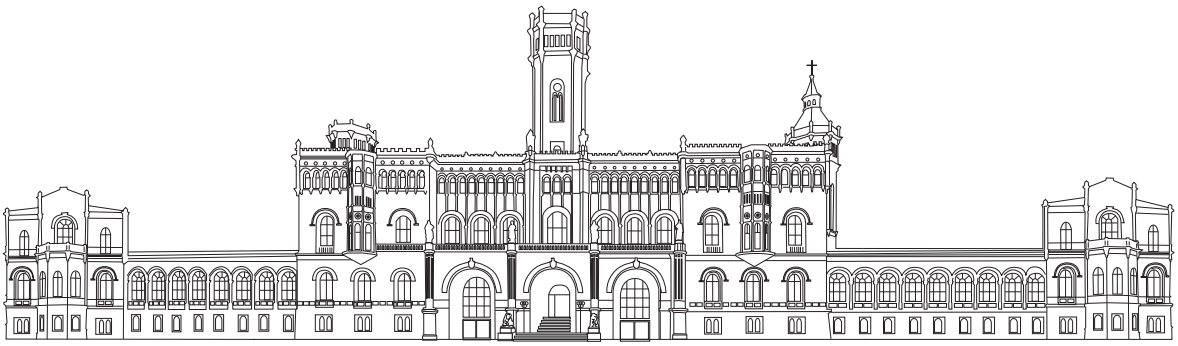


Gottfried Wilhelm Leibniz Universität
Hannover

Fakultät für Mathematik und Physik

Bachelorarbeit

im Studiengang Physik - Schwerpunkt Quanteninformation

zur Erlangung des akademischen Grades
Bachelor of Science

Thema:	Quantum Compiling - Zerlegung unitärer Quantenoperationen
Autor:	Kerstin Beer MatNr. 2951480
Version vom:	1. September 2015
Betreuer:	Prof. Dr. Tobias Osborne

Zusammenfassung

Um eine beliebige Quantenoperation in einem Quantencomputer zu implementieren, ist es nötig diese mit einer Abfolge elementarer Gates zu approximieren. Diese Arbeit stellt drei Algorithmen vor, die unitäre Quantenoperationen zerlegen. Insbesondere wird darauf eingegangen, dass Qudit-Systeme, die mehr als zwei Basiszustände einnehmen können, die Rechengeschwindigkeit verbessern und die Fehler reduzieren können. Aus diesem Grund wird die Möglichkeit behandelt, die Algorithmen auf Qudit-Systeme zu verallgemeinern. Im Gegensatz zu den anderen beiden Verfahren wurde der Solovay-Kitaev-Algorithmus bereits in der Originalquelle sowohl für Qubit- als auch Qudit-Systeme vorgestellt. Es hat sich herausgestellt, dass auch der Repeat-Until-Success-Algorithmus für Qudit-Systeme funktioniert. Allerdings konnte auf Grund fehlender Computerleistung nicht abgeschätzt werden, inwiefern die Verallgemeinerung praktisch Sinn ergibt. Ein Erfolg dieser Arbeit ist die Übertragung des Welch-Algorithmus, der beliebige diagonale, unitäre Operationen zerlegt. Es ist gelungen, diese auch in Qudit-Systemen mit elementaren Gates auszudrücken.

Abstract

To implement any quantum operation in a quantum computer it is necessary to approximate the operation with a sequence of elementary gates. Three algorithms decomposing unitary quantum operations are introduced in this work. In particular we address that qudit systems, which can take more than two base states, improve computing speed and can reduce errors. For this purpose we generalize these algorithms to qudit systems. In contrast to the other two methods the Solovay-Kitaev algorithm has already been presented in the original source for qubit and qudit systems. The Repeat-Until-Success algorithm works equally for qudit and qubit systems. We could not determine if this generalization for qudits has practical advantages because of lack of computer power. A success of this work is the transfer of the Welch algorithm which decomposes an arbitrary diagonal unitary operation. It was successful to express these operations with elementary gates in qudit systems.

Inhaltsverzeichnis

1 Einleitung

Wie in einem klassischen Computer können auch in einem Quantencomputer nicht unendlich viele Operationen eingebaut werden. Um also einen beliebigen Quantenalgorithmus zu implementieren, ist es von Bedeutung jede nötige Quantenoperation möglichst fehlerarm in eine Folge elementarer Operationen zu zerlegen. Eine Aufgabe ist die Wahl der Menge dieser Operationen. In vielen Fällen wird die sogenannte Clifford+T-Basis gewählt. Andererseits spielt die Wahl der Zerlegungsmethode eine große Rolle. Das Ziel ist es immer, eine fehlerarme Approximation mit einer möglichst kurzen Gatefolge in möglichst kurzer Zeit zu finden. Es werden verschiedene Algorithmen betrachtet, da die Forderungen unterschiedlich gut erfüllt werden. Zum Beispiel sind einige Verfahren schneller, dagegen ist bei anderen der Fehler kleiner. Des Weiteren sind die benötigten Anfangsbedingungen nicht gleich. Einige Verfahren benötigen beispielsweise zusätzliche Hilfsqudits, in anderen sind Messungen eingebaut.
Ich habe mich deshalb mit drei ausgewählten Algorithmen beschäftigt, die die Näherung einer beliebigen unitären Quantenoperation durch eine Folge elementarer Operationen aus einer festgelegten, endlichen Menge ermöglichen. Ein Ziel ist es, die in der Arbeit vorgestellten Verfahren gegenüber zu stellen. Dazu können die Anzahl der Gates oder die Zeit, die für das Implementieren einer beliebige Operation benötigt wird, einfach verglichen werden. Um den Umfang der Zerlegung zu vergleichen, wird häufig die Zahl der benötigten T-Gates angegeben, wenn in der Clifford+T-Basis zerlegt wird. Da das T-Gate als einziges Gate in der Basis nicht die vorteilhaften Eigenschaften der Clifford-Gruppe hat, ist eine fehlertolerante Implementierbarkeit besonders aufwändig..
Ein weiteres Bestreben in der vorliegenden Arbeit soll sein, die drei Verfahren weiter zu entwickeln. Zwei der vorgestellten Algorithmen werden in der Originalquelle nur für Systeme mit zwei Basiszuständen erläutert. Nach [NIE, 2010] können Systeme mit mehreren Basiszuständen verbesserte Rechengeschwindigkeiten und reduzierte Fehler aufweisen. Aus diesem Grund ist mein Ziel die Algorithmen so zu verallgemeinern, dass der Betrieb für Systeme mit beliebig vielen Basiszuständen gelingt.
Am Anfang dieser Arbeit werden zunächst einige Grundlagen aus der Quanteninformation eingeführt, die für diese Arbeit von Bedeutung sind. Anschließend wird der Solovay-Kitaev-Algorithmus [DAW, 2008] vorgestellt, der in der Originalquelle bereits für beliebig viele Basiszustände erläutert wurde. Es handelt sich um einen rekursiven Algorithmus, der als Eingabe das zu approximierende Gate sowie die gewünschte Wiederholung fordert. Die erste Approximation wird mit Hilfe einer Datenbank vorgenommen und bei jedem Durchlaufen verbessert. Daraufhin wird der Algorithmus Repeat-Until-Success [PAET, 2014] thematisiert, der im Gegensatz zum Solovay-Kitaev-Algorithmus mit Hilfsqubits und Messungen arbeitet und die gewünschte Operation nur mit einer bestimmten Wahrscheinlichkeit liefert. Bei gemessenem Misserfolg wird die Operation allerdings rückgängig gemacht und wiederholt. Dieser Vorgang wird fortgesetzt bis ein Erfolg festgestellt werden konnte. Im nächsten Kapitel stelle ich einen Algorithmus vor, der einen diagonalen, unitären Operator in Phasen-, NOT- und Toffoli-Gates zerlegen kann und verallgemeinere diesen für beliebig viele Basiszustände. Es folgen eine Zusammenfassung und ein Vergleich sowie das Fazit der Ergebnisse dieser Arbeit.

2 Grundlagen

In diesem Kapitel werden einige Grundlagen der Quanteninformation eingeführt, die für diese Arbeit von Bedeutung sind.

2.1 Quantensysteme

In der klassischen Informatik werden zur Informationsverarbeitung Bits genutzt, die sich in einem bestimmten Zustand befinden. Dieser kann 0 oder 1 sein. Das quantenmechanische Äquivalent wird Qubit genannt und hält sich mit einer bestimmten Wahrscheinlichkeit in den Basiszuständen $|0\rangle$ oder $|1\rangle$ eines zweidimensionalen komplexen Hilbertraums auf. Die beliebige normierte Superposition von $|0\rangle$ und $|1\rangle$ entspricht einem allgemeinen Qubit. Ein n-Qubitzustand ist dabei als n-faches Tensorprodukt eines Einqubit-Zustands definiert und ist somit ein normiertes Element aus $(C^2)^{\otimes n}$.
Bisher haben wir nur Qubits betrachtet, die zwei verschiedene Zustände einnehmen können. Es wurde allerdings in Erwägung gezogen, dass Qutrits Vorteile mit sich bringen könnten [NIE, 2010]. Dadurch könnte die Rechengeschwindigkeit verbessert und Fehler reduziert werden. Diese Qutrits können drei verschiedene Basiszustände einnehmen: $|0\rangle$, $|1\rangle$oder $|2\rangle$. Analog können auch Qudits definiert werden, die in d verschiedenen Basiszuständen vorkommen. Wenn die Anzahl der möglichen Zustände gleich einer Primzahl ist, spricht man von Qupits.

2.2 Elementare Quantenoperationen (Gates)

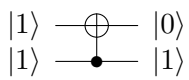
Ein klassischer Computer besteht aus Drähten, die zur Übertragung der Information dienen, und Logik-Gates. Letztere verändern die Information [NIE, 2010]. Analog zu den elektronischen Logik-Gates können auch Quantengates definiert werden. Zum Beispiel kann ein Gate den Zustand $|0\rangle$ in $|1\rangle$ und $|1\rangle$ in $|0\rangle$ umwandeln. Hierbei handelt es sich um das Pauli X-Gate, auch NOT-Gate genannt, dass in Dirac-Notation ausgeschrieben so aussieht:

$$NOT = |1\rangle\langle 0| + |0\rangle\langle 1| \ .$$

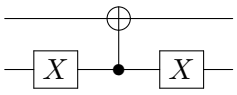
Dies kann man überprüfen:

$$NOT\,|0\rangle = |1\rangle\langle 0|0\rangle + |0\rangle\langle 1|0\rangle = |1\rangle$$
$$NOT\,|1\rangle = |1\rangle\langle 0|1\rangle + |0\rangle\langle 1|1\rangle = |0\rangle$$

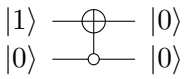
Man unterscheidet zwischen Einzel-Qubit-Gates und Multi-Qubit-Gates. Das eben behandelte NOT-Gate ist ein Singel-Qubit-Gate, da es nur auf ein Qubit wirken kann. Ein typisches Multi-Qubit-Gate ist das „controlled not“-Gate ($CNOT$-Gate). Dieses wird auf zwei Qubits angewandt, wobei eines als Kontroll- und das andere als Ziel-Qubit fungiert. Ist das Kontroll-Qubit im Zustand 1, werden die Basiszustände vertauscht, ansonsten wirkt die Identität.



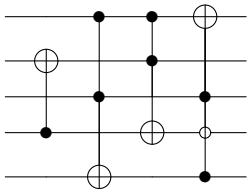
Statt auf 1 zu prüfen, kann man auch auf 0 prüfen. Dies lässt sich mit zwei NOT-Gates und einem $CNOT$-Gate beschreiben.



Symbolisch erkennt man diese Operation an einem ungefüllten Kreis auf dem Kontroll-Qubit.



Analog kann man ähnliche Operationen auch für zwei oder mehr Kontroll-Qubits definieren. Diese werden multi-controlled-NOT-Gates genannt und als „variabel“ bezeichnet, wenn die unterschiedlichen Kotrollpunkte auf unterschiedliche Weisen prüfen. In der Abbildung sind einige multi-controlled-NOT-Gates für $d = 2$ zu sehen. Das letzte Gate ist variabel und prüft zwei Kontrollqubits auf $|1\rangle$ und eines auf $|0\rangle$. Bei Qudits sind auch Gates denkbar, die auf einen Zustand $\in \{|0\rangle, ..., |d-1\rangle\}$ prüfen.



Nach dem Postulat der Quantenmechanik sind alle diese in einem geschlossenen Quantensystem erlaubten Transformationen unitär.
Da in einem Quantencomputer nicht beliebig viele Gates implementiert werden können, stellt sich nun natürlich die Frage, ob komplexe Operationen mit einer Zerlegung in elementare Operationen angenähert werden können. Gemäß [NIE, 2010] kann man zum Beispiel eine unitäre Operation auf Qudits in die Clifford+T-Basis zerlegen. Die Menge elementarer Operationen besteht in diesem Fall aus der Menge, die das T-, $CNOT$-, Hadamard- und S-Gate für alle Qudits enthält. Hierbei handelt es sich um ein Beispiel für ein universelles Gate-Set. Dies ist eine Menge von Gates, mit der alle erlaubten Transformationen fehlerarm approximiert werden können, was in der Praxis unabdingbar ist. Es folgen zwei entsprechende Definitionen.

Definition 2.1. *S sei eine ϵ-Approximation von U, wenn*

$$d(U,S) := \|U - S\| = sup_{\|\Psi\|=1} \|(U-S)\,|\Psi\rangle\,\| < \epsilon$$

Definition 2.2. *Ein universelles Gate-Set $\mathcal{G}$ für n Qudits sei eine Menge mit den folgenden Eigenschaften:*

1. *Für alle $g \in \mathcal{G}$ gilt auch $g \in SU(d^n)$, das heißt alle $g \in \mathcal{G}$ sind unitär und ihre Determinanten sind 1.*
2. *Ist $g \in \mathcal{G}$, so ist auch $g^{-1} = g^\dagger \in \mathcal{G}$.*
3. *Für jedes $U \in SU(d^n)$ und jede Genauigkeit $\epsilon > 0$ existiert ein Produkt $S = g_1...g_m$ mit $g_i \in \mathcal{G}$, so dass S eine ϵ-Approximation von U ist.*

Die Notwendigkeit des ersten und dritten Punktes liegen klar auf der Hand. Der zweite Punkt ist im Beweis des Solovay-Kitaev-Algorithmus von Bedeutung, damit $U^\dagger$ durch Umkehr der Reihenfolge und Invertieren der zerlegenden Gates approximiert werden kann, wenn U bereits zerlegt wurde. Es ist auch möglich, den zweiten Punkt der Definition auszulassen. Wendet man nämlich die Definition korrekt an, muss die Clifford+T-Basis zusätzlich zu den oben erwähnten Gates T, $CNOT$, Hadamard und S auch das $T^{-1} = T^7$ und $S^{-1} = S^3$

enthalten. Die anderen Mengenelemente sind selbst ihre Inversen. Ob diese Genauigkeit nötig ist, muss von Fall zu Fall entschieden werden. Die durch diese beiden Mengen erzeugten Gruppen sind jedoch die selben.
Im Folgenden sind die eben genannten und die anderen in dieser Arbeit genutzten Gates für Qubits definiert:

—[H]— Hadamard	$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} = \frac{\|0\rangle + \|1\rangle}{\sqrt{2}}\langle 0\| + \frac{\|0\rangle - \|1\rangle}{\sqrt{2}}\langle 1\|$
—[X]— Pauli X oder not	$NOT = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix} = \|1\rangle\langle 0\| + \|0\rangle\langle 1\|$
—[Y]— Pauli Y	$Y = \begin{bmatrix}0 & -i\\ i & 0\end{bmatrix} = i(\|1\rangle\langle 0\| - \|0\rangle\langle 1\|)$
—[Z]— Pauli Z	$Z = \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix} = \|0\rangle\langle 0\| - \|1\rangle\langle 1\|$
—[S]— Phase	$S = \begin{bmatrix}1 & 0\\ 0 & i\end{bmatrix} = \|0\rangle\langle 0\| + i\,\|1\rangle\langle 1\|$
—[T]— $\frac{\pi}{8}$Phase-Gate	$T = \begin{bmatrix}1 & 0\\ 0 & \exp\frac{i\pi}{4}\end{bmatrix} = \|0\rangle\langle 0\| + \exp\frac{i\pi}{4}\,\|1\rangle\langle 1\|$
 controlled-not	$CNOT = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix} = \|1\rangle\langle 1\| \otimes X + \|0\rangle\langle 0\| \otimes \mathbb{1}$
 Toffoli	$TOF = \begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0 & 1 & 0\end{bmatrix}$
	$= \|11\rangle\langle 11\| \otimes X + [\|01\rangle\langle 01\| + \|10\rangle\langle 10\| + \|00\rangle\langle 00\|] \otimes \mathbb{1}$

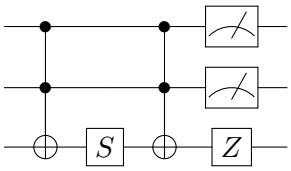
Für Qudits können analog zu den Qubit-Gates mit $F = \{1, ..., d-1\}$ folgende unitäre Gates definiert werden :

NOT	$\forall c \in F, \|a\rangle \rightarrow \|a + c\rangle$
CNOT	$\|a, b\rangle \rightarrow \|a, a + b\rangle$
Phase	$\forall c \in F, \|a\rangle \rightarrow w^{ca}\|a\rangle$ mit $w = e^{\frac{2i\pi}{d}}$
Hadamard	$\|a\rangle \rightarrow \frac{1}{\sqrt{d}}\sum_{b\in F} w^{rab}\|b\rangle \;\; \forall\, 0 < r < d$
Toffoli	$\|a\rangle\|b\rangle\|c\rangle \rightarrow \|a\rangle\|b\rangle\|c + ab\rangle$

[AHA, 1999] definiert unter anderem diese Gates für Qupits. Dass die geforderte Unitarität bei den Gates NOT, CNOT, Phase und Toffoli nicht nur für Qupits sondern auch für Qudits gilt, ist leicht einzusehen. Das verallgemeinerte Hadamard-Gate entspricht der Quantenfouriertransformation. Dessen Unitarität ist gemäß [NIE, 2010] bekannt.

2.3 Quantenschaltkreise

Die Drähte eines klassischen Computers verbinden die einzelnen Logik-Gates zu einer Schaltung. Vergleichbar versteht man unter einem Quantenschaltkreis eine Folge von Quanten-Gates. Quantenalgorithmen bestehen also aus Kombinationen von Gates, die in einem Quatenschaltkreis dargestellt werden, wie in der Abbildung beispielhaft angedeutet wird. Diese Quantenschaltkreise können auch Hilfsqudits einschließen, die zusätzlich zu den informationsübertragenden Qudits definiert werden, um bestimmte Algorithmen nutzen zu können. Außerdem können, zum Beispiel für klassische Ausgaben, auch Messungen enthalten sein. In der grafischen Darstellung eines Quantenschaltkreis repräsentiert jede waagerechte Linie ein Qudit.



2.4 Projektive Messungen

Projektive Messungen sind ein Spezialfall von POVM-Messungen (Positive Operator Valued Measure). Sie sind definiert durch projektive Operatoren $\{P_i\}_{1\leq i\leq n}$ mit $\sum_{i=1}^{n} P_i = \mathbb{1}$. Die Wahrscheinlichkeit für das Messergebnis i,bei gegebenem Zustand $|\psi\rangle$, erweist sich als $\langle\psi|P_i|\psi\rangle$. Nach der Messung befindet sich das System in dem Zustand $\frac{P_i|\psi\rangle}{\sqrt{\langle\psi|P_i|\psi\rangle}}$ [NIE, 2010].
Die Berechnungsbasis (computational basis) ist eine festgesetzte, orthonormale Basis des n-Qudit-Raums, die durch $|l\rangle$ beschrieben wird, wobei l ein String der Länge n über $\{0, 1, ..., d-1\}$ ist. Für die Messung in dieser Berechnungsbasis für ein Qubit sind die Projektoren wie folgt definiert:

$$P_0 = |0\rangle\langle 0|$$
$$P_1 = |1\rangle\langle 1|$$

Für zwei Qubits werden die Projektoren in dieser Art dargestellt:

$$P_{00} = |00\rangle\langle 00|$$
$$P_{01} = |01\rangle\langle 01|$$
$$P_{10} = |10\rangle\langle 10|$$
$$P_{11} = |11\rangle\langle 11|$$

Allgemeiner spricht man von einer Messung in der Basis $|e_1\rangle, |e_2\rangle, ..., |e_k\rangle$, wenn die Messoperatoren genau den Projektionen $|e_i\rangle\langle e_i|$ entsprechen.

2.5 Binärschreibweise und Hamming-Gewicht

Vor allem im Kapitel über die Zerlegung diagonaler Operatoren wird die Binärschreibweise benötigt. Ein String $[j] = j_1 j_2 ... j_n$ beschreibt in der Binärdarstellung mit $j_i \in \{0, 1\}$ die Zahl $j = 2^{n-1}j_1 + 2^{n-2}j_2 + ... + 2^0 j_n$. Links steht also die hochwertigste Stelle j_1 und rechts die niederwertigste.
Um den Unterschied zwischen zwei Bitstrings zu messen, kann man den Hamming-Abstand benutzen. Dieser ist als Anzahl der unterschiedlichen Stellen zweier gleichlanger Strings definiert. Der Hamming Abstand zwischen den Strings

$$\{1, 1, 0, 1, 1, 0, 1, 1, 0, 1\}$$

$$\{0,1,1,1,1,0,1,1,0,1\}$$

ist 2. Das Hamming-Gewicht eines beliebigen Bitstrings ist der Hamming-Abstand zum entsprechend gleichlangen, mit Nullen gefüllter String. Für Bits kann man das Hamming-Gewicht eines Strings $[j] = j_1 j_2 ... j_n$ als Summe aller String-Stellen berechnen $\sum_{i=1}^{n} j_i$. Beide oben abgebildeten Strings haben das Hamming-Gewicht 7. Entsprechend definiere ich das Hamming-Gewicht für beliebiges d wie folgt:

Definition 2.3. *Für Strings* $[j] = \alpha_1 \alpha_2 ... \alpha_n$ *über* $\{0, 1, ..., d-1\}$ *kann man die Zahl zur Basis* d *darstellen mit* $j = \sum_{i=0}^{n} \alpha_i d^{k_i}$ *mit* $k_r \in \mathbb{Z}$, $\alpha_i \in \{0, ..., d-1\}$ *und* $k_1 < ... < k_n$. *Dann sei das Hamming-Gewicht eines Qudit-Strings* $\sum_{i=0}^{n} \alpha_i$.

Da für $d = 2$ das α nur die Werte 0 oder 1 annehmen kann, stimmt diese Definition auch für Bitstrings.

3 Solovay-Kitaev-Algorithmus

In diesem Kapitel sollen das Solovay-Kitaev-Theorem und sein konstruktiver Beweis vorgestellt werden. Letzterer kann einen beliebigen unitären Operator auf Qudits in eine Folge von Gates aus einem Set zerlegen und beweist damit das Theorem.

3.1 Solovay-Kitaev-Theorem

Wie schon im Grundlagen-Kapitel erwähnt, ist das Approximieren eines beliebigen unitären Operation mit Hilfe von elementaren Gates eine große Notwendigkeit in der Praxis. Von großer Wichtigkeit sind dabei vor allem die Fragen, wie viele Gates für die Darstellung gebraucht werden, wie schnell diese Zahl mit der gewünschten Genauigkeit ansteigt und wie schnell man eine solche Zerlegung finden kann. Antworten auf diese Fragestellungen bekommt man im Solovay-Kitaev-Theorem.

Theorem 3.1. *Sei d und n fix und $\mathcal{G}$ ein universelles Gate-Set für $SU(d^n)$ und $\epsilon > 0$ eine beliebige Zahl. Dann gibt es eine Konstante c, sodass es für jedes $U \in SU(d^n)$ eine Gate-Folge S der Länge $O(\log^c(1/\epsilon))$ gibt mit $d(U, S) < \epsilon$.*

Beweis. Der Beweis dieses Theorems geschieht in den nächsten zwei Abschnitten durch die Vorstellung des Solovay-Kitaev-Algorithmus für ein einzelnes Qubit und schließlich für höherdimensionale Systeme. □

Die folgende Herleitung ergibt $c \approx 3.97$. Man nehme an, ein Gate kann durch eine Abfolge $U_1, ..., U_m$ von m Gates dargestellt werden. Jede einzelne Approximation der U_i muss mit der Genauigkeit m/ϵ geschehen, damit man eine Gesamtgenauigkeit von $1/\epsilon$ erhält. Insgesamt ergibt sich eine Gatefolge der Länge $O(m \log^{3.97}(m/\epsilon))$ in der Zeit von $O(m \log^{2.71}(m/\epsilon))$. Intuitiv wäre nach [NIE, 2010] eher eine quadratische Abhängigkeit von m, sodass die Länge der Gatefolge mit $O(m^2/\epsilon)$ skaliert. Der polylogarithmische Zuwachs mit m bei Verwendung des Solovay-Kitaev-Algorithmus ist im Vergleich zur quadratischen Abhängigkeit von m eine große Verbesserung.

3.2 Solovay-Kitaev-Algorithmus für ein Qubit

Um einen Überblick über den Algorithmus zu erlangen, wird dieser im Folgenden als Pseudocode dargestellt.

```
1 function Solovay-Kitaev(Gate U, depth n)
2 if(n == 0)
3         Return Basic Approximation to U
4 else
5         Set U_{n-1} = Solovay-Kitaev(U,n-1)
6         Set V, W = GC-Decompose(U U_{n-1}^†)
7         Set V_{n-1} = Solovay-Kitaev(V,n-1)
8         Set W_{n-1} = Solovay-Kitaev(W,n-1)
9         Return U_n = V_{n-1} W_{n-1} V_{n-1}^† W_{n-1}^† U_{n-1}
```

In der ersten Zeile wird deutlich, dass der Algorithmus zwei Inputs hat, das zu approximierende Gate U und die Anzahl der Gates n, in die U zerlegt werden soll. Die Approximationsgenauigkeit ϵ_n ist eine mit größer werdendem n abnehmende Funktion. Für $n \to \infty$ gilt $\epsilon_n \to 0$.

Der Output des Algorithmus ist eine Gate-Folge, die U approximiert. In der zweiten und dritten Zeile wird die Möglichkeit $n = 0$ abgearbeitet. In diesem Fall gibt die Funktion U aus.
Das rekursive Verfahren des Algorithmus im Fall $n \neq 0$ wird ab Zeile 4 dargestellt. Die Approximation wird dabei in jedem Schritt genauer, das heißt es gilt $\epsilon_n < \epsilon_{n-1}$. In Zeile 5 wird U_{n-1} gesetzt als Lösung des Solovay-Kitaev-Algorithmus für $n-1$ Schritte. Der Algorithmus ruft sich an dieser Stelle immer wieder selbst auf, wobei die Eingabe für n immer um eins erniedrigt wird. Die so verschachtelten Funktionen warten, bis die Schleife darunter ausgeführt wurde. Die erste Approximation mit der Genauigkeit ϵ_0 muss in einer Tabelle nachgeschlagen werden, auf die in Kapitel 3.3 näher eingegangen wird.
Die Zeilen 6 bis 8 erstellen eine ϵ_n-Approximation von $\Delta := UU_{n-1}^\dagger$, die von der Form $V_{n-1}W_{n-1}V_{n-1}^\dagger W_{n-1}^\dagger$ ist. Dies bringt in Zeile 9 zusammen mit der exakten Zerlegung von U_{n-1} eine ϵ_n-Approximation von U.
Auf Grund der Definition von Δ ist bekannt, dass $d(I, \Delta) < \epsilon_{n-1}$ gilt. In der Zeile 6 wird Δ in einen sogenannten Gruppenkommutator zerlegt: $\Delta = VWV^\dagger W^\dagger$, wobei V und W nahezu der Identität entsprechen. Dass es so eine Zerlegung gibt, geht aus folgendem Lemma hervor.

Lemma 3.2. *Sei $U \in SU(2)$ und $d(I, U) < \epsilon$. Dann gibt es V und W mit $d(I, V) < c_{gc}\sqrt{\epsilon}$ und $d(I, W) < c_{gc}\sqrt{\epsilon}$, sodass $U = VWV^\dagger W^\dagger$.*

Beweis. Der Beweis ist zu finden in [DAW, 2008]. Für c_{gc} ergibt sich $c_{gc} \approx \frac{1}{\sqrt{2}}$. □

Zeile 7 und 8 setzen V_{n-1} und W_{n-1} als ϵ_{n-1}-Näherung von V beziehungsweise W durch den Solovay-Kitaev-Algorithmus für $n-1$ Schritte. Auch an dieser Stelle erfolgt ein verschachteltes Aufrufen des Algorithmus. Das nächste Lemma zeigt, dass $V_{n-1}W_{n-1}V_{n-1}^\dagger W_{n-1}^\dagger$ eine $c_{approx}\epsilon_{n-1}^{3/2}$-Approximation von Δ ist.

Lemma 3.3. *Seien V, W, $\widetilde{W}$ und $\widetilde{W}$ unitäre Operationen für die $d(V, \widetilde{V}) < \Delta$, $d(W, \widetilde{W}) < \Delta$, $d(I, V) < \delta$ und $d(I, W) < \delta$ gilt. Dann gilt:*

$$d(VWV^\dagger W^\dagger, \widetilde{V}\widetilde{W}\widetilde{V}^\dagger\widetilde{W}^\dagger) < 8\Delta\delta + 4\Delta\delta^2 + 8\Delta^2 + 4\Delta^3 + \Delta^4$$

Beweis. Der Beweis erfolgt analog zu [DAW, 2008]. Da $d(V, \widetilde{V}) < \Delta$ und $d(W, \widetilde{W}) < \Delta$ gilt, sei nun $\widetilde{V} = V + \Delta_V$ und $\widetilde{W} = W + \Delta_W$ mit $|\Delta_V| < \Delta$ und $|\Delta_W| < \Delta$.

$$\begin{aligned}
\widetilde{V}\widetilde{W}\widetilde{V}^\dagger\widetilde{W}^\dagger =& (V + \Delta_V)(W + \Delta_W)(V^\dagger + \Delta_V^\dagger)(W^\dagger + \Delta_W^\dagger) \\
=& (VW + V\Delta_W + \Delta_V W + \Delta_V\Delta_W)(V^\dagger + \Delta_V^\dagger)(W^\dagger + \Delta_W^\dagger) \\
=& (VWV^\dagger + V\Delta_W V^\dagger + \Delta_V WV^\dagger + \Delta_V\Delta_W V^\dagger \\
& + VW\Delta_V^\dagger + V\Delta_W\Delta_V^\dagger + \Delta_V W\Delta_V^\dagger + \Delta_V\Delta_W\Delta_V^\dagger)(W^\dagger + \Delta_W^\dagger) \\
=& VWV^\dagger W^\dagger + \Delta_V WV^\dagger W^\dagger + V\Delta_W V^\dagger W^\dagger + VW\Delta_V^\dagger W^\dagger + VWV^\dagger\Delta_W^\dagger \\
& + O(\Delta^2) + O(\Delta^3) + O(\Delta^4)
\end{aligned}$$

In der Rechnung wird deutlich, dass es sechs Terme gibt, die von Δ^2 abhängen, vier, die von Δ^3 abhängen und einen der mit Δ^4 wächst. Daraus folgt

$$\begin{aligned}
d(VWV^\dagger W^\dagger, \widetilde{V}\widetilde{W}\widetilde{V}^\dagger\widetilde{W}^\dagger) <& \|\Delta_V WV^\dagger W^\dagger + V\Delta_W V^\dagger W^\dagger + VW\Delta_V^\dagger W^\dagger + VWV^\dagger\Delta_W^\dagger\| \\
& + 6\Delta^2 + 4\Delta^3 + \Delta^4 \ .
\end{aligned}$$

Es bleibt nun zu zeigen, dass

$$\begin{aligned}
\|\Delta_V WV^\dagger W^\dagger + VW\Delta_V^\dagger W^\dagger\| &< \Delta^2 + 4\Delta\delta + 2\Delta\delta^2 \\
\|V\Delta_W V^\dagger W^\dagger + VWV^\dagger\Delta_W^\dagger\| &< \Delta^2 + 4\Delta\delta + 2\Delta\delta^2 \ .
\end{aligned}$$

Wegen $d(I,V) < \delta$ und $d(I,W) < \delta$, sei $W = I + \delta_W$ und $V = I + \delta_V$ mit $|\delta_W| < \delta$ und $|\delta_V| < \delta$. Beide Aussagen folgen analog. Aus diesem Grund folgt nur der Beweis der ersten Ungleichung.

$$\begin{aligned}\Delta_V W V^\dagger W^\dagger + V W \Delta_V^\dagger W^\dagger =& \Delta_V (I+\delta_W) V^\dagger (I+\delta_W^\dagger) + V(I+\delta_W)\Delta_V^\dagger (I+\delta_W^\dagger) \\ =& \Delta_V V^\dagger + \Delta_V V^\dagger \delta_W^\dagger + \Delta_V \delta_W V^\dagger + \Delta_V \delta_W V^\dagger \delta_W^\dagger \\ &+ V\Delta_V^\dagger + V\Delta_V \delta_W^\dagger + V\delta_W \Delta_V^\dagger + V\delta_W \Delta_V^\dagger \delta_W^\dagger\end{aligned}$$

Mit

$$\begin{aligned} & I = (V+\Delta_V)(V^\dagger + \Delta_V^\dagger) = VV^\dagger + V\Delta_V^\dagger + \Delta_V V^\dagger + \Delta_V \Delta_V^\dagger \\ \Leftrightarrow \quad & 0 = V\Delta_V^\dagger + \Delta_V V^\dagger + \Delta_V \Delta_V^\dagger \\ \Leftrightarrow \quad & \Delta_V V^\dagger + V\Delta_V^\dagger = -\Delta_V \Delta_V^\dagger \end{aligned}$$

folgt

$$\|\Delta_V W V^\dagger W^\dagger + V W \Delta_V^\dagger W^\dagger\| < \Delta^2 + 4\Delta\delta + 2\Delta\delta^2 \ .$$

Beweist man die andere Ungleichung analog, folgt das Lemma. □

Ersetzt man nun Δ mit ϵ_{n-1} und δ mit $c_{gc}\sqrt{\epsilon_{n-1}}$ erhält man mit $c_{approx} = 8c_{gc}$

$$d(VWV^\dagger W^\dagger, \widetilde{V}\widetilde{W}\widetilde{V}^\dagger \widetilde{W}^\dagger) < 8\epsilon_{n-1} c_{gc}\sqrt{\epsilon_{n-1}} \approx c_{approx}\epsilon_{n-1}^{3/2} \ .$$

$V_{n-1}W_{n-1}V_{n-1}^\dagger W_{n-1}^\dagger$ ist dementsprechend eine $c_{approx}\epsilon_{n-1}^{3/2}$-Approximation von $\Delta = VWV^\dagger W^\dagger$. Da der Algorithmus für Tiefe n dreimal sich selbst für Tiefe n-1 aufruft und den Operator U_n in ein Produkt aus fünf Gate-Folgen zerlegt, die sich als Ausgabe des Algorithmus für Tiefe $n-1$ ergeben, kann man ϵ_n, sowie die Länge der Gatefolge l_n als auch die Laufzeit t_n rekursiv beschreiben als

$$\begin{aligned} \epsilon_n &= c_{approx}\epsilon_{n-1}^{3/2} \\ l_n &= 5l_{n-1} \\ t_n &\leq 3t_{n-1} + \text{const} \ . \end{aligned}$$

Die Konstante in der Laufzeit ist durch die anderen Berechnungsschritte eines Algorithmusdurchlaufs bedingt.

Lemma 3.4. *Aus* $\epsilon_n = c_{approx}\epsilon_{n-1}^{3/2}$ *folgt* $\epsilon_n = \frac{1}{c_{approx}^2}(\epsilon_0 c_{approx}^2)^{\frac{3}{2}^n}$.

Beweis. Der Beweis erfolgt durch Induktion.

$$\begin{aligned} \epsilon_1 &= c_{approx}\epsilon_0^{\frac{3}{2}} \\ &= \frac{1}{c_{approx}^2}(\epsilon_0 c_{approx}^2)^{\frac{3}{2}} \\ \epsilon_{n+1} &= c_{approx}\epsilon_n^{3/2} \\ &\overset{IA}{=} c_{approx}\left(\frac{1}{c_{approx}^2}(\epsilon_0 c_{approx}^2)^{\frac{3}{2}^n}\right)^{\frac{3}{2}} \\ &= \frac{c_{approx}}{c_{approx}^3}(\epsilon_0 c_{approx}^2)^{\frac{3}{2}^{n+1}} \\ &= \frac{1}{c_{approx}^2}(\epsilon_0 c_{approx}^2)^{\frac{3}{2}^{n+1}} \end{aligned}$$

□

Daraus folgen die expliziten Ausdrücke

$$\begin{aligned}\epsilon_n &= \frac{1}{c_{approx}^2}(\epsilon_0 c_{approx}^2)^{\frac{3}{2}^n} \\ l_n &= O(5^n) \\ t_n &= O(3^n)\ .\end{aligned}$$

Das Ziel ist nun, dass sich die Approximation mit jedem Schritt verbessert.

$$\begin{aligned} & & \epsilon_n &< \epsilon_{n-1} \\ \Leftrightarrow & & \frac{1}{c_{approx}^2}(\epsilon_0 c_{approx}^2)^{\frac{3}{2}^n} &< \frac{1}{c_{approx}^2}(\epsilon_0 c_{approx}^2)^{\frac{3}{2}^n}(\epsilon_0 c_{approx}^2)^{\frac{-3}{2}} \\ \Leftrightarrow & & 1 &< (\epsilon_0 c_{approx}^2)^{\frac{-3}{2}} \\ \Leftrightarrow & & \epsilon_0 &< \frac{1}{c_{approx}^2}\end{aligned}$$

Mit der Forderung $\epsilon_0 < \frac{1}{c_{approx}^2}$ wird die Approximation während des Verfahrens stets besser. Dies ist zunächst verwunderlich, da unexakte V und W genutzt werden, um $\Delta = UU_{n-1}^\dagger = VWV^\dagger W^\dagger$ zu bilden. Im Beweis von Lemma 3.3 wird allerdings klar, dass sich die Fehler gerade wegheben. Bemerkenswert ist, dass $\widetilde{V}$ und $\widetilde{W}$ Δ-Approximationen von V und W sind und dass V und W δ-Approximationen der Identität sind. Der Abstand zwischen ursprünglichem und genähertem Gruppenkommutator, der die ϵ_n-Approximation bildet, enthält allerdings keine erste Ordnung in Δ und δ. In diesen versteckt sich ϵ_{n-1}.
Auf Grund

$$c_{approx} \approx 8c_{gc} \overset{3.2}{\approx} 8\frac{1}{\sqrt{2}} = 4\sqrt{2}$$

und wegen $\epsilon_0 < \frac{1}{c_{approx}^2}$ muss $\epsilon_0 < \frac{1}{32}$ gefordert werden.
Da der Algorithmus als Eingabe n und nicht ϵ fordert, ist es nun interessant $n(\epsilon)$ zu kennen.

$$\begin{aligned} & & \epsilon_n &= \frac{1}{c_{approx}^2}(\epsilon_0 c_{approx}^2)^{\frac{3}{2}^n} \\ \Leftrightarrow & & \ln(\epsilon_n c_{approx}^2) &= \ln(\epsilon_0 c_{approx}^2)\frac{3}{2}^n \\ \Leftrightarrow & & \frac{3}{2}^n &= \frac{\ln(1/\epsilon_n c_{approx}^2)}{\ln(1/\epsilon_0 c_{approx}^2)} \\ \Leftrightarrow & & n &= \frac{\ln\left(\frac{\ln(1/\epsilon_n c_{approx}^2)}{\ln(1/\epsilon_0 c_{approx}^2)}\right)}{\ln(3/2)} \\ \Rightarrow & & n_\epsilon &= \left\lceil \frac{\ln\left(\frac{\ln(1/\epsilon_n c_{approx}^2)}{\ln(1/\epsilon_0 c_{approx}^2)}\right)}{\ln(3/2)} \right\rceil\end{aligned}$$

Die Aufrundungsfunktion muss eingefügt werden, da für n_ϵ nur natürliche Zahlen erlaubt sind. Setzt man n nun in l_n und t_n ein, erhält man l_ϵ und t_ϵ.

$$l_n = O\left(5^n\right)$$

$$
\begin{aligned}
&= O\left(5^{\frac{\ln\left(\frac{\ln(1/\epsilon_n c_{approx}^2)}{\ln(1/\epsilon_0 c_{approx}^2)}\right)}{\ln(3/2)}}\right) \\
&= O\left((e^{\ln 5})^{\frac{\ln\left(\frac{\ln(1/\epsilon_n c_{approx}^2)}{\ln(1/\epsilon_0 c_{approx}^2)}\right)}{\ln(3/2)}}\right) \\
&= O\left(\left(e^{\ln\left(\frac{\ln(1/\epsilon_n c_{approx}^2)}{\ln(1/\epsilon_0 c_{approx}^2)}\right)}\right)^{\frac{\ln 5}{\ln(3/2)}}\right) \\
&= O\left(\left(\frac{\ln(1/\epsilon_n c_{approx}^2)}{\ln(1/\epsilon_0 c_{approx}^2)}\right)^{\frac{\ln 5}{\ln(3/2)}}\right)
\end{aligned}
$$

Da $\ln(1/\epsilon_0 c_{approx}^2)$ und $\frac{\ln 5}{\ln(3/2)}$ konstant sind, folgt

$$
\begin{aligned}
l_n &= O\left(\ln^{\frac{\ln 5}{\ln(3/2)}} \frac{1}{\epsilon_0 c_{approx}^2}\right) \\
&= O\left(\ln^{\frac{\ln 5}{\ln(3/2)}} \frac{1}{\epsilon_0} - \ln^{\frac{\ln 5}{\ln(3/2)}} c_{approx}^2\right) \\
&= O\left(\ln^{\frac{\ln 5}{\ln(3/2)}} \frac{1}{\epsilon_0}\right) .
\end{aligned}
$$

Im letzten Schritt wurde die subtrahierte Konstante weggelassen. Die Rechnung für t_ϵ ist analog. Insgesamt kommt man auf

$$
\begin{aligned}
l_\epsilon &= O\left(\ln^{\ln(5)/\ln(3/2)} \frac{1}{\epsilon}\right) \\
t_\epsilon &= O\left(\ln^{\ln(3)/\ln(3/2)} \frac{1}{\epsilon}\right) .
\end{aligned}
$$

3.3 Solovay-Kitaev-Algorithmus für Multi-Qubit- und -Qudit-Systeme

In diesem Kapitel soll die Verallgemeinerung des Solovay-Kitaev-Algorithmus für Qudits vorgestellt werden. Der Pseudocode ändert sich dabei nur in der sechsten Zeile. Der Unterschied liegt darin, dass für $d^n = 2$ das Δ exakt mit einem Gruppenkommutator darstellbar ist, während mit größeren Dimensionen Δ nur approximiert werden kann.

```
1 function Solovay-Kitaev(Gate U, depth n)
2 if(n == 0)
3         Return Basic Approximation to U
4 else
5         Set U_{n-1} = Solovay-Kitaev(U,n-1)
6         Set V, W = GC-Approx-Decompose(U U_{n-1}^†)
7         Set V_{n-1} = Solovay-Kitaev(V,n-1)
8         Set W_{n-1} = Solovay-Kitaev(W,n-1)
9         Return U_n = V_{n-1} W_{n-1} V_{n-1}^† W_{n-1}^† U_{n-1}
```

Im Qubit-Fall waren V und W gesucht, die für ein $\Delta = VWV^\dagger W^\dagger$ mit $d(I, \Delta) < \epsilon_{n-1}$

$$
d(I, V) < c_{gc}\sqrt{\epsilon_{n-1}}
$$

$$d(I, W) < c_{gc}\sqrt{\epsilon_{n-1}}$$

erfüllen. Im allgemeinen Qudit-Fall lauten die Bedingungen für V und W

$$\begin{aligned} d(VWV^\dagger W^\dagger, \Delta) &< c_{gc'}\epsilon_{n-1}^{3/2} \\ d(I, V) &< c_{gc''}\sqrt{\epsilon_{n-1}} \\ d(I, W) &< c_{gc''}\sqrt{\epsilon_{n-1}} \ . \end{aligned}$$

Um zu zeigen, dass es V und W gibt, die diese Bedingungen erfüllen, benötigen wir zunächst die folgenden drei Lemmata:

Lemma 3.5. *Sei $d^n = D$ und H eine hermitesche D-dimensionale Matrix mit $Spur(H) = 0$. Dann gibt es hermitesche Matrizen F und G, sodass*

$$\begin{aligned} [F, G] &= iH \\ \|F\| &\leq D^{1/4}\left(\frac{D-1}{2}\right)^{1/2}\sqrt{\|H\|} \\ \|G\| &\leq D^{1/4}\left(\frac{D-1}{2}\right)^{1/2}\sqrt{\|H\|} \ . \end{aligned}$$

Beweis. [DAW, 2008] □

Lemma 3.6. *Seien F und G zwei hermitesche Matrizen mit $\|F\| < \delta$ und $\|G\| < \delta$. Dann gilt für eine Konstante $c_1 \approx 4$*

$$d(\exp(iF)\exp(iG)\exp(-iF)\exp(-iG), \exp(i \cdot i[F, G])) \leq c_1\delta^3 \ .$$

Beweis. In [DAW, 2008] verwiesen. □

Lemma 3.7. *Jeder unitäre Operator U kann geschrieben werden als $U = \exp^{iH}$, wobei H ein hermitescher Operator ist mit $\|H\| \leq \pi$.*

Beweis. Da jeder unitäre Operator normal ist, ist er diagonalisierbar mit einer Diagonalmatrix D und einem unitären Operator A, sodass $U = ADA^\dagger$. Für alle Diagonalelemente gilt $\|d_i\| = 1$. Damit kann auch D geschrieben werden als $D = \exp^{iD'}$. Für die Eigenwerte von D' gilt $\|D'_i\| \leq \pi$, da die Bedingung für die Diagonalelemente nur den Einheitskreis einschließt. Somit folgt

$$\begin{aligned} U &= A\exp^{iD'}A^\dagger \\ &= \exp^{iAD'A^\dagger} \\ &= \exp^{iH} \end{aligned}$$

mit $H := AD'A^\dagger$. Es folgt offensichtlich, dass H hermitesch ist und $\|H\| = \|D'\| \leq \pi$. □

Satz 3.8. *Sei Δ unitär mit $d(I, \Delta) < \epsilon$. Dann gibt es unitäre V und W mit*

$$\begin{aligned} d(VWV^\dagger W^\dagger, \Delta) &< c_{gc'}\epsilon^{3/2} \\ d(I, V) &< c_{gc''}\sqrt{\epsilon} \\ d(I, W) &< c_{gc''}\sqrt{\epsilon} \ . \end{aligned}$$

Beweis. Sei $d(I, \Delta) < \epsilon$. Dann können wir eine hermitesche Matrix H finden mit $\|H\| \leq \pi$, sodass $\Delta = \exp(-iH)$. Es gilt

$$d(I, \exp^{iH}) = max_E\|(1 - \exp^{iE})\exp^{-i\frac{E}{2}}\|$$

$$
\begin{aligned}
&= max_E \| \exp^{-i\frac{E}{2}} - \exp^{i\frac{E}{2}} \| \\
&= max_E 2 \| \sin\left(\frac{E}{2}\right) \| \\
&> \|H\| - b\|H\|^3 \ .
\end{aligned}
$$

Der letzte Schritt folgt aus der Tatsache, dass die Eigenwerte zwischen $-\pi$ und π liegen und sich somit höhere Terme der Reihenentwicklung vom Sinus so ergeben, dass die Abhängigkeit von $\|H\|^3$ die Obergrenze ist. Es folgt also

$$
\begin{aligned}
& a\|H\| \leq \|H\| - b\|H\|^3 < d(I, \Delta) < \epsilon \\
\Leftrightarrow & \|H\| < \frac{\epsilon}{a}
\end{aligned}
$$

wobei a und b Konstanten sind und $a \approx 1$ [DAW, 2008]. Nach Lemma 3.5 können wir zwei hermitesche Matrizen F und G finden, sodass

$$
\begin{aligned}
[F, G] &= iH \\
\|F\| &\leq D^{1/4}((D-1)/2)^{1/2}\sqrt{\|H\|} \\
&= c_{gc''}\sqrt{\epsilon} \\
\|G\| &\leq D^{1/4}((D-1)/2)^{1/2}\sqrt{\|H\|} \\
&= c_{gc''}\sqrt{\epsilon} \ .
\end{aligned}
$$

Da $\sqrt{\|H\|} < \sqrt{\frac{\epsilon}{a}}$ ist, ergibt sich die Approximation $c_{gc''} \approx D^{1/4}((D-1)/2)^{1/2}$. Setzt man nun $V = \exp(iF), W = \exp(iG)$ und $\delta = c_{gc''}\sqrt{\epsilon}$ in Lemma 3.5 ein, so bekommt man mit $c_{gc'} = c_1 {c_{gc''}}^3 \approx 4D^{3/4}((D-1)/2)^{3/2}$

$$
d(VWV^\dagger W^\dagger, \Delta) < c_1(c_{gc''}\sqrt{\epsilon})^3 = c_{gc'}\epsilon^{3/2} \ .
$$

Aus $\|F\| \leq c_{gc''}\sqrt{\epsilon}$ und $\|G\| \leq c_{gc''}\sqrt{\epsilon}$ folgt $d(I, V) < c_{gc''}\sqrt{\epsilon}$ und $d(I, W) < c_{gc''}\sqrt{\epsilon}$. □

Die restlichen Zeilen des Algorithmus laufen ebenso wie im Qubit-Fall, weshalb ϵ_n sowie die Länge der Gatefolge l_n und die Laufzeit t_n wie zuvor skalieren.

$$
\begin{aligned}
\epsilon_n &= O\left(\epsilon_{n-1}^{\frac{3}{2}}\right) \\
l_\epsilon &= O\left(\ln^{\ln(5)/\ln(3/2)}\frac{1}{\epsilon}\right) \\
t_\epsilon &= O\left(\ln^{\ln(3)/\ln(3/2)}\frac{1}{\epsilon}\right) \ .
\end{aligned}
$$

Wie in Kapitel 3.2 bereits erwähnt, muss die erste, also die ϵ_0-Approximation nachgeschlagen werden. Diese Tabelle wird durch Aufzählung aller möglichen Gate-Folgen einer bestimmten Länge erstellt. Da $SU(D)$ mit $D = d^n$ eine Manigfaltigkeit der Dimension $D^2 - 1$ ist, braucht man $O(1/\epsilon_0^{D^2-1})$ Gatefolgen. In einem universellen Gate-Set $\mathcal{G}$ gibt es $O(|\mathcal{G}|^l)$ Gatefolgen mit einer Länge $\leq l$, wobei $|\mathcal{G}|$ die Anzahl der Gates in dem Set ist. Nach [DAW, 2008] müssen alle Gatefolgen der Länge

$$
l_0 \leq O\left(\frac{d^{n^2}-1}{\log|\mathcal{G}|}\log(1/\epsilon_0)\right)
$$

in die Tabelle aufgenommen werden. Da l_0 exponentiell mit n ansteigt, ist dies für größere Werte für n ein sehr großer Rechenaufwand.

3.4 Schlussfolgerungen

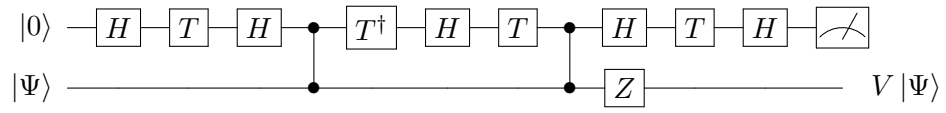
Mit dem Solovay-Kitaev-Algorithmus kann zu einer beliebigen, unitären Operation in der Zeit $t_\epsilon = O\left(\ln^{\ln(3)/\ln(3/2)} \frac{1}{\epsilon}\right) \approx O\left(\ln^{2.71} \frac{1}{\epsilon}\right)$ eine ϵ-Approximation erstellt werden.
Nach [NIE, 2010] wäre bei der Zerlegung eine quadratische Abhängigkeit der Gesamtlänge der benötigten Gates intuitiv. Solovay-Kitaev-Algorithmus ist der erste explizite und effiziente Zerlegungsalgorithmus, dessen Gateanzahl polylogarithmisch in der inversen Genauigkeit wächst, was eine große Verbesserung gegen über der quadratischen Abhängigkeit ist. Das Verfahren ist auch für Multi-Qubit und -Qudit-Systeme möglich, die Länge der Gate-Sequenz wächst allerdings exponentiell in der Anzahl der Qubits bzw. Dimension der Qudits.
Wichtig für den Vergleich mit andern Algorithmen ist an dieser Stelle noch, dass das Solovay-Kitaev-Verfahren nur eine obere Schranke für die Gesamtzahl der benötigten Gates hergibt und keine obere Schranken für einen besonderen Typus von Gates, wie beispielsweise die T-Gates, worauf sich die anderen in dieser Arbeit vorgestellten Verfahren konzentrieren. Als Obergrenze für die T-Gate-Anzahl kann man allein die Gesamtzahl an Gates betrachten. Sie ist dementsprechend nur beschränkt durch $O\left(\ln^{\ln(5)/\ln(3/2)} \frac{1}{\epsilon}\right) \approx O\left(\ln^{3.96} \frac{1}{\epsilon}\right)$.

4 Repeat-Until-Success-Circuit

Im Folgenden soll der „Repeat-Until-Success“-Algorithmus vorgestellt werden, der unitäre Einzel-Qubit-Operationen teilweise exakt, teilweise mit einer Ungenauigkeit ϵ in eine Folge von Gates aus einem Gate-Set zerlegen kann. Beispielsweise kann als Set die Clifford+T-Basis $\{T, H, S, CNOT\}$ gewählt werden. Der Algorithmus ist allerdings auch leicht auf andere Sets übertragbar [PAET, 2014].
Anders als bei dem Solovay-Kitaev-Algorithmus und dem im fünften Kapitel behandelten Zerlegungsverfahren, handelt es sich hier weniger um ein analytisches Vorgehen. Vielmehr wird die zu approximierende Operation durch Schaltkreise aus einer umfangreichen Datenbank zusammengesetzt, die gemäß eines bestimmten Protokolls erstellt wird. Es wird dabei die Zerlegung gewählt, die die geringste Anzahl an T-Gates hat. Für den Vergleich mit anderen Algorithmen wird die T-Gate-Anzahl als Kostenreferenz benutzt. Im Gegensatz zu vielen dieser benutzt der „Repeat-Until-Success“-Algorithmus Hilfsqubits und Messungen, zeichnet sich allerdings durch eine viel geringere T-Gate-Anzahl aus, verglichen mit Algorithmen, die ohne Hilfsqubits und Messungen arbeiten.
In diesem Kapitel soll das Prinzip des Algorithmus zur Datenbankerstellung und seine Implementierung vorgestellt sowie jene Gates charakterisiert werden, die exakt zerlegt werden können. Abschließend wird erläutert, wie eine beliebige Operation mit Hilfe der Datenbank implementiert werden kann.

4.1 Repeat-Until-Success-Quantenschaltkreise

Um den Algorithmus zu verstehen, wird zunächst ein Beispiel präsentiert.



Mit dem abgebildeten Quantenschaltkreis kann

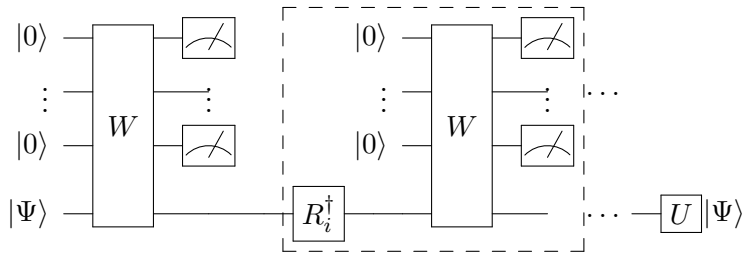
$$V_{Erfolg} = \frac{3+i}{\sqrt{10}} |0\rangle\langle 0| - \frac{1+3i}{\sqrt{10}} |1\rangle\langle 1|$$

implementiert werden. Die Hilfsqubits werden zunächst auf 0 gesetzt. Mit einer Wahrscheinlichkeit von $\frac{5}{8}$ ergibt die Messung 0. In diesem Fall wirkt V_{Erfolg} auf Ψ. In dem anderen Fall wirkt

$$V_{Misserfolg} = \frac{((-1+i)+(-1)^{1/4})}{\sqrt{3}}(|0\rangle\langle 0| + |1\rangle\langle 1|)$$

mit einer Wahrscheinlichkeit von $\frac{3}{8}$ (Rechnungen siehe Anhang). Falls kein Erfolg erzielt wurde, wird $V^{\dagger}_{Misserfolg}$ angewandt und der Quantenschaltkreis kann erneut ausgeführt werden. Dieser Vorgang wird wiederholt bis die Messung ergibt, dass sich das Hilfsqubit im Zustand 0 befinden. Aus diesem Grund heißt der im Folgenden vorgestellte Algorithmus „Repeat-Until-Success“-Algorithmus. Die inverse Misserfolgsoperation ist dabei stets eine einfache Operation, die hier der Identität mal einer globalen Phase entspricht.
Auf einem Qubit wird der Eingangszustand $|\Psi\rangle$ eingeführt. Weitere m Qubits werden als Hilfsqubits auf $|0\rangle$ initialisiert. Auf alle $m+n$ Qubits wirkt die unitäre Operation W, die sich, um

die Implementierbarkeit gewährleisten zu können, aus Gates des Sets $\mathcal{G}$ zusammensetzt. Jedes der Hilfsqubits wird in der Berechnungsbasis (computational basis) gemessen, wodurch man ein Ergebnis $i \in \{0,1\}^m$ erhält. Der Ausgangszustand $\Phi_i |\Psi\rangle$ ist von diesem Ergebnis abhängig. Dabei lassen sich die Operationen Φ_i mit $\Phi_i \in \{i : Erfolg\}$ und $\Phi_i \in \{i : Misserfolg\}$ einteilen in gewünschte und unerwünschte Operationen. In dem obigen Beispiel ist die gewünschte Operation V_{Erfolg} und die unerwünschte Operation $V_{Misserfolg}$. Wurde bei der Messung ein Erfolg festgestellt, endet der Algorithmus. Andernfalls wird die Operation Φ_i^{-1} ausgeführt und der Schaltkreis wird erneut angewendet.
In dieser allgemeinen Abbildung ist die gewünschte Operation U. Alle unerwünschten Operationen werden R_i genannt. Nach Feststellen eines Misserfolgs wird dementsprechend $R_i^\dagger$ angewendet und W erneut ausgeführt. Diese markierte Abfolge wird wiederholt ausgeführt bis ein Erfolg erzielt wurde.



W ist eine unitäre Matrix der Dimension $2^{m+1} \times 2^{m+1}$ und hat die Form

$$\begin{aligned} W =& \frac{1}{\sqrt{\sum_i |\alpha_i|^2}} \begin{pmatrix} \alpha_0 U \cdots \\ \alpha_1 R_1 \ddots \\ \vdots \\ \alpha_l R_l \cdots \end{pmatrix} \\ =& \frac{1}{\sqrt{\sum_i |\alpha_i|^2}} (\alpha_0 |0^m\rangle \langle 0^m| \otimes U + \alpha_1 |0^{m-1}1\rangle \langle 0^m| \otimes R_1 + ... + \alpha_l |1^m\rangle \langle 0^m| \otimes R_l \\ & + \beta_0 |0^m\rangle \langle 0^{m-1}1| \otimes B_1 + ...) \end{aligned}$$

wobei $\alpha_i \in \mathbb{C}$ und $U, R_1, ...R_l$ unitäre 2×2 Matrizen sind. Da die Hilfsqudits auf 0 initialisiert sind, sind nur die ersten zwei Spalten von W interessant. Dies wird in der Dirac-Schreibweise deutlich.

$$W |0^m\rangle |\Psi\rangle = \frac{1}{\sqrt{\sum_i |\alpha_i|^2}} (\alpha_0 |0^m\rangle U |\Psi\rangle + \alpha_1 |0^{m-1}1\rangle R_1 |\Psi\rangle + ... + \alpha_l |1^m\rangle R_l |\Psi\rangle)$$

Für die restlichen Spalten muss nur erfüllt sein, dass W unitär ist. l ist dabei definiert durch

$$\begin{aligned} & 2l + 2 = 2^{m+1} \\ \Leftrightarrow \quad & l + 1 = 2^m, \end{aligned}$$

wobei 2^m die Anzahl der möglichen Messergebnisse ist.

4.2 Charakterisierung exakt darstellbarer Operationen

Wie oben erwähnt, muss W mit $\mathcal{G}$ darstellbar sein, wenn eine exakte Implementierung gewünscht ist. Wählt man als $\mathcal{G}$ die „Clifford+T-Basis“ müssen die Matrixelemente von $\{\alpha_0 U, \alpha_1 R_1, ...\alpha_l R_l\}$ und $\frac{1}{\sqrt{\sum_i |\alpha_i|^2}}$ aus dem Ring $\mathbb{Z}\{i, \frac{1}{\sqrt{2}}\}$ sein. Dies folgt aus der Tatsache, dass für $n = 1$ das Set von $2^n \times 2^n$ unitären Matrizen über dem Ring $\mathbb{Z}\{i, \frac{1}{\sqrt{2}}\}$ äquivalent

zu dem Set von unitären Matrizen ist, die exakt in der Clifford+T-Basis implementiert werden können [KLIU, 2013]. Sei $N := \sum_{i=0}^{l} |\alpha_i|^2$. Da

$$\begin{aligned}\sum_{i=0}^{l}(\alpha_i R_i)(\alpha_i R_i)^\dagger &= \sum_{i=0}^{l} |\alpha_i|^2 R_i R_i^\dagger \\ &= N\mathbb{1}\end{aligned}$$

und die Matrixelemente von $\sum_{i=0}^{l}(\alpha_i R_i)(\alpha_i R_i)^\dagger$ aus dem Ring $\mathbb{Z}\{i, \frac{1}{\sqrt{2}}\}$ sind, muss somit auch N aus diesem Ring sein. Nun fordert man also auch, dass $\frac{1}{\sqrt{N}}$ aus dem Ring ist. Daraus folgt, dass auch $\frac{1}{N} \in \mathbb{Z}\{i, \frac{1}{\sqrt{2}}\}$. Die Elemente, die selbst und deren inverses Element im Ring liegen, müssen Potenzen von 2 sein. Es muss also gelten

$$N = 2^k \ .$$

Mit dem Vier-Quadrate-Satz von Lagrange, der besagt, dass jede natürliche Zahl als Summe von vier Quadratzahlen geschrieben werden kann, folgt, dass diese Gleichung für $l = 2$ erfüllt werden kann, selbst wenn wie im Folgenden $\alpha_0 \in \mathbb{Z}$ angenommen wird. Man beachte, dass die komplexen Summanden $|\alpha_1|^2$ und $|\alpha_2|^2$ jeweils der Summe zweier Quadratzahlen entsprechen. Aus dem minimalen $l = 2$ folgt mit $l + 1 = 2^m$, dass mindestens $m = 2$ gelten muss, also zwei Hilfqubits benötigt werden.
Sei U eine unitäre 2×2 Matrix mit $k \geq 0$.

$$U = \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix} = \frac{1}{\sqrt{2^k}\alpha} \begin{pmatrix} \beta_{00} & \beta_{01} \\ \beta_{10} & \beta_{11} \end{pmatrix}$$

Da U bis auf eine globale Phase implementiert werden soll, kann o.B.d.A. α als reell und nichtnegativ angenommen werden. Zudem fordern wir $\beta \in Z[i, \sqrt{2}]$ statt $\beta \in Z[i, 1/\sqrt{2}]$, da k in diesem Fall nur anders gewählt werden muss. Wählen wir $\alpha_0 = \sqrt{2^c}\alpha$, so muss aufgrund der Unitarität von U gelten

$$\alpha_0 = \sqrt{|\beta_{00}|^2 + |\beta_{10}|^2} \ .$$

Die unitären Operationen U der obigen Form lassen sich folglich exakt mit dem Algorithmus darstellen.

4.3 Erfolgswahrscheinlichkeit und Kosten

Aus $\alpha_0^2 < 2^k$ folgt $k \geq \lceil 2 \log_2 \alpha_0 \rceil$. Die Wahrscheinlichkeit nur Nullen in den Hilfsqubits zu messen und somit die gewünschte Operation zu erreichen ist somit

$$P[Erfolg] = \frac{\alpha_0^2}{2^k} \leq \frac{\alpha_0^2}{2^{\lceil 2 \log_2 \alpha_0 \rceil}} \ .$$

Der Erwartungswert an benötigten Wiederholungen ist dementsprechend $\frac{1}{P[Erfolg]}$, die Varianz ergibt sich gemäß der geometrischen Verteilung zu $\frac{1-P[Erfolg]}{P^2[Erfolg]}$. Die Kosten für die Umsetzung des Algorithmus hängen natürlich mit der Erfolgswahrscheinlichkeit zusammen. Sei $C(W)$ die Anzahl der T-Gates, die gebraucht werden, um W zu implementieren. So ist der Erwartungswert der im Algorithmus gebrauchten T-Gates $\frac{C(W)}{P[Erfolg]}$. Die Varianz beträgt $C(W)\frac{1-P[Erfolg]}{P^2[Erfolg]}$. [PAET, 2014] betrachten neben der T-Gate-Anzahl auch die Größe n und die nötigen Messungen m als Kosten. Da aber in vielen Algorithmen Messungen nicht gebraucht werden, ist die Anzahl der T-Gates ein guter Vergleichswert.

Um die erwarteten Kosten möglichst niedrig zu halten, ist es nötig $P[Erfolg]$ möglichst groß werden zu lassen. Allgemein kann man die Wirkung von W schreiben als

$$W\left|0^m\right\rangle\left|\Psi\right\rangle = \sqrt{p}\left|0^m\right\rangle U\left|\Psi\right\rangle + \sqrt{1-p}\left|\Phi^{\perp}\right\rangle$$

wobei $\left|\Phi^{\perp}\right\rangle$ von $\left|\Psi\right\rangle$ abhängig ist und $(\left|0^m\right\rangle\left\langle 0^m\right| \otimes \mathbb{1})\left|\Phi^{\perp}\right\rangle = 0$ erfüllen muss. Dabei ist $\sqrt{p}$ die Erfolgswahrscheinlichkeit und $\sqrt{1-p}$ die Wahrscheinlichkeit Misserfolg zu haben.
Um p zu vergrößern kann man Amplitudenverstärkung anwenden. Dabei wird der Operator $(-WSW^{\dagger}S)^j$ auf $W\left|0^m\right\rangle\left|\Psi\right\rangle$ angewendet für eine ganze Zahl $j > 0$. Dabei ist

$$S = \overline{CZ}(m) \otimes I$$

mit $\overline{CZ}(m) = X^{\otimes m} CZ(m) X^{\otimes m}$, wobei $CZ(m)$ ein generalisiertes controlled-Z-Gate ist, das auf m Qubits definiert ist durch

$$CZ(m)\left|x_1, x_2, ..., x_m\right\rangle = (-1)^{x_1 x_2 ... x_m}\left|x_1, x_2, ..., x_m\right\rangle \text{ [PAET, 2014]}.$$

Für die Verstärkung gilt folgender Satz:

Satz 4.1. *Sei* $W\left|0^m\right\rangle\left|\Psi\right\rangle = \sqrt{p}\left|0^m\right\rangle U\left|\Psi\right\rangle + \sqrt{1-p}\left|\Phi^{\perp}\right\rangle$ *unitär. Dann gilt für jedes* $j \in \mathbb{Z}$*:*

$$(-WSW^{\dagger}S)^j (W\left|0^m\right\rangle\left|\Psi\right\rangle) = \sin((2j+1)\theta)\left|0^m\right\rangle U\left|\Psi\right\rangle + \cos((2j+1)\theta)\left|\Phi^{\perp}\right\rangle$$

wobei $\sin(\theta) = \sqrt{p}$*.*

Beweis. Erfolgt in [PAET, 2014]. □

Das Ziel ist nun j so zu wählen, dass der Sinus-Ausdruck möglichst groß wird. Die Amplitudenverstärkung soll also bewirken, dass

$$\frac{T_{Anzahl,neu}}{P[Erfolg]_{neu}} < \frac{T_{Anzahl}}{P[Erfolg]} \ .$$

Mit Satz 4.1 und unter Beachtung, dass immer $2j$ T-Gates durch die Amplitudenverstärkung hinzugefügt werden, folgt daraus

$$\frac{(2j+1)T_{Anzahl}}{\sin^2((2j+1)\Theta)} < \frac{T_{Anzahl}}{\sin^2(\Theta)} \ .$$

Der Erwartungswert des T-Counts sinkt deshalb, wenn

$$(2j+1)\sin^2(\theta) < \sin^2((2j+1)\theta)$$

ist. Diese Bedingung kann nach [PAET, 2014] nur erfüllt werden, wenn $0 \leq P[Erfolg] < \frac{1}{3}$ gilt. Somit ist es nur sinnvoll die Amplitudenvertärkung anzuwenden, wenn die Wahrscheinlichkeit zuvor $< \frac{1}{3}$ war. Ansonsten verläuft der Algorithmus unverändert.

4.4 Erstellung der Datenbank

Wie man das oben beschriebene W implementiert, wurde bisher nicht erläutert. Bislang gibt es kein Verfahren, das W mit minimaler T-Gate-Anzahl zerlegt. Allerdings ist es möglich, eine Datenbank von Zerlegungen zu erstellen und die jeweilige T-Gate-Anzahl anzugeben. [PAET, 2014] benutzt dafür folgende Schritte:

1. Wahl von der Anzahl der Hilfsqubits m und der Anzahl der Gates
2. Konstruktion eines Schaltkreises mit der „Clifford+T-Basis“ und Berechnung der unitären, sich daraus ergebenden Matrix W

3. Zerteilen der ersten zwei Spalten von W in 2×2 Matrizen

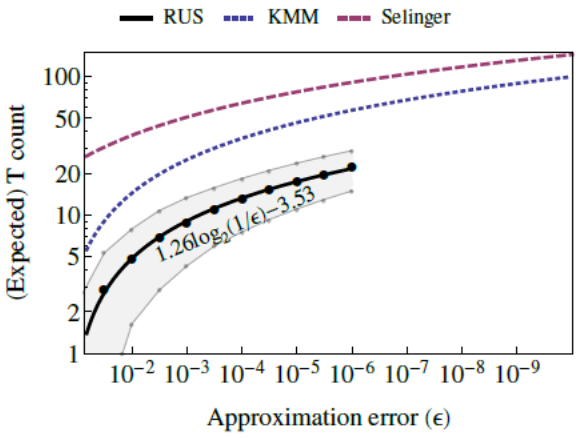
4. Finden und Markieren von all jenen Matrizen, die proportional zu einem einzelnen „Clifford“ Gate sind

5. Sind alle übrigen Matrizen proportional zu einer gleichen unitären Matrix, ist diese die gewünschte Operation und die Quantenschaltung wird beibehalten. Andernfalls wird die Quantenschaltung aussortiert.

In Schritt 4 werden T-Gates nicht erlaubt, da die Rücktransformation die T-Gate-Anzahl nicht erhöhen soll. Die nach diesem Verfahren erzeugten Schaltkreise weisen als Misserfolgsoperationen einfache Clifford-Gates auf. Wichtig ist in dem Algorithmus zudem, dass man eine kanonische Form benutzt, um die Einzel-Qubit-Operationen vergleichen zu können. Andernfalls bestünde die Möglichkeit, dass sich zwei Quantenschaltungen in der Datensammlung befinden, die die selbe Einzel-Qubit-Operation beschreiben, aber nicht als gleich identifiziert werden können.
[PAET, 2014] haben eine Datenbank aus Schaltkreisen mit einem Hilfsqubit und bis zu 15 T-Gates erstellt. Dabei wurde zuerst mit dem Algorithmus eine Schaltkreis-Sammlung erzeugt, in der allerdings noch Schaltkreise enthalten waren, die mit unterschiedlichen Erfolgswahrscheinlichkeiten und T-Gate-Anzahlen die selbe unitäre Operation darstellen. In diesen Fällen wurde der Schaltkreis ausgewählt, der die wenigsten T-Gates benötigt. Schaltkreise mit mehr als 15 T-Gates wurden ebenfalls ausgeschlossen. Die resultierende Datenbank beinhaltet 2194 Schaltkreise. Für Erfolgswahrscheinlichkeiten unter $\frac{1}{3}$ wurde Amplitudenverstärkung angewendet.

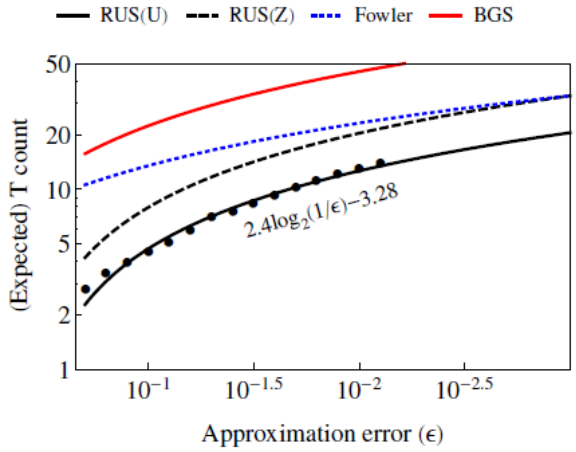
4.5 Approximieren einer vorgegebenen Operation

Auf der Basis der erstellten Datenbank können beliebige Operationen approximiert werden. Zuvor ist es sinnvoll die Datenbank zu vergrößern, in dem man beliebige Kombinationen der enthaltenen Schaltungen vornimmt. Von diesen neuen Schaltungen werden jene ebenfalls in die Datenbank aufgenommen, die eine unitäre Operation darstellen, die zuvor noch nicht dargestellt wurde. Die Menge N_k der Schaltungen, die aus k ursprünglichen Schaltungen zusammengesetzt wurde, ist mit $N_k \leq N_1^k$ beschränkt, wenn N_1 die Anzahl der Schaltungen aus der noch nicht vergrößerten Datenbank ist. Zudem werden wieder nur Schaltungen mit einer T-Gate-Anzahl bis zu einer festen Zahl T_0 aufgenommen.
[PAET, 2014] erweiterten die Datenbank auf diese Weise für Axialrotationen bis zu einer erwarteten T-Gate-Anzahl von 30. Dies erlaubt eine Genauigkeit von ca. $\epsilon = 10^{-6}$. Die Skalierung der T-Gate-Anzahl ergibt sich in diesem Bereich numerisch zu $1.26 \log_2(\frac{1}{\epsilon})$.



[PAET, 2014]

Ebenfalls wurde eine Erweiterung für beliebige Ein-Qubit-Operationen bis zu einer erwarteten T-Gate-Anzahl von 18 durchgeführt. Dies erlaubt eine Genauigkeit von ca. $\epsilon = 10^{-3}$. Hier ergibt sich die Skalierung des T-Gate-Anzahl in diesem Bereich numerisch zu $2.4 \log_2(\frac{1}{\epsilon})$.



PAET, 2014

Hat man nun eine beliebige, zu approximierende unitäre Operation U gegeben, werden alle Einträge aussortiert, für die gilt $D(U, V) \leq \epsilon$. Dabei ist

$$D(U, V) = \sqrt{\frac{2 - |Tr(U^\dagger V)|}{2}}$$

eine bekannte Metrik PAET, 2014 und ϵ die gewünschte Genauigkeit der Approximation. Von diesen Schaltkreisen wird nun der mit der minimalen T-Gate-Anzahl gewählt.

4.6 Schlussfolgerungen

Nachdem eine Datenbank mit dem Such-Algorithmus erzeugt wurde, ist es möglich eine beliebige Operation zu approximieren. Dabei muss bei der Erstellung der Datenbank bereits entschieden werden, wie groß die T-Gate-Anzahl ist. Numerisch kann dann die erreichte Genauigkeit erschlossen werden. PAET, 2014 kommen numerisch in dem von ihnen untersuchten Bereich auf eine Skalierung der T-Gate-Anzahl mit $2.4 \log_2(\frac{1}{\epsilon})$.
Bei Axialrotationen ist es möglich die Rotationen der Größe der Winkel nach zu sortieren und eine binäre Suche anzuschließen. Im allgemeinen Fall muss die Liste von oben nach unten ohne Muster durchsucht werden. Die Laufzeit bei der Approximation einer beliebigen Operation ist somit mit einer Konstante beschränkt, die von der Länge der Liste abhängt. Interessant ist des Weiteren die Frage, ob sich der „Repeat-Until-Success“-Algorithmus auf Qubits verallgemeinern lässt. Das Prinzip eine Operation mit einer Erfolgswahrscheinlichkeit auszuüben und bei Misserfolg die entsprechende Rücktransformation anzuwenden, um die Operation zu wiederholen, ist einfach auf Qudits zu übertragen. Statt der Clifford+T-Basis in Schritt 2 muss ein anderes universelles Gate-Set für Qudits gewählt werden. Auch für die zu markierenden Rücktransformationen in Schritt 4 muss eine entsprechend andere Regel entworfen werden. Nicht klar ist, wie umfangreich und nützlich eine solche Datenbank wäre. Eine noch zu lösende Aufgabe, für die mir nicht genug Computerleistung zur Verfügung steht, bleibt dementsprechend eine solche Datenbank zu erstellen.

5 Zerlegung von diagonalen unitären Operatoren

In diesem Teil der Arbeit wird ein Algorithmus thematisiert, der diagonale, unitäre Qubit-Operatoren zerlegt. In der Originalveröffentlichung wird von einer Zerlegung in die Clifford+T-Basis gesprochen. Gezeigt wird dabei nur die Zerlegung in Toffoli-, NOT- und Phasengates. Deren Darstellung in der Clifford+T-Basis wird als bekannt vorausgesetzt. Dieser von [WEL, 2015] für Qubits beschriebene Algorithmus soll hier auf Qudits verallgemeinert werden. Folglich zeige ich hier auch eine Zerlegung in Phasen-Gates $R(\phi)$, Toffoli-Gates und NOT-Gates. Von letzteren existieren $d-1$ Versionen, die im Extremfall alle gebraucht werden.

5.1 Zerlegung in „cascaded entanglers“ und Phasengates

Wir betrachten einen diagonalen Operator U auf n Qudits mit $k << d^n$ verschiedenen Phasen, wobei

$$U = diag(\underbrace{\phi_1, ...\phi_1}_{l_1}, \underbrace{\phi_2, ..., \phi_2}_{l_2}, ..., \underbrace{\phi_k, ..., \phi_k}_{l_k}) \ .$$

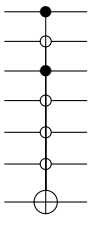
Dieser soll nun in $k-1$ diagonale Operatoren der Form

$$V(\phi, l) = diag(\underbrace{1, ...1}_{d^n - l}, \underbrace{\phi, ...\phi}_{l})$$

zerlegt werden. Mit $L_m = \sum_{j=1}^{j=m} l_j$ gilt

$$U = \phi_1 V(\frac{\phi_2}{\phi_1}, L_k - L_1) V(\frac{\phi_3}{\phi_2}, L_k - L_2) V(\frac{\phi_4}{\phi_3}, L_k - L_3) \ ... \ V(\frac{\phi_k}{\phi_{k-1}}, L_k - L_{k-1})$$

(fehlerhaft in [WEL, 2015]).
Nun müssen die einzelnen V-Gates mit Hilfe zweier multi-controlled-NOT-Gates und einem Phasengate zerlegt werden. Die multi-controlled-NOT-Gates besitzen hier mehrere und variable Kontrolllevel. Man spricht an dieser Stelle von einem „variablen multi-controlled-NOT“-Gate, wie im Grundlagenkapitel bereits erklärt. Im Qubit-Fall kann ein solches Gate, das teilweise auf 0 und teilweise auf 1 prüft, zum Beispiel so aussehen:



Für die Zerlegung wird ein Hilfsqudit auf $|0\rangle$ gesetzt und ein Phasengate $R(\phi)$ definiert, das auf dieses angewendet wird.

$$R(\phi) = |0\rangle\langle 0| + \phi\,|1\rangle\langle 1| + \alpha_1\,|2\rangle\langle 2| + ... + \alpha_{d-2}\,|d-1\rangle\langle d-1|$$

Die α_i sind beliebig wählbar und für die Zerlegung uninteressant.
In der Originalquelle wird von einer Axialrotation um die z-Achse $P(\theta) = Exp(-i\theta Z/2)$ mit $\theta \in \mathbb{R}$ gesprochen, die auf das Hilfsqubit angewandt wird. Eigentlich bräuchte man an dieser Stelle einen Phasenoperator, der nur dem Zustand $|1\rangle$ eine Phase $\phi = e^{i\theta}$ zufügt. Wendet man also eine Axialrotation an, ist der Operator U nur auf eine globale Phase ($e^{i\theta/2}$) genau bestimmt.

$$\begin{aligned} R &= |0\rangle\langle 0| + \phi\,|1\rangle\langle 1| \\ P(\theta) &= e^{-i\theta/2}\,|0\rangle\langle 0| + e^{i\theta/2}\,|1\rangle\langle 1| \\ (e^{i\theta/2})P(\theta) &= |0\rangle\langle 0| + e^{i\theta}\,|1\rangle\langle 1| \end{aligned}$$

Dies gilt aber nur für den Spezialfall $d = 2$.
Im Anschluss sollen nun die zwei unitären multi-controlled-NOT-Gates $X_n(l)$ definiert werden, die „cascaded entanglers“ genannt werden. Für Qubits sind diese zwei nach [WEL, 2015] identisch, für Qudits werden hier zwei Versionen benötigt, von denen eine das Adjungierte der anderen ist.
Bevor die weitere Zerlegung der V-Gates erläutert werden kann, wird für eine beliebige Menge $J = \{|j_1\rangle, .., |j_l\rangle\}$ die Funktion Ω_J definiert:

$$\Omega_J(j) = \begin{cases} 1, & \text{falls } |j\rangle \in J \\ 0, & \text{sonst.} \end{cases}$$

$\Omega_J(j)$ wird Aktivierungsfunktion genannt [WEL, 2015]. Für $J = \{|j\rangle : j \geq d^n - l\}$ kann $V(\phi, l)$ nun wie folgt definiert werden:

$$V(\phi, l) = \sum_{j=0}^{2^n - l} \phi^{\Omega_J(j)}\,|j\rangle\langle j|\,.$$

Insgesamt kann $V(\Phi, l)$ implementiert werden durch

$$X_{n1}(l)(I_n \otimes P(\Phi))X_{n2}(l)\ .$$

Die „cascaded entanglers“ sind dabei mit $j \in [0, .., d^n - 1]$ und $b \in [0, .., d - 1]$ definiert als

$$\begin{aligned} X_{n1}(l)\,|j\rangle\,|b\rangle &= |j\rangle\,|b \oplus \Omega_J(j)\rangle \\ X_{n2}(l)\,|j\rangle\,|b\rangle &= |j\rangle\,|b \ominus \Omega_J(j)\rangle\ . \end{aligned}$$

$|j\rangle$ bezeichnet hier die eigentlichen Qudits, $|b\rangle$ ist das Hilfsqudit, das anfangs auf 0 initialisiert wird. Da X_{n2} die Rücktransformation von X_{n1} ist, gilt $X_{n2} = X_{n1}^\dagger$ und man kann mit

$$X_n(l)\,|j\rangle\,|b\rangle = |j\rangle\,|b \oplus \Omega_J(j)\rangle$$

vereinfacht schreiben

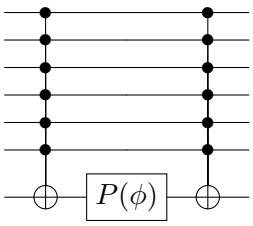
$$X_n(l)(I_n \otimes P(\Phi))X_n(l)^\dagger\ .$$

Da $V = V(\Phi, l)$ nur von Φ und l abhängt, können wir den Operator $X_n(l)$ umschreiben. l sei dargestellt in der Basis d.

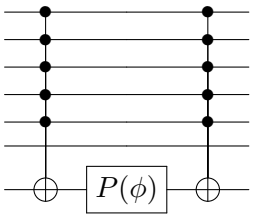
$$X_n(l)\,|j\rangle\,|b\rangle = \begin{cases} |j\rangle\,|b \oplus 1\rangle\,, & \text{falls } k \geq d^n - l \\ |j_1, ..., j_n, t\rangle\,, & \text{falls } k < d^n - l \end{cases}$$

Der Operator $X_n(l)$ ist nun so definiert, dass er angewendet auf $|j\rangle\,|0\rangle$ bewirkt, dass genau dann der Phasenfaktor ϕ aufgenommen werden kann, wenn $k \geq d^n - l$.

In der Abbildung ist das Hintereinanderschalten der „cascaded entangler“ und der Rotation für den Fall $l = 1, n = 6$ und $d = 2$ [WEL, 2015] dargestellt.



In diesem Beispiel ist X_6 einfach ein multi-controlled-NOT-Gate, da es nur einen möglichen String $[63] = \{1, 1, 1, 1, 1, 1\}$ gibt, der die Bedingung $k \geq 2^n - l$ erfüllt und mit sechs Bits darstellbar ist. Die 64 lässt sich erst mit sieben Bits darstellen. Für den Fall $l = 2$ gibt es bereits zwei Möglichkeiten $[63] = \{1, 1, 1, 1, 1, 1\}$ und $[62] = \{1, 1, 1, 1, 1, 0\}$. Auch für diesen Fall entspricht X_6 einem multi-controlled-NOT-Gate.



Die drei Möglichkeiten $[63] = 111111$, $[62] = 111110$ und $[61] = 111101$ im Fall $l = 3$ verhindern die Darstellung des X_6 mit einem einzigen multi-controlled-NOT-Gate. Bei $d > 2$ tritt dieses Problem ebenfalls auf. Es ist nötig, die „cascaded entangler“ X_n zu zerlegen. Dazu können wir diese mit $p \leq q \leq d^n$ verallgemeinern zu

$$X_n(p, q) \left|j\right\rangle \left|b\right\rangle = \left|j\right\rangle \left|b \oplus \Omega_{[p,q)}(j)\right\rangle \ .$$

Dieser Operator ändert b genau dann, wenn $p \leq j < q$ gilt. $X_n(l)$ ist somit

$$X_n(l) = X_n(d^n - l, d^n) \ .$$

5.2 Zerlegung der „cascaded entangler“

Nachdem U in V-Gates unterteilt und diese wiederum in zwei „cascaded entangler“ $X_n(l)$ und die Rotation $P(\phi)$ zerlegt wurden, wird im Folgenden gezeigt, dass ein $X_n(l)$ durch Hintereinanderschalten von variablen multi-controlled-NOT-Gates ersetzt werden kann.

Definition 5.1. *Für $n+1$ Qudits der Dimension d sei ein variables multi-controlled-NOT-Gate $\Lambda^{n[b]}(X)$ von der Form*

$$\Lambda^{n[b]}(X) \left|j_1, ..., j_n, t\right\rangle = \begin{cases} \left|j_1, ..., j_n, t \oplus 1\right\rangle, & \textit{falls } \bigwedge_{n=1}^{m} XNOR(b_k, j_k) = 1 \\ \left|j_1, ..., j_n, t\right\rangle, & \textit{sonst,} \end{cases}$$

wobei m die Anzahl der Stellen des Strings $[b]$ ist.

$\bigwedge_{n=1}^{m} XNOR(b_k, j_k) = 1$ ist äquivalent mit $\forall k : b_k = j_k$. Wie einfach zu überprüfen ist, stimmt diese Definition mit $d = 2$ überein mit [WEL, 2015] und ist auch für größere d sinnvoll.
Bei der Kostenrechnung für den Qubit-Fall werden die Kosten der Clifford-Gates vernach-

lässigt und nur die Anzahl der T-Gates gezählt. Nun wäre es denkbar, die X_n zu zerlegen in

$$\prod_{j=d^n-l}^{d^n-1} Y_n(j) \ ,$$

wobei $Y_n(j)$ ein Gate der Form $Y_n(j)\,|j\rangle\,|b\rangle = |j\rangle\,|b \oplus \delta_{kj}\rangle$ ist und $k = 0, ..., d^n - 1$. In diesem Fall wären die Kosten im schlimmsten Fall l-mal so groß wie die eines $\Lambda^n(X)$-Gates, was relativ hohe Kosten bedeutet. Aus diesem Grund schlägt [WEL, 2015] eine andere Zerlegung für den Qubit-Fall vor, die im Folgenden verallgemeinert für Qudits präsentiert wird. Für den weiteren Vorgang der Zerlegung müssen zuvor einige Lemmata bewiesen werden.

Lemma 5.2. *Für jedes $p \leq d^n$ gilt $X_n(p, p) = I(n)$.*

Beweis. Trivial. □

Lemma 5.3. *Für $p < q_1 < q_2 \leq d^n$ gilt*

$$X_n(p, q_1)X_n(q_1, q_2) = X_n(p, q_2) \ .$$

Beweis. Trivial. □

Analog zu dem Lemma 4 aus [WEL, 2015], soll nun für beliebiges d bewiesen werden:

Lemma 5.4. *Sei $p = bd^m$, $m \in \mathbb{Z}$, $b < d^{n-m}$ und $q = p + d^m = (b + 1)d^m$. Dann ist $X_n(q, p) = \Lambda^{n[b]}(X)$ und entspricht dabei einem variablen multi-controlled-NOT-Gate mit $n - m$ Kontrollleveln, das auf ein $n + 1$-Qubit-Schema wirkt.*

Beweis. Sei $|j\rangle\,|t\rangle$ im Folgenden der Zustand, auf den das Gate $\Lambda^{n[b]}(X)$ angewendet wird. Definition 5.1 definiert $\Lambda^{n[b]}(X)$ gerade als variablen multi-controlled-NOT-Gate mit $n - m$ Kontrolllevel. Wenn

$$\bigwedge_{k=1}^{n-m} XNOR(b_k, j_k) \stackrel{!}{=} 1$$

gilt, wenn also die ersten $n - m$ Stellen von j und b übereinstimmen, wird aus $|j\rangle\,|t\rangle$ der Zustand $|j\rangle\,|t \oplus 1\rangle$. Ansonsten wirkt die Identität. Es gilt offensichtlich $(p, q) = \Lambda^{n[b]}(X)$, wenn wir beweisen können, dass $bd^m = p \leq j < (b + 1)d^m = q$ genau dazu äquivalent ist, dass die ersten $n - m$ Stellen von j und b übereinstimmen.
Wegen $p = bd^m$ sieht der String von p aus wie ein mit Nullen aufgefüllter String von b. Die Zahl $q - 1$ ist die höchst möglichste Zahl für j, die die Bedingung erfüllt, da $p \leq j < q$ äquivalent ist mit $p \leq j \leq q - 1$. In der String-Notation sehen die Zahlen wie folgt aus:

$$\begin{aligned}
[d^m] &= 1, \underbrace{0, ..., 0}_{\text{m mal}} \\
[b] &= b_1, b_2, ..., b_{n-m} \\
[bd^m] = [p] &= b_1, b_2, ..., b_{n-m}, \underbrace{0, ..., 0}_{\text{m mal}} \\
[p + d^m - 1] = [bd^m + d^m - 1] = [q - 1] &= b_1, ..., b_{n-m}, \underbrace{d - 1, ..., d - 1}_{\text{m mal}}
\end{aligned}$$

Anhand dieser Darstellung sieht man, dass sich für j mit $bd^m = p \leq j < (b + 1)d^m = q$ nur die hinteren m Stellen in der Darstellung zur Basis d ändern. Diese werden allerdings nicht geprüft. Es folgt daraus $\bigwedge_{k=1}^{n-m} XNOR(b_k, j_k) = 1$. Andererseits folgt analog, wenn $bd^m = p \leq j < (b + 1)d^m = q$ nicht erfüllt wird, $\bigwedge_{k=1}^{n-m} XNOR(b_k, j_k) = 0$. □

Mit dem eben verallgemeinerten Lemma kann nun Korollar 5 aus [WEL, 2015] ebenso generalisiert werden. Dazu wird das Hamming-Gewicht so wie im Grundlagenkapitel definiert benutzt, aber statt der üblichen Darstellung zur Basis d $l = \sum_{i=0}^{s} \alpha_i d^{k_i}$ mit $k_s \in \mathbb{Z}$ für $i \in 1, ..., s$, $\alpha_i \in \{1, ..., d-1\}$ und $k_1 < ... < k_n$ eine Darstellung mit Exponenten m_i von d verwendet, wobei die m_i sortiert, jedoch nicht verschieden sein müssen. Aus $3 \cdot d^2 + 2 \cdot d$ mit dem Hamming-Gewicht $h = 3 + 2 = 5$ wird beispielsweise $d^2 + d^2 + d^2 + d + d$. Es ist offensichtlich, dass die Summe h Summanden beinhaltet und eine solche Schreibweise für jedes beliebige j existiert.

Satz 5.5. *Sei h das Hamming-Gewicht von $l < d^n$ und sei $l = \sum_{i=1}^{h} d^{m_i}$ mit $m_r \in \mathbb{Z}$ und $m_1 \leq ... \leq m_h$. Dann kann $X_n(l)$ als Produkt von h „entanglers“ vom Typ $\Lambda^{n[j]}$ geschrieben werden. Die Anzahl der Kontroll-Level ist $m = hn - \sum_{i=1}^{h} m_i$.*

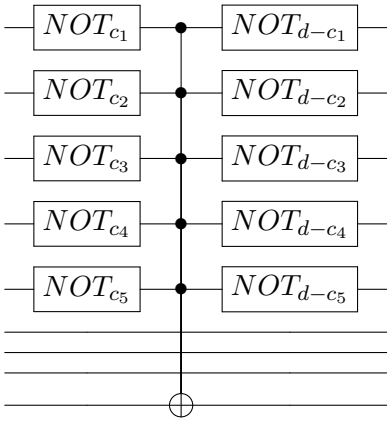
Beweis. Sei $p_r = d^n - \sum_{s=r}^{h} d^{m_s}$ wobei $r = 1, ..., h$. Da p_r durch d^{m_r} teilbar ist, kann man p_r darstellen als $p_r = b_r d^{m_r}$ mit $b_r = d^{n-m_r} - \sum_{s=r}^{h} d^{m_s - m_r}$. Sei nun $p_{h+1} = d^n$. Dies kann man auch schreiben als $p_{h+1} = p_h + d^{m_h} = d^n$ und es gilt $p_1 = d^n - l$. Mit Lemma 5.3 folgt nun

$$X_n(l) = X_n(d^n - l, d^n) = X_n(p_1, p_{h+1}) = \prod_{r=1}^{h} X_n(p_r, p_{r+1}) \ .$$

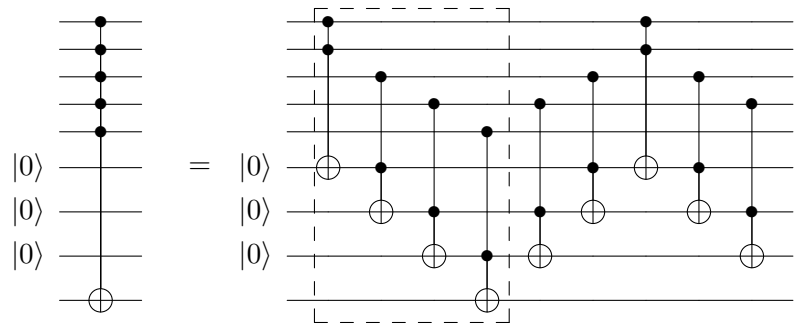
Da man schreiben kann $p_{r+1} = p_r + d^{m_r} = (b_r + 1)d^{m_r}$, erfüllt $X_n(p_r, p_{r+1})$ mit $m = m_r$ Lemma 5.4. Nun ist jedes der h im Produkt vorkommenden $X_n(p_1, p_{n+1})$ vom Typ $\Lambda^{n[b]}$ und hat $n - m_r$ Kontroll-Level. Insgesamt ist demzufolge die Anzahl der Kontroll-Level $m = hn - \sum_{i=1}^{h} m_i$. □

5.3 Zerlegung der multi-controlled-Gates

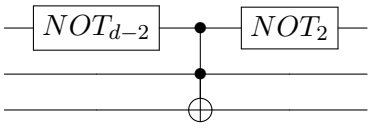
Im letzten Kapitel wurden die „cascaded entangler“ $X_n(l)$ in eine bestimmte Anzahl verschiedener variabler multi-controlled-NOT-Gates zerlegt. Nun soll gezeigt werden, dass diese exakt durch eine Abfolge von Toffoli- und verschiedener NOT-Gates zerlegt werden können. Sei m die Zahl der Kontroll-Level. Die Hilfsqudits sind am Anfang auf 0 initialisiert, werden zur Überprüfung der Datenqudits genutzt und am Ende durch eine entsprechende Gatefolge zurückgesetzt, damit ein Hintereinanderschalten der variablen multi-controlled-NOT-Gates möglich ist. Die Idee der Darstellung entspringt dem Paper [BAR, 1995], in dem die Zerlegung der multi-controlled-NOT-Gates allerdings nur für den Fall $d = 2$ und dementsprechend ohne zusätzliche NOT-Gates erfolgt.
Zunächst kann man ein variabel prüfendes multi-controlled-NOT-Gate mit Hilfe von höchstens $2m$ NOT-Gates : $|a\rangle \rightarrow |a + c\rangle$ und einem multi-controlled-NOT-Gate, das nur auf einen Wert, zum Beispiel 1, prüfen kann ausdrücken. Die Werte für c müssen dem darzustellenden variablen Gate entsprechend gewählt werden.



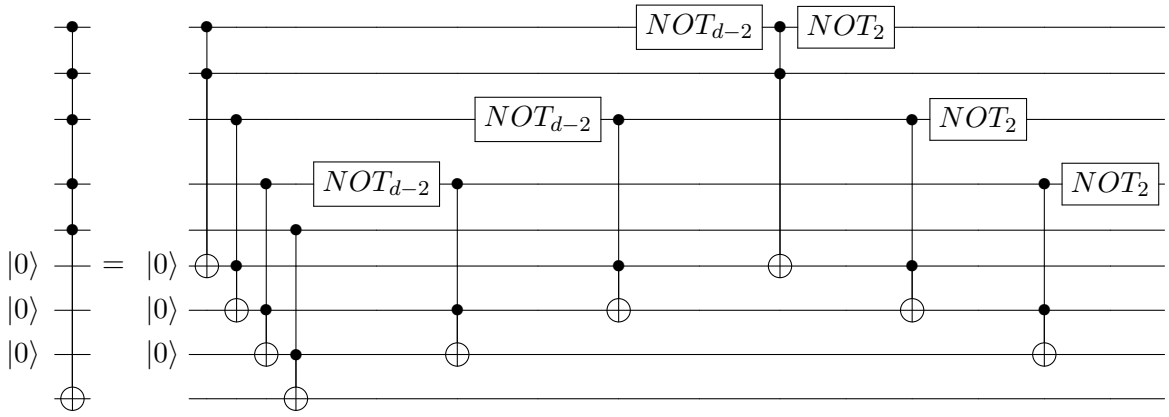
Es bleibt die Aufgabe, die multi-controlled-NOT-Gates zu zerlegen. Dies ist mit einer Folge von $\Lambda^{2[j]}$-Gates möglich. Dabei ist der erste, markierte Teil der Gates zum Kontrollieren da. Die restlichen setzen die Hilfsqudits zurück. Die Abbildung zeigt ein Beispiel für $m = 5$ und $d = 2$, wobei drei Hilfqudits benötigt werden.



Es muss nun beachtet werden, dass nur im Qubit-Fall das bekannte Toffoli-Gate $|a\rangle\,|b\rangle\,|c\rangle \to |a\rangle\,|b\rangle\,|c + ab\rangle$ und ein analog definierbares Toffoli$_-$-Gate $|a\rangle\,|b\rangle\,|c\rangle \to |a\rangle\,|b\rangle\,|c - ab\rangle$ identische Operationen sind. Im Fall $d \leq 2$ kann das Zurücksetzen der Hilfsqudits nicht durch die selben Gates geschehen, die zum Kontrollieren eingesetzt wurden. Allerdings kann das Toffoli$_-$ mit zwei NOT-Gates $|a\rangle \to |a + c\rangle$ mit $c = d-2$ und $c = 2$ und einem Toffoli ausgedrückt werden. Die Abbildung zeigt das Hintereinanderschalten.



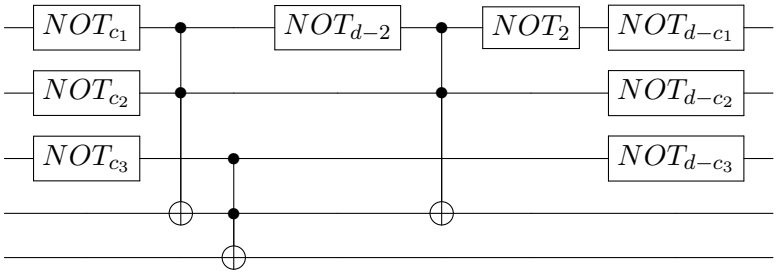
Da NOT_{d-2} und NOT_2 hintereinander geschaltet die Identität ergeben, brauchen wir weniger als zweimal so viele NOT-Gates, wie zurücksetzende Toffoli-Gates. Die daraus resultierende Zerlegung eines multi-controlled-NOT-Gates für Qudits ist in der Abbildung dargestellt.



Für das oben benutzte Beispiel kann man ein multi-controlled-NOT-Gates mit $m = 5$ demzufolge in neun Toffoli- und sechs NOT-Gates zerlegen. Interessant ist nun diese Anzahlen für beliebiges m herzuleiten.

Satz 5.6. *Ein variables multi-controlled NOT-Gate mit m Kontrollleveln kann unter Verwendung von $m-2$ in $|0\rangle$ initialisierten Hilfsqudits in $3m-6$ $Toffoli$- und $4m-4$ NOT-Gates zerlegt werden. Die Hilfsqudits befinden sich anschließend weiterhin im Zustand $|0\rangle$*

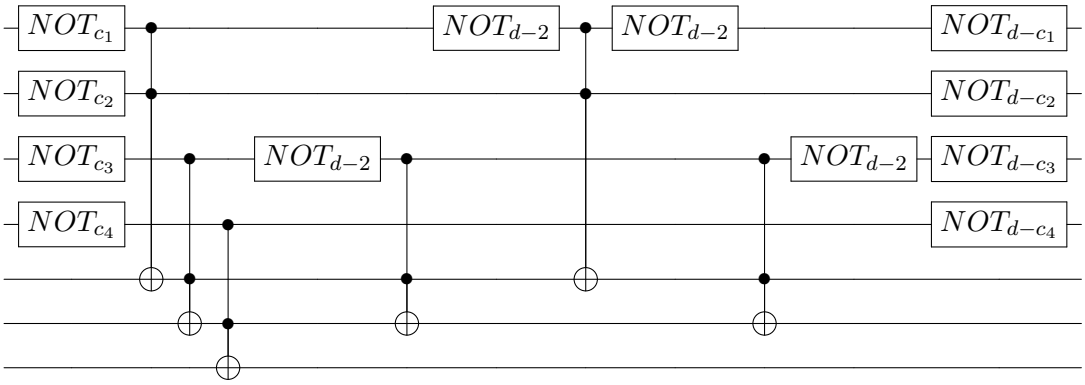
Beweis. Der Beweis folgt mit Induktion. Als Induktionsanfang wird die Anzahl der Kontrolllevel $m = 3$ gesetzt. Wie in der Abbildung zu sehen, werden $3 \cdot 3 - 6 = 3$ Toffoli-Gates und $4 \cdot 3 - 4 = 8$ NOT-Gates benötigt.



Nun folgt der Induktionsschritt $m \to m + 1$. Somit folgt

$$\begin{aligned} \text{Toffoli-Anzahl: } 3m' - 6 &= 3(m+1) - 6 = 3m - 6 + 3 \\ \text{NOT-Anzahl: } 4m' - 4 &= 4(m+1) - 4 = 4m - 4 + 4 \ . \end{aligned}$$

Wächst m um eins, werden drei Toffoli-Gates mehr benötigt. Eines davon wird zu Beginn zum Kontrollieren, zwei werden zum Implementieren der Toffoli$_-$-Gates im hinteren Teil gebraucht. Desweiteren kommen vier NOT-Gates hinzu. Zwei davon ermöglichen, dass das neue Kontrolllevel variabel ist und kommen an den Außenseiten dazu. Die anderen zwei werden für die Toffoli$_-$-Gates benötigt. Für $m = 4$ wird das variable multi-controlled-Gate durch folgende Gate-Folge implementiert:



□

Bei der Anzahl der NOT-Gates handelt es lediglich um die maximal benötigte Anzahl. Wenn das Gate ein oder mehrere Qubit auf 1 prüft, werden weniger als $2m$ NOT-Gates benötigt um aus dem multi-controlled-NOT-Gate ein variables multi-controlled-NOT-Gate zu machen. Zu dem können ist es möglich, dass eines dieser NOT-Gates und eines, das zum Implementieren der Toffoli$_-$-Gates nötig ist, sich zu der Identität ergänzen und wegfallen.

5.4 Schlussfolgerungen

Es ist gelungen einen unitären, diagonalen Operator U in Phasen-Gates $P(\theta)$, Toffoli-Gates und NOT-Gates zu zerlegen. Von letzteren existieren $d-1$ Versionen, die im Extremfall alle gebraucht werden. Wie im Grundlagenkapitel erwähnt, wurden alle für diese Zerlegung benötigten Gates von [AHA, 1999] als elementar bezeichnet. Im Qubit-Fall kann ein Toffoli-Gate exakt in der Clifford+T-Basis dargestellt werden und benötigt daher konstant viele T-Gates [NIE, 2010]. Dieser Aspekt ist allerdings nur für $d = 2$ interessant, da nur in diesem Fall die fehlertolerante Implementierbarkeit der T-Gates bekannt ist. Für allgemeine Qudits liegt der Erfolg in der Zerlegung durch elementare Gates. Ich habe gezeigt, dass auch im allgemeinen

Fall der Zusammenhang zwischen der Anzahl der Kontroll-Level und Anzahl der Toffoli-Gates linear ist. Die Anzahl der Toffoli-Gates in einem „cascaded entangler“ ist deshalb proportional zu dem Produkt aus Anzahl der multi-controlled-NOT-Gates pro „cascaded entanglers“ mal der Anzahl der Kontrolllevel pro multi-controlled-NOT-Gate. Da für jedes V_i zwei „cascaded entanglers“ benötigt werden, ist die auch der Zusammenhang zwischen der Anzahl der Phasen k und der „cascaded entanglers“ linear. Unklar bleibt, wie viele Toffoli-Gates für die k Phasengates benötigt werden, wenn diese weiter zerlegt werden. Da es mir zudem gelungen ist die „cascaded entanglers“ exakt zu zerlegen, wird die gewünschte Genauigkeit der Implementierung lediglich durch das Phasengate beeinflusst.
Im Fall $d = 2$ entsprechen die Phasengates bis auf eine Phase Einzelqubit-Rotationen. Wie in Kapitel 4 erwähnt, zeigt [PAET, 2014] numerisch einen sehr niedrigen T-Count für diese. Man könnte sie also gemäß des Repeat-Until-Success-Algorithmus zerlegen und die weitere Zerlegung der „cascaded entanglers“ nach dem eben vorgestellten Schema vollziehen.

6 Vergleich und Auswertung

Nachdem die ausgewählten Algorithmen behandelt wurden, soll in diesem Kapitel näher auf die im Einleitungskapitel erläuterten Ziele dieser Arbeit eingegangen werden.
Eine Intention war, die drei vorgestellten Verfahren zu vergleichen. Sie alle zerlegen unitäre Quantenoperationen, wobei sich der Solovay-Kitaev-Algorithmus und der Repeat-Until-Success-Algorithmus auf Einzel-Qudit-Operationen beschränken und der Algorithmus von [WEL, 2015] diagonale Operatoren zerlegt. Mit allen drei Algorithmen ist es möglich, die entsprechenden Gates in die Clifford+T-Basis zerlegen. Der Solovay-Kitaev-Algorithmus ist allerdings für ein beliebiges universelles Gate-Set definiert. Ebenfalls auf andere Gate-Sets übertragbar ist nach [PAET, 2014] der Repeat-Until-Success-Algorithmus.
Die Länge der zerlegenden Gatefolge ist allein im Solovay-Kitaev-Algorithmus mit $l_\epsilon \approx O\left(\ln^{3.96} \frac{1}{\epsilon}\right)$ angebbar und hängt von der wählbaren Genauigkeit ab. Offen geblieben ist allerdings die T-Gate-Anzahl. Als Obergrenze kann lediglich die eben thematisierte Länge der Gatefolge genannt werden. Beim Repeat-Until-Success-Algorithmus ist es nicht möglich eine konkrete Länge anzugeben, da das richtige Gate nur mit einer bestimmten Wahrscheinlichkeit implementiert wird. Ein Erfolg kann gleich beim ersten Ausführen oder erst nach oftmaligem Wiederholen und Rücktransformieren auftreten. Dies gilt ebenso für die benötigte T-Gate-Anzahl, die letztendlich für das Ausführen einer Approximation gebraucht wird. Allerdings skaliert der Erwartungswert der T-Gates, die in einem Repeat-Until-Success-Schaltkreis vorhanden sind, mit $2.4 \log_2(\frac{1}{\epsilon})$ in der Genauigkeit. Dies ist eine Verbesserung gegenüber der Obergrenze der T-Gate-Anzahl des Solovay-Kitaev-Algorithmus, auch wenn der Repeat-Until-Success-Schaltkreis möglicherweise einige Male durchlaufen werden muss und zumal die Rücktransformation keine T-Gates benötigt. Bei der Verallgemeinerung des Welch-Algorithmus auf Qubits habe ich festgestelt, dass die Anzahl der Gates, die bei gegebener Kontrolllevel-Anzahl m für ein variables multi-controlled-NOT-Gate benötigt werden, proportional zu der Phasenanzahl k ist. Da die „cascaded entanglers“ exakt zerlegt werden, hängt die T-Gate-Anzahl nicht von der Genauigkeit ab. Für das Phasengate werden allerdings ebenfalls T-Gates benötigt, deren Anzahl sich hingegen mit der geforderten Genauigkeit erhöht.
Die Implementierungszeit in Abhängigkeit von der Genauigkeit skaliert beim Solovay-Kitaev-Algorithmus mit $t_\epsilon \approx O\left(\ln^{2.71} \frac{1}{\epsilon}\right)$. Beim Repeat-Until-Success-Algorithmus ist eine solche Angabe wenig sinnvoll. Hat man die Datenbank erstellt, so ergibt sich die Laufzeit aus dem Durchlaufen einer endlichen Liste. Bei den Axialrotationen kann man die geordnete Liste per binärer Suche durchlaufen. Allerdings ist das Durchlaufen in beiden Fällen unabhängig von der Genauigkeit. Aufgrund der exakten Zerlegung der „cascaded entanglers“ beim Welch-Algorithmus wird die Laufzeit durch den Algorithmus bestimmt, der das Phasengate zerlegt.
In den Vergleich einzubeziehen sind außerdem die unterschiedlichen Voraussetzungen. Für den Repeat-Until-Success-Algorithmus werden im Gegensatz zu den anderen Algorithmen Hilfsqubits und Messungen benötigt. Außerdem muss bei diesem Algorithmus zuvor einmalig eine Datenbank mit Zerlegungen angefertigt werden. Dies ist ebenfalls beim Solovay-Kitaev-Algorithmus für die erste Approximation im Algorithmus notwendig.
Das andere Ziel der vorliegenden Arbeit war es, dass alle Algorithmen auch für Qudit-Operationen funktionieren sollen. Ein Erfolg dieser Arbeit ist das Verallgemeinern des Welch-Algorithmus auf Qudits. Mir ist es gelungen, die „cascaded entanglers“ in h variable multi-controlled-NOT-Gates zu zerlegen und diese wiederum mit Hilfe einer Abfolge von NOT- und Toffoli-Gates auszudrücken. Die Anzahl der jeweils benötigten Gates ist linear mit der

Zahl der Kontroll-Level. Es ist exakt angebbar, wie viele Toffoli-Gates benötigt werden. Für die NOT-Gates ist der maximal benötigte Wert berechenbar.
Bei der Thematisierung des Repeat-Until-Success-Algorithmus ist klar geworden, dass sich auch dessen Verfahren auf Qudits verallgemeinern lässt. Allein unklar ist geblieben, wie umfangreich und nützlich die benötigte Datenbank im Qudit-Fall ist. Leider ist die mir zur Verfügung gestellte Computerleistung nicht groß genug, um eine solche Datenbank zu erstellen. Insofern lässt sich hoffen, dass in Zukunft weiter untersucht wird wie man Qudit-Operationen zerlegen kann, um die Fehlertoleranz und Geschwindigkeit von Quantencomputern zu verbessern.

Literaturverzeichnis

[DAW, 2008] Christopher M. Dawson und Michael A. Nielson: The Solovay-Kitaev Algorithm, Februar 2008. http://arxiv.org/abs/quant-ph/0505030.

[PAET, 2014] Adam Paetznick und David R. Cheriton: Repeat-Until-Success: Non-deterministic decomposition of single-qubit unitaries, Oktober 2014. http://arxiv.org/abs/1311.1074.

[WEL, 2015] Jonathan Welch, Alex Bocharov und Krysta M. Svore: Efficient Approximation of Diagonal Unitaries over the Clifford+T Basis, April 2015. http://arxiv.org/abs/1412.5608v3.

[NIE, 2010] Nielsen, Michael A. und Isaac L. Chuang: Quantum computation and quantum information. Cambridge University Press, Cambridge, 10. Auflage, 2010.

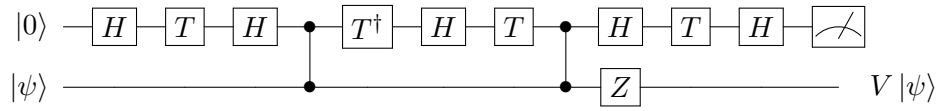
[AHA, 1999] Dorit Aharonov und Michael Ben-Or: Fault-tolerant quantum computation with constant error rate, Juni 1999. http://arxiv.org/abs/quant-ph/9906129.

[BAR, 1995] Adriano Barenco, Charles H. Bennett, Richard Cleve und weitere: Elementary gates for quantum computation, März 1995. http://arxiv.org/abs/quant-ph/9503016v1.

[KLIU, 2013] Vadym Kliuchnikov, Dmitri Maslov und Michele Mosca: Fast and efficient exact synthesis of single qubit unitaries generated by Clifford and T gates, Februar 2013. http://arxiv.org/abs/1206.5236.

Anhang

In Kapitel 4.1 wurde ein Beispiel für Repeat-Until-Success-Quantenschaltkreis vorgestellt. Der Schaltenkreis



bewirkt mit einer Wahrscheinlichkeit von $\frac{5}{8}$, dass die Operation

$$V_{Erfolg} = \frac{3+i}{\sqrt{10}} |0\rangle\langle 0| - \frac{1+3i}{\sqrt{10}} |1\rangle\langle 1|$$

auf ψ wirkt. In dem anderen Fall wirkt

$$V_{Misserfolg} = \frac{((-1+i)+(-1)^{1/4})}{\sqrt{3}}(|0\rangle\langle 0| + |1\rangle\langle 1|) \ .$$

Die nötigen Rechnungen werden im Folgenden durchgeführt:

$$\begin{aligned}
V =& H\otimes\mathbb{1}\cdot T\otimes\mathbb{1}\cdot H\otimes\mathbb{1}\cdot\mathbb{1}\otimes Z\cdot(|0\rangle\langle 0|\otimes\mathbb{1}+|1\rangle\langle 1|Z)T\otimes\mathbb{1}\cdot H\otimes\mathbb{1}\cdot T^\dagger\otimes\mathbb{1}\\
&\cdot(|0\rangle\langle 0|\otimes\mathbb{1}+|1\rangle\langle 1|Z)H\otimes\mathbb{1}\cdot T\otimes\mathbb{1}\cdot H\otimes\mathbb{1}\\
=&\frac{1}{\sqrt{2}}(|00\rangle\langle 00|+|10\rangle\langle 00|+|00\rangle\langle 10|-|10\rangle\langle 10|+|01\rangle\langle 01|+|11\rangle\langle 01|\,|01\rangle\langle 11|-|11\rangle\langle 11|)\\
&\cdot(|00\rangle\langle 00|+\exp^{\frac{i\pi}{4}}|10\rangle\langle 10|+|01\rangle\langle 01|+\exp^{\frac{i\pi}{4}}|11\rangle\langle 1|)\\
&\cdot\frac{1}{\sqrt{2}}(|00\rangle\langle 00|+|10\rangle\langle 00|+|00\rangle\langle 10|-|10\rangle\langle 10|+|01\rangle\langle 01|+|11\rangle\langle 01|\,|01\rangle\langle 11|-|11\rangle\langle 11|)\\
&\cdot(|00\rangle\langle 00|+|10\rangle\langle 10|-|01\rangle\langle 01|-|11\rangle\langle 11|)\\
&\cdot(|00\rangle\langle 00|+|01\rangle\langle 01|+|10\rangle\langle 10|-|11\rangle\langle 11|)\\
&\cdot(|00\rangle\langle 00|+\exp^{\frac{i\pi}{4}}|10\rangle\langle 10|+|01\rangle\langle 01|+\exp^{\frac{i\pi}{4}}|11\rangle\langle 11|)\\
&\cdot\frac{1}{\sqrt{2}}(|00\rangle\langle 00|+|10\rangle\langle 00|+|00\rangle\langle 10|-|10\rangle\langle 10|+|01\rangle\langle 01|+|11\rangle\langle 01|\,|01\rangle\langle 11|-|11\rangle\langle 11|)\\
&\cdot(|00\rangle\langle 00|+\exp^{\frac{-i\pi}{4}}|10\rangle\langle 10|+|01\rangle\langle 01|+\exp^{\frac{-i\pi}{4}}|11\rangle\langle 11|)\\
&\cdot(|00\rangle\langle 00|+|01\rangle\langle 01|+|10\rangle\langle 10|-|11\rangle\langle 11|)\\
&\cdot\frac{1}{\sqrt{2}}(|00\rangle\langle 00|+|10\rangle\langle 00|+|00\rangle\langle 10|-|10\rangle\langle 10|+|01\rangle\langle 01|+|11\rangle\langle 01|\,|01\rangle\langle 11|-|11\rangle\langle 11|)\\
&\cdot(|00\rangle\langle 00|+\exp^{\frac{i\pi}{4}}|10\rangle\langle 10|+|01\rangle\langle 01|+\exp^{\frac{i\pi}{4}}|11\rangle\langle 11|)\\
&\cdot\frac{1}{\sqrt{2}}(|00\rangle\langle 00|+|10\rangle\langle 00|+|00\rangle\langle 10|-|10\rangle\langle 10|+|01\rangle\langle 01|+|11\rangle\langle 01|\,|01\rangle\langle 11|-|11\rangle\langle 11|)\\
=&\frac{1}{\sqrt{2}}\begin{pmatrix}1&0&1&0\\0&1&0&1\\1&0&-1&0\\0&1&0&-1\end{pmatrix}\cdot\begin{pmatrix}1&0&0&0\\0&1&0&1\\0&0&\exp^{\frac{i\pi}{4}}&0\\0&0&0&\exp^{\frac{i\pi}{4}}\end{pmatrix}\cdot\frac{1}{\sqrt{2}}\begin{pmatrix}1&0&1&0\\0&1&0&1\\1&0&-1&0\\0&1&0&-1\end{pmatrix}
\end{aligned}$$

$$
\begin{aligned}
&\cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & \exp^{\frac{i\pi}{4}} & 0 \\ 0 & 0 & 0 & \exp^{\frac{i\pi}{4}} \end{pmatrix} \\
&\cdot \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & \exp^{\frac{-i\pi}{4}} & 0 \\ 0 & 0 & 0 & \exp^{\frac{-i\pi}{4}} \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \\
&\cdot \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & \exp^{\frac{i\pi}{4}} & 0 \\ 0 & 0 & 0 & \exp^{\frac{i\pi}{4}} \end{pmatrix} \cdot \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} \\
= &\begin{pmatrix} \frac{3}{4} + \frac{i}{4} & 0 & \frac{(1-i)+(-1)^{1/4}}{2\sqrt{2}} & 0 \\ 0 & \frac{-1}{4} - \frac{3i}{4} & 0 & \frac{(1-i)+(-1)^{1/4}}{2\sqrt{2}} \\ \frac{(-1+i)+(-1)^{1/4}}{2\sqrt{2}} & 0 & \frac{-1}{4} - \frac{3i}{4} & 0 \\ 0 & \frac{(-1+i)+(-1)^{1/4}}{2\sqrt{2}} & 0 & \frac{3}{4} + \frac{i}{4} \end{pmatrix} \\
= &\left(\frac{3}{4} + \frac{i}{4}\right) |00\rangle \langle 00| + \frac{(1-i)+(-1)^{1/4}}{2\sqrt{2}} |00\rangle \langle 10| + \left(\frac{-1}{4} - \frac{3i}{4}\right) |01\rangle \langle 01| \\
&+ \frac{(1-i)+(-1)^{1/4}}{2\sqrt{2}} |01\rangle \langle 11| + \frac{(-1+i)+(-1)^{1/4}}{2\sqrt{2}} |10\rangle \langle 00| + \left(\frac{-1}{4} - \frac{3i}{4}\right) |10\rangle \langle 10| + \\
&\frac{(-1+i)+(-1)^{1/4}}{2\sqrt{2}} |11\rangle \langle 01| + \left(\frac{3}{4} + \frac{i}{4}\right) |11\rangle \langle 11|
\end{aligned}
$$

Nun können die Wahrscheinlichkeiten berechnet werden.

$$
\begin{aligned}
out =& V(a|00\rangle + b|01\rangle) \\
=& \left(\frac{3}{4} + \frac{i}{4}\right) a|00\rangle + \left(\frac{-1}{4} - \frac{3i}{4}\right) b|01\rangle + \frac{(-1+i)+(-1)^{1/4}}{2\sqrt{2}} a|10\rangle \\
&+ \frac{(-1+i)+(-1)^{1/4}}{2\sqrt{2}} b|11\rangle \\
Prob =& out^{\dagger} \cdot (|00\rangle \langle 00| + |01\rangle \langle 01|) \cdot out \\
=& \left(\left(\frac{3}{4} + \frac{i}{4}\right) a\langle 00| + \left(\frac{-1}{4} - \frac{3i}{4}\right) b\langle 01| + \frac{(-1+i)+(-1)^{1/4}}{2\sqrt{2}} a\langle 10| \right. \\
&\left. + \frac{(-1+i)+(-1)^{1/4}}{2\sqrt{2}} b\langle 11|\right) \\
&\cdot (|00\rangle \langle 00| + |01\rangle \langle 01|) \\
&\cdot \left(\left(\frac{3}{4} + \frac{i}{4}\right) a|00\rangle + \left(\frac{-1}{4} - \frac{3i}{4}\right) b|01\rangle + \frac{(-1+i)+(-1)^{1/4}}{2\sqrt{2}} a|10\rangle + \right. \\
&\left. \frac{(-1+i)+(-1)^{1/4}}{2\sqrt{2}} b|11\rangle\right) \\
=& \left(\frac{3}{4} - \frac{i}{4}\right)\left(\frac{3}{4} + \frac{i}{4}\right) a^2 + \left(\frac{-1}{4} + \frac{3i}{4}\right)\left(\frac{-1}{4} - \frac{3i}{4}\right) b^2
\end{aligned}
$$

$$
\begin{aligned}
&= \left(\frac{9}{16} + \frac{1}{16}\right) a^2 + \left(\frac{1}{16} + \frac{9}{16}\right) b^2 \text{ mit } a^2 + b^2 = 1 \\
&= \frac{5}{8} \\
out_{Norm} &= \frac{1}{\sqrt{\frac{5}{8}}} \cdot (|00\rangle \langle 00| + |01\rangle \langle 01|) \cdot out \\
&= \frac{(3+i)a}{\sqrt{10}} |00\rangle - \frac{(1+3i)b}{\sqrt{10}} |01\rangle \\
Prob2 &= out^{\dagger} \cdot (|10\rangle \langle 10| + |11\rangle \langle 11|) \cdot out \\
&= \frac{3}{8} \\
out_{Norm2} &= \frac{1}{\sqrt{\frac{3}{8}}} (|10\rangle \langle 10| + |11\rangle \langle 11|) \cdot out \\
&= \frac{(-1+i)(-1)^{1/4}}{\sqrt{3}} a |10\rangle + \frac{(-1+i)(-1)^{1/4}}{\sqrt{3}} b |11\rangle
\end{aligned}
$$

Aus out_{Norm} und out_{Norm2} folgen

$$
\begin{aligned}
V_{Erfolg} &= \frac{3+i}{\sqrt{10}} |0\rangle \langle 0| - \frac{1+3i}{\sqrt{10}} |1\rangle \langle 1| \\
V_{Misserfolg} &= \frac{((-1+i) + (-1)^{1/4})}{\sqrt{3}} (|0\rangle \langle 0| + |1\rangle \langle 1|) \ .
\end{aligned}
$$