we call the number of T gates used for a certain task the *T-count*.

In summary, we consider a quantum computer running fault-tolerance protocol that supports initialization of its registers to a $|0\rangle$ state, gates from a Clifford+T gate set and measurements of qubits in a computational basis. The cost of running the algorithm on such a computer is dominated by the number of T gates used to run the algorithm. This cost model is very rough. For example, it does not take into account the geometry of the quantum computer. However, it serves as a good first order approximation for estimating resources needed to run quantum algorithms.

1.1.2 Topological Quantum Computer

The advantage of a topological quantum computer is that many or all gates can be made fault-tolerant by hardware design; however designing this type of computer is an even harder problem and is currently much less developed than the approach to a quantum computing based on using fault-tolerance protocols.

Non-Abelian anyons are quasi particles used for topological quantum computation. The name "anyon" related to the fact that they have different exchange statistics from Bosons and Fermions. When two anyons are exchanged their wave function can pick up an arbitrary phase in contrast to no extra phase for bosons and minus one for fermions. This interesting property is due to the fact that anyons are quasi particles that exist only in two-dimensional physical systems. If one particle is moved around the other in three dimensions its trajectory is topologically equivalent to a point — it can be continuously transformed

Topological quantum computation is performed by moving anyons around each other. The resulting transformation of the particles' wave function implemented by such movements depends only on the topological properties of particles' trajectories. Dependence of the transformation only on the topological properties is crucial for achieving fault tolerance. The result does not depend on small perturbations in the particles trajectories. More details on the topic can be found in [64].

Braids and the braid group are mathematical objects describing topological properties of anyons' trajectories. Here we discuss them informally, but on a level sufficient for this work. To describe a braid on n strands let us fix n distinct point on the plain. Now consider a continuous motion of n particles starting from the mentioned n points and ending at the same n points. The particle that started at some point can end up at any other point. Two braids are equivalent if the world lines of particles of one braid can be continuously transformed into world lines of the particles of the other braid. Figure 1.3 shows an example of two equivalent braids on three strands. Two braids are composed by stacking one on top of the other (see Figure 1.4).

Any n-strand braid can be represented using $n-1$ generating braids σ_k and their inverses. Braid σ_k involves only movement of the k^{th} and $(k+1)^{\text{th}}$ particles. In Figure 1.5 we show braids generating all braids with three strands. This gives a way to represent braids in abstractly using the braid group. The braid group on n strands is a group with $n-1$ generators σ_k such that the following identities hold

$$\begin{cases} \sigma_k\sigma_j = \sigma_j\sigma_k, & |k-j| > 2 \\ \sigma_k\sigma_{k+1}\sigma_k = \sigma_{k+1}\sigma_k\sigma_{k+1} \end{cases}.$$

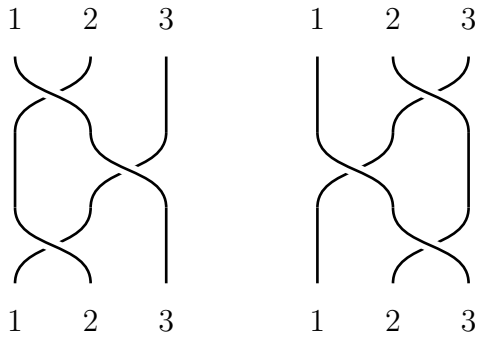

Figure 1.3: Equivalent braids on three strands

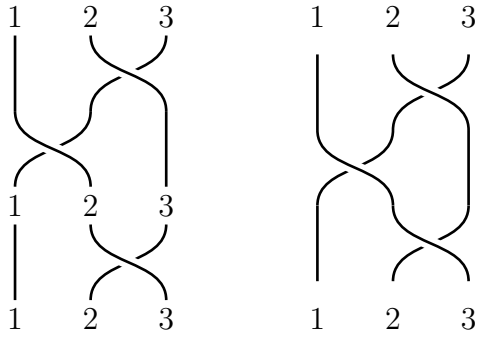

Figure 1.4: Composition of braids

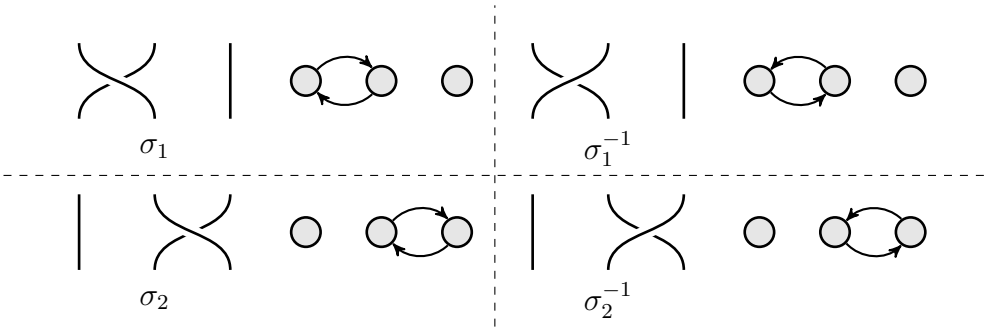

Figure 1.5: Generators of braids on three strands and their inverses. On the right near each braid corresponding particles' moves are shown.

Any braid can be represented as a word in the braid group.

Fibonacci anyons are one of the simplest non-abelian anyons that can simulate an arbitrary quantum computation [24]. They are predicted to exist in physical systems in a fractional quantum Hall state, at filling fraction $\mu = 12/5$ [58]. The method for engineering physical a system such that it is described by Fibonacci anyons was recently proposed in [56]. Now we briefly describe how to simulate an arbitrary quantum algorithm using Fibonacci anyons.

To define a qubit we use three Fibonacci anyons. In the Fibonacci model there are only two types of particles. We denote them by 0 and 1. Zero corresponds to the vacuum and 1 corresponds to the actual excitation. The important part of any non-abelian Fibonacci anyon model is fusion rules. It describes which type of particle we get if we consider a composite system of two particles. When we consider a system of two fermions we get a boson. For Fibonacci anyons the situation is more interesting and the fusion rule is

$$1 \times 1 = 0 + 1.$$

This means that if we consider a system of two 1 particles the type of resulting particle can be either 0 or 1. When we combine 0 with 1 we always get 1. On Figure 1.6 we show the states of three Fibonacci anyons corresponding to a computational basis. Grey circles denote 1 particles. The black oval around first two particles and the number near it shows the type of the first two particles combined. Determining the type of the system of particles is a projective measurement. If we measured that the type of first two particles is zero, than in all subsequent measurements we will find that their type is also zero.

Single qubit operations are performed by moving the three anyons around each other. Any possible way to move them corresponds to a braid that completely defines the single

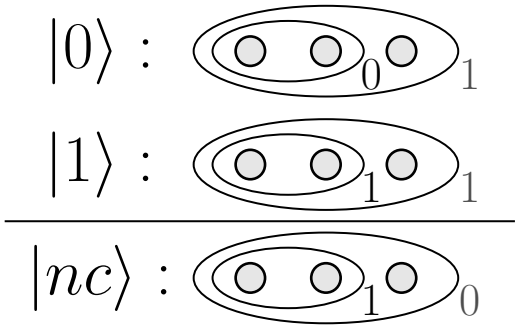

Figure 1.6: Encoding a qubit using three Fibonacci anyons

unitary being applied. To find a unitary corresponding to any braid it is sufficient to know unitaries for generators σ_1 and σ_2 of the three strand braid group. We use σ_1, σ_2 to denote the corresponding unitaries

$$\sigma_1 = \zeta_{10}^6 \begin{pmatrix} 1 & 0 \\ 0 & \zeta_{10}^7 \end{pmatrix}, \zeta_{10} = e^{i\pi/5}. \tag{1.1}$$

It is convenient to express σ_2 using a Fusion matrix F, which is one of the parameters defining the Fibonacci anyon model:

$$F = \begin{pmatrix} \tau & \sqrt{\tau} \\ \sqrt{\tau} & -\tau \end{pmatrix}, \sigma_2 = F\sigma_1 F. \tag{1.2}$$

We will see later that any single qubit unitary can be approximated by unitaries σ_1 and σ_2 within an arbitrary precision.

Note that the Hilbert space of three Fibonacci anyons has three dimensions. When we are interested in single qubit operations, we ignore the subspace spanned by $|nc\rangle$ (see Figure 1.6). Both σ_1, σ_2 leaves this subspace and the subspace spanned by $|0\rangle, |1\rangle$ invariant. The Hilbert space corresponding to n anyons has the number of dimensions equal to the $(n+1)^{th}$ Fibonacci number. Therefore when we want to simulate a two qubit operations we deal with a four dimensional subspace (computational subspace) of the thirteen dimensional Hilbert space. Some of the unitaries corresponding to braid group generators take us outside of computational Hilbert space. This phenomena is called leakage. Any operator on the computational subspace can be approximated within arbitrary precision using anyon braiding [64]. This implies that leakage can always be made arbitrary small.

In summary, in a topological quantum computer based on Fibonacci anyons each qubit is encoded using three anyons and unitaries are performed by moving anyons around each

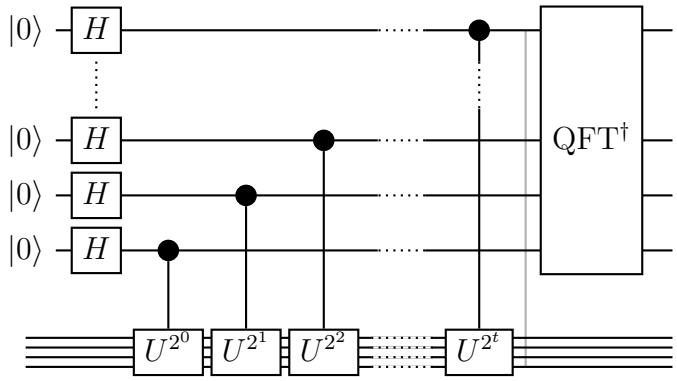

Figure 1.7: Quantum circuit for the phase estimation algorithm [60].

other. The natural way to define the cost function for a unitary implementation is the number of anyons' moves needed to implement it. In this work we are going to look at methods for finding asymptotically optimal implementation of single qubit unitaries.

1.2 Examples of tasks performed on a quantum computer

1.2.1 Integer factoring

Shor's integer factoring algorithm [60, 69] is the one of the most widely known algorithms for quantum computers. Here we review the main building blocks of its part that must be executed on a quantum computer (an order finding algorithm). We also discuss which elementary gates are needed to implemented it and what are the challenges of mapping the algorithm on one of the quantum computer's architectures presented above.

The circuit for a quantum part of the Shor's algorithm (see Figure 1.7) preforms phase estimation procedure with

$$U^{2^k} |y\rangle = |x^{2^k} y \bmod N\rangle$$

where N is a number being factored by the algorithm and x is a number for which we are looking for the minimal natural number m such that $x^m \equiv 1 (\text{mod } N)$. The first part of the circuit involving Controlled-U^{2^k} corresponds to a modular exponentiation. This part can be performed by a circuit consisting of $O(n^3)$ Toffoli, CNOT and X gates for n being a number of bits needed to represent N[44]. The Toffoli gate is a three qubit gate with two

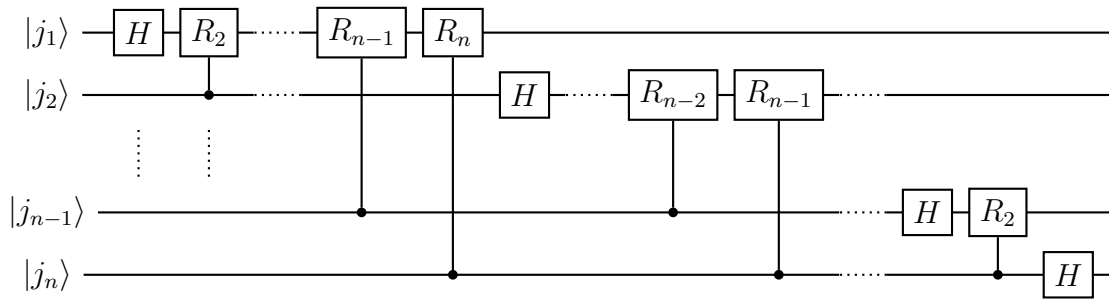

Figure 1.8: Quantum circuit for Quantum Fourier Transform [60] (up to inputs-outputs reordering). The circuit's building blocks are Hadamard and Controlled-R_k gates, where $R_k := \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}$.

control qubits and one target qubit. Its action on computational basis states is defined by the following relation

$$\text{TOF}\,|c_1, c_2, t\rangle = |c_1, c_2, (t \oplus (c_1 \wedge c_2))\rangle$$

where c_1, c_2 are control qubits and t is a target qubit.

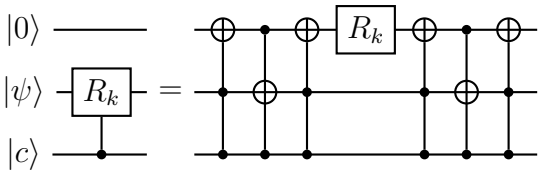

Figure 1.9: A circuit for Controlled-R_k [43]. Groups of three Toffoli gates implement Controlled-SWAP gate. First qubit from the top is an ancillary qubit, the second is a target and the third is a control qubit.

The second part of the circuit on Figure 1.7 is the inverse of Quantum Fourier Transform (QFT). The more detailed circuit for QFT is shown on Figure 1.8. It consists of Hadamard and Controlled-R_k gates where

$$R_k := \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}.$$

Note, that Controlled-R_k can be implemented using Toffoli gates and R_k gates using the circuit on Figure 1.9. In summary, to execute integer factoring on a quantum computer

with a certain architecture we need to know how to execute Toffoli, CNOT , X, Hadamard and R_k.

A quantum computer running fault tolerance protocol described in Section 1.1.1 supports Clifford+T gates. In particular, CNOT ,X and Hadamard gates are natively supported by it and are very cheap to implement. Toffoli gate requires seven T gates without using ancillary qubits and measurements [28] and can be implemented using four T gates otherwise [39]. Unitaries R_k cannot be expressed exactly using Clifford+T gates when k is greater than three and must be approximated. Finding approximations with an optimal T-count is one of the questions studied in this work. Its detailed discussion starts in Chapter 2 and is further developed in Chapters 3,4, and 5.

When using a topological quantum computer based on Fibonacci anyons none of the gates mentioned above can be implemented exactly. In this work we focus on approximation of single qubit gates including X, Hadamard and R_k. In Chapter 2 we discuss our approach to the problem and its relevance to compiling multi qubit gates. The approach is further developed and evaluated in Chapter 6.

1.2.2 Chemistry simulation

One example of the Quantum Chemistry problem that can be solved efficiently on a quantum computer, and in many cases interesting in practice, is finding the energy of a molecule in a certain state [42]. We focus on the part of the problem related to the electronic structure of the molecule. In other words, we assume that the Born-Oppenheimer approximation is used. We briefly discuss the approach to the problem using Second Quantization. We also highlight quantum compiling challenges when using this approach.

To find a contribution to the molecule energy from electrons we need to encode their state into the qubits of a quantum computer and then measure the observable we are interested in — the part of the molecule's Hamiltonian corresponding to the electrons. The second part is achieved using the Phase Estimation procedure (Figure 1.7) with unitary U corresponding to the time evolution of the Hamiltonian. When the Second Quantization is used for a molecule with M spin-orbitals the Hamiltonian is obtained using the Jordan-Wigner transformation [61] and contains $N = O(M^4)$ terms H_k [42]. To perform the evolution according to the Hamiltonian the Trotter-Suzuki formula

$$e^{-i\sum_k H_k} \approx \left(e^{-iH_1\delta t} \cdot \ldots \cdot e^{-iH_N\delta t}\right)^{t/\delta t}$$

is used. It allows one to reduce the evolution to a series of evolutions according to each term of the Hamiltonian separately. Each unitary $e^{iH_k\delta t}$ can be implemented using at most

$O(M^2)$ gates [61] including CNOT and Controlled-$R_z(\phi)$, where

$$R_z(\phi) := \begin{pmatrix} e^{i\phi/2} & 0 \\ 0 & e^{-i\phi/2} \end{pmatrix}.$$

Examples of circuits and resource estimates for the case when the water molecule is considered can be found in [71].

In summary, quantum compiling challenges for problems related to quantum chemistry are similar to integer factoring. The difference is that in chemistry simulation algorithms single qubit rotations are used much more frequently. Therefore the amount of overall resources required is more sensitive to the optimality of single qubit rotations' implementation. Problems related to quantum chemistry require less qubits to achieve interesting results, in comparison to integer factoring. This implies that methods for efficient compiling of single qubit operations will be especially important at the time when first (small) fault-tolerant quantum computers are built.

1.2.3 Randomized benchmarking

Another interesting and important question related to building a quantum computer is: "How to evaluate them, once we built one?". Randomized benchmarking protocols is one of the solutions to the problem. They are already used today to evaluate the quantum information processors (experimental implementations of physical systems having several qubits)[25, 17, 67]. One of the main advantages of randomized benchmarking protocols over the other approaches (such as, for example, Quantum Process Tomography [12]) is their scalability. We briefly describe a randomized benchmarking protocol (following [53]) and show how its implementation can benefit from the access to optimal Clifford circuits.

The randomized benchmarking protocol consists of a series of experiments. In each experiment, first the sequence $C_1, \ldots, C_N$ of Clifford unitaries chosen uniformly at random is applied and then unitary $(C_1 \cdot \ldots \cdot C_N)^\dagger$ is applied, so the overall transformation is identity in the ideal (noise free) case. After this transformation, the overlap between original and final state is measured. By averaging over different choices of random sequences $C_1, \ldots, C_N$, the average fidelity F_N is obtained. Next, the sequence F_N is fit to the model which allows one to find an average error rate of Clifford unitaries. To execute a random Clifford unitary one decomposes it into a sequence of Clifford gates. For example, into CNOT , Hadamard, Phase and single qubit Pauli gates. The efficiency of the decomposition defines the time needed for randomized benchmarking experiments. In Chapter 8 we discuss methods for finding optimal or close to optimal decompositions of Clifford group unitaries acting on up

to five qubits. We also discuss a heuristic that uses a database of optimal Clifford unitaries to optimize larger circuits.

Chapter 2

Approximate synthesis

In the previous Chapter we provided examples showing that many operations used in quantum algorithms cannot be expressed exactly using the operations supported by the discussed architectures of quantum computers. However, both architectures provide capabilities for universal quantum computation. The Clifford+T gate set and the Fibonacci gate set (the gate set supported by a topological quantum computer based on Fibonacci anyons) are *approximately universal*. A gate set is called *approximately universal* if for any unitary and for any precision ε there exists a circuit over the gate set approximating the unitary within precision ε. The corresponding problem in quantum compiling is a problem of finding a circuit that implements a given unitary within the required precision. We call it an *approximate synthesis problem*. In this chapter we review known approaches to the problem and propose a new approach to solving it. The new approach is developed and evaluated in the next chapters.

The Solovay-Kitaev algorithm [44, 18] is a general solution to the approximate synthesis problem for any approximately universal gate set. For approximation precision ε, it outputs the circuit of the size $O(\log^{3.97}(1/\varepsilon))$ and can be executed on a classical computer using $O(\log^{2.71}(1/\varepsilon))$ arithmetic operations [18]. Now we give a short description of the algorithm to discuss the drawbacks of using it in practice.

Consider a special unitary U acting on a d dimensional Hilbert space. Let the set $\mathcal{V}$ be a finite subset of $SU(d)$ such that any element of $SU(d)$ can be approximated by a product of unitaries from $\mathcal{V}$ with arbitrary precision. The Solovay-Kitaev algorithm consists of two stages. In the first stage one builds an ε-net of $SU(d)$ (a set such that for any element of $SU(d)$ there is an element of the set within distance ε from it) using the products of elements of $\mathcal{V}$. In the second stage the balanced group commutator decomposition is used

to achieve arbitrary quality of approximation using the ε-net of fixed size. This stage is recursive; here we provide an example when the algorithm is used with recursion depth one. First an approximation U_0 of U is found using the ε-net and U is represented as $U_0\Delta$ where Δ is guaranteed to be within distance ε from the identity. Next, unitary Δ is approximated using a group commutator $VWV^\dagger W^\dagger$ within precision $c_1\varepsilon^{3/2}$. Finally one approximates W, V using some elements W_0, V_0 of ε-net. As a result, the product $U_0V_0W_0V_0^\dagger W_0^\dagger$ (which can be represented as a product of elements of $\mathcal{V}$) approximates U with precision $c_2\varepsilon^{3/2}$. In summary, with one recursive step the quality of approximation can be improved from ε to $c_2\varepsilon^{3/2}$ using five times longer product of unitaries from $\mathcal{V}$. This gives the scaling $O(\log^{3.97}(1/\varepsilon))$ for the circuit size, because $\log(5)/\log(3/2) \approx 3.97$.

The actual constants hidden behind the big-O notation depend on the Solovay-Kitaev algorithm implementation. The important factor is the accuracy of the ε net used. For example, the use of a canonical form for single qubit Clifford+T circuits allowed authors of [7] to improve the results of running the Solovay-Kitaev algorithm by several orders of magnitude in comparison to the implementation by [18]. Using the ε-net with average quality of approximation 10^{-3} instead of the ε-net with average quality of approximation 10^{-1} allows one to achieve approximation quality $5 \cdot 10^{-4}$ by using around 150 T gates instead of the order of 10^5 T gates. Similar improvements of the Solovay-Kitaev algorithm's implementation were reported in [49]. They were achieved by building a better ε-net using the ideas of Chapter 3. The other drawback of the Solovay-Kitaev algorithm is the big jumps in the quality of the approximations it produces when using a different number of recursion steps.

The related question is: what is the best quality of approximation that can be achieved using the set of approximately universal unitaries $\mathcal{V}$ when approximating elements of $SU(d)$? The volume argument [50, 32] implies that for any ε there always exists an element of $SU(d)$ that requires $\Omega(\log(1/\varepsilon))$ gates to be approximated. This suggest that the best result one can hope for is to be able to approximate any element of $SU(d)$ using $O(\log(1/\varepsilon))$ elements of the set $\mathcal{V}$. In [32] gate sets that allow one to achieve scaling $O(\log(1/\varepsilon))$ are called *efficiently universal*. In the next section we discuss the relation between efficiently universal gate sets and the spectral gap problem. We will conclude that both Clifford+T and Fibonacci gate sets are efficiently universal.

2.1 Efficient universality

It was shown in [32] that efficient universality follows from the spectral gap property of the averaging operator over the set of unitaries $\mathcal{V}$ forming a finite approximately universal

gate set. Later in [8, 9] it was shown that if matrices in $\mathcal{V}$ have entries that are algebraic numbers (roots of a non-zero polynomial with rational coefficients) then the corresponding averaging operator has a spectral gap property. For example, this is the case for Fibonacci and Clifford +T gate sets. Now we briefly discuss why the spectral gap property implies efficient universality.

The averaging operator is a Hermitian linear operator defined on functions from $L^2(SU(d))$ (functions on $SU(d)$ that are square integrable with respect to the Haar measure) as follows

$$(Af)(g) = \frac{1}{2|\mathcal{V}|}\sum_{V\in\mathcal{V}} f(Vg) + f(V^\dagger g).$$

The largest eigenvalue of the averaging operator is 1 and the corresponding eigensubspace contains all constant functions in $L^2(SU(d))$. Operator A has the spectral gap property if the supremum over eigenvalues of A over the orthogonal complement of the constant functions is less than 1. Let $\mathcal{V}^n$ be the set of products of unitaries from $\mathcal{V}$ of length n. Importantly, the spectral gap property implies that the average of function f over the elements of $\mathcal{V}^n$ and their inverses (which is equal to $A^n f$) converges to the integral of f with respect to the Haar measure exponentially fast. Now we show how this implies efficient universality of $\mathcal{V}$.

Suppose we want to approximate unitary U with precision ε. Consider two closed balls in $SU(d)$. The first ball has radius $\varepsilon/2$ and its center is at the identity. The second ball has the same radius and its center is at $UW^\dagger$. If for some element W from $\mathcal{V}^n$ the balls intersect this implies that U can be approximated with precision ε by product of n unitaries from $\mathcal{V}$. The statement that the balls intersect can be written using the characteristic function χ of the first ball (the function that is equal to zero everywhere outside the ball, and equal to 1 inside it). The intersection of the balls being non-empty for some W is equivalent to the following integral over Haar measure being positive

$$\int \left(\chi(g)\frac{1}{(2|\mathcal{V}|)^n}\sum_{W\in\mathcal{V}^n}\chi(WU^\dagger g) + \chi(W^\dagger U^\dagger g)\right) dg$$

Note that the normalized sum (which is equal to $A^n\chi$) in the integral converges to the volume of the $\varepsilon/2$-ball exponentially fast due to the spectral gap property. The volume of the ball is proportional to ε^{d^2-1}. Therefore to ensure that the integral above is positive it is sufficient to choose n of the order $\log(1/\varepsilon)$. For more details on this see Theorem 1 in [32].

The other interesting corollary of the discussed results is related to a quite general class of gate sets. Consider the gate set consisting of Clifford gates and any gate that a) is

not in the Clifford group and b) has entries which are algebraic numbers. It is efficiently universal. This follows from the fact that such a gate set is approximately universal [59]. This is, for example, the case for Clifford+Toffoli gate set.

One of the aims of this work is to develop a constructive method for finding approximations that require $O(\log(1/\varepsilon))$ gates for single qubit Clifford+T and Fibonacci gate sets. The results of Chapter 6 provide an alternative proof of the fact that the single qubit Fibonacci gate set is efficiently universal. Similar result for the single qubit Clifford+T gate set was shown in [68] and relies on results developed in Chapter 3. Next we review approaches to approximate synthesis developed specifically for Clifford+T and Fibonacci gate sets.

2.2 Approximate synthesis with Clifford+T

So far we have considered the problem of approximate synthesis in the setting when a unitary operation acting on some number of qubits is approximated by a finite set of unitaries acting on the same number of qubits. In this section we consider approximation using a more general set of operations involving the use of ancillary qubits initialized to a certain states (usually called resource states), measurements and classical feedback. We review methods for solving an approximate synthesis problem in this more general setting.

Note that any multiqubit unitary can be expressed using CNOT gates and single qubit unitaries [5, 2, 57]. We focus on the approximation of single qubit unitaries. Furthermore, any single qubit unitary can be expressed as a product $R_z(\alpha)\mathrm{H}R_z(\beta)\mathrm{H}R_z(\gamma)$, so it is sufficient to solve the approximation problem for operators $R_z(\phi)$. Also, as we have seen in the previous Chapter, $R_z(\phi)$ is one of the most frequently used single qubit unitaries when constructing quantum algorithms. These are the reasons why all the methods reviewed further in this section focus on operators $R_z(\phi)$.

The method proposed in [44] requires $O(\log(1/\varepsilon))$ ancillary qubits and the following resource state to achieve precision 2^{-n}:

$$|\psi_{n,1}\rangle = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} e^{\frac{2\pi i j}{2^n}} |j\rangle .$$

To simulate $R_z(\frac{2\pi i l}{2^n})$ using the state $|\psi_{n,1}\rangle$ it is sufficient to perform the circuit in Figure 2.1. Unitary U_l performs initialization of qubits $2, \ldots, n+1$ to a binary representation of l controlled on the first qubit, and the adder performs the following transformation

$$|l\rangle |j\rangle \mapsto |l\rangle |(l+j) \bmod 2^n\rangle .$$

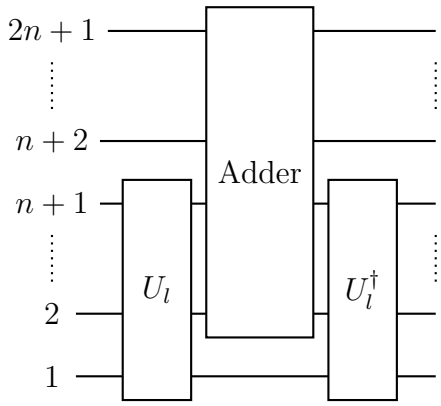

Figure 2.1: Simulating $R_z(\frac{2\pi i l}{2^n})$ using adder circuit and a resource state. First qubit is a target, qubits $2, \ldots, n+1$ are initialized to $|0\rangle$, qubits $n+2, \ldots, 2n+1$ are initialized to the Fourier state $|\psi_{n,1}\rangle$.

Taking into account that $|l\rangle\,|\psi_{n,1}\rangle$ is an eigenstate of the modular addition circuit with eigenvalue $e^{\frac{2\pi i l}{2^n}}$ we get the required result. A circuit for U_l can be implemented using at most n CNOT gates and the adder can be implemented using $O(n)$ T gates and $O(n)$ ancillary qubits [20]. The interesting detail of this method is that resource state can be reused to implement a series of rotations. Most of the complexity of this method is "moved" into the resource state preparation. An efficient approach to preparation of the state $|\psi_{n,1}\rangle$ using $O(n\log(n))$ Toffoli gates is discussed in [36]. The next method takes the idea of moving the cost of the approximation procedure into the state preparation stage to a further extreme.

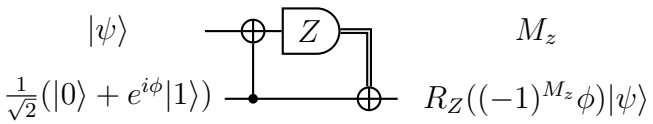

Figure 2.2: Circuit that applies $R_z(\phi)$ or $R_z(-\phi)$ probabilistically. The operator applied is indicated by the measurement outcome M_z.

In [41] authors propose the method based on the circuits with measurement and the classical feedback to simulate $R_z(\phi)$. The main building block of their circuit is shown on Figure 2.2. The circuit performs either $R_z(\phi)$, or $R_z(-\phi)$ with probability 1/2. The measurement outcome indicates which of the two was the case. If undesirable rotation $R_z(-\phi)$ is performed, the correction by $R_z(2\phi)$ is applied using the same circuit. Such a

procedure requires a series of resource states $\frac{1}{\sqrt{2}}(|0\rangle + e^{2^n i\phi}|1\rangle)$. The analysis in the paper shows that the procedure on average succeeds after a constant number of steps and each of them requires only CNOT, measurement and classical feedback. This is an example of the approach where all the complexity is moved into the state preparation phase. In contrast to the method based on the adder, this one "consumes" resource states. The circuit on Figure 2.2 is also studied in [21], but the main focus of the work is the protocol for preparation of the required resource states. In particular, it is shown that the overall cost of the protocol has a better scaling $O(\log^{1.75}(1/\varepsilon))$ than the cost of the circuits obtained using the Sololay-Kitaev algorithm.

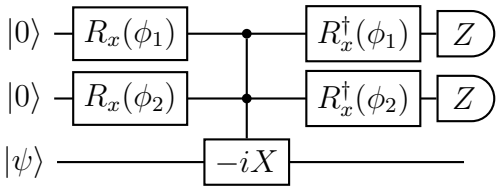

Figure 2.3: Circuit implementing $R_x(\phi)$ for $\phi \approx \phi_1^2\phi_2^2/8$ when ϕ_1 and ϕ_2 are small. The operator $R_x(\phi)$ is implemented when both measurement outcomes are 0. Otherwise the circuit implements a unitary from the Clifford group.

The other interesting class of circuits similar to the circuit on Figure 2.2 is the repeat until success circuits. Similar to the one on Figure 2.2, they perform the required operation only with a certain probability of success. The success is also indicated by the measurement outcome. The difference is that in the case of failure they apply a Clifford operation to the input state which can be undone cheaply. Therefore such a circuit can be repeated on the input state until it succeeds. In [72] the authors shows how to use this type of circuit to make the cost of approximating single qubit rotations depend more on the relative precision than on the absolute precision. Small relative precision of approximation is common for quantum chemistry simulations [72]. Figure 2.3 shows the circuit that allows one to implement unitary $R_x(\phi) := HR_z(\phi)H$ for $\phi \approx \phi_1^2\phi_2^2/8$ when ϕ_1 and ϕ_2 are small. The circuit allows one to implement the mantissa and the exponent part of ϕ separately. The work [72] also contains an efficient way of constructing $R_x(\phi)$ for very small values of ϕ to implement the exponent part $R_z(\phi_1)$. The circuit for the mantissa part $R_x(\phi_2)$ can be found using any other method for approximating R_z operators.

The authors of [62](the work that coined the name "repeat until success") also use repeat until success circuits to construct very cost efficient approximation of R_z operators. The average T-count required to implement R_z rotation scales as $1.3\log_2(1/\varepsilon) - 2.79$ and uses a fixed number of ancillary qubits. However the method relies on a brute force search

and is limited to approximation precision $\varepsilon \geq 10^{-6}$. This is better than using naive brute force to find optimal approximations of R_z with single qubit Clifford+T circuits. In the latter case one is limited to precision $\varepsilon \geq 10^{-4}$. The T-count of optimal single qubit approximations scales as $3.067 \log_2(1/\varepsilon) - 4.32$. In Chapter 5 we develop a method that allows one to find T-optimal single-qubit approximations with precision down to $\varepsilon = 10^{-15}$.

Some of the reviewed methods achieve the scaling $O(\log(1/\varepsilon))$ of T-count or close to it, by using measurements or resource states. However, they have disadvantages. For example, the method based on an adder circuit requires $O(\log(1/\varepsilon))$ ancillary qubits. This might be a problem when the size of a quantum computer is limited. Methods relying on resource states of the form $|0\rangle + e^{i\phi 2^n} |1\rangle$ require an efficient procedure for the preparation of the resource states. The method based on approximating $R_z(\phi)$ with repeat until success circuits is limited by computational resources available for a brute force search. Finally, the constructive proof of the corollary from a spectral gap property for Clifford+T gate set is interesting on its own. It could be a first step towards simplifying results in [8] or providing a better intuition about them. A construction was discovered in [68] and is based on results of Chapter 3. The author describes a probabilistic algorithm for finding a single qubit Clifford+T circuits with T-count scaling $4 \log(1/\varepsilon) + C$. The statement about the algorithm runtime relies on a conjecture regarding the frequency of prime numbers of the form $8n + 1$. In Chapter 4 we describe an algorithm discovered before [68]. It uses extra ancillary qubits and achieves T-count scaling $O(\log(1/\varepsilon))$ to approximate $R_z(\phi)$ operators. The algorithm provably has probabilistic polynomial runtime. The result relies on the efficient algorithm for solving the four squares Diophantine equation [66] and the Prime Number Theorem. This was achieved using the new approach to approximating unitary operators described later in this chapter.

2.3 Approximate synthesis with Fibonacci gate set

The methods described in the previous section are not applicable to the Fibonacci gate set because they implicitly rely on the fact that X, CNOT, and Toffoli gates can be exactly implemented. Finding a braid approximating CNOT gate is a separate challenge when compiling for Fibonacci gate set. The straightforward approach to the problem is to use the Solovay-Kitaev algorithm. Recall that we are using six Fibonacci anyons to simulate two qubits and the corresponding computational subspace is embedded into the thirteen dimensional Hilbert space of six anyons. To find an approximation of any two qubit gate we will need to write down representations of the five generators of the six strand braid group as unitaries acting on the thirteen dimensional Hilbert space. In theory, it is possible

to apply the Solovay-Kitaev algorithm in this situation. In practice, it might be difficult to build a large enough ε-net to achieve reasonable circuit sizes. This is one of the reasons why alternative methods were developed [35].

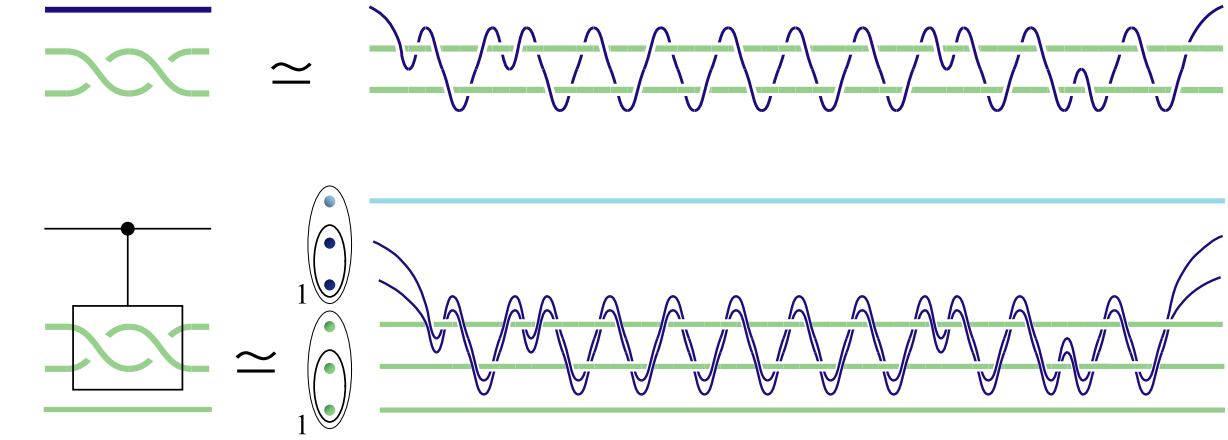

Figure 2.4: Compiling two qubit gates for Fibonacci anyons using weaves. Reprinted from [35].

At a high level, the idea developed in [35] is to reduce the problem of finding approximations of two qubit gates to a problem of finding approximations of a single qubit gates using a special type of braids called weaves. A weave is a type of braid in which positions of all particles except one are fixed and the remaining particle is moved around the other particles. The top part of Figure 2.4 shows a three strand weave. Consider now four particles. By moving two of them we can perform a controlled unitary operation. If the topological charge of the two particles is zero, nothing happens. If the topological charge is one, we perform some unitary transformation defined by weave. This implies that the algorithm for finding efficient approximations of single qubit unitaries with braids leads to an efficient algorithm for approximating two qubit unitaries. We do not consider synthesis with weaves and the application of it to the multi qubit case in this work. In Chapter 6 we develop techniques that allow one to approximate single qubit unitaries using generic braids using $O(\log(1/\varepsilon))$ generators of the braid group. The ideas discussed can be generalized to weaves. The generalization is presented in the second version of [46].

2.4 Number theoretic method

The main idea of the number theoretic method is to split the approximation procedure into two stages (Figure 2.5). In the first stage we approximate an arbitrary unitary with a unitary that can be exactly represented using the gate set we are working with. For brevity we call exactly representable unitaries *exact* unitaries. In the second stage we find a circuit for an exact unitary. Ideas and methods from number theory are used to describe which unitaries are exact, to find circuits implementing exact unitaries and also to find exact unitaries approximating arbitrary ones.

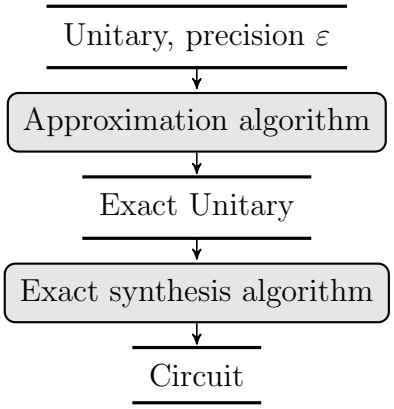

Figure 2.5: The generic scheme of the number theoretic method for approximating unitaries.

Rings of algebraic integers are used to describe the exact unitaries. Exact unitaries corresponding to Clifford+T gate set are described using the ring

$$\mathbb{Z}[\zeta_8] := \left\{ a_0 + a_1\zeta_8 + a_2\zeta_8^2 + a_3\zeta_8^3 \;\middle|\; a_j \in \mathbb{Z}, j = 0, \ldots, 3 \right\}, \zeta_8 := e^{i\pi/4}.$$

The following theorem characterizes single qubit exact unitaries over the Clifford+T gate set.

Theorem 2.4.1. *A single qubit unitary can be exactly represented using the Clifford+*T *gate set if and only if it has the following form*

$$U[x, y, k] := \begin{pmatrix} x & -y^*\zeta_8^k \\ y & x^*\zeta_8^k \end{pmatrix}, x = \frac{x'}{\sqrt{2}^n}, y = \frac{y'}{\sqrt{2}^n}, x', y' \in \mathbb{Z}[\zeta_8], k, n \in \mathbb{Z}.$$

Note that elements of the form $x'/\sqrt{2}^n$ are elements of the ring $\mathbb{Z}[i, 1/\sqrt{2}]$ and alternatively the theorem above can be stated as: "A single qubit unitary can be exactly represented using Clifford+T gate set if and only if it its entries are from the ring $\mathbb{Z}[i, 1/\sqrt{2}]$". Exact unitaries corresponding to single qubit gates from the Fibonacci gate sets are described using the ring

$$\mathbb{Z}[\zeta_{10}] := \left\{ a_0 + a_1\zeta_{10} + a_2\zeta_{10}^2 + a_3\zeta_{10}^3 \;\middle|\; a_j \in \mathbb{Z}, j = 0, \ldots, 3 \right\}, \zeta_{10} := e^{i\pi/5}$$

Note that we need only four integers to describe elements of $\mathbb{Z}[\zeta_{10}]$ because $\zeta_{10}^4 = -1 + \zeta_{10} - \zeta_{10}^2 + \zeta_{10}^3$. The characterization of exact unitaries is the following

Theorem 2.4.2. *A single qubit unitary can be exactly represented using the Fibonacci gate set if and only if it has the following form*

$$V[x, y, k] := \begin{pmatrix} x & y^*\sqrt{\tau}\zeta_{10}^k \\ y\sqrt{\tau} & -x^*\zeta_{10}^k \end{pmatrix}, x, y \in \mathbb{Z}[\zeta_{10}], \tau := \frac{\sqrt{5}-1}{2}.$$

In particular, it is not difficult to see now that Pauli X gate cannot be implemented exactly using the Fibonacci gate set. The proofs of both theorems about exact unitaries are constructive. In Chapters 3,6 we provide efficient exact synthesis algorithms for finding circuits corresponding to exact unitaries and prove their correctness.

Exact synthesis algorithms rely on the complexity measure defined for exact unitaries. The complexity measure is a function of the unitary that indicates what circuit length is needed to implement the given exact unitary. In the case of the Clifford+T gate set it is related to the smallest denominator exponent(sde). The function sde is defined on numbers of the form

$$(a + \sqrt{2}b)/\sqrt{2}^m \tag{2.1}$$

where a, b, m are integers. For a given number, the value of sde is equal to the smallest non-negative m such that the number can be written in the form (2.1). We define the complexity measure for exact unitaries over Clifford+T gate set as:

$$\mu(U[x, y, k]) := \text{sde}|x|^2. \tag{2.2}$$

For the Fibonacci gate set the complexity measure is defined using an automorphism $(\cdot)^\bullet$ of $\mathbb{Z}[\zeta_{10}]$. The automorphism is defined as

$$(\zeta_{10})^\bullet = \zeta_{10}^3, (a)^\bullet = a, \text{ for all } a \in \mathbb{Z}.$$

Input: U — exact unitary
procedure EXACT-SYNTHESIZE(U)
  $g \leftarrow \mu(U), C \leftarrow$ (empty circuit)
  while $g > 3$ **do**
    $J \leftarrow \arg\min_{j \in \{1,\ldots,8\}} \mu(\mathrm{HT}^j U)$
    $U \leftarrow \mathrm{HT}^J U, g \leftarrow \mu(U)$
    Add HT^{8-J} to the beginning of C
  end while
  Lookup circuit C_r for U
  Add C_r to the beginning of C
end procedure
Output: C — circuit over $\langle \mathrm{H}, \mathrm{T} \rangle$

(a) Clifford+T gate set

Input: U — exact unitary
procedure EXACT-SYNTHESIZE(U)
  $g \leftarrow \mu(U), C \leftarrow$ (empty circuit)
  while $g > 1$ **do**
    $J \leftarrow \arg\min_{j \in \{1,\ldots,10\}} \mu(\mathcal{F}\mathcal{T}^j U)$
    $U \leftarrow \mathcal{F}\mathcal{T}^J U, g \leftarrow \mu(U)$
    Add $\mathcal{F}\mathcal{T}^{10-J}$ to the beginning of C
  end while
  Find k, j such that $U = \zeta_{10}^k \mathcal{T}^j$
  Add $\zeta_{10}^k \mathcal{T}^j$ to the beginning of C
end procedure
Output: C — circuit over $\langle \mathcal{F}, \mathcal{T} \rangle$

(b) Fibonacci gate set

Figure 2.6: Exact synthesis algorithms.

This is sufficient to compute the action of the automorphism on any element of $\mathbb{Z}[\zeta_8]$ because it must respect the addition and multiplication operations. The complexity measure for exact unitaries over the Fibonacci gate set is defined as:

$$\mu(V[u, v, k]) := (|u|^2)^\bullet.$$

The larger the value of a complexity measure, the larger a circuit needs to be to implement the given exact unitary.

In the main loop of the exact of synthesis algorithms (Figure 2.6) the complexity measure of the unitary is reduced on each iteration. This is achieved by multiplying the unitary by matrices from a small set and choosing the one that reduces the complexity measure the most. A part of the circuit implementing the unitary is recovered on each iteration. The set of matrices that is used to reduce the complexity measure for the Clifford+T gate set is HT^k. For the Fibonacci gate set this set of matrices is $\mathcal{F}\mathcal{T}^k$ where

$$\mathcal{T} = \begin{pmatrix} 1 & 0 \\ 0 & \zeta_{10} \end{pmatrix}, \mathcal{F} = \begin{pmatrix} \tau & \sqrt{\tau} \\ \sqrt{\tau} & -\tau \end{pmatrix}, \mathcal{T} = \zeta_{10}^2 (\sigma_1)^3, \mathcal{F} = \zeta_{10}^4 \sigma_1 \sigma_2 \sigma_1$$

Our results on single qubit exact synthesis are summarized by the following theorems.

Theorem 2.4.3. *Given exact unitary U the exact synthesis algorithm for the Clifford+T gate set outputs an $\langle \mathrm{H}, \mathrm{T} \rangle$ circuit that implements U. The circuit size is in $O(\mu(U))$. The algorithm requires $O(\mu(U))$ arithmetic operations.*

Theorem 2.4.4. *Given exact unitary U the exact synthesis algorithm for the Fibonacci gate set outputs an $\langle \mathcal{F}, \mathcal{T} \rangle$ circuit that implements U. The circuit size is in $O(\log(\mu(U)))$. The algorithm requires $O(\log(\mu(U)))$ arithmetic operations.*

In Chapter 3 we prove a stronger result about the Clifford+T exact synthesis algorithm. We slightly modify the algorithm to guarantee the optimality of circuits produced by it. The result about the exact synthesis algorithm for the Fibonacci gate set is proved in Chapter 6. It is important for the second stage of our method that we can estimate the complexity of an exact unitary without running the exact synthesis algorithm.

In the second stage of the number theoretic method we need to approximate an arbitrary unitary with an exact one. Consider an example of approximating $R_z(\phi)$. To measure quality of the approximation we use a global phase invariant distance

$$d(U, V) := \sqrt{1 - |tr(UV^\dagger)|/2}.$$

In the case of approximating $R_z(\phi)$ the expression for the distance simplifies as follows:

$$\begin{aligned} d(R_z(\phi), U[x, y, k]) &= \sqrt{1 - \left|\mathrm{Re}(xe^{i\phi/2}\zeta_8^{-k/2})\right|}, \\ d(R_z(\phi), V[x, y, k]) &= \sqrt{1 - \left|\mathrm{Re}(ve^{i\phi/2}\zeta_{10}^{-k/2})\right|}. \end{aligned}$$

We see that the quality of approximation depends only on the one entry of the unitary. To approximate $R_z(\phi)$ we first chose x to achieve required precision and next find y such that $U[x, y, k]$ or $V[x, y, k]$ is a unitary.

To reconstruct a unitary we use the theory of relative norm equations [14]. In the case of the Clifford+T gate set we are dealing with the relative norm equation between $\mathbb{Z}[\zeta_8]$ and its real subring $\mathbb{Z}[\sqrt{2}]$. Suppose that x can be written in the form $x'/\sqrt{2}^n$ for x' from $\mathbb{Z}[\zeta_8]$. We look for y in the form $y'/\sqrt{2}^n$ for y' from $\mathbb{Z}[\zeta_8]$. To ensure that $U[x, y, k]$ is a unitary the following must hold

$$|y|^2 = A + B\sqrt{2}, \text{ for } A + B\sqrt{2} = 2^n - |x|^2. \tag{2.3}$$

The function absolute value squared is a relative norm between $\mathbb{Z}[\zeta_8]$ and $\mathbb{Z}[\sqrt{2}]$. The equation above is called a relative norm equation. In the case of the Fibonacci gate set we are dealing with the relative norm equation between $\mathbb{Z}[\zeta_{10}]$ and its real subring $\mathbb{Z}[\tau]$. The relative norm equation that we need to solve in this case is

$$|y|^2 = A + B\tau, \text{ for } A + B\tau = (1 - |x|^2)/\tau.$$

Both relative norm equations are well studied in the literature. They are not solvable for an arbitrary right hand side. This is one of the challenges that one needs to deal with when reconstructing the unitaries. The other challenge is that solving relative norm equations is as hard as factoring. There are several ways to overcome these challenges.

The first way presented in Chapter 4(for Clifford+T gate set) uses a constant number of ancillary qubits. The use of ancillary qubits allows one to get more freedom while attempting to reconstruct the unitary and relies on the exact synthesis algorithm for multi qubit unitaries developed in [26] and improved in Chapter 7. In this case the reconstruction problem is reduced to the four square Diophantine equation

$$a^2 + b^2 + c^2 + d^2 = A$$

which always has a solution. Even more there is a probabilistic polynomial time algorithm [66] for finding a solution.

The method for solving the Diophantine equation relies on interesting number theoretic results. We can rewrite the four square equation as:

$$|a + ib|^2 = A - c^2 - d^2.$$

One case when the equation above is solvable is when the right hand side is a prime number of the form $4n + 1$. Furthermore it can be solved efficiently, taking time polynomial in the number of bits of the right hand side. It turns out that if we choose c and d at random this happens quite frequently. This is related to the prime number density theorem, which shows that there are approximately $N/\log(N)$ primes that are less than N. It implies that we will need $O(\log(N))$ trials before we succeed. We can also check that the right hand side is a prime number using a primality test. This can be done in polynomial time. It is interesting to note that the equation above can be considered as a norm equation for the Gaussian integers $\mathbb{Z}[i]$.

The second way proposed in [68] uses the idea similar to the one used to solve the four square Diophantine equation. The author proposes a polynomial time algorithm which solves equation (2.3) in the special case when $A^2 - 2B^2$ is a prime number of the form $8n+1$. The second part of his solution is the procedure that allows one to efficiently sample entries x of $U[x, y, 0]$ which give quality of approximation ε. Each sample corresponds to a different instance of the relative norm equation. Similar to the prime number theorem, the author conjectures that the number of instances when $A^2 - 2B^2$ is a prime number of the form $8n + 1$ scales as $O(N/\log(N))$ for N proportional to $1/\varepsilon$. He concludes that an easy instance of the relative norm equation can be found after $O(\log(1/\varepsilon))$ trials. This results in the efficient algorithm for finding an approximation with single qubit Clifford+T

circuits which requires no ancillary qubit. It saturates the lower bound $4\log_2(1/\varepsilon) + C$ on the T-count required to achieve the quality of approximation ε up to an additive constant.

At a high level, we use a similar approach to develop the algorithm for finding approximations using the Fibonacci gate set (see Chapter 6). The number of gates in circuit produced by the algorithm to the achieve quality of approximation ε saturates an asymptotic lower bound $\Omega(\log(1/\varepsilon))$. The algorithm runtime is polynomial in $O(\log(1/\varepsilon))$ (subject to a conjecture similar to the one mentioned above). The results of Chapter 6 also provide an alternative proof to the fact that the Fibonacci gate set is efficiently universal.

The third way of overcoming challenges related to the relative norm equations is to rely on the computational power available. In Chapter 5 we find optimal approximations using the Clifford+T gate set by enumerating all possible x that gives required quality of approximation (starting from the best ones) and solve all instances of the relative norm equation. The approach developed requires less resources than the naive brute force search (the only approach known so far that can guarantee the optimality of approximations). We were able to reach precision 10^{-15} using computational resources available today.

The number theoretic methods allow us to significantly improve efficiency of approximating single qubit operations. In the following Chapters we will develop the required technical tools and perform more thorough evaluation of the method. For the Clifford+T gate set the comparison to the method based on an adder circuit requires more detailed analysis. The advantage of one method over the other might depend on the particular implementation of the fault-tolerant protocol and architecture details. One such comparison was recently performed in [38]. For the Fibonacci gate set the number theoretic method gives an improvement by factors between 10 and 1000 for the range of precision 10^{-10} and 10^{-30} over the Solovay-Kitaev algorithm which was the state of the art for this gate set. Performance of different methods for approximating single qubit unitaries is summarized in Table 2.1.

Table 2.1: Comparison of different methods for single qubit unitary approximation. Number of gates, depth and classical runtime are non-deterministic for some of the methods. In this case their average is reported in the table. For Clifford+T gate set the number of gates reported in the table corresponds to the number of T gates.

Name and Reference	Number of gates and depth	Classical runtime	Number of ancillae	Resource state	Gate set
Solovay-Kitaev [44, 18]	$O(\log^{3.97}(1/\varepsilon))$ $O(\log^{3.97}(1/\varepsilon))$	$O(\log^{2.71}(1/\varepsilon))$	0	no	any
Brute force [22]	$O(\log(1/\varepsilon))$ $O(\log(1/\varepsilon))$	$O((1/\varepsilon)^4)$	0	no	Clifford+T
Phase kickback [44]	$O(\log(1/\varepsilon))$ $O(\log\log(1/\varepsilon))$	$O(\log(1/\varepsilon))$	$O(\log(1/\varepsilon))$	yes [37]	Clifford+T
PAR [41]	$O(1)$ $O(1)$	$O(\log(1/\varepsilon))$	O(1)	yes	Clifford+T
States ladder [21]	$O(1)$ $O(1)$	–	$O(1)$	yes	Clifford+T
[10]	$O(\log^2(1/\varepsilon))$ $O(\log^2(1/\varepsilon))$	$O(\log(1/\varepsilon))$	0	no	Fibonacci
[47], Chapter 4	$O(\log(1/\varepsilon))$ $O(\log(1/\varepsilon))$	$O(\log^{c_1}(1/\varepsilon))$	2	no	Clifford+T
[68]	$4\log(1/\varepsilon)+O(1)$ $4\log(1/\varepsilon)+O(1)$	$O(\log^{c_2}(1/\varepsilon))$	0	no	Clifford+T
[48], Chapter 5	$\sim 3\log(1/\varepsilon))+O(1)$ $\sim 3\log(1/\varepsilon))+O(1)$	$O(1/\varepsilon)$	0	no	Clifford+T
[66]	$3\log(1/\varepsilon))+O(1)$ $3\log(1/\varepsilon))+O(1)$	$O(\log^{c_3}(1/\varepsilon))$	0	no	Clifford+T
[46], Chapter 6	$O(\log(1/\varepsilon))$ $O(\log(1/\varepsilon))$	$O(\log^{c_4}(1/\varepsilon))$	0	no	Fibonacci

Chapter 3

Exact synthesis for the Clifford+T gate set

In this chapter we prove Theorem 2.4.1 and Theorem 2.4.3 related to the exact synthesis of single qubit unitaries over the Clifford+T gate set. The theorems are corollaries of a stronger result. We prove that the exact synthesis algorithm (Algorithm 1) produces circuits with the minimal number of Hadamard and T gates over the single qubit Clifford+T gate set $\mathcal{G}$ (the gate set consisting of Hadamard, T, $T^\dagger$, P, $P^\dagger$, and Pauli-X, Y, and Z gates). We define integer-valued quantities $h(U)$ and $t(U)$ as the minimal number of Hadamard and T gates over all circuits implementing U. We call a circuit H- or T-optimal if it contains the minimal number of H or T gates, correspondingly. The following theorem also relates $h(U)$ and $t(U)$ to the complexity measure $\mu(U)$ (equation 2.2) introduced in Chapter 2.

Theorem 3.0.5. *Let U be a 2×2 unitary over the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ with a matrix entry z such that $\mathrm{sde}(|z|^2) \geq 4$. Algorithm 1 produces a circuit that implements U over $\mathcal{G}$ with:*

1. *the minimal number of Hadamard gates and $h(U) = \mu(U) - 1$, and*
2. *the minimal number of T gates and $t(U) = \mu(U) - (l \mod 2) - (j \mod 2)$, where l and j are chosen such that $h(\mathrm{HT}^l U \mathrm{T}^j \mathrm{H}) = \mu(U) + 1$.*

We also demonstrate how to use the exact-synthesis techniques to improve the implementation of the Solovay-Kitaev algorithm. They allow us to build a higher quality ε-net. To evaluate the improvement we compare our implementation of the Solovay-Kitaev algorithm to the implementation by Dawson. Taking into account the progress achieved by the number theoretic methods this improvement to the Solovay-Kitaev algorithm is more of pedagogical value than of practical value.

3.1 Reducing unitary implementation to state preparation

In this section we discuss the connection between state preparation and implementation of a unitary by a quantum circuit. In the next section, we prove the following result:

Lemma 3.1.1. *Any single-qubit state with entries in the ring* $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ *can be prepared using only* H *and* T *gates given the initial state* $|0\rangle$.

We first establish why Lemma 3.1.1 implies that any single-qubit unitary with entries in the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ can be implemented exactly using H and T gates.

Observe that any single-qubit unitary can be written in the form

$$\begin{pmatrix} z & -w^* e^{i\phi} \\ w & z^* e^{i\phi} \end{pmatrix}, z, w \in \mathbb{Z}[\frac{1}{\sqrt{2}}, i], k = 0, 1, \ldots, 7.$$

where * denotes the complex conjugate. The determinant of the unitary is equal to $e^{i\phi}$ and belongs to the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ when all entries of the unitary belong to the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$. It turns out that the only elements in the ring with the absolute value of 1 are ζ_8^k for integer k. We postpone the proof; it follows from techniques developed in Section 3.4 and discussed at the end of the appendix. For now, we conclude that the most general form of a unitary with entries in the ring is:

$$U[z, w, k] = \begin{pmatrix} z & -w^* \zeta_8^k \\ w & z^* \zeta_8^k \end{pmatrix}.$$

We next show how to find a circuit that implements any such unitary when we have a circuit that prepares its first column given the state $|0\rangle$. Suppose we have a circuit that prepares state $\begin{pmatrix} z \\ w \end{pmatrix}$. This means that the first column of a unitary corresponding to the circuit is $\begin{pmatrix} z \\ w \end{pmatrix}$ and there exists an integer k' such that the unitary is equal to:

$$\begin{pmatrix} z & -w^* \zeta_8^{k'} \\ w & z^* \zeta_8^{k'} \end{pmatrix}.$$

We can synthesize all possible unitaries with the first column $(z, w)^t$ by multiplying the unitary above by a power of T from the right:

$$\begin{pmatrix} z & -w^* \zeta_8^{k'} \\ w & z^* \zeta_8^{k'} \end{pmatrix} \mathrm{T}^{k-k'} = \begin{pmatrix} z & -w^* \zeta_8^{k} \\ w & z^* \zeta_8^{k} \end{pmatrix}.$$

Table 3.1: First four elements of sequence $(\mathrm{HT})^n |0\rangle$

n	$(\mathrm{HT})^n |0\rangle = \begin{pmatrix} z_n \\ w_n \end{pmatrix}$	$\begin{pmatrix} \lvert z_n \rvert^2 \\ \lvert w_n \rvert^2 \end{pmatrix}$
1	$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$	$\frac{1}{(\sqrt{2})^2} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$
2	$\frac{1}{(\sqrt{2})^2} \begin{pmatrix} \omega + 1 \\ 1 - \zeta_8 \end{pmatrix}$	$\frac{1}{(\sqrt{2})^3} \begin{pmatrix} \sqrt{2} + 1 \\ \sqrt{2} - 1 \end{pmatrix}$
3	$\frac{1}{(\sqrt{2})^2} \begin{pmatrix} \omega^2 - \omega^3 + 1 \\ \omega \end{pmatrix}$	$\frac{1}{(\sqrt{2})^4} \begin{pmatrix} 3 \\ 1 \end{pmatrix}$
4	$\frac{1}{(\sqrt{2})^3} \begin{pmatrix} 2\omega^2 - \omega^3 + 1 \\ 1 - \omega^3 \end{pmatrix}$	$\frac{1}{(\sqrt{2})^5} \begin{pmatrix} 3\sqrt{2} - 1 \\ \sqrt{2} + 1 \end{pmatrix}$

This also shows that given a circuit for state preparation of length n we can always find a circuit for unitary implementation of length $n + O(1)$ and vice versa.

3.2 Sequence for state preparation

We start with an example that illustrates the main ideas needed to prove Lemma 3.1.1. Next we formulate two results, Lemma 3.2.3 and Lemma 3.2.4, that the proof of Lemma 3.1.1 is based on. Afterwards, we describe the algorithm for decomposition of a unitary with entries in the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ into a sequence of H and T gates. Finally, we prove Lemma 3.2.3. The proof of Lemma 3.2.4 is more involved and it is shown in Section 3.3.

Let us consider a sequence of states $(\mathrm{HT})^n |0\rangle$. It is an infinite sequence, since in the Bloch sphere picture unitary HT corresponds to rotation over an angle that is an irrational fraction of π. Table 3.1 shows the first four elements of the sequence.

There are two features in this example that are important. First is that the power of $\sqrt{2}$ in the denominator of the entries is the same. We prove that the power of the denominator is the same in the general case of a unit vector with entries in ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$. The second feature is that the power of $\sqrt{2}$ in the denominator of $|z_n|^2$ increases by 1 after multiplication by HT. We show that in general, under additional assumptions, multiplication by H $\left(\mathrm{T}^k\right)$ cannot change the power of $\sqrt{2}$ in the denominator by more than

1. Importantly, under the same additional assumptions it is always possible to find such an integer k that the power increases or decreases by 1.

We need to clarify what we mean by power of $\sqrt{2}$ in the denominator, because, for example, it is possible to write $\frac{1}{\sqrt{2}}$ as $\frac{\zeta_8-\zeta_8^3}{2}$. As such, it may seem that the power of $\sqrt{2}$ in the denominator of a number from the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ is not well defined. To address this issue we consider the subring $\mathbb{Z}[\zeta_8]$ of ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ and the smallest denominator exponent. These definitions are also crucial for our proofs.

It is natural to extend the notion of divisibility to elements of $\mathbb{Z}[\zeta_8]$: x divides y when there exists x' from the ring $\mathbb{Z}[\zeta_8]$ such that $xx' = y$. Using the divisibility relation we can introduce the smallest denominator exponent and greatest dividing exponent.

Definition 3.2.1. *The* smallest denominator exponent, $\mathrm{sde}(z, x)$, *of base* $x \in \mathbb{Z}[\zeta_8]$ *with respect to* $z \in \mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ *is the smallest integer value* k *such that* $zx^k \in \mathbb{Z}[\zeta_8]$. *If there is no such* k, *the smallest denominator exponent is infinite.*

For example, $\mathrm{sde}(1/4, \sqrt{2}) = 4$ and $\mathrm{sde}\left(2\sqrt{2}, \sqrt{2}\right) = -3$. The smallest denominator exponent of base $\sqrt{2}$ is finite for all elements of the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$. The greatest dividing exponent is closely connected to sde.

Definition 3.2.2. *The* greatest dividing exponent, $\mathrm{gde}(z, x)$, *of base* $x \in \mathbb{Z}[\zeta_8]$ *with respect to* $z \in \mathbb{Z}[\zeta_8]$ *is the integer value* k *such that* x^k *divides* z *and* x *does not divide the quotient* $\frac{z}{x^k}$. *If no such* k *exists, the greatest dividing exponent is said to be infinite.*

For example, $\mathrm{gde}(z, \zeta_8^n) = \infty$, since ζ_8^n divides any element of $\mathbb{Z}[\zeta_8]$, and $\mathrm{gde}(0, x) = \infty$. For any non-zero base $x \in \mathbb{Z}[\zeta_8]$, when gde and sde are finite they are related via a simple formula:

$$\mathrm{sde}\left(\frac{z}{x^k}, x\right) = k - \mathrm{gde}(z, x). \tag{3.1}$$

This follows from the definitions of sde and gde. First, the assumption $\mathrm{gde}(z, x) = k_0$ implies $\mathrm{sde}\left(\frac{z}{x^k}, x\right) \geq k - k_0$. Second, the assumption $\mathrm{sde}\left(\frac{z}{x^k}, x\right) = k_0$ implies $\mathrm{gde}(z, x) \geq k + k_0$. Since both inequalities need to be satisfied simultaneously, this implies the equality. Further in the text we use notation $\mathrm{sde}(.)$ and $\mathrm{gde}(.)$ for sde and gde of base $\sqrt{2}$.

We are now ready to introduce two results that describe the change of the sde as a result of the application $\mathrm{H}(\mathrm{T})^k$ to a state:

$$\mathrm{HT}^k \begin{pmatrix} z \\ w \end{pmatrix} = \begin{pmatrix} \frac{z+w\zeta_8^k}{\sqrt{2}} \\ \frac{z-w\zeta_8^k}{\sqrt{2}} \end{pmatrix}.$$

Lemma 3.2.3. *Let* $\begin{pmatrix} z \\ w \end{pmatrix}$ *be a state with entries in* $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ *and let* $\operatorname{sde}\left(|z|^2\right) \geq 4$. *Then, for any integer* k*:*

$$-1 \leq \operatorname{sde}\left(\left|\frac{z + w\zeta_8^k}{\sqrt{2}}\right|^2\right) - \operatorname{sde}\left(|z|^2\right) \leq 1. \tag{3.2}$$

The next lemma states that for almost all unit vectors the difference in (3.2) achieves all possible values, when the power of ζ_8 is chosen appropriately.

Lemma 3.2.4. *Let* $\begin{pmatrix} z \\ w \end{pmatrix}$ *be a state with entries in* $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ *and let* $\operatorname{sde}\left(|z|^2\right) \geq 4$. *Then, for each number* $s \in \{-1, 0, 1\}$ *there exists an integer* $k \in \{0, 1, 2, 3\}$ *such that:*

$$\operatorname{sde}\left(\left|\frac{z + w\zeta_8^k}{\sqrt{2}}\right|^2\right) - \operatorname{sde}\left(|z|^2\right) = s.$$

These lemmas are essential for showing how to find a sequence of gates that prepares a state with entries in the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ given the initial state $|0\rangle$. Now we sketch a proof of Lemma 3.1.1. Later, in Lemma 3.2.6, we show that for arbitrary u and v from the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ the equality $|u|^2 + |v|^2 = 1$ implies $\operatorname{sde}(|u|^2) = \operatorname{sde}(|v|^2)$, when $\operatorname{sde}\left(|u|^2\right) \geq 1$ and $\operatorname{sde}\left(|v|^2\right) \geq 1$. Therefore, under the assumptions of Lemma 3.2.3, we may consider the sde of a single entry in any given state. Lemma 3.2.4 implies that we can prepare any state using H and T gates if we can prepare any state $\begin{pmatrix} z \\ w \end{pmatrix}$ such that $\operatorname{sde}(|z|^2) \leq 3$. The set of states with $\operatorname{sde}(|z|^2) \leq 3$ is finite and small. Therefore, we can exhaustively verify that all such states can be prepared using H and T gates given the initial state $|0\rangle$. In fact, we performed such verification using a breadth first search algorithm.

The statement of Lemma 3.2.4 remains true if we replace the set $\{0, 1, 2, 3\}$ by $\{0, -1, -2, -3\}$. Lemma 3.2.4 results in Algorithm 1 for decomposition of a unitary matrix with entries in the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ into a sequence of H and T gates. Its complexity is $O\left(\operatorname{sde}(|z|^2)\right)$, where z is an entry of the unitary. The idea behind the algorithm is as follows: given a 2×2 unitary U over the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ and $\operatorname{sde} \geq 4$, there is a value of k in $\{0, 1, 2, 3\}$ such that the multiplication by HT^k reduces the sde by 1. Thus, after $n - 4$ steps, we have expressed

$$U = \mathrm{HT}^{k_1}\mathrm{H} \ldots \mathrm{HT}^{k_{n-4}}U',$$

where any entry z' of U' has the property $\operatorname{sde}\left(|z'|^2\right) < 4$. The number of such unitaries is small enough to handle the decomposition of U' via employing a breadth-first search algorithm.

We use $n_{opt}(U)$ to define the smallest length of a circuit that implements U.

Corollary 3.2.5. *Algorithm 1 produces circuit of length $O(n_{opt}(U))$ and uses $O(n_{opt}(U))$ arithmetic operations. The number of bit operations it uses is $O(n^2_{opt}(U))$.*

Proof. Lemma 3.2.6, proved later in this section, implies that the value of $\mathrm{sde}\left(|\cdot|^2\right)$ is the same for all entries of U when the sde of at least one entry is greater than 0. For such unitaries we define $\mathrm{sde}^{|\cdot|^2}(U) = \mathrm{sde}\left(|z'|^2\right)$, where z' is an entry of U. The remaining special case is unitaries of the form

$$\begin{pmatrix} 0 & \zeta_8^k \\ \zeta_8^j & 0 \end{pmatrix}, \begin{pmatrix} \zeta_8^k & 0 \\ 0 & \zeta_8^j \end{pmatrix}.$$

We define $\mathrm{sde}^{|\cdot|^2}$ to be 0 for all of them. Consider a set $S_{opt,3}$ of optimal H and T circuits for unitaries with $\mathrm{sde}^{|\cdot|^2} \leq 3$. This is a finite set and therefore we can define N_3 to be the maximal number of gates in a circuit from $S_{opt,3}$. If we have a circuit that is optimal and its length is greater than N_3, the corresponding unitary must have $\mathrm{sde}^{|\cdot|^2} \geq 4$. Consider now a unitary U with an optimal circuit of length $n_g(U)$ that is larger than N_3. As it is optimal, all its subsequences are optimal and it does not include H^2. Let $N_{H,3}$ be the maximum of the number of Hadamard gates used by the circuits in $S_{opt,3}$. An optimal circuit for U includes at most $\left\lfloor \frac{n_g(U)-N_3}{2} \right\rfloor + N_{H,3}$ Hadamard gates and, by Lemma 3.2.3, $\mathrm{sde}^{|\cdot|^2}$ of the resulting unitary is less than or equal to $N_{H,3} + 3 + \left\lfloor \frac{n_g(U)-N_3}{2} \right\rfloor$. We conclude that for all unitaries except a finite set:

$$\mathrm{sde}^{|\cdot|^2}(U) \leq N_{H,3} + 3 + \left\lfloor \frac{n_g(U) - N_3}{2} \right\rfloor.$$

From the other side, the decomposition algorithm we described gives us the bound

$$n_g(U) \leq N_3 + 4 \cdot \mathrm{sde}^{|\cdot|^2}(U).$$

We conclude that $n_g(U)$ and $\mathrm{sde}^{|\cdot|^2}(U)$ are asymptotically equivalent. Therefore the algorithm's runtime is $O(n_g(U))$, because the algorithm performs $\mathrm{sde}^{|\cdot|^2}(U) - 4$ steps.

We note that to store U we need $O\left(\mathrm{sde}^{|\cdot|^2}(U)\right)$ bits and therefore the addition on each step of the algorithm requires $O\left(\mathrm{sde}^{|\cdot|^2}(U)\right)$ bit operations. Therefore we use $O(n^2_{opt}(U))$ bit operations in total. □

This proof illustrates the technique that we use in Section 3.5 to find a tighter connection between sde and the circuit implementation cost, in particular we prove that circuits produced by the algorithm are H- and T-optimal.

Algorithm 1 Decomposition of a unitary matrix with entries in the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$.

Input: Unitary $U = \begin{pmatrix} z_{00} & z_{01} \\ z_{10} & z_{11} \end{pmatrix}$ with entries in the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$.
$\mathbb{S}_3$ — table of all unitaries over the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$, such that *sde* of their entries is less than or equal to 3, $G_0 := \mathrm{H}$, $G_1 := \mathrm{TH}$, $G_2 := \mathrm{PH}$, $G_3 := \mathrm{ZT^\dagger H}$
Output: Sequence S_{out} of H and T gates that implements U.

```
S_out ← Empty
s ← sde(|z_00|^2)
while s>3 do
    state←unfound
    for all k ∈ {0,1,2,3} do
        while state = unfound do
            z'_00 ← top left entry of HT^{-k}U
            if sde(|z'_00|^2) = s − 1 then
                state = found
                add G_k to the end of S_out
                s ← sde(|z'_00|^2)
                U ← HT^{-k}U
            end if
        end while
    end for
end while
lookup sequence S_rem for U in S_3
add S_rem to the end of S_out
return S_out
```

We next prove Lemma 3.2.3. In Section 3.3 we use Lemma 3.2.3 to show that we can prove Lemma 3.2.4 by considering a large, but finite, number of different cases. We provide an algorithm (Algorithm 2) that verifies all these cases.

We now proceed to the proof of Lemma 3.2.3. We use equation (3.1) connecting sde

and gde together with the following properties of gde. For any base $x \in \mathbb{Z}[\zeta_8]$:

$$\operatorname{gde}(y + y', x) \geq \min(\operatorname{gde}(y, x), \operatorname{gde}(y', x)) \tag{3.3}$$

$$\operatorname{gde}\left(yx^k, x\right) = k + \operatorname{gde}(y, x) \quad \text{(base extraction)} \tag{3.4}$$

$$\operatorname{gde}(y, x) < \operatorname{gde}(y', x) \Rightarrow \operatorname{gde}(y + y', x) = \operatorname{gde}(y, x) \quad \text{(absorption)}. \tag{3.5}$$

It is also helpful to note that $\operatorname{gde}(y, x)$ is invariant with respect to multiplication by ζ_8 and complex conjugation of both x and y.

All these properties follow directly from the definition of gde; the first three are briefly discussed in Section 3.4. The condition $\operatorname{gde}(y, x) < \operatorname{gde}(y', x)$ is necessary for the third property. For example, $\operatorname{gde}\left(\sqrt{2} + \sqrt{2}, \sqrt{2}\right) \neq \operatorname{gde}\left(\sqrt{2}, \sqrt{2}\right)$.

There are also important properties specific to base $\sqrt{2}$. We use shorthand $\operatorname{gde}(\cdot)$ for $\operatorname{gde}\left(\cdot, \sqrt{2}\right)$:

$$\operatorname{gde}(x) = \operatorname{gde}\left(|x|^2, 2\right) \tag{3.6}$$

$$0 \leq \operatorname{gde}\left(|x|^2\right) - 2\operatorname{gde}(x) \leq 1 \tag{3.7}$$

$$\operatorname{gde}\left(\operatorname{Re}\left(\sqrt{2}xy^*\right)\right) \geq \left\lfloor \frac{1}{2}\left(\operatorname{gde}\left(|x|^2\right) + \operatorname{gde}\left(|y|^2\right)\right) \right\rfloor \tag{3.8}$$

$$\operatorname{gde}\left(|x|^2\right) = \operatorname{gde}\left(|y|^2\right) \Rightarrow \operatorname{gde}(x) = \operatorname{gde}(y). \tag{3.9}$$

Proofs of these properties are not difficult but tedious; furthermore, for completeness they are included in Section 3.4. We exemplify them here. In the second property, inequality (3.7), when $x = \zeta_8$ the left inequality becomes equality and for $\zeta_8 + 1$ the right one does. When we substitute $x = \zeta_8, y = \zeta_8 + 1$ in the second to last property, inequality (3.8), it turns into $0 = \left\lfloor \frac{1}{2} \right\rfloor$, so the floor function $r \mapsto \lfloor r \rfloor$ is necessary. For the third property it is important that $\operatorname{Re}\left(\sqrt{2}xy^*\right)$ is an element of $\mathbb{Z}[\zeta_8]$ when x, y itself belongs to the ring $\mathbb{Z}[\zeta_8]$. In contrast, $\operatorname{Re}(xy^*)$ is not always an element of $\mathbb{Z}[\zeta_8]$, in particular, when $x = \zeta_8, y = \zeta_8 + 1$. In general, $\operatorname{gde}(x) = \operatorname{gde}(y)$ does not imply $\operatorname{gde}\left(|x|^2\right) = \operatorname{gde}\left(|y|^2\right)$. For instance, $\operatorname{gde}(\zeta_8 + 1) = \operatorname{gde}(\zeta_8)$, but $|\zeta_8 + 1|^2 = 2 + \sqrt{2}$ and $|\zeta_8|^2 = 1$.

In the proof of Lemma 3.2.3 (page 34) we use $x = z\left(\sqrt{2}\right)^{\operatorname{sde}(z)}, y = w\left(\sqrt{2}\right)^{\operatorname{sde}(w)}$ that are elements of $\mathbb{Z}[\zeta_8]$. The next lemma shows an additional property that such x and y have.

Lemma 3.2.6. *Let z and w be elements of the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ such that $|z|^2 + |w|^2 = 1$ and $\operatorname{sde}(z) \geq 1$ or $\operatorname{sde}(w) \geq 1$, then $\operatorname{sde}(z) = \operatorname{sde}(w)$ and for elements $x = z\left(\sqrt{2}\right)^{\operatorname{sde}(z)}$ and $y = w\left(\sqrt{2}\right)^{\operatorname{sde}(w)}$ of the ring $\mathbb{Z}[\zeta_8]$ it holds that $\operatorname{gde}\left(|x|^2\right) = \operatorname{gde}\left(|y|^2\right) \leq 1$.*

Proof. Without loss of generality, suppose $\mathrm{sde}(z) \geq \mathrm{sde}(w)$. Using the relation in equation (3.1) between sde and gde, expressing z and w in terms of x and y, and substituting the result into equation $|z|^2 + |w|^2 = 1$, we obtain

$$|y|^2 \left(\sqrt{2}\right)^{2(\mathrm{sde}(z)-\mathrm{sde}(w))} = \left(\sqrt{2}\right)^{2\mathrm{sde}(z)} - |x|^2 .$$

Substituting $z = x/\left(\sqrt{2}\right)^{\mathrm{sde}(z)}$ into formula (3.1) relating sde and gde, we obtain $\mathrm{gde}(x) = 0$, and using one of the inequalities (3.7) connecting $\mathrm{gde}\left(|x|^2\right)$ and $\mathrm{gde}(x)$ we conclude that $\mathrm{gde}\left(|x|^2\right) \leq 1$. Similarly, $\mathrm{gde}\left(|y|^2\right) \leq 1$. We use the absorption property (3.5) of $\mathrm{gde}(\cdot)$ to write:

$$\mathrm{gde}\left(|y|^2 \left(\sqrt{2}\right)^{2(\mathrm{sde}(z)-\mathrm{sde}(w))}\right) = \mathrm{gde}\left(|x|^2\right).$$

Equivalently, using the base extraction property (3.4):

$$\mathrm{gde}\left(|y|^2\right) + 2\left(\mathrm{sde}(z) - \mathrm{sde}(w)\right) = \mathrm{gde}\left(|x|^2\right).$$

Taking into account $\mathrm{gde}\left(|x|^2\right) \leq 1$ and $\mathrm{gde}\left(|y|^2\right) \leq 1$, it follows that $\mathrm{sde}(z) = \mathrm{sde}(w)$. □

In the proof of Lemma 3.2.3 we turn inequality (3.2) for difference of sde into an inequality for difference of $\mathrm{gde}\left(|x|^2\right)$ and $\mathrm{gde}\left(|x+y|^2\right)$. The following lemma shows a basic relation between these quantities that we will use.

Lemma 3.2.7. *If x and y are elements of the ring $\mathbb{Z}[\zeta_8]$ such that $|x|^2 + |y|^2 = \left(\sqrt{2}\right)^m$, then*

$$\mathrm{gde}\left(|x+y|^2\right) \geq \min\left(m, 1 + \left\lfloor \frac{1}{2}\left(\mathrm{gde}\left(|x|^2\right) + \mathrm{gde}\left(|y|^2\right)\right)\right\rfloor\right).$$

Proof. The first step is to expand $|x+y|^2$ as $|x|^2 + |y|^2 + \sqrt{2}Re\left(\sqrt{2}xy^*\right)$. Next, we apply inequality (3.3) to the gde of the sum, and then the base extraction property (3.4) of the gde. We use equality $\mathrm{gde}\left(|x|^2 + |y|^2\right) = m$ to conclude that

$$\mathrm{gde}\left(|x+y|^2\right) \geq \min\left(m, 1 + \mathrm{gde}\left(Re\left(\sqrt{2}xy^*\right)\right)\right).$$

Finally, we use inequality (3.8) for $\mathrm{gde}\left(Re\left(\sqrt{2}xy^*\right)\right)$ to derive the statement of the lemma. □

Now we collected all tools required to prove Lemma 3.2.3.

Proof. Recall that, we are proving that for elements z and w of the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ and any integer k it is true that:

$$-1 \leq \operatorname{sde}\left(\left|\frac{z + w\zeta_8^k}{\sqrt{2}}\right|^2\right) - \operatorname{sde}\left(|z|^2\right) \leq 1, \text{ when } \operatorname{sde}\left(|z|^2\right) \geq 4.$$

Using Lemma 3.2.6 we can define $m = \operatorname{sde}(z) = \operatorname{sde}\left(w\zeta_8^k\right)$ and $x = \zeta_8^k z\left(\sqrt{2}\right)^m$ and $y = w\left(\sqrt{2}\right)^m$. Using the relation (3.1) between gde and sde, and the base extraction property (3.4) of gde we rewrite the inequality we are trying to prove as:

$$1 \leq \operatorname{gde}\left(|x + y|^2\right) - \operatorname{gde}\left(|x|^2\right) \leq 3.$$

It follows from Lemma 3.2.6 that $\operatorname{gde}\left(|x|^2\right) = \operatorname{gde}\left(|y|^2\right) \leq 1$. Taking into account $|x|^2 + |y|^2 = \sqrt{2}^{2m}$ and applying the inequality proved in Lemma 3.2.7 to x and y we conclude that:

$$\operatorname{gde}\left(|x + y|^2\right) \geq \min\left(2m, 1 + \operatorname{gde}\left(|x|^2\right)\right).$$

The condition $m \geq 4$ allows us to remove taking the minimum on the right hand side and replace it with $1 + \operatorname{gde}\left(|x|^2\right)$. This proves one of the two inequalities we are trying to show, $1 \leq \operatorname{gde}\left(|x + y|^2\right) - \operatorname{gde}\left(|x|^2\right)$. To prove the second inequality, $\operatorname{gde}\left(|x + y|^2\right) - \operatorname{gde}\left(|x|^2\right) \leq 3$, we apply Lemma 3.2.7 to the pair of elements of the ring $\mathbb{Z}[\zeta_8]$, $x + y$ and $x - y$. The conditions of the lemma are satisfied because $|x + y|^2 + |x - y|^2 = \sqrt{2}^{2(m+1)}$. Therefore:

$$\operatorname{gde}\left(4|x|^2\right) \geq \min\left(2(m+1), 1 + \left\lfloor \frac{1}{2}\left(\operatorname{gde}\left(|x + y|^2\right) + \operatorname{gde}\left(|x - y|^2\right)\right)\right\rfloor\right).$$

Using the base extraction property (3.4), we notice that $\operatorname{gde}\left(4|x|^2\right) = 4 + \operatorname{gde}\left(|x|^2\right)$. It follows from $m \geq 4$ that $2(m+1) \geq 4 + \operatorname{gde}\left(|x|^2\right)$. As such, we can again remove the minimization and simplify the inequality to:

$$3 + \operatorname{gde}\left(|x|^2\right) \geq \left\lfloor \frac{1}{2}\left(\operatorname{gde}\left(|x + y|^2\right) + \operatorname{gde}\left(|x - y|^2\right)\right)\right\rfloor.$$

To finish the proof it suffices to show that $\operatorname{gde}\left(|x + y|^2\right) = \operatorname{gde}\left(|x - y|^2\right)$. We establish an upper bound for $\operatorname{gde}\left(|x + y|^2\right)$ and use the absorption property (3.5) of gde. Using the non-negativity of gde and the definition of the floor function we get:

$$2\left(3 + \operatorname{gde}\left(|x|^2\right)\right) + 1 \geq \operatorname{gde}\left(|x + y|^2\right).$$

Since $\text{gde}\left(|x|^2\right) \leq 1$, $\text{gde}\left(|x+y|^2\right) \leq 9$. Observing that $2\,(m+1) > 9$ we confirm that

$$\text{gde}\left(|x-y|^2\right) = \text{gde}\left(\sqrt{2}^{2(m+1)} - |x+y|^2\right) = \text{gde}\left(|x+y|^2\right).$$

□

To prove Lemma 3.2.4 it suffices to show that $\text{gde}(\left|x+\zeta_8^k y\right|^2) - \text{gde}(|x|^2)$ achieves all values in the set $\{1,2,3\}$ as k varies over all values in the range from 0 to 3. We can split this into two cases: $\text{gde}(|x|^2) = 1$ and $\text{gde}(|x|^2) = 0$. We need to check if $\text{gde}(\left|x+\zeta_8^k y\right|^2)$ belongs to $\{1,2,3\}$ or $\{2,3,4\}$. Therefore, it is important to describe these conditions in terms of x and y. This is accomplished in the next Section.

3.3 Quadratic forms and greatest dividing exponent

We first clarify why it is enough to check a finite number of cases to prove Lemma 3.2.4. Recall how the lemma can be restated in terms of the elements of the ring $\mathbb{Z}[\zeta_8]$. Next we illustrate why we can achieve a finite number of cases with a simple example using integer numbers $\mathbb{Z}$. Then we show how this idea can be extended to the elements of the ring $\mathbb{Z}[\zeta_8]$ that are real (that is, with imaginary part equal to zero). Finally, in the proof of Lemma 3.2.4, we identify a set of cases that we need to check and provide an algorithm to perform it.

As discussed at the end of the previous Section, to prove Lemma 3.2.4 one can consider elements x and y of the ring $\mathbb{Z}[\zeta_8]$ such that $|x|^2 + |y|^2 = 2^m$ for $m \geq 4$. We know from Lemma 3.2.3 that there are three possibilities in each of the two cases:

- when $\text{gde}(|x|^2) = 0$, $\text{gde}(\left|x+\zeta_8^k y\right|^2)$ equals to $1, 2$, or 3,
- when $\text{gde}(|x|^2) = 1$, $\text{gde}(\left|x+\zeta_8^k y\right|^2)$ equals $2, 3$, or 4.

We want to show that each of these possibilities is achievable for a specific choice of $k \in \{0,1,2,3\}$.

We illustrate the idea of the reduction to a finite number of cases with an example. Suppose we want to describe two classes of integer numbers:

- integer a such that the $\text{gde}\,(a^2, 2) = 2$,

- integer a such that the $\mathrm{gde}\,(a^2, 2) > 2$.

It is enough to know $a^2 \, mod \, 2^3$ to decide which class a belongs to. Therefore we can consider 8 residues $a \, mod \, 2^3$ and find the classes to which they belong. We extend this idea to the real elements of the ring $\mathbb{Z}[\zeta_8]$, being elements of the ring $\mathbb{Z}[\zeta_8]$ that are equal to their own real part. Afterwards we apply the result to $\left|x + \zeta_8^k y\right|^2$, that is a real element of $\mathbb{Z}[\zeta_8]$.

We note that the real elements of $\mathbb{Z}[\zeta_8]$ are of the form $a+\sqrt{2}b$ where a and b are integer numbers. An important preliminary observation, that follows from the irrationality of $\sqrt{2}$, is that for any integer number c

$$\mathrm{gde}\,(c) = 2\mathrm{gde}\,(c, 2)\,. \tag{3.10}$$

The next proposition gives a condition equivalent to $\mathrm{gde}\left(a + \sqrt{2}b\right) = k$, expressed in terms of $\mathrm{gde}\,(a, 2)$ and $\mathrm{gde}\,(b, 2)$:

Proposition 3.3.1. *Let a and b be integer numbers. There are two possibilities:*

- $\mathrm{gde}\left(a + \sqrt{2}b\right)$ *is even if and only if* $\mathrm{gde}\,(b, 2) \geq \mathrm{gde}\,(a, 2)$*; in this case,* $\mathrm{gde}\,(a, 2) = \mathrm{gde}\left(a + \sqrt{2}b\right)/2$.
- $\mathrm{gde}\left(a + \sqrt{2}b\right)$ *is odd if and only if* $\mathrm{gde}\,(b, 2) < \mathrm{gde}\,(a, 2)$*; in this case,* $\mathrm{gde}\,(b, 2) = \left(\mathrm{gde}\left(a + \sqrt{2}b\right) - 1\right)/2$.

Proof. Consider the case when $\mathrm{gde}\,(b, 2) < \mathrm{gde}\,(a, 2)$. Observing, from equation (3.10), that $\mathrm{gde}\,(a)$ is always even, $\mathrm{gde}\,(a) > \mathrm{gde}\left(\sqrt{2}b\right)$, and by the absorption property (3.5) of gde we have $\mathrm{gde}\left(a + \sqrt{2}b\right) = \mathrm{gde}\left(\sqrt{2}b\right)$. Using the base extraction property (3.4) of gde and the relation (3.10) between $\mathrm{gde}\,(\cdot)$ and $\mathrm{gde}\,(\cdot, 2)$ for integers we obtain $\mathrm{gde}\left(a + \sqrt{2}b\right) = 1 + 2\mathrm{gde}\,(b, 2)$. The other case similarly implies $\mathrm{gde}\left(a + \sqrt{2}b\right) = 2\mathrm{gde}\,(a, 2)$. In terms of real elements of the ring $\mathbb{Z}[\zeta_8]$, this results in the following relations:

$$A_1 = \{\mathrm{gde}\,(b, 2) < \mathrm{gde}\,(a, 2)\} \subseteq B_1 = \left\{\mathrm{gde}\left(a + \sqrt{2}b\right) \text{ is even}\right\},$$
$$A_2 = \{\mathrm{gde}\,(b, 2) \geq \mathrm{gde}\,(a, 2)\} \subseteq B_2 = \left\{\mathrm{gde}\left(a + \sqrt{2}b\right) \text{ is odd}\right\}.$$

We note that each pair of sets $\{A_1, A_2\}$ and $\{B_1, B_2\}$ defines a partition of real elements of the ring $\mathbb{Z}[\zeta_8]$. This completes the proof since for partitions $\{A_1, A_2\}$ and $\{B_1, B_2\}$ of some set, the inclusions $A_1 \subseteq B_1, A_2 \subseteq B_2$ imply $A_1 = B_1$ and $A_2 = B_2$. □

To express $\left|x+\zeta_8^k y\right|^2$ in the form $a+\sqrt{2}b$ in a concise way, we introduce two quadratic forms $P(\cdot)$ and $Q(\cdot)$ with the property:

$$|x|^2 = P(x) + \sqrt{2}Q(x). \tag{3.11}$$

Given that x, an element of $\mathbb{Z}[\zeta_8]$, can be expressed in terms of the integer number coordinates as follows, $x = x_0 + x_1\zeta_8 + x_2\zeta_8^2 + x_3\zeta_8^3$, we define the quadratic forms as:

$$P(x) := x_0^2 + x_1^2 + x_2^2 + x_3^2, \tag{3.12}$$

$$Q(x) := x_0(x_1 - x_3) + x_2(x_1 + x_3). \tag{3.13}$$

Let us rewrite the equality $\operatorname{gde}\left(|x+y|^2\right) = 4$ in terms of these quadratic forms and the gde of base 2. Using Proposition 3.3.1 we can write:

$$\operatorname{gde}\left(P\left(x+\zeta_8^k y\right), 2\right) = 2,$$

$$\operatorname{gde}\left(Q\left(x+\zeta_8^k y\right), 2\right) \geq 2.$$

Similar to the example given at the beginning of this section, we see that it suffices to know the values of the quadratic forms modulo 2^3. To compute them, it suffices to know the values of the integer coefficients of x and y modulo 2^3. This follows from the expression of the product $\zeta_8 y$ in terms of the integer number coefficients:

$$\zeta_8\left(y_1 + y_2\zeta_8 + y_3\zeta_8^2 + y_4\zeta_8^3\right) = -y_4 + y_1\zeta_8 + y_2\zeta_8^2 + y_3\zeta_8^3,$$

and from the following two observations:

- integer number coefficients of the sum of two elements of the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ is the sum of their integer number coefficients,
- for any element of $\mathbb{Z}[\zeta_8]$, x, the values of quadratic forms $P(x)$ and $Q(x)$ modulo 2^3 are defined by the values modulo 2^3 of the integer number coefficients of x.

In summary, to check the second part of Lemma 3.2.3 we need to consider all possible values for the integer coefficients of x and y modulo 2^3. There are two additional constraints on them. The first one is $|x|^2 + |y|^2 = 2^m$. Since we assumed $m \geq 4$, we can write necessary conditions to satisfy this constraint, in terms of the quadratic forms, as:

$$P(x) \equiv -P(y) \left(\operatorname{mod} 2^3\right),$$

$$Q(x) \equiv -Q(y) \pmod{2^3}.$$

The second constraint is $\text{gde}\left(|x|^2\right) = \text{gde}\left(|y|^2\right)$ and $\text{gde}\left(|x|^2\right) \leq 1$. To check it, we use the same approach as in the example with $\text{gde}\left(|x+y|^2\right) = 4$.

We have now introduced the necessary notions required to prove Lemma 3.2.4.

Proof. Our proof is an exhaustive verification, assisted by a computer search. We rewrite the statement of the lemma formally as follows:

$$\left.\begin{array}{l} \mathcal{G}_j = \left\{ (x,y) \in \mathbb{Z}[\zeta_8] \times \mathbb{Z}[\zeta_8] \;\middle|\; \begin{array}{c} \exists m \geq 4 \text{ s.t. } |x|^2 + |y|^2 = 2^m, \\ \text{gde}(x) = \text{gde}(y) = j \end{array} \right\}, j \in \{0,1\}, \\ \text{for all } (x,y) \in \mathcal{G}_j, \text{for all } s \in \{1,2,3\} \text{ there exists } k \in \{0,1,2,3\} \\ \text{such that } \text{gde}\left(\left|x + \zeta_8^k y\right|^2\right) = s + j. \end{array}\right\} \tag{3.14}$$

The sets $\mathcal{G}_j$ are infinite, so it is impossible to perform the check directly. As we illustrated with an example, equality $\text{gde}\left(\left|x + \zeta_8^k y\right|^2\right) = s+j$ depends only on the values of the integer coordinates of x and y modulo 2^3. If the sets $\mathcal{G}_j$ were also defined in terms of the residues modulo 2^3 we could just check the lemma in terms of equivalence classes corresponding to different residuals. More precisely, the equivalence relation $\sim$ we would use is:

$$\sum_{p=0}^{3} x_p \zeta_8^p \sim \sum_{p=0}^{3} y_p \zeta_8^p \overset{def}{\iff} \text{ for all } p \in \{0,1,2,3\} : x_p \equiv y_p \pmod{2^3}.$$

To address the issue, we introduce sets $\mathcal{Q}_j$ that include $\mathcal{G}_j$ as subsets:

$$\mathcal{Q}_j = \left\{ (x,y) \in \mathbb{Z}[\zeta_8] \times \mathbb{Z}[\zeta_8] \;\middle|\; \begin{array}{c} \text{gde}(x) = \text{gde}(y) = j \\ P(x) + P(y) \equiv 0 \pmod{2^3} \\ Q(x) + Q(y) \equiv 0 \pmod{2^3} \end{array} \right\}, j \in \{0,1\}.$$

Therefore, in terms of the equivalence classes with respect to the above defined relation $\sim$ the more general problem can be verified in a finite number of steps. However, the number of equivalence classes is large. This is why we employ a computer search that performs verification of all cases. To rewrite (3.14) into conditions in terms of the equivalence classes it suffices to replace $\mathcal{G}_j$ by $\mathcal{Q}_j$, replace x and y by their equivalence classes, and replace $\mathbb{Z}[\zeta_8]$ by the set of equivalence classes $\mathbb{Z}[\zeta_8] / \sim$.

Algorithm 2 verifies Lemma 3.2.4. We use bar (e.g., $\overline{x}$ and $\overline{y}$) to represent 4-dimensional vectors with entries in $\mathbb{Z}_8$, the ring of residues modulo 8. The definition of bilinear forms, multiplication by ζ_8 and the relations $\text{gde}\left(|\cdot|^2\right) = 1,2,3,4$ extend to $\overline{x}$ and $\overline{y}$. We implemented Algorithm 2 and the result of its execution is *true*. This completes the proof. □

Algorithm 2 Verification of Lemma 3.2.4.

Output: Returns *true* if the statement of Lemma 3.2.4 is correct; otherwise, returns $false$.

▷ Here, $G_{j,a,b}$ is the set of all residue vectors $\overline{x}$ such that $\mathrm{gde}(\overline{x}) = j, P(\overline{x}) = a, Q(\overline{x}) = b$.

for all $x_1, x_2, x_3, x_4 \in \{0, \ldots, 7\}$ **do** ▷ generate possible residue vectors;
  $\overline{x} \leftarrow (x_1, x_2, x_3, x_4)$
  $j \leftarrow \mathrm{gde}(|\overline{x}|^2)$, $a \leftarrow P(\overline{x})$, $b \leftarrow Q(\overline{x})$
  if $j \in \{0, 1\}$ **then**
    add $\overline{x}$ to $G_{j,a,b}$
  end if
end for
for all $j \in \{0, 1\}, a_x \in \{0, 7\}, b_x \in \{0, 7\}$ **do**
  $a_y \leftarrow -a_x \bmod 8$, $b_y \leftarrow -b_x \bmod 8$ ▷ consider only those pairs that
  for all $(\overline{x}, \overline{y}) \in G_{j,a_x,b_x} \times G_{j,a_y,b_y}$ **do** ▷ satisfy necessary conditions;
    for all $d \in \{1, 2, 3\}$ **do**
      state $\leftarrow$ unfound
      for all $k \in \{0, 1, 2, 3\}$ **do**
        $\overline{t} \leftarrow \overline{x} + \zeta_8^k \overline{y}$
        if $\mathrm{gde}(|\overline{t}|^2) = \mathrm{d} + \mathrm{j}$ **then**
          state $\leftarrow$ found
        end if
      end for
      if state = unfound **then**
        return $false$
      end if
    end for
  end for
end for
return *true*

3.4 Properties of the greatest dividing exponent

Here we prove properties of the greatest dividing exponent that was defined and used in Section 3.2. We first discuss the base extraction property (3.4) of gde and then proceed to the proof of special properties of gde $(\cdot, \sqrt{2})$. The base extraction property simplifies proofs of all statements related to gde $(\cdot, \sqrt{2})$.

Proposition 3.4.1 (Base extraction property)**.** *If $x, y \in \mathbb{Z}[\zeta_8]$, then for any non negative integer number k*

$$\operatorname{gde}\left(yx^k, x\right) = k + \operatorname{gde}(y, x).$$

Proof. Follows directly from the definition of gde. □

The base extraction property together with non-negativity of gde provide a simple formula to lower bound the value of gde: if x^k divides y then $\operatorname{gde}(y, x) \geq k$. Inequality for gde of a sum (3.3) follows directly from this—$x^{\min(\operatorname{gde}(y,x),\operatorname{gde}(y',x))}$ divides $y + y'$. The proof of absorption property (3.5) follows easily, as well.

Now we prove properties of gde specific to base $\sqrt{2}$. Instead of proving them for all elements of $\mathbb{Z}[\zeta_8]$ it suffices to prove them for elements of $\mathbb{Z}[\zeta_8]$ that are not divisible by $\sqrt{2}$. We illustrate this with an example $\operatorname{gde}\left(x, \sqrt{2}\right) = \operatorname{gde}\left(|x|^2, 2\right)$. We can always write $x = x'\left(\sqrt{2}\right)^{\operatorname{gde}(x)}$. By the definition of gde, $\sqrt{2}$ does not divide x'. By substituting the expression for x into $\operatorname{gde}\left(|x|^2, 2\right)$ and then using the base extraction property we get:

$$\operatorname{gde}\left(|x|^2, 2\right) = \operatorname{gde}\left(|x'|^2, 2\right) + \operatorname{gde}\left(x, \sqrt{2}\right).$$

Therefore, it suffices to show that $\operatorname{gde}\left(|x'|^2, 2\right) = 0$ when $\sqrt{2}$ does not divide x', or, equivalently, when $\operatorname{gde}(x') = 0$.

The quadratic forms defined in Section 3.3 will be a useful tool for later proofs. Bilinear forms that generalize them are important for the proof of relation for $\operatorname{gde}(\operatorname{Re}(xy^*))$. Effectively, we only need the values of the mentioned forms modulo 2. For this reason, we also introduce forms that are equivalent modulo 2 and more convenient for the proofs.

We define function $F(\cdot, \cdot)$ for $x, y \in \mathbb{Z}[\zeta_8]$ as follows:

$$F(x, y) := x_0y_0 + x_1y_1 + x_2y_2 + x_3y_3.$$

Note that the following equality holds, and provides some intuition behind the choice to introduce $F(\cdot, \cdot)$:

$$\operatorname{Re}(xy^*) = F(x, y) + \frac{1}{\sqrt{2}} F\left(\sqrt{2}x, y\right).$$

Using formula $\sqrt{2} = \zeta_8 - \zeta_8^3$ we can rewrite multiplication by $\sqrt{2}$ as a linear operator:

$$\sqrt{2}x = \sqrt{2}(x) : x_0 + x_1\zeta_8 + x_2\zeta_8^2 + x_3\zeta_8^3 \mapsto (x_1 - x_3) + (x_0 + x_2)\zeta_8 + (x_1 + x_3)\zeta_8^2 + (x_2 - x_0)\zeta_8^3. \tag{3.15}$$

In particular, it is easy to verify that:

$$F\left(\sqrt{2}x, y\right) = (x_1 - x_3)\, y_0 + (x_0 + x_2)\, y_1 + (x_1 + x_3)\, y_2 + (x_2 - x_0)\, y_3,$$

and, substituting $y = x$,

$$F\left(\sqrt{2}x, x\right) = 2\,(x_1 - x_3)\, x_2 + 2\,(x_1 + x_3)\, x_0 = 2Q\,(x)\,,$$

which corresponds to the earlier definition shown in equation (3.13). The definition of $F\,(\cdot,\cdot)$ written for $x = y$ results in an earlier definition (3.12). This shows how $F\,(\cdot,\cdot)$ generalizes and ties together previously introduced $P(\cdot)$ and $Q(\cdot)$.

Furthermore, in modulo 2 arithmetic the following expressions hold true:

$$P\,(x) \equiv (x_1 + x_3) + (x_0 + x_2)\,(\mathrm{mod}\,2) \tag{3.16}$$

$$Q\,(x) \equiv (x_1 + x_3)\,(x_0 + x_2)\,(\mathrm{mod}\,2) \tag{3.17}$$

$$F\left(\sqrt{2}x, y\right) \equiv (x_1 + x_3)\,(y_0 + y_2) + (x_0 + x_2)\,(y_1 + y_3)\,(\mathrm{mod}\,2)\,. \tag{3.18}$$

It is easy to verify these equations by expanding the left and right hand sides.

The next proposition shows how we use equivalent quadratic and bilinear forms.

Proposition 3.4.2. *If* $\mathrm{gde}\,(x) = 0$ *there are only two alternatives:*

- $P\,(x)$ *is even and* $Q\,(x)$ *is odd,*
- $P\,(x)$ *is odd and* $Q\,(x)$ *is even.*

Proof. The equality $\mathrm{gde}\,(x) = 0$ implies that 2 does not divide $\sqrt{2}x$. Using expression (3.15) for $\sqrt{2}x$ in terms of integer coefficients we conclude that at least one of the four numbers $x'_1 \pm x'_3$, $x'_0 \pm x'_2$ must be odd. Suppose that $x'_1 + x'_3$ odd. Using formulas (3.16,3.17) we conclude that the values of $P(x)$ and $Q(x)$ must have different parity. The remaining three cases are similar. □

An immediate corollary is: $\mathrm{gde}\,(x) = 0$ implies $\mathrm{gde}\left(|x|^2, 2\right) = 0$. To show this it suffices to use expression (3.11) for $|x|^2$ in terms of quadratic forms.

We can also conclude that $\sqrt{2}$ divides x if and only if 2 divides $|x|^2$. Sufficiency follows from the definition of gde. To prove that 2 divides $|x|^2$ implies $\sqrt{2}$ divides x, we assume that 2 divides $|x|^2$ and $\sqrt{2}$ does not divide x, which leads to a contradiction. This also results in the inequality $\mathrm{gde}\left(|x|^2\right) \leq 1$ when $\mathrm{gde}\,(x) = 0$.

We use the next two propositions to prove the inequality for $\mathrm{Re}\left(\sqrt{2}xy^*\right)$.

Proposition 3.4.3. *Let* $\operatorname{gde}(x) = 0$*:*

- *if* $\sqrt{2}$ *divides* $|x|^2$ *then* $P(x)$ *is even and* $Q(x)$ *is odd,*
- *if* $\sqrt{2}$ *does not divide* $|x|^2$ *then* $P(x)$ *is odd and* $Q(x)$ *is even.*

Proof. As discussed, the previous proposition implies that $\sqrt{2}$ divides y if and only if 2 divides $|y|^2$. We apply this to $|x|^2$. By expressing $|x|^4$ in terms of the quadratic forms we get:

$$|x|^4 = P(x)^2 + 2Q(x)^2 + 2\sqrt{2}P(x)Q(x).$$

We see that 2 divides $|x|^4$ if and only if 2 divides $P(x)^2$, or, equivalently, $\sqrt{2}$ divides $|x|^2$ if and only if $P(x)$ even. Using the previous proposition again, this time for x, we obtain the required result. □

Proposition 3.4.4. *Let* $\operatorname{gde}(x) = 0$ *and* $\operatorname{gde}(y) = 0$*. If* $\sqrt{2}$ *divides* $|x|^2$ *and* $\sqrt{2}$ *divides* $|y|^2$ *then* $\sqrt{2}$ *divides* $\operatorname{Re}\left(\sqrt{2}xy^*\right)$*.*

Proof. By the previous proposition, $\sqrt{2}$ divides $|x|^2$ and $\sqrt{2}$ divides $|y|^2$ implies that $Q(x)$ and $Q(y)$ are odd. Formula (3.17) implies that in terms of the integer number coefficients of x and y integer numbers $x_1 + x_3$, $x_0 + x_2$, $y_1 + y_3$, $y_0 + y_2$, are all odd. Expressing $\operatorname{Re}\left(\sqrt{2}xy^*\right)$ in terms of $F(\cdot,\cdot)$,

$$\operatorname{Re}\left(\sqrt{2}xy^*\right) = \sqrt{2}F(x,y) + F\left(\sqrt{2}x, y\right),$$

and using expression (3.18), we conclude that 2 divides $F\left(\sqrt{2}x, y\right)$; therefore $\sqrt{2}$ divides $\operatorname{Re}\left(\sqrt{2}xy^*\right)$. □

Now we show $\operatorname{gde}\left(\operatorname{Re}\left(\sqrt{2}xy^*\right)\right) \geq \left\lfloor \frac{1}{2}\left(\operatorname{gde}\left(|x|^2\right) + \operatorname{gde}\left(|y|^2\right)\right)\right\rfloor$. As we discussed in the beginning, we can assume $\operatorname{gde}(x) = 0$ and $\operatorname{gde}(y) = 0$ without loss of generality. This implies $\operatorname{gde}\left(|x|^2\right) \leq 1$ and $\operatorname{gde}\left(|y|^2\right) \leq 1$. The expression $\left\lfloor \frac{1}{2}\left(\operatorname{gde}\left(|x|^2\right) + \operatorname{gde}\left(|y|^2\right)\right)\right\rfloor$ can only be equal to 0 or 1. The second one is only possible when $\operatorname{gde}\left(|x|^2\right) = 1$ and $\operatorname{gde}\left(|y|^2\right) = 1$, in which case the previous proposition implies $\operatorname{gde}\left(\operatorname{Re}\left(\sqrt{2}xy^*\right)\right) \geq 1$. In the first case inequality is true because of the non-negativity of gde.

We can also use quadratic forms to describe all numbers z in the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ such that $|z|^2 = 1$. Seeking a contradiction, suppose $\operatorname{sde}(z) \geq 1$. We can always write $z = \frac{x}{\left(\sqrt{2}\right)^k}$ where $k = \operatorname{sde}(z)$ and $\operatorname{gde}(x) = 0$. From the other side $|x|^2 = P(x) + \sqrt{2}Q(x) = 2^k$. Thus

we have a contradiction with the statement of Proposition 3.4.2. We conclude that z is an element of $\mathbb{Z}[\zeta_8]$. Therefore we can write z in terms of its integer number coordinates, $z = z_0 + z_1\zeta_8 + z_2\zeta_8^2 + z_3\zeta_8^3$. Equality $|z|^2 = 1$ implies that $F(z, z) = z_0^2 + z_1^2 + z_2^2 + z_3^2 = 1$. Taking into account that z_j are integer numbers we conclude that $z \in \{\zeta_8^k, k = 0, \ldots, 7\}$.

3.5 Exact synthesis algorithm optimality

Proof of Theorem 3.0.5. **1: H-optimality.** Using brute force, we explicitly verified that the set of H-optimal circuits with precisely 3 Hadamard gates is equal to the set of all unitaries over the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ with $\text{sde}(|z|^2) = 4$. Suppose we have a unitary U with $\text{sde}(|z|^2) = n \geq 4$. With the help of Algorithm 1 we can reduce it to a unitary with $\text{sde}(|z|^2) = 4$ while using $n - 4$ Hadamard gates to accomplish this. As such, there exists a circuit with $n - 1$ Hadamard gates that implements U.

Now consider an H-optimal circuit C that implements U. Using brute force, we established that if C has less than 3 Hadamard gates, then $\text{sde}(|z|^2)$ is less than 4. Suppose C contains $m \geq 3$ Hadamard gates. Its prefix, containing 3 Hadamard gates, must also be H-optimal, and therefore $\text{sde}(|z|^2)$ of the corresponding unitary is 4. Now, using the inequality from Lemma 3.2.3, we conclude that $\text{sde}(|z|^2)$ of the unitary corresponding to C is less than $m + 1$. This implies $n \leq m + 1$. Since we already know that $m \leq n - 1$, we may conclude that $m = n - 1$ and m is the number of Hadamard gates in the circuit produced by Algorithm 1 in combination with the brute force step.

2: T-optimality. To prove T-optimality we introduce a normal form for circuits over $\mathcal{G}$. We call a circuit HT-normal if there is precisely one T gate between every two H gates and, symmetrically, precisely one H gate between every two T gates. It is not difficult to modify Algorithm 1 to produce a circuit in HT-normal form while preserving its H-optimality. To accomplish that, first, recall that $\text{HT}^3 = \text{HZT}^\dagger$ and that all circuits generated during the brute force stage are both H-optimal and in the HT-normal form. Second, any circuit produced by the algorithm is H-optimal and does not contain a non-H-optimal (up to global phase) subcircuit $\text{HT}^2\text{H} = \text{HPH} = \zeta_8\text{PHP}$.

We will show that any H-optimal circuit in HT-normal form is also T-optimal. We start with a special case of HT-normal circuits—those that begin and end with the Hadamard gate, in other words, those that can be written as $\text{HS}_1\text{H}\ldots\text{HS}_k\text{H}$, and are H-optimal. Let U be a unitary corresponding to this circuit. Due to HT-normality, each S_i contains exactly one T gate, the number of T gates in the circuit is k, and $h(U) = k + 1$; therefore, $t(U) \leq h(U) - 1$. To prove that $t(U) = h(U) - 1$, it suffices to show that $t(U) \geq h(U) - 1$.

Let us write a T-optimal circuit for U as $C_0TC_1T\ldots TC_k$. Each subcircuit C_k implements a unitary from the Clifford group. Each unitary from the single-qubit Clifford group can be implemented using at most one H gate (recall, that we are concerned with the implementations up to global phase), therefore $h(U) \leq t(U) + 1$, as required.

In the general case, consider a circuit obtained by Algorithm 1 and implementing a unitary V with $h(V) \geq 3$ that is H-optimal and written in HT-normal form and show that it is T-optimal. We can write it as $S_0HS_1H\ldots HS_kHS_{k+1}$. By Lemma 3.2.4 we can always find such l and j that $C :=HT^lS_0HS_1H\ldots HS_kH\ S_{k+1}T^jH$ is also an H-optimal circuit. Indeed, according to Lemma 3.2.4, using the connection between $\mathrm{sde}(\cdot)$ and $h(\cdot)$ described in the first part of the proof, given $h(V) = k + 1$ we can always find l such that $h(HT^lV) = k + 2$. From the other side, circuit $HT^lS_0HS_1H\ldots HS_kHS_{k+1}$ contains $k + 2$ Hadamard gates and therefore is H-optimal. We repeat the same procedure to find j.

Considering the different possible values of l and j allows to complete the proof of the Theorem. This is somewhat tedious, and we illustrate how to handle different cases with a representative example of $l = 3$ and $j = 2$. In such a case, we can rewrite circuit C as $C' =HTPS_0HS_1H\ldots HS_kHS_{k+1}PH$. We conclude that S_0 must have zero T gates and S_{k+1} must have one T gate. Otherwise subcircuits $HTPS_0H$ and $HS_{k+1}PH$ will not be H-optimal. As such, we reduced the problem to the special case considered above, therefore circuit C' is T-optimal and $S_0HS_1H\ldots HS_kHS_{k+1}$ is T-optimal as its subcircuit. In the general case, the following formula may be developed $t\,(V) = h(U) - 1 + (l \mod 2) + (j \mod 2)$. □

3.6 Experimental results

Table 3.2 summarizes the results of first obtaining an approximation of the given rotation matrix by a unitary over the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ using our implementation of the Solovay-Kitaev algorithm [18, 44], and then decomposing it into a circuit using the exact synthesis Algorithm 1 presented in this chapter. We note that the implementation of our synthesis Algorithm 1 (runtimes found in the column t_{decomp}) is significantly faster than the implementation of the Solovay-Kitaev algorithm used to approximate the unitary (runtimes reported in the column t_{approx}). Furthermore, we were able to calculate approximating circuits using 5 to 7 iterations of the Solovay-Kitaev algorithm followed by our synthesis algorithm. The total runtime to approximate and decompose unitaries ranged from approximately 11 to 600 seconds, correspondingly, featuring best approximating errors on the order of 10^{-50}, and circuits with up to millions of gates. Actual specifications of all circuits reported, as well as those synthesized but not explicitly included in the Table 3.2, due to space constraints, may be obtained from http://qcirc.iqc.uwaterloo.ca/.

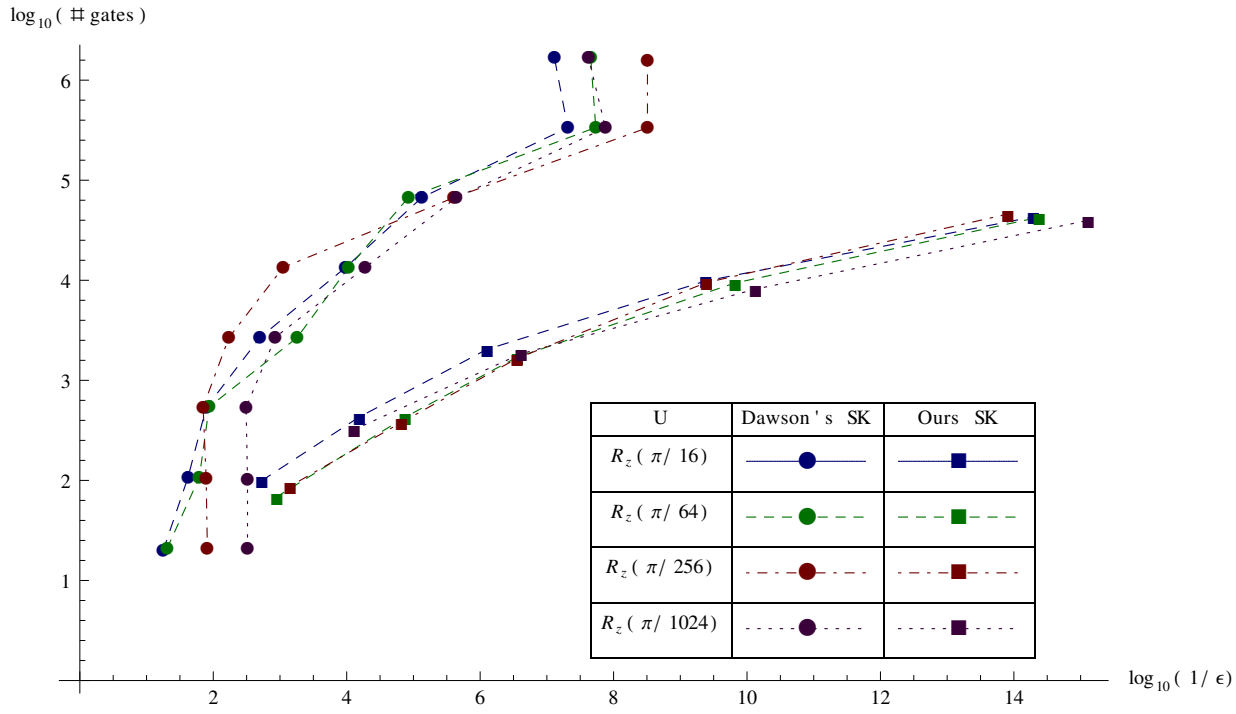

Figure 3.1: Comparison between ours and Dawson's implementations of the Solovay-Kitaev algorithm. Vertical axis shows $\log_{10}$ of the number of gates and horizontal axis shows $\log_{10}(1/\epsilon)$, where ϵ is the projective trace distance between unitary and its approximation.

On each step Algorithm 1 chooses from one of four small circuits: H, HT, HT^2 (=HP), and HT^3 to reduce sde. In practice, the Pauli-Z gate is often easier to implement than either Phase or T gate. The cost of $P^\dagger$ and $T^\dagger$ gates is usually the same as that of the respective P and T gates. We took this into account by writing circuit HT^3 using an equivalent and cheaper form $HZT^\dagger$. This significantly reduces the number of Phase gates required to implement a unitary. If there is no preference between the choice of P or Z, or P is preferred to Z, the HPT could be used in place of HT^3.

The RAM memory requirement during the unitary approximation stage for our implementation is 2.1GB. In our experiments we used a single core of the Intel Core i7-2600 (3.40GHz) processor.

We performed a comparison to Dawson's implementation of the Solovay-Kitaev algorithm (see Figure 3.1) available at `http://gitorious.org/quantum-compiler/`. We ran Dawson's code using the gate library {H, T} with the maximal sequence length equal to

Table 3.2: Results of the approximation of $\mathrm{R}_z(\varphi) = \begin{pmatrix} e^{-i\varphi} & 0 \\ 0 & e^{i\varphi} \end{pmatrix}$ by our implementation. Column N_I contains the number of iterations used by the Solovay-Kitaev algorithm, n_g—total number of gates (sum of the next four columns), n_T—number of T and $\mathrm{T}^\dagger$ gates, n_H—number of Hadamard gates, n_P—number of P and $\mathrm{P}^\dagger$ gates, n_{Pl}—number of Pauli gates (note that the combined number of Pauli-X and Pauli-Y is never more than three for any of the circuits, so n_{Pl} is dominated by Pauli-Z gates), *dist*—trace distance to approximation, t_{approx}—time spent on the unitary approximation using the Solovay-Kitaev algorithm (in seconds), t_{decomp}—time spent on the decomposition of the approximating unitary into circuit, per Algorithm 1 (in seconds). Circuit specifications are available at http://qcirc.iqc.uwaterloo.ca/.

U	N_I	n_g	n_T	n_H	n_P	n_{Pl}	*dist*	t_{approx}	t_{decomp}
$R_z\left(\frac{\pi}{64}\right)$	0	54	22	23	2	7	8.05585×10^{-4}	0.08827	0.00020
	1	344	136	136	3	69	9.57729×10^{-6}	0.12163	0.00080
	2	1414	564	564	3	283	1.97877×10^{-7}	0.38928	0.00326
	3	7769	3086	3087	3	1593	1.08884×10^{-10}	0.98522	0.01954
	4	35456	14170	14171	2	7113	3.00267×10^{-15}	3.05440	0.12384
$R_z\left(\frac{\pi}{128}\right)$	0	72	28	29	1	14	9.59916×10^{-4}	0.08822	0.00023
	1	344	136	137	3	68	1.79353×10^{-5}	0.12143	0.00081
	2	1588	634	634	4	316	3.67734×10^{-7}	0.39048	0.00368
	3	7519	3004	3005	2	1508	4.23657×10^{-10}	0.98045	0.01890
	4	34388	13722	13722	2	6942	1.32046×10^{-14}	2.86740	0.11832
$R_z\left(\frac{\pi}{256}\right)$	0	71	28	29	2	12	5.06207×10^{-4}	0.01819	0.00023
	1	326	136	136	2	52	1.08919×10^{-5}	0.05474	0.00079
	2	1389	566	567	3	253	2.00138×10^{-7}	0.30498	0.00332
	3	7900	3174	3175	3	1548	2.91716×10^{-10}	0.91405	0.02060
	4	38188	15290	15291	1	7606	8.87785×10^{-15}	2.98030	0.13545
$R_z\left(\frac{\pi}{512}\right)$	0	76	30	29	2	15	3.62591×10^{-4}	0.01749	0.00023
	1	319	126	126	2	65	1.95491×10^{-5}	0.05171	0.00075
	2	1722	680	680	2	360	2.76529×10^{-7}	0.30618	0.00396
	3	8122	3242	3242	2	1636	1.87476×10^{-10}	0.92576	0.02109
	4	34974	13992	13992	1	6989	5.66762×10^{-15}	3.16060	0.11920
$R_z\left(\frac{\pi}{1024}\right)$	0	0	0	0	0	0	2.16938×10^{-3}	0.08622	0.00005
	1	264	106	105	2	51	5.57373×10^{-5}	0.13615	0.00063
	2	1541	622	622	3	294	1.74595×10^{-7}	0.23445	0.00366
	3	6791	2722	2722	1	1346	5.39912×10^{-11}	0.82811	0.01703
	4	32983	13188	13188	1	6606	5.54995×10^{-16}	2.98480	0.11494

22 and tile width equal to 0.14. During this experiment, the memory usage was around 6 GB. For the purpose of the comparison gate counts for our implementation are also provided in the {H, T} library (Z=T^4 and P=T^2). We used the projective trace distance to measure the quality of approximation as it is the one used in Dawson's code. Because of the larger ε-net used in our implementation we were able to achieve better approximation quality using fewer iterations of the Solovay-Kitaev algorithm. Usage of The GNU Multiple Precision Arithmetic Library allowed us to achieve precision up to 10^{-50} while Dawson's code encounters convergence problem when precision reaches 10^{-8}. The latter explains the behaviour of the last set of points in the experimental results for Dawson's code reported in Figure 3.1.

Two other experiments that we performed with Dawson's code are a resynthesis of the circuits generated by it using our exact decomposition algorithm. We first resynthesised circuits that were generated by Dawson's code using the {H,T} library. In most cases, the gate counts reduced by about 10-20% (our resulting circuits were further decomposed such as to use the {H, T} gate library). In the other experiment we used {H, T, P, Z} gate library with Dawson's implementation. In this case, we were able to run Dawson's code with the sequences of length 9 only, and it used 6 GB of memory. The gate counts, using our algorithm, decreased by about 40-60%.

Chapter 4

Asymptotically optimal approximations with the Clifford+T gate set

In this chapter we describe an algorithm that approximates the $R_z(\phi)$ operator using two ancillary qubits and $O(\log(1/\varepsilon))$ gates to achieve the quality of approximation ε. The algorithm has a provably probabilistic polynomial runtime and relies on the probabilistic polynomial time algorithm for solving the four squares Diophantine equation.

In the second part of this chapter we show that in the worst case one needs $O(\log(1/\varepsilon))$ gates to approximate a unitary even when using the constant number of ancillary qubits. This result was originally shown in [50] and we provide the proof for completeness.

4.1 Main result

We focus on the approximation of the following operator:

$$\Lambda(e^{i\phi}) : \alpha |0\rangle + \beta |1\rangle \mapsto \alpha |0\rangle + \beta e^{i\phi} |1\rangle .$$

which is equal to $R_z(\phi)$ up to a global phase.

There are two main steps in our algorithm:

1. Find a circuit C consisting of Clifford and T gates such that the result of applying C to $|00\rangle$ is close to $e^{i\phi} |00\rangle$.

2. Apply circuit C controlled on the first qubit to perform a transformation close to:

$$\alpha\,|000\rangle + \beta\,|100\rangle \mapsto \alpha\,|000\rangle + \beta e^{i\phi}\,|100\rangle\,.$$

The net effect of such transformation may be described as the application of $\Lambda(e^{i\phi})$ to the first qubit. To accomplish the first step we approximate $e^{i\phi}\,|00\rangle$ with a four dimensional vector $|v\rangle$ with entries in the ring $\mathbb{Z}\left[i, \frac{1}{\sqrt{2}}\right]$. We then employ an algorithm for multiple qubit exact synthesis to find a circuit C that prepares $|v\rangle$ starting from $|00\rangle$ using at most one ancilla qubit. It was shown in [4] that any circuit using Clifford and T gates can be transformed into its exact (meaning no further approximation is required) controlled version with only a linear overhead in the number of gates, and using at most one ancilla qubit in the state $|0\rangle$ that is returned unchanged. Our analysis shows that, however, on this step we do not need to use this additional ancilla. The resulting total number of ancillae is thus at most two.

4.1.1 Approximating $e^{i\phi}\,|00\rangle$

The key is the reduction of the approximation problem to expressing an integer number as a sum of four squares. In particular, we are looking for an approximation of:

$$e^{i\phi}\,|00\rangle = (\cos(\phi) + i\sin(\phi), 0, 0, 0)$$

by a unit vector:

$$|v\rangle := \frac{1}{2^k}\left(\left\lfloor 2^k\cos(\phi)\right\rfloor + i\left\lfloor 2^k\sin(\phi)\right\rfloor, 0, a+ib, c+id\right),$$

where $k \in \mathbb{N}; a, b, c, d \in \mathbb{Z}$. Without loss of generality we can assume that $0 \leq \phi \leq \frac{\pi}{4}$. The power k of the denominator determines precision of our approximation and complexity of the resulting circuit. As $|v\rangle$ must be a unit vector, the remaining four parameters ($a, b, c,$ and d) should satisfy the integer equation:

$$a^2 + b^2 + c^2 + d^2 = 4^k - \left\lfloor 2^k\cos(\phi)\right\rfloor^2 - \left\lfloor 2^k\sin(\phi)\right\rfloor^2.$$

Lagrange's four square theorem states that this equation always has a solution. Furthermore, there exists an efficient probabilistic algorithm for finding a solution. For the right hand side M it requires on average $O(\log^2(M)\log\log M)$ operations with integers smaller than M. It is described in Theorem 2.2 in [66]. We get a reduction to such a simple Diophantine equation at the expense of using two qubits instead of one.

Furthermore, in estimating the classical complexity of the algorithm for finding the approximating circuit, we will rely on the observation that

$$4^k - \left\lfloor 2^k \cos(\phi) \right\rfloor^2 - \left\lfloor 2^k \sin(\phi) \right\rfloor^2 \leq 4 \times 2^k + O(1) \in O(2^k).$$

4.1.2 Precision and complexity analysis

Let us introduce $\gamma = \left(\left\lfloor 2^k \cos(\phi) \right\rfloor + i \left\lfloor 2^k \sin(\phi) \right\rfloor \right) / 2^k$ and express $|v\rangle$ as:

$$|v\rangle = \gamma |00\rangle + |1\rangle \otimes |g\rangle .$$

The application of the circuit C controlled on the first qubit will transform $(\alpha |0\rangle + \beta |1\rangle) \otimes |00\rangle$ into:

$$\alpha |000\rangle + \beta\gamma |100\rangle + \beta |01\rangle \otimes |g\rangle .$$

The distance of the result to the desired state $\alpha |000\rangle + \beta e^{i\phi} |100\rangle$ is:

$$\sqrt{|\beta (e^{i\phi} - \gamma)|^2 + |\beta|^2 \, \||g\rangle\|^2}.$$

By the choice of γ we have $\left|\gamma - e^{i\phi}\right| \leq \frac{\sqrt{2}}{2^k}$, therefore the first term in the sum above is in $O(1/2^{2k})$. The norm squared of $|g\rangle$ equals $1 - |\gamma|^2$. The complex number γ approximates $e^{i\phi}$, and the distance of its absolute value to identity can be estimated using the triangle inequality:

$$\left| |\gamma| - \left| e^{i\phi} \right| \right| \leq \left| \gamma - e^{i\phi} \right| .$$

Therefore, $1 - |\gamma|^2$ is in $O(1/2^k)$. In summary, the distance to the approximation is in $O(1/2^{0.5k})$.

The same estimate is true if we consider the circuit C as a part of a larger system. In this case we should start with the state $(\alpha |\phi_0\rangle \otimes |0\rangle + \beta |\phi_1\rangle \otimes |1\rangle) \otimes |00\rangle$. Similar analysis shows that the distance to the approximation remains $O(1/2^{0.5k})$.

As shown in [26], it is possible to find a circuit that prepares $|v\rangle$ using $O(k)$ Clifford and T gates ([26], Lemma 20 (Column lemma)). The classical complexity of constructing a quantum circuit implementing $|v\rangle$ is in $O(k)$. In the controlled version of this circuit the number of gates remains $O(k)$ ([4], Theorem 1). In summary, we need $O(\log(1/\varepsilon))$ gates to achieve precision ε. The complexity of the classical algorithm for constructing the entire approximating circuit is thus dominated by complexity of finding a solution to the Diophantine equation, which is in $O(\log^2(1/\varepsilon) \log\log(1/\varepsilon))$.

4.1.3 How many ancillae are needed?

A straightforward calculation shows that the number of ancillae used is three. However, we can get over using only two ancillae. To understand how, we need to go into the details of the proof of Lemma 5 (Column lemma) from [26]. It shows how to find a sequence of two-level unitaries of type $i\mathrm{X}$, $\mathrm{T}^{-m}(i\mathrm{H})\mathrm{T}^{m}$, and W [26] and length $O(k)$ that allows to prepare a state with the denominator 2^k. A controlled version of the two level unitary is again a two level unitary. In [26], Lemma 24, it was also shown that any such unitary required can be implemented using no extra ancillae. Therefore, the controlled version of the circuit C will not use any additional ancilla and we need only two of them in total.

4.2 Lower bound on the number of gates when ancillae are allowed

Lemma 4.2.1. *Let G be a universal gate set, and let M_V be a set of unitaries, that simulate a unitary V acting on n qubits, using m ancillary qubits:*

$$M_V = \left\{ U \in \mathbb{U}\left(2^{m+n}\right) | U\left(|0\rangle \otimes |\phi\rangle\right) = |0\rangle \otimes \left(V|\phi\rangle\right) \right\}.$$

Then, for any ε there always exists a unitary $V(\varepsilon)$ such that the number of gates from G needed to construct a unitary within the distance ε to $M_{V(\varepsilon)}$ is in $\Omega(\log(1/\varepsilon))$.

We use the volume argument similar to the one presented in [32].

Let $N = 2^n$, ρ be the distance induced by Frobenius norm and μ be the Haar measure on $\mathbb{U}(N)$. For the unitary U we define the volume of its ε-neighbourhood as:

$$v(U, \varepsilon) = \mu\left\{ V \in \mathbb{U}(N) | \rho(M_V, U) \leq \varepsilon \right\}.$$

Let G^k be the set of all unitaries that can be constructed using k gates from the library G. Suppose that for any unitary V we can find a unitary U from G^k within the distance ε from M_V. This implies:

$$\mu\left(\mathbb{U}(N)\right) \leq \sum_{U \in G^k} v(U, \varepsilon) \leq |G|^k \max_{U \in G^k} v(U, \varepsilon).$$

We will show that the volume $v(U, \varepsilon)$ is upper bounded by $C_0 \varepsilon^{N^2}$, for some constant C_0, therefore:

$$k \geq \frac{1}{\log|G|} \log\left(\frac{\mu\left(\mathbb{U}(N)\right)}{C_0 \varepsilon^{N^2}} \right). \tag{4.1}$$

We next show how to estimate $v(U,\varepsilon)$. Let U_0 be a submatrix of U defined as follows:

$$U_0 := \{\langle e_i| \otimes \langle 0|) U (|0\rangle \otimes |e_j\rangle\}$$

where $\{|e_i\rangle\}$ is the standard (computational) basis in $\mathbb{C}(N)$. Taking into account that the distance ρ is induced by Frobenius norm, we write $\rho(U, M_V) \geq \rho(U_0, V)$. Therefore:

$$v(U,\varepsilon) = \mu\{V|\rho(M_V, U) < \varepsilon\} \leq \mu\{V|\rho(U_0, V) < \varepsilon\}.$$

Let us define V_{min} to be a unitary closest to U_0. To estimate $v(U,\varepsilon)$ it suffices to consider the case when $\rho(V_{min}, U_0) < \varepsilon$. The distance ρ is unitarily invariant, therefore $\rho\left(V_{min}^{\dagger} U_0, I\right) < \varepsilon$ and

$$\{V|\rho(U_0, V) < \varepsilon\} = \left\{V|\rho\left(V_{min}^{\dagger} U_0, V\right) < \varepsilon\right\}.$$

From the triangle inequality,

$$\rho(I, V) \leq \rho\left(V_{min}^{\dagger} U_0, I\right) + \rho\left(V_{min}^{\dagger} U_0, V\right),$$

we conclude that

$$\left\{V|\rho\left(V_{min}^{\dagger} U_0, V\right) < \varepsilon\right\} \subseteq \{V|\rho(I, V) < 2\varepsilon\}.$$

Finally,

$$v(U,\varepsilon) \leq \mu\{V|\rho(I, V) < 2\varepsilon\}.$$

As shown in [32], there exists a constant C_0 such that the volume of the ball $\{V|\rho(I, V) < 2\varepsilon\}$ is less than $C_0 \varepsilon^{N^2}$.

Estimate (4.1) on k shows that we need circuits of the size at least $\Omega(\log(1/\varepsilon))$ to cover the full group $\mathbb{U}(N)$. If k is chosen in such a way that the inequality (4.1) does not hold, due to the volume argument, there exists a unitary $V(\varepsilon)$ such that it is not possible to approximate any unitary from $M_{V(\varepsilon)}$ with precision ε using at most k gates.

Chapter 5

Practical approximations with the Clifford+T gate set

In this chapter we describe the algorithm for finding optimal approximations of single qubit rotations $R_z(\phi)$ using the Clifford+T get set. The algorithm is based on exact synthesis results from Chapter 3. With the exact synthesis results we reduce the problem of finding the best approximation by a Clifford and T circuit to the problem of finding the best approximation by an exact unitary. We call this problem the Closest Unitaries Problem (CUP). The algorithm runtime and memory usage scales exponentially with the T-count of the unitaries used for approximation. However, the algorithm proposed is much more efficient that naive brute force search [22]. It allows us to achieve the quality of approximation 10^{-15} using the computational resources available today. The naive brute force search can only reach the quality of approximation 10^{-4} [22].

5.1 Closest Unitaries Problem

Now we state the Closest Unitaries Problem more formally and briefly discuss why it is easier to solve this problem than the similar problem involving circuits. Recall that we use the global phase invariant distance to measure the quality of approximation. It is defined on single qubit unitaries as

$$d(U, V) = \sqrt{1 - |tr(UV^\dagger)|/2}.$$

Our aim is to find the best approximation using at most the given number of T gates. To restrict the set of exact unitaries we use for approximation we introduce a T-count. For

a given exact unitary U its T-count $t(U)$ is the minimal number of T gates required to implement U up to a global phase when using the Clifford and T gate library. In other words, U can be written in the form

$$e^{i\alpha}C_1TC_2T\dots C_nTC_{n+1}, \text{where } C_i \text{ is a Clifford unitary}$$

for $n = t(U)$ and cannot be written in the form above when $n < t(U)$.

Problem 5.1.1. CUP$[n, \phi]$ *(Closest Unitaries Problem) is a problem of finding:*

- *the distance* $\varepsilon[n, \phi]$ *between* $R_z(\phi)$ *and the set of exact unitaries with T-count at most* n*,*
- *the subset* $D[n, \phi]$ *of all exact unitaries with T-count at most* n *and within distance* $\varepsilon[n, \phi]$ *from* $R_z(\phi)$*; T-count of all elements of* $D[n, \phi]$ *must be minimal.*

The requirement for elements of $D[n, \phi]$ to have minimal possible T-count is non-trivial. This is because the set of all exact unitaries with T-count at most n and within distance $\varepsilon[n, \phi]$ from $R_z(\phi)$ may contain unitaries with different T-count. We provide an example of this situation in the next subsection.

A naive brute-force solution to CUP$[n, \phi]$ [22] requires one to enumerate all exact unitaries with T-count at most n. Our algorithm allows us to significantly reduce the size of the search space used for solving CUP$[n, \phi]$ when $\varepsilon[n-1, \phi]$ is known. Informally, if one has already solved CUP$[n-1, \phi]$ there is no need to solve CUP$[n, \phi]$ from scratch: one just needs to check if using exact unitaries with T-count at most n instead of exact unitaries with T-count at most $n-1$ allows them to improve the quality of approximation over previously achieved $\varepsilon[n-1, \phi]$. It is much easier to accomplish this when approximating by unitaries than when approximating by circuits. In the next section we describe in more details the problem that we need to solve in addition to CUP$[n-1, \phi]$ to find the solution to CUP$[n, \phi]$. We call this problem the Restricted Closest Unitaries Problem.

5.2 Restricted Closest Unitaries Problem

We introduce the notion of minimal unitaries that is crucial for the definition of Restricted Closest Unitaries Problem (RCUP) and use it to show the relation between CUP and

RCUP. The definition of minimal unitaries is motivated by the fact that the distance between the exact unitary $U[x, y, k]$ and $R_z(\phi)$ can be simplified as

$$d(R_z(\phi), U[x, y, k]) = \sqrt{1 - \left|\mathrm{Re}(xe^{i\phi/2}\zeta_8^{-k/2})\right|}. \tag{5.1}$$

In particular, we see that the distance depends only on x but not on y. We say that unitary $U[x, y, k]$ is *minimal* if its T-count is equal to the minimum of the T-count over all unitaries of the form $U[x, y', k]$ for y' from $\mathbb{Z}[i, \frac{1}{\sqrt{2}}]$. Here is an example of minimal and not minimal unitaries that we found when approximating $R_z(\pi/16)$:

$$\begin{aligned} &U[3{+}5\zeta_8{-}3\zeta_8^2{-}2\zeta_8^4, -2{+}2\zeta_8^2{-}3\zeta_8^3, 0]/8, \\ &U[3{+}5\zeta_8{-}3\zeta_8^2{-}2\zeta_8^4, \quad 3{-}2\zeta_8{+}2\zeta_8^3, 0]/8, \zeta_8 := e^{i\pi/4}. \end{aligned}$$

The first unitary has T-count ten and the second twelve. We state RCUP using the notion of minimal unitaries as following:

Problem 5.2.1. RCUP$[n, \phi, \delta]$ *(Restricted Closest Unitaries Problem) is a problem of finding:*

- *The distance* $\varepsilon[n, \phi, \delta]$ *between* $R_z(\phi)$ *and the set of minimal exact unitaries within distance* δ *from* $R_z(\phi)$ *and with T-count equal to* n *(in the case that there is no such unitary we define* $\varepsilon[n, \phi, \delta] = \delta$*),*
- *The set* $D[n, \phi, \delta]$ *of minimal exact unitaries within distance* $\varepsilon[n, \phi, \delta]$ *from* $R_z(\phi)$ *and T-count equal to* n*.*

The next Lemma shows the relation between CUP and RCUP.

Lemma 5.2.2. CUP$[n, \phi]$ *reduces to* CUP$[n{-}1, \phi]$ *and* RCUP$[n, \phi, \delta]$ *for* $\delta = \varepsilon[n{-}1, \phi]$ *as following:*

- *If* $\varepsilon[n, \phi, \delta] \geq \varepsilon[n{-}1, \phi]$ *then* $\varepsilon[n, \phi] = \varepsilon[n{-}1, \phi]$ *and* $D[n, \phi] = D[n{-}1, \phi]$
- *If* $\varepsilon[n, \phi, \delta] < \varepsilon[n{-}1, \phi]$ *then* $\varepsilon[n, \phi] = \varepsilon[n, \phi, \delta]$ *and* $D[n, \phi] = D[n, \phi, \delta]$.

Proof. There are two alternatives for given n, ϕ: using unitaries with T-count n in addition to unitaries with T-count $n-1$ either allows one to achieve better approximation quality or does not. In the first case the only possibility is $\varepsilon[n, \phi, \delta] < \varepsilon[n-1, \phi]$, in the second case — $\varepsilon[n, \phi, \delta] \geq \varepsilon[n-1, \phi]$. By definition $D[n, \phi]$ contains only minimal unitaries, and condition $\varepsilon[n, \phi] < \varepsilon[n-1, \phi]$ implies that all unitaries in $D[n, \phi]$ must have T-count equal to n. Therefore, we conclude that $D[n, \phi, \delta] = D[n, \phi]$ when $\varepsilon[n, \phi, \delta] < \varepsilon[n-1, \phi]$. It is also not difficult to see that if using unitaries with T-count n does not improve the approximation quality then $D[n, \phi] = D[n-1, \phi]$. □

Usually in practice one is interested in solving the set of problems CUP$[n, \phi]$ for n between 0 and N where N is defined by amount of computing resources available. Lemma 5.2.2 shows that this task is equivalent to solving the set of RCUP$[n, \phi, \delta_n]$ for n in the same range as above and properly chosen δ_n.

5.3 Algorithm

In this section we present an algorithm for solving RCUP$[n, \phi, \delta]$ and prove its correctness. In the previous section we noticed that the distance between $R_z(\phi)$ and exact unitary $U[x, y, k]$ depends only on x, k. Our algorithm searches for approximations of x instead of directly searching for $U[x, y, k]$. This motivates the definition of T-count for the elements of $\mathbb{Z}[i, \frac{1}{\sqrt{2}}]$:

$$t_k(x) = \min \{t(U[x, y, k]) \,|\, U[x, y, k] \text{ — exact unitary}\}$$

If the minimum above is taken over the empty set we define $t_k(x) = \infty$. In other words, this means that there is no unitary over the ring $\mathbb{Z}[i, \frac{1}{\sqrt{2}}]$ such that x is its entry. We discuss the conditions on x that guarantee the existence of $U[x, y, k]$ later in Section 5.5. Function $t_k(x)$ is useful for both the algorithm description and the proof of its correctness. It has several properties:

Proposition 5.3.1. *T-count for elements of $\mathbb{Z}[i, \frac{1}{\sqrt{2}}]$ has the following properties:*

- *T-count of any minimal unitary $U[x, y, k]$ is $t_k(x)$,*
- $t_k(x) = t_{k \bmod 2}(x)$,
- $t_k(x) = t_k(\zeta_8 x)$,
- *if* $4 \leq t_k(x) < \infty$
  then $t_k(x) = \mathrm{sde}(|x|^2) - 2 + (\mathrm{sde}(|x|^2) + k) \bmod 2$.

The function sde (the smallest denominator exponent) is defined on numbers of the form

$$(a + \sqrt{2}b)/\sqrt{2}^m \tag{5.2}$$

where a, b, m are integers. For a given number, the value of sde is equal to the smallest non-negative m such that the number can be written in the form (5.2).

To solve RCUP$[n, \phi, \delta]$ we go through elements x of $\mathbb{Z}[i, \frac{1}{\sqrt{2}}]$ such that $t_k(x) = n$ in an efficient way. There are two sources of efficiency: we split the search problem into two smaller subproblems (the search for real and imaginary parts of x) and take into account the necessary conditions that real and imaginary parts of the solutions must satisfy. The latter further shrinks the search space. In more detail, the properties of $t_k(x)$ imply that we only need to consider k equal to zero and one and limit the set of possible x using the relation between $t_k(x)$ and sde. The next proposition shows constraints on x that can be obtained in this way.

Proposition 5.3.2. *Let x be from $\mathbb{Z}[i, \frac{1}{\sqrt{2}}]$ and $t_k(x) \geq 4$ then x can be written as $(a_0 + \sqrt{2}b_0 + i(a_1 + \sqrt{2}b_1))/\sqrt{2^m}$ for integers a_j, b_j, m the following conditions hold:*

- $m \leq \lfloor (t_k(x) + 5)/2 \rfloor$,
- $a_0^2 + 2b_0^2 \leq 2^m, a_1^2 + 2b_1^2 \leq 2^m$.

Note that we separated conditions on integers a_0, b_0 and a_1, b_1 defining the real and imaginary parts of x. The other set of constraints follows from the inequality

$$\sqrt{1 - \left|\mathrm{Re}(xe^{i\phi/2}\zeta_8^{-k/2})\right|} \leq \delta$$

(see (5.1)) and leads to additional constraints on a_j, b_j that are also separate for the real and imaginary parts of x:

Proposition 5.3.3. *Let $x = (a_0 + \sqrt{2}b_0 + i(a_1 + \sqrt{2}b_1))/\sqrt{2^m}$ for integers a_j, b_j and non-negative integer m and $|x|^2 \leq 1$. If $\sqrt{1 - |\mathrm{Re}(xe^{-i\theta})|} \leq \delta \leq 1$ then the following conditions hold:*

- $|(a_0 + \sqrt{2}b_0) - \cos(\theta)\sqrt{2^m}| \leq \delta\sqrt{2^{m+1}}$,
- $|(a_1 + \sqrt{2}b_1) - \sin(\theta)\sqrt{2^m}| \leq \delta\sqrt{2^{m+1}}$.

In the first part of the algorithm (Fig. 5.1) we build arrays of a_j, b_j satisfying conditions from Propositions 5.3.2,5.3.3 by calling the FIND-HALVES (Fig. 5.2) procedure. In addition, in FIND-HALVES we compute contributions ε_{re} and ε_{im} from the real and imaginary parts of x to $\sqrt{1-\left|\mathrm{Re}(xe^{i\phi/2}\zeta_8^{-k/2})\right|}$. More details on this are provided by the following proposition:

Proposition 5.3.4. *Let $\delta \leq 1/2$ and let $x = (a_0 + \sqrt{2}b_0 + i(a_1 + \sqrt{2}b_1))/\sqrt{2^m}$ for integers a_j, b_j and non-negative integer m. If the following holds:*

- $|(a_0 + \sqrt{2}b_0) - \cos(\theta)\sqrt{2^m}| \leq \delta\sqrt{2^{m+1}}$,
- $|(a_1 + \sqrt{2}b_1) - \sin(\theta)\sqrt{2^m}| \leq \delta\sqrt{2^{m+1}}$,

then

$$
\begin{aligned}
(\sqrt{1-|\mathrm{Re}(xe^{-i\theta})|})^2\sqrt{2^m} &= \varepsilon_{re} + \varepsilon_{im},\\
\varepsilon_{re} &= \cos(\theta)(\sqrt{2^m}\cos(\theta) - (a_0 + b_0\sqrt{2})),\\
\varepsilon_{im} &= \sin(\theta)(\sqrt{2^m}\sin(\theta) - (a_1 + b_1\sqrt{2})).
\end{aligned}
$$

In the next steps of the algorithm we enumerate x from $\mathbb{Z}[i, \frac{1}{\sqrt{2}}]$ satisfying the necessary conditions. We start with those that give the best approximations, in other words the smallest value of $\sqrt{1-\left|\mathrm{Re}(xe^{i\phi/2}\zeta_8^{-k/2})\right|}$. For each candidate x we compute $t_k(x)$ using procedure MIN-T-COUNT. When x with the required T-count is found, procedure ALL-UNITARIES finds all minimal exact unitaries of the form $U[x, y, k]$ and the algorithm terminates. Details on MIN-T-COUNT and ALL-UNITARIES are provided in Section 5.5.

It is important to note that step 8 of the algorithm (Fig. 5.1) is performed efficiently. We first choose tuples corresponding to the real parts such that ε_{re} belongs to the interval $I = [\alpha_0, \alpha_1]$ and then, for each ε_{re}, choose tuples with ε_{im} in the interval $[\alpha_0 - \varepsilon_{re}, \alpha_1 - \varepsilon_{re}]$. The proofs of the propositions presented above are tedious and we postpone them to Section 5.4. Now we use them to prove the correctness of the algorithm.

Theorem 5.3.5. *The RCU-Algorithm (Fig. 5.1) solves* RCUP$[n, \phi, \delta]$ *– the Restricted Closest Unitaries Problem with T-count equal to n, angle ϕ and threshold δ when $n \geq 4$ and $\delta \leq 1/2$.*

Proof. We first formally describe the output of the algorithm $(\varepsilon^*, \partial)$ and then prove that it is indeed the solution to RCUP$[n, \phi, \delta]$. Let us define $\theta_k = \frac{\pi k}{8} - \frac{\phi}{2}$ for $k = 0, 1$ and the following sets:

$$S_k := \left\{ x \;\middle|\; t_k(x) = n,\ \sqrt{1 - |\mathrm{Re}(xe^{-i\theta_k})|} \leq \delta \right\}.$$

We first consider the case when at least one of the S_k is non-empty and show that the algorithm outputs a pair $(\varepsilon^*, \partial)$ such that

$$\begin{aligned} \varepsilon^* &= \min(\varepsilon_1^*, \varepsilon_2^*), \text{ where} \\ \varepsilon_k^* &= \min \left\{ \sqrt{1 - |\mathrm{Re}\,(xe^{-i\theta_k})|} \;\middle|\; x \in S_k \right\}. \end{aligned}$$

Let us also denote by ∂_k the elements of S_k within distance ε^* from $R_z(\phi)$. It is not difficult to see that at least one of the ∂_k is non-empty.

By the definition of S_k, the value ε^* is in the interval $[0, \delta]$. Therefore on some iteration of the *while* loop ε^* will belong to interval I and will be in the list $\varepsilon_1, \ldots, \varepsilon_M$. Indeed, suppose that ∂_{k^*} is non-empty and x is its element. In other words x is such that $t_{k^*}(x) = n$ and $\sqrt{1 - |\mathrm{Re}(xe^{-i\theta_{k^*}})|} = \varepsilon^*$. Proposition 5.3.2 implies that x can be represented as

$$(a_0 + b_0\sqrt{2} + a_1 + b_1 i\sqrt{2})/\sqrt{2^m}, \text{ for } m = \lfloor (n+1)/2 \rfloor + 2.$$

Propositions 5.3.2, 5.3.3 imply that integers a_i, b_i satisfy following the inequalities

$$\begin{aligned} &a_0^2 + 2b_0^2 \leq 2^m, a_1^2 + 2b_1^2 \leq 2^m \\ &|(a_0 + \sqrt{2}b_0) - \cos(\theta_{k^*})\sqrt{2^m}| \leq \sqrt{2^{m+1}}\delta \\ &|(a_1 + \sqrt{2}b_1) - \sin(\theta_{k^*})\sqrt{2^m}| \leq \sqrt{2^{m+1}}\delta. \end{aligned}$$

This implies that after executions of procedure FIND-HALVES, for

$$\begin{aligned} \varepsilon_{re} &= \sqrt{2^{-m}}\cos(\theta)(\sqrt{2^m}\cos(\theta) - (a_0 + b_0\sqrt{2})) \\ \varepsilon_{im} &= \sqrt{2^{-m}}\sin(\theta)(\sqrt{2^m}\sin(\theta) - (a_1 + b_1\sqrt{2})) \end{aligned},$$

the triples $(\varepsilon_{re}, a_0, b_0)$ and $(\varepsilon_{im}, a_1, b_1)$ belong to L_{re,k^*} and L_{im,k^*} correspondingly. Now from Proposition 5.3.4 we conclude that $\varepsilon^* = \varepsilon_{re} + \varepsilon_{im}$ and therefore tuple $(\varepsilon^*, a_0, b_0, a_1, b_1)$ belongs to array A and ε^* is in list $\varepsilon_1, \ldots, \varepsilon_M$. Let m_0 denote the position of ε^* in the list. When the *for* loop reaches $m = m_0$ the algorithm will terminate. It is not difficult to see that the algorithm does not terminate before: this would contradict the minimality of ε^*. The only way for the the algorithm to terminate earlier is if there is an x such that $\sqrt{1 - |\mathrm{Re}\,(x_k e^{-i\theta_k})|} < \varepsilon^*$ and $t_k(x) = n$.

The procedure ALL-UNITARIES is designed to output the set

$$\partial = \bigcup_{\substack{k=0,1 \\ g=0,7}} \left\{ \zeta_8^g U[x,y,k] \in \mathcal{U}^{\min} \;\middle|\; x \in \partial_k, y \in \mathbb{Z}[i, \frac{1}{\sqrt{2}}] \right\}$$

where by $\mathcal{U}^{\min}$ we denote the set of all exact minimal unitaries.

Let us now show that $(\varepsilon^*, \partial)$ is the solution to RCUP$[n, \phi, \delta]$. Suppose that the set $D[n, \phi, \delta]$ is not empty and let $U[x, y, k]$ be some element of the set such that the distance to it from $R_z(\phi)$ is minimal. The distance can be expressed as:

$$d(R_z(\phi), U[x,y,k]) = \sqrt{1 - \left|\mathrm{Re}(xe^{i\phi/2}\zeta_8^{-k/2})\right|} < \delta.$$

Proposition 5.3.1 implies that $t_k(x) = n$ because $U[x, y, k]$ is a minimal unitary with T-count n. Now we show that we can make k equal to zero or one. Indeed, let us write $k = k_0 + 2s$ where k_0 is either zero or one, then

$$\left|\mathrm{Re}(xe^{i\phi/2}\zeta_8^{-k/2})\right| = \left|\mathrm{Re}(x\zeta_8^s e^{i\phi/2}\zeta_8^{-k_0/2})\right|.$$

Again using Proposition 5.3.1 we see that $t_k(x\zeta_8^s) = n$. In addition, we have

$$d(R_z(\phi), U[x,y,k]) = \sqrt{1 - \left|\mathrm{Re}((x\zeta_8^s)e^{-i\theta_{k_0}})\right|}.$$

This implies that $x\zeta_8^s$ is in S_{k_0} and ε^* is less then or equal to $\varepsilon[n, \phi, \delta]$. Now we show that $\varepsilon^* \geq \varepsilon[n, \phi, \delta]$. Suppose that ∂_{k_0} is non-empty for some k_0 and let x be one of its elements. Then, we have $t_{k_0}(x) = n$ and there exists a y from $\mathbb{Z}[i, \frac{1}{\sqrt{2}}]$ such that unitary $U[x, y, k_0]$ is minimal with T-count n. We also notice that $d(R_z(\phi), U[x, y, k_0]) = \varepsilon^*$ which concludes the proof of equality $\varepsilon^* = \varepsilon[n, \phi, \delta]$.

Next we show that ∂ coincides with the set $D[n, \phi, \delta]$. Consider some element $\zeta_8^g U[x, y, k]$ of ∂. The set $D[n, \phi, \delta]$ contains any unitary U together with all unitaries $\zeta_8^g U$, therefore it is enough to show that $U[x, y, k]$ is in $D[n, \phi, \delta]$. The fact that $U[x, y, k]$ is a minimal unitary and $t_k(x) = n$ implies that $U[x, y, k]$ has T-count n. On the other hand, by definition of ∂_k we have that $d(R_z(\phi), U[x, y, k]) = \varepsilon^*$ which shows that ∂ is a subset of $D[n, \phi, \delta]$. Let us now show that $D[n, \phi, \delta]$ is a subset of ∂. Let $U[x, y, k]$ be an element of $D[n, \phi, \delta]$. We first note that $U[x, y, k]$ can be equivalently written as $\zeta_8^s U[x\zeta_8^{-s}, y\zeta_8^{-s}, k{-}2s]$. We chose s in such a way that $k_0 = k{-}2s$ is either zero or one. We note that $U[x\zeta_8^{-s}, y\zeta_8^{-s}, k_0]$ is a minimal unitary and therefore $t_{k_0}(x\zeta_8^{-s}) = n$. Distance d is global phase invariant, which

implies $d(R_z(\phi), U[x\zeta_8^{-s}, y\zeta_8^{-s}, k_0]) = \varepsilon^*$ and $\sqrt{1 - \left|\mathrm{Re}\left(x\zeta_8^{-s}e^{-i\theta_k}\right)\right|} = \varepsilon^*$. We conclude that $x\zeta_8^{-s}$ is in ∂_{k_0} and $U[x\zeta_8^{-s}, y\zeta_8^{-s}, k-2s]$ is in ∂. It is not difficult to see from the definition of ∂ that if U is in ∂ then for any integer g unitary $\zeta_8^g U$ is also in ∂. This concludes the proof of the equality $D[n, \phi, \delta] = \partial$.

It remains to handle the special case when the problem has no solutions. Suppose that $D[n, \phi, \delta]$ is an empty set. It is not difficult to show that this implies that both S_k are empty and vice-versa using the ideas from the main part of the proof. □

The restrictions on n and δ in the theorem statement are not significant. It is much easier to solve CUP$[n, \phi]$ directly when $n < 4$. From our numeric experiments we found that $D[3, \phi]$ is always less than 0.1376 therefore each time RCUP$[n, \phi, \delta]$ is used it is used with parameter $\delta < 1/2$.

It is possible to make the FIND-HALVES procedure (Fig. 5.2) slightly more efficient. We found that the length of the interval $[a_{\min}, a_{\max}]$ in it is usually less than $1/2$ therefore the internal *for* loop can be replaced with the function *round*. It is also not difficult to see that the *while* loop of the procedure can be easily parallelized. In our implementation of the algorithm we benefit from both of these observations.

5.4 Technical details

In this section we prove Propositions 5.3.1-5.3.4. First we need to recall some useful results and definitions from Chapter 3. It is possible to extend sde on exact unitaries as $\mathrm{sde}(U[x, y, k]) = \mathrm{sde}(|x|^2)$. The following result relates it to a unitary T-count:

Lemma 5.4.1 (Corollary of Theorem 3.0.5)**.** *Let U be a unitary over $\mathbb{Z}[i, \frac{1}{\sqrt{2}}]$ such that $\mathrm{sde}(U) \geq 4$ and j, l be integers such that $\mathrm{sde}(HT^jUT^lH) = \mathrm{sde}(U) + 1$, then*

$$t(U) = \mathrm{sde}(U) - (j \bmod 2) - (l \bmod 2).$$

Now we provide proofs of all the propositions.

Proof of Proposition 5.3.1. The first property follows directly from the definition of $t_k(x)$ and minimal unitaries. To prove the second one we notice that multiplication by the Phase gate $P := diag\{1, i\}$ does not change the T-count of a unitary and $U[x, y, k]P = U[x, y, k+2]$; the definition of $t_k(x)$ implies $t_k(x) = t_{k+2}(x)$. To prove the third one we

notice that $\zeta_8 U[x,y,k] = U[\zeta_8 x, \zeta_8 y, k+2]$. To prove the fourth property let us consider minimal unitary $U[x,y,k]$. Its T-count is at least 4 and therefore it requires at least three Hadamard gates to be implemented. This implies that $\mathrm{sde}(U[x,y,k])$ is greater than four. Lemma 5.4.1 applies to $U[x,y,k]$ and implies that there are three possible values of $t_k(x)$

$$\mathrm{sde}(U[x,y,k]) - 2, \mathrm{sde}(U[x,y,k]) - 1, \mathrm{sde}(U[x,y,k]).$$

A minimal unitary cannot have T-count equal to $\mathrm{sde}(U[x,y,k])$. Indeed, if this were true the unitary $TU[x,y,k]T^\dagger$ equal to $U[x,y\zeta_8,k]$ would have T-count $\mathrm{sde}(U[x,y,k])-2$ which contradicts the minimality of $U[x,y,k]$. Now we show that the T-count of $U[x,y,k]$ is completely defined by the parity of sde and k. The determinant of $U[x,y,k]$ is equal to ζ_8^k. In addition, the T-count of $U[x,y,k]$ must have the same parity as k because the T gate is the only gate in a Clifford+T library with determinant equal to an odd power of ζ_8. For example, if sde is odd and k is even the T-count can only be equal to $\mathrm{sde}(U[x,y,k]) - 1$. By considering other possible parities we get the required expression for $t_k(x)$. □

Proof of Proposition 5.3.2. Any x from $\mathbb{Z}[i, \frac{1}{\sqrt{2}}]$ can be written as $(a_0 + \sqrt{2}b_0 + i(a_1 + \sqrt{2}b_1))/\sqrt{2^m}$. Let us choose m to be minimal. Integer m must be positive, otherwise x either cannot be an entry of the unitary or its T-count must be zero. Note that at least one of the a_j in the expression must be odd, otherwise m is not minimal. It is useful to expand $|x|^2$ as

$$\frac{(a_0^2 + a_1^2) + 2(b_0^2 + b_1^2) + 2\sqrt{2}(a_0 b_0 + a_1 b_1)}{2^m}.$$

If one of the a_j is odd and the other one is even we get $\mathrm{sde}(|x|^2) = 2m$. Let us now consider the case when both a_j are odd. In the case that b_1, b_2 have different parity we get

$$(a_0^2 + a_1^2) + 2(b_0^2 + b_1^2) = 0 \bmod 4, a_0 b_0 + a_1 b_1 = 1 \bmod 2$$

and conclude that $\mathrm{sde}(|x|^2) = 2m - 3$. In the other case, when b_1, b_2 have the same parity, we get $(a_0^2 + a_1^2) + 2(b_0^2 + b_1^2) = 2 \bmod 4$ and $\mathrm{sde}(|x|^2) = 2m - 2$. In the worst case $2m - 3 \leq t_k(x) + 2$ which gives us a bound on m.

To prove the second part of proposition we consider minimal unitary a $U[x,y,k]$ and note that $|x|^2$ and $|y|^2$ can be expressed as

$$(x_0 + \sqrt{2}x_1)/2^m, (y_0 + \sqrt{2}y_1)/2^m.$$

The equality $|x|^2 + |y|^2 = 1$ implies $x_0 + y_0 = 2^m$. Using the non-negativity of y_0 we get

$$x_0 = (a_0^2 + a_1^2) + 2(b_0^2 + b_1^2) \leq 2^m$$

which leads to bounds on a_j, b_j. □

Proof of Proposition 5.3.3. First we show that $|x - e^{i\theta}| \leq \sqrt{2}\delta$. We expand $|x - e^{i\theta}|^2$ and note that $|x|^2 \leq 1$ implies $|x - e^{i\theta}|^2 \leq 2 - 2\mathrm{Re}(xe^{-i\theta})$. Then we get the bound $2 - 2\mathrm{Re}(xe^{-i\theta})) \leq 2\delta^2$ from $\sqrt{1 - |\mathrm{Re}(xe^{-i\theta})|} \leq \delta \leq 1$. Second, as for any complex number z, absolute values of its real and imaginary parts are less than $|z|$, from $|x - e^{i\theta}| \leq \sqrt{2}\delta$ we conclude:

$$\begin{aligned} |(a_0 + \sqrt{2}b_0) - \cos(\theta)\sqrt{2^m}| &\leq \sqrt{2^{m+1}}\delta \\ |(a_1 + \sqrt{2}b_1) - \sin(\theta)\sqrt{2^m}| &\leq \sqrt{2^{m+1}}\delta \end{aligned} .$$

□

Proof of Proposition 5.3.4. First we show that $\mathrm{Re}(xe^{-i\theta}) > 0$. Inequalities for the real and imaginary parts of $\sqrt{2^m}(x - e^{i\theta})$ imply $|x - e^{i\theta}| \leq 2\delta$ and $|x| \geq 1 - 2\delta$. Using that $|x - e^{i\theta}|^2 = 1 + |x|^2 - 2\mathrm{Re}(xe^{-i\theta})$ we conclude that $\mathrm{Re}(xe^{-i\theta}) \geq 1 - 2\delta$ which is always non-negative when $\delta \leq 1/2$.

Second, we use $\mathrm{Re}(xe^{-i\theta}) > 0$ to write

$$\begin{aligned} &(\sqrt{1 - |\mathrm{Re}(xe^{-i\theta})|})^2\sqrt{2^m} = \\ &\sqrt{2^m}(1 - \cos(\theta)\mathrm{Re}(x) - \sin(\theta)\mathrm{Im}(x)). \end{aligned}$$

By replacing 1 with $\cos(\theta)^2 + \sin(\theta)^2$ we come to the required conclusion. □

5.5 Norm equations

In this section we discuss mathematical tools required to compute $t_k(x)$ (procedure MINT-COUNT) and to enumerate all minimal exact unitaries with top-left entry x — unitaries of the form $U[x, y, k]$ (procedure ALL-UNITARIES). Alternatively these two problems can be reformulated using equation

$$|y|^2 = 1 - |x|^2 \tag{5.3}$$

which expresses that $U[x, y, k]$ must be a unitary matrix. Proposition 5.3.1 implies that it is easy to find $t_k(x)$ when the equation above has a solution for some y from $\mathbb{Z}[i, \frac{1}{\sqrt{2}}]$. The problem of enumerating all unitaries is directly related to enumerating all the solutions of the equation above.

We reduce equation (5.3) to a relative norm equation between two rings of algebraic integers:

$$\mathbb{Z}\left[\zeta_8\right] := \left\{ a_0 + a_1\zeta_8 + a_2\zeta_8^2 + a_3\zeta_8^3 \;\middle|\; a_i \in \mathbb{Z} \right\}, \zeta_8 := e^{i\pi/4}$$

and its real subring $\mathbb{Z}\left[\sqrt{2}\right] := \left\{a + b\sqrt{2} \,\middle|\, a, b \in \mathbb{Z}\right\}$. Indeed x and y can be expressed as $x'/\sqrt{2}^m$ and $y'/\sqrt{2}^m$ where x' and y' are from $\mathbb{Z}[\zeta_8]$. Equation (5.3) can be rewritten as

$$|y'|^2 = 2^m - |x'|^2 = A + B\sqrt{2} \in \mathbb{Z}\left[\sqrt{2}\right] \tag{5.4}$$

which is a special case of a relative norm equation well studied in the literature[14, 13]. Conditions when this equation is solvable and methods for enumerating its solutions are well known. Algorithms required for this are implemented in the package PARI/GP [16]. Next we give a brief simplified overview of results related to equation (5.4).

Symmetries of equations frequently provide useful insights about their solvability. As it was observed in [68], the automorphism of $\mathbb{Z}[\zeta_8]$ and its real subring $\mathbb{Z}\left[\sqrt{2}\right]$ defined as

$$(\zeta_8)^\bullet = -\zeta_8, (\sqrt{2})^\bullet = -\sqrt{2}, (a)^\bullet = a, \text{ for } a \in \mathbb{Z}$$

is useful when studying equation (5.4). It is not difficult to check that mapping $(\cdot)^\bullet$ preserves addition, multiplication and commutes with complex conjugation and $|\cdot|^2$. Therefore, if y' is a solution to equation (5.4) then $(y')^\bullet$ must be a solution to equation $(A + B\sqrt{2})^\bullet = A - B\sqrt{2}$. This implies that the necessary condition for equation (5.4) to be solvable is

$$A + B\sqrt{2} \geq 0, A - B\sqrt{2} \geq 0. \tag{5.5}$$

However, as we will see below, this condition is not sufficient.

The problem of solving the relative norm equation can be reduced to a set of subproblems. Suppose that $A + B\sqrt{2}$ can be written as a product $(A_1 + B_1\sqrt{2})(A_2 + B_2\sqrt{2})$ and y_j are solutions to the relative norm equation with right-hand side $A_j + B_j\sqrt{2}$. In this case $\zeta_8^k y_1 y_2$, $\zeta_8^k y_1^* y_2$, $\zeta_8^k y_1 y_2^*$, $\zeta_8^k y_1^* y_2^*$ are solutions to equation (5.4). More generally, any element $A + B\sqrt{2}$ of $\mathbb{Z}\left[\sqrt{2}\right]$ can be represented as:

$$u\sqrt{2}^{k(0)} \xi_1^{k(1)} \dots \xi_N^{k(N)} p_1^{l(1)} \dots p_N^{l(M)}$$

where u is a unit in $\mathbb{Z}\left[\sqrt{2}\right]$ (such element of $\mathbb{Z}\left[\sqrt{2}\right]$ that $u(u)^\bullet = 1$) and ξ_j are such that $\xi_j(\xi_j)^\bullet$ are prime numbers of the form $8n \pm 1$ and p_j are prime numbers of the form $8n \pm 3$. This is related to the general theory of quadratic extensions and classification of primes into ramified, split and inert (see Chapter 3.4 of [15]). When the necessary condition (5.5) holds the right hand side can be represented as

$$(\sqrt{2} - 1)^{l(0)} \sqrt{2}^{k(0)} \xi_1^{k(1)} \dots \xi_N^{k(N)} p_1^{l(1)} \dots p_N^{l(M)} \tag{5.6}$$

where $\xi_j^{k(j)}$ and $(\xi_j^{k(j)})^\bullet$ are positive. Here $\sqrt{2}-1$ is a fundamental unit of $\mathbb{Z}\left[\sqrt{2}\right]$ (for results on unit groups of quadratic extensions see Chapter 3.4.2 of [15]). The condition (5.5) implies that the sum $k(0)+l(0)$ must be even. The equation with right-hand side $A+B\sqrt{2}$ is solvable if and only if each of the equations $|y'|^2=\xi_j^{k(j)}$, $|y'|^2=p_j^{l(j)}$ is solvable. Next we discuss the solvability of the equations above in more details.

When $k(j)$ and $l(j)$ are even the equations obviously have a solution. Equation $|y'|^2=\xi_j$ has a solution if and only if $\xi(\xi)^\bullet$ is a prime of the form $8n+1$. A probabilistic polynomial time algorithm for finding a solution to it can be found in [68]. In this case, in the theory of cyclotomic number fields, it is said that prime $\xi(\xi)^\bullet$ splits completely [70]. The solvability of equation $|y'|^2=p_j$ is related to two subrings of $\mathbb{Z}[\zeta_8]$: $\mathbb{Z}[i\sqrt{2}]$ and $\mathbb{Z}[i]$. When p_j has form $8n-3$ it splits in $\mathbb{Z}[i]$. In other words, there is a solution to the equation of the form $y'=a\pm bi$ for a,b integers. When p_j has form $8n+3$ it splits in $\mathbb{Z}[i\sqrt{2}]$ — there is a solution to the equation of the form $y'=a\pm ib\sqrt{2}$. Both cases can be solved in probabilistic polynomial time (see Chapter 4.8 [13]).

When decomposition (5.6) is known the solution to the equation (5.4) can be found efficiently. The problem of finding the decomposition can be reduced to factoring integers. We note that mapping $\mathcal{N}(\xi):=\xi(\xi)^\bullet$ is multiplicative and applying it to (5.6) gives us integer

$$(-1)^{l(0)+k(0)}2^{k(0)}\mathcal{N}(\xi_1)^{k(1)}\ldots\mathcal{N}(\xi_1)^{k(N)}p_1^{2l(1)}\ldots p_M^{2l(M)}.$$

Therefore by using integer factoring we can find $\mathcal{N}(\xi_j)$,$k(j)$,p_j,$l(j)$. Recall that $\mathcal{N}(\xi_j)$ are prime numbers of the form $8n\pm1$ and p_j are prime numbers of the form $8n\pm3$. Numbers ξ_j can be found in probabilistic polynomial time using the algorithm from Chapter 4.8 [13] which is also a part of PARI/GP [16]. Therefore to implement the predicate IS-SOLVABLE in the MIN-T-COUNT procedure (Fig. 5.3) it is sufficient to first check the necessary conditions (5.5). If they hold, compute the norm $\mathcal{N}$ of the equation right-side, and find $\mathcal{N}(\xi_j)$,$k(j)$,p_j,$l(j)$. Return TRUE if there is no $\mathcal{N}(\xi_j)$ of the form $8n-1$ such that $k(j)$ is odd, and FALSE otherwise.

To demonstrate how to enumerate all the solutions to (5.4) we provide an example. Consider the equation:

$$|y'|^2=1\,828\,037\,034-1\,292\,617\,383\sqrt{2}.$$

The norm $\mathcal{N}$ of the right-hand side is $2\cdot193\cdot2297\cdot3^2$. We find that equation can be rewritten as

$$|y'|^2=(\sqrt{2}-1)^{15}\cdot\sqrt{2}\cdot(15-4\sqrt{2})\cdot(53-16\sqrt{2})\cdot3.$$

The general form of the solution is

$$y'=(\sqrt{2}-1)^7\zeta_8^k y_0\cdot Y_1\cdot Y_2\cdot Y_3,\ Y_j\in\{y_j,y_j^*\},$$

where $y_0 = 1 - \zeta_8$, $y_1 = -1 - 3\zeta_8 + \zeta_8^2 - 2\zeta_8^3$, $y_2 = 3 - 6\zeta_8 - 2\zeta_8^2 + 2\zeta_8^3$, $y_3 = 1 \pm i\sqrt{2}$. We do not consider complex conjugation of y_0 because $(y_0)^* = \zeta_8^3 y_0$. This is related to the fact that two is the only ramified prime in $\mathbb{Z}[\zeta_8]$ [70]. Taking into account that k can take values from zero to seven we find that there are 64 different solutions to the example equation. To implement ALL-UNITARIES procedure we need to factorize the right-hand side of the relative norm equation (5.4), find all possible solutions of it as in the example above, write down all the unitaries and pick those which are minimal.

5.6 Experimental results

In this section we report empiric estimates of our algorithm performance and quality of approximations produced by the algorithm. We study memory, processing time required by our algorithm and precision achieved by it. For our experiments we used a high performance server with eight Quad-Core AMD Opteron 8356 (2.30 GHz) processors and 128 GB of RAM memory. Our current algorithm implementation completely utilizes the processing power of the server and runs 32 threads in parallel. The binary and the source code are available at `https://code.google.com/p/sqct/`. We also discuss experimental verification of our algorithm's implementation.

To get the estimates for the time and memory required to run our algorithm we found T-optimal approximations of R_z rotations by angles of the form $2\pi k/1000$ for $k = 1, \ldots, 1000$. We used circuits with up to 109 T gates for the approximation. The only known algorithm that gives the same optimality guarantees as our algorithm is a naive brute force search [22]. Let N_{BFS} and t_{BFS} be the number of records and user time needed for brute force search. Since the number of unitaries with T-count at most n scales as $192 \cdot (3 \cdot 2^n - 2)$ [55] both $\log_2(N_{\mathrm{BFS}})$ and $\log_2(t_{\mathrm{BFS}})$ equal to n up to an additive constant. The most memory consuming part of our algorithm is the FIND-HALVES procedure. The base two logarithm of the number of records it produces scales as $0.17n + 5.07$ on average and as $0.25n + 3.15$ in the worst case (Fig. 5.4). On Fig. 5.5 we see that FIND-HALVES is also the most time consuming part of the algorithm and, on average, logarithm base two of the time in milliseconds required to run the procedure scales as $0.21n - 10.41$. Though our algorithm requires an exponential amount of time and memory, the constants in the exponent are between four and five times better than for naive brute force search. This allows us to find T-optimal approximations with precisions up to 10^{-15} using modern computers. This is sufficient for most applications, as we discussed before.

On average, the number of T gates needed to achieve a given quality of approximation ε scales as $3.067 \log(1/\varepsilon) - 4.322$ (Fig. 5.6). The knowledge of this scaling is important for

estimating the resources required to run quantum algorithms having R_z rotations as their building block. The best previous estimate was based on values of $\log_2(1/\varepsilon)$ less then 14 due to the inefficiency of the naive brute force approach [22].

We computed optimal circuits for R_z rotations by angles $\pi/2^k$ for $k = 3, \ldots, 27$. This set of rotations is used in the Quantum Fourier Transform which is a common building block for many quantum algorithms. We found approximations reaching precision up to 10^{-15} (Fig. 5.8). To make a direct comparison to [68] we also found optimal approximations of $R_z(0.1)$ using up to 153 T gates and reaching precision $3.18 \cdot 10^{-16}$ (Fig. 5.7). Computing all the approximations of $R_z(0.1)$ took us 33.2 hours in total (user time). Our circuits are around 25% shorter than those obtained using the algorithm from [68]. All optimal circuits found by our algorithm are available online at `https://code.google.com/p/sqct/`.

To verify the implementation of our algorithm, we implemented a naive brute-force search algorithm that also solves Closest Unitaries Problem. We ran both algorithms to find all optimal approximations of rotations $R_z(2\pi k/1000)$ for $k \in [0, 1000)$ with at most 18 T gates. For this set of inputs algorithms produce identical results. The verification procedure is a part of SQCT 0.2 and can be executed via the command line option "-B".

Input: n, ϕ, δ ▷ n — T-count, $R_z(\phi)$ — target rotation

$m \leftarrow \lfloor (n+1)/2 \rfloor + 2$
for $k = 0, 1$ **do**
  $L_{re,k} \leftarrow$ FIND-HALVES$(\cos(\phi - \pi k/8), m, \delta)$
  $L_{im,k} \leftarrow$ FIND-HALVES$(\sin(\phi - \pi k/8), m, \delta)$
    ▷ (described on Fig. 5.2)
end for
Interval $I \leftarrow [0, \alpha]$ ▷ Pick $\alpha > 0$ based on $L_{re,k}, L_{im,k}$
while $I \cap [0, \delta] \neq \varnothing$ **do**
  Find an array A of tuples $(\varepsilon, a_0, b_0, a_1, b_1, k)$ s.t.:
    • $(\varepsilon_{re}, a_0, b_0)$ from $L_{re,k}$
    • $(\varepsilon_{im}, a_1, b_1)$ from $L_{im,k}$
    • $\varepsilon = \varepsilon_{re} + \varepsilon_{im}$ and $\varepsilon \in I \cap [0, \delta]$
  Sort A by ε in ascending order
  $\varepsilon_1 < \ldots < \varepsilon_M \leftarrow$ all distinct ε that occur in A
  for $j = 1$ to M **do**
    $\partial \leftarrow \varnothing$
    for all $(\varepsilon_j, a_0, b_0, a_1, b_1) \in A$ **do**
      $x' \leftarrow a_0 + b_0\sqrt{2} + i(a_1 + b_1\sqrt{2})$
      $n_0 \leftarrow$ MIN-T-COUNT(x', m, k)
        ▷ (computes $t_k(x'/\sqrt{2^m})$, see Sec. 5.5)
      if $n = n_0$ **then**
        $\partial \leftarrow \partial \cup$ ALL-UNITARIES(x', m, k)
          ▷ (enumerates minimal unitaries
          ▷ $U[x'/\sqrt{2^m}, y, k]$, see Sec. 5.5)
      end if
    end for
    if $\partial \neq \varnothing$ **then**
      return $(\varepsilon_j, \partial)$ ▷ Solution
    end if
  end for
  Replace $I = [\alpha_0, \alpha_1]$ by $I = [\alpha_1, 2\alpha_1 - \alpha_0]$
end while
return $(\delta, \varnothing)$ ▷ No solutions

Output: $(\varepsilon_n^R, \partial_{n,\phi}^{\delta})$

Figure 5.1: RCU-Algorithm: the algorithm for RCUP$[n, \phi, \delta]$.

Input: $\alpha, \delta \in \mathbb{R}, m \in \mathbb{Z}, m \geq 0$

1: **procedure** FIND-HALVES(α, m, δ)
2: $W \leftarrow \alpha\sqrt{2^{-m}}, \varepsilon \leftarrow \delta\sqrt{2^m}$
3: $b \leftarrow \lfloor -\sqrt{2^m} \rfloor$
4: $v \leftarrow \alpha\sqrt{2^m} - b\sqrt{2}$ ▷ true on every step
5: $R \leftarrow \varnothing$
6: **while** $b \leq \lceil \sqrt{2^m} \rceil$ **do**
7: $a_{\min} = \lceil v - \varepsilon \rceil, a_{\max} = \lfloor v + \varepsilon \rfloor$
8: **for all** $a \in [a_{\min}, a_{\max}] \cap \mathbb{Z}$ **do** ▷ See Sec. [*]
9: **if** $a^2 + 2b^2 \leq 2^m$ **then** ▷ for discussion
10: $R \leftarrow R \cup \{((x - a)W, a, b)\}$
11: **end if**
12: **end for**
13: $b \leftarrow b + 1, x \leftarrow x - \sqrt{2}$
14: **end while**
15: Sort R by first element in ascending order
16: **return** R
17: **end procedure**

Figure 5.2: FIND-HALVES procedure. Finds all numbers of the from $a + \sqrt{b}$ that satisfy conditions $|\alpha\sqrt{2^m} - (a + b\sqrt{2})| \leq \delta\sqrt{2^m}$, $a^2 + 2b^2 <= 2^m$. Returns a list of tuples $(\alpha\sqrt{2^{-m}}(\alpha\sqrt{2^m} - (a + b\sqrt{2})), a, b)$ sorted by first entry.

Input: $x' = a_0 + b_0\sqrt{2} + i(a_1 + b_1\sqrt{2}), m, k \in \mathbb{Z}, m \geq 0$

1: **procedure** MIN-T-COUNT(x', m, k)
2: $a + b\sqrt{2} \leftarrow 2^m - |x'|^2, s \leftarrow \mathrm{sde}(|x'|^2/2^m)$
3: **if** $s \leq 4$ **then**
4: **return** ∞
5: **end if**
6: **if** IS-SOLVABLE($a+\sqrt{2}b$) **then**
7: **return** $s - 2 + (k + s) \bmod 2$
8: **else**
9: **return** ∞
10: **end if**
11: **end procedure**

Output: n

Figure 5.3: MIN-T-COUNT Procedure. Outputs $t_k(x'/\sqrt{2^m})$ if it is greater or equal to 4 and ∞ otherwise.

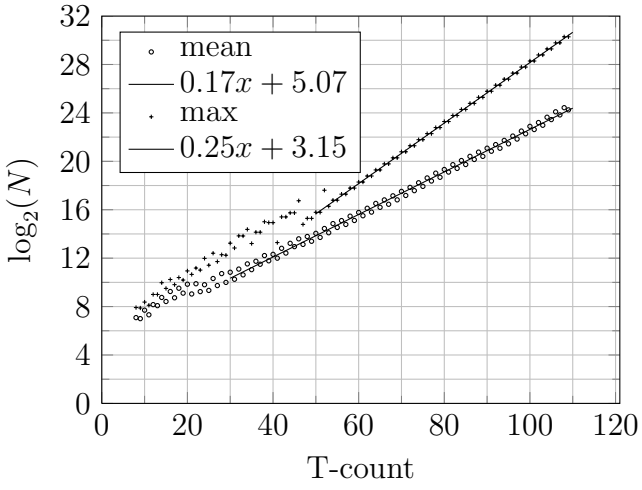

Figure 5.4: Scaling of the number of records found by FIND-HALVES procedure when executing $RCUP$ algorithm for a given T-count. Graph shows average and maximum number of records per given T-count. The data is aggregated over T-optimal approximations of R_z rotations by angles $2\pi k/1000$ for $k = 1, \ldots, 1000$.

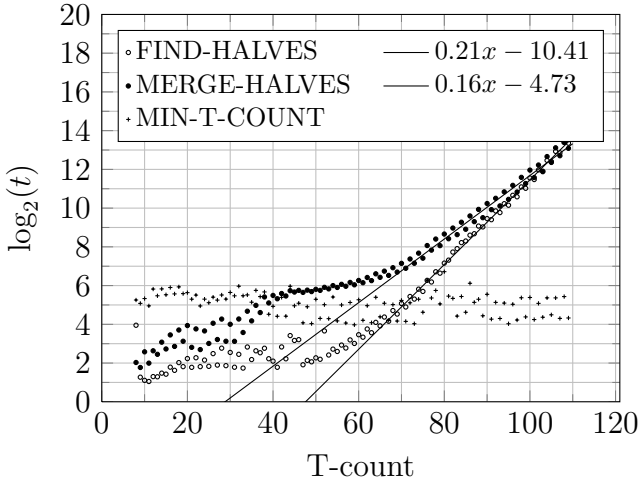

Figure 5.5: Average user time t (milliseconds) required to run different parts of $RCUP$ algorithm for a given T-count. Each point on the graph was obtained by averaging over T-optimal approximations of R_z rotations by angles $2\pi k/1000$ for $k = 1, \ldots, 1000$.

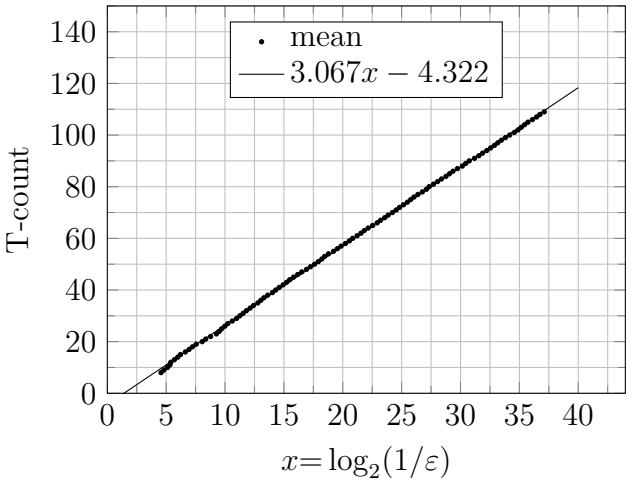

Figure 5.6: Scaling of a circuit T-count required to achieve given approximation precision ε. Each point on the graph shows average precision that can be achieved when using circuits with given T-count. The average is taken over T-optimal approximations of R_z rotations by angles $2\pi k/1000$ for $k = 1, \ldots, 1000$.

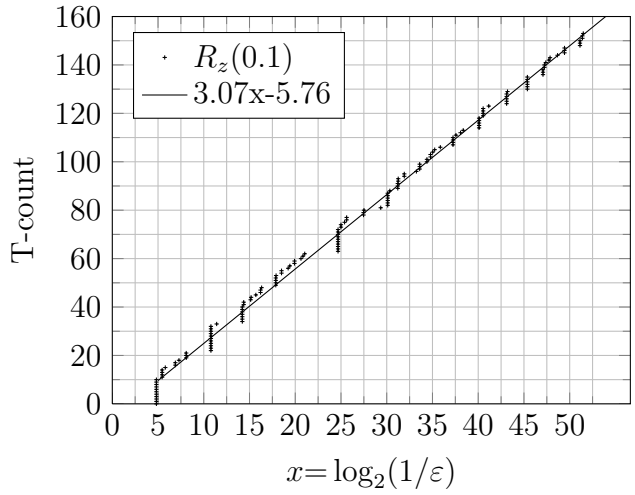

Figure 5.7: Scaling of a circuit T-count required to achieve given precision ε when approximating $R_z(0.1)$.

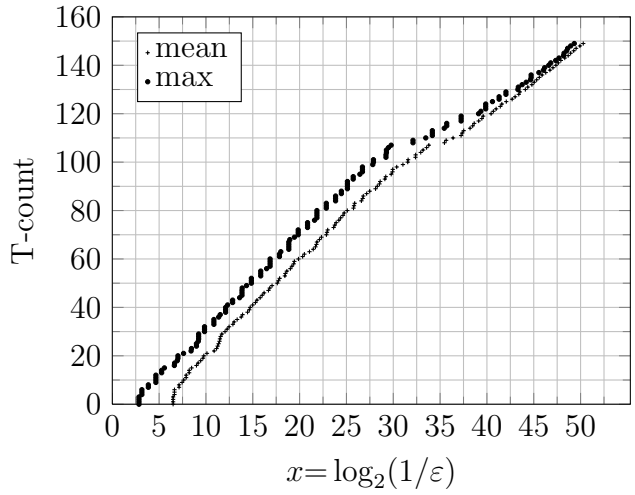

Figure 5.8: Scaling of a circuit T-count required to achieve given approximation precision ε. Graph shows average precision that can be achieved when using circuits with given T-count and maximal T-count required to achieve given precision. The results are aggregated over T-optimal approximations of R_z rotations by angles $\pi/2^k$ for $k = 3, \ldots, 27$.

Chapter 6

Single qubit approximation with the Fibonacci gate set

In this chapter we develop all the technical tools required to solve the approximate synthesis problem for the Fibonacci gate set using a number theoretic method. First we prove Theorem 2.4.2 and Theorem 2.4.4 related to the exact synthesis of single qubit unitaries over the Fibonacci gate set. Next we develop a probabilistic algorithm for approximating single qubit unitaries by exact unitaries. As a result we show how to approximate any single qubit unitary with precision ε using a circuit of length $O(\log(1/\varepsilon))$. On average the circuit is found in time polynomial in $\log(1/\varepsilon)$ (subject to Conjecture 6.3.4). At the end of the chapter we provide the numerical evidence supporting Conjecture 6.3.4 and evaluate the implementation of the approximation algorithm. We show that the algorithm finds circuits that are only around 20% longer than optimal ones found using the brute force search. We also estimate that the algorithm produces shorter circuits than the Solovay-Kitaev algorithm. The circuits we find are 10 times shorter for precision 10^{-10} and 1000 times shorter for precision 10^{-30} than those found by the Solovay-Kitaev algorithm (see Figure 6.13).

6.1 Exact synthesis algorithm

The goal of this section is two-fold: first, to classify all single-qubit unitaries that can be implemented exactly by braiding Fibonacci anyons and second, to describe an efficient algorithm for finding a circuit that corresponds to a given exactly implementable unitary.

We start by introducing rings of integers and use them to describe the most general form of exactly implementable unitaries. Next, we introduce a complexity measure over the unitaries by using ring automorphisms. Finally, we describe the exact synthesis algorithm: a process, guided by the complexity measure, to find a circuit for an exactly synthesizable unitary.

For the purpose of this section it is more convenient to consider the elementary gates

$$\mathcal{T} = \begin{pmatrix} 1 & 0 \\ 0 & \zeta_{10} \end{pmatrix}, \mathcal{F} = \begin{pmatrix} \tau & \sqrt{\tau} \\ \sqrt{\tau} & -\tau \end{pmatrix}, \tau = \frac{\sqrt{5}-1}{2}, \zeta_{10} = e^{2i\pi/10}.$$

instead of σ_1 and σ_2 (see equations (1.1),(1.2)). We call a circuit composed of $\mathcal{F}, \mathcal{T}$ and $\zeta_{10}\mathcal{I}$ gates an $\langle \mathcal{F}, \mathcal{T} \rangle$-circuit, where $\mathcal{I}$ is the identity gate and $\zeta_{10}\mathcal{I}$ is a global phase gate. The following relations imply that any unitary that is implementable by an $\langle \mathcal{F}, \mathcal{T} \rangle$-circuit is also implementable by a $\langle \sigma_1, \sigma_2 \rangle$-circuit and vice-versa:

$$\begin{aligned} \sigma_1 &= (\zeta_{10}\mathcal{I})^6 \mathcal{T}^7, & \mathcal{T} &= (\zeta_{10}\mathcal{I})^2 (\sigma_1)^3, \\ \sigma_2 &= (\zeta_{10}\mathcal{I})^6 \mathcal{F}\mathcal{T}^7\mathcal{F}, & \mathcal{F} &= (\zeta_{10}\mathcal{I})^4 \sigma_1\sigma_2\sigma_1. \end{aligned} \tag{6.1}$$

For the applications considered in this work the global phase gate $\zeta_{10}\mathcal{I}$ is irrelevant.

Two rings are crucial for the exact synthesis algorithm: $\mathbb{Z}[\zeta_{10}]$ and $\mathbb{Z}[\tau]$. It is not difficult to check that both definitions presented in the main text are correct: the sets defined above are both rings. It is straightforward to check that both sets $\mathbb{Z}[\zeta_{10}]$ and $\mathbb{Z}[\tau]$ are closed under addition. To show that both sets are closed under multiplication it is sufficient to check the following equalities:

$$\zeta_{10}^4 = -1 + \zeta_{10} - \zeta_{10}^2 + \zeta_{10}^3, \quad \zeta_{10}^5 = -1, \quad \tau^2 = 1 - \tau.$$

The equality $\tau = \zeta_{10}^2 - \zeta_{10}^3$ implies that $\mathbb{Z}[\tau]$ is a subring of $\mathbb{Z}[\zeta_{10}]$. Both rings are well studied in algebraic number theory.

The proof of Theorem 2.4.4. We first show that any single qubit unitary over the Fibonacci gate set can written as follows

$$V[x, y, k] := \begin{pmatrix} x & y^* \sqrt{\tau} \zeta_{10}^k \\ y\sqrt{\tau} & -x^* \zeta_{10}^k \end{pmatrix}, \; x, y \in \mathbb{Z}[\zeta_{10}].$$

Let us first note that both $\mathcal{F}$ and $\mathcal{T}$ are unitaries of the form above. The product of a unitary $V[x, y, k]$ and $\mathcal{F}$ or $\mathcal{T}$ also can be again expressed in the form above. This shows

that any unitary corresponding to a circuit consisting of the $\mathcal{F}$ and $\mathcal{T}$ gates can be written as $V[x, y, k]$. The required statement follows from this, because σ_1 and σ_2 can be expressed in terms of $\mathcal{F}$ and $\mathcal{T}$ gates.

The proof of the converse statement follows from Theorem 6.1.3 proved later in this section. □

Recall that, an automorphism of $\mathbb{Z}[\zeta_{10}]$ that plays a fundamental role in the construction of the exact synthesis algorithm is the mapping

$$(.)^\bullet : \mathbb{Z}[\zeta_{10}] \mapsto \mathbb{Z}[\zeta_{10}] \text{ such that } \zeta_8^\bullet = \zeta_8^3. \tag{6.2}$$

By definition, a ring automorphism must preserve the sum and the product. For example, we find that

$$\tau^\bullet = (\zeta_8^2 - \zeta_8^3)^\bullet = (\zeta_8^2)^\bullet - (\zeta_8^3)^\bullet = (\zeta_8^\bullet)^2 - (\zeta_8^\bullet)^3 = -\varphi.$$

Taking into account that $\varphi = \tau + 1$ (i.e., φ is from $\mathbb{Z}[\tau]$) we see that $(.)^\bullet$ restricted on $\mathbb{Z}[\tau]$ is also an automorphism of $\mathbb{Z}[\tau]$. The other important property of automorphism $(.)^\bullet$ is its relation to complex conjugation (which is another automorphism of $\mathbb{Z}[\zeta_{10}]$):

$$(x^\bullet)^\bullet = x^*. \tag{6.3}$$

For example, this implies the equality $(x^*)^\bullet = (x^\bullet)^*$, which is used in several proofs in this work. It also implies that $|x^\bullet|^2 = (|x|^2)^\bullet$ because $|x|^2 = xx^*$. Recall that, the complexity measure μ is defined as $\mu(u) = (|u|^2)^\bullet$.

The notion of the Gauss complexity measure [52] $G(u) := \mu(u) + |u|^2$ is a useful tool in our proofs. The example in Figure 6.1 illustrates the intuition behind G and μ. It is always the case that $\mu(u) \leq G(u) \leq \mu(u) + 1$. The following proposition summarizes important properties of the Gauss complexity measure.

Proposition 6.1.1. *For any x from $\mathbb{Z}[\zeta_{10}]$, the Gauss complexity measure $G(x)$ is an integer, and there are four possibilities:*

$$\begin{aligned}
&(a) \quad G(x) = 0, \text{ then } x = 0,\\
&(b) \quad G(x) = 2, \text{ then } x = \zeta_{10}^k \text{ for } k \text{ — integer},\\
&(c) \quad G(x) = 3 \text{ then } x \in \left\{\tau\zeta_{10}^k, \varphi\zeta_{10}^k\right\}, k \text{ — integer},\\
&(d) \quad G(x) \geq 5,
\end{aligned}$$

and $G(a + b\zeta_{10} + c\zeta_{10}^2 + d\zeta_{10}^3) \leq 5/2\,(a^2 + b^2 + c^2 + d^2)$.

n	u_n	$G(u_n)$
0	1	1
1	$\zeta_{10}^2 - \zeta_{10}^3$	3
2	$2 - \zeta_{10} + \zeta_{10}^3$	13
3	$-3 + 5\zeta_{10} - 2\zeta_{10}^2 - 1$	57

Figure 6.1: Values of the Gauss complexity measure for the family of unitaries $U[u_n, v_n, k_n] = (\mathcal{FT})^n$, for n from $\{0, 1, 2, 3\}$.

Proof. As observed in [52], $G\left(a + b\zeta_{10} + c\zeta_{10}^2 + d\zeta_{10}^3\right)$ is a positive definite quadratic form when viewed as a function of a, b, c, d. We found by direct computation that it takes integer values when a, b, c, d are integers, its minimal eigenvalue is $1/2$ and its maximal eigenvalue is $5/2$. This implies an upper bound on G in terms of a, b, c, d and an inequality

$$G\left(a + b\zeta_{10} + c\zeta_{10}^2 + d\zeta_{10}^3\right) \geq (a^2 + b^2 + c^2 + d^2)/2.$$

The implication in (a) follows immediately from above. The inequality also allows us to prove the proposition by considering a small set of special cases

$$(a, b, c, d) \in S = \{-3, -2, -1, 0, 1, 2, 3\}^4.$$

In all other cases, $G\left(a + b\zeta_{10} + c\zeta_{10}^2 + d\zeta_{10}^3\right)$ is greater than eight, which corresponds to alternative (d). To finish the proof it is sufficient to exclude quadruples corresponding to $0, \zeta_{10}^k, \tau\zeta_{10}^k, \varphi\zeta_{10}^k$ from S and find that the minimum of $G\left(a + b\zeta_{10} + c\zeta_{10}^2 + d\zeta_{10}^3\right)$ over the remaining set is 5. □

To prove the correctness of the exact synthesis algorithm (Figure 2.6) we need the following technical result.

Lemma 6.1.2. *For any u, v from $\mathbb{Z}[\zeta_{10}]$ such that $\mu(u) \geq 2$, there exists $k_0(u, v)$ such that:*

$$(a) \quad \mu\left((u + \zeta_{10}^{k_0(u,v)} v)\tau\right)/\mu(u) < C_0 < 1,$$
$$(b) \quad \mu\left((u + \zeta_{10}^{k_0(u,v)} v)\tau\right)/\mu(u) < \tfrac{\varphi^2}{2}(\sqrt[4]{5} - 1)^2 + r_0(\mu(u)),$$
where $r_0(x)$ is in $O(1/x)$

If in addition $\mu(u) \geq 4$, *there exist* $k_1(u,v)$ *such that:*

$$\begin{aligned}
&(c) \quad \mu\left((u+\zeta_{10}^{2k_1(u,v)}v)\tau\right)/\mu(u) < C_1 < 1,\\
&(d) \quad \mu\left((u+\zeta_{10}^{2k_1(u,v)}v)\tau\right)/\mu(u) < \varphi^2(\varphi-\sqrt{\varphi}) + r_1(\mu(u)),\\
&\qquad \textit{where } r_1(x) \textit{ is in } O(1/x)
\end{aligned}$$

For any k, *the ratio* $\mu\left((u+\zeta_{10}^{k}v)\tau\right)/\mu(u)$ *is upper bounded by* $\varphi^2(1+\sqrt{\tau})^2$.

We first prove that the exact synthesis algorithm is correct and efficient and then proceed to the proof of Lemma 6.1.2. Theorem 2.4.4 is the corollary of the following theorem.

Theorem 6.1.3. *For any exact unitary* U *over the Fibonacci gate set:*

1. *the exact synthesis algorithm (Figure 2.6) terminates and produces a circuit that implements* U,
2. n, *the minimal length of* $\langle \mathcal{F}, \mathcal{T} \rangle$*-circuit implementing* U, *is in* $\Theta(\log(\mu(U)))$,
3. *the algorithm requires at most* $O(n)$ *arithmetic operations and outputs an* $\langle \mathcal{F}, \mathcal{T} \rangle$*-circuit with* $O(n)$ *gates*

Proof. The termination of the while loop in the algorithm follows the fact that by Lemma 6.1.2 the complexity measure of U strictly decreases by a constant factor. Indeed, from Figure 2.6 consider ratio $\mu(\mathcal{F}\mathcal{T}^j V)/\mu(V)$. If we denote the upper left entry of U_r by u and the lower left by $v\sqrt{\tau}$, then the ratio is precisely equal to the one considered in Lemma 6.1.2. By picking j that minimizes $\mu(\mathcal{F}\mathcal{T}^j V)$ we ensure that implications (a) and (b) of the Lemma hold. After the loop execution we have $\mu(V) < 2$. By relation between μ and G we have $G(u) \leq 2$ and according to Proposition 6.1.1, the only possible values of G of the upper left entry u of U are either 0 or 2. There is no exactly synthesizable unitary with $u = 0$. In other words, there is no v from $\mathbb{Z}[\zeta_{10}]$ such that equation $\tau|v|^2 = 1$ is solvable. The only remaining case is $\mu(u) = 2$. By Proposition 6.1.1, u must be a power of ζ_{10}, therefore V must be representable as $\zeta_{10}^k \mathcal{T}^j$. Correctness of the algorithm follows from the fact that during the algorithm execution at steps 3 and 6, it is always true that $U = U_C V$, where U_C denotes a unitary that is implemented by circuit C. This implies that any exactly synthesizable unitary U can be represented as an $\langle \mathcal{F}, \mathcal{T} \rangle$-circuit.

Now we prove the second and the third statements. Taking into account that $\mathcal{F}^2 = \mathcal{I}$ and $\mathcal{T}^{10} = \mathcal{I}$, any exact unitary can be represented as the following matrix product

$$\zeta_{10}^{k(m+1)} \mathcal{T}^{k(m)} \mathcal{F} \mathcal{T}^{k(m-1)} \mathcal{F} \mathcal{T}^{k(m-2)} \dots \mathcal{F} \mathcal{T}^{k(0)} \tag{6.4}$$

where $k(j)$ are from $\{0, \dots, 9\}$. The exact synthesis algorithm produces the circuit that leads precisely to representation (6.4) and m in this case is the number of steps performed by the algorithm. Lemma 6.1.2 implies that m is upper bounded by $\log_3(\mu(U)) + c_1$, as $\frac{\varphi^2}{2}(\sqrt[4]{5} - 1)^2 < 1/3$. This gives an upper bound on the length of the circuit produced by the algorithm and also on n, the minimal possible length of any circuit that implements U.

Consider now representation (6.4) obtained from a minimal length $\langle \mathcal{F}, \mathcal{T} \rangle$-circuit implementing U. We prove a bound on $\mu(U)$ by induction. First, by direct computation, $\mu\left(\mathcal{F}\mathcal{T}^{k(0)}\right)$ is less than three. Next, we introduce $V_j = \mathcal{F}\mathcal{T}^{k(j-1)} \dots \mathcal{F}\mathcal{T}^{k(0)}$ and by the third part of Lemma 6.1.2 find

$$\mu\left(\mathcal{F}\mathcal{T}^{k(j)} V_j\right) \leq \varphi^2 (1 + \sqrt{\tau})^2 \mu(V_j).$$

We note that multiplication by $\zeta_{10}^{k(m+1)} \mathcal{T}^{k(m)}$ does not change the complexity measure and conclude that $\log(\mu(U))$ is upper bounded by some linear function of m. It is not difficult to see that m is less than n. This finishes the proof of the theorem. □

The important property of automorphism $(\cdot)^\bullet$ used below is its relation to complex conjugation $(x^\bullet)^\bullet = x^*$. In particular, it implies that $|u^\bullet|^2 = (|u|^2)^\bullet$.

Proof of Lemma 6.1.2. To estimate ratio $\mu\left((u + \zeta_{10}^k v)\tau\right) / \mu(u)$ we first expand $\left|(u + \zeta_{10}^k v)^\bullet\right|^2$ and see that it is equal to

$$|u^\bullet|^2 + |v^\bullet|^2 + 2\mathrm{Re}(u^\bullet (v^\bullet)^* \zeta_{10}^{-3k}).$$

Here we used that $\zeta_{10}^\bullet = \zeta_{10}^3$. It is convenient to introduce

$$e^{i\phi} := \frac{u^\bullet}{|u^\bullet|} \left(\frac{v^\bullet}{|v^\bullet|}\right)^*, \ \alpha := \frac{|v^\bullet|^2}{|u^\bullet|^2}.$$

In terms of $\alpha, e^{i\phi}$ we find

$$\frac{\left|(u + \zeta_{10}^k v)^\bullet\right|^2}{|u^\bullet|^2} = 1 + \alpha + 2\mathrm{Re}\left(e^{i(\phi - 3\pi k/5)}\right)\sqrt{\alpha}. \tag{6.5}$$

It is always possible to choose $k_0(u,v)$ and $k_1(u,v)$ such that for $k = k_0(u,v)$ and $k = 2k_1(u,v)$ the right hand side of the equation above is upper bounded by $f_0(\alpha) := 1 + \alpha - 2\sqrt{\alpha}\cos(\pi/10)$ and $f_1(\alpha) := 1 + \alpha - 2\sqrt{\alpha}\cos(\pi/5)$ correspondingly. Next we note that α is a function of $\mu(u)$. Indeed, $|u|^2 + \tau|v|^2 = 1$ implies

$$1 = (|u|^2 + \tau|v|^2)^\bullet = |u^\bullet|^2 - \varphi|v^\bullet|^2 .$$

We find that $\alpha = \tau(\mu(u)-1)/\mu(u)$. To obtain estimates $(a), (c)$ we notice that $f_0(\alpha(\mu(u)))$ and $f_1(\alpha(\mu(u)))$ are both monotonically decreasing functions of $\mu(u)$. Therefore we can choose C_0 to be the value of $f_0(\alpha)$ when $\mu(u) = 2$ and C_1 to be value of $f_1(\alpha)$ when $\mu(u) = 4$. In both cases we find by direct computation that C_0 and C_1 are strictly less than 1. To prove (c) and (d) we define $r_0(\mu(u)) = \varphi^2 f_0(\alpha(\mu(u)) - \frac{\varphi^2}{2}(\sqrt[4]{5} - 1)^2$ and $r_1(\mu(u)) = \varphi^2 f_1(\alpha(\mu(u)) - \varphi^2(\varphi - \sqrt{\varphi})$. Again by direct computation we find that $r_{1,2}(x)$ are in $O(1/x)$. □

6.2 Norm Equation

In this section we summarize the tools needed to reconstruct an exact unitary given one of its entries. This is a crucial part of the approximation algorithm presented in the next section. As we will discuss, in the most general setting this problem is hard. Here we are in particular interested in identifiying the instances of the problem that can be solved in probabilistic polynomial time. We present efficient algorithms for both: the identification of an "easy" instances and solving them. This is a short summary of a more detailed discussion of this topic presented in [46].

There are two distinct cases of an exact unitary reconstruction problem and both are reduced to the problem of finding x from $\mathbb{Z}[\zeta_{10}]$ such that

$$|x|^2 = \xi \text{ for } \xi \text{ from } \mathbb{Z}[\tau] .$$

The first case is when we are interested in finding an exact unitary $V[u,v,k]$ given u and the second is when we given v; in the first case we find v by solving the equation with $\xi = \varphi(1 - |u|^2)$ and in the second we find u by considering $\xi = 1 - \tau|v|^2$. The equation we introduced called a relative norm equation because mapping $|\cdot|^2$ from $\mathbb{Z}[\zeta_{10}]$ to $\mathbb{Z}[\tau]$ is a relative norm of a relative extension $\mathbb{Z}[\zeta_{10}]/\mathbb{Z}[\tau]$.

We first discuss how to identify "easy" instances of the problem. We present an efficiently computable criterion for these instances summarized on Figure 6.2. The key

Input: $\xi \in \mathbb{Z}[\tau]$

```
1: procedure EASY-SOLVABLE(ξ)
2:     if ξ < 0 or ξ• < 0 then
3:         return FALSE
4:     end if
5:     n ← N_τ(ξ)
6:     while n = 0 mod 5 do
7:         n ← n/5
8:     end while
9:     if n = 1 mod 5 then
10:        return IS-PRIME(n)
11:    else
12:        return FALSE
13:    end if
14: end procedure
```

Output: TRUE if ξ is an easy and solvable instance of the problem: find x from $\mathbb{Z}[\zeta_{10}]$ such that $|x|^2 = \xi$; FALSE otherwise.

Figure 6.2: Procedure EASY-SOLVABLE. Checks if the given instance of the norm equation can be solved in polynomial time.

observation for this and for solving the more general instances of the problem is that $|\cdot|^2$ is a multiplicative function. In other words, if the right-hand side of the equation can be written as $\xi_1 \cdot \xi_2$ we can find solutions for each ξ_i and combine them to get solution to the original equation. For this reason, the relation of divisibility on elements of $\mathbb{Z}[\tau]$ is one of the basic tools we are using. We say that ξ_1 divides ξ if there exist ξ_2 from $\mathbb{Z}[\tau]$ such that $\xi = \xi_1\xi_2$. As usual, we use notation $\xi_1|\xi$ for this relation. In this case we also write $\xi{=}0 \bmod \xi_1$, or, more generally $\eta_1{=}\eta_2 \bmod \xi_1$ when $\xi_1|(\eta_2 - \eta_1)$. The alternative way of understanding the divisibility of elements of $\mathbb{Z}[\tau]$ is using an algebraic norm of $\mathbb{Z}[\tau]$

$$N_\tau(a + b\tau) := (a + b\tau)(a + b\tau)^\bullet = a^2 - ab - b^2.$$

It turns out that $\xi_1|\xi$ if and only if $N_\tau(\xi_1)\,|N_\tau(\xi)$. This way we can think about divisibility of elements of $\mathbb{Z}[\tau]$ using more familiar notion of divisibility of usual integers. Using N_τ we define the most basic instances of the norm equation problem — elements of $\mathbb{Z}[\tau]$ such that their algebraic norm N_τ is a prime number. It turns out that they are not just basic but are the ones that can be solved in probabilistic polynomial time (as we show later in

this section). We call an instance of the problem "easy" if for the right-hand side ξ

$$N_t(\xi) = 5^n p, \text{ where } n \text{ is an integer and } p \text{ is prime.} \tag{6.6}$$

We use a polynomial time primality test (subroutine IS-PRIME in procedure EASY-SOLVABLE) to efficiently determine if the instance is "easy".

The following conditions are necessary and sufficient for an "easy" instance of the equation to be solvable:

- $\xi > 0, \xi^\bullet > 0$
- $p = 5n + 1$ for integer n.

Condition $\xi > 0$ required because $|x|^2$ is always positive. The same must hold for $\xi^\bullet$ because $|x^\bullet|^2 = (|x|^2)^\bullet$ (this follows from the relation between automorphism $(\cdot)^\bullet$ and complex conjugation discussed in Section 6.1). The condition on p is more involved and we will discuss it further in this section. All these conditions are checked by procedure EASY-SOLVABLE; if EASY-SOLVABLE returns TRUE there exists a solution to the norm equation that can be found efficiently.

Now we discuss an efficient method for solving "easy" instances of the problem. The representation (6.6) of $N_\tau(\xi)$ implies that ξ itself can be represented as $\xi_1(2-\tau)^n$. It is not difficult to find that $N_\tau(2-\tau) = 5$. However, an element $2-\tau$ is not the only element of $\mathbb{Z}[\tau]$ with an algebraic norm 5 and it is not obvious a priori that the mentioned representation of ξ always exists. We postpone the discussion of this until we introduce the required mathematical concepts further in the section. The problem of solving an "easy" instance is now reduced to two subproblems. The first one is to solve $|x|^2 = 2 - \tau$. It is not difficult to check that $\zeta_{10} + \zeta_{10}^4$ is a solution. The second subproblem is $|x|^2 = \xi_1$ where $N_\tau(\xi_1)$ is prime. The rest of the section is devoted to finding a solution to it in an efficient way.

To find a solution efficiently we exploit the relation between norm equations and divisibility on a deeper level. This time we need a relation of divisibility defined on elements of $\mathbb{Z}[\zeta_{10}]$, which can be defined similarly to the way we did it for $\mathbb{Z}[\tau]$. We rewrite the equation as

$$x \cdot x^* = \xi_1.$$

If there is a solution x this will mean that x divides ξ_1 when they both considered as elements of $\mathbb{Z}[\zeta_{10}]$. Therefore we need to search for the solution to the norm equation inside the set of possible divisors of ξ_1. As ξ_1 is prime in $\mathbb{Z}[\tau]$ this suggests there are

not too many possible divisors. To make this intuition precise we need to understand the relation of divisibility a bit deeper.

Consider an example with usual integers. Suppose p is prime and integer p' divides it. There are only two possibilities for p': it is either p or $-p$. In other words, p' is unique up to a factor -1 — a unit of the ring of integers $\mathbb{Z}$. One can also say that p is the only divisor of p non equal to 1 up to invertible elements of $\mathbb{Z}$. The situation is very similar in the ring $\mathbb{Z}[\tau]$. Its set of units is a set of all elements u such that N_τ is equal to 1 and for each u there is an inverse that is equal to $u^\bullet$. Any unit u can be represented as $\pm\tau^n$ for integer n. Procedure UNIT-LOG shows an efficient way of finding this representation which will be useful for us later. Consider now η from $\mathbb{Z}[\tau]$ such that $N_\tau(\eta)$ is prime. It is possible to show that if η' from $\mathbb{Z}[\tau]$ divides η then η' equals to $s\tau^n\eta$ for s equal to 1 or -1 and integer n. Now we can provide some high-level idea as to why the representation (6.6) for N_τ implies that ξ can be always written as $\xi_1(2-\tau)^n$. For given η with prime algebraic norm there is at most one other prime that has the same algebraic norm and is not equal to η up to a multiplication by unit. If there is one it equals to $\eta^\bullet$. The only case when $\eta^\bullet$ is equal to η up to a unit is when $\eta = 2-\tau$.

Moving from divisibility in $\mathbb{Z}[\tau]$ to $\mathbb{Z}[\zeta_{10}]$ gives us more freedom. For example, $2-\tau$ is divisible by $\zeta_{10}+\zeta_{10}^4$. It is not difficult to check that $\zeta_{10}+\zeta_{10}^4$ is purely imaginary, therefore there is no contradiction to our previous observation that $2-\tau$ is divisible only by itself up to a unit when divisibility is considered over $\mathbb{Z}[\tau]$. Condition $N_\tau(\xi_1) = 5n+1$ ensures that there exists element z of $\mathbb{Z}[\zeta_{10}]$ such that $x_r x_r^*$ is equal to ξ_1 up to multiplication by a unit in $\mathbb{Z}[\tau]$. Finding x_r is a key step in solving the norm equation. Once it is done we get $|x_r|^2 = u\xi_1$ for some unit u in $\mathbb{Z}[\tau]$. Conditions $\xi > 0$ and $\xi^\bullet$ ensure that unit u is a square of some other unit v and therefore $|x_r v|^2 = \xi_1$.

To find x_r we construct an element z of $\mathbb{Z}[\zeta_{10}]$ with the following properties

- $\xi_1 \nmid z$ and $\xi_1 \nmid z^*$
- $\xi_1 | zz^*$

If the conditions above are satisfied either x_r or x_r^* must divide both ξ_1 and z and we can find x_r by using the algorithm for Greatest Common Divisor in $\mathbb{Z}[\zeta_{10}]$ (Procedure BINARY-GCD). To ensure that the first group of conditions is satisfied we search for z in the form $M+\zeta_{10}+\zeta_{10}^4$ for M — integer. The key here is that $\zeta_{10}+\zeta_{10}^4$ is a purely imaginary number and divisibility of z or z^* by ξ would imply that ξ_1 divides $\zeta_{10}+\zeta_{10}^4$ which is not possible. The second condition can be rewritten as

$$\xi_1 | M^2 + (2-\tau).$$

Input: unit $u = a + b\,\tau \in U(\mathbb{Z}[\tau])$

procedure UNIT-DLOG(u)
  $s \leftarrow 1$, $k \leftarrow 0$
  if $a < 0$ **then**
    $a \leftarrow -a$; $b \leftarrow -b$; $s \leftarrow -s$
  end if
  $\mu \leftarrow a\,b$
  while $|\mu| > 1$ **do**
    if $\mu > 1$ **then**
      $(a, b) \leftarrow (b, a - b)$; $k \leftarrow k - 1$
    else
      $(a, b) \leftarrow (a, a - b)$; $k \leftarrow k + 1$
    end if
    $\mu \leftarrow a\,b$
  end while ▷ $|\mu| = 1$ here
  match $v = a + b\,\tau$ with one of the $\{\pm 1, \pm\tau, \pm\tau^2, \pm\tau^{-1}\}$
  adjust s, k accordingly
  return (s, k)
end procedure

Output: (s, k) such that $s = -1, 1, k$ – integer and $u = s\tau^k$

Figure 6.3: Procedure UNIT-DLOG. Finds a discrete logarithm of the unit u. The procedure runtime is in $O(\log(\max\{|a|, |b|\}))$. Reprinted from [46]

.

We reduce a problem of finding integer M to a problem of taking square root modulo p which can be solved in probabilistic polynomial time by the Tonelli-Shanks Algorithm (procedure TONNELI-SHANKS). Primality of $N_\tau(\xi_1)$ implies that there exist t such that $t = \tau \bmod \xi_1$. Even more, it can be computed in polynomial time (procedure TAU-TO-INT). This implies that $\xi_1 | M^2 + (2 - t)$ which is equivalent to $p | M^2 + (2 - t)$; this can be written as $M^2 = (2 - t) \bmod p$. In other words, M is a square root of $(2 - t)$ modulo p; its existence follows from the condition $p = 5n + 1$.

To conclude that we can solve "easy" instances in probabilistic polynomial time it remains to show that the greatest common divisor of two elements of $\mathbb{Z}[\zeta_{10}]$ can be found in polynomial time. The BINARY-GCD procedure corresponds to such an algorithm based on [11].

Input: $a + b\tau \in \mathbb{Z}[\tau]$, assume EASY-SOLVABLE$(a + b\tau)$ is TRUE

procedure SOLVE-NORM-EQUATION$(a + b\tau)$
▷ Find integer n and prime p such that $N(a + b\tau) = 5^n p$
  $n \leftarrow 0$
  while $(2 - \tau) | a + b\tau$ **do**
    $a + b\tau \leftarrow (a + b\tau)/(2 - \tau)$
    $n \leftarrow n + 1$
  end while ▷ $N_\tau(a + b\tau) = p$
▷ Solve the norm equation for prime righthand side
  $p \leftarrow N_\tau(a + b\tau)$ ▷ p is prime
  $t \leftarrow (\text{TAU-TO-INT}(a + b\tau) - 2) \bmod p$
  $M \leftarrow \text{TONELLI-SHANKS}(t, p)$ ▷ $M^2 = t \bmod p$
  $x_r \leftarrow \text{BINARY-GCD}(a + b\tau, M - (\zeta_{10} + \zeta_{10}^4))$
▷ $(\zeta_{10} + \zeta_{10}^4)^2 = \tau - 2$
  $u \leftarrow (a + b\tau)/|x_r|^2$ ▷ u — unit, $u > 0$, $u^\bullet > 0$
  $(s, m) \leftarrow \text{UNIT-DLOG}(u)$ ▷ $s = 1$, m — even
  return $\tau^{m/2}(\zeta_{10} + \zeta_{10}^4)^{n_1} x_r$
end procedure

Output: x from $\mathbb{Z}[\zeta_{10}]$ such that $|x|^2 = a + b\tau$

Figure 6.4: Procedure SOLVE-NORM-EQUATION. Finds a solution to an "easy" instance of the norm equation in probabilistic polynomial time. Based on the similar procedure from [46].

6.3 Approximation

In this section we combine methods developed in previous sections and describe an algorithm for approximating unitaries of the form $R_z(\phi)$, $R_x(\phi)$ and $R_z(\phi)X$ with $\langle \mathcal{F}, \mathcal{T} \rangle$-circuits (or equivalently, $\langle \sigma_1, \sigma_2 \rangle$-circuits). Using this method we can approximate any single qubit unitary as it can always be represented as $R_z(\alpha_1)R_x(\alpha_2)R_z(\alpha_3)$. We consider the class of unitaries $R_z(\phi)X$ separately because it includes a commonly used Pauli X gate. Recall that we measure the quality of approximation ε using the global phase-invariant distance

$$d(U, V) = \sqrt{1 - |tr(UV^\dagger)|/2}. \tag{6.7}$$

The quality of approximation ε defines the problem size. Our algorithm produces circuits of length $O(\log(1/\varepsilon))$ which meets the asymptotic worst-case lower bound [32] for such a circuit. The algorithm is probabilistic in nature and on average requires running time in

Input: $a + b\tau \in \mathbb{Z}[\tau], a + b\tau$ — prime

1: **procedure** TAU-TO-INT($a + b\tau$)

2: $p \leftarrow N_\tau(a + b\tau)$ ▷ $GCD(p, b) = 1$

3: $(u, v) \leftarrow$ EXTENDED-EUCLIDEAN(b, p)

4: **return** $-au \bmod p$ ▷ $bu + (v(a - b\tau))(a + b\tau) = 1$

5: **end procedure**

Output: m — integer such that $m = \tau \bmod (a + b\tau)$

Figure 6.5: Procedure TAU-TO-INT. Finds an integer m such that $m = \tau \bmod (a + b\tau)$. The procedure runtime is in $O(\log(\max\{|a|, |b|\}))$.

$O(\log^c(1/\varepsilon))$ to find an approximation where c is a constant close to but smaller than 2.0, according to our empirical estimates .

We first discuss all details of the algorithm for approximating $R_z(\phi)$ and then show how the same tools allow us to find approximations of $R_z(\phi)X$.

There are two main stages in our algorithm: the first stage approximates $R_z(\phi)$ with an exact unitary $V[u, v, 0]$ and then uses the exact synthesis algorithm (Figure 2.6) to find a circuit implementing $V[u, v, 0]$. The second stage is completely described in Section 6.1; here we focus on the first stage.

The expression for the quality of approximation in the first case simplifies to

$$d(R_z(\phi), V[u, v, 0]) = \sqrt{1 - |\mathrm{Re}(ue^{i\phi/2})|}.$$

We see that the quality of approximation depends only on u, the top left entry of $V[u, v, 0]$. Therefore, to solve the first part of the problem it is sufficient to find u from $\mathbb{Z}[\zeta_{10}]$ such that $\sqrt{1 - |\mathrm{Re}(ue^{i\phi/2})|} \leq \varepsilon$. However, there is an additional constraint that must be satisfied: there must exist a v from $\mathbb{Z}[\zeta_{10}]$ such that $V[u, v, 0]$ is unitary, in other words the following equation must be solvable:

$$|v|^2 = \xi, \text{ for } \xi = \varphi(1 - |u|^2). \tag{6.8}$$

This is precisely the equation studied in Section 6.2. As discussed, in general the problem of deciding whether such a v exists and then finding it is hard. It turns out, however, that in our case there is enough freedom to pick (find) "easy" instances and obtain a solution in polynomial time without sacrificing too much quality.

There is an analogy to this situation: it is well known that factoring a natural number into prime factors is a hard problem. However, checking that the number is prime can be

Input: ϕ — defines $R_z(\phi)$, ε — precision

$C \leftarrow \sqrt{\varphi/4}$
$m \leftarrow \lceil \log_\tau(C\varepsilon) \rceil + 1$
Find k such that $\theta = -\phi/2 - \pi k/5 \in [0, \pi/5]$
not-found $\leftarrow$ true, $u \leftarrow 0, v \leftarrow 0$
while *not-found* **do**
    $u_0 \leftarrow$ RANDOM-SAMPLE$(\theta, \varepsilon, 1)$ ▷ See Figure 6.7
    $\xi \leftarrow \varphi\left(\varphi^{2m} - |u_0|^2\right)$
    $fl \leftarrow$ EASY-FACTOR(ξ)
    if EASY-SOLVABLE(fl) **then**
        not-found $\leftarrow$ false
        $u \leftarrow \zeta_{10}^k \tau^m u_0$
        $v \leftarrow \tau^m$SOLVE-NORM-EQUATION(ξ)
    end if
end while
$C \leftarrow$ EXACT-SYNTHESIZE$(V[u, v, 0])$

Output: Circuit C such that $d(C, R_z(\phi)) \leq \varepsilon$

Figure 6.6: The algorithm for approximating $R_z(\phi)$ by an $\langle \mathcal{F}, \mathcal{T} \rangle$-circuit with $O(\log(1/\varepsilon))$ gates and precision at most ε. Runtime is probabilistically polynomial as a function of $\log(1/\varepsilon)$.

done in polynomial time. In other words, given a natural number N one can efficiently decide if it is an easy instance for factoring. Now imagine the following game: one is given a uniformly chosen random number from the interval $[0, N]$ and one wins each time they can factor it. How good is this game? The key here is the Prime Number Theorem. It states that there are $\Theta(N/\log(N))$ primes in the interval $[0, N]$. Therefore one can win the game with probability at least $\Omega(1/\log(N))$. In other words, the number of trials one needs to make before winning scales as $O(\log(N))$. In our case the situation is somewhat similar and N is of order $1/\varepsilon$.

At a high level, during our approximation procedure (Figure 6.6) we perform a number of trials. During each trial we first randomly pick a u from $\mathbb{Z}[\zeta_{10}]$ that achieves precision ε and then check that the instance of the norm equation can be easily solved. Once we find such an instance we compute v and construct a unitary $V[u, v, 0]$.

We generate a random element u from $\mathbb{Z}[\zeta_{10}]$ that has the desired precision using procedure RANDOM-SAMPLE. To achieve a better constant factor in front of $\log(1/\varepsilon)$ for the length of the circuit we randomly chose $u_0 = u\varphi^m$ instead of u. It is easy to recover

Input: θ — angle between 0 and $\pi/5$, ε — precision, $r \geq 1$

procedure RANDOM-SAMPLE(θ, ε, r) ▷ See Fig. 6.8
  $C \leftarrow \sqrt{\varphi/(4r)}$
  $m \leftarrow \lceil \log_\tau(C\varepsilon r) \rceil + 1$
  $N \leftarrow \lceil \varphi^m \rceil$
  $y_{min} \leftarrow r\varphi^m(\sin(\theta) - \varepsilon\left(\sqrt{4-\varepsilon^2}\cos(\theta) + \varepsilon\sin(\theta)\right)/2)$
  $y_{max} \leftarrow r\varphi^m(\sin(\theta) + \varepsilon\left(\sqrt{4-\varepsilon^2}\cos(\theta) - \varepsilon\sin(\theta)\right)/2)$
  $x_{max} \leftarrow r\varphi^m((1-\varepsilon^2/2)\cos(\theta) - \varepsilon\sqrt{1-\varepsilon^2/4}\sin(\theta))$
  $x_c \leftarrow x_{max} - r\varepsilon^2\varphi^m/(4\cos(\theta))$
  Pick random integer j from $[1, N-1]$
  $y \leftarrow y_{min} + j(y_{max} - y_{min})/N$
  $a_y + \tau b_y \leftarrow$ APPROX-REAL($y/\sqrt{2-\tau}, m$) ▷ Fig. 6.9
  $x \leftarrow x_c - ((a_y + b_y\tau)\sqrt{2-\tau} - y_{min})\tan(\theta)$
  $a_x + \tau b_x \leftarrow$ APPROX-REAL(x, m) ▷ Fig. 6.9
  return $a_x + \tau b_x + \sqrt{\tau - 2}(a_y + \tau b_y)$
end procedure

Figure 6.7: The algorithm for picking a random element of $\mathbb{Z}[\zeta_{10}]$ that is in the dark grey region in Figure 6.8. Number of different outputs of the algorithm is in $O(1/\varepsilon)$.

u as $\tau = \varphi^{-1}$ and $u = u_0\tau^n$.

In Figure 6.8, when $r = 1$, the light grey circular segment corresponds to such u_0 that $V[u, v, 0]$ is within ε from $R_z(\phi)$. The element u_0 is a complex number and, as usual, the x-axis of the plot corresponds to the real part and the y-axis to the imaginary part. All random samples that we generate belong to the dark grey parallelogram and have the form $a_x + b_x\tau + i\sqrt{2-\tau}(a_y + b_y\tau)$ (note that $i\sqrt{2-\tau}$ is equal to $\zeta_{10} + \zeta_{10}^4$ and belongs to $\mathbb{Z}[\zeta_{10}]$). We first randomly choose an imaginary part and then a real part. To find an imaginary part we randomly choose a real number y and then approximate it with $\sqrt{2-\tau}(a_y + b_y\tau)$ using the APPROX-REAL (Figure 6.9) procedure. Once we find $\sqrt{2-\tau}(a_y + b_y\tau)$, we choose the x-coordinate as shown in Figure 6.8 and approximate it with $a_x + b_x\tau$.

The problem of approximating real numbers with numbers of the form $a+bz$ for integers a, b and irrational z is well studied. The main tool is continued fractions. It is also well known that Fibonacci numbers $\{F_n\}$,

$$F_0 = 0, F_1 = 1, F_n = F_{n-1} + F_{n-2}, n \geq 2,$$

are closely related to the continued fraction of the Golden section φ and its inverse τ. The correctness of procedure APPROX-REAL (Figure 6.9) is based on the following very

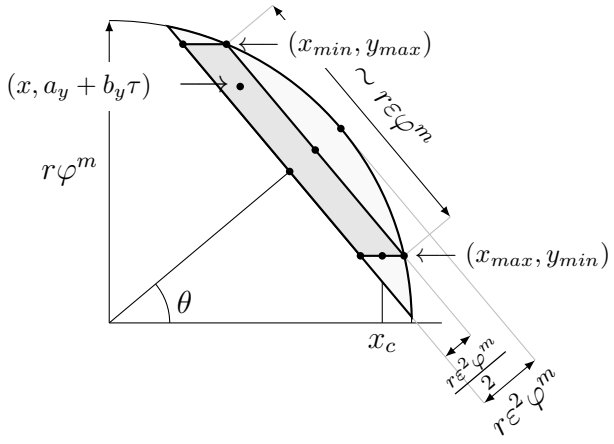

Figure 6.8: An ε-region and visualization of variables used in RANDOM-SAMPLE procedure (Figure 6.7).

well-known result connecting τ and the Fibonacci numbers; we state it in a convenient form and provide the proof for completeness.

Proposition 6.3.1. *For any integer n*

$$\left|\tau - \frac{F_n}{F_{n+1}}\right| \leq \frac{\tau^n}{F_{n+1}}$$

and $F_n \geq (\varphi^n - 1)/\sqrt{5}$.

Proof. First we define the family of functions $\{f_n\}$ such that $f_1(x) = x$ and $f_n(x) = 1/(1 + f_{n-1}(x))$ for $n \geq 2$. It is not difficult to check that $f_n(\tau) = \tau$. It can be shown by induction on n that $f_n(1)$ is equal to F_n/F_{n+1}. Therefore, to prove the first part of the proposition we need to show that

$$|f_n(\tau) - f_n(1)| \leq \tau^n/F_{n+1}.$$

We proceed by induction. The statement is true for n equal to 1. Using the definition of f_n it is not difficult to show

$$|f_{n+1}(\tau) - f_{n+1}(1)| = f_{n+1}(\tau)f_{n+1}(1)\,|f_n(\tau) - f_n(1)|\,.$$

Using $f_{n+1} = \tau$ and the inequality for $|f_n(\tau) - f_n(1)|$ we complete the proof of the first part.

Input: x — real number, n — defines precision as $\tau^{n-1}(1-\tau^n)$

1: **procedure** APPROX-REAL(x,n)
2: $p \leftarrow F_n, q \leftarrow F_{n+1}$ ▷ F_n — Fibonacci numbers
3: $u \leftarrow (-1)^{n+1}F_n, v \leftarrow (-1)^n F_{n-1}$ ▷ $up + vq = 1$
4: $c \leftarrow \lfloor xq \rceil$
5: $a \leftarrow cv + p\lfloor cu/q \rceil$
6: $b \leftarrow cu - q\lfloor cu/q \rceil$
7: **return** $a + \tau b$
8: **end procedure**

Output: $a + b\tau$ s.t. $|x - (a + b\tau)| \leq \tau^{n-1}(1-\tau^n), |b| \leq \varphi^n$

Figure 6.9: The algorithm for finding integers a, b such that $a + b\tau$ approximates the real number x with precision $\tau^{n-1}(1-\tau^n)$.

To show the second part it is enough to use the well-known closed form expression for Fibonacci numbers $F_n = (\varphi^n - (-\tau)^n)/\sqrt{5}$.

□

At a high level, in procedure APPROX-REAL we approximate τ with a rational number p/q and find the approximation of x by a rational of the form $a+bp/q$. To get good resulting precision we need to ensure that $b(\tau - p/q)$ is small, therefore we pick b in such a way that $|b| \leq q/2$. The details are as follows.

Lemma 6.3.2. *Procedure APPROX-REAL (Figure 6.9) outputs integers a, b such that $|x - (a + b\tau)| \leq \tau^{n-1}(1-\tau^n)$, $|b| \leq \varphi^n$ and terminates in time polynomial in n.*

Proof. First, the identity $F_{n+1}F_{n-1} - F_n^2 = (-1)^n$ for Fibonacci numbers implies $uv + pq = 1$. Next, by choice of c we have $|xq - c| \leq 1/2$, therefore $|x - c/q| \leq 1/2q$. By Proposition 6.3.1, $1/2q$ is less than $\tau^n\sqrt{5}(1-\tau^n)/2$; it remains to show that c/q is within distance $\tau^n/2$ from $a + b\tau$ by the triangle inequality. By choice of a, b we have $c = aq + bp$ and

$$|c/q - (a + b\tau)| = |b|\,|\tau - p/q|\,.$$

Using Proposition 6.3.1, equality $q = F_{n+1}$ and inequality $|b| \leq q/2$ we conclude that $|c/q - (a + b\tau)| \leq \tau^n/2$.

The complexity of computing the n^{th} Fibonacci number is polynomial in n. Assuming that the number of bits used to represent x is proportional to n all arithmetic operations also have complexity polynomial in n. □

There are two main details of the RANDOM-SAMPLE procedure we need to clarify. The first is that the result is indeed inside the dark grey parallelogram on Figure 6.8. This is achieved by picking real values x, y far enough from the border of the parallelogram and then choosing the precision parameter m for APPROX-REAL in such a way that $a_x + b_x\tau(\text{close to } x)$ and $a_y + b_y\tau(\text{close to } y/\sqrt{2-t})$ stay inside the parallelogram.

The second important detail is the size of resulting coefficients a_x, b_x, a_y, b_y. It is closely related to the number of gates in the resulting circuit. Therefore it is important to establish an upper bound on the sizes of the coefficients. The following lemma provides a rigorous summary:

Lemma 6.3.3. *When the third input $r \geq 1$, procedure RANDOM-SAMPLE has the following properties:*

- *there are $O(1/\varepsilon)$ different outputs and each of them occurs with the same probability,*
- *the procedure outputs an element u_0 of $\mathbb{Z}[\zeta_{10}]$ from the dark grey parallelogram P in Figure 6.8,*
- *the complexity measure μ of u_0 is in $O(1/\varepsilon)$.*

Proof. By construction, the algorithm produces $N-1$ outputs with equal probability. It is not difficult to check that N is in $O(1/\varepsilon)$. We first show that the outputs are all distinct and their y coordinate is in $[y_{min}, y_{max}]$. This follows from an estimate

$$|y - (a_y + b_y\tau)|\sqrt{2-\tau} \leq (y_{max} - y_{min})/2N$$

because each randomly generated y is at least distance $(y_{max} - y_{min})/N$ from any other randomly generated y and also y_{min}, y_{max}. To show the estimate we use the result of Lemma 6.3.2 and check that

$$\tau^{m-1}(1-\tau^m)\sqrt{2-\tau} \leq (y_{max} - y_{min})/2N$$

which is straightforward, but tedious. We concentrate only on terms that are first order in ε (we use $\lesssim$ and $\gtrsim$ to emphasize this):

$$\tau^{m-1}(1-\tau^m) \lesssim C\varepsilon r, (y_{max} - y_{min})/2N \gtrsim \varepsilon\cos(\theta)r. \tag{6.9}$$

The constraint on θ gives $\cos(\theta) \geq \varphi/2$. From $C\sqrt{2-\tau} < \varphi/2$ we conclude that the inequality is true for the terms that are first order in ε.

To show that the procedure output belongs to the parallelogram P, it is sufficient to check that $a_x + b_x\tau$ is within distance $\varphi^m r\varepsilon^2/4$ (half of the parallelogram height) from $x_c - (a_y + b_y\tau - y_{min})\sin(\theta)$. Again using Lemma 6.3.2, it is sufficient to show that

$$\tau^{m-1}(1-\tau^m) \leq \varphi^m r\varepsilon^2/4.$$

We again analyze the expression up to the first order terms in ε. We note that $\varphi^m r\varepsilon^2/4 \approx \varepsilon/(4C\tau)$; combining it with the inequality above, using (6.9) and $C = \sqrt{\varphi/(4r)}$, we conclude that all outputs of the algorithm are inside parallelogram P.

To show the last property, we note that by Lemma 6.3.2 both $|b_x|, |b_y|$ are bounded by φ^m. The same is true for $|a_x|, |a_y|$. Indeed, $|x|, |y|$ are both of order φ^m and

$$|a_y| \lesssim |y/\sqrt{2-\tau} - b_x\tau| + C\varepsilon r,\ |x_y| \lesssim |x - b_x\tau| + C\varepsilon r.$$

This implies that if we write u_0 as $\sum_k \alpha_k \zeta_{10}^k$ each integer α_k will be of order φ^m which is the same as $O(1/\varepsilon)$. Using the upper bound on the Gauss complexity measure $G(u_0)$ in terms of α_k from Proposition 6.1.1 we conclude that $G(u_0)$ and $\mu(u_0)$ is in $O(1/\varepsilon)$. □

The technical tools that we have developed so far are sufficient to verify that our approximation algorithm achieves the required precision and produces circuits of length $O(\log(1/\varepsilon))$. The remaining part is to show that on average the algorithm requires $O(\log^c(1/\varepsilon))$ steps. It relies on the following conjecture, similar in nature to the Prime Number Theorem:

Conjecture 6.3.4. *Let $\pi(M)$ be the number of elements ξ from $\mathbb{Z}[\tau]$ such that $N_\tau(\xi)$ is a prime representable as $5n+1$ and less than M, then $\pi(M)$ is in $\Theta(M/\log(M))$.*

The conjecture defines the frequency of easy instances of the norm equation during the sampling process. Finally we prove the main theorem.

Theorem 6.3.5. *Approximation algorithm (Figure 6.6) outputs a $\langle \mathcal{F}, \mathcal{T} \rangle$-circuit C of length $O(\log(1/\varepsilon))$ such that $d(C, R_z(\phi)) \leq \varepsilon$. On average the algorithm runtime is in $O(\log^c(1/\varepsilon))$ if Conjecture 6.3.4 is true.*

Proof. First we show that the algorithm achieves the desired precision and produces a $\langle \mathcal{F}, \mathcal{T} \rangle$-circuit of length $O(\log(1/\varepsilon))$. Both statements follow from Lemma 6.3.3. It is not difficult to check that the light gray segment on Figure 6.8 defines all u_0 such that $d(V[u_0\tau^m\zeta_{10}^k, v, 0], R_z(\phi))$ is less than ε. Therefore, by picking samples from the dark grey parallelogram P, we ensure that we achieve precision ε. The value of the Gauss complexity

measure is in $O(1/\varepsilon)$, therefore by Theorem 6.1.3 the length of the resulting circuit is in $O(1/\log(\varepsilon))$.

There are two necessary conditions for predicate EASY-SOLVABLE to be true:

(1) $N_\tau(\xi)$ is prime and equals $5n+1$ for integer n,

(2) $\xi > 0, \xi^\bullet > 0$.

Let p_M be the probability that the first condition is true when we choose ξ uniformly at random and $N_\tau(\xi)$ is bounded by M. Following Conjecture 6.3.4, we assume that p_M is in $O(1/\log(M))$. In our case M is of order φ^{2m} and therefore the probability of getting an instance solvable in polynomial time is in $O(1/\log(1/\varepsilon))$.

Now we show that the second condition is satisfied by construction. Part $\xi > 0$ is trivial because procedure RANDOM-SAMPLE always generates u_0 such that $|u_0| \leq \varphi^m$. For the second part we use Proposition 6.1.1 and note that for non-zero $u_0\tau^n$ the value of the Gauss complexity measure is

$$G(u_0\tau^n) = |(u_0\tau^n)^\bullet|^2 + |u_0\tau^n|^2 \geq 2.$$

We conclude that $|(u_0\tau^n)^\bullet|^2 \geq 1$ which gives

$$\xi^\bullet = \tau^{2m+1}(|(u_0\tau^n)^\bullet|^2 - 1) \geq 0$$

as required.

In summary, checking that an instance of ξ is easily solvable can be done in time polynomial in $\log(1/\varepsilon)$ using, for example, the Miller-Rabin Primality Test, the average number of loop iterations is in $O(\log(1/\varepsilon))$, an instance of the norm equation when ξ is prime can be solved in time that is on average is in $O(\log^d(1/\varepsilon))$ for some positive d. We conclude that on average the algorithm runs in time $O(\log^c(1/\varepsilon))$ for some positive constant c. □

The algorithm for approximating $R_z(\phi)X$ (Figure 6.10) can now be easily constructed based on ideas discussed above. First we simplify the expression for the distance

$$d(V[u,v,0], R_z(\phi)X) = \sqrt{1 - \sqrt{\tau}\,|\mathrm{Re}(ve^{-i(\phi/2+\pi/2)})|}$$

and notice that in this case it depends only on the bottom left entry of the unitary $V[u,v,0]$. Now u and v have opposite roles in comparison to the algorithm for approximating $R_z(\phi)$. Again, to get a better constant factor in front of $\log(1/\varepsilon)$ in the circuit size, we randomly

Input: ϕ — defines $R_z(\phi)X$, ε — precision
1: $r \leftarrow \sqrt{\varphi}, C \leftarrow \sqrt{\varphi/(4r)}$
2: $m \leftarrow \lceil \log_\tau(C\varepsilon r) \rceil + 1$
3: Find k such that $\theta = \phi/2 + \pi/2 - \pi k/5 \in [0, \pi/5]$
4: *not-found* $\leftarrow$ true, $u \leftarrow 0, v \leftarrow 0$
5: **while** *not-found* **do**
6: $u_0 \leftarrow$ RANDOM-SAMPLE(θ, ε, r) ▷ See Figure 6.7
7: $\xi \leftarrow \varphi^{2m} - \tau\,|u_0|^2$
8: **if** EASY-SOLVABLE(ξ) **then**
9: *not-found* $\leftarrow$ false
10: $v \leftarrow \zeta_{10}^k \tau^m u_0$
11: $u \leftarrow \tau^m$SOLVE-NORM-EQUATION(ξ)
12: **end if**
13: **end while**
14: $C \leftarrow$ EXACT-SYNTHESIZE$(V[u, v, 0])$
Output: Circuit C such that $d(C, R_z(\phi)X) \leq \varepsilon$

Figure 6.10: The algorithm for approximating $R_z(\phi)X$ by an $\langle \mathcal{F}, \mathcal{T} \rangle$-circuit with $O(\log(1/\varepsilon))$ gates and precision at most ε. The runtime is probabilistic polynomial as a function of $\log(1/\varepsilon)$.

pick v_0 such that $d(V[u, \varphi^m v_0, 0], R_z(\phi)X) \leq \varepsilon$. We use procedure RANDOM-SAMPLE to generate random v_0. When calling the procedure, we set the third input parameter r to $\sqrt{\varphi}$ to take into account that bottom left entries of exact-unitaries are rescaled by factor $\sqrt{\tau}$. Once we picked v_0 we check that there exist an exact unitary with bottom right entry $v = \tau^m v_0 \sqrt{\tau}$. In other words we solve norm equation

$$|u|^2 = \xi \text{ for } \xi = 1 - \tau|v|^2.$$

The necessary condition $\xi^\bullet \geq 0$ is always satisfied because $1 + \varphi|v^\bullet|^2$ is always positive. Once we find an "easy" instance of the norm equation we solve it and construct an exact unitary that gives the desired approximation. Our result regarding the approximation algorithm for $R_z(\phi)X$ is summarized by the following theorem.

Theorem 6.3.6. *Approximation algorithm (Figure 6.10) outputs a $\langle \mathcal{F}, \mathcal{T} \rangle$-circuit C of length $O(\log(1/\varepsilon))$ such that $d(C, R_z(\phi)X) \leq \varepsilon$. On average the algorithm runtime is in $O(\log^c(1/\varepsilon))$ if Conjecture 6.3.4 is true.*

The proof is completely analogous to the proof of Theorem 6.3.5 and we do not present

it here. We also discuss a method for approximating

$$R_x(\phi) := \begin{pmatrix} \cos(\phi/2) & -\sin(\phi/2) \\ \sin(\phi/2) & \cos(\phi/2) \end{pmatrix}$$

by using the same round off method for real numbers, but reducing a problem to solving a different norm equation. This is particularly useful for finding an approximation to the Hadamard gate. We first observe that the distance between $U[u, v, 0]$ and $R_x(\phi)$ can be simplified as:

$$\sqrt{1 - \left|\mathrm{Re}(u)\cos(\phi/2) + \mathrm{Re}(v)\sqrt{\tau}\sin(\phi/2)\right|}.$$

We find real parts of u and v by rounding off $\cos(\phi/2)$ and $\sin(\phi/2)$ using APPROX-REAL procedure. Imaginary parts of u and v must satisfy the following equation:

$$(2-\tau)(\mathrm{Im}(u)^2 + \tau\mathrm{Im}(v)^2) = 1 - \mathrm{Re}(u)^2 - \tau\mathrm{Re}(v)^2.$$

When the right-hand side is divisible by $2-\tau$, the problem reduces to the relative norm equation defined by polynomial $X^2+\tau$. To solve it we rely on PARI/GP. In this case we are not seeking a right-hand side that is easy instance, but one that is divisible by $2-\tau$. Our experiments show that this does not degrade the performance. We can also approximate, for example, $iR_x(\phi)$. Two examples of this rotation, $i\mathcal{F}$ and iX, are important building blocks for implementing multi-qubit gates [35].

6.4 Experimental results

In this section we evaluate the approximation quality of our algorithm as a function of $\langle\sigma_1, \sigma_2\rangle$-circuit size (depth) and the algorithm runtime. We not only confirm the results established in previous sections, but also show that constants hidden in the big-O notation are quite reasonable, making our algorithm useful in practice.

We experiment over several input sets of input unitaries and precisions, as summarized in Table 6.1. Each experiment is performed similarly. First, we request an approximation of a set of unitaries for certain precisions. In some experiments, we run the algorithm for the same unitary and precision several times to see the influence of the probabilistic nature of the algorithm on the result. Next, we aggregate collected data for a given precision by taking the mean, min or max of the parameter of interest over the set of all unitaries considered in the experiment. We compare to a Brute Force Search algorithm, from which we request an approximation of the set of unitaries that outputs the best precision that can

Name	Unitaries formula	k_{min}	k_{max}	Precisions formula	k_{min}	k_{max}	Runs per unitary
U-RZ-SMALL	$R_z(\frac{2\pi k}{10^3})$	1	$1 \cdot 10^3$	10^{-k}	2	14	1
U-RZ-BIG	$R_z(\frac{2\pi k}{4 \cdot 10^3})$	1	$4 \cdot 10^3$	10^{-k}	2	30	1
RZ-HIGH	$R_z(\frac{\pi}{2^k})$	2	$6 \cdot 10^1$	10^{-k}	2	100	10
U-RZ-LOW	$R_z(\frac{2\pi k}{10^3})$	1	$1 \cdot 10^3$	$10^{-k/8}$	8	12	1
U-RZ	$R_z(\frac{\pi k}{10^4})$	1	$1 \cdot 10^4$	—	—	—	—
U-RZX-BIG	$R_z(\frac{2\pi k}{4 \cdot 10^3})X$	1	$4 \cdot 10^3$	10^{-k}	2	30	1
U-RZX-LOW	$R_z(\frac{2\pi k}{10^3})X$	1	$1 \cdot 10^3$	$10^{-k/8}$	8	12	1
U-RZX	$R_z(\frac{\pi k}{10^4})X$	1	$1 \cdot 10^4$	—	—	—	—
X-HIGH	X	—	—	10^{-k}	2	100	500

Table 6.1: Sets of inputs used for the experiments.

be achieved using at most N σ gates for each input angle. The largest N for our database is 25. In this case we aggregate collected data for a given N.

We implemented our algorithm using C++. There are two third-party libraries used: PARI/GP[16] which provides a relative norm equation solver and primality test, and *boost::multiprecision* which includes high-precision integer and floating-point types. All experimental results described in this section were obtained on a computer with Intel Core i7-2600 (3.40GHz) processor and 8 GB of RAM. Our implementation does not use any parallelism.

6.4.1 Quality Evaluation

We evaluate the approximation quality of our algorithm on four large sets of inputs. Two of them are used to evaluate the approximation quality of $R_z(\phi)$ rotations and the other two for $R_z(\phi)X$. For both rotation types, one set covers uniformly the range of angles $[0, 2\pi]$ (U-RZ-BIG, U-RZX-BIG) and the other one includes rotations that are particularly important in applications. For $R_z(\phi)$ rotations, we study angles ϕ of the form $\frac{\pi}{2^n}$ used in the Quantum Fourier Transform (input set RZ-HIGH); for $R_z(\phi)X$, we look at the

quality of the Pauli X gate approximation (input set X-HIGH). The results are presented in Figure 6.11. In addition to the average number of gates, we show minimal and maximal number of gates needed to achieve the required precision, which demonstrates the stability of the quality of our algorithm.

One of the baselines we compare the quality of our algorithm to is Brute Force Search. We built a database of optimal $\langle \sigma_1, \sigma_2 \rangle$-circuits with up to 25 gates and used it to find optimal approximations of unitaries from datasets U-RZ and U-RZX. The highest average precision that we were able to achieve is around $10^{-2.5}$. We evaluated our number theoretic algorithm on the same range of precisions; the results are presented in Figure 6.12. For our algorithm, the average coefficient in front of $\log_{10}(1/\varepsilon)$ is only 18% larger than the average for the optimal approximations of $R_z(\phi)$ and 40% larger for $R_z(\phi)X$.

Figure 6.13b shows the exponential scaling of the number of optimal circuits with a given number of $\langle \sigma_1, \sigma_2 \rangle$ gates and confirms that the Brute Force Search becomes infeasible exponentially quickly. In summary, our algorithm finds circuits for unitaries $R_z(\phi)$ and $R_z(\phi)X$ exponentially faster than Brute Force Search and with very moderate overhead.

The previous state-of-the-art method for solving the unitary approximation problem for the Fibonacci braid basis in polynomial time is the Solovay-Kitaev algorithm. The Solovay-Kitaev algorithm can be applied to any gate set and does not take into account the number theoretic structure of the approximation problem. Here we provide a rough estimate of its performance when approximating using $\langle \sigma_1, \sigma_2 \rangle$-circuits.

We consider approximation using special unitaries in this case. The version of the Solovay-Kitaev algorithm described in [18] boosts the quality of the approximation provided by a fixed-size ε-net. It is crucial for the overall estimate to find the quality provided by an epsilon net depending on the maximal size of the optimal circuit in it. We use the scaling of the size of our database (Figure 6.13b) of optimal circuits and a rough volume argument to get the estimate.

Consider the problem of approximating states, which is the same as approximating single-qubit special unitaries. Our ε net should cover the Bloch sphere with overall surface area 4π. Each state will cover roughly an area of the sphere equal to $\pi\varepsilon^2$. Therefore, for n being the size of the longest circuit:

$$\pi\varepsilon^2 10^{(0.275x+0.592)}/2 \simeq 4\pi.$$

We divide the estimate for the number of unitaries by two to get the estimate for the number of states. This is because there are only up to global phase two distinct exact unitaries of the form $U[x, y, k]$ for given x, y (for k=0 and k=1). Other values of k can be

reduced to 0 or 1 using the identity $\zeta_8^s U[x, y, k] = U[x\zeta_8^s, y\zeta_8^s, k + 2s]$. We also assume that $U[x, y, k]$ and $U[x\zeta_8^s, y\zeta_8^s, k]$ have very similar cost.

The database we are using in other experiments includes circuits with up to 25 gates and requires around 5GB of RAM to be built. In our estimate for the performance of the Solovay-Kitaev algorithm we assume that with enough engineering effort one can build a database with up to 30 gates. Our estimates result in the following:

$$\log_{10}(1/\varepsilon_n) \simeq 0.137n - 0.155, \log_{10}(1/\varepsilon_{30}) \simeq 3.97$$
$$n(\log_{10}(1/\varepsilon_n)) \simeq 7.27\varepsilon + 1.127.$$

The estimate is more optimistic in comparison to results of our Brute Force Search as now we consider the full special unitary group instead of its subsets $R_z(\phi)$ and $R_z(\phi)X$. With these numbers in hand, we use the analysis of the Solovay-Kitaev algorithm in [18].

Figure 6.13 compares our estimates for the size of circuits produced by the Solovay-Kitaev algorithm and the sizes from running our algorithm. In particular, for precision 10^{-10} our algorithm produces twenty times smaller circuits and for precision 10^{-30}, one thousand times smaller circuits, which corresponds to the difference between the algorithms with respective scalings in $O(\log^{3.97}(1/\varepsilon))$ and $O(\log(1/\varepsilon))$.

6.4.2 Performance evaluation

Our experiments confirm that the algorithm described in the paper has a probabilistic polynomial runtime. In addition, constants and the power of the polynomial are such that our algorithm is practically useful. In Figure 6.12c we show the runtime of different parts of our algorithm. The approximation part corresponds to the runtime of the algorithm approximating unitaries $R_z(\phi)$ by exact unitaries (Figure 6.6), excluding time needed to solve the norm equation; the synthesis part corresponds to the runtime of the exact synthesis algorithm that produces an $\langle \mathcal{F}, \mathcal{T} \rangle$-circuit; the resynthesis part corresponds to the runtime of peephole optimization performed on $\langle \sigma_1, \sigma_2 \rangle$-circuits obtained from $\langle \mathcal{F}, \mathcal{T} \rangle$-circuits using identities from Section 6.1.

We separately show the runtime of the relative norm equation solver because for our implementation we used a generic solver which is a part of the PARI/GP library (function *rnfisnorm* [16]). Since the library documentation does not describe the function performance in detail, we performed an evaluation to confirm that it is polynomial time on "easy" instances of the problem. We also rely on another PARI/GP function *ispseudoprime* to

perform a primality test. It is a combination of several probabilistic polynomial time primality tests [16]. The runtime of the primality test is included in the approximation part of the figure.

Our claim about the approximation algorithm runtime depends on Conjecture 6.3.4; Figure 6.12d shows the number of trials performed in the main loop of the approximation algorithm before finding an easy instance. The scaling of the average number of trials shown in the figure supports the conjecture. To perform peephole optimization [63] we used the database of optimal $\langle \sigma_1, \sigma_2 \rangle$-circuits with up to 19 gates which has size 85.7 MB.

High-precision integer and floating-point data types are necessary to implement our algorithm. We use *cpp_int* and *cpp_dec_float* from *boost::multiprecision* library. The number of bits used by these types can be specified at compile time. This allows us to avoid dynamic memory allocation when performing arithmetic operations; much faster stack memory is used instead. For this reason, runtime scaling (Figure 6.12c) of our code is a function of the number of arithmetic operations, and not of the bit size of numbers used in the algorithm.

In Figure 6.13a we show how the runtime of our algorithm changes when using different arithmetic types on the same set of inputs. The first pair of types — 512 bit integers and 200 decimal digits floating-point numbers — is sufficient for precision up to 10^{-35}, the second pair — 1024 bits and 400 decimal digits — for precision up to 10^{-70}. Figure 6.12c shows runtime scaling for different parts of our algorithm in more detail when using the second pair of types. This shows that our algorithm is practical and can readily be used as a subroutine when compiling quantum algorithms with large numbers of different single-qubit operations.

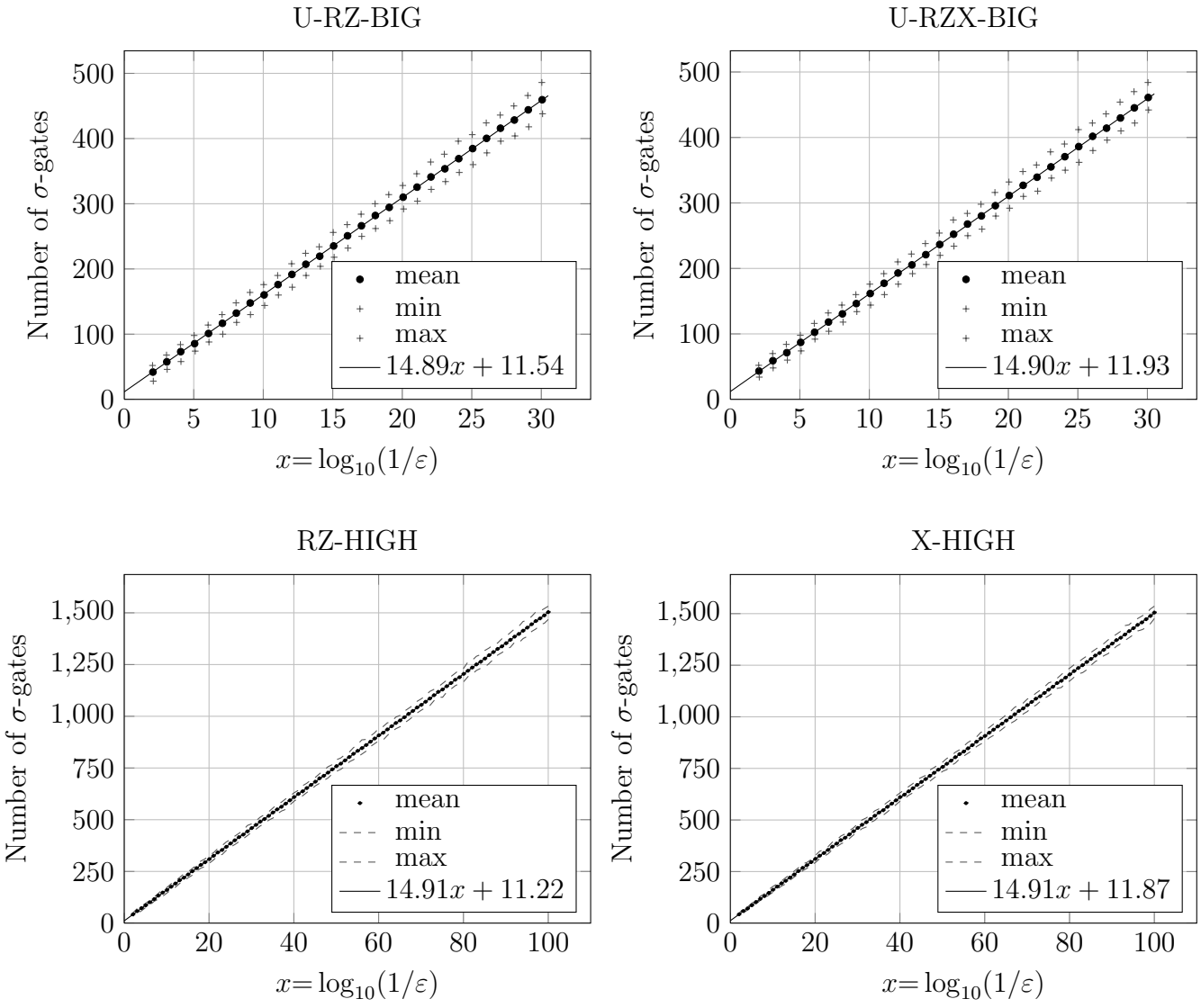

Figure 6.11: Number of σ gates needed to achieve the quality of approximation ε using the Number Theoretic Algorithm on sets of inputs U-RZ-BIG, U-RZX-BIG, RZ-HIGH, X-HIGH (see Table 6.1). Includes approximation of X gate and R_z rotations used in the Quantum Fourier Transform.

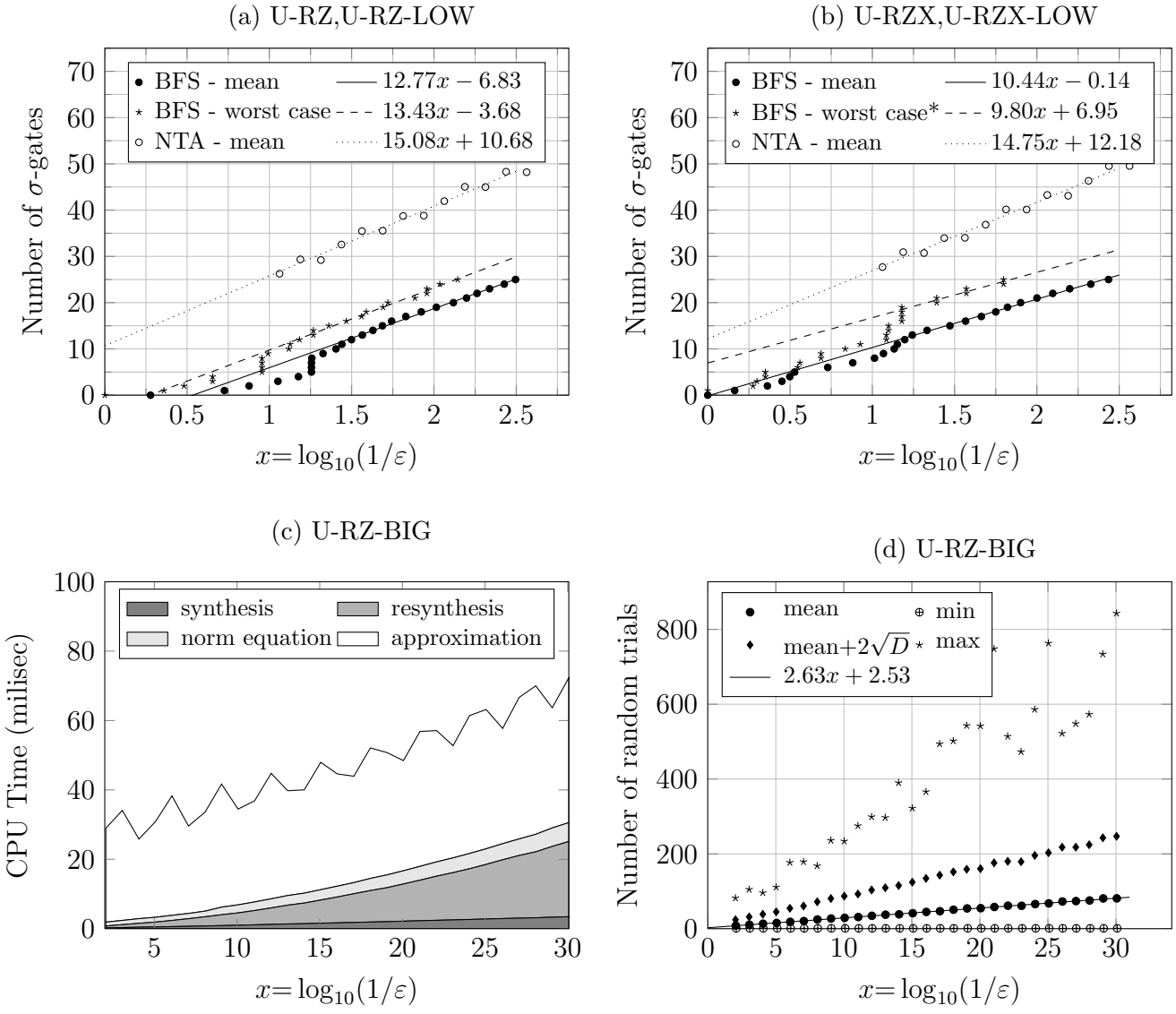

Figure 6.12: (a),(b) comparison of the number of σ gates needed to achieve the quality of approximation ε using the Number Theoretic Algorithm (NTA) and using Brute Force Search (BFS). Input sets U-RZ-LOW and U-RZX-LOW were used for the NTA and U-RZ, U-RZX for BFS; (c) average runtime of parts of the NTA, using U-RZ-BIG input set; (d) number of trials performed in the main loop of the NTA before an "easy" instance was found, using U-RZ-BIG input set. See Table 6.1 for input set descriptions.

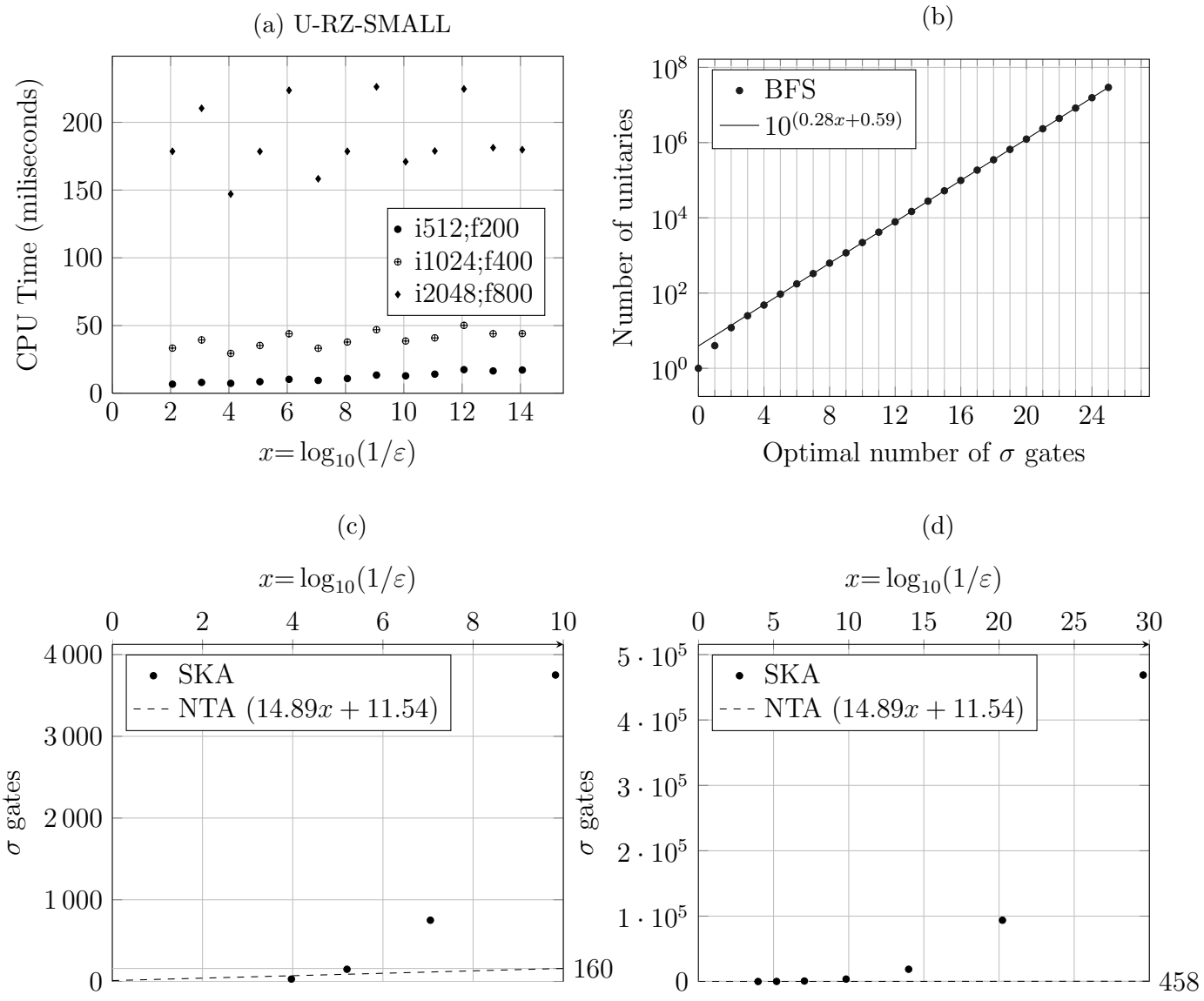

Figure 6.13: (a) runtime of the Number Theoretic Algorithm when using different arithmetic data types. Type i**N** corresponds to a signed, unchecked boost::multiprecision::cpp_int with MinDigits and MaxDigits set to **N**; type f**N** corresponds to boost::multiprecision::cpp_dec_float with Digits10 set to **N**; (b) number of distinct (up to a global phase) unitaries, such that their optimal implementation requires given number of σ gates. (c),(d) comparison of the estimated size of circuits produced by the Solovay-Kitaev algorithm (SKA) and the Number Theoretic Algorithm (NTA).

Chapter 7

Multiple qubit exact and approximate synthesis for the Clifford+T gate set

In this chapter we apply the number theoretic method to the approximation of multiple qubit unitaries using the Clifford+T gate set. B. Giles and P. Selinger[26] showed that any n qubit unitary over the ring $\mathbb{Z}\left[i, 1/\sqrt{2}\right]$ with entries of the form $((a+b\sqrt{2})+i(c+d\sqrt{2}))/\sqrt{2}^{\kappa}$ can be synthesized exactly using at most one ancilla. However, their synthesis method requires $O\left(3^{2^n} n\kappa\right)$ gates, which is far from information theoretic bounds. The doubly exponential scaling of the circuits size with the number of qubits destroys the potential advantages of using the number theoretic method for multiple qubit unitaries. In this chapter we propose an exact synthesis method that uses two ancillae and requires $O\left(4^n n\kappa\right)$ gates.

We also apply the proposed exact synthesis method to the approximate synthesis of unitaries acting on multiple qubits and show that any n qubit unitary can be approximated with precision ε using at most $O\left(4^n n\left(\log\left(1/\varepsilon\right)+n\right)\right)$ Clifford and T gates and two ancillae. Our procedure results in a slightly larger asymptotic number of gates in comparison to first using an asymptotically optimal decompositions of an n qubit unitary into $O\left(4^n\right)$ single qubit unitaries and CNOT gates[57, 2] and then approximating each single qubit unitary with Clifford and T circuit. The second approach requires $O\left(4^n n\left(\log\left(1/\varepsilon\right)+n\right)\right)$ gates to achieve precision ε. Both approaches are asymptotically optimal when the number of qubits is fixed, according to the lower bounds established in [50, 32]. However, our decomposition is more directly related to the structure of the initial unitary and may be

beneficial in two cases: when the unitary is partially specified, or when most of the entries of the unitary are over the ring $\mathbb{Z}\left[i, 1/\sqrt{2}\right]$. In contrast to decompositions used in [2, 57, 60] our decomposition does not require taking square roots. We use only addition, subtraction and multiplication by $1/\sqrt{2}$; these operations preserve elements of $\mathbb{Z}\left[i, 1/\sqrt{2}\right]$.

The main results of the chapter are summarized in the following theorems:

Theorem 7.0.1. *Any n qubit unitary can be approximated within Frobenius distance ε using $O\left(4^n n\left(C\log\left(1/\varepsilon\right)+n\right)\right)$ Clifford and* T *gates and at most two ancillae.*

Theorem 7.0.2. *Any n qubit unitary with entries of the form $((a+b\sqrt{2})+i(c+d\sqrt{2}))/\sqrt{2}^{\kappa}$ can be exactly implemented using $O\left(4^n n\kappa\right)$ Clifford and* T *gates using at most two ancillae.*

To prove both results we use a variant of the Householder decomposition which expresses a matrix as a product of reflection operators and a diagonal unitary matrix. In our case the diagonal unitary matrix is always the identity. We define a reflection operator constructed from a unit vector $|\psi\rangle$ as $R_{|\psi\rangle} = \mathbb{I} - 2\left|\psi\right\rangle\left\langle\psi\right|$. Our structure-preserving decomposition is given by the following lemma:

Lemma 7.0.3. *Let U be a unitary acting on n qubits and let $\{u_1, \ldots, u_{2^n}\}$ be the columns of U. The unitary U can be simulated using the unitary*

$$U' = |0\rangle\langle 1| \otimes U + |1\rangle\langle 0| \otimes U^{\dagger}.$$

Unitary U' is a product of reflection operators constructed from the family of unit vectors

$$\left|w_j^{\pm}\right\rangle = \left(|1\rangle \otimes |j\rangle \pm |0\rangle \otimes |u_j\rangle\right)/\sqrt{2}, \text{ for } j = 1, \ldots, 2^n.$$

Proof. By direct calculation we check that U' maps $|1\rangle \otimes |\phi\rangle$ to $|0\rangle \otimes U|\phi\rangle$ for any n qubit state $|\phi\rangle$. Next we observe that $\left|w_j^{\pm}\right\rangle$ are eigenvectors of U' with eigenvalues ± 1. Defining $P_j^{\pm}$ to be projectors on subspaces spanned by $\left|w_j^{\pm}\right\rangle$ and using the spectral theorem we express U' as $\sum_{j=1}^{2^n}\left(P_j^{+} - P_j^{-}\right)$. Since $\sum_{j=1}^{2^n}\left(P_j^{+} + P_j^{-}\right)$ is the identity operator and all projectors $P_j^{\pm}$ are orthogonal we can write:

$$U' = I - 2\sum_{j=1}^{2^n} P_j^{-} = \prod_{j=1}^{2^n}\left(I - 2P_j^{-}\right).$$

The right hand side is a product of 2^n reflection operators, as required. □

It is not difficult to see that if $|u_j\rangle$ has coordinates in the computational basis over the ring $\mathbb{Z}\left[i, 1/\sqrt{2}\right]$ then the unit vector $\left|w_j^{\pm}\right\rangle$ also does. This is the reason we call our decomposition structure-preserving. Exactly synthesizing a reflection operator is not more difficult than exactly preparing corresponding unit vector:

Lemma 7.0.4. *Any reflection operator $R_{|\phi\rangle}$ where $|\phi\rangle$ has coordinates in the computational basis of the form $((a+b\sqrt{2})+i(c+d\sqrt{2}))/\sqrt{2}^{\kappa}$ can be implemented using $O\left(2^n n\kappa\right)$ Clifford and* T *gates and at most one ancillae.*

Proof of lemma 7.0.4. Note that $UR_{|0\rangle}U^{\dagger} = R_{U|0\rangle}$. Therefore, to implement the reflection operator with corresponding unit vector $|\phi\rangle$ it is enough to find a U that prepares $|\phi\rangle$ starting from $|0\rangle$. The column lemma [26] provides a construction for U requiring $O\left(2^n n\kappa\right)$ Clifford and T gates and one ancilla. Unitary $R_{|0\rangle}$ is a multiple controlled Z operator and can be implemented with $O\left(n\right)$ gates and one ancilla, for example using the construction from [60]. We conclude that we need $O\left(2^n n\kappa\right)$ Clifford and T gates in total to implement $R_{|\phi\rangle}$. □

Now we have all results required to prove Theorem 7.0.2:

Proof of theorem 7.0.2. The construction from lemma 7.0.3 allows one to simulate an n qubit unitary using one ancilla and 2^n reflection operators. The unit vectors corresponding to each reflection operator have coordinates of the form $((a+b\sqrt{2})+i(c+d\sqrt{2}))/\sqrt{2}^{\kappa+1}$ in the computational basis. According to lemma 7.0.4 these reflection operators can be implemented using one ancilla and $O\left(2^n n\kappa\right)$ Clifford and T gates. Therefore we need $O\left(4^n n\kappa\right)$ Clifford and T gates and at most two ancillae to implement the unitary exactly. □

To show the approximation result we use the decomposition above and then approximate each reflection operator separately. First we note the following relation between approximating reflection operators and their corresponding unit vectors:

Proposition 7.0.5. *The distance induced by the Frobenius norm between two reflection operators is bounded by the Euclidean distance between corresponding unit vectors as:*

$$\left\|R_{|\psi\rangle} - R_{|\phi\rangle}\right\|_{Fr} \leq 2\sqrt{2}\left\||\psi\rangle - |\phi\rangle\right\|.$$

To prove the proposition it is enough to use the definition of the Frobenius norm $\|A\|_{Fr}^2 = \mathrm{Tr}\left(AA^{\dagger}\right)$, express $\left\|R_{|\psi\rangle} - R_{|\phi\rangle}\right\|_{Fr}^2$ in terms of $|\langle\phi|\psi\rangle|^2$, use that $\mathrm{Re}\,\langle\phi|\psi\rangle \leq$

$|\langle\phi|\psi\rangle|$ and express $\mathrm{Re}\,\langle\phi|\psi\rangle$ in terms of $\||\psi\rangle - |\phi\rangle\|^2$. Next we show how to approximate arbitrary reflection operator by at most two reflection operators with corresponding unit vectors over the ring $\mathbb{Z}\left[i, 1/\sqrt{2}\right]$.

Lemma 7.0.6. *Any n qubit reflection operator $R_{|\psi\rangle}$ can be approximated within Frobenius distance ε by the product of two reflection operators, such that coordinates in the computational basis of corresponding unit vectors have the form $(a + b\sqrt{2} + ci + di\sqrt{2})/2^{m(\varepsilon)}$ where $m(\varepsilon) = \lceil n/2 \rceil + O(\log(1/\varepsilon))$ and using at most one ancilla. If unit vector $|\psi\rangle$ has at least two coordinates in the computational basis equal to zero it is sufficient to use one reflection operator and no ancilla is required.*

Proof of lemma 7.0.6. First we construct the approximating unitary in a special case where no ancilla are required. Consider the reflection operator $R_{|\psi\rangle}$. Let $\{\alpha_k\}$ be the coordinates of $|\psi\rangle$ in computational basis. First we consider the case when $|\psi\rangle$ has at least two zero entries, say α_j and α_l. We use an idea similar to [47] and define the approximating unitary as a reflection operator $R_{|\phi\rangle}$ corresponding to the vector

$$\begin{aligned} |\phi\rangle &= \frac{a+bi}{2^m}|j\rangle + \frac{c+di}{2^m}|l\rangle + \sum_{k=1,\,k\neq j,l}^{2^n} \beta_k |k\rangle, \\ \beta_k &= \frac{\lfloor 2^m \mathrm{Re}\alpha_k \rfloor + i\lfloor 2^m \mathrm{Im}\alpha_k \rfloor}{2^m}, a, b, c, d \in \mathbb{Z}. \end{aligned}$$

The norm of $|\phi\rangle$ must be equal to 1, therefore:

$$a^2 + b^2 + c^2 + d^2 = 4^m \left(1 - \sum_{k=1,\,k\neq j,l}^{2^n} \left|\beta_k^2\right|\right). \tag{7.1}$$

The Diophantine equation above always has a solution according to Lagrange's four-square theorem and it can be efficiently found using a probabilistic algorithm[66] that has runtime polynomial in number of bits required to write the right hand side of equation (7.1).

By Proposition 7.0.5, to estimate the distance between the reflection operator and its approximation it is enough to estimate the square of the distance between $|\psi\rangle$ and $|\phi\rangle$. We approximated each non-zero entry with precision $2^{-m}\sqrt{2}$, therefore

$$\||\psi\rangle - |\phi\rangle\|^2 \leq 2^n \left(2^{-m}\sqrt{2}\right)^2 + 4^{-m}\left(a^2 + b^2 + c^2 + d^2\right).$$

The second summand of the right hand side of the inequality above can be estimated as:

$$\begin{aligned}
1 - \sum_{k=1,\, k\neq j,l}^{2^n} \left|\beta_k^2\right| &= \sum_{k=1,\, k\neq j,l}^{2^n} \left(\left|\alpha_k^2\right| - \left|\beta_k^2\right|\right) \\
&\leq \sum_{k=1,\, k\neq j,l}^{2^n} \left||\alpha_k| - |\beta_k|\right| \left(|\alpha_k| + |\beta_k|\right) \\
&\leq 2^{-m}\sqrt{2} \sum_{k=1,\, k\neq j,l}^{2^n} \left(|\alpha_k| + |\beta_k|\right) \\
&\leq 2^{-m}\sqrt{2}\left(2 \cdot 2^{n/2}\right).
\end{aligned}$$

We used the Cauchy–Schwarz inequality to estimate sums involving $|\alpha_k|$ and $|\beta_k|$. For example:

$$\sum_{k=1,\, k\neq j,l}^{2^n} |\alpha_k| \leq \sqrt{2^n} \sqrt{\sum_{k=1,\, k\neq j,l}^{2^n} |\alpha_k|^2} = 2^{n/2}.$$

In summary we get:

$$\left\||\psi\rangle - |\phi\rangle\right\|^2 \leq 2 \cdot 2^{n-2m} + 2\sqrt{2} \cdot 2^{(n/2-m)}.$$

By choosing $m = \lceil n/2 + \log_2\left(1/\varepsilon^2\right)\rceil + 5$ and using Proposition 7.0.5 we get $\left\|R_{|\psi\rangle} - R_{|\phi\rangle}\right\|_{Fr} \leq \varepsilon$ when $\varepsilon \leq 1$.

In the case when all entries of $|\psi\rangle$ in the computational basis are non-zero, we add an ancilla and express the reflection around $|\psi\rangle$ using two reflection operators with unit vectors that can be approximated without using ancilla:

$$\begin{aligned}
\mathbb{I}_1 \otimes R_{|\psi\rangle} &= \mathbb{I}_1 \otimes \mathbb{I}_n - 2\mathbb{I}_1 \otimes |\psi\rangle\langle\psi| \\
&= \mathbb{I}_1 \otimes \mathbb{I}_n - 2\left(|0\rangle\langle 0| + |1\rangle\langle 1|\right) \otimes |\psi\rangle\langle\psi| \\
&= R_{|0\psi\rangle} R_{|1\psi\rangle}.
\end{aligned}$$

□

Now we have all tools needed to prove Theorem 7.0.1:

Proof of theorem 7.0.1. We use the construction from lemma 7.0.3 to simulate an n qubit unitary using one ancilla and 2^n reflection operators. The unit vectors corresponding to

each reflection operator have at least two zero coordinates in the computational basis, therefore we can use lemma 7.0.6 to approximate each reflection with a reflection operator without using ancillae with precision $2^{-n}\varepsilon$. Unit vectors of each approximating reflection have entries of the form $(a+b\sqrt{2}+ci+di\sqrt{2})/2^{m(n,\varepsilon)}$ for $m(n,\varepsilon) = \lceil 5n/2 \rceil + O(\log(1/\varepsilon))$. Each reflection operator requires one ancillae and $O(2^n n \cdot m(n,\varepsilon))$ Clifford and T gates according to lemma 7.0.4. Therefore, in total we need two ancillae and $O(4^n n(\log(1/\varepsilon) + n))$ Clifford and T gates to approximate the unitary within Frobenius distance ε. □

Our improved method for multiple qubit exact synthesis outputs circuits with an exponential number gates as a function of the number of qubits. This improves the previously known method[26] which requires doubly exponential number of gates. Further improvements over our result may be possible: for example, removing the factor of n from the expression $O(4^n n(\log(1/\varepsilon) + n))$. Furthermore, our improved multiple qubit exact synthesis algorithm leads to an alternative algorithm for multiple qubit approximate synthesis, which may also lead to practical advantages in some cases.

Chapter 8

Clifford circuits optimization

In this chapter we describe the algorithm for finding optimal circuits for Clifford unitaries acting on a small number of qubits. As we discussed in Chapter 1 it is useful for implementing the randomized benchmarking protocols. We also use the optimal implementations of Clifford operations acting on a small number of qubits in peephole optimization [63] of larger Clifford circuits. Peephole optimization is a heuristic approach to circuit optimization that relies on the access to the database of optimal circuits for a set of unitaries acting on the small number of qubits. The larger the database available the better peephole optimization will perform. Therefore it is important to be able to build as large a database of optimal circuits as possible given limited memory and computational power available. We address this practical issue in several ways. First, we represent Clifford unitaries using a small number of bits. Second, we compress the search space by using equivalence classes of unitaries. Finally, we store only the necessary information about equivalence classes needed to recover the optimal circuit for any unitary in the database. We find optimal Clifford circuits for up to four qubits, optimal Clifford circuits up to input/output permutation for up to five qubits, and optimize larger Clifford circuits by a factor of roughly two.

The two optimality measures that we consider are the minimal number of Clifford gates required and the minimal depth of the circuit implementing the given unitary. For brevity, we call them the *gate count* and the *depth* of the unitary. Our ideas extend to other optimality measures, such as the weighted gate counts/depth.

In addition, we apply the ideas developed in this chapter to find an optimal encoding circuit for the $[[5, 1, 3]]$ five-qubit error correcting code. This method can be applied to synthesize encoding circuits for other error correcting codes that use a small number of qubits.

8.1 Clifford unitaries representation

Compact representation of any unitary that can be computed by a Clifford circuit is a direct consequence of the fact that it maps elements of the n qubit Pauli group to themselves by conjugation. Taking into account the identity $\mathrm{Y} = i\mathrm{XZ}$, it suffices to know the action by conjugation of the n-qubit unitary on $2n$ Pauli matrices. The result of the application of the circuit to each Pauli matrix can be encoded using $2n + 1$ bits [1]. Each single-qubit Pauli matrix can be encoded in two bits, as follows:

$$\mathrm{I} \sim (0|0)\,, \mathrm{X} \sim (1|0)\,,\ \mathrm{Z} \sim (0|1)\ \ \mathrm{Y} \sim (1|1)\,.$$

It is convenient to separate X and Z parts when encoding larger circuits:

$$\mathrm{I} \sim (\mathbf{0}|\mathbf{0})\,, \mathrm{X} \sim (1|0)\,, -\mathrm{I} \otimes \mathrm{X} \sim (\mathbf{0}1|\mathbf{0}0|1)\,.$$

One additional bit shown at the end is used to encode the overall overall phase, here restricted to ± 1. For any unitary the sign can be adjusted by applying the round of Pauli gates at the end of the computation. In most of our applications this can be done for free. As a result, we will consider only the $2n \times 2n$ part of the encoding matrix. Commutativity relations between Pauli matrices are preserved under conjugation and induce additional constraint on the encoding matrix—it must be symplectic (a square block matrix $M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$ is called symplectic iff the following three conditions hold: $A^T C = C^T A$, $B^T D = D^T B$, and $A^T D - C^T B = I$). Furthermore, the canonical decomposition theorem [1] shows that any binary symplectic matrix encodes some Clifford circuit.

The above matrix representation can be efficiently updated [1] when adding new gates to the end of an existing circuit. Computationally, adding a gate requires updating one or two columns of the encoding matrix. In particular, the application of the Phase gate to qubit k corresponds to the addition modulo 2 of column k to column $n+k$, the Hadamard gate on qubit k corresponds to exchanging columns k and $n + k$, and the CNOT gate with control k and target j corresponds to the addition of column k to column j and the addition of column $n + j$ to column $n + k$. An empty Clifford circuit corresponds to the identity matrix. These rules suffice to determine the $2n \times 2n$ binary symplectic matrix encoding the unitary computed by a given Clifford circuit.

For linear reversible circuits (those composed only with CNOT gates) it suffices to store only the top left $n \times n$ part of the binary symplectic matrix. The described procedure

for updating columns immediately implies that the binary symplectic matrix for linear reversible circuits should be of the following form:

$$\left(\begin{array}{c|c} A & 0 \\ \hline 0 & B \end{array}\right).$$

As the matrix must be symplectic, we have $A^T B = I$ (per third equation in the definition), which uniquely determines B given A. Therefore, we can store linear reversible unitaries more efficiently than a generic Clifford operation.

8.2 Algorithms

The main challenge in our approach to finding optimal circuits is coping with the size of the search space, that grows very rapidly, as illustrated in Table 8.1. The size of the database for Clifford unitaries acting on n qubits reported in this table is calculated using the following formula: $4n^2 \times N_G/n!$, where $4n^2$ corresponds to the storage space (in bits) required by a single unitary, N_G is the number of elements in the respective Clifford group, and the division by $n!$ corresponds to the estimated reduction due to the input/output relabelling invariance ($n!$ marks a maximal possible reduction, therefore the entire figure provides a lower bound on the size of the database).

Our algorithm is based on Breadth First Search. The number of distinct unitaries computed by Clifford circuits grows as $2^{\Theta(n^2)}$. We address the resulting challenge in several ways. First, each node of the search tree corresponds to an equivalence class of unitaries instead of the unitary itself. Second, we use the meet in the middle technique to avoid building the full tree [27]. Finally, we use a special data structure to store the search tree in a compact way. It is described in more details in Section 8.2.

The equivalence relation we use to reduce the size of the search space is the following: two unitaries are equivalent if they can be computed by circuits that are the same up to simultaneous relabelling of their inputs and outputs. Both gate count and depth of a unitary are invariant with respect to such simultaneous relabelling. During the search we store only a canonical representative of each class. For n inputs this results in a reduction of the number of unitaries to be stored by a factor of approximately $n!$. The number of unitaries corresponding to the same canonical representative is not always $n!$, but this is the most common case. In particular, the fraction of four-qubit unitaries that have less than 24 ($= 4!$) elements in their equivalence class is less than 9.7×10^{-5}. To search for five-qubit optimal Clifford circuits we used the equivalence relation corresponding to the

G	n	N_G	$Size_{Gr}(GB)$
Sp	3	1,451,520	1.01×10^{-3}
	4	47,377,612,800	14.71
	5	24,815,256,521,932,800	2.41×10^{6}
Gl	6	20,158,709,760	0.12
	7	163,849,992,929,280	185.44

Table 8.1: G — group: Sp — symplectic part of Clifford group, Gl — group generated by linear reversible circuits; n — number of qubits, N_G — size of the corresponding group, $Size_{Gr}$ — lower bound on the size of the database taking into account input/output relabelling (GB).

independent relabelling of the inputs and outputs, in other words, we ignored SWAP gates. This further shrinks the search space, but the results are suboptimal in the scenario when SWAP has a non-zero cost.

The idea of the meet in the middle (MiM) technique is based on the optimality of subcircuits of any optimal circuit. Given a database DB_c of all unitaries with the cost at most c, MiM allows us to find optimal circuits for unitaries with the cost at most $2c$. Suppose we are looking for an optimal circuit computing a unitary f with cost $c+d \leq 2c$. We can always split the optimal circuit into two optimal circuits with d and c gates. Therefore, there always exists a unitary g with cost $d \leq c$ such that its composition with f has cost c, and it is in our database. We can find g by trying all unitaries from the database and checking if $g \circ f$ is also in the database. In the worst case, using meet in the middle increases the time required to find a circuit by a factor proportional to the size of DB_c, in comparison to using the database DB_{2c}. At the same time, meet in the middle significantly reduces the required memory. For example, in the case of four qubits the maximal number of gates required is 17 and the size of the database is 14.72 GB. Using the database with optimal circuits up to 9 gates reduces the required memory to just 108 MB. Meet in the middle is vital for the search of optimal five-qubit Clifford circuits up to input/output permutation. In this case, the size of the full database would have been about 2.41×10^6 GB.

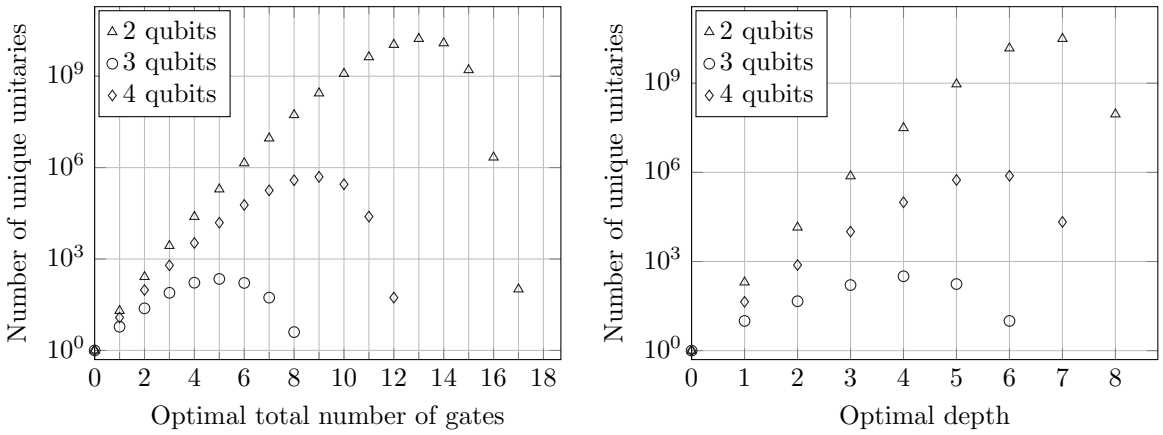

Figure 8.1: The number of unique Clifford unitaries on 2, 3, and 4 qubits per optimal gate count and depth.

Computing canonical representative

To find the canonical representative with respect to the *simultaneous* relabelling of the inputs and outputs we compute all elements of the equivalence class, encode them as bit strings, and find the minimum. We need to go though all possible permutations. This is accomplished by applying a single transposition at each step. Exchanging inputs k and j of an n-qubit Clifford circuit corresponds to swapping columns and rows of the binary symplectic matrix. The pair of columns $(k, k+n)$ must be swapped with $(j, j+n)$, the pairs of rows with the same indexes must be swapped also. Internally we represent each binary matrix as an array of integers. Each integer corresponds to a column of the binary symplectic matrix. We precompute the required transpositions of the bit strings of length $2n$ and use a lookup table to speed up the swapping of rows of the binary symplectic matrix.

When we allow the *independent* relabelling of the inputs and outputs we apply a more efficient procedure for canonical representative computation. In most cases we have $(n!)^2$ representatives corresponding to the same equivalence class. First we find all $n!$ representatives corresponding to the different row permutations [1]. Then we store columns k and $k+n$ together in one bit string and sort the resulting bit strings using a sorting network. This gives a canonical representative with respect to the column permutation for a fixed

[1]By row permutation we mean a permutation acting simultaneously on first n rows and rows $n+1, \ldots, 2n$.

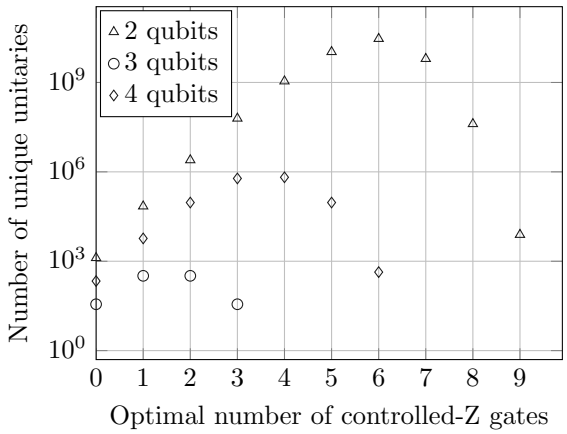

Figure 8.2: The number of unique Clifford unitaries on 2, 3, and 4 qubits per optimal number of controlled-Z gates.

row permutation. Finally, we encode the representative for each row permutation as a bit string and find the minimum.

We apply the same idea for linear reversible circuits. To exchange two inputs k and j we just need to swap columns k and j and rows k and j of the matrix encoding the circuit. This approach also extends to partially specified matrices.

Implementation details

The main bottleneck in our search is the amount of memory available. In addition to using the canonical representation, we tried to minimize the memory overhead caused by the data structures. Here we describe the details of the gate count optimal search. The same ideas were adopted for depth optimal search and can be extended to more general cost functions. We did not set out to study all possible optimizations in a systematic way. We present a set of solutions that allowed us to obtain the results in a reasonable amount of time and designed our software to be scalable enough to support different types of search.

Possible costs of unitaries belong to a short range of integer values. Once we found all unitaries with some fixed cost we store them as a sorted array. We call it a *layer*. It allows us to quickly lookup unitaries with a given cost, however, it is expensive to ensure consistency of this data structure when inserting new elements into it. When searching for unitaries with specific cost we use C++ *set* container to store only unique elements. We

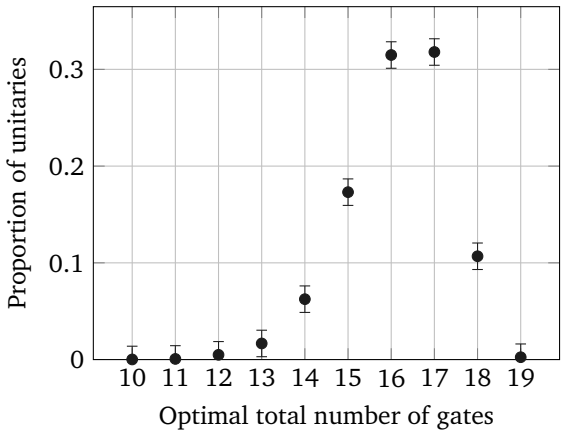

Figure 8.3: Estimated proportion of the 5-qubit Clifford unitaries per optimal gate count (independent input/output relabelling allowed).

build layers one by one. To build the layer k we pick an element of the layer $k-1$—we call it a *parent* unitary. Then we compose it with all possible gates and check if the resulting unitary was not found earlier. The only possible costs of the resulting unitary are k, $k-1$, or $k-2$. If we get cost less than $k-2$ this contradicts the knowledge that the cost of the parent unitary is indeed $k-1$. Therefore, during the search we need to keep only two previous layers in the memory. We repeat the procedure for all unitaries in the layer $k-1$. It can be executed in parallel for several parent unitaries. Only the addition of the unitaries with cost k to the *set* container must be synchronized. After the layer was built we copy the content of the *set* container into the sorted array and start building a new layer.

Finally, we describe how to find a circuit using the precomputed layers. If we find that a unitary belongs to layer k this means that there exists a circuit with k gates computing the unitary. Therefore, by removing the last gate in the circuit we obtain an optimal circuit with $k-1$ gates which corresponds to a unitary with cost $k-1$. By composing the source unitary with all possible gates and checking the cost of the result we identify the last gate in the optimal circuit. We proceed further in a similar fashion, until we reach the canonical representative of the identity. In the case when we relabel inputs and output simultaneously we always get an identity in the end. When relabelling of inputs and outputs is independent we obtain a circuit that is composed entirely of SWAP gates that represents a permutation of the inputs.

8.3 Experimental results

In this section we describe the results of our search together with the optimization experiments that rely on the databases of the optimal circuits we found. For the experiments that require more than 8 GB of RAM memory we used a high performance server with eight Quad-Core AMD Opteron 8356 (2.30 GHz) processors and 128 GB of RAM memory. These are the experiments involving 4- and 5-qubit Clifford unitaries. For all other experiments we used a machine with a single quad-core Intel Core i7-2600 (3.40 GHz) processor and 8 GB of RAM.

Distribution of the optimal circuits

We found optimal circuits for Clifford unitaries acting on 2 to 4 qubits (Figs. 8.1, 8.2) and optimal linear reversible circuits acting on up to 6 qubits. In both cases we found both circuits with the optimal gate count and those with the minimal depth. For the case of Clifford unitaries we also found circuits with optimal number of controlled-Z gates (i.e., replacing the CNOT with the controlled-Z gate in the elementary gate set).

Distributions reported in Figs. 8.1, 8.2 are interesting for the randomized benchmarking of quantum information processing systems. The benchmarking protocol [54] involves the application of a large number of randomly chosen Clifford unitaries. Knowledge of the distribution of the number of gates allows us to estimate the average time required for each experiment, and evaluate its feasibility due to, e.g., the effects of the decoherence. Using optimal circuits also minimizes the time required for the experiment. Finally, this data may be used to estimate the average fidelity of the two-qubit gates used to perform the benchmarking protocol. This is because it is based on the knowledge of the average number of two-qubit gates used [25]; the latter follows directly from our results.

Five-qubit Clifford unitaries

The search for five-qubit unitaries up to input/output order is challenging, but it is still tractable using modern computers. The number of different unitaries on five qubits is about 2.4×10^{17} (Table 8.1). We need 100 bits to store each group element. Factoring out simultaneous relabelling of inputs and output allows us to reduce the size of the database by a factor of approximately 120. However, one still needs at least 2.41×10^6 GB to store the full database in this case. To allow the search of any 5-qubit Clifford unitary up to input/output order we allowed the independent relabelling of the inputs and outputs of

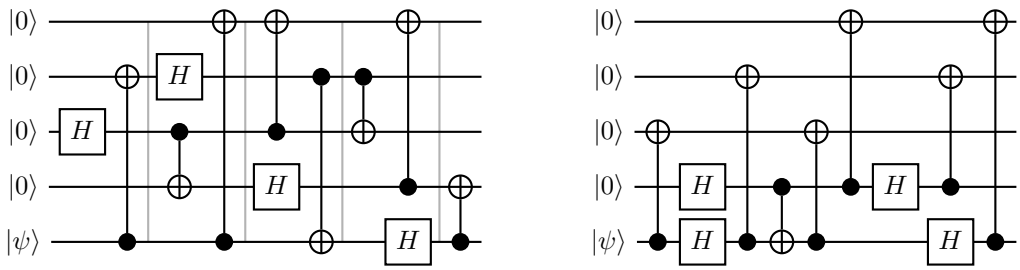

Figure 8.4: Optimal encoding circuits for the five-qubit code: (left) depth optimal circuit, depth=5; (right) circuit with the minimal number of gates, being 11 gates. Input marked $|\psi\rangle$ corresponds to the state being encoded.

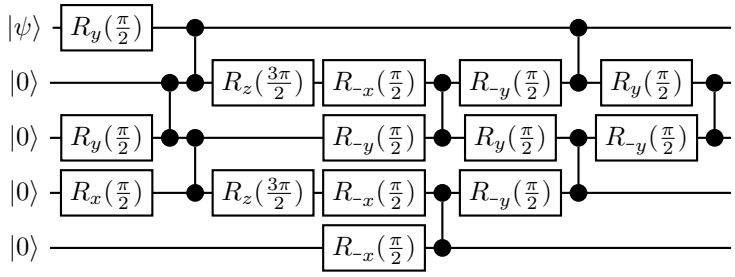

Figure 8.5: Encoding circuit for the five-qubit code used in [51]. The two-qubit gate corresponds to $e^{-iZZ\pi/4}$. Eight of them are required to implement the encoding circuit.

the circuits and used the meet in the middle [27] approach. We synthesized all 5-qubit unitaries that use up to 11 gates which allowed us to search for unitaries that require up to 22 gates. It is unknown what is the maximum number of gates needed to implement any 5-qubit Clifford unitary. We ran an experiment to estimate the distribution of the number of gates required to implement a unitary. We used the algorithm described in [19] to generate uniformly distributed random Clifford unitaries and found their gate count. The distribution of the number of gates for 5-qubit unitaries, shown in Fig. 8.3, was obtained using 20,000 samples. We used the Hoeffding inequality [34] to estimate errors for confidence level 0.999. Based on the above calculation, we concluded that the use of the 11-layer database and the meet in the middle approach should allow finding optimal circuits for any 5-qubit Clifford unitary up to input/output order.

Peephole optimization

We used the database of the optimal 4-qubit Clifford circuits to perform peephole optimization. Briefly, peephole optimization works as follows [63]. First, choose a pivot gate from the circuit and enumerate all subcircuits including it and acting on the number of qubits less or equal to 4 (otherwise, some small parameter of choice, which in our case was 4). Next, for each subcircuit find its cost and the optimal cost of the unitary that it computes. When beneficial, replace the less efficient subcircuit with its optimal version. This procedure is repeated until it is no longer possible to reduce the cost of some subcircuit corresponding to some pivot element.

When enumerating subcircuits we take into account that some gates commute and we can build larger subcircuits by moving gates. This requires us to examine the whole circuit at each step and results in quadratic complexity of the algorithm in the number of gates in the circuit. In practice, the algorithm runtime depends on the circuit structure. Furthermore, for large size circuits a different and more efficient algorithm may be employed to find suitable subcircuits, including limiting the "window" in which the gates are to be found (limiting this window to a constant size results in the reduction of the algorithm complexity from quadratic to linear). We did not investigate this further since in practice the subcircuit extraction algorithm with unbounded window did not take very long to complete for the circuits we tried. A more detailed description of the peephole optimization may be found in [63], along with a description of the version of the algorithm that produces slightly worse results, but requires linear time in the number of gates.

We applied peephole optimization to encoding circuits for quantum error correcting codes (QECCs). To obtain an encoding circuit for QECC one starts with the stabilizer generators of the code and applies an algorithm that produces the encoding circuit. We implemented two algorithms. The first one is a version of the canonical decomposition theorem [1] for stabilizers that produces stages of CNOT, H, and P gates (Algorithm 1). The second one (Algorithm 2), taken from [29], produces circuits that do not have an expressed staged structure. Table 8.2 summarizes the results of our experiment with codes from [31]. The code for these experiments was not parallelized. Applying peephole optimization to the circuits produced by Algorithm 2 results in a reduction of the number of gates by 45-53%.

Optimal encoding circuit for five-qubit quantum error correcting code

Using a slightly modified version of our algorithm we found a depth optimal circuit for the five-qubit $[[5, 1, 3]]$ error correcting code. This code encodes one qubit and corrects any single qubit error. In this case only first four out of 10 lines of the binary symplectic matrix are specified. We first found depth optimal circuits that produce matrices with different first four lines. The problem has an extra degree of freedom—the addition of lines of the binary symplectic matrix to each other does not change the code. In other words, left multiplication of the specified part of the binary symplectic matrix by a 4×4 invertible binary matrix leaves the code unchanged. Search for all four-bit optimal linear reversible circuits gave us a database of all 4×4 invertible binary matrices. We used it to go through all matrices equivalent to the one that defines the five-qubit code. Depth and gate count optimal circuits found are shown in Fig. 8.4. One of the best previously known circuits is illustrated in Fig. 8.5. Our approach may also be used to synthesize optimal circuits for other quantum error correcting codes that use a small number of qubits.

Conclusions

We explored the limitations of the brute force search for optimal circuits implementing Clifford and linear reversible unitaries. Using typical memory and processing power available today, it is possible to search for up to four-qubit optimal Clifford unitaries and six-qubit linear reversible unitaries. We also demonstrated that additional assumptions allow to search for optimal Clifford unitaries with up to five qubits. It is possible to make further assumptions resulting in greater sub-optimality, but reducing the size of the search space. For example, one may allow to apply Hadamard gates to each output in the end of the circuit for free. This will further reduce the size of search space by approximately 2^n, where n is the number of qubits. It is easy to come up with canonical form computation for this case. Of course, circuits produced by the algorithm will not be exactly optimal. However, the results will be very close to optimal if the cost of Hadamard gates is small. If we do not have the symmetry between all qubits we will have to deal with the larger search space.

Using lookup in our database as a part of the peephole optimization shows that this is an efficient and promising approach for the optimization of larger Clifford circuits.

Code	c_1	c_{1o}	c_2	c_{2o}	t_{1o}	t_{2o}.
[[25,1,9]]	440	285	387	205	22.2707	6.57012
[[26,1,9]]	444	287	389	207	22.8359	7.97804
[[26,4,8]]	500	336	528	250	30.073	18.6791
[[27,1,9]]	592	396	479	241	31.8254	15.1547
[[27,2,9]]	559	377	568	295	39.9342	18.5159
[[27,3,9]]	566	373	566	274	38.629	10.6849
[[27,4,8]]	504	335	530	252	28.0685	22.9666
[[27,8,6]]	498	341	558	305	45.949	15.2595
[[27,9,6]]	453	310	588	305	32.2922	17.1963
[[27,10,5]]	428	293	563	293	59.7698	10.5993
[[27,11,5]]	409	279	541	295	29.6078	12.5559
[[28,0,10]]	652	446	526	248	45.3604	18.2336
[[28,1,10]]	660	446	531	284	41.0448	13.5861
[[28,2,10]]	666	427	592	285	44.143	16.4625
[[28,3,9]]	570	378	568	276	60.009	10.2351
[[29,0,11]]	726	479	597	288	63.5229	12.0342
[[29,1,11]]	709	477	572	294	59.994	10.7589
[[29,2,10]]	670	430	594	287	42.623	19.8844
[[29,3,9]]	574	380	570	278	59.022	12.6315
[[29,4,8]]	512	341	534	256	30.7405	30.2718
[[29,5,7]]	492	305	518	263	29.5458	31.1485
[[29,6,7]]	602	409	577	318	52.8024	14.8819
[[29,7,6]]	549	376	593	298	28.9031	20.17
[[29,8,6]]	488	318	576	313	45.6243	10.709
[[30,0,12]]	813	524	662	310	71.1554	18.601
[[30,1,11]]	713	479	574	296	60.7773	12.9511
[[30,2,10]]	674	432	596	289	39.7054	24.7658
[[30,4,8]]	516	349	536	258	34.0143	34.5184
[[30,8,7]]	627	425	707	378	75.6614	22.7352
[[40,30,4]]	452	311	679	362	198.226	41.9046

Table 8.2: The results of application of the peephole optimization to encoding circuits for Quantum Error Correcting codes. [[n,k,d]] denotes the code that uses n physical qubits, encodes k logical qubits and has distance d, c_k — number of gates in the circuit obtained using Algorithm k, c_{ko} — number of gates in the circuit after application of the peephole optimization using the database of 4-qubit optimal Clifford circuits, t_{ko} — runtime of peephole optimization software (in seconds, user time, using single core of Intel Core i7-2600) as applied to the circuits produced by the Algorithm k.

Chapter 9

Conclusions

There are two main themes in this work. The first theme is bringing new tools to quantum compiling. This includes the exact synthesis algorithms for unitaries, the theory of relative norm equations and continued fractions. Combining these tools allows one to solve the approximate synthesis problem in an efficient way and to find approximations that are remarkably close to the optimal. Contributions by the author of this work and his collaborators were originally published in several papers [49, 47, 48, 45, 46]. Other important contributions to the topic include [26, 68, 6]. This results in significant reduction of resources needed to execute quantum algorithms fault-tolerantly using the Clifford+T gate set or on a topological quantum computer based on Fibonacci anyons. Interestingly, for the Clifford+T gate set, very similar reduction in the number of resources can be achieved by taking a different approach (developed in parallel with the number theoretic method) and summarized in [38]. For the Fibonacci gate set we achieved the improvement by factors from 20 to 1000 for the range of precisions 10^{-10} to 10^{-30} in comparison to the previous state of the art.

The second theme is related to finding optimal solutions to quantum compiling problems (Chapters 5,8). The resources required for these methods usually scale exponentially with the precision or the number of qubits used. However, we find that these methods are useful in a practical setting. For example, the method developed in Chapter 5 allows us to find circuits that are on average 25% shorter than those produced by the polynomial time algorithm that solves the same problem [68], and we handle precision of approximation down to 10^{-15}. We also demonstrated that one can find optimal circuits for Clifford unitaries acting on up to four qubits. We showed how to use this to reduce the number of gates used in encoding circuits by 40%-50%. The success of using these practical methods depends not only on the theoretical ideas behind them, but also on the quality of their

implementation. The implementation defines how small the constants behind the big O are and how far one can push one idea or the other. As the field of quantum compiling becomes more and more mature, the role of these details might become more and more pronounced. For these reason the implementation of the algorithm described in Chapter 5 is available on the Internet as an open source project at `http://code.google.com/p/sqct`.

There are many interesting questions left in quantum compiling. The unified framework for the number theoretic method is one of them. It would be interesting to develop a unified approach to describing exact synthesis algorithms. This is, probably, one of the most serious bottlenecks on the way to the generalization of the number theoretic method. The theory of relative norm equations is very well developed and can be applied in far more general situations. The other obstacle could be the difference between the round-off procedure used for the Clifford+T gate set [68] and the Fibonacci gate set. In the first case one needs to do extra work to satisfy the necessary condition for the norm equation to be solvable. In the second case the necessary condition is satisfied automatically and one can just use continued fractions to find the approximation. The approximation procedure from [68] might be related to the Minkowski theorem from the geometry of numbers. It will be interesting to investigate this connection further.

The more general question is where the next big resource reduction is possible. There are several interesting directions which one might investigate. The first one is related to understanding what is the necessary overhead when designing different parts of a fault tolerant protocol. One of the possible ways to address the question is new methods for proving the threshold theorems that take into account more detailed noise models of the hardware used. The second direction, is the design of parts of quantum algorithms that accounts for more refined cost models. The automatic resource estimation tools for quantum algorithm will be useful for this task. It could be the case that different architectures will require different ways of designing algorithm building blocks. For example, it is not possible to implement reversible circuits using the Fibonacci gate set exactly. For this reason, the adders based on Quantum Fourier Transform could be more suitable for this architecture. The third interesting direction is a more generic framework for circuit synthesis methods based on resource states. The specific question that could be interesting in this direction is finding tight lower bounds for multiple qubit state preparation when using Clifford+T or Fibonacci gate sets.

I look forward to the next advances in quantum compiling. This field is crucial for harnessing the power of quantum computers as soon as the first moderately large quantum computers are built.

References

[1] Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5), November 2004. arXiv:0406196, doi:10.1103/PhysRevA.70.052328.

[2] Alfred V. Aho and Krysta M. Svore. Compiling Quantum Circuits using the Palindrome Transform. November 2003. arXiv:0311008.

[3] Panos Aliferis, Daniel Gottesman, and John Preskill. Quantum accuracy threshold for concatenated distance-3 codes. *Quantum Information & Computation*, 6(2):97–165, April 2006. arXiv:0504218.

[4] Matthew Amy, Dmitri Maslov, Michele Mosca, and Martin Roetteler. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(6):1–20, June 2013. arXiv:arXiv:1206.0758v1, doi:10.1109/TCAD.2013.2244643.

[5] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5):3457–3467, November 1995. arXiv:9503016, doi:10.1103/PhysRevA.52.3457.

[6] Alex Bocharov, Yuri Gurevich, and Krysta M. Svore. Efficient Decomposition of Single-Qubit Gates into V Basis Circuits. *Physical Review A*, 88(1):1–13, July 2013. arXiv:arXiv:1303.1411v1, doi:10.1103/PhysRevA.88.012313.

[7] Alex Bocharov and Krysta M. Svore. Resource-Optimal Single-Qubit Quantum Circuits. *Physical Review Letters*, 109(19):190501, November 2012. arXiv:1206.3223, doi:10.1103/PhysRevLett.109.190501.

[8] Jean Bourgain and Alexander Gamburd. On the spectral gap for finitely-generated subgroups of SU (2). *Inventiones mathematicae*, 171(1):83–121, 2008.

[9] Jean Bourgain and Alexander Gamburd. Spectral gaps in SU (d). *Comptes Rendus Mathematique*, 348(11):609–611, 2010.

[10] Michele Burrello, Haitan Xu, Giuseppe Mussardo, and Xin Wan. Topological Quantum Hashing with the Icosahedral Group. *Physical Review Letters*, 104(16):160502, April 2010. doi:10.1103/PhysRevLett.104.160502.

[11] Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors. *Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. URL: http://link.springer.com/10.1007/11523468, doi:10.1007/11523468.

[12] Isaac L. Chuang and Michael A. Nielsen. Prescription for experimental determination of the dynamics of a quantum black box. *Journal of Modern Optics*, 44(11-12):2455–2467, November 1997. doi:10.1080/09500349708231894.

[13] Henri Cohen. *A Course in Computational Algebraic Number Theory*. Graduate Texts in Mathematics. Springer, 1993.

[14] Henri Cohen. *Advanced Topics in Computational Number Theory*. Graduate Texts in Mathematics. Springer New York, 2000.

[15] Henri Cohen. *Number Theory: Volume I: Tools and Diophantine Equations*. Graduate Texts in Mathematics. Springer, 2007.

[16] Henri Cohen, Karim Belabas, and Others. PARI/GP, a computer algebra system. URL: http://pari.math.u-bordeaux.fr/.

[17] Antonio D. Córcoles, Jay M. Gambetta, Jerry M. Chow, John A. Smolin, Matthew Ware, Joel Strand, Britton L. T. Plourde, and Matthias Steffen. Process verification of two-qubit quantum gates by randomized benchmarking. *Physical Review A*, 87(3):030301, March 2013. doi:10.1103/PhysRevA.87.030301.

[18] Christopher M. Dawson and Michael A. Nielsen. The Solovay-Kitaev algorithm. *Quantum Information & Computation*, 6(1):81–95, May 2005. arXiv:0505030.

[19] David P. DiVincenzo, Debbie W. Leung, and Barbara M. Terhal. Quantum data hiding. *IEEE Transactions on Information Theory*, 48(3):580–598, March 2002. arXiv:0103098, doi:10.1109/18.985948.

[20] Thomas G. Draper, Samuel A. Kutin, Eric M. Rains, and Krysta M. Svore. A logarithmic-depth quantum carry-lookahead adder. *Quantum Information & Computation*, 6(4-5):351–369, 2006. arXiv:0406142.

[21] Guillaume Duclos-Cianci and Krysta M. Svore. Distillation of nonstabilizer states for universal quantum computation. *Physical Review A*, 88(4):042325, October 2013. arXiv:arXiv:1210.1980v1, doi:10.1103/PhysRevA.88.042325.

[22] Austin G. Fowler. Constructing arbitrary Steane code single logical qubit fault-tolerant gates. *Quantum Information & Computation*, 11(9):8, November 2011. arXiv:0411206.

[23] Austin G. Fowler, Ashley Stephens, and Peter Groszkowski. High-threshold universal quantum computation on the surface code. *Physical Review A*, 80(5):052312, November 2009. doi:10.1103/PhysRevA.80.052312.

[24] Michael H. Freedman, Michael Larsen, and Zhenghan Wang. A Modular Functor Which is Universal for Quantum Computation. *Communications in Mathematical Physics*, 227(3):605–622, June 2002. doi:10.1007/s002200200645.

[25] John Gaebler, Adam Meier, Ting Rei Tan, Ryan Bowler, Yiheng Lin, David Hanneke, Jone Jost, Jonathan Home, Emanuel Knill, Dietrich Leibfried, and David Wineland. Randomized Benchmarking of Multiqubit Gates. *Physical Review Letters*, 108(26):1–5, June 2012. doi:10.1103/PhysRevLett.108.260503.

[26] Brett Giles and Peter Selinger. Exact synthesis of multiqubit Clifford+T circuits. *Physical Review A*, 87(3):032332, March 2013. arXiv:1212.0506, doi:10.1103/PhysRevA.87.032332.

[27] Oleg Golubitsky and Dmitri Maslov. A Study of Optimal 4-Bit Reversible Toffoli Circuits and Their Synthesis. *IEEE Transactions on Computers*, 61(9):1341–1353, September 2012. doi:10.1109/TC.2011.144.

[28] David Gosset, Vadym Kliuchnikov, Michele Mosca, and Vincent Russo. An algorithm for the T-count. August 2013. arXiv:1308.4134.

[29] Daniel Gottesman. (Unpublished).

[30] Daniel Gottesman. What is the Overhead Required for Fault-Tolerant Quantum Computation? October 2013. arXiv:1310.2984.

[31] Markus Grassl. Encoding Circuits for Quantum Error-Correcting Codes, 2007. URL: http://i20smtp.ira.uka.de/home/grassl/QECC/circuits/index.html.

[32] Aram W. Harrow, Benjamin Recht, and Isaac L. Chuang. Efficient discrete approximations of quantum gates. *Journal of Mathematical Physics*, 43(9):4445, November 2002. arXiv:0111031, doi:10.1063/1.1495899.

[33] Matthew B. Hastings. Decoding in Hyperbolic Spaces: LDPC Codes With Linear Rate and Efficient Error Correction. December 2013. arXiv:1312.2546.

[34] Wassily Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association*, 58(301):13–30, 2012.

[35] Layla Hormozi, Georgios Zikos, Nicholas Bonesteel, and Steven H. Simon. Topological quantum compiling. *Physical Review B*, 75(16):165310, April 2007. doi:10.1103/PhysRevB.75.165310.

[36] N. Cody Jones. Distillation protocols for Fourier states in quantum computing. pages 1–18, 2013. arXiv:1303.3066.

[37] N. Cody Jones. Distillation protocols for Fourier states in quantum computing. pages 1—-18, March 2013. URL: http://arxiv.org/abs/1303.3066, arXiv:1303.3066.

[38] N. Cody Jones. Logic Synthesis for Fault-Tolerant Quantum Computers. pages 1–201, October 2013. arXiv:1310.7290.

[39] N. Cody Jones. Low-overhead constructions for the fault-tolerant Toffoli gate. *Physical Review A*, 87(2):022328, February 2013. doi:10.1103/PhysRevA.87.022328.

[40] N. Cody Jones, Rodney Van Meter, Austin G. Fowler, Peter L. McMahon, Jungsang Kim, Thaddeus D. Ladd, and Yoshihisa Yamamoto. Layered Architecture for Quantum Computing. *Physical Review X*, 2(3):031007, July 2012. doi:10.1103/PhysRevX.2.031007.

[41] N. Cody Jones, James D. Whitfield, Peter L. McMahon, Man-hong Yung, Rodney Van Meter, Alán Aspuru-Guzik, and Yoshihisa Yamamoto. Faster quantum chemistry simulation on fault-tolerant quantum computers. *New Journal of Physics*, 14(11):115023, November 2012. arXiv:1204.0567, doi:10.1088/1367-2630/14/11/115023.

[42] Ivan Kassal, James D. Whitfield, Alejandro Perdomo-Ortiz, Man-Hong Yung, and Alán Aspuru-Guzik. Simulating chemistry using quantum computers. *Annual review of physical chemistry*, 62:185–207, January 2011. doi:10.1146/annurev-physchem-032210-103512.

[43] Alexei Kitaev. Quantum measurements and the Abelian Stabilizer Problem. November 1995. arXiv:9511026.

[44] Alexei Kitaev, Alexander Shen, and Mikhail Vyalyi. *Classical and Quantum Computation*. Graduate studies in mathematics, v. 47. American Mathematical Society, Boston, MA, USA, 2002.

[45] Vadym Kliuchnikov. Synthesis of unitaries with Clifford+T circuits. June 2013. arXiv:1306.3200.

[46] Vadym Kliuchnikov, Alex Bocharov, and Krysta M. Svore. Asymptotically Optimal Topological Quantum Compiling. October 2013. arXiv:1310.4150.

[47] Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. Asymptotically optimal approximation of single qubit unitaries by Clifford and T circuits using a constant number of ancillary qubits. *Physical Review Letters*, 110(19):1–5, December 2012. arXiv:1212.0822, doi:10.1103/PhysRevLett.110.190502.

[48] Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. Practical approximation of single-qubit unitaries by single-qubit quantum Clifford and T circuits. December 2012. arXiv:1212.6964.

[49] Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. Fast and efficient exact synthesis of single qubit unitaries generated by Clifford and T gates. *Quantum Information & Computation*, 13(7-8):0607–0630, June 2013. URL: http://arxiv.org/abs/1206.5236, arXiv:1206.5236.

[50] Emanuel Knill. Approximation by Quantum Circuits. (May):1–23, August 1995. arXiv:9508006.

[51] Emanuel Knill, Raymond Laflamme, Rodolfo A. Martinez, and Camille Negrevergne. Benchmarking Quantum Computers: The Five-Qubit Error Correcting Code. *Physical Review Letters*, 86(25):5811–5814, June 2001. doi:10.1103/PhysRevLett.86.5811.

[52] Hendrik W. Lenstra Jr. Euclid's algorithm in cyclotomic fields. *Journal of the London Mathematical Society*, 10:457–465, 1975.

[53] Easwar Magesan, Jay M. Gambetta, and Joseph Emerson. Scalable and Robust Randomized Benchmarking of Quantum Processes. *Physical Review Letters*, 106(18):180504, May 2011. URL: http://link.aps.org/doi/10.1103/PhysRevLett.106.180504, doi:10.1103/PhysRevLett.106.180504.

[54] Easwar Magesan, Jay M. Gambetta, and Joseph Emerson. Characterizing quantum gates via randomized benchmarking. *Physical Review A*, 85(4):042311, April 2012. doi:10.1103/PhysRevA.85.042311.

[55] Ken Matsumoto and Kazuyuki Amano. Representation of Quantum Circuits with Clifford and pi/8 Gates. June 2008. arXiv:0806.3834.

[56] Roger S. K. Mong, David J. Clarke, Jason Alicea, Netanel H. Lindner, Paul Fendley, Chetan Nayak, Yuval Oreg, Ady Stern, Erez Berg, Kirill Shtengel, and Matthew P. A. Fisher. Universal topological quantum computation from a superconductor/Abelian quantum Hall heterostructure. July 2013. arXiv:1307.4403.

[57] Mikko Möttönen, Juha Vartiainen, Ville Bergholm, and Martti Salomaa. Quantum Circuits for General Multiqubit Gates. *Physical Review Letters*, 93(13):130502, September 2004. arXiv:0404089, doi:10.1103/PhysRevLett.93.130502.

[58] Chetan Nayak, Ady Stern, Michael H. Freedman, and Sankar Das Sarma. Non-Abelian anyons and topological quantum computation. *Reviews of Modern Physics*, 80(3):1083–1159, September 2008. doi:10.1103/RevModPhys.80.1083.

[59] Gabriele Nebe, Eric M. Rains, and Neil J.A. Sloane. *Self-Dual Codes and Invariant Theory*. Algorithms and Computation in Mathematics. Springer, 2006.

[60] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[61] Gerardo Ortiz, James Gubernatis, Emanuel Knill, and Raymond Laflamme. Quantum algorithms for fermionic simulations. *Physical Review A*, 64(2):022319, July 2001. doi:10.1103/PhysRevA.64.022319.

[62] Adam Paetznick and Krysta M. Svore. Repeat-Until-Success: Non-deterministic decomposition of single-qubit unitaries. pages 1–24, 2003. arXiv:1311.1074.

[63] Aditya K. Prasad, Vivek V. Shende, Ketan N. Patel, Igor L. Markov, and John P. Hayes. Data structures and algorithms for simplifying reversible circuits. *ACM Journal on Emerging Technologies in Computing Systems*, 2(4):277–293, October 2006. doi:10.1145/1216396.1216399.

[64] John Preskill. Lecture Notes for Physics 219: Quantum Computation. Topological quantum computation. URL: http://www.theory.caltech.edu/~preskill/ph219/topological.pdf.

[65] John Preskill. Reliable quantum computers. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 454(1969):385–410, January 1998. doi:10.1098/rspa.1998.0167.

[66] Michael O. Rabin and Jeffery O. Shallit. Randomized algorithms in number theory. *Communications on Pure and Applied Mathematics*, 39(S1):S239–S256, 1986. doi:10.1002/cpa.3160390713.

[67] Colm Ryan, Martin Laforest, and Raymond Laflamme. Randomized benchmarking of single- and multi-qubit control in liquid-state NMR quantum information processing. *New Journal of Physics*, 11(1):013034, January 2009. arXiv:0808.3973, doi:10.1088/1367-2630/11/1/013034.

[68] Peter Selinger. Efficient Clifford+T approximation of single-qubit operators. December 2012. arXiv:1212.6253.

[69] Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing*, 26(5):1484–1509, October 1997. doi:10.1137/S0097539795293172.

[70] Lawrence C. Washington. *Introduction to cyclotomic fields*, volume 83. Springer, 1997.

[71] Dave Wecker, Bela Bauer, Bryan K. Clark, Matthew B. Hastings, and Matthias Troyer. Can quantum chemistry be performed on a small quantum computer? page 15, December 2013. arXiv:1312.1695.

[72] Nathan Wiebe and Vadym Kliuchnikov. Floating point representations in quantum circuit synthesis. *New Journal of Physics*, 15(9):093041, September 2013. arXiv:1305.5528, doi:10.1088/1367-2630/15/9/093041.