QUANTUM CIRCUIT SYNTHESIS USING SOLOVAY-KITAEV ALGORITHM AND OPTIMIZATION TECHNIQUES

by

OLA AL-TA'ANI

B.S., Mutah University, Jordan, 2008
M.S., Kansas State University, USA, 2012

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Electrical and Computer Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2015

Abstract

Quantum circuit synthesis is one of the major areas of current research in the field of quantum computing. Analogous to its Boolean counterpart, the task involves constructing arbitrary quantum gates using only those available within a small set of universal gates that can be realized physically. However, unlike the latter, there are an infinite number of single qubit quantum gates, all of which constitute the special unitary group SU(2).

Realizing any given single qubit gate using a given universal gate family is a complex task. Although gates can be synthesized to arbitrary degree of precision as long as the set of finite strings of the gate family is a dense subset of SU(2), it is desirable to accomplish the highest level of precision using only the minimum number of universal gates within the string approximation.

Almost all algorithms that have been proposed for this purpose are based on the Solovay-Kitaev algorithm. The crux of the Solovay-Kitaev algorithm is the use of a procedure to decompose a given quantum gate into a pair of group commutators with the pair being synthesized separately. The Solovay-Kitaev algorithm involves group commutator decomposition in a recursive manner, with a direct approximation of a gate into a string of universal gates being performed only at the last level, i.e. in the leaf nodes of the search tree representing the execution of the Solovay-Kitaev algorithm.

The main contribution of this research is in integrating conventional optimization procedures within the Solovay-Kitaev algorithm. Two specific directions of research have been studied. Firstly, optimization is incorporated within the group commutator decomposition, so that a more optimal pair of group commutators are obtained. As the degree of precision of the synthesized gate is explicitly minimized by means of this optimization procedure, the enhanced algorithm allows for more accurate quantum gates to be synthesized than what the original Solovay-Kitaev algorithm achieves. Simulation results with random gates indicate that the obtained accuracy is an order of magnitude better than before.

Two versions of the new algorithm are examined, with the optimization in the first version being invoked only at the bottom level of Solovay-Kitaev algorithm and when carried out across all levels of the search tree in the next. Extensive simulations show that the second version yields better results despite equivalent computation times. Theoretical analysis of the proposed algorithm is able to

provide a more formal, quantitative explanation underlying the experimentally observed phenomena.

The other direction of investigation of this research involves formulating the group commutator decomposition in the form of bi-criteria optimization. This phase of research relaxed the equality constraint in the previous approach and with relaxation, a bi-criteria optimization is proposed. This optimization algorithm is new and has been devised primarily when the objective needs to be relaxed in different stages. This bi-criteria approach is able to provide comparably accurate synthesis as the previous approach.

QUANTUM CIRCUIT SYNTHESIS USING SOLOVAY-KITAEV ALGORITHM AND OPTIMIZATION TECHNIQUES

by

OLA AL-TA'ANI

B.S., Mutah University, Jordan, 2008
M.S., Kansas State University, USA, 2012

A DISSERTATION

submitted in partial fulfillment of the requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Electrical and Computer Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2015

Approved by:

Major Professor
SANJOY DAS

Abstract

Quantum circuit synthesis is one of the major areas of current research in the field of quantum computing. Analogous to its Boolean counterpart, the task involves constructing arbitrary quantum gates using only those available within a small set of universal gates that can be realized physically. However, unlike the latter, there are an infinite number of single qubit quantum gates, all of which constitute the special unitary group SU(2).

Realizing any given single qubit gate using a given universal gate family is a complex task. Although gates can be synthesized to arbitrary degree of precision as long as the set of finite strings of the gate family is a dense subset of SU(2), it is desirable to accomplish the highest level of precision using only the minimum number of universal gates within the string approximation.

Almost all algorithms that have been proposed for this purpose are based on the Solovay-Kitaev algorithm. The crux of the Solovay-Kitaev algorithm is the use of a procedure to decompose a given quantum gate into a pair of group commutators with the pair being synthesized separately. The Solovay-Kitaev algorithm involves group commutator decomposition in a recursive manner, with a direct approximation of a gate into a string of universal gates being performed only at the last level, i.e. in the leaf nodes of the search tree representing the execution of the Solovay-Kitaev algorithm.

The main contribution of this research is in integrating conventional optimization procedures within the Solovay-Kitaev algorithm. Two specific directions of research have been studied. Firstly, optimization is incorporated within the group commutator decomposition, so that a more optimal pair of group commutators are obtained. As the degree of precision of the synthesized gate is explicitly minimized by means of this optimization procedure, the enhanced algorithm allows for more accurate quantum gates to be synthesized than what the original Solovay-Kitaev algorithm achieves. Simulation results with random gates indicate that the obtained accuracy is an order of magnitude better than before.

Two versions of the new algorithm are examined, with the optimization in the first version being invoked only at the bottom level of Solovay-Kitaev algorithm and when carried out across all levels of the search tree in the next. Extensive simulations show that the second version yields better results despite equivalent computation times. Theoretical analysis of the proposed algorithm is able to

provide a more formal, quantitative explanation underlying the experimentally observed phenomena.

The other direction of investigation of this research involves formulating the group commutator decomposition in the form of bi-criteria optimization. This phase of research relaxed the equality constraint in the previous approach and with relaxation, a bi-criteria optimization is proposed. This optimization algorithm is new and has been devised primarily when the objective needs to be relaxed in different stages. This bi-criteria approach is able to provide comparably accurate synthesis as the previous approach.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I owe my gratitude to all wonderful people whom I have been blessed by. Who have made this dream come true and helped me while I pursued my graduate studies. First and foremost, I would like to express my sincere gratitude to my major professor, Dr. Sanjoy Das, for his guidance, encouragement, understanding, and most importantly, his friendship during my Ph.D study at Kansas State University. His expertise helped me expand my knowledge and simulated me to think more deeply about the world.

Many thanks to my committee members, Dr. Andrew Rys, Dr. Vinod Kumarappan, Dr. Doina Caragia for the valuable insights they have given to me. I would like to extend my thanks to Dr. Ivan Blank for serving as outside chair for my defense. I would like to acknowledge Dr. Andrew Rys for his support and encouragement from the first beginning. I would also like to thank my advisor during my Master study, Dr. Caterina Scoglio, for her guidance and support.

Most importantly, I would like to thank all my family and my family in-law for their continued support and endless love. Words are not capable of describing how grateful I am to them. I couldn't have made it this far in my academic career without each of them. Specifically, my deepest gratitude goes to my father, Dr. Ali Al-Ta'ani for being my role model and for his ultimate support. I am grateful as well to my mother, Faten Al-Tal, for always having faith in me, and encouraging me to pursue this advanced degree. I also thank my sisters and brothers, my father in-law, Mr. Mahmoud Gharaibeh, and all of my friends for their endless spiritual support, encouragement, trust, and love. Without all of you this would not be possible.

Last but not least, special thanks go to the person who supports me more than anyone else, my husband, Dr. Mohammed Gharaibeh. I cannot thank you enough for your unwavering love, unyielding devotion, and especially unconditional support and encouragement through this experience. I extend my thanks to my beloved children, Karam, Rose, and Arz, for being the joy of my life. I am thankful they took this journey with me.

Dedication

To my loved ones.

Chapter 1 - Introduction

In this chapter, we provide a brief overview of quantum circuit synthesis - the primary topic of interest in this dissertation. In section 1.1, quantum circuit model and the challenges associated with synthesizing quantum circuits are introduced. Prior efforts related to synthesizing quantum circuits are reviewed in section 1.2. Also, the drawbacks and limitations in this area are briefly discussed. Finally, in section 1.4, contributions to the area of quantum computer synthesis are outlined.

1.1 Background

Quantum computers upon realization can be able to perform large scale of computation several order of magnitude faster than their classical counterparts [Simon 1997, Knill 2010]. An example is the integer factorization problem [Shor 1994]. Quantum computers that would be able to solve in polynomial time several problems known to be NP- hard are currently being investigated [Nielsen and Chuang 2010, Hsieh and Le Gall 2011]. Several models of quantum computing have been proposed to implement such tasks with quantum circuits perhaps being the most widely studied one.

Unlike digital circuit where each operation carried out is typically accompanied by a loss of information, quantum circuits are logically reversible in the sense that it is possible to infer the exact inputs merely by observing the outputs [Bennett 2003]. As a direct implication of logical reversibility, the number of outputs in any quantum circuit should be equal to the number of inputs. Hence, numerous amounts of number-crunching can be accomplished theoretically with zero power consumption [Landauer 1992, De Vos 2011, Saeedi and Markov 2013].

The fundamental unit of information in quantum circuits is the *qubit* (Quantum bit). Quantum circuits are built using quantum gates as their building blocks. The operations performed on qubits by quantum gates must be unitary and hence quantum gates have same number of inputs and outputs [Barenco et al. 1995]. The simplest quantum gates are single qubit gates with one quantum input and one quantum output. In contrast of classical Boolean logic case where we have only one single logic gate (NOT gate), in quantum we have infinite number of single quantum gates [Nielsen and Chuang 2010].

Unfortunately, processing data at the quantum level cannot be easily obtained at the desired level of precision. In other words, quantum computing tends to be noisy [Ladd et al. 2010, Jones et al. 2012]. In order to overcome this limitation, fault tolerant quantum computing is required [Ladd et al. 2010, Van Meter 2010, Dyakonov 2012]. Quantum circuits must exhibit a high level of fault tolerance. This restricts the feasibility of physically realizing quantum gates [Lin et al. 2013]. There exit universal gate families which serve as the basic components upon which to build other quantum gates [Nielson and Chuang 2010]. This situation is analogous to classical Boolean logic where {AND, OR, NOT} is a universal gate set and {NAND} is another set. There exist universal gate setss even for single qubit gates. The simplest one is perhaps the set of Hadamard and T gates [Nielson and Chuang 2010].

Quantum algorithms are realized by quantum circuits [Pittenger 1999]. The design of quantum algorithms requires synthesizing quantum circuits. In other words, decomposing quantum gates into a sequence of gates taken from a *universal gate set*. In this context, a universal gate set is a small set of gates, an arbitrary sequence of which can generate a dense subset of all gates [Chow et al. 2012, Van den Nest 2013]. Hence, synthesizing quantum circuits efficiently is of crucial importance in the design of quantum algorithms [Pérez-Delgado 2011]. But it is more complicated than the classical case since the set of all quantum gates that can be used to realize a certain circuit fault tolerantly is infinite [Shor 1996]. Though, decomposing an arbitrary quantum gate into an exact sequence of fault tolerant quantum operations is hard to obtain but succeeded in some cases [Amy et al. 2013, Fowler 2011]. More often, finding approximation to the highest level of precision is considered when exact equivalence is not possible.

There are many contributions in this direction [Han and Kim 2000, Dawson and Nielsen 2005, Kliuchnikov et al. 2013]. All approaches proposed try to approximate a desired gate as a sequence of a few directly realizable ones belong to any given universal gate set such that the distance between the desired gate and its approximation is the minimum. The notion of distance determines the error in approximation and quantifies the effectiveness of the approach used in approximation.

1.2 Prior Efforts and Motivation

While many methods proposed in the literature focus on single or multi qubit approximations, few are developed to carry optimal synthesis of quantum circuits [Aliferis 2006]. Optimal synthesis means exact decomposition of a target unitary with optimizing some particular cost [Hayes and Markov 2006]. In particular, circuit depth is defined here as the maximum length path in the circuit.

The depth of a circuit reflects the running time efficiency and error rates and hence is used as the main optimization cost [Arabzadeh et al. 2013]. Most of the methods proposed in the quantum circuit synthesis field usually target the universal gate set of Clifford gates combined with any single qubit gate [Jozsa 2006]. A common choice for this additional gate is the **T** gate or $\pi/8$-gate (see chapter 4). Implementing the non-Clifford gates in the universal gate set is considered the most expensive one. Hence, the number of non-Clifford gates in the approximated circuit is considered as another optimizing cost.

One of the algorithms proposed to find depth-optimal circuits is the meet-in-the-middle (MITM) algorithm [Amy et al. 2013]. It computes an exact circuit that implements a given quantum gate. The MITM algorithm applies to multiple-qubit circuits over a specific universal gate set. It can also be modified to other optimization costs such as the number of non-Clifford group gates. However, it is only applicable for small numbers of qubits. Another algorithm that implements only single-qubit gates optimally with respect to the number of **T** gates is presented [Bocharov 2012]. In this algorithm, a canonical form for single-qubit circuits is proposed to reduce any arbitrary single-qubit circuit into this form. As a consequence, the algorithm provides a speed up over brute force searching method. However, it applies only over the universal set consisting of the Hadamard and **T** gates.

In some cases exact decomposition of a unitary does not exist and hence optimal approximation is employed [Jaeger 2007]. A simple depth-optimal algorithm that handles the optimal approximation of arbitrary single qubit gates over any universal set is developed [Fowler 2011]. Instead of exhaustively search over sequences, a mechanism is obtained to skip over equivalent ones. However, searching over unique sequences still scales exponentially with the length of sequences. Another method for single qubit decomposition has been proposed [Bocharov et al. 2013]. However, it is only practical over the universal V set. Approximating arbitrary multi qubit unitaries has been studied as well [Kliuchnikov 2013]. This particular approach shows that any n qubit unitary can be approximated to arbitrary precision using Clifford and **T** gates by using two additional ancillae qubits.

Another direction of research concentrates on approximating single-qubit gates without incorporating ancillary qubits [Dawson and Nielsen 2006, Nagy 2006]. The Solovay-Kitaev algorithm is perhaps the standard method used for generating sufficiently good approximations of any desired unitary [Dawson and Nielsen 2006]. The original Solovay-Kitaev algorithm is a recursive algorithm

that decompose any quantum gate to an arbitrary degree of accuracy as a sequence of gates belong to any given universal gate set. While the algorithm gives efficient approximation sequences, it performs exhaustive search which increases the run time of the algorithm. As an attempt to go around this problem, a technique called geometric nearest-neighbor access trees (GNATs) is incorporated to replace exhaustive search [Brin 1995]. More recently, subsequent work to the Solovay-Kitaev algorithm uses GNAT recursively in the search space expansion (SSE) approach [Pham 2013]. These investigations have reduced the computation time significantly but increased the length of the possible approximation sequences.

Several attempts to apply genetic algorithms to synthesize quantum circuits have been investigated in the literature [Lukac and Perkowski 2002, Ruican et al. 2008, Zhang 2011, and Lukac et al. 2013]. This interest is motivated by the ability of genetic algorithms to evolve circuit approximation towards the best among all potential approximations. For instance, an algorithm that applies the genetic operators: mutation, crossover, and reproduction to the quantum circuit is presented [Han and Kim 2000]. The genetic algorithm searches for the best quantum gate composition. Then the developed circuit is evaluated based on a fitness function until the desired accuracy is obtained.

Although the Solovay-Kitaev algorithm is the mainstay of almost all quantum circuit synthesis algorithms, they share a few limitations. These limitations serve as the motivating factors behind the proposed research as outlined below:

- At each stage of recursion, the length of the approximating sequence grows exponentially with the stage of recursion. Although this intrinsic limitation will not be directly addressed by the proposed approach, due to the above two, the proposed algorithm reduces the precision error by a larger factor per recursive stage.
- Multiple factors contribute in reducing the error of approximation. However, they are not simultaneously considered for minimization. In the proposed approach, bi-criteria minimization method is applied to simultaneously minimize the factors considered.
- The tree-based search is too sparse, hence good solutions are not often considered. The proposed bi-criteria method would consider multiple approximations for the desired gate to be approximated instead of single one leading to a more thorough search algorithm.

1.3 Contributions

In this dissertation, conventional optimization procedures are integrated within the Solovay-Kitaev algorithm. As the degree of precision of the synthesized gate is explicitly minimized by means of this optimization procedure, the proposed algorithms allow for more accurate quantum gates to be synthesized than what the original Solovay-Kitaev algorithm achieves. Two specific directions of research have been studied,

- In the first phase, optimization has been modeled as a constraint optimization problem with equality constraints. Two versions of the new algorithm are proposed, with the optimization in the first version being invoked only at the bottom level of Solovay-Kitaev algorithm (Single-level enhancement algorithm) and when carried out across all levels of the search tree (Multi-level enhancement algorithm) in the other.

- In the second phase of research, the equality constraint in the previous approach is relaxed and with relaxation, a bi-criteria optimization is proposed. This optimization algorithm is new and has been devised primarily when the objective needs to be relaxed in different stages.

The rest of this thesis is organized as follows, chapter 2 outlines the basic concepts required to understand this work. This chapter begins with an introduction to quntum bits (*qubits*). Unlike a bit, which is its classical counterpart, a qubit displays quantum characteristics that are exploited to devise clever quantum circuits, some of whose classical counterparts do not exist. All the tools needed to study this behavior, are also introduced. Chapter 3 provides mathematical foundations on the graphical representation of qubits and their transformations. In chapter 4, the fundamental definitions necessary to understand the Solovay-Kitaev algorithm is presented along with a detailed description of the original algorithm. Another search method called geometric nearest-neighbor access trees that is incorporated in the proposed algorithms instead of the existing linear search method is also presented. Chapter 5 introduces the first of the two approaches proposed in this research. All algorithms and subroutines involved in this part of the study are detailed here, while the simulation results and theoretical analysis are addressed in chapter 6. The second approach is represented in chapter 7. The bi-criteria optimization formulation and algorithm are discussed thoroughly here. The simulation results are presented in chapter 8. Finally, a summary of contributions of this dissertation and future research directions are presented in chapter 9.

Chapter 2 - Physical Foundations

The basic unit of information in quantum circuits is the *qubit* (quantum bit). This chapter begins in section 2.1 with an introduction to qubits. Unlike a bit, which is its classical counterpart, a qubit displays quantum characteristics that are exploited to devise clever quantum circuits, some of whose classical counterparts do not exist. All the tools needed to study this behavior, are also introduced. In section 2.2, the operations conducted on quantum bits are represented by quantum gates. Generalization into multiple qubits is shown in section 2.3. The postulates of quantum mechanics are listed in section 2.4.

2.1 Quantum Bits

The fundamental unit of information in classical computations is the bit, which is binary valued, with the values being OFF and ON, FALSE and TRUE, or 0 and 1. Fortunately, subatomic particles too can take on two discrete states, allowing one to devise quantum counterparts of digital circuits. An electron's spin can be either UP or DOWN; the polarization of a single photon can be either horizontal or vertical. Using the Dirac notation that is commonly used in quantum literature, these two states are denoted as $|0\rangle$ and $|1\rangle$ (the operator $|\cdot\rangle$ is read as *ket*). The states $|0\rangle$ and $|1\rangle$ are called *pure states*. The quantum analogue of the bit is a qubit (quantum bit), which can acquire the two states, $|0\rangle$ and $|1\rangle$. However, unlike their classical counterparts, qubits can also acquire states that are linear combinations of the pure states. A qubit in such a combination of pure states is said to be in a state of superposition.

In general, a qubit $|\boldsymbol{\varphi}\rangle$ in an arbitrary state of superposition can be expressed as the linear combination of $|0\rangle$ and $|1\rangle$, as $|\boldsymbol{\varphi}\rangle = a|0\rangle + b|1\rangle$. Here a and b are complex quantities, with $a, b \in \mathbb{C}$, where $\mathbb{C}$ is the set of complex numbers. The pure states $|0\rangle$ and $|1\rangle$ are treated as basis vectors.

Sometimes it is convenient to express a qubit as a 2×1 column vector. In such a case the two pure states are represented in vector form in the following manner,

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}, \qquad |1\rangle = \begin{bmatrix}0\\1\end{bmatrix}.$$

This allows one to express the qubit $|\boldsymbol{\varphi}\rangle$ as follows,

$$|\boldsymbol{\varphi}\rangle = a|0\rangle + b|1\rangle = \begin{bmatrix}a\\b\end{bmatrix}.$$

Using Dirac's notation, the conjugate transpose of $|\boldsymbol{\varphi}\rangle$, which is a row vector, is denoted as $\langle\boldsymbol{\varphi}|$ ($\langle .|$ is read as *bra*). Thus:

$$\langle\boldsymbol{\varphi}| = a^*\langle 0| + b^*\langle 1| = [a^* \quad b^*].$$

The quantities a and b in $|\boldsymbol{\varphi}\rangle = a|0\rangle + b|1\rangle$ are the *amplitudes* of $|\boldsymbol{\varphi}\rangle$. The amplitudes of a qubit have a physical interpretation that has to do with the act of observation. When any quantum particle that exists as a superposition of the basis states, is observed by an external observer, or recorded by an instrument, its state *collapses* into one of the two basis states. Thus the act of observation causes the qubit $|\boldsymbol{\varphi}\rangle$ to collapse into either a $|0\rangle$ or a $|1\rangle$. Here $a^*a = |a|^2$ and $b^*b = |b|^2$ are the probabilities of observing the qubit $|\boldsymbol{\varphi}\rangle$ to be $|0\rangle$ or $|1\rangle$:

$$\mathrm{P}(|\boldsymbol{\varphi}\rangle = |0\rangle) = |a|^2,$$
$$\mathrm{P}(|\boldsymbol{\varphi}\rangle = |1\rangle) = |b|^2.$$

In the above, $\mathrm{P}(.)$ is the probability of the occurrence of its input argument. We will often express the probabilities more succinctly by dropping the left side of the argument when the meaning is implicit from the context of the discussion. Thus $\mathrm{P}(|\boldsymbol{\varphi}\rangle = |0\rangle) = \mathrm{P}(|0\rangle)$, whereas $\mathrm{P}(|\boldsymbol{\varphi}\rangle)$ would be interpreted to mean either $\mathrm{P}(|0\rangle)$ or $\mathrm{P}(|1\rangle)$.

There are only two possible outcomes of observing the qubit $|\boldsymbol{\varphi}\rangle$; either the latter is observed as a $|0\rangle$ or as a $|1\rangle$. Hence, the sum of their probabilities must add up to unity. In other words, $|a|^2 + |b|^2 = 1$. Noting that the inner product of the row and column vectors $\langle\boldsymbol{\varphi}||\boldsymbol{\varphi}\rangle$, which we denote simply as $\langle\boldsymbol{\varphi}|\boldsymbol{\varphi}\rangle$ is a scalar quantity, it is seen that:

$$\langle\boldsymbol{\varphi}||\boldsymbol{\varphi}\rangle = \langle\boldsymbol{\varphi}|\boldsymbol{\varphi}\rangle = [a^* \quad b^*]\begin{bmatrix} a \\ b \end{bmatrix} = a^*a + b^*b = |a|^2 + |b|^2 = 1.$$

We formalize the above discussion below.

Definition 2.1: (qubit) The state of a quantum particle can be represented as a single qubit $|\boldsymbol{\varphi}\rangle$, as,

$$|\boldsymbol{\varphi}\rangle = a|0\rangle + b|1\rangle = \begin{bmatrix} a \\ b \end{bmatrix},$$

where,

$$a, b \in \mathbb{C},$$
$$\langle\boldsymbol{\varphi}|\boldsymbol{\varphi}\rangle = |a|^2 + |b|^2 = 1.$$

2.2 Quantum Gates

Qubits can evolve dynamically from one state to another. The evolution of a quantum state of a closed quantum system can be described by Schrodinger's equation [Bohm and Loewe 1986]:

$$i\hbar\frac{d}{dt}|\boldsymbol{\varphi}_t\rangle = \mathbf{H}|\boldsymbol{\varphi}_t\rangle,$$

where ħ is Planck's constant and $\mathbf{H}$ is a 2×2 Hermitian operator ($\mathbf{H} = \mathbf{H}^{\dagger}$), called the Hamiltonian. The solution of the above differential equation is of the form,

$$|\boldsymbol{\varphi}_t\rangle = e^{-i\hbar\mathbf{H}t}|\boldsymbol{\varphi}_0\rangle.$$

These transformations on quantum states can be performed by quantum gates. Suppose the time is discretized into fixed intervals of τ, the transformation can be regarded as a linear operator applied to the qubit $|\boldsymbol{\varphi}_0\rangle$ to obtain $|\boldsymbol{\varphi}_\tau\rangle$ as shown,

$$|\boldsymbol{\varphi}_\tau\rangle = e^{-i\hbar\mathbf{H}\tau}|\boldsymbol{\varphi}_0\rangle.$$

The linear operator $e^{-i\hbar\mathbf{H}\tau}$ is a 2×2 matrix that is equivalent to the quantum gate $\mathbf{U}$. In other words,

$$\mathbf{U} = e^{-i\hbar\mathbf{H}\tau}.$$

Using the fact that $\mathbf{H}$ is Hermitian, it can be seen that $\mathbf{U}$ is a unitary matrix, since,

$$\begin{aligned}\mathbf{U}^{\dagger} &= e^{i\hbar\mathbf{H}^{\dagger}t},\\ &= e^{i\hbar\mathbf{H}t}.\end{aligned}$$

Post-multiplying the above with $\mathbf{U}$, we get,

$$\mathbf{U}^{\dagger}\mathbf{U} = \left(e^{-i\hbar\mathbf{H}\tau}\right)\left(e^{i\hbar\mathbf{H}t}\right) = \mathbf{I}.$$

Hence, $\mathbf{U}$ is a unitary matrix.

Definition 2.2: (gate) A quantum gate $\mathbf{U}$ is a linear operator acting upon a qubit that can be represented as a 2×2 unitary matrix.

There are several examples of quantum gates that are commonly used. For instance, the quantum NOT gate denoted as $\mathbf{U_{NOT}}$ is analogous to the classical NOT gate as it takes state $|0\rangle$ to state$|1\rangle$, and vice versa. It acts linearly on the superposition state, meaning that,

$$\mathbf{U_{NOT}}(a|0\rangle + b|1\rangle) = b|0\rangle + a|1\rangle.$$

The quantum gate $\mathbf{U_{NOT}}$ is represented in matrix form in the following manner,

$$\mathbf{U_{NOT}} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

Another important single qubit gate is the Hadamard gate denoted as $\mathbf{U_H}$. It creates an equal superposition of the pure states as shown below,

$$\mathbf{U_H}|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle),$$

$$\mathbf{U_H}|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

More succinctly, we can rewrite the above as,

$$\mathbf{U_H}|i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + (-1)^i|1\rangle), \quad i \in \{0,1\}.$$

The Hadamard gates can be represented in matrix form in the following manner,

$$\mathbf{U_H} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

If $\mathbf{U}_1$ and $\mathbf{U}_2$ are two different quantum gates that are cascaded in series (with $\mathbf{U}_1$ applied first), then they can be represented as a single gate $\mathbf{U}$ (see figure 2.1). The net transformation that the cascaded gates perform on a qubit $|\boldsymbol{\varphi}\rangle$ is expressed by the dot product shown,

$$\mathbf{U} = \mathbf{U}_2\mathbf{U}_1.$$

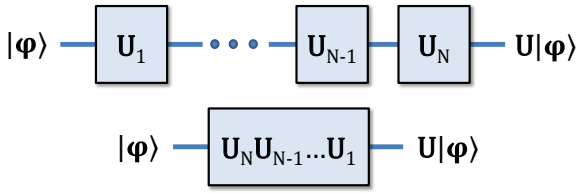


Figure 2.1 Quantum gates in series

This fact can be generalized to an arbitrary sequence $\mathbf{U}_i$, $i = 1 \ldots \mathrm{N}$ of N gates,

Observation 2.1: (Series combination of gates)

$$\mathbf{U} = \prod_{i=1}^{N} \mathbf{U}_i.$$

2.3 Multiple Qubits

As a first step we first consider the case of a 2-qubit register that consists of a pair of qubits. Just as a qubit can be observed in one of its two possible basis states $|0\rangle$ and $|1\rangle$, similarly a 2-qubit register can be observed in any of its own four possible basis states, $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$, where the left and right binary values represent the states of the pair of qubits that make up the register. Thus the register's state $|00\rangle$ corresponds to the situation when both the qubits are $|0\rangle$. The 2-qubit register's basis states can also be expressed as 4×1 column vectors:

$$|00\rangle = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \quad |01\rangle = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \quad |10\rangle = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \quad |11\rangle = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}.$$

We saw previously that a qubit's quantum state prior to observation may be a superposition of the two basis states. Likewise, a 2-qubit register can also be in a quantum state of superposition of the four basis states, as shown:

$$|\boldsymbol{\varphi}\rangle = a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle = \begin{bmatrix}a\\b\\c\\d\end{bmatrix}.$$

The four amplitudes, $a, b, c, d \in \mathbb{C}$ and define the probabilities of observing the quantum register in each basis state. That is:

$$\mathrm{P}(|\boldsymbol{\varphi}\rangle = |00\rangle) = |a|^2, \ \ \mathrm{P}(|\boldsymbol{\varphi}\rangle = |01\rangle) = |b|^2, \ \ \mathrm{P}(|\boldsymbol{\varphi}\rangle = |01\rangle) = |c|^2, \ \ \mathrm{P}(|\boldsymbol{\varphi}\rangle = |01\rangle) = |d|^2.$$

The law of total probability now warrants that the sum of all four probabilities must add up to unity such that,

$$|a|^2 + |b|^2 + |c|^2 + |d|^2 = 1.$$

Definition 2.3: (Quantum register) The quantum mechanical analogue of the digital register is the *quantum register*, which is simply an ordered sequence of qubits.

Systems with two qubits can be formed merely by arranging two separate qubits into a sequence. Consider two arbitrary qubits $|\boldsymbol{\alpha}\rangle$ and $|\boldsymbol{\beta}\rangle$.

$$|\boldsymbol{\alpha}\rangle = a_0|0\rangle + a_1|1\rangle = \begin{bmatrix}a_0\\a_1\end{bmatrix},$$

$$|\boldsymbol{\beta}\rangle = b_0|0\rangle + b_1|1\rangle = \begin{bmatrix}b_0\\b_1\end{bmatrix}.$$

When they are paired up into a 2-qubit system, the latter is denoted as $|\boldsymbol{\alpha\beta}\rangle$ and is obtained as the *tensor product* of $|\boldsymbol{\alpha}\rangle$ and $|\boldsymbol{\beta}\rangle$, which we denote as $|\boldsymbol{\alpha}\rangle\otimes|\boldsymbol{\beta}\rangle$. The operator $\otimes$ is used to indicate tensor multiplication. The product is computed in the following manner:

$$|\boldsymbol{\alpha\beta}\rangle = |\boldsymbol{\alpha}\rangle\otimes|\boldsymbol{\beta}\rangle = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \otimes \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} = \begin{bmatrix} a_0 b_0 \\ a_0 b_1 \\ a_1 b_0 \\ a_1 b_1 \end{bmatrix}.$$

In terms of Dirac notation this translates to:

$$\begin{aligned} |\boldsymbol{\alpha\beta}\rangle &= |\boldsymbol{\alpha}\rangle\otimes|\boldsymbol{\beta}\rangle \\ &= (a_0|0\rangle + a_1|1\rangle)\otimes(b_0|0\rangle + b_1|1\rangle) \\ &= a_0 b_0|00\rangle + a_0 b_1|01\rangle + a_1 b_0|10\rangle + a_1 b_1|11\rangle. \end{aligned}$$

In the previouse section, we saw that when quantum gates are connected in series, dot product of the matrix representations of the corresponding gates is evaluated in reverse order. On the other hand, the net effect of quantum gates connected in parallel (see figure 2.2) is computed by tensor product of their matrix representations.

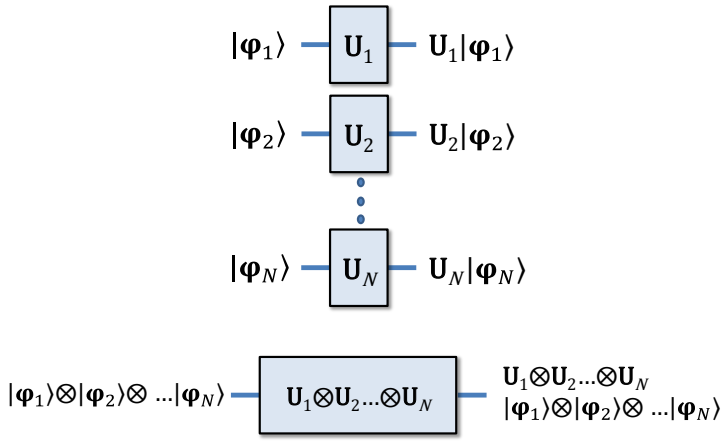


Figure 2.2 Quantum gates in parallel

Suppose $|\boldsymbol{\alpha}\rangle$ and $|\boldsymbol{\beta}\rangle$ are two qubits. Let us apply the 1-qubit gates $\mathbf{U_P}$ and $\mathbf{U_Q}$ separately to $|\boldsymbol{\alpha}\rangle$ and $|\boldsymbol{\beta}\rangle$ (cascaded in parallel). We obtain $\mathbf{U_P}|\boldsymbol{\alpha}\rangle$ and $\mathbf{U_Q}|\boldsymbol{\beta}\rangle$ at the output.

Equivalently, given the 2-qubit input $|\boldsymbol{\varphi}\rangle = |\boldsymbol{\alpha}\rangle\otimes|\boldsymbol{\beta}\rangle$, we obtain the 2-qubit output $|\boldsymbol{\psi}\rangle = \mathbf{U_P}|\mathbf{a}\rangle\otimes\mathbf{U_Q}|\boldsymbol{\beta}\rangle$. This is equivalent to saying that the input $|\boldsymbol{\varphi}\rangle$ to a 2-qubit gate $\mathbf{U_P}\otimes\mathbf{U_Q}$ produces a 2-qubit output $|\boldsymbol{\psi}\rangle$.

Observation 2.2: (Parallel combination of gates)

$$\mathbf{U_P}|\boldsymbol{\alpha}\rangle\otimes\mathbf{U_Q}|\boldsymbol{\beta}\rangle = \mathbf{U_P}\otimes\mathbf{U_Q}(|\boldsymbol{\alpha}\rangle\otimes|\boldsymbol{\beta}\rangle).$$

Check:

$$\mathbf{U_P}\otimes\mathbf{U_Q} = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix}\otimes\mathbf{U_Q} = \begin{bmatrix} p_{11}\mathbf{U_Q} & p_{12}\mathbf{U_Q} \\ p_{21}\mathbf{U_Q} & p_{22}\mathbf{U_Q} \end{bmatrix}.$$

$$|\boldsymbol{\alpha}\rangle\otimes|\boldsymbol{\beta}\rangle = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}\otimes|\boldsymbol{\beta}\rangle = \begin{bmatrix} a_1|\boldsymbol{\beta}\rangle \\ a_2|\boldsymbol{\beta}\rangle \end{bmatrix}.$$

Thus,

$$\begin{aligned}
&\mathbf{U_P}\otimes\mathbf{U_Q}(|\boldsymbol{\alpha}\rangle\otimes|\boldsymbol{\beta}\rangle) \\
&= \begin{bmatrix} p_{11}\mathbf{U_Q} & p_{12}\mathbf{U_Q} \\ p_{21}\mathbf{U_Q} & p_{22}\mathbf{U_Q} \end{bmatrix}\begin{bmatrix} a_1|\boldsymbol{\beta}\rangle \\ a_2|\boldsymbol{\beta}\rangle \end{bmatrix} \\
&= \begin{bmatrix} p_{11}a_1\mathbf{U_Q}|\boldsymbol{\beta}\rangle + p_{12}a_2\mathbf{U_Q}|\boldsymbol{\beta}\rangle \\ p_{21}a_1\mathbf{U_Q}|\boldsymbol{\beta}\rangle + p_{22}a_2\mathbf{U_Q}|\boldsymbol{\beta}\rangle \end{bmatrix} \\
&= \begin{bmatrix} p_{11}a_1 + p_{12}a_2 \\ p_{21}a_1 + p_{22}a_2 \end{bmatrix}\otimes\mathbf{U_Q}|\boldsymbol{\beta}\rangle \\
&= \begin{bmatrix} p_{11}a_1 + p_{12}a_2 \\ p_{21}a_1 + p_{22}a_2 \end{bmatrix}\otimes\mathbf{U_Q}|\boldsymbol{\beta}\rangle \\
&= \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix}\begin{bmatrix} a_1 \\ a_2 \end{bmatrix}\otimes\mathbf{U_Q}|\boldsymbol{\beta}\rangle \\
&= \mathbf{U_P}|\boldsymbol{\alpha}\rangle\otimes\mathbf{U_Q}|\boldsymbol{\beta}\rangle.
\end{aligned}$$

We saw that two qubits can readily be combined as a 2-qubit quantum register using tensor products. The natural question that arises is whether the converse of that statement necessarily holds, that is, can any 2-qubit register in a state of superposition always be expressed as the tensor product of two qubits? The surprising answer is in the negative!

When a 2-qubit register cannot be expressed as the tensor product of two qubits, the associated particles are said to be in a state of *entanglement*. Two particles are entangled thus, exhibit a curious phenomenon in that the outcome of observing any one of them influences the observation probabilities of the other. That is, their observations are not statistically independent events.

$$\mathrm{P}(|\boldsymbol{\beta}\rangle||\boldsymbol{\alpha}\rangle) \neq \mathrm{P}(|\boldsymbol{\beta}\rangle),\quad \mathrm{P}(|\boldsymbol{\alpha}\rangle||\boldsymbol{\beta}\rangle) \neq \mathrm{P}(|\boldsymbol{\alpha}\rangle).$$

As a matter of fact, the conditional dependences of observation probabilities is one way to define entanglement.

Entanglement is a surprising phenomenon that is not observed in the macroscopic world. Two particles that are in an entangled state can in fact influence each other instantaneously, no matter how large the spatial distance between them may be. Even when the entangled particles are many light years apart, the act of observing one particle of in an entangled pair in any basis state instantaneously alters the state of the unobserved one. This apparent departure from special relativity has given rise to much metaphysical speculations regarding the nature of space-time. From our standpoint of quantum computing quantum entanglement can be exploited in quantum teleportation and quantum communication. While these topics will be addressed in subsequent chapters, the remainder of this chapter is used to introduce qantum registers with over two qubits.

Definition 2.4: (Entanglement) When a 2-qubit register cannot be expressed as the tensor product of two qubits $|\boldsymbol{\alpha}\rangle$ and $|\boldsymbol{\beta}\rangle$, the associated particles are said to be in a state of entanglement. That is,

$$\mathrm{P}(|\boldsymbol{\beta}\rangle||\boldsymbol{\alpha}\rangle) \neq \mathrm{P}(|\boldsymbol{\beta}\rangle), \quad \mathrm{P}(|\boldsymbol{\alpha}\rangle||\boldsymbol{\beta}\rangle) \neq \mathrm{P}(|\boldsymbol{\alpha}\rangle).$$

As we saw that a sequence of two qubits constitutes a 2-qubit quantum register. By extension, we can treat an ordered sequence of n qubits as an n-qubit quantum register. Since each of the n qubits of the quantum register have two basis states $|0\rangle$ and $|1\rangle$, the n-qubit register can be observed in any of $N = 2^n$ possible basis states, $|00 \ldots 0\rangle$, $|00 \ldots 1\rangle$, until $|11, \ldots 1\rangle$. The situation is analogous to an n-bit classical system being able to store all possible binary numbers between 0 and $2^{\mathrm{n}} - 1$. The difference between the two lies in the fact that quantum systems can exist in a mixed states as well, which are the superposition of its basis states. In general, a state of n-qubits can be expressed in the following manner,

$$|\boldsymbol{\varphi}\rangle = \sum_{i=0}^{N-1} c_i |i\rangle.$$

In the above equation, each quantity $c_i \in \mathbb{C}$ is an amplitude. Where,

$$\sum_{i=0}^{N-1} |c_i|^2 = 1.$$

The probability of observing the n particles in a given state $|k\rangle$ is,

$$\mathrm{P}(|\boldsymbol{\varphi}\rangle = |k\rangle) = |c_k|^2.$$

Definition 2.5: (n-qubit quantum register) Any arbitrary n-qubit quantum register can be expressed as,

$$|\boldsymbol{\varphi}\rangle = \sum_{i=0}^{N-1} c_i |i\rangle,$$

where,

$$\sum_{i=0}^{N-1} |c_i|^2 = 1.$$

2.4 The Postulates of Quantum Mechanics

Quantum mechanics is the physics underlying quantum computing [Sakurai & Napolitano 2011]. Quantum computing uses quantum mechanical properties to efficiently perform computations. In order to overview the basics of quantum computing, a brief listing of the important properties of quantum mechanics is presented in this section. Quantum mechanics is characterized by four fundamental principles or so-called *postulates*,

Postulate 1: (The state space postulate) A closed quantum system is described by a unit vector in a complex inner product space in a Hilbert space.

Postulate 2: (The measurement postulate) After a system is measured from outside, its state appears to collapse to exactly match the measured outcome.

Postulate 3: (The time evolution postulate) The evolution of a closed quantum system is described by unitary transformation.

Postulate 4: (Composite systems) The state space of a composite system is the tensor product of the individual state spaces of the component systems.

Chapter 3– Mathematical Foundations

This chapter describes the relevant characteristics of a qubit using Bloch sphere representation in section 3.1. Any transformation performed on a qubit can be represented as a rotation on the Bloch sphere. The graphical representation of quantum states and their transformations is very important in order to understand quantum operations. In section 3.2, the Pauli and the standard rotation matrices are introduced. Arbitrary rotations are discussed in section 3.3. Section 3.4 represent important definitions of the special unitary group and the special orthogonal group.

3.1 Bloch Sphere

Any arbitrary single-qubit can be expressed geometrically in the following manner:

$$|\boldsymbol{\varphi}\rangle = e^{i\omega}\left(\cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle\right),$$

where ω, θ, ϕ are real numbers. In the factor $e^{i\omega}$, the quantity ω is referred to as the global phase of the qubit. As the probabilities of observing $|\boldsymbol{\varphi}\rangle$ in the basis states, $\mathrm{P}(|0\rangle) = \cos^2\frac{\theta}{2}$, $\mathrm{P}(|1\rangle) = \sin^2\frac{\theta}{2}$, do not incorporate the quantity ω, the global state has no observable effect. In other words, the overall phase of the state has no physical meaning ($|\boldsymbol{\varphi}\rangle$ and $e^{i\omega}|\boldsymbol{\varphi}\rangle$ represent the same physical state). Therefore, we can drop the factor involving global phase ω and further simplify it to the form:

$$|\boldsymbol{\varphi}\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle.$$

In vector form,

$$|\boldsymbol{\varphi}\rangle = \begin{bmatrix} \cos\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2} \end{bmatrix}.$$

Observation 3.1: Any qubit gate has a unique Bloch sphere representation,

$$|\boldsymbol{\varphi}\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle$$
$$= \begin{bmatrix} \cos\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2} \end{bmatrix}.$$

Expressing $|\varphi\rangle$ in the above manner allows us to visualize it as a point on the unit three-dimensional sphere as shown in figure 3.1. This is called the Bloch sphere representation with $\theta \in [0, \pi]$ being the

polar angle, i.e. the angle that the vector $|\boldsymbol{\varphi}\rangle$ forms with the z-axis, while $\phi \in [0,2\pi]$, the azimuthal angle, i.e. that between $|\boldsymbol{\varphi}\rangle$ and the xy-plane. A unit vector in spherical coordinates (called the Bloch vector) is,

$$\boldsymbol{\varphi} = \begin{bmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{bmatrix}.$$

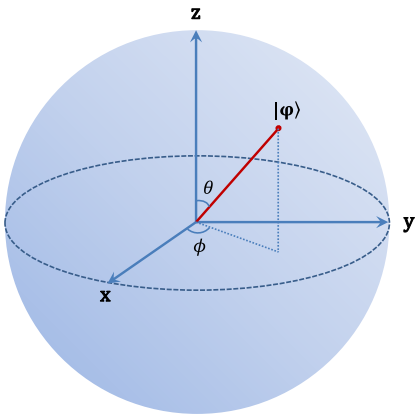


Figure 3.1 A qubit state $|\boldsymbol{\varphi}\rangle$ represented on the Bloch sphere

Bloch sphere representation is a very handy visualization tool in investigating the role of the quantum equivalents of Boolean gates. The vectors $|0\rangle$ and $|1\rangle$ are directed towards and against the z-axis, and can be regarded as the north and south poles of the Bloch sphere. Similarly, those of the form $\frac{1}{\sqrt{2}}\left(|0\rangle + e^{i\phi}|1\rangle\right)$ lie on the xy-plane and define its equator (see figure 3.2).

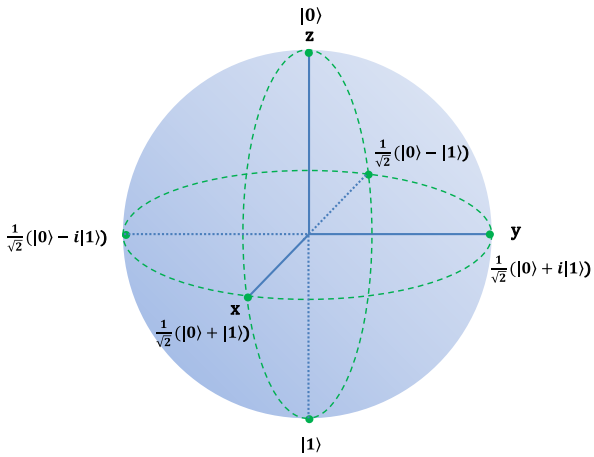


Figure 3.2 Examples of states on the Bloch sphere

3.2 Pauli Matrices

Any transformation performed on a qubit can be represented as a rotation on the Bloch sphere. This transformation can be obtained by a combination of one or more rotations around the x, y, and z axes. The rotations around these axes are the rotation matrices. These matrices can be represented in terms of a special set of matrices called the Pauli matrices.

Definition 3.2: (Pauli matrices) Pauli matrices are 2×2 matrices

$$\boldsymbol{\sigma}_{\mathrm{X}} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \boldsymbol{\sigma}_{\mathrm{Y}} = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad \boldsymbol{\sigma}_{\mathrm{Z}} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

Rotation operators rely on the properties of matrix exponentiation. For instance, rotation around the x axis by angle ψ, represented by $\mathbf{U_{ROTX}}(\psi)$ matrix, can be expressed in terms of Pauli matrix $\boldsymbol{\sigma}_{\mathrm{X}}$ in the following manner,

$$\mathbf{U_{ROTX}}(\psi) = e^{-i\frac{\psi}{2}\boldsymbol{\sigma}_{\mathrm{X}}}$$

where ψ is the angle of rotation.

In general, the exponential $e^{-i\frac{\psi}{2}\boldsymbol{\sigma}}$ of a matrix $\boldsymbol{\sigma}$ can be written as,

$$e^{-i\frac{\psi}{2}\boldsymbol{\sigma}} = \cos\frac{\psi}{2}\mathbf{I} - i\sin\frac{\psi}{2}\boldsymbol{\sigma},$$

where $\mathbf{I}$ is the identity matrix,

$$\mathbf{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Then the corresponding rotation matrix is,

$$\begin{aligned} \mathbf{U_{ROTX}}(\psi) &= \cos\frac{\psi}{2}\mathbf{I} - i\sin\frac{\psi}{2}\boldsymbol{\sigma}_{\mathrm{X}}, \\ &= \cos\frac{\psi}{2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - i\sin\frac{\psi}{2}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \\ &= \begin{bmatrix} \cos\frac{\psi}{2} & -i\sin\frac{\psi}{2} \\ -i\sin\frac{\psi}{2} & \cos\frac{\psi}{2} \end{bmatrix}. \end{aligned}$$

Similarly, rotations around the y and z axes can be represented in the following manner,

$$\mathbf{U_{ROTY}}(\psi) = e^{-i\frac{\psi}{2}\sigma_Y} = \begin{bmatrix} \cos\frac{\psi}{2} & \sin\frac{\psi}{2} \\ -\sin\frac{\psi}{2} & \cos\frac{\psi}{2} \end{bmatrix}.$$

$$\mathbf{U_{ROTZ}}(\psi) = e^{-i\frac{\psi}{2}\sigma_Z} = \begin{bmatrix} e^{i\frac{\psi}{2}} & 0 \\ 0 & e^{-i\frac{\psi}{2}} \end{bmatrix}.$$

Definition 3.3: (Rotations) Rotations around the x, y, and z axes can be represented by 2×2 unitary matrices,

$$\mathbf{U_{ROTX}}(\psi) = \begin{bmatrix} \cos\frac{\psi}{2} & -i\sin\frac{\psi}{2} \\ -i\sin\frac{\psi}{2} & \cos\frac{\psi}{2} \end{bmatrix}, \quad \mathbf{U_{ROTY}}(\psi) = \begin{bmatrix} \cos\frac{\psi}{2} & \sin\frac{\psi}{2} \\ -\sin\frac{\psi}{2} & \cos\frac{\psi}{2} \end{bmatrix}, \quad \mathbf{U_{ROTZ}}(\psi) = \begin{bmatrix} e^{i\frac{\psi}{2}} & 0 \\ 0 & e^{-i\frac{\psi}{2}} \end{bmatrix}.$$

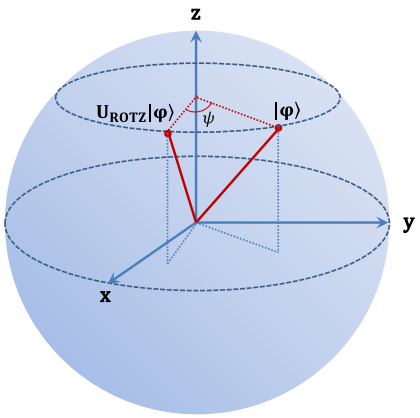


Figure 3.3 Geometrical representation of the $\mathbf{U_{ROTZ}}$ operator

3.3 Arbitrary Rotations

In the previous section, we showed that any transformation can be seen as a combination of rotations about the x, y, or z axes. More generally, any quantum gate acting on single qubit unitary matrix can expressed as a rotation about any arbitrary real unit vector axis $\mathbf{n}$,

$$\mathbf{n} = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}.$$

Any rotation of a qubit by angle ψ about the axis $\mathbf{n}$ can be represented by the following matrix,

$$R_{\mathbf{n}}(\psi) = e^{-i\frac{\psi}{2}\mathbf{n}\cdot\boldsymbol{\Sigma}}$$

where the matrix $\boldsymbol{\Sigma}$ is defined as,

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\sigma}_{\mathrm{X}} \\ \boldsymbol{\sigma}_{\mathrm{y}} \\ \boldsymbol{\sigma}_{\mathrm{Z}} \end{bmatrix},$$

so that,

$$\mathbf{n} \cdot \boldsymbol{\Sigma} = \mathbf{n}\boldsymbol{\Sigma}^{\mathrm{T}} = n_x \boldsymbol{\sigma}_{\mathrm{X}} + n_y \boldsymbol{\sigma}_{\mathrm{Y}} + n_z \boldsymbol{\sigma}_{\mathrm{Z}}.$$

Then the corresponding rotation matrix is,

$$R_{\mathbf{n}}(\psi) = \cos\frac{\psi}{2}\mathbf{I} - i\sin\frac{\psi}{2}(n_x \boldsymbol{\sigma}_{\mathrm{X}} + n_y \boldsymbol{\sigma}_{\mathrm{Y}} + n_z \boldsymbol{\sigma}_{\mathrm{Z}}).$$

Observation 3.3: (General representation of any unitary matrix) For any single-qubit unitary matrix **U**, there exist real numbers ω, α, β, and γ such that,

$$\mathbf{U} = \begin{bmatrix} \mathrm{e}^{i(\omega - \frac{\alpha}{2} - \frac{\gamma}{2})}\cos\frac{\beta}{2} & -\mathrm{e}^{i(\omega - \frac{\alpha}{2} + \frac{\gamma}{2})}\sin\frac{\beta}{2} \\ \mathrm{e}^{i(\omega + \frac{\alpha}{2} - \frac{\gamma}{2})}\sin\frac{\beta}{2} & \mathrm{e}^{i(\omega + \frac{\alpha}{2} + \frac{\gamma}{2})}\cos\frac{\beta}{2} \end{bmatrix}.$$

In general, any unitary gate on a single qubit can be represented as a combination of rotations, along with global phase shift. In particular, any 2×2 unitary matrix **U** can be implemented using only z and y rotations $\mathbf{U_{ROTZ}}$ and $\mathbf{U_{ROTY}}$ as shown below,

Observation-3.4: (zy-decomposition) Any 2×2 unitary matrix **U** can be expressed as,

$$\mathbf{U} = \mathrm{e}^{i\omega}\mathbf{U_{ROTZ}}(\alpha)\mathbf{U_{ROTY}}(\beta)\mathbf{U_{ROTZ}}(\gamma).$$

Using Observation 3.3, any unitary matrix can be expressed in the form,

$$\mathbf{U} = \begin{bmatrix} \mathrm{e}^{i(\omega - \frac{\alpha}{2} - \frac{\gamma}{2})}\cos\frac{\beta}{2} & -\mathrm{e}^{i(\omega - \frac{\alpha}{2} + \frac{\gamma}{2})}\sin\frac{\beta}{2} \\ \mathrm{e}^{i(\omega + \frac{\alpha}{2} - \frac{\gamma}{2})}\sin\frac{\beta}{2} & \mathrm{e}^{i(\omega + \frac{\alpha}{2} + \frac{\gamma}{2})}\cos\frac{\beta}{2} \end{bmatrix}$$

this yields to,

$$= \mathrm{e}^{i\omega}\begin{bmatrix} \mathrm{e}^{i\frac{\alpha}{2}} & 0 \\ 0 & \mathrm{e}^{-i\frac{\alpha}{2}} \end{bmatrix}\begin{bmatrix} \cos\frac{\beta}{2} & \sin\frac{\beta}{2} \\ -\sin\frac{\beta}{2} & \cos\frac{\beta}{2} \end{bmatrix}\begin{bmatrix} \mathrm{e}^{i\frac{\gamma}{2}} & 0 \\ 0 & \mathrm{e}^{-i\frac{\gamma}{2}} \end{bmatrix}$$

$$= \mathrm{e}^{i\omega}\mathbf{U_{ROTZ}}(\alpha)\mathbf{U_{ROTY}}(\beta)\mathbf{U_{ROTZ}}(\gamma).$$

3.4 Groups $\mathbf{SU(2)}$ and $\mathbf{SO(3)}$

Definition 3.5: (Group) A group is a set of elements G under a composition operator '$\circ$' with the following properties.

Closure,

$$a, b \in G \Rightarrow a \circ b \in G.$$

Existence of identity element I,

$$\exists I \in G \text{ such that } I \circ a = a \circ I = a.$$

Inversion,

$$a \in G \Rightarrow a^{-1} \in G, a \circ a^{-1} = a^{-1} \circ a = I.$$

Associativity,

$$a, b, c \in G \Rightarrow (a \circ b) \circ c = a \circ (b \circ c)$$

Additionally, if the operator $\circ$ obeys the commutative property, $\forall a, b \in G, a \circ b = b \circ a$, the group G is an Abelian group, otherwise non-Abelian.

Definition 3.6: (Non-Abelian group) If group G under a composition operator '$\circ$' is non-Abelian if and only if,

$$\exists a, b \in G, a \circ b \neq b \circ a.$$

Definition 3.7: (Group SU(2)) The special unitary group SU(2) consists of all 2×2 complex, unitary matrices with unit determinants. Hence the following properties hold,

$$\mathrm{SU}(2) \subset \mathbb{C}^{2\times 2}$$

$$\forall \mathbf{U} \in \mathrm{SU}(2), \mathbf{U}^{\dagger} = \mathbf{U}^{-1}.$$

$$\forall \mathbf{U} \in \mathrm{SU}(2), |\mathbf{U}| = 1.$$

Observation 3.4: Such matrices can be expressed in the form,

$$\mathbf{U} = \begin{bmatrix} a & b \\ -b^* & a^* \end{bmatrix}, \quad a, b \in \mathbb{C}.$$

Since $\det(\mathbf{U}) = 1$, clearly

$$|a|^2 + |b|^2 = 1.$$

Observation 3.5: Another convenient way of representing any such matrix is in terms of angles, ω, ϕ, and θ,

$$\mathbf{U} = \begin{bmatrix} \mathrm{e}^{i\omega}\cos\theta & \mathrm{e}^{i\phi}\sin\theta \\ -\mathrm{e}^{-i\phi}\sin\theta & \mathrm{e}^{-i\omega}\cos\theta \end{bmatrix}.$$

It can easily be seen that the two representations are equivalent.

In general, each rotation matrix can be written explicitly in terms of the rotation angle ψ and the rotation axis $\mathbf{n}$ which is defined by the unit vector:

$$\mathbf{n} = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}.$$

The corresponding rotation matrix $\mathbf{R_n}(\psi)$ can be represented as 3×3 real matrix in the following manner:

$$\mathbf{R_n}(\psi) = \begin{bmatrix} n_x^2(1-\cos(\psi)) + \cos(\psi) & n_x n_y(1-\cos(\psi)) - n_z\sin(\psi) & n_x n_z(1-\cos(\psi)) + n_y\sin(\psi) \\ n_x n_y(1-\cos(\psi)) + n_z\sin(\psi) & n_y^2(1-\cos(\psi)) + \cos(\psi) & n_y n_z(1-\cos(\psi)) - n_x\sin(\psi) \\ n_x n_z(1-\cos(\psi)) - n_y\sin(\psi) & n_y n_z(1-\cos(\psi)) + n_x\sin(\psi) & n_z^2(1-\cos(\psi)) + \cos(\psi) \end{bmatrix}.$$

Rotations around the x, y, and z axes by an arbitrary angle ψ can be represented as the following matrices:

$$\mathbf{R_x}(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix},$$

$$\mathbf{R_y}(\psi) = \begin{bmatrix} \cos\psi & 0 & \sin\psi \\ 0 & 1 & 0 \\ -\sin\psi & 0 & \cos\psi \end{bmatrix},$$

$$\mathbf{R_z}(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Definition 3.8: (Group SO(3)) The group $\mathrm{SO}(3) \subset \mathbb{R}^{3\times3}$ is the set of all possible such rotations $R_{\mathbf{n}}(\psi)$ under matrix product. Since rotations in 3-dimensions do not commute, SO(3) is a non-Abelian group. Note that rotations preserve length, implying that the matrix determinants $\det(\mathbf{O})$ of all $\mathbf{O} \in \mathrm{SO}(3)$ are unity. Reflections are excluded from SO(3) disallowing matrices with determinants of -1. The inverse $\mathbf{O^{-1}}$ of each rotation $\mathbf{O}$ is simply a rotation in the opposite direction by the same angle and around the same axis and is simply the transposed matrix $\mathbf{O}^{\mathrm{T}}$, so that $\mathbf{O}^{\mathrm{T}}\mathbf{O} = \mathbf{I}$.

Chapter 4 - Synthesis of Quantum Gates

Synthesizing any arbitrary single qubit quantum gate involves constructing the gate using only those available within a small set of universal gates that can be realized physically. However, there are an infinite number of single qubit quantum gates. Hence, synthesizing quantum gates to arbitrary degree of precision is considered. Solovay-Kitaev algorithm has been introduced to progressively enhance the accuracy of any arbitrary matrix approximation. In this chapter, the fundamental definitions necessary to understand the Solovay-Kitaev algorithm is presented. In section 4.1, the definition of universal gate set which is used to realize any quantum gate is shown. The error in approximation is discussed in section 4.2. In section 4.3, the naïve linear search method used in the Solovay-Kitaev algorithm is discussed. A detailed description of the original Solovay-Kitaev algorithm is thoroughly discussed in section 4.4. Finally, another search method proved to outperform the linear search method called geometric nearest-neighbor access trees (GNAT) is presented in section 4.5.

4.1 Universal Gate set

The set of gates from which any classical operation may be built is called the universal gate set [Barenco et al. 1995]. It is well known in the classical Boolean that the set of AND and NOT gates form a universal one [Deutsch et al. 1995]. Such a set of gates can be used to perform any classical operation. This situation is analogous to the quantum circuit model where unitary operations need to be realized. A universal quantum set exist and can be used to implement any arbitrary unitary operation with a circuit exclusively made of gates belongs to the set. A set of quantum gates is called universal if an arbitrary sequence of which can generate a dense subset of all gates. In other words,

Definition 4.1: (Universal gate set in SU(2)) $\mathbf{\Gamma}$ is a universal gate set

iff $\mathbf{\Gamma}^* \subset_D \mathrm{SU}(2)$,

where $\mathbf{\Gamma}^*$ is the set of all possible finite strings of the elements of $\mathbf{\Gamma}$.

Definition 4.2: (Dense subset) For every $\mathbf{U} \in \mathrm{SU}(2)$, either

(i) $\mathbf{U} \in \mathbf{\Gamma}^*$ or

(ii) $\exists\, \widetilde{\mathbf{U}} \in \mathbf{\Gamma}^*$ such that,

$\left\|\widetilde{\mathbf{U}} - \mathbf{U}\right\| \leq \varepsilon$,

here $\|\widetilde{\mathbf{U}} - \mathbf{U}\|$ is a measure of distance between $\mathbf{U}$ and its approximation $\widetilde{\mathbf{U}}$ that is introduced in the next section.

The simplest univesral quantum set consists of the two single qubit gates, Hadamard and T gate (which is a z-rotation of $\frac{\pi}{8}$) [Nielsen 2010]. The Two gates can be represented in matrix form as follows,

$$\mathbf{H} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \qquad \mathbf{T} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix}.$$

There exist uncountably many unitary operations that cannot be all implemented using a universal set of finite quantum gates [Mermin 2007]. This means that a desired unitary operation can generally only be approximated up to arbitrary accuracy using gates taken only from a universal set. In order to quantify this accuracy, a notion of distance between operators is defined in the following section.

4.2 Error in quantum gate approximations

Error in approximation can be defined as the distance between an exact value and some approximation to it. In other words, it quantifies how far the approximate value from the original one is. When dealing with quantum gates, a function of distance between unitary matrices is needed. There are several distance measures for matrices, but for our purposes the trace distance will suffice. In more details, if $\mathbf{U}_1, \mathbf{U}_2 \in \mathrm{SU}(2)$, then the distance between $\mathbf{U}_1$ and $\mathbf{U}_2$, denoted as $\|\mathbf{U}_1 - \mathbf{U}_2\|$ operator, is defined as follows:

$$\|\mathbf{U}_1 - \mathbf{U}_2\| = \frac{1}{2}\,\mathrm{Tr}\left(\sqrt{(\mathbf{U}_1 - \mathbf{U}_2)^\dagger(\mathbf{U}_1 - \mathbf{U}_2)}\right).$$

Where for any $n \times n$ matrix $\mathbf{A}$, the trace of $\mathbf{A}$ is defined by,

$$\mathrm{Tr}(\mathbf{A}) = \sum_{j=1}^{n} \mathbf{A}_{j,j}^2.$$

Definition 4.3: (Distance) The distance between any $\mathbf{U}_1, \mathbf{U}_2 \in \mathrm{SU}(2)$ is given by,

$$\|\mathbf{U}_1 - \mathbf{U}_2\| = \frac{1}{2}\,\mathrm{Tr}\left(\sqrt{(\mathbf{U}_1 - \mathbf{U}_2)^\dagger(\mathbf{U}_1 - \mathbf{U}_2)}\right).$$

4.3 Linear Search

The simplest and most straightforward search method for finding approximation to any arbitrary quantum gate is the linear search method. It is based on finding the closest candidate among all possible strings of gates taken from a universal set of finite number of gates. This collection of all possible gate strings is called the net (i.e. $\boldsymbol{\Gamma}^*$). It can be constructed by recursively walking the entire tree of matrix products of the universal gate set. If the tree generated has l levels, then the candidate strings will involve at most l elements of $\boldsymbol{\Gamma}$. The number of gates in the candidate string is referred to as the length of the string (i.e. l)

Definition 4.4: (Net $\boldsymbol{\Gamma}^*$) The representation of all possible finite strings of the elements of a universal gate set $\boldsymbol{\Gamma}$ arranged in a form of a tree is called the net (i.e. $\boldsymbol{\Gamma}^*$).

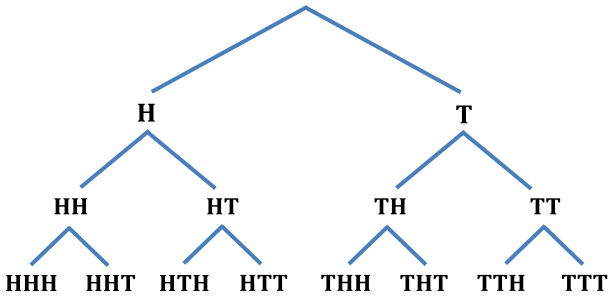


Figure 4.1 Example of search tree of $\boldsymbol{l = 3}$

The advantage of arranging the strings as a tree is to enable the linear search to proceed in a systematic manner starting from a root node. For example (see figure 4.1), if $\boldsymbol{\Gamma}$ is the set of the gates $\{\mathbf{H}, \mathbf{T}\}$ then the first level of the tree consists of $\{\mathbf{H}, \mathbf{T}\}$. On the second level, the strings of $\{\mathbf{HH}, \mathbf{HT}, \mathbf{TH}, \mathbf{TT}\}$ are obtained. This proceeds in a recursive manner such that on level l, strings of length l are achieved. In the example provided where the number of level $l = 3$, $\boldsymbol{\Gamma}^*$ is the set of the strings $\{\mathbf{H}, \mathbf{T}, \mathbf{HH}, \dots, \mathbf{TT}, \mathbf{HHH}, \mathbf{HTH}, \dots, \mathbf{TTT}\}$.

The actual linear search subroutine takes a target gate to be approximated as an input, in addition to the pre-generated net $\boldsymbol{\Gamma}^*$. The subroutine evaluates the distance between the target gate and all possible strings in $\boldsymbol{\Gamma}^*$. The string with the minimum distance to the target gate is chosen. However,

searching among all strings for the best approximation is an exhaustive procedure and quickly becomes infeasible if $\mathbf{\Gamma}^*$ is large.

4.4 The Solovay-Kitaev algorithm

The Solovay-Kitaev algorithm is perhaps the best known method for generating sufficiently good approximations. The Solovay-Kitaev algorithm is a recursive procedure that decomposes any arbitrary single qubit gate $\mathbf{U} \in \mathrm{SU}(2)$ into a sequence of gates from a universal gate family $\mathbf{\Gamma} \subseteq \mathrm{SU}(2)$ [Dawson & Nielsen 2005], such that,

$$\widetilde{\mathbf{U}} \equiv \mathbf{U}_1\mathbf{U}_2 \dots \mathbf{U}_l \in \mathbf{\Gamma}.$$

The effectiveness of the algorithm is quantified in terms of the error Δ which is also the distance between the desired unitary matrix $\mathbf{U}$ and the approximation, $\widetilde{\mathbf{U}}$ that the algorithm yields, i.e. $\Delta = \left\|\mathbf{U} - \widetilde{\mathbf{U}}\right\|$. Since this physically realizable matrix $\widetilde{\mathbf{U}}$ is representative of $\mathbf{U}$ only up to a maximum accuracy of Δ, it will be referred to hereafter, as the Δ-approximation of $\mathbf{U}$.

A corresponding Solovay-Kitaev theorem provides the necessary theoretical underpinnings behind this algorithm. This theorem states that, given any quantum gate $\mathbf{U} \in \mathrm{SU}(2)$ there exists a product $\widetilde{\mathbf{U}} \equiv \mathbf{U}_1\mathbf{U}_2 \dots \mathbf{U}_l$ of gates from any universal set $\mathbf{\Gamma}$ which is a Δ-approximation to $\mathbf{U}$, for any desired accuracy $\Delta > 0$, no matter how small. However the tradeoff is that the length of the string l grows rapidly with decreasing Δ.

Observation 4.1: Given any universal gate family $\mathbf{\Gamma}$, a unitary matrix $\mathbf{U}$ in group $\mathrm{SU}(2)$, the length l of the approximating sequence $\widetilde{\mathbf{U}}$ in $\mathbf{\Gamma}^*$ increases, such that $l = O(\log^c \Delta)$. The value of the quantity c is around 4.

Mathematically speaking,

$$\forall\, \mathbf{U} \in \mathrm{SU}(2) \text{ and } \Delta > 0,$$

$$\exists\, \widetilde{\mathbf{U}} = \prod_{k=1}^{l} \mathbf{U}_k\,, \qquad \mathbf{U}_k \in \mathbf{\Gamma}\ (l = 1,2, \dots, l),$$

such that,

$$\left\|\mathbf{U} - \widetilde{\mathbf{U}}\right\| \leq \Delta,$$

$$l = O(\log^c \Delta),$$

where $c \approx 4$.

The length of the initial approximated gate ($\widetilde{\mathbf{U}}$) is denoted as follows,

$$l = |\widetilde{\mathbf{U}}|.$$

The Solovay-Kitaev algorithm is a tree-based search process. It performs a recursive search over a binary tree such that each level of recursion results in the bifurcation of node into two child nodes. Note that this tree is different from the net (Definition 4.4) which also has a similar tree structure. Each recursive instance of the Solovay-Kitaev algorithm terminates at the level $n = 0$, i.e. at the leaf nodes. Due to the one-to-one correspondence between the depth of the algorithm with the level of this tree, the terms 'level' and 'depth' will be used interchangeably throughout this dissertation.

The basic Solovay-Kitaev algorithm is as follows:

```
function Ũ_n = Solovay-Kitaev (U, n)
if (n = 0)
      Ũ_0 = approx(U)
      return Ũ_0
else
      Ũ_{n-1} = Solovay-Kitaev (U, n − 1)
      V_{n-1}, W_{n-1} = GC-Decomp (U, Ũ_{n-1})
      Ṽ_{n-1} = Solovay-Kitaev (V_{n-1}, n − 1)
      W̃_{n-1} = Solovay-Kitaev (W_{n-1}, n − 1)
      Ũ_n = Ṽ_{n-1} W̃_{n-1} Ṽ†_{n-1} W̃†_{n-1} Ũ_{n-1}
      return Ũ_n
end
```

The Solovay-Kitaev algorithm procedure accepts as its input the target gate $\mathbf{U}$, which is the gate needed to be approximated, as well as the desired depth n such that $n \geq 0$. The function returns an approximate gate $\widetilde{\mathbf{U}}_n$ that lies within a distance Δ of the target gate $\mathbf{U}$,

$$\|\mathbf{U} - \widetilde{\mathbf{U}}_n\| = \Delta,$$

such that $\widetilde{\mathbf{U}}_n$ is a sequence of gates taken from a universal gate set.

At each recursive level n, the algorithm calls itself to obtain a coarser stage $n-1$ approximation $\widetilde{\mathbf{U}}_{n-1}$. Then, a group decomposition subroutine is invoked to obtain $\mathbf{V}_{n-1}$, $\mathbf{W}_{n-1}$ such that,

$$\mathbf{U}\widetilde{\mathbf{U}}_{n-1}^{\dagger} = \mathbf{V}_{n-1}\mathbf{W}_{n-1}\mathbf{V}_{n-1}^{\dagger}\mathbf{W}_{n-1}^{\dagger},$$

where $\mathbf{V}_{n-1}, \mathbf{W}_{n-1} \in \mathrm{SU}(2)$.

The outputs of decomposing $\mathbf{U}\widetilde{\mathbf{U}}_{n-1}^{\dagger}$ are called group commutators $\mathbf{V}_{n-1}$ and $\mathbf{W}_{n-1}$ which are unitary matrices that are close to the identity matrix $\mathbf{I}$, such that,

$$\delta = \max\{\|\mathbf{I} - \mathbf{V}_n\|, \|\mathbf{I} - \mathbf{W}_n\|\},$$

Subsequently, the group commutators themselves are approximated further by calling the same algorithm with $n-1$ level. The approximate values are denoted as $\widetilde{\mathbf{V}}_{n-1}$and $\widetilde{\mathbf{W}}_{n-1}$ and they are of Δ distance of $\mathbf{V}_{n-1}$ and $\mathbf{W}_{n-1}$, such that,

$$\Delta = \max\{\|\widetilde{\mathbf{V}}_{n-1} - \mathbf{V}_n\|, \|\widetilde{\mathbf{W}}_{n-1} - \mathbf{W}_n\|\}.$$

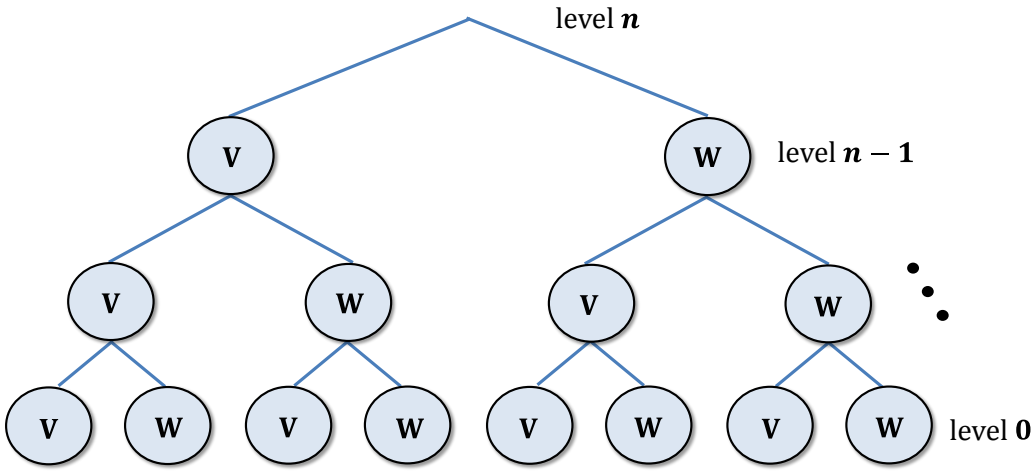


Figure 4.2 The binary tree bifurcates into two child nodes (**V** and **W**) at each level of recursion.

The algorithm performs a recursive search over the binary tree shown in figure 4.2.There is a preprocessing step needed in order to implement the first stage of the algorithm ($n = 0$) when no recursion is involved. The preprocessing step involves generating a net of all possible sequences up to a certain length l_0. This step can be implemented offline and then used within the algorithm, specifically when $(n = 0)$, to determine the best approximation of $\mathbf{U}_0$ as a sequence $\widetilde{\mathbf{U}}_0$. The subroutine for the basic approximation is described in further details in the next chapter.

The Solovay Kitaev function returns the strings approximating the group commutator $\widetilde{\mathbf{U}}_n$ such that,

$$\widetilde{\mathbf{U}}_n = \widetilde{\mathbf{V}}_{n-1}\widetilde{\mathbf{W}}_{n-1}\widetilde{\mathbf{V}}_{n-1}^{\dagger}\widetilde{\mathbf{W}}_{n-1}^{\dagger}\widetilde{\mathbf{U}}_{n-1}$$

At each stage of recursion, the length of the approximating string grows exponentially with the stage of recursion. Specifically, the length of the approximating string grows by a factor of 5 with every stage. Thus,

Observation 4.2: If l_0 is the length of the initial approximate sequence $\widetilde{\mathbf{U}}_0$,

$$l_0 = \left| \widetilde{\mathbf{U}}_0 \right|.$$

Then the length of the approximating string at stage n is given by,

$$\left| \widetilde{\mathbf{U}}_n \right| = 5 \times n \times l_0.$$

The power of the Solovay-Kitaev algorithm lies in the fact that with each increasing stage, the $\widetilde{\mathbf{U}}_n$ of $\mathbf{U}$ is a better approximation than those obtained from the previous stage. This can be expressed as the following relationship:

$$\left\| \widetilde{\mathbf{V}}_{n-1} \widetilde{\mathbf{W}}_{n-1} \widetilde{\mathbf{V}}_{n-1}^{\dagger} \widetilde{\mathbf{W}}_{n-1}^{\dagger} - \mathbf{V}_n \mathbf{W}_n \mathbf{V}_n{}^{\dagger} \mathbf{W}_n{}^{\dagger} \right\| = O(\delta^2 + \Delta\delta).$$

4.5 GNAT Search

In the original Solovay-Kitaev algorithm, an exhaustive search over all possible approximations is performed in the initial approximation stage ($n = 0$) in order to obtain $\widetilde{\mathbf{U}}_0$. The naïve linear search requires finding all the distances between the desired gate to be approximated and every possible approximation in the search net $\boldsymbol{\Gamma}^*$. Hence, looking for the best approximation in the whole search space significantly increases the algorithm run time. Therefore, obtaining the initial approximation more efficiently can significantly speed up the algorithm. In order to accomplish this, a technique that doesn't require exhaustive search over all possible approximations is implemented. This search technique is called the geometric nearest-neighbor access tree search (GNAT) [Brin 1995]. It is a powerful technique which significantly reduces the search space to the point that naïve linear search can be feasibly performed.

The GNAT search technique (figure 4.3) starts with the net of all possible strings of matrix products with matrices only taken from a universal gate family $\boldsymbol{\Gamma}^*$. A fixed number of elements in the net (called root nodes) are picked randomly. Initially, the distances between each root node and every other remaining element in $\boldsymbol{\Gamma}^*$ are evaluated. Consequently, each element is assigned to its closest

root node. This procedure is applied recursively until the size of the node collection is small enough that a linear search is feasible.

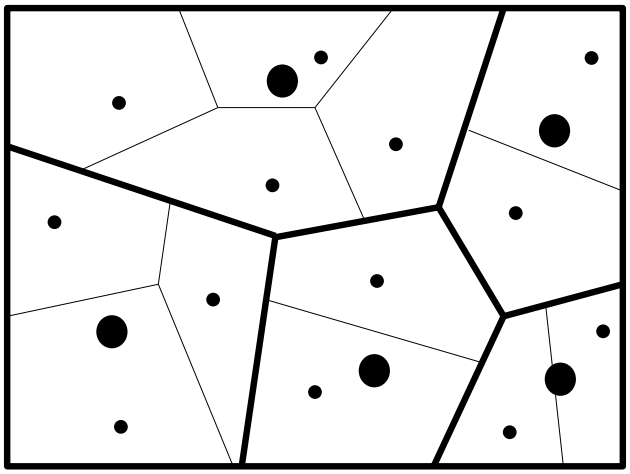


Figure 4.3 A diagram of a simple GNAT with 5 root nodes.

This whole procedure can be performed offline before the actual GNAT search subroutine proceeds as follows. It takes the target gate to be approximated as an input and instead of comparing it with the whole net $\mathbf{\Gamma}^*$ as in the linear search, the GNAT subroutine compares it with the root nodes only. Then it recursively searches within the closest node collection. The GNAT search ends with the naïve linear search among the collection of the last sub-node. This technique implements the search method more efficiently by drastically reducing the search space. Hence, significant speed up over the original Solovay-Kitaev algorithm is achieved.

Chapter 5- Stochastic Group Commutator Enhancement

This chapter introduces the first of the two approaches proposed in this research. All algorithms and subroutines involved in this part of the study are detailed here, while the simulation results and theoretical analysis has been postponed until the next chapter. In this aspect of the research, the group commutator decomposition is subject to explicit optimization in order to enhance the accuracy of the synthesized quantum gates. This task has been modeled as a constraint optimization problem with equality constraints. Since there is a pair of matrices acting as the group commutators, a separate procedure is invoked for each, which is done in a stochastic manner. The framework of the proposed approach is discussed in section 5.1. Two versions of the new approach have been addressed, the difference being in the level within the search tree where this optimization is invoked. In section 5.2, the first version of single-level group commutator enhancement is introduced. The second is presented in section 5.3 with enhancement applied to all levels of the algorithm.

5.1 Framework

There is no group commutators involved at the lowest level ($n = 0$), because at the leaf node of the search tree, the actual sequences of the physically realizable quantum gates in $\boldsymbol{\Gamma}$ are obtained. The implication of this observation is that the proposed algorithm which focuses in retuning the group commutators to lower the error, does not differ from the original Solovay-Kitaev algorithm (chapter 4). Consequently, the skeleton of the new algorithm which is now described, does not consider $n = 0$. It is formulated to work for higher depths, so that the subroutine for group commutator decomposition is always called. The new version of the Solovay-Kitaev algorithm is presented below.

$[\widetilde{\mathbf{U}}_n] = \texttt{SolovayKitaev2}(\mathbf{U}, n)$
    `if` $n == 1$
        $[\widetilde{\mathbf{U}}_0] = \texttt{basicApprox}(\mathbf{U})$
        $[\mathbf{V}_0, \mathbf{W}_0] = \texttt{GCDecompose}(\mathbf{U}\widetilde{\mathbf{U}}_0^{\dagger})$
        $[\widetilde{\mathbf{V}}_0] = \texttt{basicApprox}(\mathbf{V}_0)$
        $[\widetilde{\mathbf{W}}_0] = \texttt{basicApprox}(\mathbf{W}_0)$
    `else`
        $[\widetilde{\mathbf{U}}_{n-1}] = \texttt{SolovayKitaev2}(\mathbf{U}, n-1)$
        $[\mathbf{V}_{n-1}, \mathbf{W}_{n-1}] = \texttt{GCDecompose}(\mathbf{U}\widetilde{\mathbf{U}}_{n-1}^{\dagger})$
        $[\widetilde{\mathbf{V}}_{n-1}] = \texttt{SolovayKitaev2}(\mathbf{V}_{n-1}, n-1)$

$$[\tilde{\mathbf{W}}_{n-1}] = \texttt{SolovayKitaev2}(\mathbf{W}_{n-1}, n-1)$$

end

$$\tilde{\mathbf{U}}_n = \tilde{\mathbf{V}}_{n-1}\tilde{\mathbf{W}}_{n-1}\tilde{\mathbf{V}}_{n-1}^{\dagger}\tilde{\mathbf{W}}_{n-1}^{\dagger}\tilde{\mathbf{U}}_{n-1}$$

This algorithm involves two main subroutines with one of them being the basic approximation (`basicApprox()`) subroutine and the other is the group commutator decomposition (`GCDecompose()`) subroutine. The (`basicApprox()`) subroutine is called only at the lowest level of the search tree. It returns basic ϵ_0 −approximation to a given gate $\mathbf{U}$ denoted as $\tilde{\mathbf{U}}_0$. A lookup routine is performed to find the closest approximation among all strings of gates in $\boldsymbol{\Gamma}^*$, such that,

$$\|\mathbf{U} - \tilde{\mathbf{U}}_0\| = \epsilon_0.$$

The lookup routine used in the standard Solovay-Kitaev algorithm is linear search (addressed in chapter 4). However, the GNAT search (discussed in chapter 4) notably outperforms the naïve linear version. Although one drawback of the GNAT search is the time needed to construct node collections, this procedure is performed offline and simply called during the implementation of the subroutine. Hence, it is the search method used in this basic approximation subroutine to lookup the ϵ_0 −approximation to a given gate $\mathbf{U}$. Subsequently, it is the method adopted in the new proposed algorithms.

$[\tilde{\mathbf{U}}_0] = \texttt{basicApprox}(\mathbf{U})$

Find

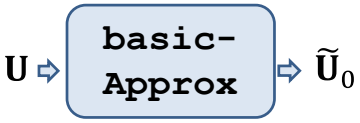
$\tilde{\mathbf{U}}_0$

To minimize

$\epsilon_0 = \|\mathbf{U} - \tilde{\mathbf{U}}_0\|$

Subject to

$\tilde{\mathbf{U}}_0 \in \boldsymbol{\Gamma}^*$



The other subroutine is the group commutator decomposition (`GCDecompose()`) which is called in all levels of the new version of the algorithm. This subroutine is shown below,

$[\mathbf{V}_{n-1}, \mathbf{W}_{n-1}] = \texttt{GCDecompose}(\mathbf{U}\tilde{\mathbf{U}}_{n-1}^{\dagger})$

Find

$\mathbf{V}_{n-1}, \mathbf{W}_{n-1}$

Subject to

$\|\mathbf{I} - \mathbf{V}_{n-1}\|, \|\mathbf{I} - \mathbf{W}_{n-1}\| \leq \delta_{n-1}$

$\mathbf{U}\tilde{\mathbf{U}}_{n-1}^{\dagger} = \mathbf{V}_{n-1}\mathbf{W}_{n-1}\mathbf{V}_{n-1}^{\dagger}\mathbf{W}_{n-1}^{\dagger}$

$\|\tilde{\mathbf{U}}_{n-1} - \mathbf{U}\| = \epsilon_{n-1}$

The approximation matrix $\widetilde{\mathbf{U}}_{n-1}$ is imployed in the (`GCDecompose()`) subroutine to find the product $\mathbf{U}\widetilde{\mathbf{U}}_{n-1}^{\dagger}$. Since both $\mathbf{U}$ and $\widetilde{\mathbf{U}}_{n-1}^{\dagger}$ are unitaries, the product matrix is close to the identity. Specifically, $\mathbf{U}\widetilde{\mathbf{U}}_{n-1}^{\dagger}$is of distance ϵ_{n-1} of the identity,

$$\left\|\mathbf{I}-\mathbf{U}\widetilde{\mathbf{U}}_{n-1}^{\dagger}\right\|=\epsilon_{n-1}.$$

Consequently, this product $\mathbf{U}\widetilde{\mathbf{U}}_{n-1}^{\dagger}$ is decomposed in terms of two matricies $\mathbf{V},\mathbf{W}\in \mathrm{SU}(2)$called the group commutators in the following manner,

$$\mathbf{U}\widetilde{\mathbf{U}}_{n-1}^{\dagger}=\mathbf{V}_{n-1}\mathbf{W}_{n-1}\mathbf{V}_{n-1}^{\dagger}\mathbf{W}_{n-1}^{\dagger}.$$

Both matrices are of distance δ_{n-1} of the identity,

$$\|\mathbf{I}-\mathbf{V}_{n-1}\|,\|\mathbf{I}-\mathbf{W}_{n-1}\|\leq\delta_{n-1}.$$

The `GCDecompose` subroutine is invoked when $n=1$ as well as heighr levels. The outputs themselves $\mathbf{V}_{n-1}$ and $\mathbf{W}_{n-1}$ are then being approximated using the (`SolovayKitaev2()`) algorithm to get $\widetilde{\mathbf{V}}_{n-1}$and $\widetilde{\mathbf{W}}_{n-1}$. After n recursions, the final approximation of the target gate $\mathbf{U}$ is obtained as follows,

$$\widetilde{\mathbf{U}}_{n}=\widetilde{\mathbf{V}}_{n-1}\widetilde{\mathbf{W}}_{n-1}\widetilde{\mathbf{V}}_{n-1}^{\dagger}\widetilde{\mathbf{W}}_{n-1}^{\dagger}\widetilde{\mathbf{U}}_{n-1}.$$

5.2 Solovay Kitaev Approximation with Single-Level Group Commutator Enhancement

In the first version of the enhanced Solovay-Kitaev algorithm (`SolovayKitaev3()`), optimization is incorporated within the group commutator decomposition only at the bottom level ($n=1$) of Solovay-Kitaev algorithm. The enhanced version of the original Solovay-Kitaev algorithm is presented below,

```
[Ũ_n] = SolovayKitaev3(U, n)
    if n == 1
        [Ũ_0] = basicApprox(U)
        [V_0, W_0] = GCDecompose(UŨ_0†)
        % Simulate Markov Chain
        repeat
            either
```

$[\tilde{\mathbf{V}}_0] = \texttt{basicApprox}(\mathbf{V}_0)$

$[\mathbf{W}_0] = \texttt{GCEnhance}(\mathbf{U}\tilde{\mathbf{U}}_0^\dagger, \tilde{\mathbf{V}}_0)$

$[\tilde{\mathbf{W}}_0] = \texttt{basicApprox}(\mathbf{W}_0)$

or

$[\tilde{\mathbf{W}}_0] = \texttt{basicApprox}(\mathbf{W}_0)$

$[\mathbf{V}_0] = \texttt{GCEnhance}(\mathbf{U}\tilde{\mathbf{U}}_0^\dagger, \tilde{\mathbf{W}}_0)$

$[\tilde{\mathbf{V}}_0] = \texttt{basicApprox}(\mathbf{V}_0)$

until (condition satisfied)

else

$[\tilde{\mathbf{U}}_{n-1}] = \texttt{SolovayKitaev3}(\mathbf{U}, n-1)$

$[\mathbf{V}_{n-1}, \mathbf{W}_{n-1}] = \texttt{GCDecompose}(\mathbf{U}\tilde{\mathbf{U}}_{n-1}^\dagger)$

$[\tilde{\mathbf{V}}_{n-1}] = \texttt{SolovayKitaev3}(\mathbf{V}_{n-1}, n-1)$

$[\tilde{\mathbf{W}}_{n-1}] = \texttt{SolovayKitaev3}(\mathbf{W}_{n-1}, n-1)$

end

$\tilde{\mathbf{U}}_n = \tilde{\mathbf{V}}_{n-1}\tilde{\mathbf{W}}_{n-1}\tilde{\mathbf{V}}_{n-1}^\dagger\tilde{\mathbf{W}}_{n-1}^\dagger\tilde{\mathbf{U}}_{n-1}$

The new Solovay-Kitaev algorithm shown above accepts as input arguments, a target unitary gate $\mathbf{U}$ that is to be approximated. The initial approximation of $\mathbf{U}$ is evaluated $\tilde{\mathbf{U}}_0$ using the basic approximation (`basicApprox()`) subroutine. Then, the group commutators decomposition (`GCDecompose()`) subroutine is invoked to decompose the product $\mathbf{U}\tilde{\mathbf{U}}_0^\dagger$ into the group commutater pair $\mathbf{V}_0, \mathbf{W}_0$ such that $\mathbf{U}\tilde{\mathbf{U}}_0^\dagger = \mathbf{V}_0\mathbf{W}_0\mathbf{V}_0^\dagger\mathbf{W}_0^\dagger$. The optimization step is applied next either to enhance $\mathbf{V}$ or $\mathbf{W}$ by invoking the group commutator enhancement subroutines for $\mathbf{V}$ and $\mathbf{W}$. Transitioning between the two subroutines of enhancing $\mathbf{V}$ and $\mathbf{W}$ is stochasticaly modeled as a Markov chain with two states. The first of the two states is where $\mathbf{V}$ is enhanced and its second, where $\mathbf{W}$ is enhanced. The transition probability p, which is predetermined to any value, refers to the probability of the algorithm switching from enhancing one of the group commutators to the other. When no such transition occurs, the algorithm tries to re-enhance the group commutator for higher accuracy. The Markov model is shown in the figure below.

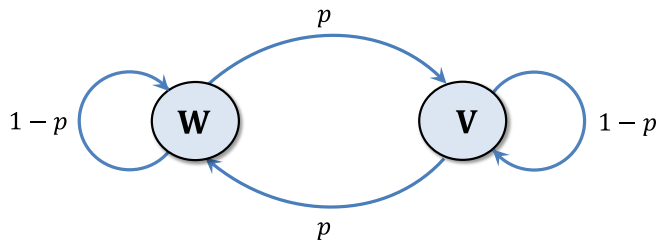


Figure 5.1 The standard Markov model

It must be noted that the enhancement of one of them may ocassionally fail. This event automatically triggers a transition from one state to the other, regardless of the value of p. Three Markov models, each with its own p are investigated.

- Model 1: $(p = 1)$

  Here, the transition probability is set at a value of unity. In other words, the algorithm alternately switches between enhancing **V** and **W**.

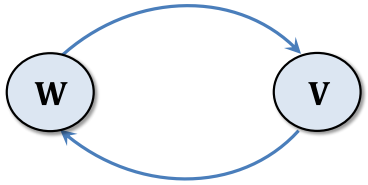


Figure 5.2 Model 1 with transitioning probability $\boldsymbol{p = 1}$

- Model 2: $(p = 0)$

  In this model, the algorithm switches into enhancing the other group commutator just if the the first one failed to enhance or the number of iterations predetermined for the corresponding subroutine is 0.

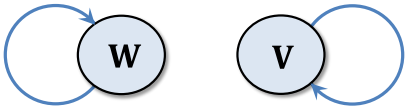


Figure 5.3 Model 2 with transitioning probability $\boldsymbol{p = 0}$

- Model 3: $(p = 0.5)$

  Here, the steps of enhancement don't follow a certain pattern. Transitioning between the two subroutines of **V** and **W** is totally random with $p = 0.5$.

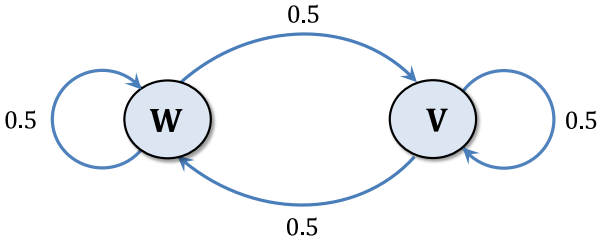


Figure 5.4 Model 3 with transitioning probability $\boldsymbol{p = 0.5}$

In order to implement the optimization step, two separate subroutines for the group commutator enhancement are presented. For porposes of consinessness, both are called the same name but one for $\mathbf{V}$ and the other for $\mathbf{W}$,

$[\mathbf{V}_0] = \texttt{GCEnhance}(\mathbf{U}\tilde{\mathbf{U}}_{n-1}^{\dagger}, \tilde{\mathbf{W}}_{n-1})$

Find

$\mathbf{V}_{n-1}$

To minimize

$\delta_{n-1}^{v} = \|\mathbf{I} - \mathbf{V}_{n-1}\|$

Subject to

$\mathbf{U}\tilde{\mathbf{U}}_{n-1}^{\dagger} = \mathbf{V}_{n-1}\tilde{\mathbf{W}}_{n-1}\mathbf{V}_{n-1}^{\dagger}\tilde{\mathbf{W}}_{n-1}^{\dagger}$

$[\mathbf{W}_0] = \texttt{GCEnhance}(\mathbf{U}\tilde{\mathbf{U}}_{n-1}^{\dagger}, \tilde{\mathbf{V}}_{n-1})$

Find

$\mathbf{W}_{n-1}$

To minimize

$\delta_{n-1}^{w} = \|\mathbf{W}_{n-1} - \mathbf{I}\|$

Subject to

$\mathbf{U}\tilde{\mathbf{U}}_{n-1}^{\dagger} = \tilde{\mathbf{V}}_{n-1}\mathbf{W}_{n-1}\tilde{\mathbf{V}}_{n-1}^{\dagger}\mathbf{W}_{n-1}^{\dagger}$

Although (`GCEnhance()`) subroutine in the SLE algorithm is only invoked when $n = 1$, It is been written in terms of n. The reason is that the same (`GCEnhance()`) subroutine is going to be used in the multi-level enhancement algorithm discussed in the next section. The following description is applied only to $n = 1$ where this subroutine is called to either enhance $\mathbf{V}_0$ or $\mathbf{W}_0$. Initially, the algorithm starts as usual by decomposing the product $\mathbf{U}\tilde{\mathbf{U}}_0^{\dagger}$ to its group commutators. The approximation of only one of the group commutators is evaluated. In the proposed enhacement algorithm, the product $\mathbf{U}\tilde{\mathbf{U}}_0^{\dagger}$ is decomposed using the matrix of the group commutator being approximated to bring the other matrix of the group commutators closer to the identity $\mathbf{I}$. In other words, one of the group commutators is fixed and the other is tuned using the corresponding enhancement subroutine (`GCEnhance()`). For instance, if the subroutine being invoked is to enhance $\mathbf{V}_0$, the approximation of the other group commutators $\tilde{\mathbf{W}}_0$ is fixed. The product $\mathbf{U}\tilde{\mathbf{U}}_0^{\dagger}$ is decomposed in the following manner,

$$\mathbf{U}\tilde{\mathbf{U}}_0^{\dagger} = \mathbf{V}_0\tilde{\mathbf{W}}_0\mathbf{V}_0^{\dagger}\tilde{\mathbf{W}}_0^{\dagger}.$$

Subsequently, $\mathbf{V}_0$ is enhanced if the distance of the identity δ_0^v is minimized, where,

$$\delta_0^v = \|\mathbf{I} - \mathbf{V}_0\|.$$

This mechanism is proposed since $\widetilde{\mathbf{V}}_0\widetilde{\mathbf{W}}_0\widetilde{\mathbf{V}}_0{}^{\dagger}\widetilde{\mathbf{W}}_0{}^{\dagger}\,\widetilde{\mathbf{U}}_0$ deviates further from the desired gate $\mathbf{U}$ than $\mathbf{V}_0\widetilde{\mathbf{W}}_0\mathbf{V}_0^{\dagger}\widetilde{\mathrm{W}}_0^{\dagger}\,\widetilde{\mathrm{U}}_0$. The implication of this corrective step is that the quantity $\Delta_{\mathrm{n}-1}^{\mathrm{w}}$ is ideally reduced to zero so that,

$$\Delta_{n-1}^{w} = \left\|\widetilde{\mathbf{W}}_{n-1} - \mathbf{W}_{n-1}\right\| = 0.$$

When either one of the subroutines fail to meet the condition given, or in other words fails to produce a more accurate version of the input, any future enhancement using this specific subroutine is redundant. Hence, a mechanism that switches immediately to the other subroutine is applied such that unnecessary computations are avoided.

5.3 Solovay Kitaev Approximation with Multi-Level Group Commutator Enhancement

Another proposed algorithm is called multi-level group commutator enhancement (MLE) algorithm. In this algorithm, the optimization is not only invoked at the bottom level but carried out across all levels of the search tree $n \geq 1$. This allows multiple enhancements to be applied on the group commutators $\mathbf{V}$ and $\mathbf{W}$. Hence yields better results than the single level enhancement. The algorithm is presented below,

$\left[\widetilde{\mathbf{U}}_n\right] =$ `SolovayKitaev4`$(\mathbf{U}, n)$

    if $n == 1$
        $\left[\widetilde{\mathbf{U}}_0\right] =$ `basicApprox`$(\mathbf{U})$
    else
        $\left[\widetilde{\mathbf{U}}_{n-1}\right] =$ `SolovayKitaev4`$(\mathbf{U}, n-1)$
        $[\mathbf{V}_{n-1}, \mathbf{W}_{n-1}] =$ `GCDecompose`$(\mathbf{U}\widetilde{\mathbf{U}}_{n-1}^{\dagger})$
        % Simulate Markov Chain
        repeat
            either
                $\left[\widetilde{\mathbf{V}}_{n-1}\right] =$ `SolovayKitaev4`$(\mathbf{V}_{n-1})$
                $[\mathbf{W}_{n-1}] =$ `GCEnhance`$(\mathbf{U}\widetilde{\mathbf{U}}_{n-1}^{\dagger}, \widetilde{\mathbf{V}}_{n-1})$

$[\widetilde{\mathbf{W}}_{n-1}] = \texttt{SolovayKitaev4}(\mathbf{W}_{n-1})$

or

$[\widetilde{\mathbf{W}}_{n-1}] = \texttt{SolovayKitaev4}(\mathbf{W}_{n-1})$

$[\mathbf{V}_{n-1}] = \texttt{GCEnhance}(\mathbf{U}\widetilde{\mathbf{U}}_{n-1}^{\dagger}, \widetilde{\mathbf{W}}_{n-1})$

$[\widetilde{\mathbf{V}}_{n-1}] = \texttt{SolovayKitaev4}(\mathbf{V}_{n-1})$

until (condition satisfied)

$\widetilde{\mathbf{U}}_n = \widetilde{\mathbf{V}}_{n-1}\widetilde{\mathbf{W}}_{n-1}\widetilde{\mathbf{V}}_{n-1}^{\dagger}\widetilde{\mathbf{W}}_{n-1}^{\dagger}\widetilde{\mathbf{U}}_{n-1}$

end

This new algorithm shown above takes a target unitary gate **U** that is to be approximated as an input. The basic approximation (`basicApprox()`) subroutine (discussed in the previous section) is invoked to obtain $\widetilde{\mathbf{U}}_0$ at the level $n = 1$. In higher levels, the group commutators $\mathbf{V}_{n-1}, \mathbf{W}_{n-1}$are evaluated for the product $\mathbf{U}\widetilde{\mathbf{U}}_0^{\dagger}$ by calling the group commutators decomposition (`GCDecompose()`) subroutine. Subsequently, the optimization step is applied either to enhance **V** or **W** by invoking the group commutator enhancement subroutines (`GCEnhance()`) for **V** and **W.** Transitioning between the two subroutines of enhancing **V** and **W** is stochasticaly modeled in the same manner as mentioned in the previous section.

Chapter-6 Stochastic Group Commutators Enhancement: Results & Analysis

As part of this research, extensive computational and theoretical analyses are performed. A detailed overlook on the tools created for the implementation of the Solovay-Kitaev algorithm, single-level enhancement algorithm and multi-level enhancement algorithm are covered in this chapter. The original Solovay-Kitaev algorithm is compared with the two proposed algorithms for a set of 23 single qubit gates randomly chosen. Multiple sets of simulations are conducted and the results of computational analysis are presented in section 6.1. In section 6.2, theoretical analysis is discussed in details.

6.1 Results of Computational Analysis

All implementations are in MATLAB programming language. MATLAB is specifically suitable for simulation of this nature, as the software has been specifically designed for numerical processes involves extensive use of matrices. In order to test the proposed approach presented in chapter 5, the original Solovay-Kitaev algorithm, the single-level (SLE) and multi-level enhancement (MLE) algorithms are implemented. A set of 23 different randomly generated matrices in SU(2) is generated. Each matrix is approximated using the three different algorithms. The universal gate set used is the set $\mathbf{\Gamma} = \{\mathbf{H}, \mathbf{T}, \mathbf{T}'\}$ and it is used to generate strings up to lengths $l_0 = 16, 17, 18,$ or 19 separately.

The entire simulations are implemented using each of the three different Markov models (see chapter 5) with transitioning probability of ($p = 1$) in model 1, ($p = 0$) in model 2 and ($p = 0.5$) in model 3. While finding approximation for each generated single qubit gate $\mathbf{U} \in \mathrm{SU}(2)$, multiple factors are taken under consideration,

- The error in approximation evaluated as, $\epsilon = \|\mathbf{U} - \widetilde{\mathbf{U}}\|$.
- The length of the approximation string $\widetilde{\mathbf{U}}$ evaluated by the number of **T** gates in $\widetilde{\mathbf{U}}$**.** The reason behind specifically choosing gate **T** is that it has the largest cost while implementation [Kliuchnikov et al. 2012].
- The number of iterations (M) defined as the number of a specific enhanced group commutator subroutine (`GCEnhance()`) is invoked within the SLE or MLE algorithms. Multiple values are considered ($M = 1, 5, 10, 24$).

Multiple set of simulations are performed. The objective of the first set of simulations is to compare the performance of the proposed algorithm (SLE) for different Markov models ($p = 1, 0, 0.5$). In the second set of simulations, the effect of increasing the total number of enhancement steps M in investigated. Finally, the third set compares the performance of the two proposed algorithms (SLE) and (MLE).

In the first set of simulations, the proposed single-level enhancement algorithm (SLE) is implemented over the three Markov models ($p = 1, 0, 0.5$). For each of the randomly generated gates, the error in approximation at each level ($n = 2,3,$ and 4) of SLE algorithm is recorded separately along with number of **T** gates in the approximated string $\widetilde{\mathbf{U}}$. The following table shows the first, second (median), and third quartiles of errors in approximations for each depth, using sequences in the search tree of lengths up to 16, 17, 18, or 19.

Table 6.1 Table listing the error in approximation using the SLE algorithm for depth $\boldsymbol{n}$, initial length $\boldsymbol{l_0}$ applied to three Markov models. The first, second (median), and third quartiles are recorded.

	p	0			0.5			1		
n	l_0	Q1	Med	Q3	Q1	Med	Q3	Q1	Med	Q3
2	16	0.0031	0.0044	0.0071	0.0031	0.0045	0.0071	0.0021	0.0041	0.0074
	17	0.0026	0.0039	0.0050	0.0026	0.0039	0.0054	0.0031	0.0039	0.0049
	18	0.0022	0.0031	0.0041	0.0020	0.0032	0.0044	0.0024	0.0036	0.0054
	19	0.0016	0.0028	0.0038	0.0020	0.0030	0.0039	0.0022	0.0033	0.0039
3	16	0.3105×10^{-3}	0.5096×10^{-3}	0.7779×10^{-3}	0.3941×10^{-3}	0.5024×10^{-3}	0.7711×10^{-3}	0.3720×10^{-3}	0.6002×10^{-3}	0.9857×10^{-3}
	17	0.2909×10^{-3}	0.4299×10^{-3}	0.6565×10^{-3}	0.2644×10^{-3}	0.3909×10^{-3}	0.5442×10^{-3}	0.3571×10^{-3}	0.4918×10^{-3}	0.6065×10^{-3}
	18	0.2341×10^{-3}	0.3083×10^{-3}	0.4937×10^{-3}	0.2367×10^{-3}	0.3809×10^{-3}	0.5497×10^{-3}	0.3886×10^{-3}	0.4776×10^{-3}	0.6019×10^{-3}
	19	0.1276×10^{-3}	0.1686×10^{-3}	0.2418×10^{-3}	0.1456×10^{-3}	0.2296×10^{-3}	0.2974×10^{-3}	0.1703×10^{-3}	0.2296×10^{-3}	0.2890×10^{-3}
4	16	0.0722×10^{-4}	0.1356×10^{-4}	0.2432×10^{-4}	0.1232×10^{-4}	0.1614×10^{-4}	0.2282×10^{-4}	0.1535×10^{-4}	0.2122×10^{-4}	0.3313×10^{-4}
	17	0.0717×10^{-4}	0.1610×10^{-4}	0.1867×10^{-4}	0.0901×10^{-4}	0.1498×10^{-4}	0.2461×10^{-4}	0.0684×10^{-4}	0.1093×10^{-4}	0.1871×10^{-4}
	18	0.0738×10^{-4}	0.1394×10^{-4}	0.1896×10^{-4}	0.0964×10^{-4}	0.1403×10^{-4}	0.1862×10^{-4}	0.0822×10^{-4}	0.1251×10^{-4}	0.1742×10^{-4}
	19	0.0253×10^{-4}	0.0546×10^{-4}	0.0840×10^{-4}	0.0387×10^{-4}	0.0473×10^{-4}	0.0898×10^{-4}	0.0376×10^{-4}	0.0538×10^{-4}	0.1157×10^{-4}

The results shown in table 6.3 are very similar. It is hypothesized that the output when enhancing one matrix of the group commutators is not influenced by the extent to which the other matrix was enhanced. The next figure (figure 6.1) summarizes the results for the three Morkov models and compares them with the original Solovay-Kitaev algorithm.

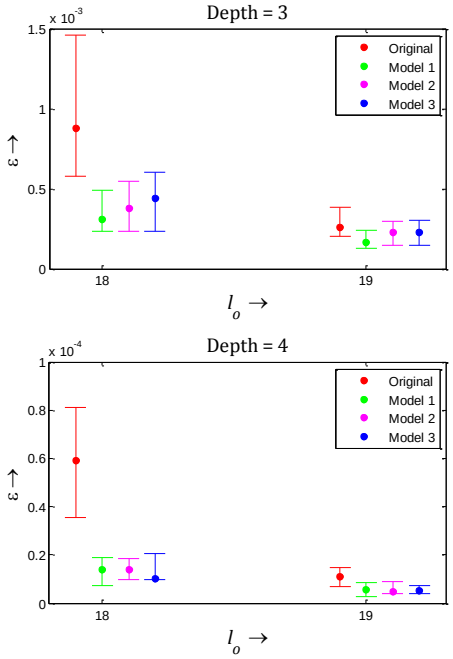


Figure 6.1 The error of the proposed algorithm for different values of the transition probability, which are $\boldsymbol{p = 1,\ 0,\ 0.5}$ and the original Solovay-Kitaev algorithm.

All Markov models show better results than the original Solovay-Kitaev algorithm. However, there is no significant difference between their performances.

Another set of simulations is conducted to investigate the relationship between the length of the approximated string (The number of T gate) and the error in approximation in the different Markov models ($p = 1,0,$ and 0.5) (shown in figures 6.2, 6.3, 6.4 respectively).

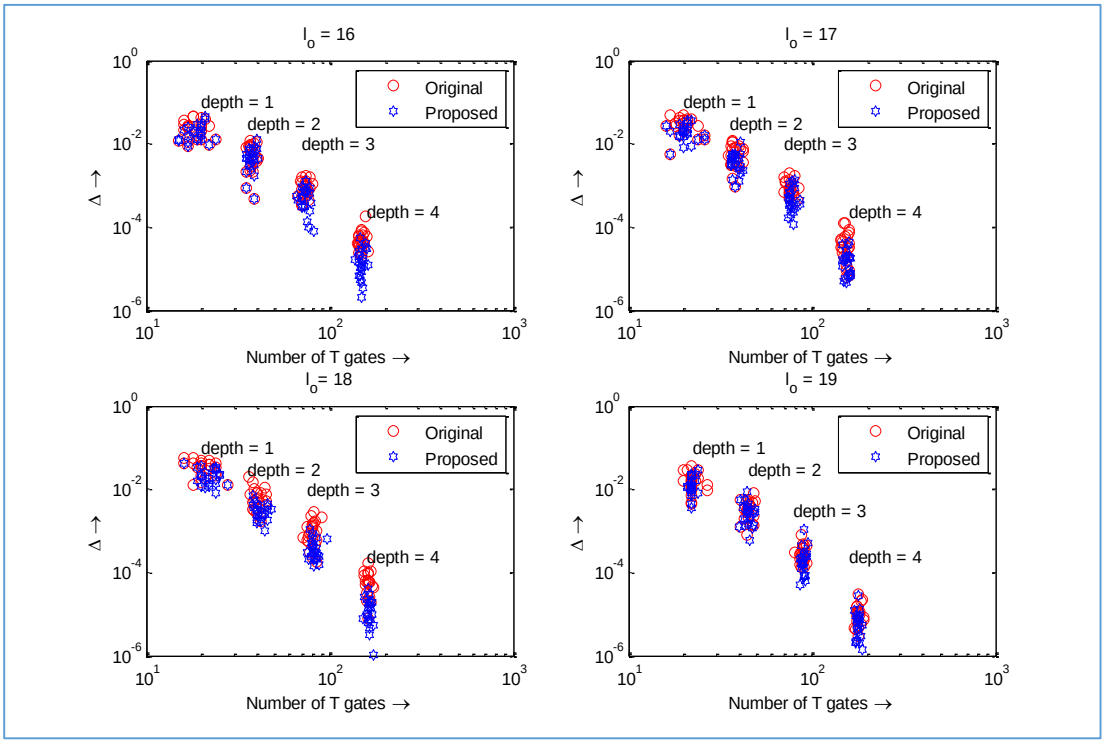


Figure 6.2 Number of T gates versus error in approximation using SLE algorithm of depth $\boldsymbol{n = 1, 2, 3, 4}$ and initial lengths $\boldsymbol{l_0 = 16, 17, 18, 19}$ in Markov model ($\boldsymbol{p = 1}$)

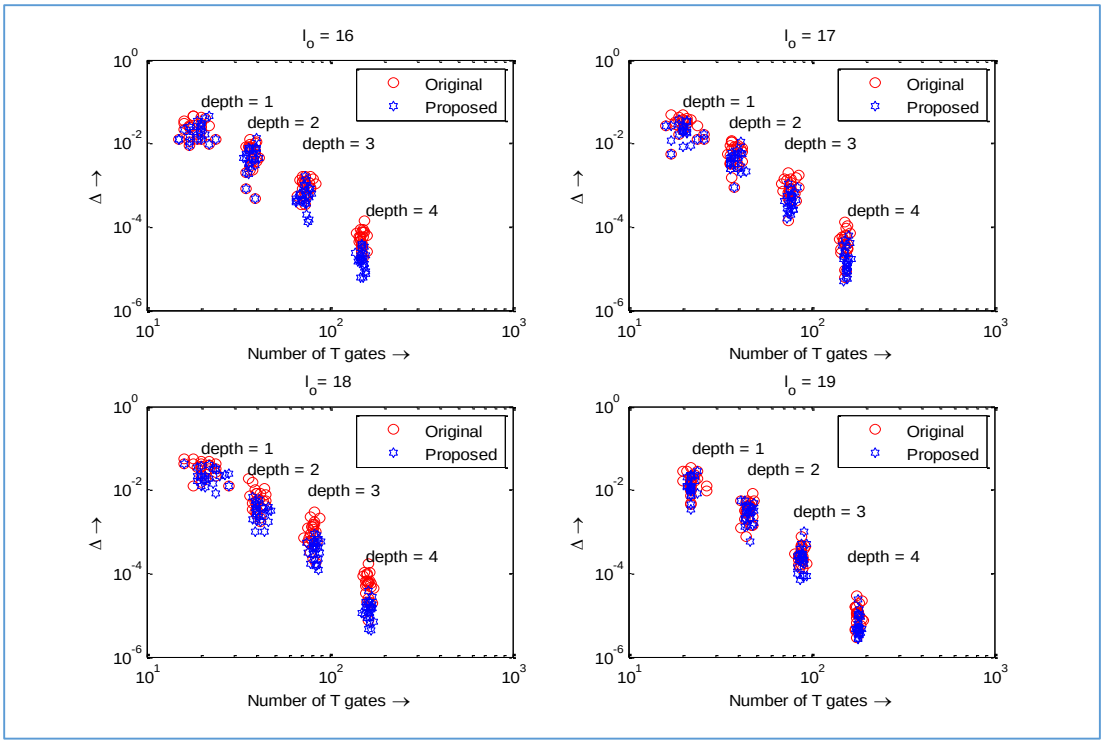


Figure 6.3 Number of T gates versus error in approximation using SLE algorithm of depth $\boldsymbol{n = 1, 2, 3, 4}$ and initial lengths $\boldsymbol{l_0 = 16, 17, 18, 19}$ in Markov model ($\boldsymbol{p = 0}$)

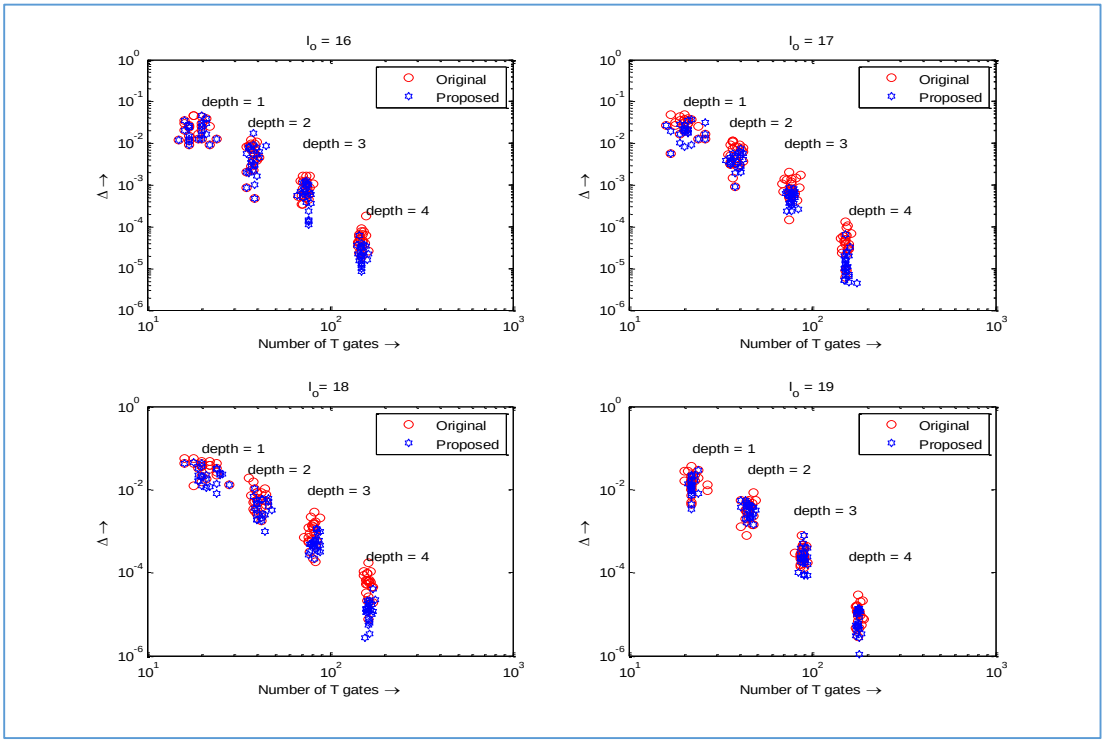


Figure 6.4 Number of T gates versus error in approximation using SLE algorithm of depth $\boldsymbol{n = 1, 2, 3, 4}$ and initial lengths $\boldsymbol{l_0 = 16, 17, 18, 19}$ in Markov model ($\boldsymbol{p = 0.5}$)

The optimization step in the SLE algorithm is performed for a certain number of iterations denoted as M. In the second set of simulations, the relationship between the number of iterations M and the error in approximation is investigated (see table 6.2).

Table 6.2 Table listing the error in approximation using the SLE algorithm of depth $\boldsymbol{n}$, initial length $\boldsymbol{l_0 = 17}$ and number of iteration $\boldsymbol{M}$ applied to the Markov model $\boldsymbol{p = 0.5}$

M	M = 1			M = 5			M = 10			M = 24		
n	Q1	Med	Q3	Q1	Med	Q3	Q1	Med	Q3	Q1	Med	Q3
1	0.0118	0.0176	0.0243	0.0136	0.0193	0.0259	0.0129	0.0191	0.0241	0.0129	0.0191	0.0262
2	0.0030	0.0039	0.0059	0.0031	0.0039	0.0049	0.0030	0.0037	0.0052	0.0030	0.0035	0.0052
3	0.0004	0.0006	0.0010	0.3571×10^{-3}	0.4918×10^{-3}	0.6065×10^{-3}	0.2978×10^{-3}	0.4058×10^{-3}	0.5448×10^{-3}	0.2908×10^{-3}	0.5390×10^{-3}	0.6484×10^{-3}
4	0.1313×10^{-4}	0.2388×10^{-4}	0.3319×10^{-4}	0.0684×10^{-4}	0.1093×10^{-4}	0.1871×10^{-4}	0.0971×10^{-4}	0.1049×10^{-4}	0.2287×10^{-4}	0.0672×10^{-4}	0.0990×10^{-4}	0.2195×10^{-4}

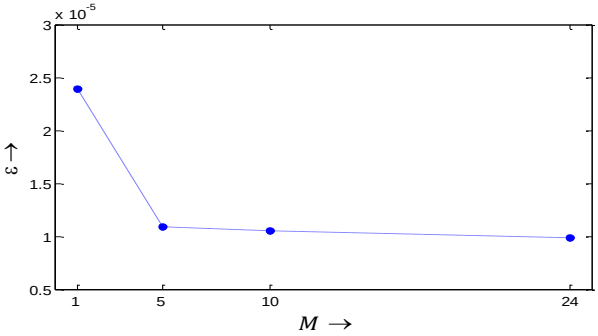


Figure 6.5 Number of iterations versus error in approximation of the SLE algorithm of depth $\boldsymbol{n = 4}$, initial length $\boldsymbol{l_0 = 17}$ using the Markov model $\boldsymbol{p = 0.5}$.

It can be seen from this figure that although an increase in the number of enhancement steps is accompanied by a corresponding decrease in the error, an onset of saturation is seen. This is because $M = 10$ and $M = 24$ show very similar errors despite the extra computational overhead associated with the latter. In contrast increasing M from unity to $M = 5$ yields a significant drop in the error.

The third set of simulations are implemented to study the relationship between the two proposed algorithms, with the optimization in the first one being invoked only at the bottom level of Solovay-Kitaev algorithm (SLE) and when carried out across all levels of the search tree in the other (MLE). The figure 6.6 show that the second version (MLE) yields better results with respect to the accuracy of approximation (ϵ).

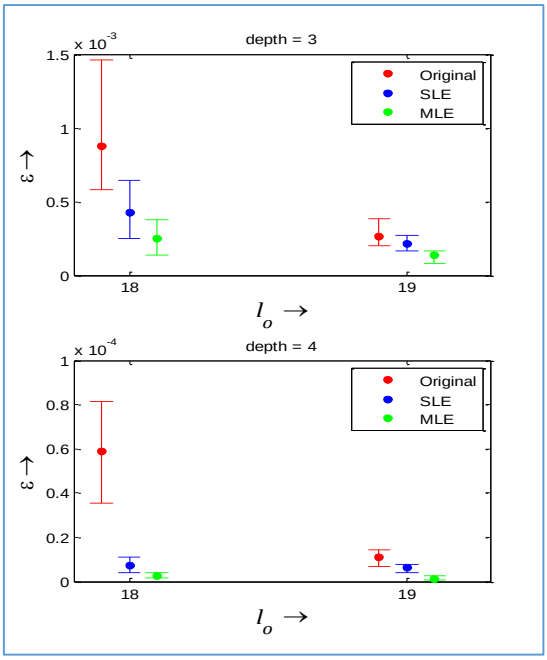


Figure 6.6 Initial length versus error in approximation for depth $\boldsymbol{n = 4}$ using the Markov model ($\boldsymbol{p = 0.5}$) and $\mathbf{M = 24}$.

6.1 Theoretical Analysis

The purpose of this analysis is to provide a quantitative framework for the performance of our algorithm, establishing a theoretical basis for the observations made, as well as justifying the use of the multi-level enhancement algorithm that is addressed in chapter 5. Before proceeding with the theoretical analysis, the assumptions made are first outlined, which are listed below.

Assumption-1: The errors Δ_{n-1}^{v}, Δ_{n-1}^{w} are constant. These errors are determined by the basic approximation subroutine (`basicApprox()`) and follow an unknown random distribution around a mean value. To simplify the derivation, we simply consider the means.

Assumption-2: The errors δ_{n-1}^{v} and δ_{n-1}^{w}, prior to enhancement, are equal to those of the original Solovay-Kitaev algorithm. This is trivially justified since if the group commutator pair is not subject to any enhancement, the algorithm is identical to the original method.

Assumption-3: Expected improvement is invariant with respect to the order in which the enhancement operations are applied. This assumption is based on our earlier observation that the Markov model with different transition probability have no significant difference in performances. Before outlining the next assumption, it should be noted that a matrix of the group commutator pair is enhanced in the subroutine (`GCEnhance()`), which has been formulated as a constraint optimization problem with a maximum allowable limit being placed on the number of optimization iterations. When this underlying optimization is unable to meet the constraint requirements or fails to converge for other extraneous reasons within the stipulated iterations, we term the invocation a failure; otherwise the optimization task is deemed a success.

Assumption-4: The probability of success ρ of the group commutator enhancement subroutine is independent of the step m. Thus we assume that the optimization algorithm's behavior is not dependent on how many times the group commutator matrices have been enhanced before. Although results have not been provided, it should be mentioned that this phenomenon has been supported experimentally. Moreover, although we choose to retain ρ, as our simulations use a sufficiently large number of optimization steps, ρ is very close to unity, and a simpler approach would be to simply let $\rho = 1$ in our calculations.

Let the distance between $\mathbf{W}_{m-1}^{(m)}$ and the identity matrix after m steps be denoted as $\delta_{n-1}^{(m)}$ so that,

$$\delta_{n-1}^{(m)} = \left\| \mathbf{W}_{n-1}^{(m)} - \mathbf{I} \right\|.$$

Assumption-5: The distance $\delta_{n-1}^{(m)}$ decreases asymptotically with increasing m. For simplicity we further assume that the drop is exponential, i.e.

$$\delta_{n-1}^{w\,(m)} = e^{-\alpha m} \delta_{n-1}^{w}.$$

Assumption-6: The probability ρ' of success for a single application of the GC enhancement in $\mathbf{V}_{n-1}$ is identical to that while enhancing $\mathbf{W}_{n-1}$, $\rho' = \rho$.

Assumption-7: At the end of M steps of enhancements, either Δ_{n-1}^{v} or Δ_{n-1}^{w} is zero, but not both. If the enhancement process ends with the unitary matrix $\mathbf{W}_{n-1}$ being enhanced last, then Δ_{n-1}^{v} is not to be considered in the approximation error. Conversely if $\mathbf{V}_{n-1}$ is the last one to be enhanced, then Δ_{n-1}^{w} can be dropped.

The above seven assumptions greatly simplify the derivation of the upper bound on the approximation error ε_n. For simplicity, the subscripts indicating the depth n of the recursion of all quantities have been dropped in the derivations, except when necessary.

In the group commutators, we are given two matrices $\mathbf{V}, \mathbf{W} \in \mathrm{SU}(2)$ such that,

$$\mathbf{V} = \mathbf{I} + \delta^{v},$$
$$\mathbf{W} = \mathbf{I} + \delta^{w}.$$

Let the approximations of $\mathbf{V}, \mathbf{W}$ be $\widetilde{\mathbf{V}}, \widetilde{\mathbf{W}}$, such that,

$$\widetilde{\mathbf{V}}, \widetilde{\mathbf{W}} \in \Gamma^{*} \subset_{D} \mathrm{SU}(2),$$
$$\widetilde{\mathbf{V}} = \mathbf{V} + \Delta^{v},$$
$$\widetilde{\mathbf{W}} = \mathbf{W} + \Delta^{w}.$$

Since $\widetilde{\mathbf{V}} \in \mathrm{SU}(2)$,

$$\begin{aligned} \mathbf{I} &= \widetilde{\mathbf{V}}\widetilde{\mathbf{V}}^{\dagger} \\ &= (\mathbf{V} + \Delta^{v})(\mathbf{V} + \Delta^{v})^{\dagger} \\ &= \mathbf{I} + \Delta^{v}\mathbf{V}^{\dagger} + \mathbf{V}\Delta^{v\dagger} + \Delta^{v}\Delta^{v\dagger}. \end{aligned}$$

Hence,

$$\Delta^v \mathbf{V}^\dagger + \mathbf{V}\Delta^{v\dagger} = -\Delta^v \Delta^{v\dagger}.$$

Likewise,

$$\Delta^w \mathbf{W}^\dagger + \mathbf{W}\Delta^{w\dagger} = -\Delta^w \Delta^{w\dagger}.$$

Consider the term $\widetilde{\mathbf{V}}\widetilde{\mathbf{W}}\widetilde{\mathbf{V}}^\dagger\widetilde{\mathbf{W}}^\dagger$ which is product of the group commutators and their Hermitians in correct order,

$$\widetilde{\mathbf{V}}\widetilde{\mathbf{W}}\widetilde{\mathbf{V}}^\dagger\widetilde{\mathbf{W}}^\dagger = (\mathbf{V}+\Delta^v)(\mathbf{W}+\Delta^w)(\mathbf{V}+\Delta^v)^\dagger(\mathbf{W}+\Delta^w)^\dagger.$$

From direct expansion and neglecting third and fourth order terms, we get the following,

$$\begin{aligned}\widetilde{\mathbf{V}}\widetilde{\mathbf{W}}\widetilde{\mathbf{V}}^\dagger\widetilde{\mathbf{W}}^\dagger = {} & \mathbf{V}\mathbf{W}\mathbf{V}^\dagger\mathbf{W}^\dagger + \Delta^v\mathbf{W}\mathbf{V}^\dagger\mathbf{W}^\dagger + \mathbf{V}\Delta^w\mathbf{V}^\dagger\mathbf{W}^\dagger + \mathbf{V}\mathbf{W}\Delta^{v\dagger}\mathbf{W}^\dagger + \mathbf{V}\mathbf{W}\mathbf{V}^\dagger\Delta^{w\dagger} + \\ & \Delta^v\Delta^w\mathbf{V}^\dagger\mathbf{W}^\dagger + \Delta^v\mathbf{W}\Delta^{v\dagger}\mathbf{W}^\dagger + \Delta^v\mathbf{W}\mathbf{V}^\dagger(\Delta^w)^\dagger + \mathbf{V}\Delta^w\Delta^{v\dagger}\mathbf{W}^\dagger + \mathbf{V}\Delta^w\mathbf{V}^\dagger\Delta^{w\dagger} + \mathbf{V}\mathbf{W}\Delta^{v\dagger}\Delta^{w\dagger}.\end{aligned}$$

Subtracting $\mathbf{V}\mathbf{W}\mathbf{V}^\dagger\mathbf{W}^\dagger$ from both sides are rearranging terms we get,

$$\begin{aligned}\widetilde{\mathbf{V}}\widetilde{\mathbf{W}}\widetilde{\mathbf{V}}^\dagger\widetilde{\mathbf{W}}^\dagger - \mathbf{V}\mathbf{W}\mathbf{V}^\dagger\mathbf{W}^\dagger = {} & \{\Delta^v\mathbf{W}\mathbf{V}^\dagger\mathbf{W}^\dagger + \mathbf{V}\mathbf{W}\Delta^{v\dagger}\mathbf{W}^\dagger\} + \{\mathbf{V}\Delta^w\mathbf{V}^\dagger\mathbf{W}^\dagger + \mathbf{V}\mathbf{W}\mathbf{V}^\dagger\Delta^{w\dagger}\} + \\ & \Delta^v\Delta^w\mathbf{V}^\dagger\mathbf{W}^\dagger + \Delta^v\mathbf{W}\Delta^{v\dagger}\mathbf{W}^\dagger + \Delta^v\mathbf{W}\mathbf{V}^\dagger\Delta^{w\dagger} + \mathbf{V}\Delta^w\Delta^{v\dagger}\mathbf{W}^\dagger + \mathbf{V}\Delta^w\mathbf{V}^\dagger\Delta^{w\dagger} + \mathbf{V}\mathbf{W}\Delta^{v\dagger}\Delta^{w\dagger}. \quad (1)\end{aligned}$$

Since both $\mathbf{V}$ and $\mathbf{W}$ are close to the identity matrix we let $\mathbf{V} = \mathbf{I} + \boldsymbol{\delta}^v$ and $\mathbf{W} = \mathbf{I} + \boldsymbol{\delta}^w$. Now consider the first two terms enclosed with parenthesis in the right hand side of the above Eqn. (1),

$$\begin{aligned}\Delta^v\mathbf{W}\mathbf{V}^\dagger\mathbf{W}^\dagger + \mathbf{V}\mathbf{W}\Delta^{v\dagger}\mathbf{W}^\dagger &= \Delta^v(\mathbf{I}+\boldsymbol{\delta}^w)\mathbf{V}^\dagger(\mathbf{I}+\boldsymbol{\delta}^w)^\dagger \\ &= \Delta^v(\mathbf{I}+\boldsymbol{\delta}^w)\mathbf{V}^\dagger(\mathbf{I}+\boldsymbol{\delta}^{w\dagger}) + \mathbf{V}(\mathbf{I}+\boldsymbol{\delta}^w)\Delta^{v\dagger}(\mathbf{I}+\boldsymbol{\delta}^{w\dagger}) \\ &= -\Delta^v(\Delta^v)^\dagger + \Delta^v\boldsymbol{\delta}^w\mathbf{V}^\dagger + \Delta^v\mathbf{V}^\dagger\boldsymbol{\delta}^{w\dagger} + \Delta^v\boldsymbol{\delta}^w\mathbf{V}^\dagger\boldsymbol{\delta}^{w\dagger} \\ &\quad + \mathbf{V}\boldsymbol{\delta}^w\Delta^{v\dagger} + \mathbf{V}\Delta^{v\dagger}\boldsymbol{\delta}^{w\dagger} + \mathbf{V}\delta^w\Delta^{v\dagger}\boldsymbol{\delta}^{w\dagger}.\end{aligned}$$

Under Assumption 1, and taking the norms on both sides and applying the triangle inequality yields,

$$\left\|\Delta^v\mathbf{W}\mathbf{V}^\dagger\mathbf{W}^\dagger + \mathbf{V}\mathbf{W}\Delta^{v\dagger}\mathbf{W}^\dagger\right\| \leq (\Delta^v)^2 + 4\delta^w\Delta^v + 4\delta^v\delta^w\Delta^v + 2(\delta^w)^2\Delta^v + 2\delta^v(\delta^w)^2\Delta^v.$$

Dropping the third and fourth order terms,

$$\left\|\Delta^v\mathbf{W}\mathbf{V}^\dagger\mathbf{W}^\dagger + \mathbf{V}\mathbf{W}\Delta^{v\dagger}\mathbf{W}^\dagger\right\| \leq (\Delta^v)^2 + 4\delta^w\Delta^v. \quad (2)$$

Similarly, from the third and fourth terms of Eqn. (1) we obain upon dropping higher order terms,

$$\left\|\mathbf{V}\Delta^{w}\mathbf{V}^{\dagger}\mathbf{W}^{\dagger}+\mathbf{V}\mathbf{W}\mathbf{V}^{\dagger}\Delta^{w\dagger}\right\| \leq (\Delta^{w})^{2}+4\delta^{v}\Delta^{w}. \qquad (3)$$

Terms in Eqn. (1) involving two Δs can be simplified easily. For example,

$$\left\|\Delta^{v}\Delta^{w}\mathbf{V}^{\dagger}\mathbf{W}^{\dagger}\right\| = \left\|\Delta^{v}\Delta^{w}(\mathbf{I}+\boldsymbol{\delta}^{v})^{\dagger}(\mathbf{I}+\boldsymbol{\delta}^{w})^{\dagger}\right\|.$$

Dropping higher order terms,

$$\left\|\Delta^{v}\Delta^{w}\mathbf{V}^{\dagger}\mathbf{W}^{\dagger}\right\| = \Delta^{v}\Delta^{w}. \qquad (4)$$

This treatment can be extended readily to the remaining terms of Eqn. (1).

Applying the norm to both sides of Eqn. (1) and using the triangle inequality,

$$\begin{aligned}\left\|\widetilde{\mathbf{V}}\widetilde{\mathbf{W}}\widetilde{\mathbf{V}}^{\dagger}\widetilde{\mathbf{W}}^{\dagger}-\mathbf{V}\mathbf{W}\mathbf{V}^{\dagger}\mathbf{W}^{\dagger}\right\| \leq{}& \left\|\Delta^{v}\mathbf{W}\mathbf{V}^{\dagger}\mathbf{W}^{\dagger}+\mathbf{V}\mathbf{W}\Delta^{v\dagger}\mathbf{W}^{\dagger}\right\|+\left\|\mathbf{V}\Delta^{w}\mathbf{V}^{\dagger}\mathbf{W}^{\dagger}+\mathbf{V}\mathbf{W}\mathbf{V}^{\dagger}\Delta^{w\dagger}\right\| \\ &+\left\|\Delta^{v}\Delta^{w}\mathbf{V}^{\dagger}\mathbf{W}^{\dagger}\right\|+\left\|\Delta^{v}\mathbf{W}\Delta^{v\dagger}\mathbf{W}^{\dagger}\right\|+\left\|\Delta^{v}\mathbf{W}\mathbf{V}^{\dagger}\Delta^{w\dagger}\right\|+\left\|\mathbf{V}\Delta^{w}\Delta^{v\dagger}\mathbf{W}^{\dagger}\right\| \\ &+\left\|\mathbf{V}\Delta^{w}\mathbf{V}^{\dagger}\Delta^{w\dagger}+\mathbf{V}\mathbf{W}\Delta^{v\dagger}\Delta^{w\dagger}\right\|.\end{aligned}$$

Using Eqns. (2) – (4) and replacing all other terms in Eqn. (1) with ones analogous to Eqn. (4), we get,

$$\left\|\widetilde{\mathbf{V}}\widetilde{\mathbf{W}}\widetilde{\mathbf{V}}^{\dagger}\widetilde{\mathbf{W}}^{\dagger}-\mathbf{V}\mathbf{W}\mathbf{V}^{\dagger}\mathbf{W}^{\dagger}\right\| \leq 2(\Delta^{v})^{2}+2(\Delta^{w})^{2}+4\Delta^{v}\Delta^{w}+4\delta^{v}\Delta^{w}+4\delta^{w}\Delta^{v}.$$

This yields to the approximation error,

$$\varepsilon_{n} \leq 2(\Delta^{v}+\Delta^{w})^{2}+4\delta^{v}\Delta^{w}+4\delta^{w}\Delta^{v}. \qquad (5)$$

The upper bound established upon ε_n is significant because it contains terms that are quadratic in Δ and δ, which being small quantities themselves show that ε_n is even smaller, by an order of magnitude.

In the Solovay-Kitaev algorithm, the group commutators are simultaneously replaced with their approximations, which then is used to approximate the given unitary matrix, yielding the approximation error in Eqn. (5). Suppose instead that the above operation is carried out in two stages. At first the $\mathbf{V}_{n-1}$ and $\mathbf{W}_{n-1}$ are obtained. Only the approximation of one of the pair, say $\widetilde{\mathbf{V}}_{n-1}$, is obtained while that of the other deferred until after enhancement. This enhancement step re-establishes the equality $\mathbf{U}\widetilde{\mathbf{U}}_{n-1}^{\dagger}=\mathbf{V}_{n-1}\mathbf{W}_{n-1}\mathbf{V}_{n-1}^{\dagger}\mathbf{W}_{n-1}^{\dagger}$ once again,

$$\mathbf{U}\widetilde{\mathbf{U}}_{n-1}^{\dagger}=\widetilde{\mathbf{V}}_{n-1}\mathbf{W}_{n-1}\widetilde{\mathbf{V}}_{n-1}^{\dagger}\mathbf{W}_{n-1}^{\dagger}.$$

For simplicity, we overload the notation so that the unitary matrix $\mathbf{W}_{n-1}$ now refers to the enhanced one, which is then approximated as $\widetilde{\mathbf{W}}_{n-1}$ leading to the approximation of $\widetilde{\mathbf{U}}_n$. The implication of this corrective step is that the quantity Δ^v no longer plays a role in the approximation error as $\mathbf{W}_{n-1}$ is determined using the approximation $\widetilde{\mathbf{V}}_{n-1}$ instead of $\mathbf{V}_{n-1}$. Hence it can be replaced with zero in Eqn. (5). The new $\mathbf{W}_{n-1}$ is then approximated to give a new approximate sequence $\widetilde{\mathbf{W}}_{n-1}$, providing the normed difference, $\Delta_{n-1}^w = \left\|\widetilde{\mathbf{W}}_{n-1} - \mathbf{W}_{n-1}\right\|$. With these, the approximation error now is,

$$\varepsilon_n \leq 2(\Delta_{n-1}^w)^2 + 4\delta_{n-1}^v \Delta_{n-1}^w. \qquad (6)$$

If the quantity Δ_{n-1}^w appearing in Eqns. (5) and (6) are treated as (approximately) equal it is immediately seen that the new approximation error above is significantly lower than that in Eqn. (5). Additionally, note that δ_{n-1}^w is now lowered from the earlier value in Eqn. (5) because the optimization algorithm explicitly minimizes it. Suppose this minimization causes δ^w to drop by a fraction $\gamma \in (0,1)$ of that in Eqn. (5).

Next, we apply the enhancement on the other group commutator matrix $\mathbf{V}_{n-1}$. The condition that is now satisfied can be seen to be equal to,

$$\mathbf{U}\widetilde{\mathbf{U}}_{n-1}^{\dagger} = \mathbf{V}_{n-1}\widetilde{\mathbf{W}}_{n-1}\mathbf{V}_{n-1}^{\dagger}\widetilde{\mathbf{W}}_{n-1}^{\dagger}.$$

The new $\mathbf{V}_{n-1}$ is now subject to basic approximation to yield a new $\widetilde{\mathbf{V}}_{n-1}$, with

$$\Delta_{n-1}^v = \left\|\widetilde{\mathbf{V}}_{n-1} - \mathbf{V}_{n-1}\right\|.$$

The approximation error at this stage is,

$$\varepsilon_n \leq 2(\Delta_{n-1}^v)^2 + 4\gamma\delta_{n-1}^w \Delta_{n-1}^v. \qquad (7)$$

Note the presence of the factor $\gamma\delta_{n-1}^w$ in the second term to the right. This is because of the earlier enhancement of $\mathbf{W}_{n-1}$. The effect of the minimization shows up. Since $\gamma < 1$, this represents a further reduction in ε_n than that in Eqn. (6), albeit not as much as earlier.

Assumptions 3, 4, and 5 are used next to establish upper bounds on the approximation error ε_n for multiple enhancement steps, instead of a single application to each group commutator matrix that Eqns. (6) and (7) are able to provide. The error after M^v applications of group commutator enhancement on $\mathbf{V}_0$, with m successes and $M^v - m$ failures is given by the following expression,

$$\varepsilon^{(m)} \leq 2(\Delta^v)^2 + 2(\Delta^w)^2 + 4\Delta^v\Delta^w + 4\delta^v\Delta^w + 4e^{-\alpha m}\delta^w\Delta^v.$$

The error after $M^w = M - M^v$ applications of group commutator enhancement on **W**, with m' successes and $M^w - m' = M - (M^v + m')$ failures is given by the following expression,

$$\varepsilon^{(m,m')} \leq 2(\Delta^v)^2 + 2(\Delta^w)^2 + 4\Delta^v\Delta^w + 4e^{-\alpha m'}\delta^v\Delta^w + 4e^{-\alpha m}\delta^w\Delta^v.$$

Using Assumption 6, from the multinomial theorem after M^v applications of GC enhancement to **V** the expected error is,

$$\begin{aligned}
\varepsilon &\leq \sum_{m'=0}^{M^w}\sum_{m=0}^{M^v} \frac{M^v!\ M^w!\,\rho^{m+m'}(1-\rho)^{M-(m+m')}}{m!\,m'!\,(M^v-m)!\,(M^w-m')!}\varepsilon^{(m,m')} \\
&= \sum_{m'=0}^{M^w}\sum_{m=0}^{M^v} \frac{M^v!\ M^w!\,\rho^{m+m'}(1-\rho)^{M-(m+m')}}{m!\,m'!\,(M^v-m)!\,(M^w-m')!}\Big[2(\Delta^v)^2 + 2(\Delta^w)^2 + 4\Delta^v\Delta^w + 4e^{-\alpha m'}\delta^v\Delta^w \\
&\qquad + 4e^{-\alpha m}\delta^w\Delta^v\Big] \\
&= 2(\Delta^v)^2 + 2(\Delta^w)^2 + 4\Delta^v\Delta^w \\
&\qquad + \sum_{m'=0}^{M^w}\sum_{m=0}^{M^v} \frac{M^v!\ M^w!\,\rho^{m+m'}(1-\rho)^{M-(m+m')}}{m!\,m'!\,(M^v-m)!\,(M^w-m')!}\Big[4e^{-\alpha m'}\delta^v\Delta^w\ + 4e^{-\alpha m}\delta^w\Delta^v\Big] \\
&= [2(\Delta^v)^2 + 2(\Delta^w)^2 + 4\Delta^v\Delta^w] + \left[\sum_{m'=0}^{M_w} \frac{M_w!\,p^{m'}q^{M_w-m'}}{m'!\,(M_w-m')!}[4e^{-\alpha m\prime}\delta^v\Delta^w]\right] \\
&\qquad + \left[\sum_{m=0}^{M_v} \frac{M_v!\ p^m q^{M_v-m}}{m!\,(M_v-m)!}[4e^{-\alpha m}\delta^w\Delta^v]\right] \\
&= [2(\Delta^v)^2 + 2(\Delta^w)^2 + 4\Delta^v\Delta^w] + \left[4\delta^v\Delta^w\sum_{m'=0}^{M_w} \frac{M_w!\,(pe^{-\alpha})^{m'}q^{M_w-m'}}{m'!\,(M_w-m')!}\right] \\
&\qquad + \left[4\delta^w\Delta^v\sum_{m=0}^{M_v} \frac{M_v!\ (pe^{-\alpha})^m q^{M_v-m}}{m!\,(M_v-m)!}\right] \\
&= [2(\Delta^v)^2 + 2(\Delta^w)^2 + 4\Delta^v\Delta^w] + \left[4\delta^v\Delta^w(\rho e^{-\alpha} + 1 - \rho)^{M^w}\right] + \left[4\delta^w\Delta^v(\rho e^{-\alpha} + 1 - \rho)^{M^v}\right]. \\
&= 2[(\Delta^v)^2 + (\Delta^w)^2] + 4\left[\Delta^v\Delta^w + \delta^v\Delta^w\gamma^{M^w} + \delta^w\Delta^v\gamma^{M^v}\right]. \qquad (8)
\end{aligned}$$

In the last equation $\gamma = 1 - \rho(1 - e^{-\alpha})$. Since $\alpha > 0$ and $\rho > 0$ it can be seen that $0 < \gamma < 1$.

Assumption 7 leads us to drop either Δ^v or Δ^w from Eqn. (8). Moreover, both cases enable us to drop their product term. Since we are considering an upper bound on the approximation error, we arrive at the following,

$$\varepsilon \leq 2(\max(\Delta^v, \Delta^w))^2 + 4\max\left(\delta^v \Delta^w \gamma^{M^w}, \delta^w \Delta^v \gamma^{M^v}\right). \qquad (9)$$

The multi-level enhancement can be better illustrated if we simplify Eqn. (9) by letting,

$$\Delta_{n-1} = \max(\Delta^v, \Delta^w),$$
$$\delta_{n-1} = \max(\delta^v, \delta^w).$$

This yields to,

$$\varepsilon_n \leq 2\Delta^2 + 4\gamma^M \delta\Delta. \qquad (10)$$

Note that the quantity γ in Eqn. (10) is actually equal to $\gamma^{\frac{1}{M}\min(M_v, M_w)} \approx \sqrt{\gamma}$ of Eqn. (9), since enhancements are applied with equal probabilities to each matrix. Eqn. (10) provides a more compact expression for the approximation error. For the sake of comparison, the equivalent expression for that of the original Solovay-Kitaev algorithm is given below,

$$\varepsilon_n \leq 8\Delta_{n-1}^2 + 8\delta_{n-1}\Delta_{n-1}. \qquad (11)$$

Consider the situation when enhancements are being performed two levels below n, i.e. at $n-2$ in Eqn. (10). At level $n-1$ the algorithm approximation is carried out separately for the two group commutators for a total of M times. The approximation errors ε_{n-1} of this level are the normed difference between each group commutator and its approximation. Hence, during the approximation of $\mathbf{V}_{n-1}$, the approximation error ε_{n-1} is identical to $\Delta_{n-1}^v = \left\|\widetilde{\mathbf{V}}_{n-1} - \mathbf{V}_{n-1}\right\|$. In a similar manner, when approximating $\mathbf{W}_{n-1}$, the approximation error ε_{n-1} becomes $\Delta_{n-1}^w = \left\|\widetilde{\mathbf{W}}_{n-1} - \mathbf{W}_{n-1}\right\|$. Using Eqn. (10), we get,

$$\Delta_{n-1}^v \leq 2\Delta_{n-2}^2 + 4\gamma^M \delta_{n-2}\Delta_{n-2}, \qquad (12)$$

and,

$$\Delta_{n-1}^w \leq 2\Delta_{n-2}^2 + 4\gamma^M \delta_{n-2}\Delta_{n-2}. \qquad (13)$$

Letting $\Delta_{n-1}^v = \Delta_{n-1}^w = \Delta_{n-1}$, if we only use the Solovay-Kitaev algorithm at level n, then from Eqn. (11) we get,

$$\varepsilon_n \leq 8\Delta_{n-1}^2 + 8\delta_{n-1}\Delta_{n-1}$$
$$\leq 8(2\Delta_{n-2}^2 + 4\gamma^M \delta_{n-2}\Delta_{n-2})^2 + 8\delta_{n-1}(2\Delta_{n-2}^2 + 4\gamma^M \delta_{n-2}\Delta_{n-2}).$$

Retaining only cubic terms,

$$\varepsilon_n \leq 16\delta_{n-1}\Delta_{n-2}^2 + 32\gamma^M \delta_{n-1}\delta_{n-2}\Delta_{n-2}. \qquad (14)$$

Next consider another situation where a smaller number $M_{n-1} < M$ of enhancement steps are applied to level $n-1$ and the remaining $M_n = M - M_{n-1}$ are used above, at level n. In this case, the approximation error can be obtained by using the value of Δ_{n-1} obtained through Eqns. (12) and (13), with M_{n-1} in place of M, in Eqn. (10),

$$\varepsilon_n \leq 2(2\Delta_{n-2}^2 + 4\gamma^{M_{n-1}}\delta_{n-2}\Delta_{n-2})^2 + 4\gamma^{M_n}\delta_{n-1}(2\Delta_{n-2}^2 + 4\gamma^{M_{n-1}}\delta_{n-2}\Delta_{n-2}).$$

As before, discarding all but the lowest order terms,

$$\varepsilon_n \leq 8\gamma^{M_n}\delta_{n-1}\Delta_{n-2}^2 + 16\gamma^{M_{n-1}}\gamma^{M_n}\delta_{n-1}\delta_{n-2}\Delta_{n-2}. \qquad (15)$$

It is immediately seen that the earlier strategy of distributing the enhancement steps across multiple levels as is done in the second version of the proposed algorithm, produces a lower approximation error.

Chapter 7 - Bi-Criteria Group Commutator Enhancement

In this chapter bi-criteria group commutator optimization is presented. The proposed Incremental Constraint Relaxation method improves the precision of the original Solovay-Kitaev algorithm by subjecting the pair of group commutators provided by the decomposition subroutine, to one or more rounds of explicit optimization. In section 7.1, introduction about bi-criteria optimization is discussed along with the primarily used methods in this area, the normalized boundary intersection and the normalized normal constraint methods. In section 7.2, the bi-criteria optimization of the group commutators is presented. Finally, the proposed Incremental Constraint Relaxation approach is discussed in section 7.3.

7.1 Bi-Criteria Optimization

Bi-objective optimization problem is an optimization problem that involves two objective functions [Das el al. 2010]. Optimal solutions in the trade-off surface are referred to as Pareto optimal solutions. All Pareto optimal solutions are considered equally good. In other words, if a solution is better than another in terms of one of the two objectives, it must necessarily be worse in comparison to the latter in terms of the other objective. The optimal solution with respect to each objective is obtained when each is minimized independently (called anchor points). The line joining them is called the Utopia line (see fig 7.1). Several approaches for generating sets of Pareto solutions are addressed in the literature with the normalized boundary intersection (NBI) and the normalized normal constraint (NC) method being the primarily used methods.

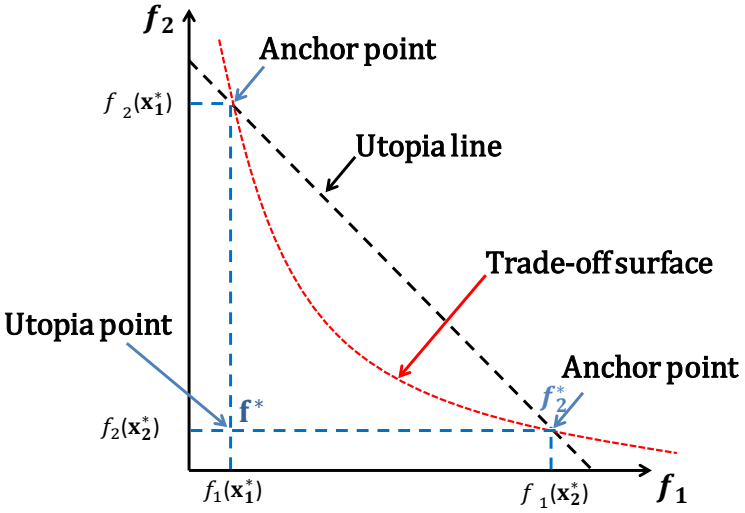


Figure 7.1 Bi criteria optimization

The normalized boundary intersection (NBI) imposes additional constraints (equality constraints) in the objective space [Das and Dennis 1998]. In addition to existing design variable, a new variable λ is added (figure 7.2) which denotes the distance away from the Utopia line toward the Utopia point. Maximizing λ causes movement towards the Pareto optimal surface.

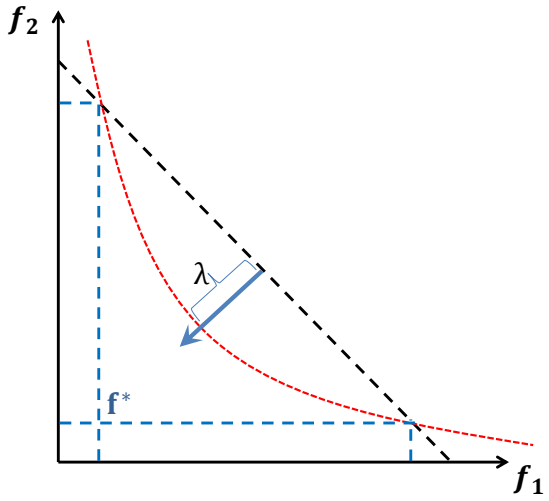


Figure 7.2 Normalized boundary intersection for bi-objective case

The normalized normal constraint (NC) approach reformulates a bi-criteria optimization problem into a single criterion optimization so that it can make use of any known optimization algorithm [Messac et al. 2003 and Messac, A., & Mattson, C. A. 2004]. An inequality constraint is applied perpendicular to the Utopia line in the normalized space. One of the two objective functions is then optimized, depending on the direction of the inequality constraint; this limits the feasible domain. A general graphical perspective of the normal constraint method for a three objective case is shown in figure 7.3.

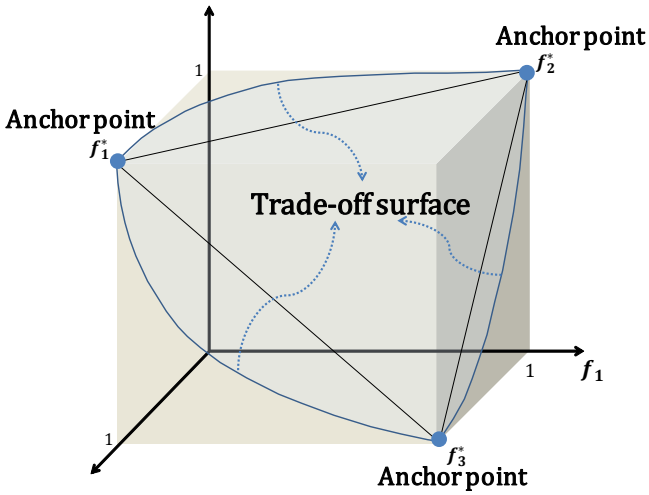


Figure 7.3 Normalized normal constraint method for a three-objective case

7.2 Bi-Criteria Optimization of Group Commutators

The proposed method improves the precision of the original Solovay-Kitaev algorithm discussed earlier by subjecting the pair of group commutators provided by the decomposition subroutine, to one or more rounds of explicit optimization. In this version of the new approach, this optimization takes place at $n = 1$, the calling level for basic approximation. The purpose of optimizing a group commutator is to enhance it so that the error ε_1 can be reduced. Although some of the results shown later pertain to the group commutators optimized only at $n = 1$, the following description holds for any level n. The bi-criteria optimization algorithm which is discussed next is similar to the Normalized Boundary Intersection (NBI) and more so to the Normalized Normal Constraint (NC) approaches and for vector optimization [Messac et al. 2003].

The objective of the algorithm is to reduce the error in approximation ε_n which is evaluated by the distance between the desired gate $\mathbf{U}$ and its approximation $\widetilde{\mathbf{U}}_n$ such that,

$$\varepsilon_n = \left\| \widetilde{\mathbf{U}}_n - \mathbf{U}_n \right\|.$$

The error ε_n depends on the two quantities Δ_{n-1} and δ_{n-1} associated with the group commutator pairs $\mathbf{V}_{n-1}$ and $\mathbf{W}_{n-1}$, such that,

$$\Delta_{n-1} = \left\| \widetilde{\mathbf{V}}_{n-1} - \mathbf{V}_{n-1} \right\| = \left\| \widetilde{\mathbf{W}}_{n-1} - \mathbf{W}_{n-1} \right\|,$$
$$\delta_{n-1} = \|\mathbf{I} - \mathbf{V}_{n-1}\| = \|\mathbf{I} - \mathbf{W}_{n-1}\|.$$

The algorithm optimizes the group commutator pair in order to enhance it so that it gets closer to the identity matrix $\mathbf{I}$. In other words, the bi-criteria optimization aims to reduce δ_{n-1} and subsequently, reduces the error in approximation ε_n so that $\delta_{n-1}^{v} = \|\mathbf{V}_{n-1} - \mathbf{I}\|$ and $\delta_{n-1}^{w} = \|\mathbf{W}_{n-1} - \mathbf{I}\|$ together, form one of the two criteria for optimization. Instead of being treated separately, δ_{n-1}^{v} and δ_{n-1}^{w} will be subject to alternate optimizations as will be described shortly. Under the assumption that lower level calls to either the Solovay-Kitaev algorithm or to the basic approximation subroutine yield approximate matrices that are independent of any procedures that the group commutators may subject to prior to the subroutine call, the quantity Δ_{n-1} remains outside the scope of the bi-criteria optimization.

The other criterion that is considered during the bi-criteria optimization algorithm is the approximation error ε_n itself. However, this criterion is formulated as a constraint within the new bi-

criteria optimization framework proposed in this research. This is a major difference from NBI and NC which are initialized simultaneously with several points that usually lie on a certain hyperplane located at a distance away from the origin. The optimization algorithm then proceeds to move the initial points towards the origin, until the Pareto set is reached. The points can be updated either in parallel or sequentially within NNC and NBI. In contrast, the proposed Incremental Constraint Relaxation (ICR) is initialized with the group commutators obtained from the decomposition subroutine (`GCDecomposition()`) and initial approximation (`basicApprox()`).

This initial point is shown (red square) in Figure 7.4 which illustrates how the first increment of the proposed ICR approach is done. For the purpose of this explanation, assume that the calling program is at depth $n = 1$. The group commutators are $\mathbf{V}_0$ and $\mathbf{W}_0$ and the approximation error ε_0 is,

$$\varepsilon_0 = \left\| \mathbf{U}\tilde{\mathbf{U}}^\dagger - \tilde{\mathbf{V}}_0\tilde{\mathbf{W}}_0\tilde{\mathbf{V}}_0^\dagger\tilde{\mathbf{W}}_0^\dagger \right\|.$$

This is also the approximation error that the Solovay-Kitaev method would provide. Here the matrices $\tilde{\mathbf{V}}_0$ and $\tilde{\mathbf{W}}_0$ are approximating sequences in Γ^* that are closest to $\mathbf{V}_0$ and $\mathbf{W}_0$, the group commutators of the decomposition $\mathbf{U}\tilde{\mathbf{U}}^\dagger = \mathbf{V}_0\mathbf{W}_0\mathbf{V}_0^\dagger\mathbf{W}_0^\dagger$.

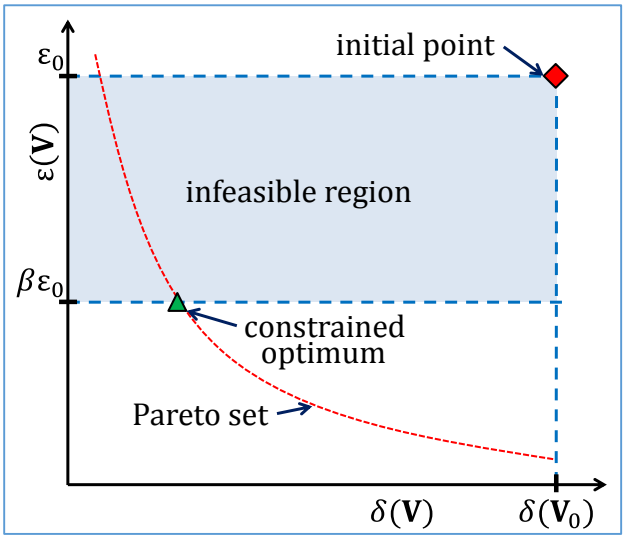


Figure 7.4 Problem formulation of a single increment of Incremental Constraint Relaxation optimization.

Next, ICR proceeds to fine tune the group commutators in increments, with $\mathbf{V}$ and $\mathbf{W}$ being refined alternately. Suppose $\mathbf{V}$ is the first one of the pair that is refined first. The proposed ICR method fixes the other with the approximate sequence $\tilde{\mathbf{W}}_0$ and optimizes $\mathbf{V}$ so that it is brought as close as possible to the identity matrix, while simultaneously ensuring that the resulting error does not exceed a fraction $\beta \in (0,1)$ of ε_0. The optimization problem is formulated as shown below,

Minimize,

$$\delta(\mathbf{V}) \triangleq \|\mathbf{V} - \mathbf{I}\|.$$

Subject to,

$$\varepsilon(\mathbf{V}) \triangleq \left\|\mathbf{U}\tilde{\mathbf{U}}^{\dagger} - \mathbf{V}\tilde{\mathbf{W}}_0\mathbf{V}^{\dagger}\tilde{\mathbf{W}}_0^{\dagger}\right\| \leq \beta\varepsilon_0,$$

$$\mathbf{V} \in \mathrm{SU}(2).$$

As a result of the optimization, the new value of $\mathbf{V}$ that is obtained lies in the Pareto set (green triangle in Figure 7.4). This is the first increment of ICR. How the solution is alternately incremented by the proposed ICR algorithm is described in the next section.

7.3 Incremental Relaxation

The Incremental relaxation algorithm is shown below,

$[\tilde{\mathbf{U}}_n] = \texttt{SolovayKitaev2}(\mathbf{U}, n)$
`if` $n == 1$
    $[\tilde{\mathbf{U}}_0] = \texttt{basicApprox}(\mathbf{U})$
    $[\mathbf{V}_0, \mathbf{W}_0] = \texttt{GCDecompose}(\mathbf{U}\tilde{\mathbf{U}}_0^{\dagger})$
    $[\tilde{\mathbf{V}}_0] = \texttt{basicApprox}(\mathbf{V}_0)$
    $[\tilde{\mathbf{W}}_0] = \texttt{basicApprox}(\mathbf{W}_0)$
    `Repeat`
        `%Optimize W`
        $[\mathbf{W}_0] =$ `ICR(`$\mathbf{U}\tilde{\mathbf{U}}_0^{\dagger}, \tilde{\mathbf{V}}_0$`)`
            $[\tilde{\mathbf{W}}_0] =$ `basicApprox(`$\mathbf{W}_0$`)`
        `%Optimize V`
            $[\mathbf{V}_0] =$ `ICR(`$U\tilde{U}_0^{\dagger}, \tilde{\mathbf{W}}_0$`)`
            $[\tilde{\mathbf{V}}_0] =$ `basicApprox(`$\mathbf{V}_0$`)`
    `until (condition satisfied)`
`else`
    $[\tilde{\mathbf{U}}_{n-1}] = \texttt{SolovayKitaev2}(\mathbf{U}, n-1)$
    $[\mathbf{V}_{n-1}, \mathbf{W}_{n-1}] = \texttt{GCDecompose}(\mathbf{U}\tilde{\mathbf{U}}_0^{\dagger})$
    $[\tilde{\mathbf{V}}_{n-1}] = \texttt{SolovayKitaev2}(\mathbf{V}_{n-1}, n-1)$
    $[\tilde{\mathbf{W}}_{n-1}] = \texttt{SolovayKitaev2}(\mathbf{W}_{n-1}, n-1)$
`end`

$$\tilde{\mathbf{U}}_n = \tilde{\mathbf{V}}_{n-1}\tilde{\mathbf{W}}_{n-1}\tilde{\mathbf{V}}_{n-1}^{\dagger}\tilde{\mathbf{W}}_{n-1}^{\dagger}\tilde{\mathbf{U}}_{n-1}$$

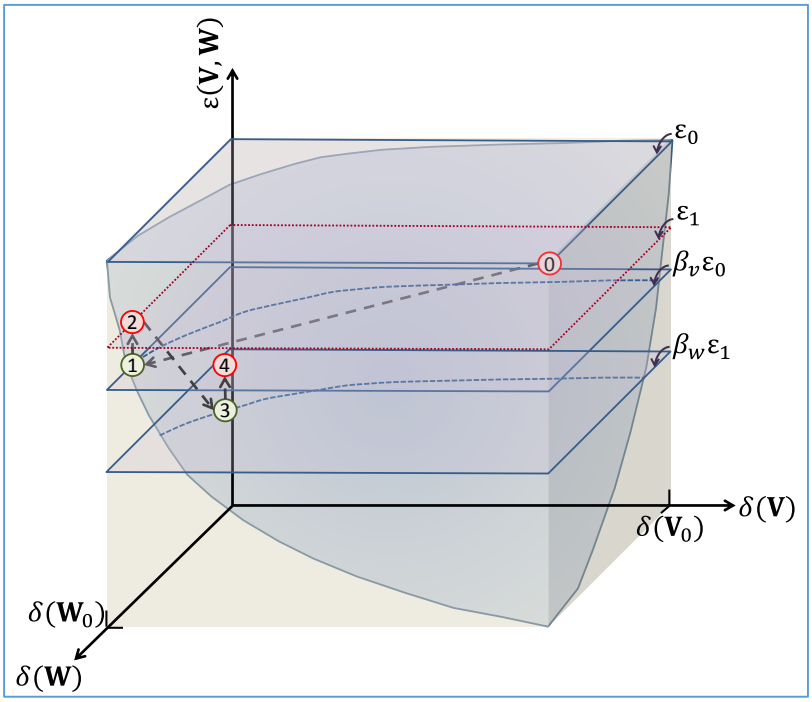


Figure 7.5 Schematic of the proposed Incremental Constraint Relaxation optimization showing the first four steps. The points shown as green circles are those in the Pareto set, while those not in the Pareto set are colored red.

Optimizing group commutator $\mathbf{V}$ while the other is replaced with its approximation $\widetilde{\mathbf{W}}_0$ as has been detailed earlier, results in the point labeled '1' in Figure 7.5 (green triangle in Figure 7.4). The new error is $\beta\varepsilon_0$ (or $\beta_v\varepsilon_0$ in Figure 7.5 where β_v and β_w are the earlier fraction β, with subscripts used for clarity). The optimized $\mathbf{V}$ is next approximated with the new sequence $\widetilde{\mathbf{V}}_0$ resulting in the approximation error ε_1. Since ε_1 is more than $\beta_v\varepsilon_0$ due to the approximation the new point, labeled '2' is not in the Pareto set.

Next, the other group commutator $\mathbf{W}$ is subject to optimization to minimize its deviation away from $\mathbf{I}$. The constraint being imposed ensures that following this increment, the error does not exceed a fraction β_w of ε_1 which was previously obtained. This yields the point labeled '2' in figure 7.5, which is in the Pareto set. Replacement of the new value of $\mathbf{W}$ with its approximation $\widetilde{\mathbf{W}}_0$ moves the solution away from the Pareto set, to the new location labeled '4' in the figure 7.5. In this manner, the ICR approach alternately minimizes the errors $\delta(\mathbf{V})$ and $\delta(\mathbf{W})$ with each optimization followed by an approximation step for several iterations M.

Chapter 8 - Bi-Criteria Group Commutators Enhancement: Results & Analysis

As part of this research, extensive computational analysis is performed using MATLAB programming language to study the behavior of the incremental constraint relaxation approach proposed in chapter 7. Multiple sets of simulations are conducted and the results of computational analysis are presented in this chapter.

8.1 Results of Computational Analysis

Multiple sets of simulations are performed here which follow similar scenario to that presented in chapter 6. In order to study the performance of the new bi-criteria optimization approach, incremental constraint relaxation (ICR), a set of 23 different randomly generated matrices in SU(2) is generated. The universal gate set used is the set $\mathbf{\Gamma} = \{\mathbf{H}, \mathbf{T}, \mathbf{T}'\}$ and it is used to generate strings up to lengths $l_0 = 16, 17, 18,$ or 19 separately.

The entire simulations are implemented using ICR. While finding approximation for each generated single qubit gate $\mathbf{U} \in \mathrm{SU}(2)$, multiple factors are taken under consideration,

- The error in approximation evaluated as,
  $$\epsilon = \left\|\mathbf{U} - \widetilde{\mathbf{U}}\right\|.$$
- The number of iterations $M_v(M_w)$ defined as the number of a specific enhanced group commutator ($\mathbf{V}$ or $\mathbf{W}$) is enhanced. Multiple values are considered ($M = 1,2,3,4,8,16$).
- The fractions β_v and β_w are the fraction β defined in the optimization constraint,
  $$\left\|\mathbf{U}\widetilde{\mathbf{U}}^{\dagger} - \mathbf{V}\widetilde{\mathbf{W}}_0\mathbf{V}^{\dagger}\widetilde{\mathbf{W}}_0^{\dagger}\right\| \leq \beta\varepsilon_0.$$
  Different scenarios of β are considered,
  - Linearly spaced values between 0 and 1 of lengths $M_v(M_w)$.
  - Several Beta-binomial distributions with different values of α and β in the probability mass function of the Beta-binomial distribution are considered.

    In general, if X~$\mathbf{Bin}(n,p)$, and p~$\mathbf{Beta}(\alpha,\beta)$, then,

$$P(\mathrm{X} = k) = \binom{n}{k}\frac{\mathbf{Beta}(k+\alpha, n-k+\beta)}{\mathbf{Beta}(\alpha,\beta)}.$$

The following figure shows the different Beta-binomial distribution distributions tested.

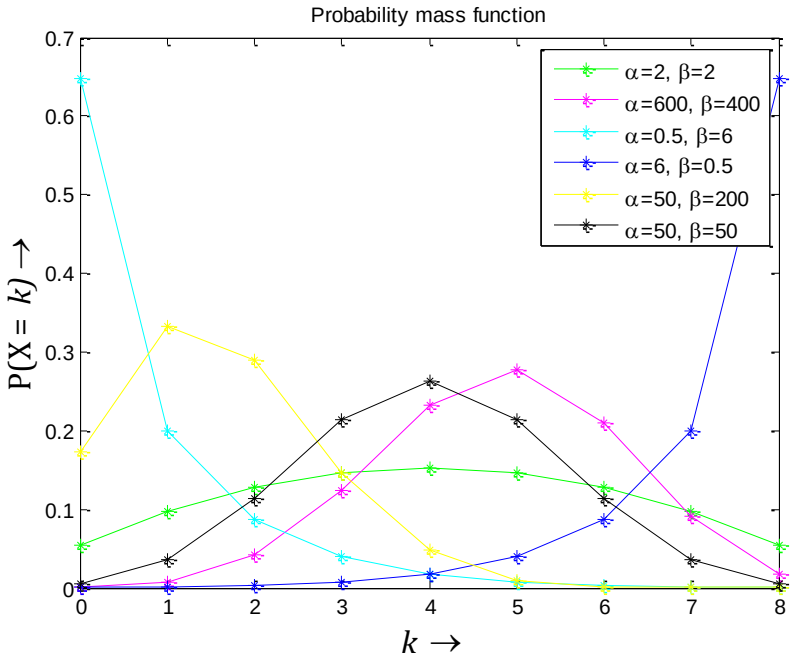


Figure 8.1 The probability mass function of the Beta-binomial distribution

Multiple set of simulations are performed. In the first set of simulations, the performance of the proposed approach under different distributions is studied. The objective of the second set of simulations is to compare the performance of the bi-criteria optimization proposed algorithm (ICR) with the previous approaches presented in chapter 5 (Single level enhancement (SLE) algorithm for the different Markov models ($p = 1, 0, 0.5$)) as well as the original Solovay-Kitaev algorithm.

In the first set of simulations, linearly spaced values of $\beta \in [0,1]$ are tested. For different number of iteration to enhance the group commutators **V** and **W** (denoted as M_v and M_w) which can be also seen as the step size of β. The ICR algorithm is performed with different cases of M_v, M_w. The cases considered and shown in figure 8.3 are $(4 \times 4, 8 \times 2, 16 \times 1)$

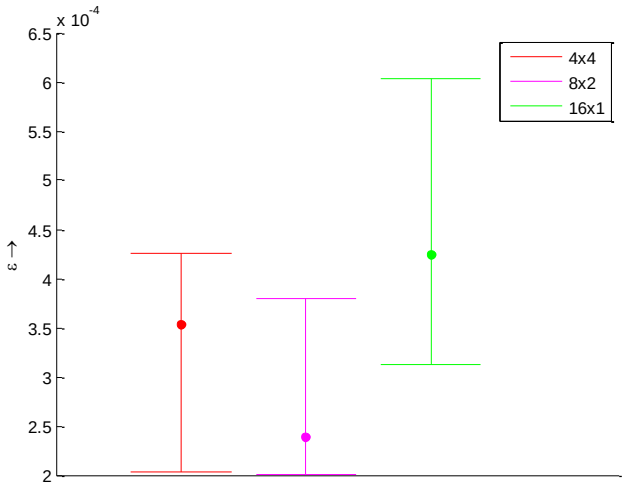


Figure 8.2 The performance of ICR optimization with linearly spaced values of $\boldsymbol{\beta} \in [\mathbf{0}, \mathbf{1}]$. Different number of iteration to enhance the group commutators $\boldsymbol{V}$ and $\boldsymbol{W}$ $(\boldsymbol{M_v} \times \boldsymbol{M_w})$ are tested.

The proposed ICR algorithm is also implemented when β follows different Beta-binomial distributions. The following figures show a comparison between ICR with different distributions and a comparison between ICR and the originalSolovay-Kitaev algorithm (figure 8.3 and 8.4 respectively). It can be seen that the ICR algorithm gives better accuracy in approximation than the original Solovay-Kitaev algorithm.

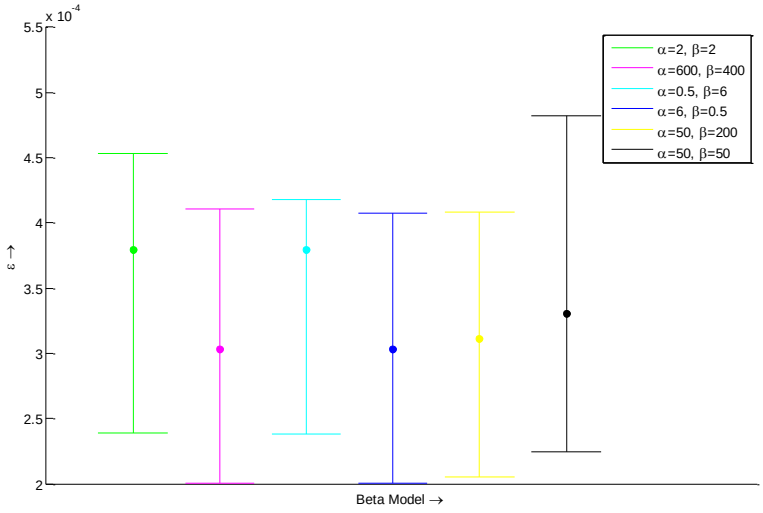


Figure 8.3 The performance of ICR optimization under different Beta-binomial distributions.

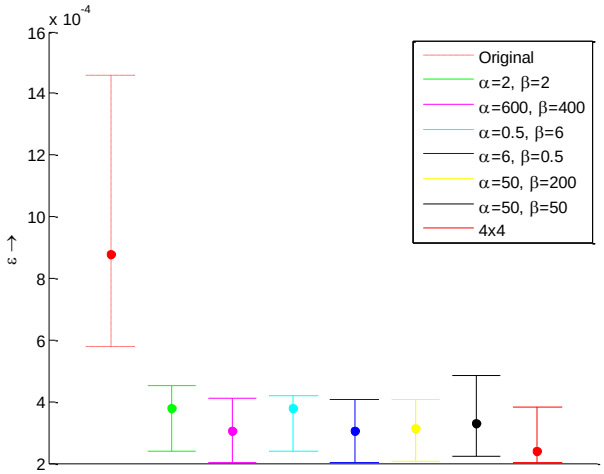


Figure 8.4 The performance of ICR optimization when $\boldsymbol{\beta}$ follows different Beta-binomial distributions and when linearly spaced. The original Solovay-Kitaev algorithm performance is also provided.

In the second set of simulations, the proposed incremental constraint relaxation approach (ICR) is implemented over the randomly generated matrices in SU(2). For each gates, the error in approximation at each level ($n = 3$, and 4) of ICR algorithm is recorded separately. Figure 8.2 shows the comparison between the proposed ICR algorithm with different Markov models ($p = 1, 0, 0.5$) implemented by the SLE algorithm. The proposed Incremental Constraint Relaxation method yields better results with respect to the accuracy of approximation than the other Markov models.

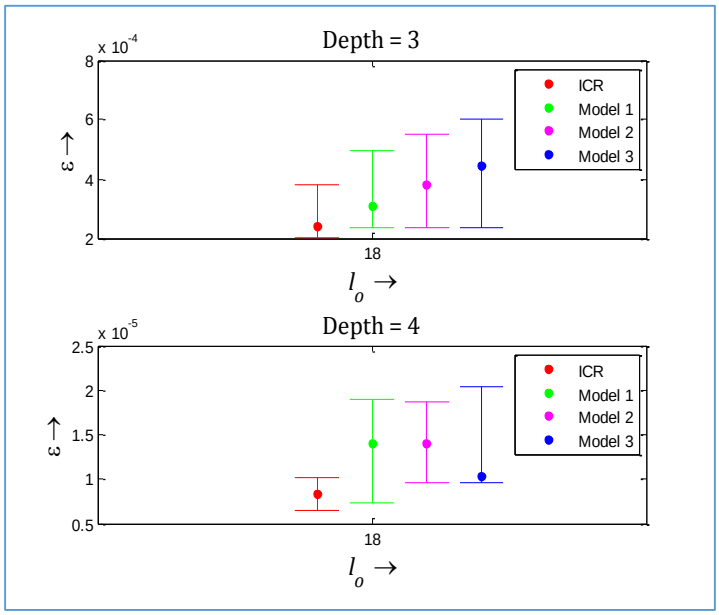


Figure 8.5 The performance of ICR compared to Single level enhancement (SLE) algorithm for the different Markov models ($\boldsymbol{p = 1,\ 0,\ 0.5}$)

Chapter 9 - Conclusion and Future Work

In this chapter, we summarize the contributions of this dissertation and discuss future research directions.

9.1 Summary of Key Contributions

The main contribution of this research is in integrating conventional optimization procedures within the Solovay-Kitaev algorithm. Specifically, optimization is incorporated within the group commutator decomposition, so that a more optimal pair of group commutators is obtained. As the degree of precision of the synthesized gate is explicitly minimized by means of this optimization procedure, the proposed algorithms allow for more accurate quantum gates to be synthesized than what the original Solovay-Kitaev algorithm achieves.

Two specific directions of research have been studied. Firstly, optimization of the group commutator decomposition has been modeled as a constraint optimization problem with equality constraints. Since there is a pair of matrices acting as the group commutators, a separate procedure is invoked for each, which is done in a stochastic manner. Two versions of the new algorithm are examined, with the optimization in the first version being invoked only at the bottom level of Solovay-Kitaev algorithm (Single-level enhancement algorithm) and when carried out across all levels of the search tree (Multi-level enhancement algorithm) in the next.

Extensive simulations show that the multi-level enhancement version yields better approximations than the single-level version, even when the net amount of computation involved is limited to that of the other version. The theoretical analysis of the proposed algorithm is able to provide a more formal, quantitative explanation of why the proposed method is able to improve over the original Solovay-Kitaev algorithm. It also is able to explain the saturation observed in figure 6. And why the multi-level enhancement (MLE) algorithm provides better results than the single-level enhancement (SLE) algorithm despite equivalent computation times.

The next phase of research relaxed the equality constraint in the previous approach and with relaxation, a bi-criteria optimization is proposed. This optimization algorithm is new and has been

devised primarily when the objective needs to be relaxed in different stages. This bi-criteria approach is able to provide more accurate synthesis than the previous approach.

9.2 Future Work

Some possible future research directions are provided in this section. Extensions to our work proposed in chapters 5 and 7 are mentioned next,

- The algorithms proposed are implemented only to synthesis single-qubit gates. A generalization into multiple qubits can be investigated. ($\mathbf{U} \in \mathrm{SU}(2^n)$).

- All group commutators can go through enhancement depending on who gives better errors using some heuristic method [Nilsson 2014]. This procedure creates an unbalanced tree instead of the currently used tree. It can also take under account the number of T gates. Best first search, branch and bound and heuristic deepening can be used here.

- Integration of the recursive search space expansion technique (SSE) into the proposed algorithms [Pham et al. 2013]. The SSE method increases the space that is searched at the initial level of recursion in the original algorithm, and consequently, reduces the number of levels of recursion required by the modified algorithm to reach a given level of accuracy.

- Other optimization algorithms are fine tuned for this application including evolutionary algorithms and other stochastic optimization techniques [Zhou et al. 2011].

- Application of the proposed algorithms to topological quantum computing. It essentially encodes qubit states in topological properties of a system, which are much more robust [Pachos 2012].

References

Aliferis P., Gottesman D., and Preskill J. (2006). "Quantum accuracy threshold for concatenated distance-3 codes," Quantum Information and Computation, vol. 6, pp. 97–165

Al-Rabadi, A. N. (2004). *Reversible logic synthesis: From fundamentals to quantum computing*. Springer Science & Business Media.

Amy, M., Maslov, D., Mosca, M., & Roetteler, M. (2013). A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, *32*(6), 818-830, arXiv:1206.0758.

Arabzadeh, M., Zamani, M. S., Sedighi, M., & Saeedi, M. (2013). Depth-optimized reversible circuit synthesis. *Quantum information processing*, *12*(4), 1677-1699.

Barenco, A., Bennett, C. H., Cleve, R., DiVincenzo, D. P., Margolus, N., Shor, P., ... & Weinfurter, H. (1995). Elementary gates for quantum computation.*Physical Review A*, *52*(5), 3457.

Bell, J. S. (2004). *Speakable and Unspeakable in Quantum Mechanics: Collected Papers on Quantum Philosophy*. Cambridge University Press, Cambridge, United Kingdom.

Bennett, C. H., & DiVincenzo, D. P. (2000). Quantum information and computation. *Nature*, 404(6775), 247-255.

Bocharov and K. M. Svore, (2012)."Resource-optimal single-qubit quantum circuits," Physical Review Letters, vol. 109, p. 190501, arXiv:1206.3223.

Bocharov, A., Gurevich, Y., & Svore, K. M. (2013). Efficient decomposition of single-qubit gates into V basis circuits. *Physical Review A*, *88*(1), 012313.

Bocharov, A., Roetteler, M., & Svore, K. M. (2014). Efficient synthesis of probabilistic quantum circuits with fallback. *arXiv preprint arXiv:1409.3552*.

Bocharov, A., Roetteler, M., & Svore, K. M. (2015). Efficient synthesis of universal repeat-until-success quantum circuits. Physical Review Letters.

Bohm, A., & Loewe, M. (1986). *Quantum mechanics: foundations and applications* (p. 609). New York: Springer-Verlag.

Brin, S. (1995). Near neighbor search in large metric spaces. In *21th International Conference on Very Large Data Bases (VLDB 1995)*, 11-15.

Chow, J. M., Gambetta, J. M., Corcoles, A. D., Merkel, S. T., Smolin, J. A., Rigetti, C., ... & Steffen, M. (2012). Universal quantum gate set approaching fault-tolerant thresholds with superconducting qubits. *Physical review letters*, *109*(6), 060501.

Childs, A. M., & Van Dam, W. (2010). Quantum algorithms for algebraic problems. *Reviews of Modern Physics*, *82*(1), 1.

Dawson C. M., Nielsen M. A., (2006). "The Solovey-Kitaev Algorithm", *Quantum Information & Computation,* Vol. 6, No. 1, pp. 81–95

Das, I., & Dennis, J. E. (1998). Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization*, *8*(3), 631-657.

Deutsch, D., Barenco, A., & Ekert, A. (1995). Universality in quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, *449*(1937), 669-677.

De Vos, A. (2011). *Reversible computing: fundamentals, quantum computing, and applications*. John Wiley & Sons.

DiVincenzo, D. P. (1995). Two-bit gates are universal for quantum computation. *Physical Review A*, 51(2), 1015.

Dyakonov, M. I. (2012). Revisiting foundations for the fault-tolerant quantum computation. *arXiv preprint arXiv: 1210.1782*.

Forcer, T. M., Hey, A. J., Ross, D. A., & Smith, P. G. R. (2002). Superposition, entanglement and quantum computation. *Quantum Information and Computation*, 2(2), 97-116.

Fowler A. G.,(2011). "Constructing arbitrary Steane code single logical qubit fault-tolerant gates," Quantum Info. Comput., vol. 11, pp. 867–873, quant-ph/0411206.

Golubitsky O. & Maslov D., (2012). A study of optimal 4-bit reversible Toffoli circuits and their synthesis. IEEE Transactions on Computers 61(9):1341–1353, arXiv:1103.2686

Han, K. H., & Kim, J. H. (2000). Genetic quantum algorithm and its application to combinatorial optimization problem. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on* (Vol. 2, pp. 1354-1360). IEEE.

Harrow A. W., Recht B., and Chuang I. L., (2002). Efficient discrete approximations of quantum gates. J. Math. Phys., 43:4445, quant-ph/0111031.

Hayes, J. P., & Markov, I. L. (2006). *Quantum Approaches to Logic Circuit Synthesis and Testing*. Michigan University Ann Arbor.

Hsieh, M. H., & Le Gall, F. (2011). NP-hardness of decoding quantum error-correction codes. *Physical Review A*, *83*(5), 052331.

Jaeger, G. (2007). *Quantum information* (pp. 81-89). Springer New York.

Jones, N. C., Van Meter, R., Fowler, A. G., McMahon, P. L., Kim, J., Ladd, T. D., & Yamamoto, Y. (2012). Layered architecture for quantum computing. *Physical Review X*, *2*(3), 031007.

Jozsa, R. (1997). Entanglement and quantum computation. *Geometric Issues in the Foundations of Science. Oxford University Press. New York, USA.*

Jozsa, R., & Linden, N. (2003). On the role of entanglement in quantum-computational speed-up. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 459(2036), 2011-2032.

Jozsa, R. (2006). An introduction to measurement based quantum computation. *NATO Science Series, III: Computer and systems sciences*, *199*, 137-158.

Khan, M. H., & Perkowski, M. (2005). Evolutionary algorithm based synthesis of multi-output ternary functions using quantum cascade of generalized ternary gates. *Int. J. Multi-Valued Log. Soft Comput*.

Kliuchnikov, V., Maslov, D., & Mosca, M. (2012). Practical approximation of single-qubit unitaries by single-qubit quantum Clifford and T circuits. *arXiv preprint arXiv:1212.6964*.

Kliuchnikov, V. (2013). Synthesis of unitaries with clifford+ t circuits. *arXiv preprint arXiv:1306.3200*.

Kliuchnikov V., Maslov D., and Mosca M., (2013). "Fast and efficient exact synthesis of single qubit unitaries generated by Clifford and T gates," Quantum Info. Comput., vol. 13, pp. 607–630, arXiv:1206.5236.

Knill, E. (2010). Physics: quantum computing. *Nature*, *463*(7280), 441-443.

Ladd, T. D., Jelezko, F., Laflamme, R., Nakamura, Y., Monroe, C., & O'Brien, J. L. (2010). Quantum computers. *Nature*, *464*(7285), 45-53.

Landauer R., (1992). "Information is physical," in Physics and Computation, 1992. PhysComp'92., Workshop on, pp. 1–4.

Lin, C., Chakrabarti, A., & Jha, N. (2013). FTQLS: Fault-Tolerant Quantum Logic Synthesis. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 1-1.

Lukac, M., & Perkowski, M. A. (2002). Evolving Quantum Circuits Using Genetic Algorithm. *Evolvable Hardware, 2002. Proceedings. NASA/DoD Conference on*, 177 - 185.

Lukac, M., Kameyama, M., Miller, M., & Perkowski, M. (2013). High Speed Genetic Algorithms in Quantum Logic Synthesis: Low Level Parallelization vs. Representation. *Journal of Multiple-Valued Logic & Soft Computing*, *20*.

Mosca, M. (2009). Quantum algorithms. In *Encyclopedia of Complexity and Systems Science* (pp. 7088-7118). Springer New York.

Messac, A., Ismail-Yahaya, A., & Mattson, C. A. (2003). The normalized normal constraint method for generating the Pareto frontier. *Structural and multidisciplinary optimization*, *25*(2), 86-98.

Messac, A., & Mattson, C. A. (2004). Normal constraint method with guarantee of even representation of complete Pareto frontier. *AIAA journal*, *42*(10), 2101-2111.

Nagy, A. B. (2006). On an implementation of the Solovay-Kitaev algorithm. *arXiv preprint quant-ph/0606077*.

Nielsen, M. A., & Chuang, I. L. (2010). *Quantum computation and quantum information*. New York, NY, USA, Cambridge University Press.

Nilsson, N. J. (2014). *Principles of artificial intelligence*. Morgan Kaufmann.

Pachos, J. K. (2012). *Introduction to topological quantum computation*. Cambridge University Press.

Paetznick, A., & Svore, K. M. (2013). Repeat-Until-Success: Non-deterministic decomposition of single-qubit unitaries. *arXiv preprint arXiv:1311.1074*.

Penrose, R. (1998). Quantum computation, entanglement and state reduction. *Philosophical Transactions A*, 356(1743), 1927.

Pittenger, A. O. (1999). *An introduction to quantum computing algorithms*. Birkhauser Boston.

Pérez-Delgado, C. A., & Kok, P. (2011). Quantum computers: Definition and implementations. *Physical Review A*, *83*(1), 012303.

Pham, T. T., Van Meter, R., & Horsman, C. (2013). Optimization of the Solovay-Kitaev algorithm. *Physical Review A*, *87*(5), 052332.

Ruican, C., Udrescu, M., Prodan, L., & Vladutiu, M. (2008). A Genetic Algorithm Framework Applied to Quantum Circuit Synthesis. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2007).* Springer Berlin Heidelberg. 419-429.

Saeedi, M., & Markov, I. L. (2013). Synthesis and optimization of reversible circuits—a survey. *ACM Computing Surveys (CSUR)*, *45*(2), 21.

Shende, V. V., Bullock, S. S., & Markov, I. L. (2006). Synthesis of quantum-logic circuits. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, *25*(6), 1000-1010.

Shor, P. W. (1994). Algorithms for quantum computation: Discrete logarithms and factoring. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on* (pp. 124-134). IEEE.

Shor, P. W. (1996). Fault-tolerant quantum computation. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on* (pp. 56-65). IEEE.

Shor, P. W. (2002). Introduction to quantum algorithms. In *Proceedings of Symposia in Applied Mathematics* (Vol. 58, pp. 143-160).

Simon, D. R. (1997). On the power of quantum computation. *SIAM journal on computing*, *26*(5), 1474-1483.

Van den Nest, M. (2013). Universal quantum computation with little entanglement. *Physical review letters*, *110*(6), 060504.

Yao A. , (1993) "Quantum circuit complexity," in Proc. 34th Ann. Symp. on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, CA, pp. 352-361.

Zhou, A., Qu, B. Y., Li, H., Zhao, S. Z., Suganthan, P. N., & Zhang, Q. (2011). Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, *1*(1), 32-49.