5-2018

Efficient quantum approximation : examining the efficiency of select universal gate sets in approximating 1-qubit quantum gates.

Brent A. Mode
University of Louisville

Efficient Quantum Approximation: Examining the Efficiency of Select Universal Gate Sets in Approximating 1-Qubit Quantum Gates

By
Brent Alan William Mode

A Dissertation
Submitted to the Faculty of the
College of Arts and Sciencesof the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Bachelor of Science
in Physics

Department of Physics and Astronomy
University of Louisville
Louisville, Kentucky

May 2018

Efficient Quantum Approximation: Examining the Efficiency of Select Universal Gate Sets in Approximating 1-Qubit Quantum Gates

By

Brent Alan William Mode

Dissertation approved on

March 22, 2018

by the following dissertation Committee:

Dissertation Director
Prof. David N. Brown

Dr. Steven B. Damelin

Prof. Eugene Mueller

DEDICATION

For Catie and whatever undergraduates decide to flip through this for some reason.

ACKNOWLEDGMENTS

This project was supported in part by both Dr. Dave Brown (known as Ole Doc B' to his students) and Dr. Steven Damelin. Dr. Brown has been a mentor to me since before I came to the University of Louisville in 2014, and I will be eternally grateful for the impact that he has had on my life. He is an excellent teacher, mentor, advisor, and researcher. I hope to one day be half the professor he is. Dr. Damelin took me on as a summer research student as part of the physics REU program at the University of Michigan. As a mathematician, mentoring a physics student was, I am sure, a novel experience. I am grateful both for his genuine patience and constant optimism as I worked on that project and this one. I am perhaps most grateful for his advice on navigating the world of academia beyond its idealization.

With a project of of this kind, there is also an enormous list of resources, services, and software to acknowledge: Python, Numpy, Scipy, C++, Christopher Dawson's original C++ implementation of the Solovay-Kitaev algorithm on Github, and Dawson and Nielsen's seminal article on the subject.

ABSTRACT

Quantum computation is of current ubiquitous interest in physics, computer science, and the public interest. In the not-so-distant future, quantum computers will be relatively common pieces of research equipment. Eventually, one can expect an actively quantum computer to be a common feature of life. In this work, I study the approximation efficiency of several common universal quantum gate sets at short sequence lengths using an implementation of the Solovay-Kitaev algorithm. I begin by developing from almost nothing the relevant formal mathematics to rigorously describe what one means by the terms universal gate set and covering efficiency. I then describe some interesting results on the asymptotic covering properties of certain classes of universal gate sets and discuss the theorem which the Solovay-Kitaev algorithm is based on.

Moving from mathematical introduction to experimental method, I then describe how sets will be compared. I use the commonly studied sets H+T, Pauli+V, V, and Clifford+T to determine which is the most efficient at approximating randomly generated unitaries. By doing so, we get an understanding of how well each set would perform in the context of a general quantum computer processor. This was accomplished by using the same implementation of the Solovay-Kitaev algorithm throughout, with roughly equal-sized preprocessed libraries formed from each gate set, over approximations for $10,000$ randomly generated unitary matrices at algorithm depth $n = 5$. Ultimately, the Pauli+V and V sets were the most efficient and had similar performance qualities. On average the Pauli+V set produced approximations of length $15,491$ and accuracy 0.0002686. The V basis produced approximations of average sequence length $16,403$ and accuracy 0.0001465. This performance is about equal given this particular implementation of the Solovay-Kitaev algorithm.

We conclude that this result is somewhat surprising as the general behavior and efficiency of these particular choices of gate set are expected to be similar. It is possible though that the asymptotic efficiencies of these gate sets vary by a relatively wide margin and this has effected the experiment. It is also possible that some aspect of a naive implementation of the Solovay-Kitaev algorithm resulted in the Hadamard gate based sets performing more poorly than the V basis sets overall. Due to constraints on computational power, this result could also be limited to this particular accuracy regime and could even out as tolerance ε is taken to be arbitrarily small. Further possibilities of this result as well as further work are then discussed.

TABLE OF CONTENTS

Dedication iii

Acknowledgments iv

Abstract v

Table of Contents vii

List of Tables viii

Introduction 1
- Qualitative Introduction to Quantum Computing 1
- Background Mathematics 3
- Approximation of Quantum Logic Gates 14

Method 17
- Group Factor Commutator 17
- The Solovay-Kitaev Algorithm 18
- Implementation of the Solovay-Kitaev Algorithm in Python 19
- Using the Solovay-Kitaev Algorithm 19

Results 21

Summary and Further Work 24
- Summary 24
- Further Work 24

Bibliography 26

LIST OF TABLES

1 Cayley table for $\mathbb{Z}_3$ 9

2 Results for $10,000$ approximations using common gate sets. 22

CHAPTER I

Introduction

1 Qualitative Introduction to Quantum Computing

Computers have inexorably become a part of public life over the last three decades, giving rise to entire industries, redefining almost all of modern culture, and providing a secure background to almost every financial transaction that occurs anywhere in the first world. They live in our pockets, on our desks, above our hearths, and in our laboratories. Contemporary scientific progress is wholly unthinkable in the absence of computer-based data collection and analysis. We create computers to monitor every minute aspect of an experiment and write programs that analyze an unthinkable amount of information in a matter of moments. It is perhaps impossible to understate the overwhelming effects of technology saturating life.

Underlying every computer that has made all of this possible is a relatively simple theoretic basis: Boolean logic. Known by mathematicians, philosophers, and computer scientists alike, and implicitly understood by most researchers, the simple notions of 'True' and 'False' as well as the AND, OR, and NOT relations between them make up all of regular computing. By combining logic gates that physically implement the AND, OR, and NOT operations, and feeding in appropriate True-False inputs, one may arrive at any program that has ever been run on a standard computer. Fundamentally, even the software that I am using to compose this document could be programmed into a complicated web of logic gates and logical inputs, resulting in outputs that are subsequently delivered to the screen as words and spaces. Of course this would be very inefficient from a design standpoint. The point of Intel and AMD spending time and resources designing tiny, power-efficient circuit boards is so that we can avoid starting from scratch every time we want to write a piece of software.

Processors have built in programming languages called assemblers that are specific to their design and allow for the most basic level of writing software that is not explicitly in the form of logic circuits[1]. From there, assembler can be used to write code defining more usable programming languages like C++ and programs called compilers that translate C++ code into machine language, the steps that the processor takes to run a program. At this point in history, most programming is far abstracted from the hardware that provides its foundations, with many modern languages such as Python being written in older, more low-level languages like C++ and Fortran. Though this abstracting necessarily means that software written in these languages

[1]Or more accurately, assemblers are a step up from machine language, which is a step up from hardware. However, assemblers are in close if not identical correspondence to machine language, so the difference between the two is minimal.

will run slower than a comparable program written in assembler, we have progressed to the point at which such differences are usually minimal. Modern languages offer benefits such as more readable syntax, simpler or more powerful program design, and the ability to run the same program on most computers regardless of operating system or CPU architecture.

The point is that any program that can be realized using Boolean logic can be exactly implemented on what will now be referred to as a 'classical' computer. The notion of approximating a program or relying on a logic gate that does not exactly provide the intended operation is nonsensical. Either a NOT gate returns the logical opposite of its input, or it does not. Unless it is broken, there is not any sense to the notion of a NOT gate taking a True input and returning almost-False. A further interesting note is that the gates AND, OR, and NOT are not the only gates that can be used to build any classical logic circuit[2]. The NAND gate, which is the AND gate with opposite output, is also universal. This can be shown be creating AND, OR, and NOT gates using only NAND gates. So ultimately, there is a notion of universality in classical computers, but the logic and gates are discretized and so are relatively simple.

Quantum computers represent a fundamental departure from classical computing. Gone is the simplicity of Boolean logic. Gone is the notion of logic that is always just True or just False, just 1 or 0[3]. Contrary to popular belief, the existence of a functional quantum computer at a scale sufficient to break the most common encryption schemes will not pose a real threat to national or global security[4]. It is also a common misconception that some aspect of quantum computers will allow them to run classical programs much more quickly than today's most advanced hardware will allow; quantum computing offers no benefits in running the same program as a classical computer[5]. However, quantum computers will allow for the possibility of algorithms that go beyond what is possible with classical computers. The most common example is of course Shor's algorithm, designed by Peter Shor and shown to be capable of factoring very large numbers quickly. Quantum computers are also very exciting to physicists for the possibility of running simulations of quantum systems, something not feasible at a large scale with conventional computers.

The difference between classical computers and quantum computers lies in how logic is implemented. In quantum computers, instead of having True and False logic values that are represented by some discrete physical property like voltage, there are quantum states that represent logical outputs. Instead of True we can say that we have the state[6] $|1\rangle$ and instead of False we have the state $|0\rangle$. These states

[2]AND, OR, and NOT are universal in the sense that any Boolean logic circuit can be implemented using these gates.

[3]To be clear, 1 and 0 are common mathematical representations of True and False, though of course, they are not really the same thing.

[4]This is because there are encryption schemes already invented that do not rely on factoring very large numbers. The research area focused on this is called post-quantum cryptography.

[5]This belief could perhaps be attributed to the quantum algorithm for quickly factoring large numbers. However, this is not a computation that most programs perform often, so it is of little benefit to common programs.

[6]The reason for the notation used here will be introduced in the following section.

are treated as a mathematical basis, which ultimately means that instead of being limited to creating a circuit that passes only these values, it is also possible to have combinations of the two states. Instead of calling the states bits, as they are called in classical computing, the quantum states are called qubits. The potential for having intermediary logic outputs that are neither $|1\rangle$ or $|0\rangle$ but somewhere in between means that even the simplest logic gates, the 1-qubit quantum logic gates that take a single input and return a single output, are much more complicated than before. With Boolean logic, there are only two conceivable 1-bit logic gates, and one of them is trivial: the NOT gate which flips the input and some sort of pass-through gate that does nothing (which of course doesn't exist). There are an infinite number of 1-qubit quantum logic gates. Unfortunately, without a proper mathematical introduction, it will be difficult to describe quantum computers or the current academic challenges being faced without a sufficient background in mathematics. The next section will attempt to build to that point from the ground up. For more on the basics of quantum computing, the standard text is [1].

2 Background Mathematics

In the sciences, full mathematical rigor is typically unnecessary. However, in the study of quantum computation theory, mathematics provides a necessary theoretical backbone. The intention of this section is to assume as little as possible about the reader's background and begin with fundamentals before building up to the formality required for an appropriate understanding of the subject.

Quantifiers

This is a brief section outlining common symbols that are used almost exclusively in physics and mathematics to shorten formal statements. The symbol $\in$ is read 'in,' 'is an element of,' or 'is a member of.' It is specifically used to describe membership in a set. So we would say that $1 \in \mathbb{R}$, which reads '1 is an element of the set of all real numbers' which of course, it is. The symbol $\forall$ is read 'for any,' or 'for all.' It is usually used when describing something that is true for every member of some structure, such as every element of a set. The symbol $\exists$ is read 'there exists' and is used to state that some such element of a set (or some other structure) has a given property. The symbol $\Rightarrow$ is read 'implies that' and is used to describe a logical implication, a statement where if the hypothesis is true, then the conclusion must also be true. Finally, the symbol $\Leftrightarrow$ is read 'is equivalent to' or 'is biconditional to' and is used when if one side of the statement is true, then the other is true. Finally the symbol $\ni$ simply means 'such that.'

Sets

One of the most fundamental branches of mathematics is set theory. Essentially, a set is a mathematical box that contains objects of no certain type. Sets can contain numbers, matrices, variables, functions, other sets, any kind of mathematical object.

A very simple example of a set is

$$A = \{1, 2, 3\}.$$

Here the set A contains a few integers. An important property of sets is the number of things they contain. This is called *cardinality*. The cardinality of A, denoted $|A|$, is 3.

There are several very important sets that are commonly used in mathematics. They are listed below:

$$\begin{aligned}
\emptyset &= \{\} \\
\mathbb{N} &= \{1, 2, 3, \ldots\} \\
\mathbb{Z} &= \{\ldots, -2, -1, 0, 1, 2, \ldots\} \\
\mathbb{Q} &= \{\frac{m}{n} | m \in \mathbb{Z}, n \in \mathbb{N}\} \\
\mathbb{R} &= \{\text{the set of all real numbers}\} \\
\mathbb{C} &= \{a + ib | a, b \in \mathbb{R}\}.
\end{aligned} \tag{1}$$

Here we have, in order, the empty set, the set of natural numbers, the set of integers, the set of rational numbers, the set of real numbers, and the set of complex numbers. In set notation, | should be read as "such that." It is important to note here that $|\emptyset| = 0$ while the cardinality of all the other sets is infinite. However, cardinality is no longer a specific enough distinction when describing the size of infinite sets. One says that the sets $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{Q}$ are countably infinite because it is possible to describe a function mapping each sets members to the natural numbers in a way that resembles counting[7]. On the other hand, the sets of real and complex numbers, $\mathbb{R}$ and $\mathbb{C}$, are much larger. We say that they are uncountably infinite. This distinction will show its importance shortly.

As regards common notation, there are several operations between sets. If two sets have the same elements, then we write that they are equal, the same as any other statement of equality. If the elements of one set are contained in another set than we say that one is a subset of the other, i.e. if all the elements of A are contained in B then we say that $A \subset B$. On the other hand, in the same scenario, we would say that B is a superset of A and write $A \supset B$. The notation is similar to that for greater than and less than. There are many other set operations. These include in particular union and intersection. The union of two sets is the set that contains all of the elements from both sets, e.g. $A = \{1, 2, 6\}$, $B = \{1, b, \pi\}$, and $A \cup B = \{1, 2, \pi, 6, b\}$, where $A \cup B$ is "A union B." On the other hand, the intersection of two sets is the set containing only the elements in common, e.g. with A and B as before, $A \cap B = \{1\}$. Set notation also exists to describe the union or intersection of many sets at once. Suppose for example that we have sets described by $A_n = \{n, n + 1\}$. Then we can

[7]It is much more accurate to say that the sets $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{Q}$ are countably infinite because there exists a bijective (or one-to-one) function $f : \mathbb{N} \to A$ from $\mathbb{N}$ to one of the three sets, represented as A. If no such bijection exists, the set is uncountably infinite.

describe the union of all such sets where n is a natural number in a way similar to summation:

$$\bigcup_{n=1}^{\infty} A_n = A_1 \cup A_2 \cup \cdots = \{1,2\} \cup \{2,3\} \cup \cdots = \mathbb{N}.$$

Multiple intersections are denoted similarly. There are two other common operations on sets, complements and differences. Suppose that the set A is a subset of U where U is considered universal, i.e. U contains all of the possible elements. Then the complement of A, written $\overline{A}$, is the set of all elements in U that are not in A. Thus, $A \cup \overline{A} = U$. The difference between two sets is the set of elements that the first set does not have in common with the second set. Thus, if $A = \{1,2,3\}$ and $B = \{1,2\}$, then $A - B = \{3\}$.

It is of course possible to describe ordered pairs using set theory as well. Suppose we are considering the sets A and B and we would like to create the set of all ordered pairs of elements from these sets. Then we would want $A \times B = \{(a,b) | a \in A, b \in B\}$, the Cartesian product of A and B. We notate the Cartesian product of a set with itself using exponential notation. For example $\mathbb{R} \times \mathbb{R}$ is $\mathbb{R}^2$, which is the set of points in the well-known Cartesian plane.

An additional useful notion is the *power set* of a given set. Consider a set A. Then the power set of A, denoted by $\mathcal{P}(A)$, is the set of all subsets of A. If A has finite cardinality $|A|$, then the cardinality of $\mathcal{P}(A)$ is $2^{|A|}$. Also note that $\emptyset \in \mathcal{P}(A)$ and $A \in \mathcal{P}(A)$.

Finally, this introduction to sets would be incomplete without briefly explaining a ubiquitous shorthand for particular subsets of $\mathbb{R}$ called intervals. Specifically, (a,b) represents a subset of the reals composed of all the real numbers between a and b where $a < b$, i.e. $(a,b) = \{x \in \mathbb{R} | a < x < b\}$. Here a parenthesis denotes that the endpoint is not included. The interval (a,b) is thus called an open interval. Similarly, the interval $[a,b] = \{x \in \mathbb{R} | a \leqslant x \leqslant b\}$ is called a closed interval because it contains both of its endpoints. The intervals $(a,b]$ and $[a,b)$ are also possible. This notation is very powerful when combined with the other ideas in this section, and in particular when describing functions of real numbers.

Functions

The function is the most commonly understood mathematical entity beyond basic arithmetic, finding regular use in every scientific discipline, including such disparate fields as economics, psychology, and physics. However, for the sake of notation, it would be wise to quickly review the formal definition of functions, as well as a few important classificiations that do not find use outside of mathematical theory.

Simply put, a function maps elements of one set to elements of another set (or the same set), with the caveat that to be a function one element from the first set is mapped to only one element of the other set. In the sciences, functions are typically accepted as mapping real numbers to real numbers, and this is implicitly understood. However, in the elaboration of quantum computation theory, this is not always desired. In formal mathematics, functions are initially described, for

instance, as $f : (0, \infty) \to \mathbb{R}$, where in this case the function f only takes inputs that are nonnegative reals, and outputs a real number. Perhaps this is the function $f(x) = \sqrt{x}$. This kind of initial notation helps clarify what the function is expected to do. When the initial set (or the final set) involves a different kind of object than a real number, this provides necessary clarity.

Though there are many classifications that can be used to describe various kinds of functions, there are three overarching classifications that will be described here for organizational reasons[8]. Suppose that for the following definitions we are considering the function $f : A \to B$. The first such classification is *injective* or *one-to-one*. We say that f is one-to-one if each element of A maps to a unique element of B, i.e. $\forall a, b \in A, f(a) = f(b) \Rightarrow a = b$. The second such classification is *surjective* or *onto*. We say that f is onto if each element of B gets mapped to by at least one element of A, i.e. $\forall b \in B, \exists a \in A \ni f(a) = b$. When a function is both one-to-one and onto, we say that it is bijective. A bijective function is one where every element of A is mapped to a unique element of B and all of B is covered. This often has important implications about mathematical structures, beyond the relatively simple notion of sets. Specifically, if f is bijective then $|A| = |B|$ which is an important property in and of itself. This first allowed mathematicians to demonstrate that cardinality is not sufficient for comparing the sizes of infinite sets. We say that an infinite set is *countable* if there exists a bijective function from $\mathbb{N}$ to the set in question. Thus, the elements of the set could in some way be 'counted' even if the set is infinite. Using this definition, we find that $\mathbb{Z}$ and even $\mathbb{Q}$ are countable sets. However, it can also be shown that there is no bijective function between $\mathbb{N}$ and $\mathbb{R}$. So we say that $\mathbb{R}$ is uncountable. This distinction will return later[9].

Matrices

An $m \times n$ matrix A refers to an array of numbers organized into m rows and n columns and denoted by

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}.$$

Here, the numbers m and n denote the dimension of the matrix. Matrices are governed by more particular rules of arithmetic than numbers. In order to add or subtract two matrices, they must have the same size or dimension. Addition occurs element by

[8]Specifically, these three classifications, one-to-one (injective), onto (surjective), and one-to-one correspondence (bijective), do not imply any underlying structure other than set theory, and so belong to the purest possible discussion of functions.

[9]As will also be described later, it is interesting to note the qualitative relationship between $\mathbb{R}$ and $\mathbb{Q}$, the reals and the rationals. With little thought, it becomes not surprising that one can create a rational number that is arbitrarily close to any real number. For example, imagine creating rational numbers that come closer and closer to the value of pi. This property is not unique to $\mathbb{Q}$ or even to the reals, and will prove of fundamental usefulness in describing quantum approximation.

element, e.g.

$$\begin{pmatrix} 2 & 1 \\ 1 & 4 \end{pmatrix} \begin{pmatrix} 4 & 1 \\ -1 & 2 \end{pmatrix} = \begin{pmatrix} 6 & 2 \\ 0 & 6 \end{pmatrix}$$

In order to multiply two matrices, the number of columns in the first must be the same as the number of rows in the second (the inner dimensions must agree, i.e. $A_{m,n} \times B_{n,p}$ is allowed). The operation of matrix multiplication is relatively complex. For example, when multiplying the $m \times n$ matrix A with the $n \times p$ matrix B, the first element of the product AB, ab_{11} would be given by $ab_{11} = a_{11}b_{11} + a_{12}b_{21} + \cdots + a_{1n}b_{n1}$. A full example of the product of two 2×2 matrices might be

$$\begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} 2 & 4 \\ 1 & 2 \end{pmatrix} = \begin{pmatrix} 4 & 8 \\ 7 & 14 \end{pmatrix}.$$

Given the importance of dimension and the relative complexity of the operation, it is important to note a few facts about matrix multiplication. First, matrix division is not an operation that is always allowed (or arguably exists in the same way that division does for the reals), something that can only be truly approximated using square matrices. Second, there is an equivalent to the number 1 in terms of matrix multiplication, a matrix whose product with any other matrix is that matrix. It is called the identity matrix and is given as

$$I_n = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}.$$

There are a few other possible operations on matrices. One such operation is the *determinant* of a matrix. For a given matrix, the determinant only exists if the matrix is square. For example,

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \det(A) = ad - bc,$$

shows the determinant of a 2×2 matrix. The determinant of the more complex $n \times n$ matrix is much more difficult to calculate. An important classification of matrices are those with nonzero determinants. Such matrices are called *nonsingular*. If a square matrix is nonsingular, then there exists an inverse of the matrix, i.e. $A_{n \times n}, \det(A) \neq 0 \Rightarrow \exists A^{-1} \ni AA^{-1} = A^{-1}A = I_n$. Multiplication by an inverse matrix is the closest thing to division that is accessible by matrices. All square matrices also have a *trace*. The trace of a matrix A, denoted $\mathrm{Tr}(A)$, is the sum of the entries on the main diagonal, i.e. if

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

then

$$\mathrm{Tr}(A) = a_{11} + a_{22} + \cdots + a_{nn}.$$

Another important elementary unary operation on a matrix is transposition. The transpose of an $m \times n$ matrix A is the $n \times m$ matrix $A^\intercal$. Each matrix entry essentially has its indices swapped, so for a_{ij} in A, the same entry is the element a_{ji} in $A^\intercal$. In other words, the transpose of a matrix is the reflection of the matrix about its main diagonal. The final unary operation that will be required is the Hermitian adjoint of a matrix. The Hermitian adjoint of matrix A is the complex conjugate transpose, denoted by $A^\dagger$. If $A = A^\dagger$, then we say that A is Hermitian. We will find matrices to be of utmost importance when describing the fundamental structure of 1-qubit quantum gates.

Group Theory

We begin by defining the group structure. Let X be a set and $\cdot$ be a binary operation on the elements of X. X paired with $\cdot$ and most generally denoted as $(X, \cdot)$ is called a *group* if the following axioms are satisfied[10]:

1. $\exists e \in X \ni \forall a \in X, ae = ea = a$ (Identity Existence),
2. $\forall a \in X, \exists a^{-1} \in X \ni aa^{-1} = a^{-1}a = e$ (Inverse Existence),
3. $\forall a, b \in X, ab \in X$ (Closure Under Binary Operation),
4. $\forall a, b, c \in X, a(bc) = (ab)c$ (Associative).

Notice that it is not generally the case that for $a, b \in X, ab = ba$. If a group does have this property, commutativity, it is called an *Abelian group*. What follows are a series of important examples and definitions to introduce the necessary components of group theory used in quantum computing theory.

Example. $(\mathbb{R}, +)$ *is a group. The identity* $e = 0 \in \mathbb{R}$. *For any real number* r, $r - r = r + (-r) = 0 = rr^{-1}$, *so there are inverses. Closure and association are trivial properties of the real numbers. Also, for all of the same reasons,* $(\mathbb{Z}, +)$ *is a group.*

Example. $(\mathbb{R}, \cdot)$ *is not a group. As* 0 *does not have a multiplicative inverse, this should not be surprising. I.e.* $1/0 = 0^{-1}$ *does not exist. However,* $(\mathbb{R} - \{0\}, \cdot)$ *and* $(\mathbb{R}_{>0}, \cdot)$ *are groups.*

Example. *Groups can be finite as well. One of the most accessible examples is the group* $(\mathbb{Z}_3, +)$, *where* $\mathbb{Z}_3$ *is the modular arithmetic set* $\{0, 1, 2\}$. *The operation* $+$ *could be more clearly defined thusly:* $+ : \mathbb{Z}_3 \times \mathbb{Z}_3 \to \mathbb{Z}_3, +(a, b) \equiv (a + b)(mod 3)$. *So in this group,* $1 + 2 = 0$. *It would be useful to introduce the notion of a Cayley table to show how all of the unique elements of the group act on each other. This is the cayley table*

[10]The binary operation is clearly omitted in general when discussing group theory. This is because there is only one operation in the group structure and it can be inferred from context.

Table 1. Cayley table for $\mathbb{Z}_3$

$+$	0	1	2
0	0	1	2
1	1^*	2	0
2	2	0	1

*for $\mathbb{Z}_3$. The operations read such that for the box marked with a * we show $1+0=1$, in that order. Since a group is not necessarily Abelian (commutative), the order is important to keep straight. However, since this group* is *Abelian, there is some extra symmetry to notice.*

Example. *Groups can describe more than just integers and real numbers, more than simple numerical structure. The definition of a group is both broad and abstract enough that it can be adapted to a variety of situations. Let* $GL_2(\mathbb{R}) = \{\begin{bmatrix} a & b \\ c & d \end{bmatrix} \mid a, b, c, d \in \mathbb{R}, \det \neq 0\}$*, which is called the 2-dimensional general linear group over* $\mathbb{R}$*. The operation of the group is matrix multiplication. Note that this is an example of a nonAbelian group.*

Example. *One of the most important uses of a group is to describe symmetries such as rotations. For objects like the square or the equilateral triangle, these groups are finite because there are a limited number of rotations and reflections that leave the shape unchanged. However, for spheres, or more generally vectors, there are an infinite number of such rotations. Objects with this kind of continuous symmetry are typically described by continuous groups known as Lie groups (pronounced like 'Lee'). A good example here is describing rotations on the unit sphere*[11] $S^2 = \{x \in \mathbb{R}^3 || x| = 1\}$*. The group of rotations (and only rotations) in three dimensions is the special orthogonal group* $SO(3) = \{X \in GL_3(\mathbb{R}) | X^\intercal X = I_3, \det(X) = 1\}$ *where* $GL_3(\mathbb{R})$ *is the group of nonsingular* 3×3 *matrices with real entries.*

Typically, when discussing an arbitrary group G, the operation is not written along with the set name or during computation. E.g. the group formed from set G with operation $\cdot$ will always be referred to as G and not $(G, \cdot)$, while for two elements $a, b \in G$, the operation between them will be written ab and not $a \cdot b$. Let G be a group and $H \subseteq G$. H is called a *subgroup* of G if H also satisfies the group axioims. Stated without proof, there is a theorem that simplifies the determination of whether a subset is a subgroup: $\forall a, b \in H, ab^{-1} \in H \Rightarrow H \leq G$ where $H \leq G$ is the common notation indicating subgroup status.

[11] Usage for the word sphere differs outside of mathematics. In vernacular use, the word sphere typically refers to the supposedly three-dimensional volume that looks like the earth. Throughout this work, and generally in mathematics, the word sphere usually has a dimension specified, like the 2-sphere defined here, and can be thought of as the shell surrounding a ball in one dimension higher. Note that we define the 2-sphere in terms of $\mathbb{R}^3$. This is because doing so is simple, however, as the name suggests, the 2-sphere can actually be described by only two numbers, and is fundamentally a two-dimensional object. Keep this in mind throughout.

Let G be a group. Let $Z(G) = \{z \in G \mid gz = zg \forall g \in G\}$. $Z(G)$ is called the *center* of G and contains the elements of G that commute with every element of G. Nontrivially, $Z(G) \leq G$. Every group at least has the trivial center $\{e\}$. Let G be a group and $S \subseteq G$ a subset of the elements of G. We call, for $g \in G$, $gS = \{gs \mid s \in S\}$ a *left coset* of G. The right coset is defined similarly. If G is Abelian, $gS = Sg \forall g \in G$. Let G be a group and $N \leq G$. The subgroup N is called *normal* if $\forall g \in G, gN = Ng$. Obviously, every subgroup of an Abelian group is normal. Normality is notated as $N \trianglelefteq G$. Let G and H be groups.

A function $\phi : G \to H$ is called a *homomorphism* if $\forall a, b \in G$, $\phi(a), \phi(b) \in H$ and $\phi(ab) = \phi(a)\phi(b)$, where the operation between a and b is from G and the operation between $\phi(a)$ and $\phi(b)$ is from H. This is called a structure preserving map and specifically a group homomorphism. Let G and H be groups. A function $\phi : G \to H$ is called a *group isomorphism* if ϕ is a bijective homomorphism. I.e.:

1. $\forall a, b \in G, \phi(ab) = \phi(a)\phi(b)$
2. $\phi(a) = \phi(b) \Rightarrow a = b \forall a, b \in G$
3. $\forall h \in H, \exists g \in G \ni \phi(g) = h$.

If there exists an isomorphism between G and H, the two groups are called isomorphic. A key property of both morphisms is that the identity of G maps to the identity of H. Also, the isomorphism class is notated $G \approx H \Leftrightarrow G \simeq H$. In addition, $|G| = |H|$, trivially. Essentially, groups that are isomorphic have the same structure and that structure is just expressed in different ways. Any fact about one group is true for every group it is isomorphic to.

Let G be a group and $N \trianglelefteq G$. We define $G/N = \{gN \mid g \in G\}$ and call it the *factor group* or *quotient group* with the composition operation. Namely, $\forall a, b \in G, (aN)(bN) = (ab)N$. $\forall n \in N, nN = N$, so we say that a quotient group absorbs elements. Let G be a group and $S \subseteq G \ni S = \{s_1, s_2, \cdots, s_n\}$. Let $\langle s_1, s_2, \cdots, s_n \rangle = \langle S \rangle$. If $\langle S \rangle = G$, we say that the elements of S generate G. Alternatively, one can start with a large group G and consider $\Gamma \leq G$, the subgroup generated by the elements in S. The meaning of $\langle s_1, s_2, \cdots, s_n \rangle$ is that we are interested in the subgroup made from strings of these elements and any other elements you can make with them. The information in this section and above can be distilled from [2], a popular textbook on group theory.

Quantitative Model for 1-Qubit Logic Gates

Here, we define the qubit as well as discuss the notation used in physics to label quantum states. For a quantum computing system utilizing qubits, each particle can be in one of two basis states when measured, $|0\rangle$ or $|1\rangle$. While evolving, each state $|\psi\rangle$ representing a qubit is in a linear combination of the basis states. So then $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $|\alpha|^2 + |\beta|^2 = 1$. This requirement guarantees the conservation of probability in quantum measurement. Note that here $\alpha, \beta \in \mathbb{C}$. Mathematically speaking, we consider $|0\rangle$ and $|1\rangle$ to be orthonormal basis vectors in $\mathbb{C}^2$ that are

represented by $\begin{pmatrix}1\\0\end{pmatrix}$ and $\begin{pmatrix}0\\1\end{pmatrix}$ respectively. It follows that $|\psi\rangle$ can be represented by the vector $\begin{pmatrix}\alpha\\\beta\end{pmatrix}$.

Recall from the qualitative introduction that logic gates act on computational states to return logical outputs. Thus, logic gates are represented mathematically by linear transformations. So, we want a mathematical object that takes $|\psi\rangle$ and maps it to the new state $|\psi'\rangle = \alpha'|0\rangle + \beta'|1\rangle$. By considering the vector representation of $|\psi\rangle$, it becomes clear that 1-qubit quantum logic gates can be represented by 2×2 matrices that preserve the property that $|\alpha|^2 + |\beta|^2 = 1$. Most succinctly, we can describe such matrices as a group.

The most general group describing matrices with complex entries is

$$GL_2(\mathbb{C}) = \{A = \begin{pmatrix}a & b\\c & d\end{pmatrix} | a, b, c, d \in \mathbb{C}, \det(A) \neq 0\} \tag{2}$$

the general linear group of 2×2 matrices with complex entries. The operation for matrix groups is typically matrix multiplication, and there is then a requirement that the matrices are nonsingular. So then, the group of matrices that most generally describes the set of all 1-qubit quantum gates is $U(2) = \{P \in GL_2(\mathbb{C}) | P^\dagger P = I_2\}$. However, we find that due to the nature of quantum mechanical states, we actually only need consider the equivalent of direction for these matrices, and not worry about the 'length' of the matrix. So then, if we consider $U(2)$ under the mapping $X \in U(2)$, $X \sim \frac{X}{\sqrt{\det(X)}}$, which takes the determinant to 1. Thus, we are concerned with $SU(2) = \{X \in U(2) | \det(X) = 1\}$ which is called the special unitary group. Finally, in the context of quantum computing, we also find global phase to not have any practical effect on gates, so we can work with the additional mapping wherein X and $-X$ are considered equivalent. Since the center of $SU(2)$ is $Z(SU(2)) = \{I_2, -I_2\}$, we define $PSU(2) = SU(2)/Z(SU(2))$ to be the projective special unitary group. Based on this group theoretic underpinning, we can define additional structures that will ultimately be used to describe the covering properties of universal subgroups of $SU(2)$ and $PSU(2)$.

Topology and the Topological Structure of $SU(2)$

We begin by defining the basic structures used in topology.

Let X be a set and τ be a collection of subsets of X. A *topological space* follows the following axioms:

1. $\emptyset, X \in \tau$
2. Any union of elements of τ are in τ.
3. The intersection of a finite number of elements of τ is in τ

where τ is called a topology. Let $S \subset X$. A point $x \in X$ is a *limit point* if every neighborhood of x has a nonempty intersection with S. Here, the neighborhood of a

point can be thought of as an open set containing points that are arbitrarily close to the point in question.

A subset A of a topological space X is called *dense* if $\forall x \in X$:

1. $x \in A$ or
2. x is a limit point of A.

Let G be a topological group, i.e. a group that is also a topological space, and $S \leq G$, such that $S = \{s_1, s_2, \cdots, s_n\}$. Let $\Gamma = \langle S \rangle$. S is called *universal* if Γ is dense in G. A nonempty subset $S \leq G$ is *symmetric* if $S = S^{-1}$ where $S^{-1} = \{s^{-1} | s \in S\}$. In other words, a subset of a topological space is dense if its points are arbitarily close to any points in the space. For example, the rationals $\mathbb{Q}$ are dense in $\mathbb{R}$. In the context of topological groups, a subset S is called universal if the subgroup it generates is dense. If S contains the inverse of every point, it is symmetric. The properties of universal and symmetric are required by the Solovay-Kitaev algorithm for approximating arbitrary unitaries in $SU(2)$. With regards to the research directly achieved in this work, these two properties are most important.

Similar to homomorphism and isomorphism over groups, topological structure allows for the possibility of diffeomorphisms. Consider a differentiable manifold to be a topological space that locally resembles the reals enough for calculus to be defined. All of our topological groups and spaces will be differentiable manifolds, specifically $SU(2)$, $PSU(2)$, $SO(3)$, S^2, and S^3. A diffeomorphism is a bijective function between differentiable manifolds with a continuous inverse. Similar to group isomorphism, diffeomorphism implies that two spaces share the same structure in some way. With that in mind, one can show that $SU(2)$ is a double cover of S^2 and $SO(3)$ and is diffeomorphic to S^3. It is also the case that $PSU(2)$ is homomorphic to $SO(3)$, which provides a connection to S^2. These connections will be used to describe the covering efficiency of certain choices of universal gate set.

Metric Spaces and the Metric Structure of $SU(2)$

A *metric* or distance function on a set X is defined as $d : X \times X \to \mathbb{R}_{\geq 0}$ satisfying $\forall x, y, z \in X$:

1. $d(x, y) \geq 0$
2. $d(x, y) = 0 \Leftrightarrow x = y$
3. $d(x, y) = d(y, x)$
4. $d(x, z) \leq d(x, y) + d(y, z)$.

The trace of a matrix, defined early, is interestingly invariant under a change in basis, and is the sum of the eigenvalues. Because of these properties, we can use the trace

to define a common invariant metric on $SU(2)$. Specifically, for $X, Y \in SU(2)$, the metric we use is given by

$$d(X,Y) = \sqrt{1 - \frac{|\mathrm{Tr}(X^\dagger Y)|}{2}}. \tag{3}$$

This metric is invariant, meaning that $\forall h \in SU(2)$ and $\forall X, Y \in SU(2)$, $d(hX, hY) = d(Xh, Yh) = d(X, Y)$.

We also would like to define the notion of a set of points a fixed distance from some $X \in SU(2)$. In mathematics, this is what is meant by a *ball*. Formally, for some group G with metric d, the ball of radius ε centered at $\gamma \in G$ is

$$B_G(\gamma, \varepsilon) = \{x \in G | d(x, \gamma) < \varepsilon\}.$$

Measure Theory and the Haar Measure

For the sake of completeness, it is necessary to briefly introduce the notion of a measure as well as its purpose and a type of measure used in describing the efficiency of a universal gate set. Let X be a set and $\mathcal{P}(X)$ be the power set of X. Then $\Sigma \subseteq \mathcal{P}(X)$ is called a *σ-algebra* if it satisfies the following:

1. $X \in \Sigma$
2. $\forall A \in \Sigma, X - A \in \Sigma$
3. $\forall A_1, A_2, \cdots \in \Sigma, A_1 \cup A_2 \cup \cdots \in \Sigma$

The elements of a σ-algebra are called measurable sets. In a topological space X, a *Borel set* is any set that can be formed from open sets using countable unions, countable intersections, and relative complements. The collection of all Borel sets on X forms a σ-algebra called the *Borel algebra*. Further, the Borel algebra is the smallest σ-algebra containing all open sets.

In a metric space (X, d), *compactness* is equivalent to the statement that every infinite subset of X has at least one limit point in X. Similarly, a *compact* group is a group whose topology is compact. The groups $SU(2)$ and $PSU(2)$ are compact. Intuitively a measure can be thought of as a function that returns the 'size' of a set. The origins of measure theory come from real analysis. The usual measure of an open interval is simply the length; following on, one finds that measure is fundamentally a generalization of integration. The usual measure on the reals is the Lebesgue measure, defined by Henri Lebesgue, wherein the measure of finite and countably infinite sets is zero. However, the Lebesgue measure is defined specifically for $\mathbb{R}^n$.

Thus, since we are interested in using the concept of measure, we require a generalization for any compact group, which is the Haar measure. Let G be a compact group. A normalized *Haar measure* $\mu : \Sigma \to \mathbb{R}_{\geq 0}$ on G where Σ is the Borel algebra of G satisfies:

1. $\mu(G) = 1$,

2. $\forall x \in G$ and $S \in \Sigma, \mu(xS) = \mu(S)$.

We will use a Haar measure on $SU(2)$ to determine how well a universal gate set generates a covering of $SU(2)$. Details for this will be provided shortly. The information in the sections on qubits, topology, metric structure, and measure theory is expanded from a discussion in [3]

3 Approximation of Quantum Logic Gates

With an appropriate mathematical background in place, it is possible to describe the difficulties caused by the plethora of potential 1-qubit quantum gates and beyond. Classical algorithms, implemented using Boolean logic gates, can be universally achieved using only the NAND gate, which could be described as a 2-bit classical gate. Because quantum computing has such a different underlying mathematical structure, the intricacies of implementing an arbitrary algorithm are much more complex. The set $SU(2)$ describing all potential 1-qubit gates is uncountably infinite. The ramifications of that are that naively, an arbitrary algorithm would require a unique set of gates to perform every step, a practical impossibility. Implementation would require the manufacturing of inordinate kinds of quantum gates. Considering that modern computing using Boolean gates has only been achieved thanks to the economies of scale introduced by using a minimal number of unique gates, it is clear that quantum computing in general is untenable without a way to overcome this obstacle. What is desired is a finite set of 1-qubit quantum gates that can be connected to approximate an arbitrary 1-qubit quantum gate.

However, as hinted at earlier, uncountably infinite compact topological groups such as $SU(2)$ permit the existence of dense subgroups, and such dense subgroups could potentially be generated by a finite set of elements. This turns out to be the case for $SU(2)$; there are many such universal sets of 1-qubit quantum gates that generate a dense subgroup of $SU(2)$. This only solves one of the practical problems posed. The existence of universal subsets of $SU(2)$ does not necessarily guarantee that the approximations are efficient. It is perhaps thinkable that there are universal subsets that would require very long strings of gates in order to approximate certain elements of $SU(2)$. What is realistically desired is a set that covers $SU(2)$ to within an arbitary accuracy with relatively short chains of gates. This could be termed to be particularly efficient choices of universal subset. We must formalize these ideas to create an exact enough mathematical description of efficient covering that we can determine which gate sets are 'good'.

We begin by noting an interesting property of elements of $SU(2)$. Recall that $SU(2) = \{X \in U(2) | \det(X) = 1\}$. Due to the requirement that $X^\dagger X = I_2$, we have that

$$X = \begin{pmatrix} a & b \\ -\overline{b} & \overline{a} \end{pmatrix}.$$

Then it must be the case that $a\overline{a} + b\overline{b} = 1$ since $\det(X) = 1$. Since $a, b \in \mathbb{C}$ we can choose real numbers $x_1, x_2, x_3, x_4 \in \mathbb{R}$ such that $a = x_1 + x_2 i$ and $b = x_3 + x_4 i$. It then follows that the matrices in $SU(2)$ must have entries comprised of real numbers that

satisfy $x_1^2 + x_2^2 + x_3^2 + x_4^2 = 1$. In other words, all of the elements of $SU(2)$ correspond to points on the sphere S^3.

Let $S = \{s_1, s_2, \ldots, s_n\}$ be a universal, symmetric subset of $G = SU(2)$. Then let $\Gamma = \langle S \rangle$ be the dense subgroup of G generated by S. We define a weight function $w : S \to \mathbb{R}_{\geqslant 0}$ that assigns a 'cost' to each element of S. In practice, the cost represents the computational difficulty assigned to using each gate to approximate a different one. Typically, for any $s \in S$, we take $w(s) = 1$. Next we define a height function $h : \Gamma \to \mathbb{R}_{\geqslant 0}$ such that for $\gamma \in \Gamma$,

$$h(\gamma) = \min\{\Sigma_{k=1}^{m} w(x_k) | \gamma = x_1 x_2 \ldots x_m, x_k \in S\}.$$

That is, h represents the combined weight of the most efficient approximation γ. We typically suppose that we utilize the most direct combination of gates to arrive at any point in Γ.

Using these functions, we can define a few sets that help summarize key ideas. First, we define a set of specific height to be

$$U(t) = \{\gamma \in \Gamma | h(\gamma) = t\}. \tag{4}$$

Similarly, we define the set of bounded height by

$$V(t) = \{\gamma \in \Gamma | h(\gamma) \leqslant t\} = \bigcup_{i=1}^{t} U(i). \tag{5}$$

Using these sets, it is simple to begin composing a notion of a set being particularly efficient. We would like to specify a fault-tolerance ε and consider the points in the set $V(t_\varepsilon)$, where t_ε is the height required so that

$$G \subset \bigcup_{\gamma \in V(t_\varepsilon)} B(\gamma, \varepsilon) \tag{6}$$

is satisfied. This in other words is the height required to generate a set of points that covers all of G to within the tolerance. For the purpose of this thesis, we are directly interested in considering the approximation of specific gates using different choices of universal gate set; however, considering the larger issue of approximating all of $SU(2)$ is of great importance to the field as a whole. The notation used here is derived from [3, 4, 5]. With this in mind, we introduce the Solovay-Kitaev theorem [6], which serves as the crux of this research.

Theorem (Solovay-Kitaev): Fix $\varepsilon > 0$ and S a universal, symmetric subset of $SU(2)$. If not otherwise specified, ε is small enough and $t > 0$. Then there exists a constant c such that for all $X \in SU(2)$, there exists γ a finite sequence of gates from S of length $O(\log^c(\frac{1}{\varepsilon}))$ approximating X within ε error[12]. Typically we can consider $c = 2.71$.

It is useful to also briefly discuss concepts that are used to describe the efficiency of universal gate sets. Note that the Solovay-Kitaev theorem guarantees the existence

[12] In other words, $d(\gamma, X) < \varepsilon$.

of an approximation, but it does not guarantee the length on its own. Further, it does not describe how quickly all of the points of $SU(2)$ become approximated to within ε. There are a few common options used in the study of point distributions and covering properties. The first of these is the definition of covering radius:

$$K(S) \equiv \limsup_{\varepsilon \to 0} \frac{\log|V(t_\varepsilon)|}{\log\left(\frac{1}{\mu(B_G(\varepsilon))}\right)}. \tag{7}$$

The covering exponent of a universal subset is between 1 and ∞ where 1 is the most optimal efficiency. As far as is known, there are not any universal subsets that have optimal efficiency everywhere. The work in [7] describes an algorithm that works optimally well for a wide class of universal gate set in approximating a common type of unitary. The best known sets are called arithmetic golden gate sets and super golden gate sets, explored in [3, 4, 8, 9, 5]. The known efficiency for this class of sets is between $\frac{4}{3}$ and 2. Further, these gate sets can be shown to be optimally efficient almost everywhere. This means that all of points of $SU(2)$ except for up to a countably infinite set of points can be approximated optimally. Attempting to narrow the known bound efficiency for gate sets of this type has proven exceedingly difficult. Previous research on this specific topic was completed at the University of Michigan during 2017 and can be viewed here [3, 4]. This differs strongly from the research performed for this thesis. Namely, the research at Michigan concerned a theoretical question of showing fundamental improvements in the understanding of the asymptotic efficiency of certain classes of universal gate set, while this research concerns numerical experiments using the Solovay-Kitaev algorithm and other common universal gate sets. As concerns viewing these gate sets under the framework of the covering properties of points on the sphere, [10] is a nice introduction.

CHAPTER II

Method

Here, we describe the algorithm that is explicitly used to prove the Solovay-Kitaev theorem in [6], and how it will be used to judge the efficiency of relatively short sequences from very common universal subsets. Let S be a universal, symmetric subset of $SU(2)$ and U be an arbitrary gate in $SU(2)$ that is to be approximated. Then the algorithm proceeds roughly by running recursively over S to build an approximation to U. This will be discussed more thoroughly below, after describing the mathematics of a key step of the algorithm.

1 Group Factor Commutator

The Solovay-Kitaev algorithm relies on a particular decomposition of an element of $SU(2)$ referred to by Dawson and Nielsen as a balanced group commutator. Given a unitary U, then a group commutator is a set of two unitaries V and W such that $U = VWV^\dagger W^\dagger$. There are infinite choices of group commutator for a given unitary, so the goal is to determine one which satisfies the algorithm.

Consider U to be a rotation by some angle θ about some axis $\hat{n}$ on the Bloch sphere[1]. In [6], it is described that a solution to

$$\sin(\theta/2) = 2\sin^2(\phi/2)\sqrt{1-\sin^4(\phi/2)} \tag{8}$$

for ϕ, can be used to generate rotations that form a group commutator that is similar[2] to U. Let $V = R_x(\phi)$ and $W = R_y(\phi)$ where R_x and R_y are rotations on the Bloch sphere about the x and y axes, respectively[3]. Then $VWV^\dagger W^\dagger$ is similar to U. Let S

[1]The Bloch sphere is a particular way to represent qubit states up to global phase as points on or in the 2-sphere. In this representation, we consider quantum gates to be rotation matrices acting on Bloch sphere states.

[2]For the sake of relative brevity, similarity of matrices was not described in the introduction. Consider two matrices A and B. Then we say that A is similar to B if there exists some matrix S such that $A = SBS^{-1}$. It can be shown with relative ease that, among other properties, similar matrices have the same eigenvalues, which ultimately means that we can use them to map our initial group commutator to one that we want.

[3]In the interest of detail [1],

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

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

be the unitary matrix such that $U = SVWV^\dagger W^\dagger S^\dagger$. Finally, the matrices that form our balanced group commutator are $\tilde{V} = SVS^\dagger$ and $\tilde{W} = SWS^\dagger$. So then, we have a subalgorithm for decomposing U into $\tilde{V}\tilde{W}\tilde{V}^\dagger\tilde{W}^\dagger$. This is what was required.

2 The Solovay-Kitaev Algorithm

Borrowing the very simple pseudocode from [6], the Solovay-Kitaev algorithm can be expressed in just a few lines:

```
function Solovay-Kitaev(Gate U, depth n)
if (n == 0)
   return nearest approximation to U
else
   set U_{n-1} = Solovay-Kitaev(U, n-1)
   set V, W = Group-Factor-Decompose(U U_{n-1}^†)
   set V_{n-1} = Solovay-Kitaev(V, n-1)
   set W_{n-1} = Solovay-Kitaev(W, n-1)
   return U_n = V_{n-1} W_{n-1} V_{n-1}^† W_{n-1}^†
```

We now break this down stey by step. We immediately see that the algorithm does not specify either a gate length or a fault-tolerance. Instead, we specify a gate to be approximated and a depth of recursions. Then, iteration n gives an instruction set of tolerance ε_n where ε_n is a decreasing function of n. Of course, as was previously discussed, the difficulty in rigorously proving results on the long-term efficiency of each universal gate set is an inability to understand the underlying connection between ε_n and n.

The algorithm operates by working through each iteration recursively until $n = 0$ is reached. At this point the algorithm returns an initial approximation to U. This pivotal step presumes that some initial library has been built up out of the universal gate set being used to search through. In other words, at the base of the algorithm is a search through some $V(t)$ to find the best approximation relative to the metric function being used. Realistically, the $V(t)$ that is utilized depends on the amount of computer memory and processing power. Exponential increases in the computational cost of the search dramatically increase the amount of memory and time required to run the algorithm at the specified depth. In practice, t does not have to be very great in order to create a perfectly usable starting set.

From here, the algorithm builds up better approximations to U. The unitary $UU_{n-1}^\dagger$ gives a matrix that is within a distance ε_{n-1} of the identity[4]. We arrive at an approximation to $UU_{n-1}^\dagger$ by taking the group factor decomposition and then using the algorithm to approximate V and W to a depth $n-1$ using our prebuilt library of gates. It turns out that doing this actually leads to a better approximation to U given by $V_{n-1}W_{n-1}V_{n-1}^\dagger W_{n-1}^\dagger$.

[4] $UU_{n-1}^\dagger$ is sort of like the difference between U and the current best approximation U_{n-1}. Recall that $UU^\dagger$ is identically the identity, being that this is the defining property of unitary matrices.

3 Implementation of the Solovay-Kitaev Algorithm in Python

It was originally intended that this thesis include an original implementation of the Solovay-Kitaev algorithm in Python utilizing only [6] as a guide. This was partly intended as a self-pedagogical exercise and partly as a replication of the ease with which the authors describe such a program. As should be clear, the algorithm itself is very simple, consisting mostly of recursive calls on itself and library searches. The notable exception is the group-factor decomposition process described in the preceding section. This is also described rather briefly by Dawson and Nielsen. After initial difficulties in running an initial implementation of the algorithm, it was realized that this decomposition function was at fault. Once this was noted, attention was turned to dissecting this process and determining where it was failing. Considerable effort was put into consulting mathematics resources to piece together a practical calculation of each portion of the decomposition.

It became clear that finding θ such that U could be described as a rotation about some axis $\hat{n}$ on the Bloch sphere was a simple process. Subsequently determining ϕ from θ using 8 was an analytically solvable problem with a relatively simple solution. From there, V and W are determined simply using well-known formulas describing the most basic Bloch sphere rotations. This also seemed an unlikely place from which to derive a failure. However, further investigation at this stage revealed that the failure of the decomposition had already occurred. Simple analysis using the Python interpreter and Numpy revealed that $VWV^\dagger W^\dagger$ was not similar to U. Specifically, the eigenvalues of $VWV^\dagger W^\dagger$ differed from those of U. This is one of several properties that similar matrices must have in common. Since they were not similar, it was impossible to find the correct S such that $U = SVWV^\dagger W^\dagger S^\dagger$. We were ultimately not able to parse out where this process had failed, especially since each step seems very established mathematically. A substantial amount of time was spent attempting to determine how this had occurred. Though the eigenvalues differed, it was clear that they were related in very obvious ways. However, actually finding a mechanism responsible for this difference became too time-consuming for further study. Comparison with the established implementation in [11] was also unhelpful, as both programs appeared to be performing the decomposition in the same manner. Ultimately, the goal of writing an implementation was abandoned, and Dawson's implementation in C++ was adopted as a vehicle for performing the proposed research.

4 Using the Solovay-Kitaev Algorithm

Though the Solovay-Kitaev theorem guarantees the existence of approximations to arbitrary unitaries with elements from universal gate sets that converge relatively quickly and corresponds to a simple algorithm, that does not mean that the problem of approximation is anywhere close to settled. In reality, the Solovay-Kitaev algorithm is just an impressive first step, a general algorithm waiting to be improved upon. The review paper detailing the algorithm, [6], is somewhat old now, and the underlying theorem is older still. There have been many other algorithms developed for quantum approximation, some specialized for specific gate sets, others improving upon the

Solovay-Kitaev algorithm itself. Table 1 in [7] provides a nice summary of much of the recent work on more specific algorithms than the Solovay-Kitaev algorithm. Notice though that for approximating any element of $SU(2)$ using any available gate set, the Solovay-Kitaev algorithm stands alone. All of the other given algorithms are limited in some way.

The goal of this work is to use the Solovay-Kitaev algorithm in conjunction with several popular universal gate sets to test efficiency at low tolerance and short sequences, considering each gate used to have equal cost. Originally, this comparison would be accomplished by compiling approximations to commonly used quantum logic gates, such as those used in Shor's algorithm. However, upon further research and thought, this scale seemed to be too specialized and limited. While there are certainly several interesting algorithms that would be worthy candidates, it seemed perhaps more universally applicable to run this experiment over a much larger sample size of randomly generated unitary matrices. Then by considering the average sequence length and accuracy, a much better understanding of performance at this small scale would be obtained.

Performing this computational experiment required an implementation of the Solovay-Kitaev algorithm. Initially, I attempted to create my own implementation in Python. However, due to an as yet not understood difficulty in implementing the group commutator decomposition step, I ultimately utilized a modified version of Chris Dawson's original implementation of the algorithm [11]. This was possible because his code is a faithful representation of the algorithm that generates a library of gates $V(t)$ and runs the algorithm. I adapted his underlying work by adding in more gates to use for testing and altering the metric norm used to match that which is defined in [3, 4, 5] and the introduction of this work. It does bear noting that this implementation contains flaws that are characteristic of naively using the Solovay-Kitaev Algorithm. Principally, neither the algorithm or this implementation accounts for the possibility of each of the individual steps combining in a way that places a matrix and its inverse next to each other. However, as larger preprocessed libraries are used, the frequency of this occurring in practice diminishes substantially. In order to keep the compilation runtime appropriate, the depth of library generated at the beginning was limited to be similar for each gate set and the depth of recursion for running the Solovay-Kitaev algorithm was restricted to $n = 5$.

To address exact specifics, the tests were ran on an Early 2015 Apple MacBook Pro with 1867 MHz DDR3 RAM. The program in C++ from [11] was compiled in g++, a common compiler. The resulting data was processed in Python. The program was modified very little from its source code; no changes were made to the implementation of the algorithm. The only other changes were to add more quantum gates to the program than those originally included and swapping the default norm for the trace norm that is more commonly used for the $SU(2)$ metric space in the contemporary literature. These alterations are both minor and peripheral. Code already included was used to generate the random quantum logic gates for testing. The size of the initial library was altered so that the sequences generated were of the same order of magnitude.

CHAPTER III

Results

I begin this chapter by reiterating that the goal of the experiment was to determine, over a large number of randomly generated unitaries, which common universal gate set on average gave the shortest sequences with the most accurate approximations using the Solovay-Kitaev Algorithm. Perusing the literature, it was clear which universal gate sets were the most widely studied. Defined below, the sets used in this experiment were the H+T set, the Pauli+V set, the V basis, and the Clifford+T set.

The H+T set stands for Hadamard and T, where T is the common name given to the $\frac{\pi}{8}$ phase gate. The matrices for these gates are

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

$$T = \begin{pmatrix} e^{\frac{-i\pi}{8}} & 0 \\ 0 & e^{\frac{i\pi}{8}} \end{pmatrix}.$$

Many papers that involve the H+T set consider only the number of T gates that appear in a particular approximation; however, upon further examination, this does not ultimately alter the conclusion of this experiment. This and the Clifford+T gate sets are described in [1]. The Clifford+T set consists of the Clifford circuits H and S, which form a finite group, as well as T. We have that

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}.$$

The gate S is called the phase shift gate.

The Pauli+V set is a modification of the V basis, so both will be described at once. The V basis is discussed in [12, 5], while the Pauli+V set is further explored in [12, 3, 4]. The V basis consists of the elements A, B, and C defined below as well as their inverses. Here, X, Y, and Z are the Pauli matrices. The Pauli+V set contains the gates of the V basis as well as iX, iY, and iZ. The matrices for these gates are

$$A = \frac{1}{\sqrt{5}}(I + 2iX),$$

$$B = \frac{1}{\sqrt{5}}(I + 2iY),$$

$$C = \frac{1}{\sqrt{5}}(I + 2iZ),$$

where I is the identity matrix. The gates A, B, and C correspond to Bloch sphere rotations about the x, y, and z axes by the angle $\arccos(\frac{3}{5})$. Their covering properties have been well studied [5].

Each of these sets were entered into the Solovay-Kitaev implementation. A library was generated to varying depth depending on the number of base elements. For H+T, the only base elements are H, T, and $T^\dagger$, so the base library was generated to $V(14)$. Continuing on, Pauli+V was generated to $V(5)$, the V basis to $V(6)$, and the Clifford+T set to $V(9)$. Each of these generation depths was balanced with the number of different random gates that were approximated. Each universal gate set was used to approximate $10,000$ randomly generated unitary matrices to a depth of $n = 5$ of the Solovay-Kitaev algorithm. The underlying goal was to be cognizant of the early days of quantum computing, where each gate used in an approximation is quite costly to implement. Under these conditions, each gate set made an approximation with sequences of length the same order of magnitude. The averages over the $10,000$ trials are shown below in Table 2. Recall that accuracy approaching 0 is desirable because accuracy is measured as the trace norm distance between the final approximation and the approximated quantum gate. The improvements in accuracy shown are significant since they are accompanied by modest decreases in sequence length.

Table 2. Results for $10,000$ approximations using common gate sets.

Set	Length	Accuracy
Clifford+T	25575	0.04431
H+T	40405	0.004376
Pauli+V	15491	0.0002686
V	16403	0.0001465

By examining the table, one comes to the conclusion that the V basis and the Pauli+V set are distinctively in a league of their own. Both have similar sequence lengths and have accuracies at the same order of magnitude. Ultimately, this is not very surprising, especially comparing the two sets' asymptotic behavior. Though as discussed above this is not the case for this particular naive implementation of the Solovay-Kitaev algorithm, under an optimal expression of the approximation one finds that the elements iX, iY, or iZ appear at most once in a sequence from Pauli+V [3, 4]. It is then no surprise that Pauli+V and the V basis have similar approximation efficiency throughout. To be clear, both of these sets produced approximations that are shorter *and* better than the other two sets examined. On the contrary, H+T achieved an average accuracy that was only one order of magnitude above the V based sets, but the sequences produced were much longer. The sequences from Clifford+T are of moderate relative length, but delivered the worst average approximation. There are many algorithms based on Clifford+T that might work much better than Solovay-Kitaev [13, 14].

This is perhaps a somewhat surprising results. It has been shown that asymptotically, all of these gate sets should have similar efficiency [5, 3, 4]. It is possible that

the universal gate sets that consist of Hadamard and T exhibit more initial 'clumping' behavior in their distribution at small t (as in $V(t)$). Examining the sequence length and accuracy for individual runs using these gates exhibits a remarkable variance, of several orders of magnitude. Some gates would be approximated very poorly, at an accuracy ~ 0.1, and others very well, at an accuracy close to $\sim 10^{-5}$. It is of course also possible that a naive implementation of the Solovay-Kitaev algorithm that does not account for incidental adjacent inverses is the contributing factor.

Regardless, the ultimate result from this experiment is not surprising: the correct gate set to use depends strongly on what gate is being approximated. The underlying assumption guiding this work is that a manufacturer or scientist can only produce one kind of universal gate set at an appropriate fault tolerance and quantity for meaningful quantum computing. In this case, the V basis or the Pauli+V set are the clear winning choices on average in this admittedly limited experiment. However, if this is not a true objection, then the correct way to go about building an early quantum computer is to design it on a case by case basis. Or better, if it is not too difficult, the quantum computing researcher should manufacture exactly the gates to be used. It would not be incorrect to consider this study then in light of the very early, 2 kB quantum computer processor. For any larger array of qubits, it would be more pertinent to use a more powerful computer than a laptop to determine a better algorithm and perform a more exhaustive study of general approximation efficiency. It could also be pertinent at that point in development to consider using multiple gate compiling units that are built from two gate sets, where ideally the most common large gaps in one's covering are well-approximated by the other. This is mere speculation though. Only time will tell the ultimate use of work on general quantum gate compilation or the worthwhileness of the Solovay-Kitaev algorithm.

CHAPTER IV

Summary and Further Work

1 Summary

By examining an extensive foundation of mathematics, it is possible to rigorously discuss quantum computing, the evolution of quantum states, 1-qubit quantum gates, and the covering properties of universal, symmetric subsets of $SU(2)$. In doing so, we find that the asymptotic behavior of such universal gate sets is not well-known, and the proven bounds on such behavior are quite wide. However, we also discuss the Solovay-Kitaev theorem, which gurantees that one can use the dense subgroup generated by a universal gate set to approximate an arbitrary unitary in $SU(2)$. In particular, the theorem guarantees that such an approximation converges relatively quickly and describes a related algorithm that performs this approximation to an arbitrary unitary using an arbitrary universal gate set.

We determine that having only a loose theoretical understanding of the bounds on the efficiency of this approximation leaves room for one commonly studied gate set to perhaps be more efficient than another. To this end, we select several of the most commonly studied gate sets and determine which returns the shortest sequence of gates to the best approximation on average, when approximating a significant number of randomly generated quantum gates. More specifically, the sets H+T, Pauli+V, V, and Clifford+T were each used to approximate $10,000$ random unitaries to an accuracy and sequence length appropriate to early, general quantum computers. The result found that the Pauli+V set and V basis were the most effective. This result could be in some way flawed for several reasons, most of which concern the naive implementation of the Solovay-Kitaev algorithm used for the study, however, it remains an interesting and striking result.

2 Further Work

There are myriad other directions that work of this type can be taken. Of particular interest to myself, having primarily theoretical experience in this area, is a stronger result on the bounds of $K(S)$ for a given universal gate set S. It has proven incredibly difficult for even the most accomplished and experienced mathematicians to improve on what is now a decades old result on the covering exponent bounds for the V basis gate set (and extrapolated to be true for all so-called Golden Gate sets). I would speculate that a new number theoretic tool might need to be developed before an improved upper bound is known. The current result was proven by Sarnak using the method of Hecke operators [5]. Some progress has been made retooling the problem in terms of optimal strong approximation and using the Kloosterman circle

method[1][9]. However, this in itself is not a true improvement because the notion that the sets in question have the optimal strong approximation property is a conjecture. So ultimately, there has been little to no real progress on this problem in several decades. See [3, 4] for a recent summary as well as a small conjecture on the subject.

Another related area of research regards the properties of a particular class of universal gate sets known as golden gates. This type of set is explored in [3, 4, 8, 9, 5]. The golden gates are notable because the mathematical graph formed from them have particularly nice navigation properties on their interiors. Of the gates studied in this experiment, the Pauli+V and V sets are examples of golden gate sets. There are also gate sets known as super golden gates that have been explored in [8]. These sets have similarly nice navigation properties at their edges. It is entirely possible that gates of these types have different asymptotic covering properties than their non-golden counterparts. If this is the case, then it could explain why the sets using the V gates performed better in this experiment. It will be exciting to see if focusing on these particular gate sets will lead to any improvements on the known bounds on covering exponent.

With regards to the work specifically completed in this project, many improvements could be made to the implementation of the Solovay-Kitaev algorithm being used. Most importantly, it would be more accurate if it deleted adjacent inverses that occasionally arise. This would most likely be simple to implement at the end of the algorithm, once a result is achieved. For the Pauli+V set, it would also be pertinent to specialize the algorithm so that it only applies one of iX, iY, or iZ in the sequence. This would reduce the length of the sequence delivered without affecting the accuracy. It would also be interesting to expand all of the parameters being used to push the limits of accuracy for the algorithm. With more processing power and greater memory, one could very easily increase the size of the library stored at the beginning of the algorithm and the depth of recursion for running the algorithm. It would also be interesting to explore other universal gate sets. From an academic perspective, there are gate sets in [8] with many elements that would be interesting to explore with the algorithm. On the other hand, it would be interesting to see if there are any other small, universal gate sets that are even better suited for general approximation than the V basis. Further, it would be interesting to push the boundary on using the Solovay-Kitaev compilation algorithm as opposed to one of the quicker but more limited algorithms developed since its initial publication and subsequent revitalization [7]. There are also improvements to the basic Solovay-Kitaev algorithm that could be implemented using more interesting data structures to provide a significant boost in efficiency without sacrificing approximation quality.

[1]The Kloosterman circle method itself a lower dimensional extension of the Hardy-Littlewood circle method.

Bibliography

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, UK, 2010), 2nd ed.

[2] J. A. Gallian, *Contemporary Abstract Algebra* (Brooks Cole, 2016), 9th ed.

[3] S. B. Damelin, Q. Liang, and B. A. W. Mode, "On Golden Gates and Discrepancy," arXiv:1506.05785 (2017).

[4] S. B. Damelin and B. A. W. Mode, "A Note on a Quantitative Form of the Solovay-Kitaev Theorem," arXiv:1709.03007 (2017).

[5] P. Sarnak, "Letter to Scott Aaronson and Andy Pollington on the Solovay-Kitaev Theorem and Golden Gates," (2015). `http://publications.ias.edu/sarnak/paper/2637`.

[6] C. M. Dawson and M. A. Nielsen, "The Solovay-Kitaev Algorithm," Quantum Information and Computation **6**, 81–95 (2006).

[7] V. Kliuchnikov, A. Bocharov, M. Roetteler, and J. Yard, "A Framework for Approximating Qubit Unitaries," arXiv preprint arXiv:1510.03888 (2015).

[8] O. Parzanchevski and P. Sarnak, "Super-Golden-Gates for $PU(2)$," arXiv preprint arXiv:1704.02106 (2017).

[9] N. T. Sardari, "Optimal Strong Approximation for Quadratic Forms," arXiv preprint arXiv:1510.00462 (2017).

[10] J. Bourgain, P. Sarnak, and Z. Rudnick, "Local statistics of lattice points on the sphere," arXiv preprint arXiv:1204.0134 (2012).

[11] C. Dawson, "Solovay-Kitaev Algorithm," `https://github.com/cmdawson/sk`.

[12] A. Bocharov, Y. Gurevich, and S. K., "Efficient decomposition of single-qubit gates into V basis circuits," Phys. Rev. A **88** (2013).

[13] N. Ross and P. Selinger, "Optimal ancilla-free Clifford+T approximation of z-rotations," Quantum Information and Computation **16**, 901–953 (2016).

[14] P. Selinger, "Generators and relations for n-qubit Clifford operators," Logical Methods in Computer Science **2** (2015).