Quantum Computing

Table of Contents

This section is treating several issues of the classical theory of computation. Two classical computational models will be introduced: the Turing machine and the Circuit model. Further some details about the resource question will be presented. Here one will encounter several complexity classes as a method of classification of problems. Afterwards the energy dissipation of a computational problem will be explained and one will learn how it is possible to perform a task without energy loss. With the introduction of the Toffoli gate one also will get to know a universal, reversible gate which later becomes important for the quantum computation.

The theory is based on the textbooks [1] and [2].

1.1 Models of computation

What is a computational model? It's a set of allowable operations which can be used in order to compute an algorithm. One can easily imagine that different models require different resources, such as execution time of memory space. Thats why its important to know which model is used in the computation and which resources it requires. In this first chapter two models of computation will be introduced: the Turing machine and the Circuit model.

1.1.1 Turing machine

In this first section we describe an extremely basic computational device – the Turing machine. Despite its simplicity, it can be used to simulate the logics of any computer! This device was described in 1936 by Alan Turing, who is often considered to be the father of modern computer science. The Turing machine was never meant to be built in practice, but was designed as an abstract device which should provide answers on certain thought experiments about the limits of computation.

The concept of a Turing machine is based on the idea of a person executing a well-defined procedure by changing the contents on an unlimited tape. In Fig. (1.1) one can see the elements of such a machine:

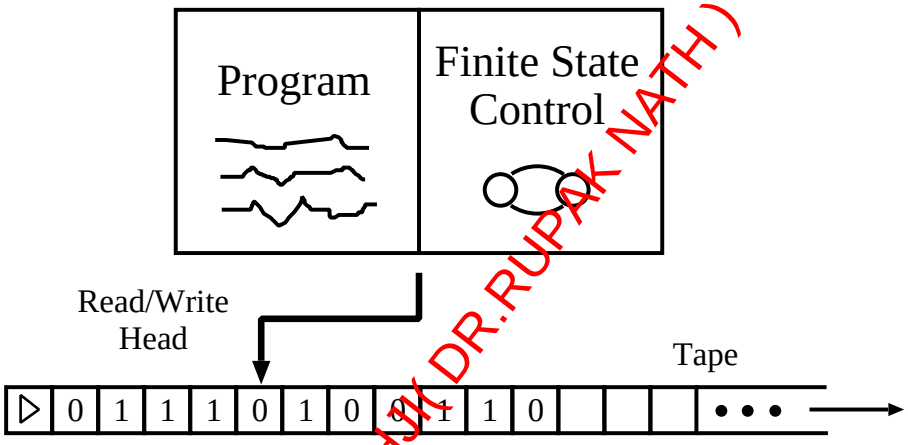


Figure 1.1: Turing machine

There are four elements representing the machine:

- Tape
- Read-write tape-head
- State control
- Program

The **tape** represents the computer memory. It's an one-dimensional object divided into squares, which are numbered (0, 1, 2, ...) and each contain one symbol from some alphabet. In the figure above the alphabet is chosen to consist of four symbols: 0, 1, b = blank, Δ = left edge of the tape.

The **read-write head** is used as a pointer. It identifies a single square on the tape and then reads its content or writes new input inside.

The **state control** is used for coordination of the processes. It consists of a set of internal states denoted by q_1, ..., q_m and two special states q_s (starting) and q_h (halting). Starting in q_s, the internal states change during the program, until it terminates and the halting state q_h is set. The way internal states are used in the program has the following structure: "if your internal state is 5 and the symbol you see on the tape is a 0, then replace is with a 1, move one symbol to the right and assume state 6 as your new state".

The **program** consists of table of instruction of the form "if your state is ... and on the tape you read ..., then do ..." . These instructions are represented as ordered list of program lines of the form <q, x, q, x, s>, where q is an internal state from the set of internal states and x a symbol of the alphabet. The way of working is then to look through this list and search for the line <q, x, ... > and, if found, execute it. If no such line is in the program list, then the procedure is terminated and internal state changes to q_h. The content on the tape is then the output of the algorithm.

Example:

One starts with the binary number x on the tape followed by blanks:

$$x = \triangle 01110010bbbb$$

The machine has a starting and a halting state (q_s and q_h) and three additional internal states: q_1, q_2, q_3. The program consists of seven lines shown in Table (1.1). As one can see following the seps in the table, the last output on the tape is just a 1, i.e. the function determined by this algorithm is $f(x) = 1$, where x is the binary numer on the input.
There are a lot of Turing machines differing by the number of tapes, possible operations, etc. An ordinary (**deterministic**) Turing machine has only one possibility for each possible configuration of the tape symbol and an internal state. A **non-deterministic** Turing machine differs in that the state and tape symbol no longer uniquely specify the action. Now many different possibilities can be applied for the same combination of state and symbol. For example, if we have a 1 on the tape and state 3, we might write a 0 on the tape and switch the internal state to 4 or leave the 1 and switch the state to 5. The way of choosing

Table 1.1: Program lines

Program lines	What does it do?
1: $< q_s, \Delta, q_1, \Delta, +1 >$	tape-head moves right without changing tape-content, internal state changed to q_1
2: $< q_1, 0, q_1, b, +1 >$	while internal state is q_1 and tape-content is 0 tape-head moves right and overwrites tape-content to blank
3: $< q_1, 1, q_1, b, +1 >$	while internal state is q_1 and tape-content is 1 tape-head moves right and overwrites tape-content to blank
4: $< q_1, b, q_2, b, -1 >$	moves left, internal state changes to q_2
5: $< q_2, b, q_2, b, -1 >$	moves left while blanks are on the tape -> returns to starting point
6: $< q_2, \Delta, q_3, \Delta, +1 >$	internal state is changed to q_3, tape-head moves one to the right
7: $< q_3, b, q_h, 1, 0 >$	print out 1

the action is to pick the transition which eventually leads to the termination of the algorithm, i.e. the machine branches into many copies, each of which follows one of the possible transitions, where a deterministic Turing machine only has one computation path. Another type of machines is the **probabilistic** Turing machine which is a non-deterministic machine which now randomly chooses between the available transitions according to some probability distribution. As a consequence, a probabilistic Turing machine can have stochastic results.

What classes of functions can be computed with a Turing machine?

In the example above a very simple function $f(x) = 1$ was presented, but is it possible to implement more challenging algorithms on the Turing machine? The answer is yes. There is an enormous variety of functions which can be accomplished on this simple device. Examples would be all basic arithmetical operations or searching through text represented as strings of bits on the tape. It turns even out that one can simulate all operations which can be performed on a modern computer. And that's the statement of the Church-Turing thesis:

The class of functions computable by a Turing machine corresponds exactly to the class of functions which we would naturally regard as being computable by an algorithm.

Where by an algorithm we mean a process which terminates. It isn't obvious that every function which we would intuitively regard as being computable in fact can be computed by the Turing machine. However, in sixty years no evidence for the contrary has been found. The Church-Turing thesis is also valid for quantum computers, i.e. a quantum computer can compute the same class of functions as is computable on a Turing machine. The difference is only in the efficiency. So there might be algorithms on the quantum computer which require less resources as they would on Turing machine.

1.1.2 Circuits

There is another model of computation which is equivalent to the Turing machine in terms of computational power - the Circuit model. This model is more convenient and realistic because it's build of wires (which carry the information around) and gates (which perform simple computation tasks). The simplest gate would be the NOT gate, flipping a 0 to a 1 and vice versa.

As one can see from this example a logical gate represents a function $f : 0,1^k \rightarrow 0,1^l$ and wires represent the movement of the bit through space or time. A circuit may involve many input and output bits, many wires and many logical gates. Some of the elements of a circuit are shown in Fig. (1.2).
There are also other elements possible. For example a FANOUT gate which would just replace a bit with a two copies of itself, or the CROSSOVER gate which would interchange the values of two bits. Using only few gates (for example AND, OR and NOT) one can construct any circuit and so compute any function!

Example: addition of two n-bit integer

Image one would like to add the two numbers 7 and 13. In binary representation $7 = 111$ and $13 = 1101$. First one would add modulo 2 the numbers on the last position: $1+1(\text{mod}2) = 0$, but there is a 1 which is carried forward to the position one before last. Then the numbers on this position are added to the carrying bit, and so on. That's exactly the way one implements the algorithm in the circuit model. First one defines a half-adder as shown in Fig. (1.3), which just takes two bits, x and y, and outputs the sum x + y (mod2) and the carry bit c (c = 1 if x and y are 1, c = 0 else). The half-adder is used to perform the operation one does on the last position. Then two half-adders build a full-adder (Fig. (1.4)), which is meant to accomplish the operation on the position one before the last.

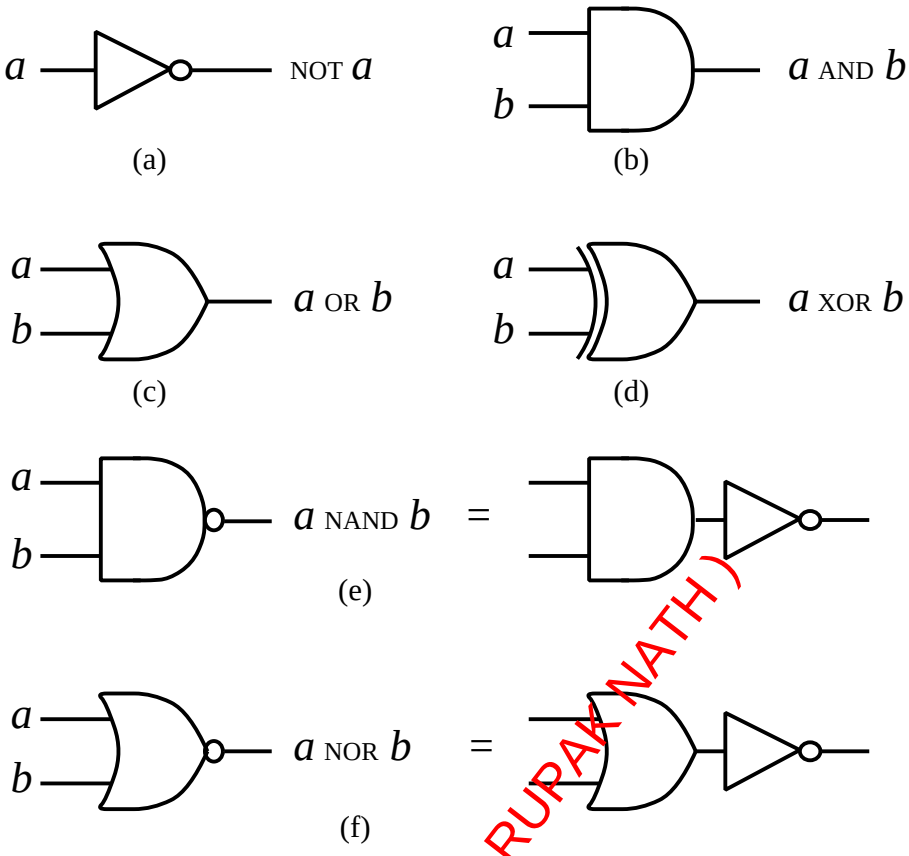


Figure 1.2: Examples of elements of a circuit

Many full-adders together allow to compute the whole addition (Fig. (1.5)).

1.1.3 Implementation of a circuit on a quantum computer?

A quantum circuit model can be defined in an analogous manner, but there are several challenges which arise during the extension from the classical to the quantum case. For example there is the question if one can construct wires which would preserve quantum bits, i.e. quantum bits should not interact with the wires! Another problem is the implementation of the FANOUT gate, which would create a copy of the quantum bit. According to the no cloning theorem [3] it's not possible to create an identical copy of an arbitrary unknown quantum state without changing the initial state. A further point is the irreversibility of the AND or XOR gates. In quantum circuit the gates will be described by unitary matrices which definitely are invertible! Thus an AND or a XOR gate have to be implemented in an other way.

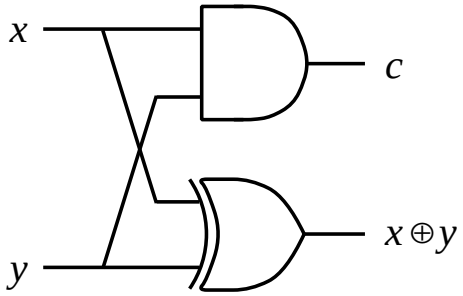


Figure 1.3: Half-adder

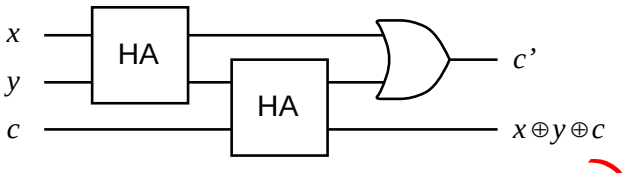


Figure 1.4: Full-adder

1.2 Analysis of computational problems

In the computational science there are three main questions which arise during the discussion of a computational problem:

1. **What is the computational problem?**
   Here one can find the whole range of problems starting with multiplying two numbers together and ending with programming a computer to exceed human abilities in writing of poetry. Later one becomes acquainted with a special class of problems - the decision problem.

2. **How can one design an algorithm to solve this problem?**
   The answer on this question is treated in many books (example: [4]) which

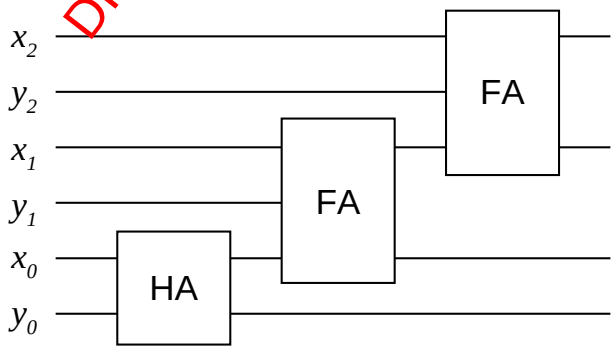


Figure 1.5: Addition of two three-bit integers

cover several general techniques, but also treat the question of how one might be sure that the algorithm behaves as it's claimed.

3. **What are the minimal resources required?**
   This question is considering the recourses time, space and energy. So one would like to find out for example how many computational steps must be executed or how much space is used in order to find the solution. A further interesting issue would be the possibility of classification of the problems according to the resources required.

1.2.1 How to quantify computational resources?

As already mentioned, the possibility of classification problems according to their resource requirements is one of the main concerns of the computational science. There is a challenge which lies in the differentiation of the computational models leading to different resource requirements. Using for example a two-tape machine you may need less steps to perform the same task as with a single-tape machine. In order to quantify resource requirements independently of changes in computational model one uses the idea of summarization of the essential behavior of the function using the asymptotic notation.

Example:

Suppose one needs $24n + 2\log(n) + 16$ gates to perform an addition of two numbers. In limit for large n only the $24n$-term matters, so the essential behavior would be number of operation required scales like n, where n is the number of bits in the numbers being added.

Asymptotic notation:

- $O(f(n))$: upper bound (useful for studying the worst-case behavior)
- $\Omega(f(n))$: lower bound
- $\Theta(f(n))$: asymptotic behavior

Examples:

- $2n$ is $O(n^2)$
- 2^n is $\Omega(n^3)$

- $7n^2 + \sqrt{n}\log(n)$ is $\Theta(n^2)$

Example:

Suppose one would like to sort an n element list of names into alphabetical order with the "compare-and-swap" procedure. The number of operations required would be $(n-1) + (n-2) + (n-3) + \cdots + 1 = \frac{n(n-1)}{2}$, i.e. the asymptotic behavior would be $\Theta(n^2)$.

1.3 Computational complexity

In the computational science a problem is regarded as being efficiently solvable if there is a algorithm, which consumes little space when running. There might be problems which are impossible to solve, not because one can't find an algorithm, but because all known algorithms consume too much space or time, so they become useless. That's the reason why the computational complexity - the study of time and space resources - is such an important issue.

There is a chief distinction which can be made in order to classify problems according to their resources requirements:

- Problems which can be solved using polynomial resources
- Problems which require resources growing faster than polynomial, i.e. "exponential"

According to this distinction one denotes a problem as being easy/tractable if its algorithm uses only polynomial resources and hard/intractable if the best possible algorithm requires exponential resources. An example of an easy problem would be the addition of two binary numbers and a hard one would be the "factoring into prime factors" problem.

One might ask oneself how the different models of computation affect the resource requirements. The solution to this problem gives the Strong Church-Turing thesis:
Any model of computation can be simulated on a probabilistic Turing machine with at most a polynomial increase in the number of elementary operations require

Researchers found that if it was possible to compute a function using k elementary operation in some model that was not the Turing machine, then it was always

possible to compute the same function in the Turing model using at most p(k) elementary operations (where p(k) is a polynomial in k). The Strong Church-Turing thesis is great news for theory of computational complexity. It implies that one may consider only problems on Turing machine. If there is no polynomial solution on it, there won't be one on any other computing device.

The important implication for quantum computation is that its computational power can be related to some major open problems in classical computational complexity theory.

1.3.1 Decision problems and the complexity classes

There are a lot of ways of formulating the problems - one of them is decision problem formulation. A decision problem is a problem with a yes or no answer: "is a given number prime?". Also here it's again important how much time and space an algorithm consumes, so one can make the following classification: a problem is in $TIME(f(n))$ if there exists a Turing machine which allows to an algorithm to decide the problem in time $O(f(n))$. Using this definition one can now summarize together similar problems and build complexity classes.

The complexity class P is the collection of decision problems which are in $TIME(n^k)$, i.e. can be solved by a deterministic Turing machine using a polynomial amount of time. P is often viewed as the class of computational problems which are "efficiently solvable", but there exist problems in P which are intractable in practical terms, for example a problem requiring $n^{10000000}$ operation. An example of an "tractable" problem in P would be the calculation of the greatest common divisor.

There is a further computational class which is considered as one of the most fundamental - the NP class. The abbreviation NP refers to "Non-deterministic Polynomial time", i.e. these are problems for which the "yes" answer has simple proof of the fact that the answer is indeed "yes". More precisely, these proofs can be verified in polynomial time by a deterministic Turing machine. A problem which is in NP would be the following: imagine one is given some integers, such as $-7, -3, -2, 5, 8$ and one would like to know whether some of these integers sum up to zero or not. In this example the answer is "yes" and can be proven by summation $(-3) + (-2) + 5 = 0$.

It's clear that the complexity class P is contained in NP and the most important open question in complexity theory is the P = NP problem, asking whether the

both are equal or not. In the NP class there is a subclass called NP-complete which covers many important problems of NP for which no polynomial-time algorithms are known. If a polynomial solution for these problems could be found, the P = NP problem would be solved, but for the moment the question is still open.

Another important class is BPP - the class of decision problems solvable by a probabilistic Turing machine in polynomial time with an error probability of at most 1/3. BPP refers to Bounded-error, Probabilistic, Polynomial time. If a problem is in BPP, then there is an algorithm for it that is allowed to make random decisions. On any given run of the algorithm, it has a probability of at most 1/3 of giving the wrong answer. The idea is that there is a probability of error, but if the algorithm is run many times, the chance that the majority of the runs are wrong drops off exponentially. BPP is one of the largest practical classes of problems, i.e. most problems of interest in BPP have efficient probabilistic algorithms that can be run quickly. One important example of a problem in BPP is the polynomial identity testing: the problem of determining whether a polynomial is identically equal to the zero polynomial.

1.3.2 Energy and computation

Besides space and time resources there is another important component of computations: the energy resource. Surprisingly it turns out, that computations (classical and quantum) can be done without expending any energy! This argument is based on the link between energy consumption and reversibility of a process. For example a NOT gate is reversible, i.e. knowing the output one always is able to construct back the input. Otherwise a NAND gate is not reversible: if the output is 1, the input could have been 00, 01 or 10. The irreversibility though can be understood in terms of information erasure. A reversible computation corresponds to no information being erased, i.e. the input always can be recovered from the output.

There is an other important issue: the Landauer's principle, saying that the erasure of a single bit of information is associated with an energy dissipation of at least $k_B \ln 2$. This value denotes only the lower boundary of energy dissipation. Existing computers of the year 2000 dissipate for example $500 k_B \ln 2$ energy for each elementary logical operation. So they are not very close to the lower bound.

Would it be possible to perform a universal computation without information

erasure? The answer is yes. There are different reversible circuit-based models performing with reversible logical gates. One of them - the Toffoli gate - is presented below. Later it will be of great use for the quantum computer discussion.

Toffoli gate

The Toffoli gate is universal for classical computations and very useful for quantum computation. "Universal" means that any other gate can be implemented using only this gate. The Toffoli gate is build of three inputs a, b, c and gives three outputs a', b', c'. The first two bits a and b are control bits, i.e. they are not changed during the computation. The third gate c is the target bit, that is flipped if $a = b = 1$ (figure 1.6).

Inputs			Outputs		
a	*b*	*c*	*a'*	*b'*	*c'*
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

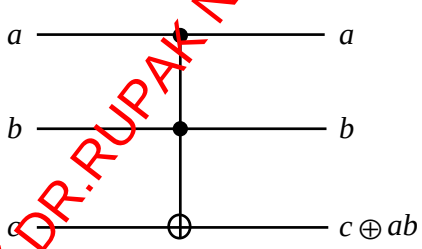


Figure 1.6: Toffoli gate

An example of implementation of a common gate using the Toffoli gate would be the following: imagine one would like to describe a NAND gate using Toffoli gate. The way to do it would be to set the the target bit equal to one: $c = 1$. Then the table one would get would be the following table 1.2.
In this implementation one sees that the target bit c stays the same, unless for the last state $a = b = 1$. Then it's flipped to zero. Looking at the output of c' one can easily see that this table exactly corresponds to the output values of a NAND gate.

Table 1.2: Implementation of a NAND on the Toffoli gate

a	b	c	a'	b'	c'
0	0	1	0	0	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

1.4 Summary

In this chapter two models of classical computation were introduced. The Turing machine and the Circuit model. We have seen that Turing machine can be easily implemented using four elements: a tape, a read-write head, a state control and a program. This simple model represents a very powerful device and the Church-Turing thesis says that every computable function can be implemented on the Turing machine.

Further the second computational model was introduced: The Circuit model which consists of wires and gates. Using very simple operations like NOT, OR and AND one can easily construct quite difficult circuits. The advantage of this model is that it is more convenient and useful in praxis. This model also can be extended to the quantum case, but there are several challenges to solve, like the non-cloning theorem of quantum mechanics which says that no quantum bit can be cloned without destroying the original one.

Another important topic was the analysis of a computational problem, especially its computational resources like space and time. Here we have built groups of problems using the asymptotic notation and then introduced a special notation for classes of decision problems (with a "yes" or "no" answer). So the class P was defined as decision problem which can be solved by a deterministic Turing machine using a polynomial amount of time. There was also the class NP ("Non-deterministic Polynomial time") defined as problems which has a simple proof of the fact that the answer is indeed "yes".

The last issue was the energy resource question. Here one has learned that a computational problem in general could perform without energy dissipation, if the implementation would consist of reversible gates, i.e. no information would get lost. Such a gate is the Toffoli gate which also becomes important for the quantum case.

Bibliography

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2000).

[2] M. Sipser, *Introduction to the Theory of Computation* (Course Technology, 2005).

[3] W. Wootters and W. Zurek, *A Single Quantum Cannot be Cloned*, Nature **299**, 9 (1982).

[4] S. Dasgupta, C. H. Papadimitriou, and U. Vazirani, *Algorithms* (McGraw-Hill Science, 2006).

Chapter 2

Factorization and applications

Mauro Calderara
supervisor: Ingo Kirsch

In the context of quantum computing, one of the goals is the efficient factorization of numbers. This particular operation - or rather: the unavailability thereof for current, Turing based computers - is a requirement for the most widely deployed cryptographic algorithms, to be secure. Given the progress recently made in the field of quantum computing, in particular the theoretical possibility of efficient algorithms for factorization and the discrete logarithm-problem due to Shor's algorithm, the landscape of cryptography could therefore change significantly. In this text we will to outline the connection between currently deployed cryptographic algorithms and advances in quantum computing.

This chapter is organized as follows: We shall start with an introduction to symmetric and asymmetric cryptographic algorithms in more general terms followed by an introduction to public-key cryptosystems. As a case study of a public-key cryptosystem, we will briefly explain the RSA algorithm as proposed by Rivest, Shamir and Adleman [1]. We will continue with the so called RSA-problem and its relation to the factorization problem. In the concluding notes, links to other algorithms and the concepts of post-quantum cryptography will be provided as well.

2.1 Introduction and Motivation

2.1.1 Cryptography before 1976

Cryptographic algorithms have been known and used to communicate over insecure channels throughout history, ranging from simpler algorithms such as ROT13 allegedly used by Julius Caesar [2] over more sophisticated ones like ENIGMA [2] used by the German army during world war II to the nowadays widely deployed DES [3]. Yet, all of the cited share one major drawback: in order to communicate, both sender and receiver must share a secret commonly referred to as the *key*. The fact that both ends are required to share the same key led to the term "symmetric cryptography". Establishing the "symmetry" of both sides sharing the same key, can be achieved in many ways. There are many possible solutions to this problem: One can tell the key to the recipient in person, have the key embedded in a device in the possession of the receiver or simply exchange it over a secure telephone line. But obviously none of these procedures scales very well for bigger or anonymous systems such as today's internet. This so called "key distribution problem" is an issue for many practical applications that require confidential communication with previously unknown parties. When one has a confidential channel, it is easy to establish the symmetry needed to later encrypt the communication for transmit over an insecure channel, be that radio-waves or the public internet - yet if you already have established such a secure channel, why would you want to send your information over an insecure medium in the first place?

2.1.2 Asymmetric Cryptography and the Foundation of Public-key Cryptosystems

Diffie and Hellman published the concept of a so called public-key cryptosystem by which the above difficulties could be mitigated in 1976 [1], though without presenting a concrete realization thereof. The scheme is surprisingly simple and works as follows: each participant publicly reveals an encryption procedure E and keeps the corresponding decryption procedure D private. Sending encrypted messages is done in the obvious way: the sender fetches the addressee's published encryption scheme, applies it to the message and sends the resulting ciphertext over the insecure channel to the recipient.
In order to constitute a secure and efficient public-key cryptosystem, Diffie and Hellman further proposed four properties to be satisfied by the encryption and decryption schemes E and D, respectively. The original form can be found in [4]

and corresponds to the following, less abstract version:

1. Deciphering an enciphered message must yield the message
2. $D(M)$ and $E(M)$ are easy[1] to compute
3. Given the encryption key E or any other public information X, an attacker cannot efficiently[2] derive the decryption key D
4. Enciphering a deciphered message must yield the message

For the purpose of this text, we shall have a slightly more concrete formulation in terms of computational complexity than the originally proposed version:

1. $D(E(M)) = M$
2. $D(M),\ E(M)\, are\, O(n^{k_1})$
3. $\nexists\, F(E, X)\ s.t.\ (\, F(E, X) = D\ \wedge\ F\, is\, O(n^{k_2}))$
4. $E(D(M)) = M$ ($\Rightarrow$ E, D are bijections)

Here $O(f(n))$ denotes bit-complexity or running-time dependency on the input-length n and X denotes any publicly available information.
Rivest, Shamir and Adleman presented their proposal of such a system in 1978 in the form of the RSA-Algorithm[4]. Algorithms satisfying all of the above properties are commonly referred to as *trap-door one-way permutations*, those lacking property 4 as *trap-door one-way functions*. Obviously this naming convention hints at their above stated properties: these functions are easy to compute in one direction but supposedly hard to invert (thus "one-way") unless one happens to know the private information to open the "trap-door". As we shall see, a cryptosystem can be constructed using algorithms satisfying properties 1 to 3, i.e. using trap-door one-way functions. Yet, in order to establish a signature scheme property 4, permutability, will be necessary.

[1] For a more elaborate explanation on how computational problems and complexity can be classified, please see chapter 1 of this volume

[2] As footnote 1

2.1.3 Building a Public-Key Cryptosystem from a Trap-Door One-Way Function

Making use of the rather abstract specifications given before, we shall outline here how to establish a so called public-key cryptosystem.
To establish a public-key cryptosystem, each user S performs the following steps:

1. Create an encryption-scheme E_S and a corresponding decryption-scheme D_S, commonly referred to as *public key* and *private key*, respectively.
2. Make the encryption-scheme E_S publicly available[3].

The message transfer using the public-key cryptosystem from sender S and receiver R is realized by the following procedure:

1. S gets the encryption-scheme E_R that corresponds to the receiver R.
2. S encrypts her message M using E_R, thereby generating the enciphered message $C = E_R(M)$.
3. S sends C over the insecure channel to R.
4. R recovers the message M by calculating $D_R(C) = D_R(E_R(M)) = M$.

Note that this process as *works* since E and D satisfy property 1 and 2 only. It results in a state where only the recipient R and sender S are in hold of the message M in plain text. *Confidentiality* of this system critically relies on property 3, as in this model the attacker is expected to have access to both E and C. The fact that there is no efficient way to compute D from the publicly available E is thus the core requirement for such a scheme to be considered secure. The combination of properties 2 and 3 allow the system to cope with high computing power on the attacker side[4].

[3]There are several ways of achieving this. For two examples of widely deployed implementations see e.g. PGP[5] for an implementation of a *web of trust* or x509 [6] for an implementation using so called *certificate authorities*.

[4]To illustrate this claim, assume S is protecting the message against an adversary Q that has 10^6 times the processing power S does. Since according to property 2 all algorithms Q can use to derive M from C have bit-complexity $O(k_2^n)$ whereas decryption and encryption only are of bit-complexity $O(n^{k_1})$, S can increase the input-length n by virtue of Bernoulli's inequality such that $Q's$ algorithms consume the computing time for S and R multiplied by an arbitrarily high factor f. Note that, while theoretically achievable with all algorithms satisfying the above properties, in order to be practical, the processing-time for encryption and decryption should be reasonably low even with high advantage-factors f, that is k_1 should preferably be small and k_2 should be large.

We conclude that this system - satisfying properties 1 to 3 only - allows to establish confidential communication over an insecure channel without relying on the state of shared secrets, thereby solving the key distribution problem[5]. In order to be of practical use, a public-key cryptosystem should also ensure *integrity* and *authenticity*[6].

2.1.4 Digital Signature Schemes and Integrity of Messages

Symmetric cryptography as depicted above inherently links confidentiality and authenticity by the assumption that the encryption scheme is available to trusted parties only. By assuming that only the legitimate sender has access to the shared secret the receiver R can therefore rely on the enciphered message C coming from the sender S. Obviously this property is not satisfied in a public-key cryptosystem the way we established it previously: any participant of the system can easily send a confidential message to the recipient R impersonating S as the sender. The system as defined does not provide any means for R to cryptographically verify the identity of the sender.
Using property 4 we can deploy a signature scheme that solves this problem. In order to encrypt *and* digitally sign a message the sender S

1. computes the signature $X = D_S(M)$ using his private key D_S
2. enciphers X with E_R thereby creating $C = E_S(X) = E_R(D_S(M))$

To recover the message M from the signed ciphertext C, the recipient R performs these steps:

1. decipher C using his private key D_R, thereby revealing $X = D_R(C)$
2. verify the message by using the publicly available E_S to compute $M = E_S(X) = E_S(D_R(C))$

[5]It is worth mentioning that as of today in practical implementations, systems like RSA or Diffie-Hellmann [1] are often only used to establish a shared secret or signatures. The secret is then used as key for symmetric algorithms due to the better performance of symmetric as compared to asymmetric algorithms on both current special and current general purpose hardware.

[6]For further information, the reader might read up on the so called CIA-triad in information security.

Again note that the signature creation[7] and verification[8] process is well defined due to property 4, permutability. Furthermore, it can only be done by the sender S as long as the private key is kept secret in accordance with the requirements of the model. Note also that this property is even stronger than mere authentication where only R needs to be convinced that the originator of the message actually is S. Digital signatures can also be used by R to prove to a *third party* such as a judge that S indeed sent the message. This is because it is guaranteed that R has not modified the message and created a matching signature herself. This claim holds since R must have D_S in order to create a signature (step 1 for the sender). This in turn cannot be the case since R neither has access to D_S by the model assumption nor can she compute it due to property 3.
In the form outlined so far, both symmetric as well as asymmetric procedures on their own fail to provide integrity of messages. Assuming the attacker can intercept the enciphered message and modify it arbitrarily even though without recovering the plain-text message M, the recipient R does not have any means to detect this tampering. For many practical implementations such as IPSEC or SSL/TLS timeliness of a message is also considered to be of importance. These issues do normally get resolved by attaching (H)MACs[9] and/or time stamps to the plaintext before enciphering it. Note that another issue that needs to be addressed by a public-key cryptosystem is how to ensure that a given digital public-key belongs to a non-digital entity such as a real or legal person. There is a selection of different, currently widely deployed realizations of such schemes pointed to in [5], S/MIME [7] and x509-Certificate-Authorities [6].

2.2 Case Study: The Rivest Shamir Adleman (RSA) Algorithm

Up to that point, we did only rely on the properties 1 to 4 to prove our claims, yet without giving an example of a function that satisfies them. We shall thus have a closer look at a concrete public-key cryptosystem and its underlying trap-door one-way permutation, namely RSA. According to [8], asymmetric encryption schemes were available to intelligence agencies before the publication of Rivest, Shamir and Adleman, yet RSA is considered to be the first such algorithm available to the scientific community and thus to the public. Furthermore RSA is nowadays still very widely deployed in operative information processing systems

[7] corresponding to step 1 for the sender S
[8] corresponding to step 2 for the recipient R
[9] (H)MAC - (Hash-based) Message Authentication Code, see [7]

affecting many aspects of our every day life[10].
First we give the detailed, mathematical description of the RSA algorithm, then the proofs of the Diffie-Hellmann properties 1 and 4 and finally an outline of an algorithm that guarantees property 2. We conclude this section with pointers regarding the "proof" of property 3.

2.2.1 The RSA Algorithm

The RSA algorithm operates on integers that are representing the messages to be transmitted. It is therefore prerequisite that messages are stored as integer numbers in order to apply the scheme. Even though sounding like a non-trivial concept to the novice, the conversion of human readable text to integers is inherently done on virtually all computers as they store information in bit patterns, which in turn can be interpreted as integers. In order to understand the mathematical specification that is to be given, we first introduce an equivalence relation on the set of integers[11]

Definition Two integers a and b are called *equivalent modulo* m, if their difference $a - b$ is an integer multiple of m:

$$a \equiv b \pmod{m} \iff \exists\, k \in \mathbb{N} \text{ s.t. } a - b = k \cdot m$$

Now we can proceed with the algorithms for key-generation, encryption and decryption. The key-generation process for each participant works as follows:

Algorithm: Key-Pair generation

1. Generate two sufficiently large[12] random and distinct primes p and q of roughly the same size.
2. Compute $n = pq$ and $\phi = (p-1)(q-1)$.
3. Select a random integer e, $1 < e < \phi$, such that gcd[13]$(e, \phi) = 1$.

[10] Examples are all secured website access including eBanking or credit card data transmission over browser frontends (x509), VPNs (IPSEC or SSL) and likely large parts of backend financial and governmental services.

[11] The very same concept is also used for introductory examples of fields in courses on algebra.

[12] As of today 512 bit numbers for each, p and q, are considered secure also against concerted attacks. Current implementations range from 1024 bits (OpenSSH [9] to 8192 bits (Microsoft RSA Cryptographic Service Providers [10]) for both p and q, resulting in 2048 and 16384 bit public-keys.

[13] gcd: greatest common divisor.

4. Compute the unique integer d, $1 < d < \phi$, such that $ed \equiv 1 \pmod{\phi}$ using the extended Euclidean algorithm [7].

5. Publish the set of integers (n,e) as public-key E and keep the integer d as private-key D.

In order for the sender S to encrypt a message M for the recipient R using the public key E_R, the following algorithm is applied:

ALGORITHM: ENCRYPTION

1. S obtains $R's$ public-key E_R

2. Represent the message M as an integer[14] m with $m < n - 1$.

3. Compute the enciphered message as $C = m^e \bmod n$

4. Send the ciphertext C to the recipient R

Note that since encryption as well as decryption are bijections from the set $L = \{ x \epsilon \mathbb{N} \mid x < n - 1\}$ onto itself, the storage requirements for the ciphertext C are in general the same as for m. Both m and C are elements of L, thus in statistical average require the same amount of memory to be stored.
In order to decrypt a given ciphertext C with the corresponding private key D, the following algorithm is to be applied:

ALGORITHM: DECRYPTION

1. Compute the integer m representing the message as $m = C^d \bmod n$ (derive the message M from m using the same convention as S used)

A SMALL NUMBERED EXAMPLE

In order to illustrate the algorithm, we present a concrete example with artificially small parameters.

[14] As stated above, this can be trivially done but given the condition that $m < n-1$ it might be necessary to split the message into blocks. This does not affect the procedure's security as each block fulfills the requirements on it's own.

Key generation

1. Chose $p = 7$, $q = 19$
2. Calculate $\phi = (p-1) \cdot (q-1) = 6 \cdot 18 = 108$, $n = p \cdot q = 7 \cdot 19 = 133$
3. Chose $e = 5$. We have chosen a prime here for the sole reason that this guarantees $\gcd(e, \phi) = 1$.
4. Find d such that $ed \equiv 1 \pmod{n}$.
5. Thus we have our keypair: private-key $D = (d) = 65$, public-key $E = (e, n) = (5, 77)$

Encryption

1. Find M's integer-representation m. Let $m = 6$ for the purpose of this example.
2. Using the presented algorithm calculate $C = m^e \bmod n = 6^{37} \bmod 133 = 62$

Decryption

1. Using the presented algorithm calculate $m = C^d \bmod n = 62^{65} \bmod 133 = 6$

2.2.2 Proving that RSA is a "trap-door one-way permutation"

This proof can currently only partially be mathematically rigid since - as we shall see later - it is as of today not known whether property 3, the one that guarantees the security of the cryptosystem mathematically, is satisfied by RSA. Property 1 and 4 can be proven rigidly using modular arithmetic, *Fermat's* theorem and the chinese remainder theorem. Furthermore we shall present an example algorithm for modular exponentiation that does have bit-complexity $O(n^k)$ thereby proving property 2.

Proof of properties 1 and 4

Due to the definition of the algorithm, properties 1 and 4 are proven once we can show that

$$(m^e)^d \overset{Prop.1}{\equiv} m \overset{Prop.4}{\equiv} (m^d)^e \pmod{n}.$$

That is, we want to show that deciphering an enciphered message yields the message and that enciphering a deciphered message also yields the message.
We begin by using that $ed \equiv 1 \pmod{\phi}$ is assured by the way d was calculated, together with our definition of 'equivalence modulo an integer', and instantly conclude that

$$\exists\, k \,\epsilon\, \mathbb{N} \text{ s.t. } ed = 1 + k\phi.$$

We now consider two cases regarding $\gcd(m, p)$:

- p does not divide $m \Rightarrow \gcd(m, p) = 1$ as p is prime and thus itself does not have any divisors other than 1 and p.
- p does divide $m \Rightarrow \gcd(m, p) = p$ with the same argument.

For the first case, using Fermat's theorem[15] we can conclude that

$$m^{p-1} \equiv 1 \pmod{p}$$

After rising both sides of the congruence to the power $k \cdot (q-1)$ and then multiplying both sides by m we get:

$$m^{k(p-1)(q-1)+1} = m^{1+k\phi} \equiv m \pmod{p}$$

In the second case, the above congruence also holds: If $\gcd(m, p) = p$, then

$$\exists\, r \,\epsilon\, \mathbb{N} \text{ s.t. } m^x \equiv (p \cdot r)^x \equiv p^x \cdot r^x = 0^x \cdot r^x \equiv^{16} 0 \pmod{p} \quad \forall\, x$$

Thus for both cases we have by definition of d and e

$$m^{e \cdot d} \equiv m \pmod{p}$$

Substituting q for p using the same arguments,

$$m^{e \cdot d} \equiv m \pmod{q}$$

Given that p and q are distinct primes, it follows from the chinese remainder theorem[17] that

$$m^{e \cdot d} \equiv m \equiv m^{d \cdot e} \pmod{n = p \cdot q}$$

Thus properties 1 and 4 are proven.

[15] Fermat's Theorem
Let a be an integer, p be a prime:

$$\gcd(a,\, p) = 1 \;\Rightarrow\; a^{p-1} \equiv 1 \pmod{p}$$

[16] And for $x_1 = 1 + k\phi$, $x_2 = 1$ in particular, thus: $m^{x_1} \equiv m^{x_2} = m \;(\equiv 0) \pmod{p}$

[17] The Chinese Remainder Theorem
Let $p_1, p_2 \ldots p_i$ be pairwise relatively prime (i.e. $\gcd(p_l, p_m) = 1 \;\forall\; l, m \,\epsilon\, \{1, \ldots i\}, l \neq m$).

Proof of property 2

Because property 2 states that encryption and decryption both should be computationally efficient algorithms, the existence of an algorithm that can efficiently compute $m^e \bmod n$ and $m^d \bmod n$ proves this property in the case of RSA. Note that if such an algorithm could not be found, it is easy to see that the intuitive way to perform encryption and decryption would be of exponential bit-complexity in the key-length.

Algorithm for modular exponentiation

To apply the algorithm, let $a, k \,\epsilon\, \mathbb{N}$, $k < n$ and $\sum_{i=0}^{t} k_i 2^i = k$ be k's binary representation. It can easily be verified that the first condition on a, k is met by RSA and that every $k \,\epsilon\, \mathbb{N}$ has a binary representation of this form. The algorithm itself is defined as:

1. Set $b \leftarrow 1$. If $k = 0$ then return b.
2. Set $A \leftarrow a$.
3. If $k_0 = 1$ then set $b \leftarrow a$.
4. For i from 1 to t do the following:
   (a) Set $A \leftarrow A^2 \bmod n$.
   (b) If $k_i = 1$ then set $b \leftarrow A \cdot b \bmod n$.
5. Return $b = a^k \bmod n$.

This algorithm has bit-complexity $O((\ln n)^3)$ [7] thus property 2 is proven.

For any given set $a_1, a_2, ..., a_i, a_m \,\epsilon\, \mathbb{N}$, z can be found such that the system of simultaneous congruences

$$\begin{aligned} z &\equiv a_1 \pmod{n_1} \\ z &\equiv a_2 \pmod{n_2} \\ &\vdots \\ z &\equiv a_i \pmod{n_i} \end{aligned}$$

is solved. Furthermore,

$$z \equiv a_m \pmod{\prod n_k} \quad \forall\, z \text{ solving the system}$$

"Proof" of property 3, the relation to factorization

As mentioned above, there is no rigid proof of property 3, or stated differently: the mathematical security of RSA. An RSA encrypted message can be read without previously having access to D if an attacker can efficiently calculate e^{th} roots modulo n, n being a composite. Stated mathematically:

$$\text{Find } m \text{ such that } C = m^e \bmod n.$$

If C, e and n are created according to our specification, this is called the *RSA problem*. One possible way to find a solution to this equation is to factorize n into p and q and then run the algorithm backwards, that is derive D as $D = ((p-1)\cdot(q-1)-1)\cdot e^{-1}$. As of today, there are no factoring algorithms known that are of bit-complexity $O(n^k)$, i.e. run in polynomial time. Note that it is currently not proven that factorization of n is the most efficient way to solve the *RSA problem* and according to [11] there is evidence that this can in fact not be proven.
What can be shown though is that if an attacker can find d efficiently using only the publicly available information E (consisting of n and e) by whatever means, he also can factorize n efficiently. We will show this relation here, that is we prove that if you can compute d from a given public-key (n, e), you can factorize n efficiently. The latter in turn is as of today *considered* impossible on Turing machines and therefore serves as a *hint* towards the impossibility of cracking RSA using Turing machines.

Proof Since $ed \equiv 1 \pmod{\phi}$, by definition of the equivalence, an integer k can be found such that $ed - 1 = k\phi$. Hence by Euler's theorem[18]

$$a^{ed-1} \equiv 1 \pmod{n} \; \forall\, a \text{ relatively prime to } n$$

Let $ed - 1 = 2^s t$, where t is an odd integer. It can be shown that

$$a^{2^{s-1}t} \not\equiv \pm 1 \pmod{n}$$

for at least half of all a that are relatively prime to n. If a is such an integer then

$$\gcd(a^{2^{s-1}t} - 1,\, n)$$

[18]Euler's Theorem
Let $n \geq 2$ be an integer. If a is relatively prime to n, then $a^{\varphi(n)} \equiv 1 \pmod{n}$. Here, $\varphi(n)$ is defined as the Euler totient function giving the number of positive integers less than n which are relatively prime to n.

is a non-trivial factor of n. The attacker therefore needs to randomly select an integer a that is relatively prime to n and compute $\gcd(a^{2^{s-1}t}-1,\, n)$. Statistically a non-trivial factor of n is obtained after 2 steps as the equation above holds for half of all candidates for a. Given that the $\gcd(x, y)$ can be calculated in polynomial time, this constitutes an efficient factorization method for n and the claim is proven.

2.3 Attacks on the RSA Algorithm, their Mitigation and Relation to Quantum Computing

There are different attacks that can be driven on RSA crypto systems, all but one of which as of today can be mitigated by various small extensions on the algorithm as proposed above. The one attack that is of systematic nature is the one based on efficient factorization of n. A non-exhaustive list of other attacks as well as suggestions for countermeasures can be found in [7]. Note that as of today no efficient algorithm for factoring numbers are known, thus the quality of an RSA implementation vastly depends on the consideration of the above referred non-systematic issues related to RSA. In our context though, the focus shall be on the systematic one since it is the only one that directly relates to quantum computing.

2.3.1 Efficiency of Factorization Algorithms

In section 1.2.2 we mentioned the problem of factorization. If an attacker can efficiently factor n (which, together with e is part of the public key and thus public knowledge by the model assumption), the private key D can easily be obtained the same way the key pair was generated by the legitimate user:

- Reveal p and q instantly by factoring n, as $n = p \cdot q$.
- Derive $\varphi = (p-1) \cdot (q-1)$.
- Compute the unique integer d, $1 < d < \phi$, such that $ed \equiv 1 \pmod{\phi}$ and thereby obtain the private key $D = \{d\}$.
- Any plaintext m_1 can be computed by the attacker from the corresponding C_1 using d.

Given the fact that efficient factorization algorithms will therefore break RSA's security, it is worthwhile to have a closer look at the status quo of integer factorization. Integer factorization is a challenge that has a long history ranging even further back than Fermat and Euler trying to find efficient algorithms for the problem. As of today, it is not clear to what complexity class the functional version of the factorization problem belongs to. It is trivial to show that it lies in FNP but the question whether it also lies in FP is currently unanswered. The same applies to the question whether the decision-version of the problem belongs to the NP-class. Due to the fact that Shor's algorithm [12] can factorize integers in polynomial time on quantum based computers, the problem is known to be in BQP. On Turing-based machines, one of the fastest currently known algorithms, the *General Number Field Sieve* [13], is of subexponential bit-complexity[19].
In 2005, factoring a 640 bit number (130 decimal digits) took several months on a 80-node cluster using Opteron CPUs. As of today, even much bigger computing clusters will factorize 4096 bit RSA-Keys only within much larger timescales using such algorithms. If further progress is made in terms of classical computing power, it thus suggests itself to increase the key-size. Yet this strategy obviously will be inappropriate once efficient quantum computers become available. Due to the close relation of the factoring problem to the problem of the discrete logarithm used in the other widely deployed algorithms DSA [7] and ElGamal [7], these two cryptosystems face the same issue regarding progress in quantum computing as RSA itself and thus also are no solution to the problem at hand. Consequently, other cryptographic approaches are to be investigated - that is to take some other mathematical problem, one that can neither on a Turing based nor on a quantum computer be efficiently solved, to lay the ground for a concrete implementation of a trap-door one-way permutation.

2.3.2 Post-Quantum Cryptography

In terms of complexity classes, it would be favorable to use a problem that is proven to be e.g. NP-hard. Such cryptosystems do exist, one example being the McEliece cryptosystem [14] that was published in 1978 (the same year as RSA) and uses the problem of decoding arbitrary linear codes. This particular problem was shown to be NP-hard [15] but the algorithm still didn't succeed - probably because of several issues that RSA doesn't face[20]. Amongst already

[19] For the GNFS bit-complexity can be shown to be $O(e^{(c+o(1))(log\ n)^{1/3}(log\ log\ n)^{2/3}})$

[20] Amongst others that the keys are much bigger and the fact that the ciphertext is - for that time significantly - bigger than the plaintext. Also, the algorithm seemingly hasn't been investigated as intensely as RSA and uses a less understood problem at it's core.

known cryptosystems, several other concepts such as lattice-based cryptography, Merkle-type signature schemes and multivariate cryptosystems are investigated in order to cope with the expected progress in quantum computing.

2.4 Conclusions and Summary

We have seen how the bootstrapping problem of symmetric cryptography can be overcome by a simple model proposed by Diffie and Hellmann, using the concept of a *one-way trap-door function*. That is, a function that satisfies 3 relatively simple conditions. Furthermore we have shown how *one-way trap-door permutations* can be used to establish a signature scheme for digital messages that provides for the digital equivalent of classical signatures. As one concrete implementation of such a one-way trap-door permutation, we had a closer look at the proposal by Rivest, Shamir and Adleman, commonly referred to as RSA. All but the third property, that is the one that relates to the difficulty of the process inversion, have been proven rigidly. The third property turns out to be the one that links the cryptosystem to quantum computing due to the fact that it closely relates to the factorization problem. We have seen that once efficient factorization becomes possible, that is once Shor's algorithm is implemented on a quantum computer, RSA will not satisfy property 3 anymore. Thus RSA becomes insecure with the advent of quantum computing and other cryptographic algorithms such as McEliece's proposal will be necessary to provide confidential messaging over untrusted communication links.

Bibliography

[1] W. Diffie and M. E. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory **22**, 644 (1976).

[2] F. L. Bauer, *Decrypted Secrets* (Springer, New York, 2000).

[3] W. F. Ehrsam, C. H. W. Meyer, R. L. Powers, J. L. Smith, and W. L. Tuchman, *U.S. Patent Nr. 3962539* (1974).

[4] R. L. Rivest, A. Shamir, and L. M. Adleman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, Communications of the ACM **21**, 120 (1978).

[5] P. R. Zimmermann, *The Official PGP User's Guide* (MIT Press, Cambridge, 1995).

[6] C. Adams and S. Farrell, *Internet Society RFC 2510* (1999).

[7] A. J. Menezes, P. C. Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography* (CRC Press, Boca Raton, 1996).

[8] J. Ellis, *The Story of Non-Secret Encryption* (1987), http://www.cesg.gov.uk/ellisdox.ps.

[9] OpenSSH Development Team, *ssh - OpenBSD Reference Manual* (2008), http://www.openbsd.org/cgi-bin/man.cgi?query=ssh.

[10] Microsoft Inc., *CryptoAPI Cryptographic Service Providers* (2008), http://msdn.microsoft.com/en-us/library/bb931357(VS.85).aspx.

[11] D. Boneh and R. Venkatesan, *Breaking RSA may not be equivalent to factoring*, Eurocrypt Proceedings **1233**, 59 (1998).

[12] P. W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM Journal on Computing **26**, 1484 (1997).

[13] J. P. Buhler, H. W. Lenstra, and C. Pomerance, *Factoring Integers with the Number Field Sieve*, Lecture Notes in Mathematics **1554**, 50 (1993).

[14] R. J. McEliece, *A Public-Key Cryptosystem Based on Algebraic Coding Theory*, Deep Space Network Progress Report **42-44**, 104 (1978).

[15] A. Salomaa, EATCS Monographs on Theoretical Computer Science **23** (1990).

Chapter 3

Grover's Algorithm

Raffaele Solcà
supervisor: Alejandro Daleo

We have seen some algorithms based on quantum computation that are more powerful than classical algorithms. We now study Grover's algorithm. This is an algorithm designed to search for an element in a set. As many other quantum algorithms, it performs better than its classical counterparts. We study how the algorithm is defined how it converges. Then we look at the geometrical interpretation and at an example. Finally we study the case of the problem with multiple solutions.

3.1 Introduction

We saw that quantum mechanical computers are more powerful than classical computers. Grover's algorithm is another application of quantum computers that is better than any classical algorithm to solve the problem of searching an element in a set.
Suppose we have a phonebook and we are looking for a person which have a specified phone number. How we solve this problem?
Classically we choose a random record from the phone-book and we check if it has the searched phone number. If this is the correct answer we have solved our problem. There is a probability of $1/N$ (where $N = 2^n$ is the number of records and n the number of bits we need) that this is the correct answer. We can increase the probability of success if we repeat this procedure on other records. Every time we pick a new entry we increase the probability by $1/N$. To solve the problem we need in average $\frac{N}{2} = O(N) = O(2^n)$ steps. We will see that using a quantum

mechanical computer and Grover's algorithm we need only $O\left(\sqrt{N}\right) = O\left(2^{\frac{n}{2}}\right)$ steps.

3.2 The quantum computer

This section is a short introduction on quantum computers. Here it will be explained only the principal things we will use in this report. For more information see [1] and [2].
A quantum mechanical computer works with qubits. A qubit can be in states $|0\rangle$ and $|1\rangle$ at the same time through a quantum superposition. We represent the state with $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$ for $\alpha, \beta \in \mathbf{C}$ and $\alpha^2 + \beta^2 = 1$. A qubit can be represented also with an unitary vector with the basis $\begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle$ and $\begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle$ We can apply transformations to qubits using operators called quantum gates. An operator that transform a unitary vector in another unitary vector is called unitary, and satisfies $U^+U = I$. We will use the Hadamard gate, that is given by

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{3.1}$$

This gate is used to create superposition states.
For multiple qubits states there is a generalization of the Hadamard gate: the Walsh-Hadamard gate W, that is defined by $W = H^{\otimes n}$, that is a Hadamard gate acting independently on each qubit. We can write the matrix as

$$W_{ij} = 2^{\frac{n}{2}} (-1)^{\bar{i} \cdot \bar{j}}. \tag{3.2}$$

Also the Walsh-Hadamard gate is used to create superpositions of the states.
There is another important thing to remember: If we measure a qubit in the state $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$ the wavefunction collapses and we measure the state $|0\rangle$ with probability α^2 or $|1\rangle$ with probability β^2. Note that it is not possible to measure the value of α or β.

3.3 The abstracted problem

We can formulate the phone-book reach problem in an abstract way. Consider a function $f : A -> \{0, 1\}$ with $f(s) = 1$ for one $s \in A$ and $f(a) = 0$ for all $a \neq s$. A is a set with N elements that can be represented with n bits ($N = 2^n$). We are looking for the the element x of A that satisfies $f(x) = 1$.

3.4 The algorithm

To solve this problem quantum mechanically we use Grover's algorithm. This algorithm employs registers, the first with n qubits and the second with one qubit. We need also an oracle (or black box), which is a linear operator O_f dependent on the function f, such that $O_f(i,j) = |i, j \oplus f(i)\rangle$, where i the state of the first register, and j is the state of the second register. We can easily see that

$$\begin{aligned} O_f\left(|i\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right) &= \frac{O_f(|i\,0\rangle) - O_f(|i\,1\rangle)}{\sqrt{2}} \\ &= \frac{|i\,f(i)\rangle - (|i\,1 \oplus f(i)\rangle)}{\sqrt{2}} \\ &= (-1)^{f(i)}|i\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{aligned} \tag{3.3}$$

i.e. this operator simply inverts the phase of any state for which $f(i) = 1$.
The algorithm, as defined in [3] and [4], follow the following steps :

i) Initialize the first register to the superposition: $\left(\frac{1}{\sqrt{N}}, \frac{1}{\sqrt{N}}, ..., \frac{1}{\sqrt{N}}\right)$, i.e. there is the same amplitude to be in each of the N states; and the second register to the state $|\varphi\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$

ii) Repeat the following unitary operation $O\left(\sqrt{N}\right)$ times (the number of repetitions is important and will be discussed later):

   a) Apply the oracle

   b) Apply the diffusion transform D given by the matrix

$$D_{ij} = \frac{2}{N} \text{ for } i \neq j \text{ and } D_{ii} = -1 + \frac{2}{N} \tag{3.4}$$

   to the first register.

iii) Measure the resulting state of the first register. This will be the desidered state with a probability of at least $\frac{1}{2}$.

3.5 Implementation of the algorithm

We now analyze the steps of the algorithm, and how we can implement them:

i) To reach the superposition of the N basis state of the first register we can initialize it in the state $|0,0,0,...,0\rangle$ and apply the the Walsh-Hadamard operator $W = H^{\otimes n}$, i.e. we apply an Hadamard gate to each qubit.

The second register can be initialized in the state $|1\rangle$ and after the application of an Hadamard gate in will be in the state $|\varphi\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$

iia) Now we look at what happens if we apply the oracle to a state $|\phi_1\rangle = \sum_{i=0}^{N-1} \alpha_i |i\rangle$ with $\alpha_i \in \mathbf{R}, \alpha_i > 0$ and $\sum_{i=0}^{N-1} \alpha_i^2 = 1$. As we have seen in (3.3) the second register does not change. If we call $|\phi_2\rangle$ the resulting state we obtain

$$|\phi_2\rangle = \sum_{i=0}^{N-1} (-1)^{f(i)} \alpha_i |i\rangle \tag{3.5}$$

We can easily see that the only change is that the amplitude of the searched element is now negative, but have the same absolute value. From now on, we will denote the oracle with the operator U_f defined in such a way that changes the sign of the amplitude of the state of the solution of the problem and for simplicity we neglet the second register that is always in state $|\varphi\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.

iib) We start noting that the operator D, defined in (3.4), is unitary. We see that D can be represented in the form $D = -I + 2P$ where I is the identity matrix and P is a projection matrix with $P_{ij} = \frac{1}{N}$. Applying P on any vector v gives a vector with all component equal to the average of the components of v. We see that I,P and D are real symmetric operators. It follows that to prove that D is unitary is equivalent to prove that $D^2 = I$. We begin verifying that $P^2 = P$

$$\left(P^2\right)_{ij} = \sum_{l=0}^{N-1} P_{il} P_{lj} = N \left(\frac{1}{N}\right)^2 = \frac{1}{N}. \tag{3.6}$$

Using $D = -I + 2P$ and $P^2 = P$ we easily prove that $D^2 = I$ and so D is unitary.

Now we will prove that the diffusion operator D can be expressed as composition of three local quantum mechanical operations, i.e. operators that act on single qubit and that can be implemented with elementary unitary operations.

The diffusion transformation D as defined in (3.4), can be implemented as a product of three unitary transformation, $D = WRW$, where W is the Walsh-Hadamard operator and R is a phase rotation matrix with

$R_{ij} = 0, R_{ii} = -1, R_{00} = 1$. To prove this equality we need first of all a representation of the operator W. As we have seen in section 3.2 applying the operator W is like applying a Hadamard gate to each qubit, and the elements of the matrix W are given by:

$$W_{ij} = 2^{\frac{n}{2}} (-1)^{\bar{i} \cdot \bar{j}}, \tag{3.7}$$

where $\bar{i}$ is the binary representation of i and $\bar{i} \cdot \bar{j}$ is the bitwise dot product of the two bit strings.

Now, to see that $D = WRW$, we simply evaluate WRW using the fact that $R = -I + R_1$ where I is the identity matrix and $R_{1,00} = 2$ and $R_{1,ij} = 0$ for j or $i \neq 0$. Because of $HH = I$, it is simply to prove that $WW = I$:

$$W^2 = \left(H^{\otimes n}\right)^2 = \left(H^2\right)^{\otimes n} = I^{\otimes n} = I. \tag{3.8}$$

So we have that $W(-I)W = -I$. Using matrix multiplication, the definition of R_1 and defining $D_1 = WR_1W$ we have that

$$D_{1,ij} = \sum_{l,m} W_{il} R_{2,lm} W_{mj} = 2W_{i0}W_{0j} - \frac{2}{N} (-1)^{\bar{i} \cdot \bar{0} + \bar{0} \cdot \bar{j}} = \frac{2}{N} \tag{3.9}$$

It follows that $D = WRW$, so D can be implemented with elementary unitary operations.

iii) Measuring the state of first register we have a collapse of the wavefunction into the desidered state (the state which satisfied the condition) with a probability of at least $\frac{1}{2}$ (depending on the number of steps).

See [5] and [6] for more detail about the implementation, and [7] for more details about the diffusion trasformation.

3.6 Convergence

This section will prove that the algorithm needs $O\left(\sqrt{N}\right)$ steps, but we will see in next section that the probability of success is not monotonic, hence we need to know exactly how many steps we have to do to have a great probability to find the exact solution.
We start considering a vector with amplitudes k_1 for the state that satisfied the condition $f(x) = 1$ and l_1 for each of the other $N - 1$ states, with $k_1 < 0$ and

$l_1 > 0$. After applying the operator D we obtain:

$$k_2 = \left(\frac{2}{N} - 1\right) k_1 + \frac{2\,(N-1)}{N} l_1 \tag{3.10}$$

$$l_2 = \left(\frac{2}{N} - 1\right) l_1 + \frac{2}{N} k_1 + \frac{2\,(N-2)}{N} l_1 = \frac{2}{N} k_1 + \frac{(N-2)}{N} l_1 \tag{3.11}$$

Now if we consider k_2 we see that, if $N > 2$, $\left(\frac{2}{N} - 1\right)$ is negative and $\frac{2(N-1)}{N}$ is positive. By assumption k_1 is negative and l_1 is positive hence k_2 is positive. Similarly, l_2 is positive if $|k_1| < \frac{N-2}{N}$. Solving the inequality $\sqrt{N} < \frac{N-2}{2}$ we obtain that the condition $|k_1| < \frac{N-2}{N}$ is satisfied when $|k_1| < \sqrt{N}$ for $N > 4 + 2\sqrt{3}$, that is for $N \geq 8$.
We consider now the change of the amplitude k in one step. We consider $0 < |k_1| < \frac{1}{\sqrt{2}}$ and $l_1 > 0$. Let us define

$$\Delta k = k_2 - k_1 = -\frac{2k_1}{N} + 2\left(1 - \frac{1}{N}\right) l_1. \tag{3.12}$$

Because of $|k_1| < \frac{1}{\sqrt{2}}$ and conservation of the total probability (As is known from linear algebra a unitary transformation does not change the modulo of a vector, hence the total probability is conserved) it follows that

$$l_1 = \sqrt{\frac{1 - k_1^2}{N-1}} > \frac{1}{\sqrt{2\,(N-1)}} > \frac{1}{2\sqrt{N}}, \tag{3.13}$$

and because of $k_l < \sqrt{N}$ is l_2 positive. From (3.12) it follows that $\Delta k > \frac{2\frac{1}{\sqrt{2}}}{N} + 2\left(1 - \frac{1}{N}\right)\frac{1}{2\sqrt{N}} > \frac{1}{2\sqrt{N}}$. Using this result we can immediately see that it exists a number M smaller than $\sqrt{2^n}$ such that, in M repetition of step ii) of the algorithm, k will be greater than $\frac{1}{\sqrt{2}}$. Hence the system is now sampled because the probability to find the desidered state is $k^2 > \frac{1}{2}$. Therefore we proved that the amplitude increases by $O\left(\frac{1}{\sqrt{N}}\right)$ in each iteration, and so $O\left(\sqrt{N}\right)$ steps are needed to identify the solution.

3.7 Geometric representation

All amplitudes and all components of the operators of Grover's algorithm are real. Therefore we can represent each the state of the algorithm in a real subspace of the Hilbert space. Let us now define $|\psi\rangle$ as the state of the first register after step i) of the algorithm and $|s\rangle$ to be the state of the solution of the problem.

We take this as non-orthogonal basis of our representation. We define θ as in Figure 3.1, hence

$$\sin\theta = \cos\left(\pi/2 - \theta\right) = \langle s \,|\psi\rangle = \frac{1}{\sqrt{N}}. \tag{3.14}$$

If we apply U_f to $|\psi\rangle$ we have:

$$|\psi_0\rangle = U_f\,|\psi\rangle = \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1} U_f\left(|i\rangle\right) = |\psi\rangle - \frac{2}{\sqrt{N}}\,|s\rangle \tag{3.15}$$

and it follows from Figure 3.1 that

$$\langle\psi|\,U_f\,|\psi\rangle = 1 - \frac{2}{N} = \cos 2\theta. \tag{3.16}$$

To conclude the first application of step ii) of the algorithm we have to apply

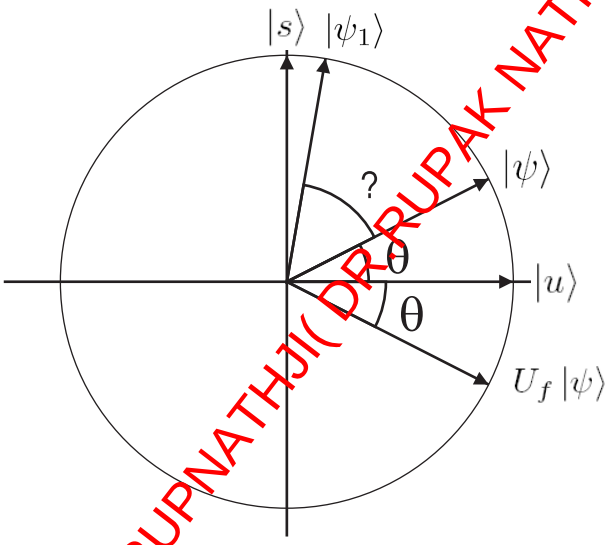


Figure 3.1: The state of first register represented in the real vector space spanned by $|s\rangle$ and $|u\rangle$. This is what happen when we apply step ii) the first time

the diffusion operation $D = -I + 2P$. For simplicity we can write the projection matrix P as $|\psi\rangle\langle\psi|$. We prove the equality of the two operators applying them on the same vector$|w\rangle = \sum_{i=0}^{N-1}\alpha_i\,|i\rangle$:

$$P\,|w\rangle \;=\; \sum_{i=0}^{N-1}\frac{1}{N}\sum_{j=0}^{N-1}\alpha_j\,|i\rangle \text{ and} \tag{3.17}$$

$$|\psi\rangle\,\langle\psi\,|w\rangle \;=\; \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}\alpha_j\,|\psi\rangle = \sum_{i=0}^{N-1}\frac{1}{N}\sum_{j=0}^{N-1}\alpha_j\,|i\rangle \tag{3.18}$$

Applying the operator D to $|\psi_0\rangle$, we obtain:

$$\begin{aligned} |\psi_1\rangle &= (2|\psi\rangle\langle\psi| - I)|\psi_0\rangle \\ &= (2|\psi\rangle\langle\psi| - I)|\psi\rangle - \frac{2}{\sqrt{N}}(2|\psi\rangle\langle\psi| - I)|s\rangle \\ &= |\psi\rangle + \frac{2}{\sqrt{N}}\langle\psi|s\rangle|\psi\rangle + \frac{2}{\sqrt{N}}|s\rangle \\ &= \left(1 - \frac{4}{N}\right)|\psi\rangle + \frac{2}{\sqrt{N}}|s\rangle \end{aligned} \tag{3.19}$$

It follows that the angle between $|\psi\rangle$ and $|\psi_1\rangle$ is

$$cos2\theta' = \langle\psi|\,\psi_1\rangle = 1 - \frac{2}{N}. \tag{3.20}$$

Therefore after the first application of step ii) of the algorithm, $|\psi\rangle$ rotates 2θ towards $|s\rangle$. In general a single application of step ii) is not enough to solve the problem. Hence we want to know what happens with other application of step ii). Let us define a unit vector $|\sigma\rangle$ making an angle α with $|\psi\rangle$ as in Figure 3.2. Let

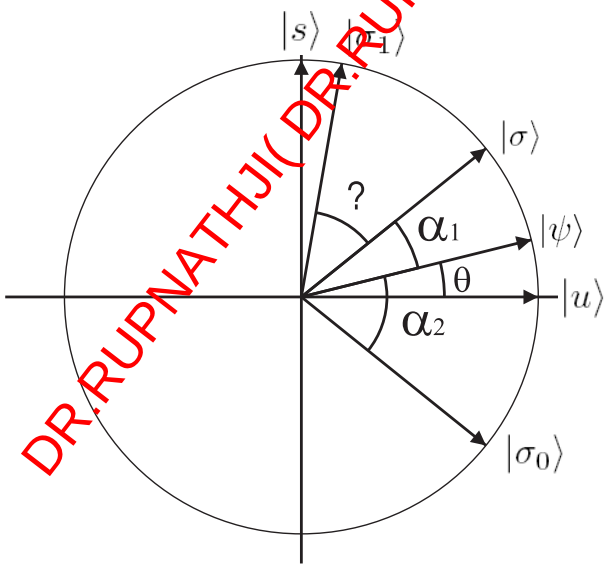


Figure 3.2: An application of step ii) on the state $|\sigma\rangle$

$|\sigma_0\rangle$ be the state after application of the operator U_f and $|\sigma_1\rangle$ after the application of the operator D. We define α_2 to be the angle between $|\psi\rangle$ and $|\sigma_0\rangle$. It is easy to see that $|\sigma_0\rangle$ is the reflection of $|\sigma\rangle$ around the horizontal axis. It follows that

$\alpha_2 - \alpha_1 = 2\theta$. Let now see what happens with the application of step ii) to $|\sigma\rangle$:

$$\begin{aligned} |\sigma_1\rangle &= D\, U_f\, |\sigma\rangle \\ &= (2\,|\psi\rangle\,\langle\psi| - I)\, U_f\, |\sigma\rangle \\ &= 2\,\langle\psi|\, U_f\, |\sigma\rangle\, |\psi\rangle - U_f\, |\sigma\rangle \\ &= 2\,\langle\psi\ |\sigma_0\rangle\, |\psi\rangle - |\sigma_0\rangle \\ &= 2\cos\alpha_2\, |\psi\rangle - |\sigma_0\rangle \end{aligned} \tag{3.21}$$

and calculate the angle between $|\sigma\rangle$ and $|\sigma_1\rangle$

$$\begin{aligned} \langle\sigma\ |\sigma_1\rangle &= 2\cos\alpha_2\,\langle\sigma\ |\psi\rangle - \langle\sigma\ |\sigma_0\rangle \\ &= 2\cos\alpha_2\cos\alpha_1 - \cos(\alpha_1 + \alpha_2) \\ &= \cos(\alpha_2 - \alpha_1) = \cos 2\theta. \end{aligned} \tag{3.22}$$

Therefore we proved that every application of step ii) of Grover's algorithm is a rotation of 2θ.
Let us define

$$|u\rangle = \frac{1}{\sqrt{N-1}} \sum_{\substack{i=1 \\ i\neq s}}^{N} |i\rangle. \tag{3.23}$$

It follows that after m applications of step ii) of the algorithm we have the state

$$(D\, U_f)^m\, |\psi\rangle = \cos((2m+1)\,\theta)\, |u\rangle + \sin((2m+1)\,\theta)\, |s\rangle\,. \tag{3.24}$$

For the probability of success to be close to 1 we have to choose m so that $(2m+1)\,\theta \approx \pi/2$ (see Figure 3.3), which happens when $m \approx (\pi - 2\theta)/4\theta$. Clearly we have to perform an integer number of iteration, it follows that $m = \lfloor \pi/4\theta \rfloor$. For big N $\sin\theta \approx \theta$ we get $m = \left\lfloor \pi/4\sqrt{N} \right\rfloor$. If we are satisfied with a probability of 1/2, as in Grover's paper, we have to perform half this number of steps, hence $m = \left\lfloor \pi/8\sqrt{N} \right\rfloor$. After m applications of step ii) we have that the probability p of finding the solution is given by

$$\begin{aligned} p &= \sin^2((2m+1)\,\theta) \\ &= 1 - \cos^2((2m+1)\,\theta) = 1 - \sin^2\left(\frac{\pi}{2} - (2m+1)\,\theta\right) \\ &> 1 - \sin^2(\theta) = 1 - \frac{1}{N}, \end{aligned} \tag{3.25}$$

so the probability of failure is no more than $\frac{1}{N}$.

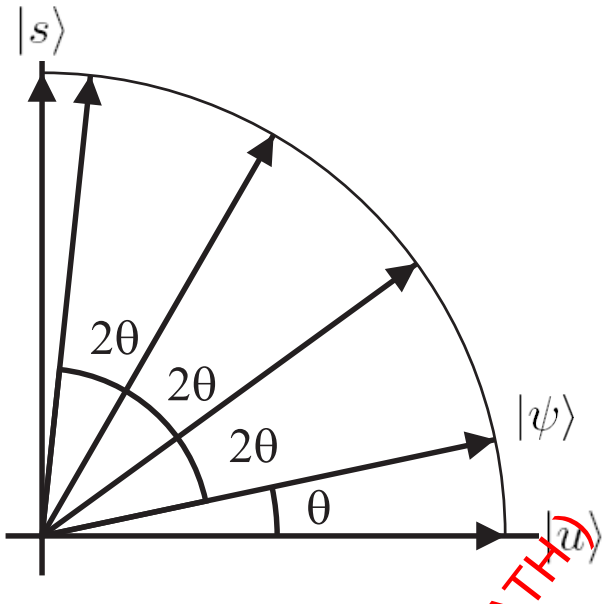


Figure 3.3: Determination of the number of steps and of the biggest failure of the angle

3.8 Example with N=4

We describe Grover's Algorithm for a search space of 4 elements, hence we need 2 qubits in the first register. Classically to have a probability of success of 1 we need to query the oracle 4 times. With a quantum computer we query the oracle only one time. This is because $\theta = \arcsin 1/\sqrt{4} = \pi/3$ so after one application of step ii) we have a probability of success of $p = \sin^2((2+1)\pi/3) = 1$.
Let us describe the initial quantum states with $|\psi_0\rangle$. It is

$$|\psi_0\rangle = |00\rangle\,. \tag{3.26}$$

After the application of step i) we have the state

$$|\psi\rangle = W|\psi_0\rangle = \frac{1}{2}\sum_{i=0}^{3}|i\rangle. \tag{3.27}$$

Suppose that the solution of the problem is the second element $|2\rangle = |10\rangle$. After the application of U_f we obtain:

$$|\psi_1\rangle = |\psi\rangle - |10\rangle\,. \tag{3.28}$$

And at the end, after the application of the diffusion transformation D we obtain:

$$|\psi_f\rangle = (2|\psi\rangle\langle\psi| - I)(|\psi\rangle - |10\rangle) = |\psi\rangle - |\psi\rangle + |10\rangle = |10\rangle\,. \tag{3.29}$$

Now we have to measure the state and we get the searched state with a probability of 100%.

3.9 The case of multiple solutions

In this section we look at our problem with t solutions of the equation $f(x) = 1$. For more details see [8].
We assume that the vale of t is known and, similarly to the case with one solution, we define an angle θ as $\sin\theta = \sqrt{\frac{t}{N}}$ for $0 < t < N$. It follows that, after m applications of step ii), we have the state:

$$|\psi_m\rangle = \sin((2m+1)\theta)|s\rangle + \cos((2m+1)\theta)|u\rangle \tag{3.30}$$

$$\text{where } |u\rangle = \sum_{f(i)=0} \frac{1}{\sqrt{N-t}}|i\rangle \tag{3.31}$$

$$\text{and } |s\rangle = \sum_{f(i)=1} \frac{1}{\sqrt{t}}|i\rangle \tag{3.32}$$

As in the case of one solution we need $m = \lfloor \pi/4\theta \rfloor$ steps to have the maximal probability to find a correct solution. Similar to (3.25) the probability of finding a correct solution after $m = \lfloor \pi/4\theta \rfloor$ steps is

$$p > 1 - \sin^2(\theta) = 1 - \frac{t}{N}, \tag{3.33}$$

and this is negligible if $t \ll N$, i.e. the number of solution is very smaller than the number of N.

3.10 Summary

We have seen how we can solve a search problem with a quantum computer using Grover's algorithm, and we have proven that that it performs better than any classical algorithm. Then we have looked at the implementation and at the workflow of the algorithm. Finally we have presented a simple example and we have discussed the geometrical interpretation to find the exact number of iterations we need to apply to have a probability of failure close to 0.

Bibliography

[1] P. Kaye, R. Laflamme, and M. Mosca, *An Introduction to Quantum Computing* (Oxford University Press, New York, 2007).

[2] M. A. Nielsen and C. I. L., *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2000).

[3] L. K. Grover, *A fast quantum mechanical algorithm for database search* (1996), `arXiv:quant-ph/9605043`.

[4] L. K. Grover, *Quantum Mechanics helps in searching for a needle haystack*, Phys. Rev. Lett. **79**, 325 (1997).

[5] C. Lavor, L. R. U. Manssur, and R. Portugal, *Grover's Algorithm: Quantum Database Search* (2003), `arXiv:quant-ph/0301079`.

[6] L. K. Grover, *Searching with Quantum Computer* (2000), `arXiv:quant-ph/0011118`.

[7] L. K. Grover, *From Schrödinger's Equation to the Quantum Search Algorithm*, Pramana **56**, 333 (2001).

[8] M. Boyer, G. Brassard, P. Hoeyer, and A. Tapp, *Tight bounds on quantum searching* (1996), `arXiv:quant-ph/9605034`.

Chapter 4

Computational Models for Quantum Computing

Pascal Steger
supervisor: Dr. Stefan Hohenegger

We will give an overview over the quantum analogues to the classical computing models, the Turing machine and the circuit model. The two basic models for quantum computing are developed: The quantum computing network or circuit, built of quantum gates, and the Quantum Turing Machine (QTM). The differences between the classical and quantum concepts are highlighted.

Connections between the two models are established by the fact that a gate is conveniently described by a QTM and every QTM can be simulated by a quantum network; we show a close relation between the resources needed by the QTM and the network simulation. The description of the dynamics using step operators T or S-matrices treats the QTM and gates in a similar way. Advantages over the classical computing models are mentioned.

4.1 Introduction

Previous contributions have introduced the basic notions for classical computing: The Turing machine as well as the circuit model emerged as classical computing models, complexity classes and their use in cryptography were outlined. We want to look at the generalization of these models to the framework of quantum mechanics. One motivation for this is that the security of several cryptographic algorithms relies on long calculation times. The latter may be shortened using

quantum computers. The search for prime factors of a big number, for instance, could be replaced with more efficient quantum versions.
We start with the circuit model in section 2.1, introducing quantum gates with examples and giving an efficient mathematical description. As we will see, one cannot perfectly imitate a classical gate using quantum gates, but an arbitrarily close approximation is possible. A constructive proof thereof follows in 2.4. The QTM will be dealt with in section 3, followed by its description with a step operator. The two models are related to each other, as we shall see in section 4, and their dynamics may be described easily with a Hamiltonian. A last part is dedicated to the possible gain in computation speed with quantum computers.

4.2 Quantum Gates

4.2.1 Terminology

The well-known classical computer performs computations using logic circuits. Let us study this in some detail, introducing some later needed terminology: A *computation* is a process that produces output depending on some input. *Input* and *output* denote abstract symbols. They are encoded in *bits* or *quanta*, which are the smallest possible quantities of non-probabilistic information. Those again are physically represented in a carrier, e.g. a transistor, or a spin 1/2-particle.
Physical processes in a quantum computer follow three steps:

1. preparation of the input states in carriers, e.g. setting the spin of electrons
2. interaction in QM elastic scattering
3. measurement of output carriers after a fixed number of steps

For most applications, one can neglect the details of step 2. It can be seen as happening in a black box. The actual scattering and projection of the state are implementation-specific and do not interfere with the theoretical model. The models should take into account, although, that errors may occur. Error correction will be the main topic of a following contribution.
What is a *logic gate*? It is a computing machine, where input and output consist of fixed number of bits. Some previously defined computation is done in a fixed time. A *quantum gate* on the other hand can have qubits as input and output. These are quantum mixtures of eigenstates of the input observable $\hat{I}$ and output observable $\hat{O}$. A *reversible gate* has the property that inputs and outputs are related by an invertible function – in the ideal case, if no errors occur. No information is deleted, therefore Landauer's principle [1] does not give an energy loss.

Reversible gates and circuits must have the same number of input and output wires. Any irreversible gate can be converted to a reversible one by repeating an appropriate number of inputs on the output side.

One can connect several gates into a circuit. The outputs of a gate after com-

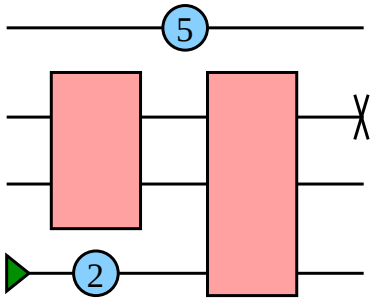


Figure 4.1: Example for a logic circuit showing different pieces.

putation step (i) can be used as inputs for another gate at step $(i+1)$, requiring that the gates are synchronized. A *logic circuit* is a computing machine consisting of logic gates, a computation is performed in fixed time. The symbols in the example circuit – see fig. 4.1 – are place holders for different parts of a general circuit: Gates are denoted by rectangles with input connections to the left and output connections to the right. Information flows from the left to the right. Sources and sinks, graphically represented by triangles and crosses, can be used to implement special input and output conditions: A *source* is has only one output that emits 0 or 1 in each step. A sink on the other hand has only one input and irreversibly deletes information. A *unit wire* propagates the carriers unchanged and computes the identity function, a fixed time dilation is indicated by a number in a circle.

4.2.2 Mathematical Descriptions

Several mathematical descriptions of gates are possible. We first have to choose a *computational basis*, which is given by the eigenstates of the input operator $\hat{I}$ and output operator $\hat{O}$ in the Schrödinger picture. They should be the same, otherwise it would be non-trivial to use an output of a gate as input for the next one; repeated gates could not be implemented easily.

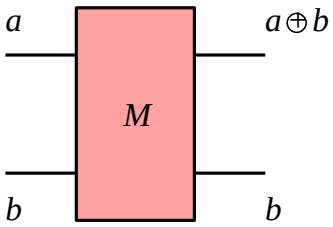


Figure 4.2: measurement gate - XOR - CNOT

Tables

In this basis one can write down the action of a gate using a table. Every combination of input eigenvalues is mapped to its output. The following example corresponds to the gate in fig. 4.2.

a	b	$a \oplus b$	b	$(a \oplus b) \oplus b$	b
0	0	0	0	0	0
0	1	1	1	0	1
1	0	1	0	1	0
1	1	0	1	1	1

(4.1)

The classical gate is called XOR since it computes the logical XOR function from inputs a and b, copying a as a second output to guarantee reversibility. The action of the quantum gate can be seen as inverting the b input if a is set and returning b unchanged otherwise. Therefore it is referred to as CNOT – controlled NOT – or measurement gate.

Permutations

Another representation of the same description is possible using permutations: let $\{|a, b\rangle\}$, $a, b \in \{0, 1\}$ be the four computational basis states for a system with two inputs and outputs. A gate maps each input state to an output state, for the measurement gate this reads as

$$\begin{aligned} |0,0\rangle &\rightarrow |0,0\rangle \\ |0,1\rangle &\rightarrow |1,1\rangle \\ |1,0\rangle &\rightarrow |1,0\rangle \\ |1,1\rangle &\rightarrow |0,1\rangle. \end{aligned} \tag{4.2}$$

S-matrix

A third way is given by using an S-Matrix, which is most suitable for quantum gates, as was shown by Deutsch [2]. In principle, it encodes the mapping from

above with a unitary matrix, thus using linear algebra.
Let us consider the measurement gate. Its S-matrix is given by

$$S^{ab}_{a'b'} = \delta^{a\oplus b}_{a'}\delta^{b}_{b'}: \qquad \begin{pmatrix}|0,0\rangle\\|0,1\rangle\\|1,0\rangle\\|1,1\rangle\end{pmatrix} = \begin{pmatrix}1&0&0&0\\0&0&0&1\\0&0&1&0\\0&1&0&0\end{pmatrix}\begin{pmatrix}|0,0\rangle\\|0,1\rangle\\|1,0\rangle\\|1,1\rangle\end{pmatrix}. \tag{4.3}$$

This unitary matrix $S^{ab}_{a'b'}$ has clumped indices ab, $a'b'$ denoting eigenstates of the input and output carriers. $a, b \in \{|0\rangle, |1\rangle\}$ results in four possible combinations, the same holds for the two outputs a', b'. We can interprete the clumped indices as binary representations of a natural number, as for instance $ab = 10\text{b} = 2$. In this interpretation, they give the position of the corresponding matrix element in S. As an example, we have for the element in the first row and the second column

$$a = 0,\ b = 0,\ a' = 0,\ b' = 1: \quad S^{a'b'}_{ab} = \delta^{0\oplus 0}_{0}\delta^{0}_{1} = 0. \tag{4.4}$$

The operation of a general gate with n in- and outputs corresponds to a matrix multiplication with $S^{ab\ldots}_{a'b'\ldots}$,

$$|a, b, \ldots\rangle \to \sum_{a',b',\ldots\in\{0,1\}} S^{ab\ldots}_{a'b'\ldots}|a', b', \ldots\rangle \equiv S|a, b, \ldots\rangle. \tag{4.5}$$

If no basis is chosen explicitly, S denotes the linear operator of a gate. Unitarity of S is necessary to describe a reversible gate. Repeated gates are represented by powers of S, for instance, the unit wire with time dilation n can be interpreted as n individual unit wires with a constant time dilation. Its action is $\mathbb{1}^n = \mathbb{1}$.

4.2.3 EXAMPLES OF QUANTUM GATES

EXAMPLE: NOT

For an more complicated example of the action of the S-matrix, consider the NOT gate in fig. 4.3. It is described by

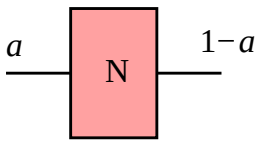


Figure 4.3: NOT gate

$$\begin{array}{c|c} a & \neg a \\ \hline 0 & 1 \\ 1 & 0 \end{array}, \qquad \begin{aligned} |0\rangle &\to |1\rangle \\ |1\rangle &\to |0\rangle \end{aligned}, \qquad S_N = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \tag{4.6}$$

as one can check by multiplication with $(0,1)^T$ or $(1,0)^T$, which are vectors denoting the states $|1\rangle$ and $|0\rangle$ in the computational basis $\{|0\rangle, |1\rangle\}$. Powers of S correspond to several successive copies of NOT, $\alpha \in \mathbb{N}$ implies that N^α is a logic gate, either the identity (α even) or the NOT gate (α odd). Nevertheless, a non-integer power $\alpha \notin \mathbb{N}$ of the operator N is perfectly well defined as well: N^α does then not describe the action of a logic gate anymore, but the action of a *quantum gate*. We can compute the S-matrix

$$S_{N^\alpha} = S_N^\alpha = \frac{1}{2}\begin{pmatrix} 1+e^{i\pi\alpha} & 1-e^{i\pi\alpha} \\ 1-e^{i\pi\alpha} & 1+e^{i\pi\alpha} \end{pmatrix}. \tag{4.7}$$

Here we see that the use of states as in-/outputs instead of simple logic 0 or 1 is justified. A gate then transforms these states into another, usually visualized as rotation on the Bloch sphere [3].

Toffoli and Q

The Toffoli gate from classical computing (see fig. 4.4) has an analogue in quantum computing. Its classical version can be described by the S-matrix

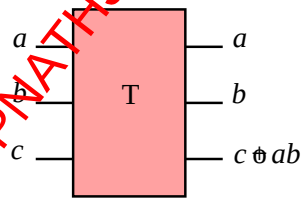


Figure 4.4: Toffoli gate

$$S^{abc}_{Ta'b'c'} = \delta^a_{a'}\delta^b_{b'}[(1-ab)\delta^c_{c'} + ab(S_N)^c_{c'}]. \tag{4.8}$$

Analogously, the quantum gate Q reads as

$$S^{abc}_{Qa'b'c'} = \delta^a_{a'}\delta^b_{b'}\left[(1-ab)\delta^c_{c'} + iabe^{-i\pi\alpha/2}(S_N^\alpha)^c_{c'}\right]. \tag{4.9}$$

and boils down to the classical form if $\alpha \in \mathbb{N}$.
In order to calculate individual matrix elements, we choose the basis $0 = |000\rangle$,

$1 = |001\rangle$, ..., $6 = |110\rangle$, $7 = |111\rangle$. The S-matrices are then computed by plugging in the different values for $abc, a'b'c'$,

$$S_T = \begin{pmatrix} \mathbb{1}_6 & & \\ & 0 & 1 \\ & 1 & 0 \end{pmatrix}, \tag{4.10}$$

$$S_Q = \begin{pmatrix} \mathbb{1}_6 & & \\ & i\cos\pi\alpha/2 & \sin\pi\alpha/2 \\ & \sin\pi\alpha/2 & i\cos\pi\alpha/2 \end{pmatrix}. \tag{4.11}$$

$\mathbb{1}_6$ denotes a 6×6-identity matrix. The classical S_T-matrix can be checked by looking at the tabular description.

4.2.4 EQUIVALENCE

A question that may arise is whether it is possible to simulate a classical logic gate by using a set of quantum gates: Consider for example the repeated use of a NOT-gate,

$$\begin{aligned} S_{N^2} &= S_N^2 = \mathbb{1}, \\ (S_{N^\alpha})^m &= S_N^{m\alpha} = S_N^{m\alpha - 2\lfloor m\alpha/2 \rfloor}. \end{aligned} \tag{4.12}$$

The exponent $m\alpha - 1\lfloor m\alpha/2 \rfloor \equiv 1+\varepsilon$ can be made arbitrarily close to 1, but never exactly for $m \in \mathbb{N}$ and an irrational α. A failure of the simulation is possible, namely if another result than the classically expected one shows up. The time before non-classical behavior is given by the reciprocal of the expectation value for the wrong result,

$$t = \frac{1}{\max_{|\Psi\rangle}(1 - |\langle\Psi|S_N^\dagger S_{N^\alpha}^m|\Psi\rangle|^2)} = \frac{1}{\sin^2 \pi\varepsilon/2} \sim \varepsilon^{-2} \overset{(\varepsilon\to 0)}{\to} \infty. \tag{4.13}$$

Two circuits are called *computationally equivalent*, if they yield the same output given the same input. Exact equivalence is not possible with quantum gates[1]. One needs to introduce another notion: F and G are *adequate sets* of gates, if there exists a series $\{g_n \in G\}$ for all $f \in F$ and a sequence $\{\phi_n\}$ of phase angles such that

$$\lim_{n\to\infty} S_{g_n} e^{i\phi_n} = S_f. \tag{4.14}$$

As an example, $F = \{N\}$ and $G = \{N^\alpha, \mathbb{1}\}$ are adequate.
One now wants to find a *universal gate*, that is a quantum gate such that the set

[1] It is not even given in the case of a probabilistic Turing machine. A redefinition of equivalence with fixed probabilities would solve this problem.

of unit wire, source and this gate is adequate to the set of all possible gates. The Toffoli gate is universal for classical gates. We will see that the Q-gate plays the same role for quantum gates:

Claim:

The Q-gate is a universal gate.

Proof:

A constructive proof is striven for; we create a repertoire of gates that Q is adequate to:

1. Toffoli gate
2. all logic gates
3. all 3-bit quantum gates
4. all n-bit quantum gates
5. all quantum gates

This procedure was proposed by Deutsch [2].

Step 1 and 2: Toffoli gate

We want to calculate powers of S_Q. The basis should be $0 = |000\rangle$, $1 = |001\rangle$, ..., $6 = |110\rangle$, $7 = |111\rangle$. The $(4n+1)$-th power of S_Q in matrix form is

$$S_Q^{4n+1} = \begin{pmatrix} \mathbb{1}_6 & & \\ & i\cos\pi\alpha(2n+1/2) & \sin\pi\alpha(2n+1/2) \\ & \sin\pi\alpha(2n+1/2) & i\cos\pi\alpha(2n+1/2) \end{pmatrix}. \tag{4.15}$$

S_Q^{4n+1} equals S_T for $\alpha(2n+1/2) = (2m+1/2)$, for some $m \in \mathbb{N}$. For the same reason that powers of QM NOT are adequate to the logical NOT – there exists an arbitrarily close approximation – the Toffoli gate is in the repertoire. Moreover, the Toffoli gate is universal for all logic gates, so Q is also adequate to the set of all logic gates.

STEP 3: 3-BIT QUANTUM GATES

Consider now powers of Q of the form $4n$ with $n \in \mathbb{N}$:

$$\begin{aligned} S_Q^{4n} &= \begin{pmatrix} \mathbb{1}_6 & & \\ & \cos 2n\pi\alpha & -i\sin 2n\pi\alpha \\ & -i\sin 2n\pi\alpha & \cos 2n\pi\alpha \end{pmatrix} = \\ U_\lambda &\equiv \begin{pmatrix} \mathbb{1}_6 & & \\ & \cos\lambda & i\sin\lambda \\ & i\sin\lambda & \cos\lambda \end{pmatrix}. \end{aligned} \tag{4.16}$$

These are in the repertoire, since there exists $m \in \mathbb{N}$ such that $|2\pi n\alpha - 2\pi m| < \varepsilon$ for ε arbitrarily small.
Permutations describe logic gates (e.g. P_{57}, it permutes qubits 5 and 7), and belong therefore to the repertoire; the limit of combinations of permutations and U_λ as well:

$$\begin{aligned} V_\lambda &\equiv \lim_{n\to\infty}[P_{56}(U_{\sqrt{\lambda/n}}P_{57})^2(U_{-\sqrt{\lambda/n}}P_{57})^2P_{56}]^n = \begin{pmatrix} \mathbb{1}_6 & & \\ & \cos\lambda & \sin\lambda \\ & -\sin\lambda & \cos\lambda \end{pmatrix}, \\ W_\lambda &\equiv \lim_{n\to\infty}[U_{\sqrt{\lambda/2n}}V_{\sqrt{\lambda/2n}}U_{-\sqrt{\lambda/2n}}V_{-\sqrt{\lambda/2n}}]^n = \mathrm{diag}(1,\ldots,1,e^{-i\lambda},e^{i\lambda}). \end{aligned}$$

A change in the global phase factor does not change the expectation value of an observable, so

$$X_\lambda \equiv \mathrm{diag}(1,\ldots,1,e^{i\lambda}) \tag{4.17}$$

describes a gate that Q is adequate to. Until now, we know that $V_\lambda, W_\lambda, X_\lambda$ are all in the repertoire. We can then construct a gate that maps the sixth qubit of a general input vector $|\Psi\rangle$ to zero, and puts together the two prefactors of qubit 6 and 7:

$$\begin{aligned} |\Psi\rangle &= \sum_{n=0}^{7} c_n|n\rangle, \quad \sum_{n=0}^{7}|c_n|^2 = 1 \\ Z_6[|\Psi\rangle] &:= X_{-\arg(c_6c_7)/2}V_{-\arctan|c_6/c_7|}W_{-\arg(c_7/c_6)/2} \\ |\Psi\rangle &\Rightarrow \sum_{n=0}^{5} c_n|n\rangle + 0 + \sqrt{|c_6|^2+|c_7|^2}|7\rangle. \end{aligned} \tag{4.18}$$

The gate Z_6 is in the repertoire, since it is a combination of gates that Q is adequate to. It follows by analogy that the same map for another qubit $i < 7$, namely $G: c_i \to 0$ – and the gate it is describing – is also in the repertoire. One

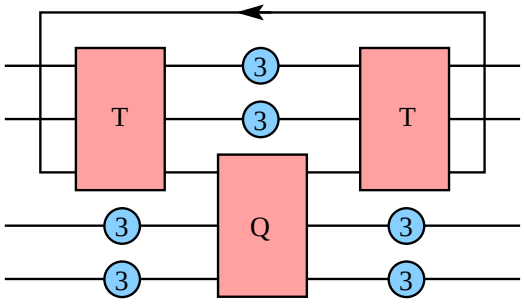


Figure 4.5: General gate with four qubits.

can now construct a gate that evolves all coefficients from $|0\rangle, \ldots, |6\rangle$ to zero and the one from $|7\rangle$ to 1:

$$\begin{aligned} S_{G[|\psi\rangle]} &= \sum_{n=0}^{7} e^{i\sigma_n} |\Psi_n\rangle\langle\Psi_n|; \\ S &= \prod_{n=0}^{7} S_{G^{-1}[|\Psi_n\rangle]} X_{\sigma_n} S_{G[|\Psi_n\rangle]}. \end{aligned} \tag{4.19}$$

This last S describes the general action of a three-bit gate and is manifestly in the repertoire. So Q is universal with relation to the set of all 3×3-matrices.

Step 4 & 5: n bit gates

Look at a possible general four-bit gate as in fig. 4.5. The loop-back is necessary to connect all inputs and outputs[2]. Its input is initialized to 0. By plugging in all 2^4 different inputs it can be verified that the output of the loop-back is always 0. The action of this gate yields

$$\begin{aligned} |a,b,0,c,d\rangle &\Rightarrow |a,b,ab,c,d\rangle \\ &\Rightarrow [1+abc(i\cos\pi\alpha/2-1)]|a,b,ab,c,d\rangle \\ &\qquad +[abc\sin\pi\alpha/2]|a,b,ab,c,1-d\rangle \\ &\Rightarrow [1+abc(i\cos\pi\alpha/2)]|a,b,0,c,d\rangle + [abc\sin\pi\alpha/2]|a,b,0,c,1-d\rangle. \end{aligned} \tag{4.20}$$

Its S-matrix, evaluated for the three gates, reads as

$$S^{abcd}_{Q_4 a' b' c' d'} = \delta^a_{a'}\delta^b_{b'}\delta^c_{c'}\left[(1-abc)\delta^d_{d'} + iabce^{-i\pi\alpha/2}(S^\alpha_N)^d_{d'}\right]. \tag{4.21}$$

One can use the same procedure to get $5, 6, \ldots, n$-bit gates. Therefore the n-bit gates are also in the repertoire.

[2]This loop-back makes the circuit reversible, the use of a source and a sink would yield irreversible gates.

4.3 Quantum Turing Machine (QTM)

4.3.1 Connection to the Classical Turing Machine

The second model for quantum computing is given by the Quantum Turing Machine, described in detail by Benioff [4]. In direct analogy to the classical one-tape Turing Machine, a one-tape QTM consists of

- an infinite memory,
- a finite processor,
- a state control and
- a program.

The head is in state $|l\rangle$ at position j of the tape and the state of the qubit at site j is denoted by $|s_j\rangle$, see fig. 4.6. The computation proceeds in steps of fixed duration Δt. During a step only the processor and a finite part of memory interact. The QTM *halts*, if two subsequent states are identical or if the halt flag is set. The *halt flag* is an observable with spectrum $\{0, 1\}$, its state should be measurable without disturbing the state of the QTM. The QTM is universal, it can simulate any other quantum computer.

The main difference between the classical TM and the QTM is the fact that a

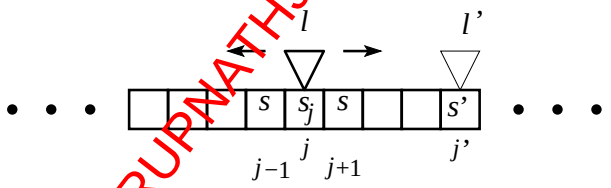


Figure 4.6: Sketch of a one-tape QTM. Program and state control are part of the head.

QTM acts with the quantum state of its head on quantum states on the tape instead of logic states. Any superposition is allowed at a given lattice site, therefore much more information than only the boolean 0 or 1 can be stored.

4.3.2 Church-Turing hypothesis

Does the existence of quantum superpositions on the tape affect some of the most fundamental statements on computing models, e.g. the Church-Turing principle? Not at all: The Church-Turing hypothesis states that

> *Every function which would naturally be regarded as computable can be computed by the universal Turing machine.*

Expressed as a physical principle, this reads as:

> *Every finitely realizable physical system can be perfectly simulated by a universal computing machine operating by finite means.*

The QTM fulfills this principle, and the original hypothesis as well. A classical Turing machine does *not* fulfill the second version, since it is finite, but continuous systems may be described with a finite number of parameters only.

4.3.3 Step Operator

Since the QTM operates in steps of finite time, it is a good idea to define a unitary *step operator* T, which must fulfill the following requirements: It should describe the interaction of the head with the tape only at one position per time interval. The head can move to the left, to the right, or stay and interact. It must be local and may describe a displacement in at most one direction. Moreover, the periodicity of the lattice sites must be taken into account. Mathematically the last three requirements are expressed in eq. 4.22. The operator $\tilde{T}$ describes that part of the step operator T involved in the interaction of the head with a single lattice qubit. $P_j = |j\rangle\langle j|$ is a projection operator for the head onto the lattice site j. Δ stands for mere displacements – to the left, not at all, to the right. It can take the respective values $-1, 0, 1$.

$$\begin{aligned} \langle l', j', s'|T|l, j, s\rangle &= \langle s'_{\neq j}|s_{\neq j}\rangle\langle l', j', s'_{j'}|\tilde{T}|l, j, s_j\rangle, \\ \tilde{T} &= \sum_{j=-\infty}^{\infty}\sum_{\Delta=-1}^{1} P_{j+\Delta}\tilde{T}P_j, \\ \langle l', j'+\Delta, s'|\tilde{T}|l, j', s\rangle &= \langle l', j+\Delta, s'|\tilde{T}|l, j, s\rangle. \end{aligned} \tag{4.22}$$

The first equation states that a change in a single step can only interact with the qubit at the position of the head. All other contributions are excluded by $\langle s'_{\neq j}|s_{\neq j}\rangle = 0$. The second equation expresses the motion in only one direction, $\tilde{T} \neq 0$ only if the motion takes the head from position j to position $j + \Delta$, such that $P_{j+\Delta}$ is not orthogonal to $\tilde{T}P_j$. The last equation shows that the matrix elements of interaction are the same for all different j and j', which is only possible if the lattice sites are periodic.

4.4 Connections

The computing models QTM and quantum circuits can be transformed into each other. One can get quantitative statements for the efficiency of the respective simulation from complexity theory. This theory is needed to determine the cost of a computation in general, i.e. the resources time, memory space and energy. What measures do exist?
The *size* of a circuit gives the number of elementary gates in a quantum circuit. The *depth* is the maximal length of a directed path from in- to output register. An *interacting pair of quantum circuits* describes a partition of the circuit with disjoint sets of inputs such that all outputs are located on one side. The *communication cost* gives then the number of wires between interacting pairs. See fig. 4.7 for an example. These measures can be used to characterize a quantum circuit, especially when optimization is considered. If one wants a fast computer, the depth and the communication cost should be small – information propagates with the speed of light at most, so fewer and shorter wires mean faster computation. If the circuit is optimized for small space, its size must be minimized. The

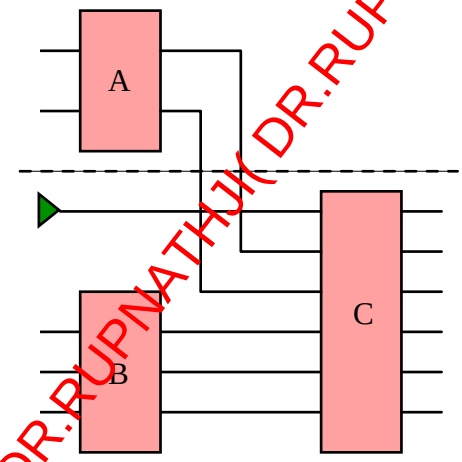


Figure 4.7: Example for a reversible interacting pair of quantum circuits. Its size is 3, the depth is also 3, the communication cost equals 2.

simulation of a given QTM by a quantum circuit could be polynomially bounded or increase exponentially. In order to distinguish between these two cases, we introduce the (n, t)-simulation: A quantum circuit C (n, t)-*simulates* the QTM M, if the input $\tilde{x} \in \{0, 1\}^n$ evolved by C is the same as the state of M after t steps, provided that the first n qubits of the QTM are assigned the same input values as the circuit.

4.4.1 Theorems by Yao

Yao [5] gives, between others, following theorems that relate the QTM and implementations on a quantum circuit:

1. Any unitary operator $U \in \mathbb{C}^{2^n}$ can be simulated by a quantum network using $2^{\mathcal{O}(n)}$ 3-gates, with $\mathcal{O}(n)$ wires.
2. Every QTM can be (n, t)-simulated by a quantum network of size $poly(n, t)$.
3. There exists a universal QTM that can simulate any other QTM with only polynomial slowdown.

The proofs are shown in the article by Yao, and go beyond the scope of this introduction.

4.4.2 Dynamics

The dynamics of a QTM or a quantum circuit can be described by their Hamiltonian. For a single gate with a given S-matrix, the Hamiltonian can be constructed in the following way, as Deutsch [2] outlined:

$$H \equiv \frac{i}{t} \ln S, \tag{4.23}$$

where the logarithm of the gate's S-matrix evaluated by a Taylor series. The Hamiltonian H for a quantum circuit can then be constructed by connecting single-gate Hamiltonians.
Feynman [6] proposed to use the step operator T to get the Hamiltonian for general QTMs,

$$H \equiv K(2 - T - T^\dagger). \tag{4.24}$$

This expression gives the kinetic energy, if T is a simple displacement without interaction. Here another connection between the two basic models shows up: T by itself can be a sum of elementary unitary step operators for QTMs describing single gates, and therefore T incorporates all information of the evolution of all the circuit, as if it were a QTM.

4.4.3 QTM vs. TM: computation speed

A QTM is not faster than a classical Turing machine on average if it is using the same algorithms. Deutsch [7] for instance considered a computer that needs one day to predict the stock market of tomorrow. If its program was implemented

within a quantum computer, it would be possible to stop the computation after half a day and get the result already, but there is a probability of 50% that no result shows up at all. One could imagine that this quantum computer takes advantage of "parallel universes" to instantiate copies of the QTM and return the result in a shorter time $t = pt_0, p < 1$, but with a failure probability of $1 - p$. However, there is an average huge speed-up, if specialized algorithms like the ones of Deutsch-Josza [8] and Grover [9] are considered. These do not destroy the superpositions of states during calculation and project only at the end of the computation. See later contributions for more details.

4.5 Summary

In this outline we have considered three models for quantum computing:
Quantum gates take qubits as input, let them interact in a QM elastic scattering and measure them at the end. The direct way from in- to output is best described using an S-matrix, which is unitary for reversible gates. The repeated application of NOT-gates motivates the use of quantum gates. Exact equivalence of quantum gates with relation to logic gates is not possible, but approximation is. The Q-gate is universal to the set of all quantum gates as the Toffoli is for all logic gates. Different combinations of powers of Q, permutations and phase factor changes are used for the proof.
The QTM uses quantum states on lattice sites and in the head, but the rest of its components are equivalent to the classical Turing machine. It fulfills the Church-Turing principle. The description with a step operator reflects that the QTM performs computations in several steps. Complexity theory characterizes quantum computers by their use of space, memory, and time. QTM and quantum circuit can simulate each other with only polynomially increased need in memory and time. Hamiltonians for the descriptions of the dynamics make use of the S-matrix and step operator T. A quantum computer can reduce the time needed for special algorithms, but for classical algorithms it takes the same time on average.

Acknowledgments

I owe Dr. St. Hohenegger a big "thank you" for his helpful advice on how to present this most interesting topic in a stringent and reader-friendly way. I would also like to thank Prof. Dr. H. Katzgraber and Prof. Dr. R. Renner for creating a pleasant atmosphere at the proseminar.

Bibliography

[1] R. Landauer, *Uncertainty Principle and Minimal Energy Dissipation in the Computer*, Int. J. Theor. Phys. **21**, 283 (1982).

[2] D. Deutsch, *Quantum computation networks*, Proc. R. Soc. Lond. A **425**, 73 (1989).

[3] I. Chuang, *Quantum physics and Church-Turing* (http://http.cs.berkeley.edu/∼vazirani/f97qcom/, 1997).

[4] P. Benioff, *Models of quantum Turing machines*, Fortsch. Phys. **46**, 423 (2007).

[5] A. C. Yao, *Quantum Circuit Complexity*, Proc. of the 34th Ann. Symp. on Found. of Comp. Sc. (FOCS) p. 352 (1993).

[6] R. P. Feynman, *Quantum mechanical computers*, Opt. News **11**, 11 (1985).

[7] D. Deutsch, *Quantum theory, the Church-Turing principle and the universal quantum computer*, Proc. R. Soc. Lond. A **400**, 97 (1985).

[8] D. Deutsch and R. Jozsa, *Rapid solutions of problems by quantum computation*, Proc. R. Soc. London A **439**, 553 (1992).

[9] L. Grover, *Quantum mechanics helps in searching for a needle in a haystack*, Phys. Rev. Lett. **79**, 325 (1997).

Chapter 5

The Deutsch-Jozsa algorithm

Thomas Strub
supervisor: Andrey Lebedev

Quantum algorithms use quantum mechanical properties like superposition and interference to increase the efficiency of certain computations. Superposition helps to compute different computational paths in parallel, whereas interference brings the different paths together. This means that their probability amplitudes can interfere, something which is not possible in a classical algorithm. The Deutsch-Jozsa algorithm presented in this chapter is one of the earliest quantum algorithm that could really show the higher performance of quantum computations.

5.1 Introduction

The question, if quantum computers can solve certain problems more efficiently than classical algorithms (deterministic or probabilistic), was answered in 1992 when the Deutsch-Jozsa algorithm was publishes by David Deutsch and Richard Jozsa. The algorithm published a few years earlier in 1985 by David Deutsch only, the so called Deutsch algorithm, was originally not deterministic and could therefore not improve the computation on average. Nevertheless, it showed the power quantum computers might have. In 1996 both, the Deutsch and the Deutsch-Jozsa algorithm were improved by R. Cleve et al. in the paper "Quantum Algorithms Revisited" which made the Deutsch algorithm also deterministic and faster than any classical algorithm. The reason for this gain in efficiency is to exploit quantum mechanical properties like superposition states, entanglement and

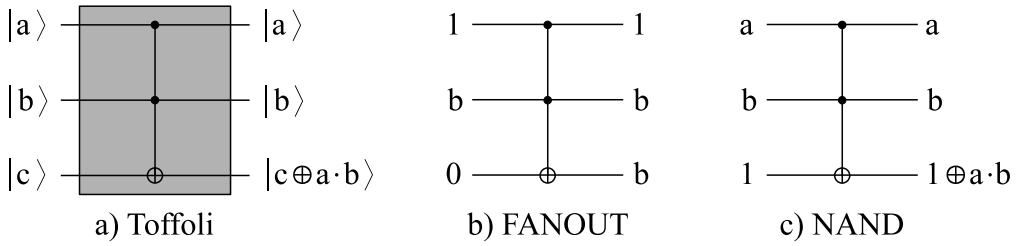


Figure 5.1: *The Toffoli gate.* Mapping $|a\rangle\,|b\rangle\,|c\rangle$ to $|a\rangle\,|b\rangle\,|c \oplus a \cdot b\rangle$ the Toffoli gate can simulate the classical FANOUT and NAND gate which form an universal set of gates.

interference. This chapter will discuss the two algorithms in the revised version and will point out their connection with the multi-particle interference.

5.2 Classical computation on quantum computers

It is not very surprising that every classical algorithm can be simulated on a quantum computer. This can be done using the reversible and universal Toffoli gate invented by Tommaso Toffoli in 1980 [1]. The Toffoli gate as seen in Fig. 5.1 uses three inputs (a, b, c) which are mapped to (a, b, $c \oplus a \cdot b$). Here $\oplus$ denotes the addition modulo two. This means that the target bit c is flipped if and only if $a = b = 1$. Thus, the Toffoli gate can be seen as an controlled-controlled-NOT gate, where a and b are the control bits. If the target qubit $|c\rangle$ is initially set to $|1\rangle$ its output will be $|1 \oplus a \cdot b\rangle$, which will simulate the exact output of a classical NAND gate with the inputs a and b. If the inputs are otherwise set to the states $|a\rangle = |1\rangle$ and $|c\rangle = |0\rangle$, the final state of the target qubit will be $|0 \oplus 1 \cdot b\rangle = |b\rangle$. That means that one gets a copy of the second control qubit which corresponds to the classical FANOUT gate. Notice that, if a classical gate is simulated the only possible inputs and outputs are the computational basis states $|0\rangle$ and $|1\rangle$. Therefore, one will not get into any trouble with the no-cloning theorem, when the classical FANOUT is simulated. It is easy to show that the NAND gate and the FANOUT gate form an universal set of gates which make the Toffoli gate universal too.

5.3 Quantum parallelism and interference

This section will show how quantum mechanical properties can improve computation. One might actually ask what the difference is between quantum algorithms and classical probabilistic algorithms, seeing that the outcome of a measurement of a quantum superposition state is probabilistic as well. The answer is quite easy: while classical after each step of a computation a certain basis state will be taken, in a quantum algorithm a qubit can stay in a superposition state until a measurement takes place. This means that in the classical case the *path* which was taken during the calculation is a posteriori known. On the other hand, in a quantum computation different paths are taken at the same time, whose probability amplitude can *interfere* before a measurement is made. Quantum computation can therefore be viewed as a generalization of probabilistic computation [2].

5.3.1 Quantum parallelism

Quantum parallelism uses the fact that qubits can be in superposition of the two computational basis states $|0\rangle$ and $|1\rangle$ while a classical bit is *either* set to 0 *or* 1. Now suppose there is a quantum gate $U(f)$ which acts on two qubits $|x\rangle$ and $|y\rangle$ in the following way: $|x\rangle|y\rangle$ is mapped to $|x\rangle|y \oplus f(x)\rangle$, where f is a one-bit boolean function $f : \{0,1\} \to \{0,1\}$. This is nothing more than the controlled-NOT gate, except that the target bit y is flipped if $f(x) = 1$, rather than if $x = 1$. Thus, $U(f)$ is called f-controlled-NOT, or short f-cNOT. Let now $|x\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|y\rangle = |0\rangle$ be the initial states, where the normalization factors will mostly be omitted in the following. Applying $U(f)$ gives

$$(|0\rangle + |1\rangle)|0\rangle \xrightarrow{U(f)} |0\rangle|f(0)\rangle + |1\rangle|f(1)\rangle\,. \tag{5.1}$$

This is a remarkable entangled state, as it contains all information about f, namely both $f(0)$ and $f(1)$, but in only one action of $U(f)$. If the first qubit $|x\rangle$ is replaced by a quantum register $|\mathbf{x}\rangle$, which is an ensemble of n single qubits $|x_i\rangle_{1\le i\le n}$, $U(f)$ would then represent a binary function $f : \{0,1\}^n \to \{0,1\}$. As before the initial state of the control register is set to an equally weighted superposition state, but this time over 2^n basis states. The action of $U(f)$ then leads to

$$|\mathbf{x}\rangle|0\rangle = \sum_{z=0}^{2^n-1} |z\rangle|0\rangle \xrightarrow{U(f)} \sum_{z=0}^{2^n-1} |z\rangle|f(z)\rangle\,. \tag{5.2}$$

These are 2^n "evaluations of f" but in only one action of $U(f)$, why it is called *quantum parallelism*. It seems clear that the information encoded in the final state

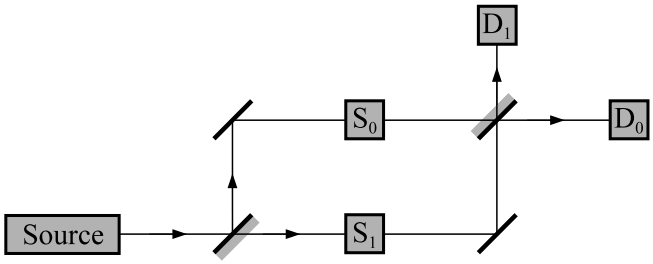


Figure 5.2: *The Mach-Zehnder interferometer.* A photon entering the interferometer is in a superposition state of the two possible paths. The phase shifters S_i shift the phase of the passing photon. When the two paths are joined together the probability amplitudes will interfere and the photon is detected in one of the detectors D_i.

of Eq.5.2 can not be used in its full size: as soon as a measurement of the control register in the computational basis is made the wave function collapses into an unpredictable state $|z^*\rangle$. The target qubit will then be in the state $|f(z^*)\rangle$ due to the entanglement and all the other information about f will be lost. In order to profit from such a entangled state, and therefore form quantum parallelism, interference can be used which will be discussed in the next section.

5.3.2 Interference and phase kick-back

Quantum algorithms can be seen as multi-particle interferometers, where the operations of quantum gates are represented in shifting phases. To understand this, look at a Mach-Zehnder interferometer (Fig. 5.2). A photon passing thought the first half-silvered mirror will with some probability amplitude, say each $1/2$, propagate along the to different paths. Along both paths are phase shifters S_0 and S_1, respectively, which shift the phase of a passing photon by ϕ_0 and ϕ_1. The two paths are then joined together with a second half-silvered mirror, which directs the photon to one of the two detectors. If now the different paths are labelled with state $|0\rangle$ and state $|1\rangle$ the photon is, after passing the phase shifters, in a superposition state $e^{i\phi_0}|0\rangle + e^{i\phi_1}|1\rangle$ which is the same as $|0\rangle + e^{i(\phi_1-\phi_0)}|1\rangle$: the global phase factor can be omitted because it has no physical meaning (i.e. it will not influence the result of a measurement). The only relevant information is therefore the relative phase difference $\phi = \phi_1 - \phi_0$. Finally, the photon is detected with the probability $\frac{1}{2}(1+\cos\phi)$ in D_0 and with the probability $\frac{1}{2}(1-\cos\phi)$ in D_1. It is important, that the measurement, namely the detection of the photon, is not made until the two paths are recombined, i.e. that they had a chance to interfere.

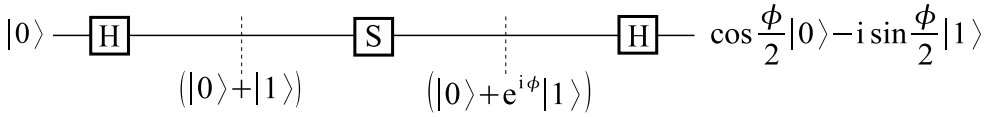


Figure 5.3: *Quantum Network of the Mach-Zehnder interferometer.* Hadamard gates replace the half-silvered mirrors and the action of the two phase shifters is represented by S.

In terms of quantum networks the Mach-Zehnder interferometer looks like Fig. 5.3. The half-silvered mirrors are identified with the single-qubit Hadamard gate H defined as

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \qquad \text{and} \qquad H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\,. \tag{5.3}$$

The Hadamard transform is actually the one dimensional case of the quantum Fourier transform defined as $\mathcal{F}\{|j\rangle\} = \frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1} e^{2\pi ijk/2^n}|k\rangle$. The action of the phase shifter S can be viewed as a single-qubit gate, as discussed above (i.e. $S|0\rangle = |0\rangle$, $S|1\rangle = e^{i\phi}|1\rangle$). Beginning with the initial state $|0\rangle$, the first Hadamard gate will produce a superposition state which then gets phase shifted by S. A second Hadamard gate brings the two computational paths together, so that either $|0\rangle$ or $|1\rangle$ with the accordant probabilities will be measured.
The control of the behavior of S (i.e. shift by ϕ or do nothing) can be realized by a f-controlled gate f-cU. According to the f-cNOT gate it is often declare that the phase is shifted if (and only if) $f(\mathbf{x}) = 1$, where $|\mathbf{x}\rangle$ therefore acts as a control register. The quantum circuit for the f-controlled shifting is shown in Fig. 5.4. The target qubit $|u\rangle$ is chosen to be an eigenstate of $U_f(\mathbf{x})$, i.e. $U_f(\mathbf{x})|u\rangle = e^{i\phi(\mathbf{x})}|u\rangle$, and is called the auxiliary qubit. It remains unchanged along the network but its eigenvalue that depends on $\mathbf{x}$ is placed ("kicked back") in front of the control register. This is called the phase kick-back.
In the example of the Mach-Zehnder interferometer the eigenvalue of U_f would

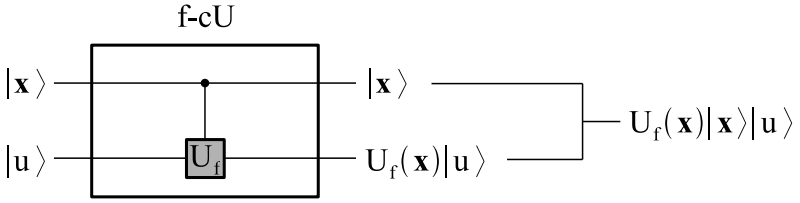


Figure 5.4: *Phase kick-back.* The eigenvalue of the auxiliary qubit $|u\rangle$, which is chosen to be an eigenstate of U_f, is placed in front of the control register $|\mathbf{x}\rangle$.

be 1 if $x = 0$ and $e^{i\phi}$ if $x = 1$. Therefore, the function f is equal to the identity function id and the algorithm would look like this:

$$\begin{aligned} |0\rangle |u\rangle \xrightarrow{H} \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) |u\rangle \xrightarrow{id-cU} \frac{1}{\sqrt{2}} \left(|0\rangle + e^{i\phi} |1\rangle\right) |u\rangle \\ \xrightarrow{H} \left(\cos\frac{\phi}{2} |0\rangle - i \sin\frac{\phi}{2} |1\rangle\right) |u\rangle . \end{aligned} \tag{5.4}$$

It is now also clear what was meant by multi-particle interference at the beginning of this chapter: the different qubits can not be viewed as separated objects, since the action of the U_f on the auxiliary qubit $|u\rangle$ has a direct influence on the control register $|\mathbf{x}\rangle$.

5.4 The Deutsch algorithm

It is now only natural to ask how the introduced devices can be used to solve an interesting computational task. The problem proposed by David Deutsch is strongly connected to the Mach-Zehnder interferometer as illustrated at the end of this section. Consider first the four possible boolean functions $f : \{0, 1\} \to \{0, 1\}$. Two of them are constant (i.e. $f(0) = f(1)$) while the other two are balanced (i.e. $f(0) \neq f(1)$). Deutsch's problem is now to deduce from the result of a single evaluation of a boolean function f whether it is constant or balanced. It is obvious that in the classical case this task is impossible to solve with just one query of f, because one has to know both $f(0)$ and $f(1)$. The goal of this section is to solve this problem with a single evaluation of f using a quantum algorithm, that is, with one single action of the accordant quantum gate. Notice, that one is only interested in a *global* property of f (constant or balanced), but not in the single values of the function.

To evaluate f in terms of a quantum gate the f-controlled gate introduced in the previous section can be used. The already discussed f-cNOT gate mapping $|x\rangle |y\rangle$ to $|x\rangle |y \oplus f(x)\rangle$, which is a special case of the f-cU gate with $\phi = \pi$, will actually suffice for this purpose. If only the computational basis states $|0\rangle$ or $|1\rangle$ are allowed for the initial states then there will be clearly no advantage over a classical calculation. However, if quantum mechanical superposition states are chosen in the same way as above, only one query of f will be needed.

The quantum circuit for the Deutsch algorithm is shown in Fig. 5.5. Consider first the initial state of the auxiliary (target) qubit set to the superposition state $(|0\rangle - |1\rangle)$. Then the action of the f-cNOT gate gives

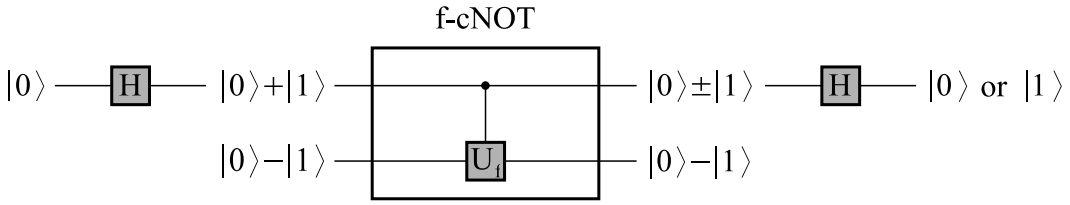


Figure 5.5: *Quantum Network of the Deutsch algorithm.* The target qubit of the f-cNOT is only used as an auxiliary qubit, since it remains unchanged. But its eigenvalue depending on $f(x)$ is kicked back to the control qubit. (global phases and normalization are omitted)

$$|x\rangle\left(|0\rangle-|1\rangle\right)\xrightarrow{f-\text{cNOT}}|x\rangle\left(|0\oplus f(x)\rangle-|1\oplus f(x)\rangle\right)=(-1)^{f(x)}|x\rangle\left(|0\rangle-|1\rangle\right). \tag{5.5}$$

The last equality holds seeing that if $f(x)=1$ then $(|0\oplus 1\rangle-|1\oplus 1\rangle)=(|1\rangle-|0\rangle)=(-1)^1(|0\rangle-|1\rangle)$. As one can see now the initial state of the auxiliary qubit $(|0\rangle-|1\rangle)$ is an eigenstate of the action $U_f:|y\rangle\to|y\oplus f(x)\rangle$ for all x. Its eigenvalue $(-1)^{f(x)}$ is kicked back in front of the control qubit $|x\rangle$. For this reason the auxiliary qubit is not interesting anymore, seeing that it remains unchanged along the network. For the initial state $|x\rangle=H|0\rangle=(|0\rangle+|1\rangle)$ the state of the control qubit after applying the f-cNOT gate is

$$\left((-1)^{f(0)}|0\rangle+(-1)^{f(1)}|1\rangle\right)=(-1)^{f(0)}\left(|0\rangle+(-1)^{f(0)\oplus f(1)}|1\rangle\right). \tag{5.6}$$

The global phase $(-1)^{f(0)}$ can be neglected, whereas the relative phase difference $(-1)^{f(0)\oplus f(1)}$ is either $+1$ if f is constant or -1 if f is balanced. Therefore, the final state after the action of a second Hadamard gate on the control qubit will be with certainty either $|0\rangle$ or $|1\rangle$, depending only on the global property of the function f:

$$\left(|0\rangle+(-1)^{f(0)\oplus f(1)}|1\rangle\right)\xrightarrow{H}|f(0)\oplus f(1)\rangle. \tag{5.7}$$

If f is constant then a measurement of the control qubit in the computational basis will give with certainty $|0\rangle$. However, if f is balanced, the result will be $|1\rangle$. In terms of the Mach-Zehnder interferometer the phase difference between the two phase shifters was $\phi=\pm\pi$, if f was balanced but $\phi=0$, if was constant. The original algorithm is not deterministic as already mentioned in the introduction. Roughly speaking, the algorithm fails in half of the cases, that is, the algorithm ends up in a separate error state [3]. Its average power is therefore the

same as the power of the fastest classical algorithm. Nevertheless, if it succeeds only one action of the f-cNOT gate is used.

5.5 The Deutsch-Jozsa algorithm

The Deutsch algorithm is just a special case of the Deutsch-Jozsa algorithm published by David Deutsch and Richard Jozsa [4]. This generalization solved the original problem for boolean functions in more than just one dimension $f : \{0,1\}^n \to \{0,1\}$ as follows. A given function f is promised to be either constant or balanced (i.e. the number of outputs that are 0 is equal to the number of outputs that gives 1), and as before the goal is to determine which of these two global properties the function has. As the Deutsch algorithm the Deutsch-Jozsa algorithm needs only one evaluation of f. Classically in the best case scenario it would take 2 evaluations and in the worst case $2^{n-1}+1$ queries of f to get the right answer with certainty.

The network representation of the Deutsch-Jozsa algorithm looks similar to that of the Deutsch algorithm except that now instead of a control qubit a control register $|\mathbf{x}\rangle$ is needed. Also, the Hadamard gate H is replaced by a n-qubit Hadamard gate $H^{\otimes n}$ which action is simply defined as $H^{\otimes n}|\mathbf{x}\rangle = \bigotimes_{i=1}^{n} H|x_i\rangle$. The auxiliary qubit state stays the same, namely $|0\rangle - |1\rangle$, and is not altered during the network. The control register input will now be $|0\rangle^{\otimes n}$ and therefore after the first Hadamard gate it will look like

$$H^{\otimes n}|0\rangle^{\otimes n}(|0\rangle - |1\rangle) = \sum_{\mathbf{x}\in\{0,1\}^n} |\mathbf{x}\rangle(|0\rangle - |1\rangle). \tag{5.8}$$

This is an equally weighted superposition over all 2^n possible basis states. The action of $U_{f(\mathbf{x})}$ gives then the following result:

$$U_{f(\mathbf{x})} \sum_{\mathbf{x}\in\{0,1\}^n} |\mathbf{x}\rangle(|0\rangle - |1\rangle) = \sum_{\mathbf{x}\in\{0,1\}^n} (-1)^{f(\mathbf{x})}|\mathbf{x}\rangle(|0\rangle - |1\rangle). \tag{5.9}$$

For the last step, the action of the second n-qubit Hadamard transform, the equality 5.10 is used. For any computational n-qubit *basis* state $|\mathbf{x}\rangle$ the following equation holds

$$H^{\otimes n}|\mathbf{x}\rangle = \sum_{\mathbf{z}\in\{0,1\}^n} (-1)^{\mathbf{x}\cdot\mathbf{z}}|\mathbf{z}\rangle, \tag{5.10}$$

where $\mathbf{x}\cdot\mathbf{z}$ is the inner product modulo 2. In order to proof this (see also chapter 6.4 of [2]) first verify that the action of a one-qubit Hadamard transform

on a computational basis state can be written as $H|x\rangle = (|0\rangle + (-1)^x|1\rangle) = \sum_{z\in\{0,1\}}(-1)^{xy}|z\rangle$. Therefore the effect of $H^{\otimes n}$ on a n-qubit basis state $|\mathbf{x}\rangle = |x_1\rangle|x_2\rangle\ldots|x_n\rangle$ is given by

$$\begin{aligned} H^{\otimes n}|\mathbf{x}\rangle &= H|x_1\rangle H|x_2\rangle\ldots H|x_n\rangle \\ &= \bigotimes_{i=1}^{n}\sum_{z_i\in\{0,1\}}(-1)^{x_i\cdot z_i}|z_i\rangle = \sum_{\mathbf{z}\in\{0,1\}^n}(-1)^{\mathbf{x}\cdot\mathbf{z}}|\mathbf{z}\rangle\,, \end{aligned} \tag{5.11}$$

which is the same as the right hand side of Eq.5.10. Using this the final state of the algorithm is

$$\begin{aligned} H^{\otimes n}\sum_{\mathbf{x}\in\{0,1\}^n}(-1)^{f(\mathbf{x})}|\mathbf{x}\rangle(|0\rangle-|1\rangle) &= \sum_{\mathbf{x}\in\{0,1\}^n}(-1)^{f(\mathbf{x})}\sum_{\mathbf{z}\in\{0,1\}^n}(-1)^{\mathbf{x}\cdot\mathbf{z}}|\mathbf{z}\rangle(|0\rangle-|1\rangle) \\ &= \sum_{\mathbf{x},\mathbf{z}\in\{0,1\}^n}(-1)^{f(\mathbf{x})\oplus\mathbf{x}\cdot\mathbf{z}}|\mathbf{z}\rangle(|0\rangle-|1\rangle)\,. \end{aligned} \tag{5.12}$$

Not surprisingly the first register holds the solution to the original task whether f is balanced or constant. To find it, a measurement in the computational basis is made and the total probability amplitude of the state $|0\rangle^{\otimes n}$ (i.e. $\mathbf{z}=\mathbf{0}$) is observed. This amplitude is given by

$$a_{|0\rangle^{\otimes n}} = \frac{1}{2^n}\sum_{\mathbf{x}\in\{0,1\}^n}(-1)^{f(\mathbf{x})}, \tag{5.13}$$

where now the normalization factor was added afterwards again. In the case where f is constant, the amplitude is $+1$ or -1 depending on the value of $f(\mathbf{x})$, and therefore all other amplitudes have to vanish. This entails that a measurement of the first register returns the state $|0\rangle^{\otimes n}$ with certainty. On the other hand, if f is balanced the amplitude $a_{|0\rangle^{\otimes n}}$ has to be 0 and a measurement will never return the state $|0\rangle^{\otimes n}$.

This version of the Deutsch-Jozsa algorithm is a slight improvement of the original algorithm, that uses two f-cNOT operations, but was deterministic as well. The problem can also be generalized to functions $f:\{0,1\}^n\to\{0,1\}^m$ with $m\leq n$, where the auxiliary qubit will be replaced by a auxiliary register [5].

Since the quantum algorithm requires only one evaluation of f but the classical deterministic algorithm needs $2^{n-1}+1$ queries, an exponential gap in the efficiency between the classical and quantum algorithm can be claimed. But if the quantum algorithm is compared to a classical *probabilistic* algorithm where a small error

probability ϵ is allowed the gap will be at most linear. It is easy to prove [6], that the probability of error is smaller than $\frac{1}{2^n}$ with only $n+1$ evaluations. Thus, the gap will be linear for an exponentially small error and even constant if only a constant error ϵ is required.

5.6 Concluding remarks

With the help of superposition, entanglement and interference of multi-particle systems quantum algorithms can solve certain problems in a way that is classically not possible. In the Deutsch-Jozsa algorithm the different values of the function f are represented in the relative phase shifts of an equally weighted superposition state computed in only one action of a f-controlled quantum gate. After the action of the Hadamard transform the only information that survives is a global property of the function. A classical deterministic algorithm would need in the worst case $2^n + 1$ evaluations producing a lot unwanted (but for the computation essential) information, namely the specific values of f. However, the quantum algorithm does not have to produce this information explicitly because it uses interference to get an over all (global) impression of the function.
Although the Deutsch-Jozsa algorithm can not solve any practical problem, it proved the higher performance of quantum computation and was an inspiration for other more interesting algorithms based on the quantum Fourier transform, for example Shor's algorithm.

Bibliography

[1] T. Toffoli, *Reversible computing*, MIT Technical Report **MIT/LCS/TM-151**, 632 (1980).

[2] P. Kaye, R. Laflamme, and M. Mosca, *An Introduction to Quantum Computing* (Oxford University Press, 2007).

[3] D. Deutsch, *Quantum theory, the church-turing principle and the universal quantum computer*, Proc. R. Soc. Lond. **400**, 97 (1985).

[4] D. Deutsch and R. Jozsa, *Rapid solutions of problems by quantum computation*, Proc. R. Soc. Lond. **439**, 553 (1992).

[5] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca, *Quantum algorithms revisited*, Proc. R. Soc. Lond. **454**, 339 (1996).

[6] A. Collins and D. Collins, *Scaling issues in ensemble implementations of the deutsch-jozsa algorithm*, Phys. Rev. **68**, 052301 (2003).

Chapter 6

Quantum error correction

Junji Shimagaki
supervisor: Dr. Jean-David Picon

If we send information from one place to another place by telephone, e-mail or anything digital/analogue ways, the information is transmitted without disturbance in an ideal world. Of course, in real, many factors can disturb the information transmission which appear as errors. We have to cope with such errors to achieve correct information transmission.
In this chapter, we first introduce error descriptions mathematically to get the idea of error correction. Next we move on to the quantum error correction by observing the most typical quantum code, Shor code[1]. Finally we will, in some sense, review them in another description, stabilizer formalism.[2, 3]

6.1 Classical error correction

In order to understand the mechanism of quantum error correction, it is needed to understand how errors in the classical information processing are detected/ corrected/ read correctly. In this section, we will introduce so-called the *classical error correction* in which data are all composed of 0s and 1s. It turns out that we can improve the accuracy of information transmission by using *encoding* and that there exist some ways of such an encoding. Finally we conclude with the general theorem which assures the definite existence of a good encoding.

6.1.1 Classical information processing and repetition code

Errors are inevitable in information processing. They are caused by computers or even by humans. We will see what errors actually are. Consider that Alice would like to give Bob a password by telephone.[1] If the password is, 101 100 111, the message Bob has written down may be 101 000 111 because of a mishearing. Therefore, what Alice will do, is to repeat the password, in order to make sure whether Bob has the correct word. As a consequence, what Alice gave to him is 101 100 111 101 100 111. Then Bob must detect the inconsistency between two passwords, so he might ask her the password once more. Then he must notice that the first memo was his mishearing and finally he will get the same password as Alice's. Although it takes longer time, the accuracy is more important in many cases.

The error is caused by mishearing in this case and has a probability p of the occurrence. In classical channels, the types of errors are restricted only to *bit flip error* i.e. $0 \to 1$ and $1 \to 0$. Such a classical channel can be described by a conditional probability distribution. Each time Alice tells Bob one bit, there always exists an error probability p with which the original bit is flipped to another. To transmit the correct information, a sender adds some *redundancies*, which are two duplicated passwords in the example, on an original word and sends it to another party. The encoded word is called *codeword* and the group of whole codewords is *code*. The code used in the example is called *repetition code*. Let us see the repetition code more precisely.

Repetition code, 2-bit → 6-bit

$$10 \xrightarrow{\text{enc.}} 10\ 10\ 10 \xrightarrow[\text{error}]{\text{channel}} 10\ 00\ 10 \xrightarrow{\text{error correction}} 10\ 10\ 10 \xrightarrow{\text{dec.}} 10$$

Consider that Alice has a message described by $\{0,1\}^k$. She encodes the message into n-bit and sends it through a channel to Bob side. We take here $k = 2$ and $n = 6$. After the transmission he gets the message which contains one error on the third bit. He can notice the error because his message does not match with any of repetition codewords.[2] Afterwards he suspects that the third bit must be wrong because the codeword 101010 is the closest[3] to this message. Then he

[1] Alice and Bob are typical names for describing two parties who are at a long distance each other. We use the names later on.

[2] Usually the two parties had already agreed with which code would be used.

[3] Here "close" means the small value in Hamming distance, which will be discussed later.

decides to flip the third bit and manages to get the codeword. What he has to do at last is to decode it into 10.

One question might arise: how effective is this code? Suppose that Alice has a channel with error probability 90%. If she would like to send only one bit, then the probability of Bob guessing the original message correctly is 90%:

$$0 \xrightarrow{\times\ 1\ \text{channel}} 0 \text{ with prob } 90\%$$

If she applies the repetition code $k = 1 \longrightarrow n = 3$, then the probability increases to 97%:

$$0 \xrightarrow{\text{enc.}} 000 \xrightarrow{\times\ 3\ \text{channels}} \begin{array}{c} 000 \text{ with prob } 73\% \\ 001,\ 010,\ 100 \text{ with prob } 24\% \end{array}$$

He can always guess correctly because he knows that the error probability is smaller than 50% therefore two bit flip errors rarely happen. The more redundancies, the higher successful probability. However it is not smart to add too many redundancies if the probability reaches enough accuracy. Information transmission always costs something such as time, electricity, memory space, etc. In the next part, we will see more general way of encoding called *linear code*.

6.1.2 Classical linear code

Classical linear code is an efficient way to encode a k-bit message into an n-bit codeword. The advantages lie in the simplicity of describing the code and detecting errors. A message $x = \{0,1\}^k$ is encoded by a *generator matrix* $G\,[n,k]$ giving a codeword y composed of n-bit:

$$Gx = y \tag{6.1}$$

Let us consider one example, $x = 10 \longrightarrow y = 1100$. Such an encode is achieved by considering x and y as column vectors and G as a matrix:

$$\begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \tag{6.2}$$

where G is called *generator matrix*. The name *linear* code originates from the linearity of each column vector in the generator matrix. A generator matrix is sometime described as $G\,[n,k]$, meaning encoded from k-bit into n-bit. Obviously it corresponds to the number of columns and rows. Analogous to four codewords

in the 2-bit code space, trivially there are also four codewords in the 4-bit code space:

$$G\begin{pmatrix}0\\0\end{pmatrix}=\begin{pmatrix}0\\0\\0\\0\end{pmatrix},\quad G\begin{pmatrix}1\\0\end{pmatrix}=\begin{pmatrix}1\\1\\0\\0\end{pmatrix},\quad G\begin{pmatrix}0\\1\end{pmatrix}=\begin{pmatrix}0\\0\\1\\1\end{pmatrix},\quad G\begin{pmatrix}1\\1\end{pmatrix}=\begin{pmatrix}1\\1\\1\\1\end{pmatrix}$$

Then, we will define another matrix, *parity check matrix* $H\,[n-k,n]$. The parity check matrix is composed of $(n-k)$ row vectors which give 0 by performing the product with every column vectors of G. Therefore, the following must be fulfilled:

$$HG=\mathbf{0}\,[n-k,k] \tag{6.3}$$

In this example, a parity check matrix is given as,[4]

$$H\,[2,4]=\begin{pmatrix}1&1&0&0\\0&0&1&1\end{pmatrix}$$

From the eq.(6.3), we can derive a useful relation with codewords:

$$Hy=H(Gx)=(HG)x=\mathbf{0} \tag{6.4}$$

Let us see the case when H is applied to a codeword y and a non-codeword y':[5]

$$Hy=H\begin{pmatrix}1\\1\\0\\0\end{pmatrix}=\begin{pmatrix}0\\0\end{pmatrix},\quad Hy'=H\begin{pmatrix}1\\0\\0\\0\end{pmatrix}=\begin{pmatrix}1\\0\end{pmatrix} \tag{6.5}$$

The result gives zero if a codeword is applied and non-zero if not. Therefore one can make use of a parity check matrix with the aim of checking whether the received message y' is a codeword. Error correction is done by searching the closest codeword to the message. The procedure is to calculate

$$y_{\text{corrected}}=\min_{\{y\}} d\,(y,y')$$

where $d\,(y,y')$ is called *Hamming distance*. The Hamming distance gives the number of different bits in two strings. In this example where $y=1100$ and $y\prime=1000$,

$$d(y,y')=0+1+0+0=1 \tag{6.6}$$

[4]There is not only one choice of parity check matrices, but several. We can change the order of row vectors arbitrary.

[5]The modulo 2 arithmetic is used in the calculation. Example: $1+1+0=0$ and not 10.

It can be easily seen that there are no other codewords which have a smaller Hamming distance than 1. Hence we can conclude y =1100 must be the original codeword.
In addition to such a simple way of encoding and detecting/correcting error, there is still advantage in linear coding. If we would like to encode $k-$bit into $n-$bit, in general we need $n \cdot 2^k$-bit to get $n-$bit. In linear coding, however, we need only $k \times n$ components of a generator matrix! This saves the amount of information dramatically because of the lack of the exponential term 2^k.

6.1.3 Gilbert-Varshamov bound

The *Gilbert-Varshamov bound*[2] gives a relation among the length k of original message, n of codeword, and t of correctable errors. The theorem assures the existence of good code if k is not too large compared to n:

$$\frac{k}{n} \geq 1 - H\left(\frac{2t}{n}\right)$$

where $H(x) \equiv -x \log x - (1-x) \log (1-x)$

6.2 Quantum error correction

Analogous to the classical information transmission, we would like to send information from one party to another party. The fundamental difference with the classical case is that information is not contained in a set of 0s and 1s, but in a quantum state $|\psi\rangle$ described by a superposition of two levels:

$$|\psi\rangle = a|0\rangle + b|1\rangle \tag{6.7}$$

In other words, *a and b are the information*. This kind of a two level system is called *qubit*.[6]
But is such information transmission possible? As we have used the term quantum *information transmission* in the previous section, errors are inevitable. Additionally, there exist three more different properties which will disturb error detections and correction:

1. *No cloning*: It is impossible to create the same quantum state. Therefore, the repetition code cannot be realized:

$$|\psi\rangle \nrightarrow |\psi\rangle|\psi\rangle|\psi\rangle \tag{6.8}$$

[6]A qubit is described by several physical systems such as a photon polarization or a spin of electron. They can be described by a superposition of two energy levels.

2. Measurements destroy a and b: If we measure the state, such as $|\psi\rangle = a|0\rangle + b|1\rangle$, to know what is the error, the state collapse to one of the two states $|0\rangle$ or $|1\rangle$.

3. Linear combination of different types of errors: In case we can detect one error, what if the error is a mixture of different errors? Unfortunately, this is the case.

However, we can still realize the quantum information transmission reliably by making use of some quantum codes! We will see how it goes by reviewing some quantum information theory and getting the idea of quantum codes. Then we will finally reach our main theme in this chapter, the *Shor code*. The history of the Shor code dates back to 1995, when his idea is published for the first time[1]. Among the many types of difficulties in the quantum computation, one of the most important issues is how to cope with *decoherence* (Notice that this term is equivalent to *error*). The Shor code proves the feasibility of quantum coding and solves the above three difficulties.

6.2.1 Quantum error description

A quantum state is transmitted through a channel which is sometimes called *quantum channel*.[7] A quantum channel contains some errors with probability p and is described by a *quantum operation* $\mathcal{E}$ from a quantum state ρ to another state $\mathcal{E}(\rho)$:

$$\mathcal{E}(\rho) = (1-p)\, I\rho I + \sum_{i=x,y,z} p_i \sigma_i^\dagger \rho \sigma_i \tag{6.9}$$

$$(1-p) + p_x + p_y + p_z = 1 \tag{6.10}$$

where σ_i is one of Pauli matrices. σ_x, σ_y and σ_z correspond to bit flip, phase + bit flip, and phase flip. Hence Eq.(6.9), (6.10) imply that:

- nothing happens to the qubit with probability $(1-p)$, i.e. the information is perfectly preserved.

- one of errors σ_i happens with probability p_i.

These are the general form of an arbitrary error effect. It is true because all physical operations are written as an Hermitian operator, which is expanded

[7] There are basically some ways to realize the quantum channel practically. One way is to transport a qubit physically. The another way is to make use of *quantum teleportation*, in which there is no way to transport a qubit but in possession of two entangled qubits and a classical 2-bit channel.

by three Pauli matrices and the identity operator. As a consequence of the expansion, the expansion parameters can be understood as the probabilities of each error.
The strategies against such errors are followings:

- To detect which errors have occurred without measuring each qubit.
- To find a way to make use of the idea of the repetition code, even though we cannot duplicate a quantum state.

Next, we will see some properties of *measurements* and *quantum code* which are in some sense keys to open the door into the quantum error correction.

6.2.2 Projective measurements

Projective measurements are of use remarkably especially in a quantum error detection process. If an appropriate projective basis is chosen, it can determine what types of error has occurred without destroying the information a and b.
To see how powerful projective measurements are, let us see two examples to compare projective measurements with **unsuitable/suitable** basis for error detection.
Projective measurement of a pair of entangled two qubits:
Suppose that a pair of entangled qubits $|\psi_i\rangle = |\psi^{(0)}\rangle \equiv a|00\rangle + b|11\rangle$ has gone through a channel which has an error probability p of bit flip on the first qubit. The initial state and final state are written as

$$\begin{array}{c} \text{Initial state} \\ |\psi_i\rangle = |\psi^{(0)}\rangle \end{array} \xrightarrow{\text{Channel}} \begin{array}{c} \text{Final state} \\ |\psi_f\rangle = \sqrt{1-p}|\psi^{(0)}\rangle + \sqrt{p}|\psi^{(1)}\rangle \end{array} \tag{6.11}$$

where $|\psi^{(1)}\rangle = a|10\rangle + b|01\rangle$. First of all, as an **unsuitable** base case, we perform a measurement with following four projective base:[8]

$$P_0' = |00\rangle\langle 00|, \quad P_1' = |10\rangle\langle 10|, \quad P_2' = |01\rangle\langle 01|, \quad P_3' = |11\rangle\langle 11| \tag{6.13}$$

After the measurement, the state is randomly determined as $|11\rangle$ i.e. the measurement destroyed a and b and leave the state $|11\rangle$. However with following two **suitable** projective base,

$$P_0 = |00\rangle\langle 00| + |11\rangle\langle 11|, \quad P_1 = |10\rangle\langle 10| + |01\rangle\langle 01| \tag{6.14}$$

[8]Measurement operators must satisfy the *completeness equation*:

$$\sum_m m M_m^\dagger M_m = I \tag{6.12}$$

the state after the measurement is

$$|\psi_f\rangle \xrightarrow{\text{After observed no error}} \frac{P_0|\psi_f\rangle}{\langle\psi_f|P_0|\psi_f\rangle} = |\psi^{(0)}\rangle$$
$$|\psi_f\rangle \xrightarrow{\text{After observed an error}} \frac{P_1|\psi_f\rangle}{\langle\psi_f|P_1|\psi_f\rangle} = |\psi^{(1)}\rangle$$

In this case, the superposition is not lost and even shows in which state the qubits are.[9]

We can generalize projective measurements. A projective measurement is a special case of the *observable measurement*, M, a self-adjoint operator acting on the observed state. The observable can always be spectrally decomposed as

$$M = \sum_m m P_m \tag{6.15}$$

where P_m is the projector onto the state with eigenvalue m. The observable gives the outcome m with the probability $\langle\psi|P_m|\psi\rangle$. In this sense, the measurement eq.(6.13) has eigenvalues of 0 or 1. Such an outcome is the crucial information for the error detection because it tells where the error is in. We call the outcome *error syndrome* in quantum error correction. Let us see one example of an observable measurement, but look at the same state as in eq.(6.11):

Example. $\sigma_{Z1}\sigma_{Z2}$ on the state $|\psi_f\rangle$ in eq.(6.11)

The observable $\sigma_{Z1}\sigma_{Z2}$ has a spectral decomposition,

$$\begin{aligned}\sigma_{Z1}\sigma_{Z2} &= (|0\rangle\langle 0| - |1\rangle\langle 1|)(|0\rangle\langle 0| - |1\rangle\langle 1|) = |00\rangle\langle 00| + |11\rangle\langle 11| - |01\rangle\langle 10| - |10\rangle\langle 01| \\ &= P_0 - P_1 = (+1)P_0 + (-1)P_1 \end{aligned} \tag{6.16}$$

where $\sigma_{Z1}\sigma_{Z2}$ corresponds to the phase flip operator on first and second bits. P_0 and P_1 are the same as in eq.(6.13). Eq.(6.16) implies that the observable gives the outcome $(+1)$ with probability $\langle\psi_f|P_0|\psi_f\rangle = \langle\psi^{(0)}|P_0|\psi^{(0)}\rangle$ and (-1) with probability $\langle\psi^{(1)}|P_1|\psi^{(1)}\rangle$. Needless to, the state collapse to the corresponding state after the observable measurement.

6.2.3 Quantum code

Now we are ready to understand how a quantum code works. Here we will introduce so-called the *bit flip code* and the *phase flip code* which construct the *Shor code*. Our task is, to send $|\psi\rangle = a|0\rangle + b|1\rangle$ to another place with an assumption that only single error occurs.

[9] One can understand this situation as a collapsing of the superposition of the initial and the bit-flipped state.

Bit flip code

1. **Encode**: We encode this 1-qubit into a 3-qubit using two CNOT-gates.

$$|\psi_B\rangle = a|000\rangle + b|111\rangle = |\psi_B^{(0)}\rangle \tag{6.17}$$

2. **A bit flip errror in a quantum channel**: There are error probabilities p_1, p_2 and p_3 on the first, second, and third qubit respectively:

$$|\psi'_B\rangle = \sqrt{1-p}|\psi_B^{(0)}\rangle + \sqrt{p_1}|\psi_B^{(1)}\rangle + \sqrt{p_2}|\psi_B^{(2)}\rangle + \sqrt{p_3}|\psi_B^{(3)}\rangle \tag{6.18}$$

where $p = p_1 + p_2 + p_3$ which imply error probabilities and $|\psi_B^{(1)}\rangle = a|100\rangle + b|011\rangle$ and so on.

3. **Error detection**: The error is detected by projective measurements. The base are

$$\begin{aligned} P_0 &= |000\rangle\langle 000| + |111\rangle\langle 111|, \quad P_1 = |100\rangle\langle 100| + |011\rangle\langle 011|, \\ P_2 &= |010\rangle\langle 010| + |101\rangle\langle 101|, \quad P_3 = |001\rangle\langle 001| + |110\rangle\langle 110| \end{aligned} \tag{6.19}$$

For now, let us suppose the result $P_1 = 1$ and $P_{j\neq 1} = 0$. Then the state we have now is

$$\frac{P_1|\psi'_B\rangle}{\sqrt{\langle\psi'_B|P_1|\psi_B\rangle}} = a|100\rangle + b|011\rangle \equiv |\psi_B^{(1)}\rangle \tag{6.20}$$

4. **Error correction**: Since we have known where the bit flip error has occurred, we can choose the appropriate error correction operator. We apply the bit flip operation on the first qubit:

$$\sigma_{X1}|\psi_B^{(1)}\rangle = |\psi_B\rangle \tag{6.21}$$

Phase flip code

1. **Encode**: We encode this 1-qubit into a 3-qubit using two CNOT-gates and three Hadamard-gates.

$$|\psi_P\rangle = a\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)^{\otimes 3} + b\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)^{\otimes 3} = a|{+}{+}{+}\rangle + b|{-}{-}{-}\rangle = |\psi_P^{\{0\}}\rangle \tag{6.22}$$

2. **A phase flip errror in a quantum channel**: There are error probabilities p_1, p_2 and p_3 on the first, second, and third qubit respectively. Since we have applied Hadamard-gates on each qubit, the effect of a phase flip error is interpreted as the bit flip from $|+\rangle$ to $|-\rangle$ and vice versa.

$$|\psi'_P\rangle = \sqrt{1-p}|\psi_P^{\{0\}}\rangle + \sqrt{p_1}|\psi_P^{\{1\}}\rangle + \sqrt{p_2}|\psi_P^{\{2\}}\rangle + \sqrt{p_3}|\psi_P^{\{3\}}\rangle$$

where $p = p_1 + p_2 + p_3$ which imply error probabilities and $|\psi_B^{\{1\}}\rangle = a|-++\rangle + b|+--\rangle$ and so on.[10]

3. **Error detection**: The error is detected by observable measurements $\sigma_{X1}\sigma_{X2}$ and $\sigma_{X2}\sigma_{X3}$. If we get $\sigma_{X1}\sigma_{X2} = -1$ and $\sigma_{X2}\sigma_{X3} = -1$, the only possible eigenfunction is $a|+-+\rangle + b|-+-\rangle$ i.e. the phase flip error has occurred on the second qubit. By this method, we can determine the state depending on the outcomes:

$$|\psi'_P\rangle \xrightarrow[\sigma_{X1}\sigma_{X2}=+1]{\text{observed}} \sqrt{1-p}|\psi_P^{\{0\}}\rangle + \sqrt{p}|\psi_P^{\{3\}}\rangle \xrightarrow[\sigma_{X2}\sigma_{X3}=-1]{\text{observed}} |\psi_P^{\{3\}}\rangle$$

4. **Error correction**: In the last step, it turned out that the phase flip error has occurred on the third qubit. To get the original state back, we apply the phase flip operation on the third qubit:

$$\sigma_{Z3}|\psi_P^{\{3\}}\rangle = |\psi_P\rangle \tag{6.23}$$

6.2.4 Shor code

Although the Shor code is so powerful that it can cope with an arbitrary error on any qubit, we can create the code only from the bit flip code and phase flip codes.
In the Shor code, a qubit is encoded as following (see also Figure 6.1):

$$|0\rangle \xrightarrow{\text{enc.}} |0_L\rangle \equiv \frac{(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)}{2\sqrt{2}} \tag{6.24}$$

$$|1\rangle \xrightarrow{\text{enc.}} |1_L\rangle \equiv \frac{(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)}{2\sqrt{2}} \tag{6.25}$$

First of all, we will see how a combination of a bit and phase flip error is detected and corrected.

Against a both phase and bit flip on qubit(s)

1. **Encode**: We encode the state using the Shor code:

$$|\psi\rangle = a|0\rangle + b|1\rangle \xrightarrow{\text{enc.}} |\psi_L\rangle = a|0_L\rangle + b|1_L\rangle \tag{6.26}$$

[10] The notations $|\psi_B^{(i)}\rangle$ and $|\psi_P^{\{j\}}\rangle$ which we have used implicity for a bit flip on i-th qubit and a phase flip bit on j-th qubit are valid throughout this chapter.

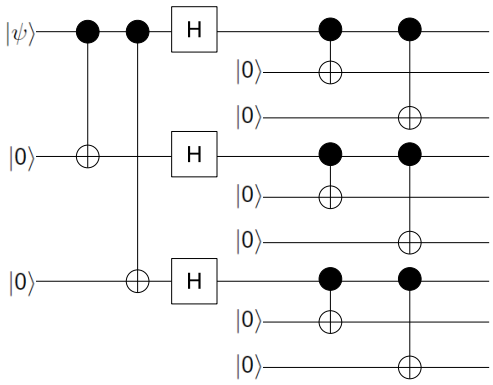


Figure 6.1: Shor code: A one-qubit is encoded into nine-qubit using CNOT-gates and Hadamard-gates.

2. **A bit and a phase flip error on qubit(s)**: There are bit flip error probabilities $p_0, \ldots, p_9$ and phase flip error probabilities $p'_0, \ldots, p'_9$ on the all nine qubits. The state after the channel is

$$|\psi'_L\rangle = \sum_{k,j=0}^{9} \sqrt{p_k p'_j} \left(a|0_L^{(k)\{j\}}\rangle + b|1_L^{(k)\{j\}}\rangle \right) \tag{6.27}$$

where $\sum_{k=0}^{9} p_k = \sum_{j=0}^{9} p'_j = 1$.

3. **Bit error detection and correction**: Since the signs in brackets of the state do not affect the projective measurements, we can deal with a bit flip error first. The error is detected by either projective measurements or observable measurements but we introduce here the former only. The error syndrome is obtained by performing projective measurement with base,

$$\begin{aligned} P_0 &= |000\ 000\ 000\rangle\langle 000\ 000\ 000| + |111\ 111\ 111\rangle\langle 111\ 111\ 111| \\ P_1 &= |100\ 000\ 000\rangle\langle 100\ 000\ 000| + |011\ 111\ 111\rangle\langle 011\ 111\ 111| \\ &\cdots \\ P_9 &= |000\ 000\ 001\rangle\langle 000\ 000\ 001| + |111\ 111\ 110\rangle\langle 111\ 111\ 110| \end{aligned} \tag{6.28}$$

Randomly, let us get the outcome $P_3 = 1$. The state after the measurement is:

$$\frac{P_3 \sum_{k,j=0}^{9} \sqrt{p_k p'_j} \left(a|0_L^{(k)\{j\}}\rangle + b|1_L^{(k)\{j\}}\rangle \right)}{\sqrt{\langle \psi'_L | P_3 | \psi'_L \rangle}} = (+1) \sum_{j=0}^{9} \sqrt{p'_j} \left(a|0_L^{(3)\{j\}}\rangle + b|1_L^{(3)\{j\}}\rangle \right)$$

As the same method in the bit flip code, we recover the state by applying

σ_{X3}:

$$\sigma_{X3}\sum_{j=0}^{9}\sqrt{p'_j}\left(a|0_L^{(3)\{j\}}\rangle + b|1_L^{(3)\{j\}}\rangle\right) = \sum_{j=0}^{9}\sqrt{p'_j}\left(a|0_L^{\{j\}}\rangle + b|1_L^{\{j\}}\rangle\right) \quad (6.29)$$

4. **Phase flip error detection and correction**: In contrast to the bit flip error case, the effects of the phase flip error on any three qubits in the same block[11] do not differ. For example,

$$|0_L^{\{1\}}\rangle = |0_L^{\{2\}}\rangle = |0_L^{\{3\}}\rangle = \frac{(|000\rangle - |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)}{2\sqrt{2}}$$

Such a phase flip error is detected by observable measurements $\sigma_{X1}\cdots\sigma_{X6}$ and $\sigma_{X4}\cdots\sigma_{X9}$. Let us get the outcomes here $\sigma_{X1}\cdots\sigma_{X6} = -1$ and $\sigma_{X4}\cdots\sigma_{X9} = -1$.[12]Although we cannot tell which qubit has caused phase flip error, we can still observe that it has occurred in the second block. To correct the sign in the second block, we apply the phase flip operator $\sigma_{Z4}\sigma_{Z5}\sigma_{Z6}$ and we get the original state back.

Against an arbitrary error on a qubit

The Shor code is applicable to a linear combination of these error operations. Remember that a quantum error in general is described by a quantum operation $\mathcal{E}$ in eq.(6.9) and consider that this quantum operation is acted on a particular qubit[13]. If we expand the quantum operation by operator-sum representation, we can measure the error syndromes that determines what kinds of errors have occurred. In other words, the measurements of error syndromes let the quantum state collapse to one of the error states we have discussed.

[11]The term "block" implies a group of three qubits enclosed by a bracket in eq.(6.24), (6.25).

[12]

$$\sigma_{X1}\cdots\sigma_{X6}|\psi_L^{\{6\}}\rangle = 2^{-\frac{3}{2}}a\,(|111\rangle + |000\rangle)(|111\rangle - |000\rangle)(|000\rangle + |111\rangle)$$
$$+2^{-\frac{3}{2}}b\,(|111\rangle - |000\rangle)(|111\rangle + |000\rangle)(|000\rangle - |111\rangle) = (-1)a|0_L^{\{6\}}\rangle + (-1)b|1_L^{\{6\}}\rangle = (-1)|\psi_L\rangle$$
$$\sigma_{X4}\cdots\sigma_{X9}|\psi_L^{\{6\}}\rangle = 2^{-\frac{3}{2}}a\,(|000\rangle + |111\rangle)(|111\rangle - |000\rangle)(|111\rangle + |000\rangle)$$
$$+2^{-\frac{3}{2}}b\,(|000\rangle - |111\rangle)(|111\rangle + |000\rangle)(|111\rangle - |000\rangle) = (-1)a|0_L^{\{6\}}\rangle + (-1)b|1_L^{\{6\}}\rangle = (-1)|\psi_L\rangle$$

[13]For simplicity, we have restricted our problem to the error on "one particular qubit". The readers might think one can not forecast on which qubit the error occurs. However, we do not have to know where the error happens. More important issue is that only one error happens somewhere. Then we can label such a qubit as "a particular qubit".

Finally, we introduce the general theorem which guarantees the definite existence of a good quantum code. Theorem 10.1 in Ref.[2] proves that the existence of an error correction operation $\mathcal{R}$ under the condition for quantum operation elements $\{E_i\}$:

$$PE_i^\dagger E_j P = \alpha_{ij} P, \quad \alpha_{ij} \in \mathcal{C}$$

where P is the projector onto the code space, where the code is defined. At least the Shor code fulfills this condition. The three difficulties we have introduced in the beginning were all solved by the Shor code.

6.3 Stabilizer code

In this last section, we will briefly introduce a new representation of quantum states and codes. We have always represented a quantum state by a summation of ket vectors. Since it is a summation, it looks sometime pretty long and ugly. On the other hand, this new formalism, the *stabilizer formalism* allows us to represent a quantum state by a set of *generators* which are actually equivalent to observable.
Let us observe an EPR pair to see how the stabilizer formalism is made. Such a state is described as

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

From the stabilizer formalism view, one can say that $|\psi\rangle$ is *stabilized* by $\sigma_{X1}\sigma_{X2}$ and $\sigma_{Z1}\sigma_{Z2}$ because,

$$\sigma_{X1}\sigma_{X2}|\psi\rangle = \frac{1}{\sqrt{2}}(|11\rangle + |00\rangle) = |\psi\rangle, \quad \sigma_{Z1}\sigma_{Z2}|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = |\psi\rangle$$

These $\sigma_{X1}\sigma_{X2}$ and $\sigma_{Z1}\sigma_{Z2}$ are called the *stabilizer* $S = \{\sigma_{X1}\sigma_{X2}, \sigma_{Z1}\sigma_{Z2}\}$ and the elements are called *generators* of a vector space $V_S = \{|00\rangle, |11\rangle\}$.
S is a subgroup of the *Pauli group* G_n:

$$G_n = \{\pm I_n, \pm iI_n, \pm\sigma_{xn}, \ldots, \pm i\sigma_{zn}\}$$

where n corresponds to the label of qubit. G_n is closed under multiplication: $S \subset G$. One can soon notice that $-I$ is not included in S because there is no state which is stabilized by $-I$ except $|\psi\rangle = \mathbf{0}$. And the elements of S commute each other.

Example. Bit flip code

The encoded state is stabilized by

$$S_{\text{Long}} = \{I, \sigma_{Z1}\sigma_{Z2}, \sigma_{Z2}\sigma_{Z3}, \sigma_{Z1}\sigma_{Z3}\} \tag{6.30}$$

However, when we take into account of the property of commutability, it turns out

$$\sigma_{Z1}\sigma_{Z2} \cdot \sigma_{Z2}\sigma_{Z3} = \sigma_{Z1}\sigma_{Z3} = \sigma_{Z2}\sigma_{Z3} \cdot \sigma_{Z1}\sigma_{Z2} \tag{6.31}$$

In this sense, we can create $\sigma_{Z1}\sigma_{Z3}$ from the other two generators, and same for I if we perform $\sigma_{Z1}\sigma_{Z2} \cdot \sigma_{Z1}\sigma_{Z2} = I$. Therefore the stabilizer form becomes

$$S = \{\sigma_{Z1}\sigma_{Z2}, \sigma_{Z2}\sigma_{Z3}\} \tag{6.32}$$

Finally we will introduce how this formalism can be applied to quantum codes. Let us use here the bit flip code as an example.

1. **Stabilizer code**: As we have seen right now, the bit flip code is spanned by the vector space $V = \{|000\rangle, |111\rangle\}$ which is stabilized by $S = \langle\sigma_{Z1}\sigma_{Z2}, \sigma_{Z2}\sigma_{Z3}\rangle$. The state is described by the vector space and the stabilizer.

2. **An error occurs**: An error causes the mutations of the vector space and stabilizer. In case the bit flip error has occurred on the first qubit, then the mutated vector space and the stabilizer are

$$S' = \langle -\sigma_{Z1}\sigma_{Z2}, \sigma_{Z2}\sigma_{Z3}\rangle, \qquad V' = \{|100\rangle, |011\rangle\} \tag{6.33}$$

3. **Error detection and correction**: To detect the error, we have to get the error syndrome which is obtained just by performing the observable measurements S. Acting a generator on the state must always give eigenvalue 1, if not something is wrong. In this case of the error on the first qubit, $\sigma_{Z1}\sigma_{Z2} = -1$, $\sigma_{Z2}\sigma_{Z3} = +1$. Therefore we can tell that the bit flip error has occurred on the first qubit and we correct the state by applying σ_{X1}.

6.4 Further reading

The application of quantum error correction is not only to the achievement of correct quantum information transmission but also to the dynamical computation. As information is processed dynamically, errors will be accumulated in time. The quantum error correction can be applied to the before/after such cases. For the further reading of this technique, we will mention the section "Fault-tolerant quantum computation" in Ref.[2].

Bibliography

[1] P. W. Shor, *Scheme for reducing decoherence in quantum computer memory*, Phys. Rev. A **52**, 2493 (1995).

[2] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, New York, 2007).

[3] G. Benenti, G. Casati, and G. Strini, *Principles of Quantum Computation and Information, Volume II: Basic Tools and Special Topics* (World Scientific, Singapore, 2007).

Chapter 7

Fault-Tolerant Quantum Computing & Quantum Fourier Transform

Markus Schmassmann
supervisor: Oscar Dahlsten

We recall how quantum errors can be detected and therefore corrected and show that using fault-tolerant operations and measurements we can construct arbitrarily large quantum computers that operate reliably, given that the error rate of every gate is below a certain constant threshold.
In the next section we start a distinct topic where we see that the Discrete Fourier Transformation on quantum states can be efficiently implemented on a quantum computer, which can be used to estimate the eigenvalues and phases of unknown quantum operators. The applications of the phase estimation are treated in the next chapter.

7.1 Introduction

In this chapter we treat two distinct topics, fault-tolerant quantum computing[1], which shows how to design quantum circuits consisting of error prone elements

[1]The student scheduled to review fault-tolerant quantum computing was unable to do so, therefore the present author agreed to review that topic in addition to the Quantum Fourier Transform

such that the output is still accurate up to a small error, and the Quantum Fourier Transform, which is an element of a class of quantum algorithms that give an exponential speedup over their classical counterparts.

7.2 Fault-Tolerant Quantum Computing

For classical computers, correction codes for transmitted information are well known, but for quantum computers, qubits can neither be cloned nor can they be read out without affecting the data. Therefore more sophisticated techniques are necessary. While for classical computing, operations on encoded data can be implemented on repetition codes without any difficulty, we need a more careful analysis for quantum computers.
We restrict ourselves to unitary evolutions on qubits. If more than one qubit is affected, the errors are assumed to be uncorrelated.

7.2.1 Quantum Errors Correction

As we have seen in the last chapter, errors can be decomposed linearly, i.e. after measurement of the error syndrome the error is clearly defined as one of a finite set of errors.
This can be understood as a state $|\psi\rangle$ and its environment $|E\rangle$ evolving under a unitary operator and afterwards tracing out the environment we are not interested in. Linear Algebra and also Imamoglu [1] shows that this is equivalent to applying a set of operators on $|\psi\rangle$ called Kraus operators $\mathscr{E}_i$,

$$Tr_E \rho = Tr_E \left(U_{err} |\psi\rangle |E\rangle \langle E| \langle \psi| U_{err}^\dagger \right) = \sum_i \mathscr{E}_i |\psi\rangle \langle \psi| \mathscr{E}_i^\dagger$$

which do not have to be unitary or Hermitian but have to fulfil the completeness relation $\sum_i \mathscr{E}_i \mathscr{E}_i^\dagger = I$. These operators may be known to you from the generalized measurement postulates, where the state after measurement is

$$\frac{\mathscr{E}_i \psi}{\sqrt{\langle \psi| \mathscr{E}_i^\dagger \mathscr{E}_i |\psi\rangle}}$$

with probability $\langle \psi| \mathscr{E}_i^\dagger \mathscr{E}_i |\psi\rangle$. Should we not be interested in the final state, we have a positive operator valued measurement (POVM). On the other hand, if we do not record the measurements, we are in our original setup where the linear superposition of all Kraus operators is applied.

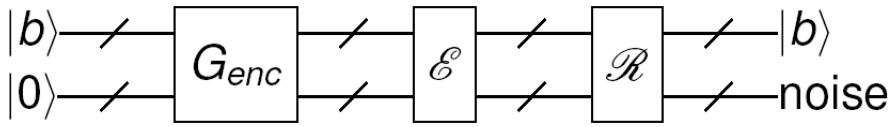


Figure 7.1: Encoding a quantum state, sending it through a noisy quantum channel and simultaneously decoding the state and recover from errors.

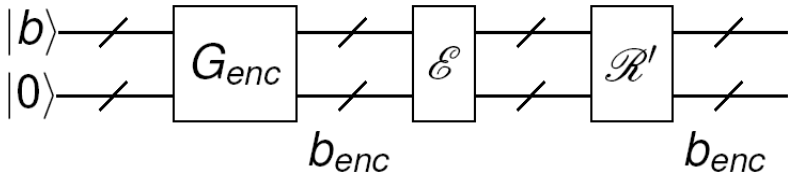


Figure 7.2: Noisy quantum channel where errors are recovered without decoding.

According to section 10.4.1 of Kaye, Laflamme & Mosca [2], the most general evolution that can occur on a single qubit can be decomposed into the Pauli Matrices and Unity $X, Y, Z\&I$, even in case of non-trivial interaction with the environment.
To send qubits through noisy quantum channels we start with encoding them with a set of gates G_{enc}. Then we transmit them through the channel where errors occur and at last the information can be decoded, where the noise introduced by errors is contained in the ancilla qubits (see fig. 7.1).
But if we want to construct a fault tolerant quantum computer, we do not only have to deal with some noisy channels but every component of the computer is subjected to noise, therefore the information we are interested in must always be encoded, i.e. the recovery operation must transform the quantum state back into the code space without decoding it (see fig. 7.2).

7.2.2 Codes

3 qubit codes

Following sections 10.1, 10.2 & 10.5.6 of Nielsen & Chuang [3] we introduce different encoding schemes for qubits that protect to a different degree from errors. The easiest code is the 3 qubit bit flip code

$$\alpha\,|0\rangle + \beta\,|1\rangle \mapsto \alpha\,|000\rangle + \beta\,|111\rangle$$

that protects against the bit flip of one of the physical qubits. After having gone through the bit flip channel, the error syndromes, that are parities between

error states - measured operators	$Z \otimes Z \otimes I$	$I \otimes Z \otimes Z$
$I \otimes I \otimes I(\alpha \|000\rangle + \beta \|111\rangle)$	$+1$	$+1$
$X \otimes I \otimes I(\alpha \|000\rangle + \beta \|111\rangle)$	-1	$+1$
$I \otimes X \otimes I(\alpha \|000\rangle + \beta \|111\rangle)$	-1	-1
$I \otimes I \otimes X(\alpha \|000\rangle + \beta \|111\rangle)$	$+1$	-1

Table 7.1: Errors on the 3 qubit bit flip code and its syndromes.

physical qubits that do not give up any information about the state of the logical qubit, are measured and the result is used to classically calculate what error occurred. Here we measure $Z \otimes Z \otimes I$ and $I \otimes Z \otimes Z$ then we look in the table 7.1 what error occurred and correct it.
Phase flips can be corrected on a similar 3 qubit code

$$\alpha \|0\rangle + \beta \|1\rangle \mapsto \alpha \|+++\rangle + \beta \|---\rangle ,$$

where $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ and $|-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$, and are detected by the syndromes measurements $X \otimes X \otimes I$ and $I \otimes X \otimes X$.

Codes protecting against all local errors

9 qubit Shor code The Shor code is nothing more than a concatenation of phase and bit flip 3 qubit codes, has therefore 9 qubits and protects against bit and phase flips as well as their combination. It encodes

$$|0_L\rangle \text{ as } (|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \frac{1}{2\sqrt{2}} \text{ and}$$

$$|1_L\rangle \text{ as } (|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle) \frac{1}{2\sqrt{2}},$$

where $|0_L\rangle$ and $|1_L\rangle$ stand for the logical 0 and 1 respectively.
Because of its easy derivation this code is often used for pedagogical purposes.

7 qubit Steane code The Steane code is the one we will use for the further discussion, since - as we see later - the relevant operations can easily be implemented and it needs less physical qubits than the Shor code. The logical qubits

are encoded like this:

$$|0_L\rangle = \frac{1}{\sqrt{8}}\Big(|0000000\rangle + |1010101\rangle + |0110011\rangle + |1100110\rangle + |0001111\rangle + |1011010\rangle + |0111100\rangle + |1101001\rangle \Big)$$
$$|1_L\rangle = \frac{1}{\sqrt{8}}\Big(|1111111\rangle + |0101010\rangle + |1001100\rangle + |0011001\rangle + |1110000\rangle + |0100101\rangle + |1000011\rangle + |0010110\rangle \Big)$$

You can easily convince yourselves that $|0_L\rangle$ and $|1_L\rangle$ are orthogonal. This code is one of the so called Calderbank-Shor-Steane codes, which are a subclass of stabilizer codes. For them a theory of error correction has been developed, but this theory is beyond the scope of this chapter.

5 qubit code and Hamming bound The 5 qubit code is mainly of pure theoretical interest, since for many applications the Steane code is more transparent. The theoretical interest comes from the fact, that there cannot be a code with less than 5 qubits that correct against an arbitrary error on one physical qubit. This comes from the Hamming bound which states, that if you have n physical qubits to encode k logical qubits and allow t errors to happen, you can faithfully decode your information only if

$$\sum_{j=0}^{t} \binom{n}{j} 3^j 2^k \leq 2^n,$$

which gives us for the desired $t = k = 1$ that

$$2(1 + 3n) < 2^n \Leftrightarrow n \geq 5.$$

$\binom{n}{j}$ counts how many possible combinations there are of having j out of n qubits affected, 3^j is the number of different combination of errors, 2^k the size of the code space and 2^n the size of the state space. Since for every possible error a separate code space that is disjoint with the other code spaces has to fit into the state space, the Hamming bound has to be fulfilled to guarantee faithful decoding.

7.2.3 Fault-Tolerant Operations

Being able to faithfully transmit qubits is a good start, but we also need to operate on the encoded states. On the 3 qubit bit flip code (and also the Steane

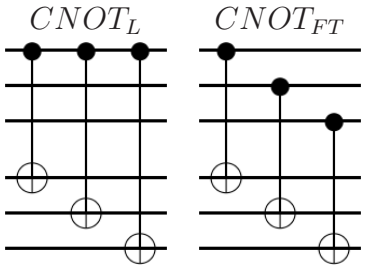


Figure 7.3: $CNOT$s on 3 qubit bit flip code, only the right one is fault tolerant; if in the left one an error occurs on the highest wire, the whole lower bundle of qubits is affected.

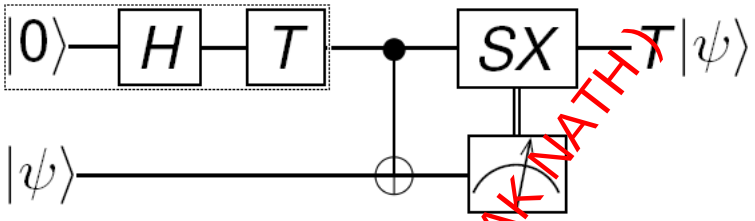


Figure 7.4: The Θ-state is prepared in the dotted box, the $CNOT$ and the classically controlled SX-operation simulate in a fault-tolerant way a $\pi/8 = T$ gate.

code) the implementation of CNOT seems straight forward. Just use a physical CNOT between one of the physical qubits of the logical control qubit and control with it all physical bits of the target qubit. But this naive approach is not fault tolerant, since a mistake in one of the physical control qubits can affect all of the physical qubits of the target qubits. We have to implement the gate in parallel, so that only one physical qubit per logical qubit is affected if there is an error (see fig. 7.3).
For the 7 qubit Steane code a universal set of gates can be implemented fault tolerantly. Section 10.6.2 of Nielsen & Chuang [3] shows us how $\widehat{H} = H^{\otimes 7}$, $\widehat{X} = X^{\otimes 7}$, $\widehat{Z} = Z^{\otimes 7}$, $\widehat{S} = (ZS)^{\otimes 7}$ and $CNOT$ are implemented in parallel.

Fault-Tolerant $\pi/8$

To complete the universal set we need construct the $\widehat{\pi/8} = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{i} \end{pmatrix}$ gate.

As a first step we have to construct a state $\Theta = \frac{|0\rangle + e^{i\pi/4}|1\rangle}{\sqrt{2}}$ fault tolerantly. We prepare the state $H|0\rangle$ and measure then fault-tolerantly $e^{-i\pi/4}SX$, get either $+1$ and have the state we want or get -1 which we can transform to Θ by applying

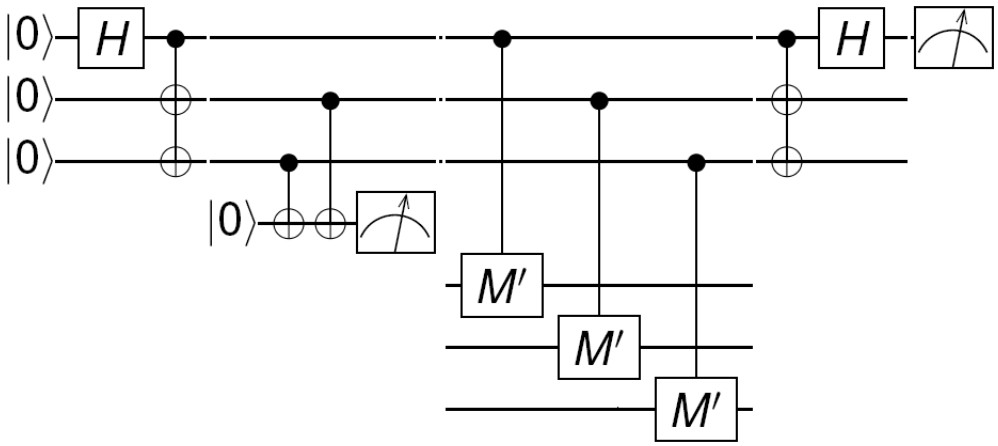


Figure 7.5: Fault-Tolerant Measurement on the 7 qubit Steane Code, for readability only shown with 3 qubits; first preparing the cat state, verifying it, applying a controlled M in parallel, decoding and measuring the result.

Z. How the fault-tolerant measurement is done will be explained further below. This state (in fig. 7.4 "prepared" in the doted box) is used as the control qubit of a $CNOT$ giving

$$\begin{aligned}&\frac{1}{\sqrt{2}}\left(|0\rangle\left(a\,|0\rangle+b\,|1\rangle\right)+e^{i\pi/4}|1\rangle\left(a\,|1\rangle+b\,|0\rangle\right)\right)\\&=\left(\left(a\,|0\rangle+b\,e^{i\pi/4}\,|1\rangle\right)|0\rangle+\left(b\,|0\rangle+a\,e^{i\pi/4}\,|1\rangle\right)|1\rangle\right)\end{aligned}$$

If the target qubit is measured afterwards as $|0\rangle$ we have the desired state, otherwise we apply a

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

to get our desired state up to a global phase.
This construction allows us to simulate the application of a $\pi/8$ gate, but requires that it is supplied with fresh $|0\rangle$ states.

7.2.4 Fault-Tolerant Measurements

For measurements to be useful in quantum computing the measurement results have to be accurate and the data qubits must remain coherent.
For the fault-tolerant measurement circuit (fig. 7.5) as introduced in section 10.6.3 of Nielsen & Chuang [3] we start with constructing a so called 'cat state'

Figure 7.6: Example of a fault-tolerant circuit, with error recovery $\mathscr{R}$ after every operation.

$(|0\cdots0\rangle + |1\cdots1\rangle)/\sqrt{2}$ (after Schrödinger's Cat that is entangled with a radioactive nucleus). The procedure is error prone; therefore we have to verify whether we produced the state we want. This is done by measuring the parities Z_iZ_j, which must be 1 otherwise we discard the state. The parity measurement is repeated on different ancilla qubits until all have been involved in a parity measurement.
Afterwards we apply a controlled-M, which can be implemented transversally for H, X, Y, Z, S and for $M = e^{-i\pi/4}SX$, while for the last one we use controlled ZSX which are followed by Ts on the ancillas.
At last we decode the ancilla and measure it, which is again error prone but has no influence on the data qubits. Therefore we need to repeat the whole measuring procedure and do majority voting.
The only errors we have not yet dealt with are those of the verify procedure. On the Steane code a Z error on an ancilla qubit causes a false measurement but no data corruption, while an X or Y error propagates to at most one data qubit.

7.2.5 Threshold Theorem

We have seen so far that there exists at least one code - the Steane Code - that corrects for all unitary errors, on which a universal set of gates can be implemented fault-tolerantly and on which fault-tolerant measurements are possible.
Using all these to construct a fault-tolerant circuit (as fig. 7.6) we assure that no error on a qubit will propagate to another qubit in its block. During transmission, operation, syndrome measurement and recovery are $c \approx 10^4$ pairs of places, where two errors can occur.
To reduce the error rate we can replace a physical qubit and the gates and measurements applied to it by a logical qubit which consists itself of several qubits that are encoded. This is called concatenation and explained in more details in section 10.6.1 of Nielsen & Chuang [3]. If we concatenate a circuit k times, the error probability is proportional to $(cp)^{2^k}/c$, while the size of the circuit is proportional to d^k, where d is the maximum number of operations used for a gate

and the following error correction.
We want to solve a problem of input size n that requires $p(n)$ gates, where $p(n)$ is a polynomial, and allow a maximal final error rate of ε, therefore the maximally allowed error rate per logical gate is $\varepsilon/p(n)$. This requires that $(cp)^{2^k}/c \leq \varepsilon/p(n)$. From this we know that for any physical error rate p that is smaller than a constant threshold $< p_{th} = 1/c$ for any size of the logical circuit $p(n)$ and any error rate of the physical gates and measurements ε there is a level of concatenation k such that

$$d^k = \left(\frac{\log\left(p(n)/c\varepsilon\right)}{\log\left(1/pc\right)}\right)^{\log d} = \mathcal{O}\left(poly\left(\log\left(p(n)/\varepsilon\right)\right)p(n)\right),$$

which means that there is only a polylogarithmical overhead over the original size of the not fault-tolerant circuit.
This can be reformulated as the *Threshold Theorem*:

> Any Quantum Circuit containing $p(n)$ gates can be simulated with probability of error at most ε using $\mathcal{O}\left(poly\left(\log\left(p(n)/\varepsilon\right)\right)p(n)\right)$ gates in hardware whose components fail with probability at most $p < p_{th} = 1/c$.

How much the threshold p_{th} can be increased is a topic of ongoing research, Knill [4] claims that $p_{th} \approx 1\%$ can be attained by combination of concatenation and post-select procedures, but the overhead would require quantum computers of the size of current PCs. He furthermore claims that in the limit of infinite computer sizes even $p_{th} \approx 3.5\%$ is possible.
Even for more pessimistic thresholds it seems reasonable to assume that the necessary hardware can be provided by experimental physicists within reasonable time, so there is no reason to assume that quantum computing will not be implemented due to noise problems.

7.2.6 Summary

Codes as well as fault-tolerant gates and measurements allow us to correct for unitary evolutions on qubits, where errors on more than one qubit are uncorrelated. With concatenation we can reduce the error in the output as much as required, if our physical elements have error rates below a certain threshold p_{th}. According to section 10.6.4 of Nielsen & Chuang [3] this conclusion can also be extended to other error models and more sophisticated codes if some physically reasonable assumptions about the type of noise hold and the architecture of the

quantum computer provides the necessary supply of fresh ancilla qubits in the ground state $|0\rangle$.

7.3 Quantum Fourier Transform

The algorithms based on the Quantum Fourier Transform (QFT) are the only quantum algorithms known up to now that give an exponential speedup over the best known classical algorithms. The other group of algorithms give only a quadratic speedup over classical algorithms and are based on quantum search, which is treated in the chapter 3 on Grover's algorithm.

7.3.1 Fourier Transform

The (classical, continuous) Fourier Transform $\mathscr{F}$ is defined as an Operator that maps one function to another:
$\mathscr{F}: f(x) \mapsto \widetilde{f}(k) = \int f(x) e^{ikx} \frac{\mathrm{d}x}{\sqrt{2\pi}}$ and its inverse is
$\mathscr{F}^{-1}: \widetilde{f}(k) \mapsto f(x) = \int f(k) e^{-ikx} \frac{\mathrm{d}k}{\sqrt{2\pi}}$.
The discrete Fourier Transform $\mathscr{F}_N$ maps a vector of the space $\mathbb{C}^N$ onto another:

$$\mathscr{F}_N: \quad x_0, x_1, \ldots, x_{N-1} \quad \mapsto \quad y_1, y_2, \ldots, y_{N-1}, \text{ where}$$

$$y_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i j k/N} \text{ and } x_k \equiv \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} y_j e^{-2\pi i j k/N}.$$

You can easily convince yourselves, that the transformation matrix is symmetric and its inverse is equal to its conjugate, therefore it is also unitary.
The Quantum Fourier Transform $\mathscr{F}_{2^n}$ maps a n qubit state to another:

$$|j\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i j k/N} |k\rangle,$$

where $|j\rangle$ means $|j_1\rangle \cdots |j_n\rangle$, its binary representation. For the rest of this chapter we understand j as $j_1 j_2 \cdots j_n = \sum_{k=1}^{n} 2^{n-k} j_k$ and $0.j_l j_{l+1} \cdots j_n$ as $\sum_{k=l}^{n} 2^{l-k-1} j_k$.
In the special case, where the QFT is applied to a state where all qubits are in their ground state it results in a equal superposition of all possible states without any phases, which can also be reached by applying a Hadamard to every qubit.

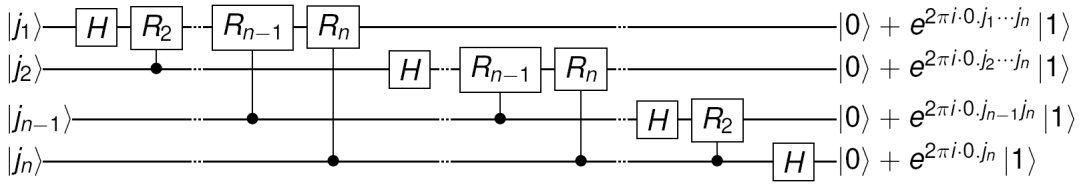


Figure 7.7: QFT circuit, normalizations of $1/\sqrt{2}$ and swaps at the end left out.

For the QFT section 5.1 of Nielsen & Chuang [3] introduces also a product representation:

$$\begin{aligned}
\mathscr{F}_{2^n} |j\rangle &= 2^{-n/2} \sum_{k=0}^{2^n-1} d^{2\pi i j k / 2^n} |k\rangle \\
&= 2^{n/2} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} e^{2\pi i j \left(\sum_{l=1}^{n} k_l 2^{-l}\right)} |k_1 \cdots k_n\rangle \\
&= \frac{1}{2^{n/2}} \otimes_{l=1}^{n} \left(\sum_{k_l=0}^{1} e^{2\pi i j k_l 2^{-l}} |k_l\rangle \right) \\
&= \frac{1}{2^{n/2}} \left(|0\rangle + e^{2\pi i \cdot 0.j_n} |1\rangle\right) \left(|0\rangle + e^{2\pi i \cdot 0.jn-1 j_n} |1\rangle\right) \\
&\quad \cdots \left(|0\rangle + e^{2\pi i \cdot 0.j_1 \cdots j_n} |1\rangle\right)
\end{aligned}$$

This product representation is also used for the calculation the Discrete Fourier Transform, where it is the basis of the (classical) Fast Fourier Transform algorithm that needs $\Theta(n2^n)$ operations as opposed to $\Theta(2^{2n})$ for the brute force method. Here the notation $f(n) = \Theta(g(n))$ means, that $f(n)$ scales as fast as $g(n)$, or more formally

$$\limsup_{n\to\infty} \left| \frac{f(n)}{g(n)} \right| < \infty \quad \text{and} \quad \liminf_{n\to\infty} \left| \frac{f(n)}{g(n)} \right| > 0.$$

7.3.2 QFT Circuit

For us it is more important that the product representation gives us a blueprint for constructing a circuit to implement the QFT (fig. 7.7).
Following section 5.1 of Nielsen & Chuang [3] we start with $|j_1\rangle |j_2 \cdots j_n\rangle$ and apply a Hadamard to the first qubit
$\frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i \cdot 0.j_1} |1\rangle\right) |j_2 \cdots j_n\rangle$,
then we control with the second qubit a rotation of $\pi/2$ of the first qubit and get
$\frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i \cdot 0.j_1 j_2} |1\rangle\right) |j_2 \cdots j_n\rangle$.

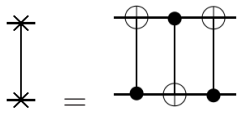


Figure 7.8: Construction a SWAP out of 3 CNOTs.

We continue by applying controlled rotations by $2\pi/2^k$ controlled with the k^{th} qubit and get
$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i \cdot 0.j_1 \cdots j_n}|1\rangle)|j_2 \cdots j_n\rangle$ after we reached the n^{th} qubit. The rotations are performed by the following operators:
$R_k \equiv \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}$ or $_cR_k \equiv \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & e^{2\pi i/2^k} \end{pmatrix}$ in the controlled version.
We repeat the procedure with the second to the n^{th} qubit and get
$\frac{1}{2}(|0\rangle + e^{2\pi i \cdot 0.j_1 \cdots j_n}|1\rangle)(|0\rangle + e^{2\pi i \cdot 0.j_2 \cdots j_n}|1\rangle)|j_3 \cdots j_n\rangle$.
Continuing like this gives
$\frac{1}{2^{n/2}}(|0\rangle + e^{2\pi i \cdot 0.j_1 \cdots j_n}|1\rangle) \cdots (|0\rangle + e^{2\pi i \cdot 0.j_n}|1\rangle)$,
for which we only have to swap the first with the n^{th}, second with the $(n-1)^{th}$ etc. to get the desired
$\frac{1}{2^{n/2}}(|0\rangle + e^{2\pi i \cdot 0.j_n}|1\rangle) \cdots (|0\rangle + e^{2\pi i \cdot j_1 \cdots 0.j_n}|1\rangle) = \mathscr{F}_{2^n}|j_1 \cdots j_n\rangle$.
The $SWAP$s are constructed of 3 $CNOT$s as shown in fig. 7.8.
The circuit for the inverse QFT we can get by either inverting the full QFT circuit or equivalently by replacing the R_k's by R_k^{-1}s. You can easily check this yourself by taking into account that cR_k is symmetric under $SWAP$ of control and target qubit.

7.3.3 Efficiency & Exactitude

The Quantum Fourier Transform needs $\Theta(n^2)$ gates, while the fastest known classical algorithm, the Fast Fourier Transform, needs $\Theta(n2^n)$ operations. Unfortunately it is neither possible to read out the amplitudes directly nor to efficiently produce any desired input state. The later we construct by applying the respective operator to the $|0 \cdots 0\rangle$ state. In chapter the chapter about Computational models for quantum computing we have seen, that any unitary operator $U \in \mathbb{C}^{2^n \times 2^n}$ can be simulated using at most $2^{\mathscr{O}(n)}$ gates, while in section 4.5.4 of Nielsen & Chuang [3] is proven that there are unitary gates that require at least $\Omega(2^n \log(1/\varepsilon)/\log(n))$ gates, which is far from an efficient construction. The $\mathscr{O}$ and Ω notations are analogous to the Θ notation on page 105, but only the left or the right inequality respectively are used. Therefore most of the applications

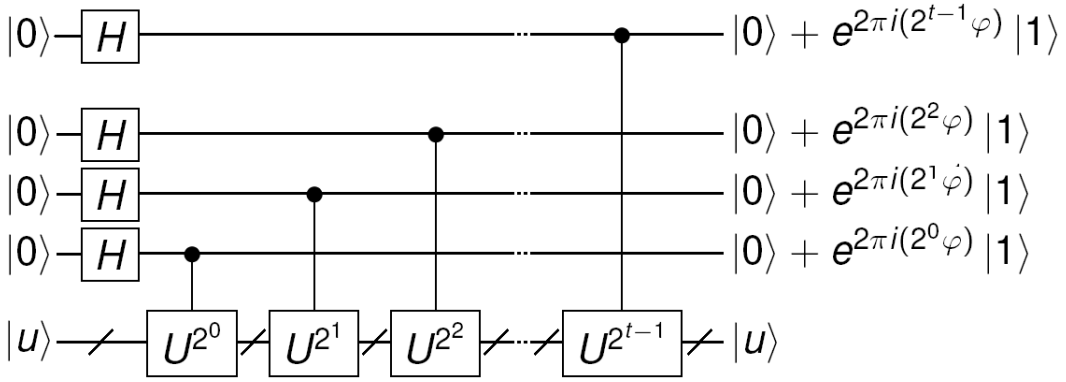


Figure 7.9: Detailed circuit of the first part of phase estimation.

of the Fourier Transform like digital signal processing and solving of partial differential equations cannot be done more efficiently on a quantum computer than on a classical one, as is confirmed in chapter 5.1 of Nielsen & Chuang [3].
Although the rotations angles of the gates we should perform decrease exponentially, chapter 5.1 of Nielsen & Chuang [3] claims it is sufficient that the precision of the gates grows polynomially. More precisely if we have a exact QFT circuit of n qubits $U = \mathscr{F}_{2^n}$, R_k gates with precision $\Delta \equiv 1/p(n)$, where $p(n)$ is a polynomial used to construct an approximate QFT $V \approx \mathscr{F}_{2^n}$ we get an error $E(U,V) \equiv max_{|\psi\rangle} \, ||(U-V)\,|\psi\rangle|\,|$ that scales as $\Theta(n^2/p(n))$.

7.4 Phase Estimation

Phase estimation is the quantum part of Shor's algorithm for factoring numbers and of similar algorithms for the rest of the hidden subgroup problems. They provide an exponential speedup over their classical counterparts, for factoring numbers this would be the number field sieve.
The phase estimation routine as introduced in section 5.2 of Nielsen & Chuang [3] starts with a state $|u\rangle$ and a Black Box cU^j for $j \leq t$ that is also called oracle as input. It gives out a n-bit approximation to φ such that $U\,|u\rangle = e^{2\pi i\varphi}\,|u\rangle$, with a success probability of $1-\varepsilon$. It requires one call to the cU^j Black Box and has a runtime of $\Theta(t^2)$, where $t = n + \lceil \log\left(2 + \frac{1}{2\varepsilon}\right)\rceil$. Its implementation is shown in figures 7.9 and 7.10.
First we apply Hadamard gates to ancillas in $|0\rangle$ states to get an equal superposition $\sum_{j=0}^{2^t-1} |j\rangle\,|u\rangle$, then we use the qubits to subsequently control U^j gates on the state $|u\rangle$ which leaves us with $\sum_{j=0}^{2^t-1} |j\rangle\,U^j\,|u\rangle = \sum_{j=0}^{2^t-1} |j\rangle\,e^{2\pi ij\varphi}\,|u\rangle$. Then

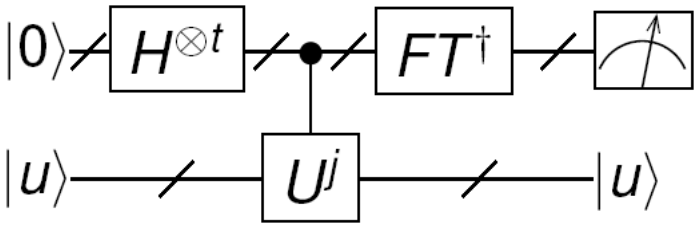


Figure 7.10: Complete circuit of phase estimation; the slash on the wire denotes a bundle of qubits.

we apply an inverse Fourier Transform to the ancillas to get $|\varphi_1\rangle \cdots |\varphi_t\rangle |u\rangle$, the binary representation of the phase ϕ of the eigenvalue of the operator $U = e^{2\pi i\varphi}$ associated with the eigenstate $|u\rangle$.
Should we not be able to prepare the eigenstate $|u\rangle$ we can prepare a superposition of different eigenstates and we will get an approximation to phase φ_u with probability $(1-\varepsilon){c_u}^2$.
How phase estimation deals with phases that are not a multiple of $2\pi 2^{-t}$ and how it is used in Shor's algorithms is shown in the next chapter by Roger Herrigel.

7.4.1 Summary

In the second and third section we have introduced Quantum Fourier Transform and Phase estimation as well as their circuits, which are the central parts of Shor's Algorithm for factoring numbers that provides an exponential speed-up over the fastest known classical algorithm.

7.5 Conclusion

We have seen that there is no reason to assume that noise is still a problem for quantum computing once a experimental setup is found such that the errors of every gate is below a constant threshold. Nevertheless a sceptic might claim that our assumptions about the noise models are too restrictive. Whether they hold can only be proven in the laboratory by an operational large scale quantum computer, or the failure to produce one.
Recall that Quantum Fourier Transform and Phase Estimation are exponentially faster than their known classical counterparts, the same holds for algorithms built of them. Given that some of these applications, that will be introduced in the next chapter, solve problems for which a lot of effort has been invested in finding

efficient classical algorithms, I think it is fair to assume that quantum computers are more efficient than classical computers.

Bibliography

[1] A. Imamoglu, *Topics in Quantum Information Processing*, Lecture notes taken by Andreas Fragner (2007), lecture of the 22.10.07 (p. 24) published on http://www.qudev.ethz.ch/content/courses/QSIT07/lecture_notes_imamoglu.pdf.

[2] P. Kaye, R. Laflamme, and M. Mosca, *An Introduction to Quantum Computing* (Oxford University Press, 2007).

[3] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, 2000).

[4] E. Knill, *Quantum Computing with Very Noisy Devices* (2004), `quant-ph/0410199v2`.

Chapter 8

Shor's algorithm

Roger Herrigel
supervisor: Wojciech De Roeck

Factoring integers on a classical computer is thought to be a hard problem. Public encryption systems like the RSA rest on this quality of integers to encode massages. In 1994, Peter Shor showed that, using a randomized algorithm on a quantum computer, the factoring problem can be solved more efficiently. While the fastest known classical algorithm scales sub-exponentially, Shor's algorithm needs only a number of steps polynomial in the input size. Thus, using a quantum computer, the widely used RSA can be easily broken.

8.1 Motivation

Up to now, there are only three known classes of quantum algorithms which provide an advantage over classical algorithms. The first class is based on the quantum Fourier transform. Examples are the Deutsch-Jozsa algorithm, Shor's algorithm and the discrete logarithm. The quantum Fourier transform can be used to solve the hidden subgroup problem, which is a generalization of these algorithms. The second class is a quantum search algorithm discovered by Grover. The third class is quantum simulation, where a computer is used to simulate a quantum system.
We concentrate on the first class of algorithms and we see how they can be used to break RSA [1] and other public encryption systems.
The RSA public-key cryptography method is based on the assumption that it is computationally infeasible to factor a large integer number N.

The general number field sieve [2] is the most efficient algorithm known for factoring integers larger than 100 digits on a **classical computer**. For an integer number N of L bits, it has a subexponential complexity of the form

$$O\left(e^{(c+o(1)L^{\frac{1}{3}}(\log L)^{\frac{2}{3}}}\right). \tag{8.1}$$

Although there is no mathematical proof, it is widely believed that on a classical computer, an integer can not be factored in polynomial time. In contrast, on a **quantum computer**, it is possible to factor in

$$O\left(L^3\right) \tag{8.2}$$

steps.

8.2 Shor's algorithm

Shor's algorithm splits a non-prime integer number N into two non-trivial factors p and q. Iterating this procedure on both factors p and q, one is able to find the prime factors of N.
Shor's algorithm can be divided into two parts.

a) A reduction of the factoring problem to a problem of order-finding. This can be done on a classical computer.

b) An algorithm solving the order-finding problem. This is done on the quantum computer.

8.3 Classical part

This section describes the reduction of the factoring problem to the order-finding problem.

8.3.1 The order-finding problem

Definition 8.3.1. *The **order of a finite group** G, is the number of elements in G and it is denoted as $|G|$.*

Definition 8.3.2. *The **order of an element** g in a group G, $\mathrm{ord}_G(g)$, is the least integer r such that $g^r = 1_G$*

Thus, the order of $g \in G$ is the order of the cyclic group generated by the element g, which is defined as $\{1_G, g, g^2, g^3, ...\}$. As a consequence we have $r \leq |G|$. The order finding problem is the following: given a group G and an element $g \in G$, we want to find the order r of the element g.

Definition 8.3.3. *Two numbers x and y are **co-prime** iff.* $\gcd(x, y) = 1$.

Definition 8.3.4. *$\mathbb{Z}_N^*$ is the set of all elements in $\{0, 1, ..., N\}$ which are co-prime to N. This set together with the multiplication modulo N defines a group.*

This means that all elements in $\mathbb{Z}_N^*$ have a unique multiplicative inverse modulo N. This inverse can be found using Euclid's algorithm. The factoring problem can be reduced to a problem of finding the order of an element of the group $\mathbb{Z}_N^*$, which is defined as

Definition 8.3.5. *For two positive integers x and N, $x < N$ and $\gcd(x, N) = 1$ the **order of** x **modulo** N is defined to be the least positive integer r such that $x^r = 1 \pmod N$.*

From now on any mentioning of the order-finding problem refers to the problem of finding the order of an element of the group $\mathbb{Z}_N^*$.

8.3.2 Reduction of the factoring to the order finding problem

For the reduction of the factoring problem to the order finding problem we use the following theorem.

Theorem 8.3.1. *Suppose N is a non-prime number and x a non-trivial solution of $x^2 = 1 \pmod N$. If neither $x = 1 \pmod N$ nor $x = -1 \pmod N$, then at least one of $\gcd(x - 1, N)$ and $\gcd(x + 1, N)$ is a non-trivial factor of N.*

Proof. From $x^2 = 1 \pmod N$, it follows that $x^2 - 1 = 0 \pmod N$ and thus $x^2 - 1 = (x - 1)(x + 1) = kN$. Since N is a non-prime number, it must have a common factor with either $x - 1$ or $x + 1$. By assumption $x \pm 1 \neq 0 \pmod N$ and therefore the common factor cannot be N itself. Thus, either $\gcd(x - 1, N)$ or $\gcd(x + 1, N)$ is a non-trivial factor of N. □

If now y is co-prime to N and has an even order r, such that $y^r = 1 \pmod N$, and $y^{r/2} \neq \pm 1 \pmod N$, then $x \equiv y^{r/2} \pmod N$ is a solution to $x^2 = 1 \pmod N$ and $x \neq \pm 1 \pmod N$.
Thus, an algorithm for reducing the factoring to the order finding problem needs the following steps. We fist pick a random number y, $1 < y < N$. We calculate

$\gcd(y, N)$. If $\gcd(y, N) > 1$, we have found a non-trivial factor of N and we are done. If $\gcd(y, N) = 1$, we know that y and N are co-prime. Now we use our order finding subroutine to calculate the order r of y modulo N. If r is even and $y^{r/2} \neq \pm 1 \pmod{N}$, then we can define $x \equiv y^{r/2} \pmod{N}$ and calculate $\gcd(x-1, N)$ and $\gcd(x+1, N)$. According to our theorem, one of these two numbers is a non-trivial factor of N.
Thus, we are interested in the probability that the order r of y modulo N is even and $y \neq \pm 1 \pmod{N}$.

Theorem 8.3.2. *Suppose $N = p_1^{c_1} p_2^{c_2} ... p_m^{c_m}$ is the prime decomposition of an odd non-prime integer. Let y be a random number uniformly chosen from $\mathbb{Z}_N^*$ and r its order. Then*

$$p\left(r \text{ even and } y^{r/2} \neq -1 \pmod{N}\right) \geq 1 - \frac{1}{2^{m-1}}. \tag{8.3}$$

Corollary 8.3.1. *For all odd integers N not of the form p^c or $2p^c$, where p is a prime number and c an integer, we have that*

$$p\left(r \text{ even and } y^{r/2} \neq \pm 1 \pmod{N}\right) \geq \frac{1}{2}. \tag{8.4}$$

The proof of this theorem is more complicated and can be found in [3] or [4].

8.3.3 Procedure

Given is an integer number N. If N is not a prime number, then the following procedure yields a nontrivial factor of N.

1. If N is even, return 2.
2. Check if $N = p^q$. If so return p.
3. Choose a number $y < N$.
4. If $\gcd(y, N) > 1$, return $\gcd(y, N)$.
5. Use the order-finding subroutine to calculate the order r of y, modulo N. This is the part calculated on the quantum computer.
6. If r is even and $y^{r/2} \neq -1 \pmod{N}$, compute $\gcd(y^{r/2} - 1, N) > 1$ and $\gcd(y^{r/2} + 1, N) > 1$, one of them is a non-trivial factor of N. Return this factor.

If r is odd or $y^{r/2} \neq -1 \pmod{N}$, this algorithm fails. But the probability that this happens is smaller or equal 1/2. The order-finding subroutine on the quantum computer (Step 5) can also fail with a probability bounded by a constant independent of L. Thus, by repeating the procedure many times, the probability that it always fails goes to zero.

8.3.4 Performance

Checking if $N = p^c$, calculating the greatest common divisor, and the modular exponentiation can each be done in $O(L^3)$ operations.

8.3.5 Euclid's algorithm

Euclid's algorithm for finding the greatest common divisor of two positive integers $x > y$, denoted as $\gcd(x, y)$, works as follows:
Since $x > y$, we can write $x = k_1 y + r_1$, with $r_1 < y$. Observing that $\gcd(x, y) = \gcd(y, r_1)$, we can iteratively go on and write $y = k_2 r_1 + r_2$. We continue this procedure until the remainder r_{n+2} is zero and we obtain $\gcd(x, y) = \gcd(y, r_1) = \gcd(r_1, r_2) = ... = \gcd(r_n, r_{n+1}) = r_{n+1}$.

We want to calculate the performance of this algorithm. Suppose x and y are numbers of at most L bits. It follow that also k_1 and r_1 have at most L bits. The main point is that $r_{i+2} \leq r_i/2$. For the case $r_{i+1} \leq r_i/2$, it is clear that $r_{i+2} \leq r_i/2$. For $r_{i+1} > r_i/2$, it follows that $r_i = r_{i+1} + r_{r+2}$ and thus $r_{i+2} \leq r_i/2$. Since the remainder is reduced every second step by at least one bit, we need $O(L)$ divider-and-remainder operations. Each divider-and-remainder operation needs $O(L^2)$ steps such that the total performance is $O(L^3)$.

8.3.6 Modular exponentiation

Given x, z and N we want to compute $x^z \mod N$ efficiently. First we write z as $z = z_t 2^{t-1} + ... + z_t 2^0$, where $z_i \in \{0, 1\}$. Thus, $z_t z_{t-1} ... z_0$ is the binary representation of the t bit number z. The modular exponentiation $x^z \mod N$ can thus be written as

$$x^z \mod N = \left(x^{z_t 2^{t-1}} \mod N\right) ... \left(x^{z_1 2^0} \mod N\right) \mod N. \qquad (8.5)$$

The trick is that we can compute the expressions $(x^{2^l} \mod N)$ by squaring $(x^{2^{l-1}} \mod N)$. And since $z_i \in \{0, 1\}$, each factor $(x^{z_l 2^{l-1}} \mod N)$ in equation 8.5 is

either 1 if $z_l = 0$ or $x^{2^{l-1}} \mod N$ if $z_l = 1$. Squaring a L bit numbers needs $O(L^2)$ steps and there are $t - 1 = O(L)$ of these squarings. Thus, this product can be computed using $O(L^3)$ steps and the space used is $O(L)$.

8.3.7 N not of the form $N = p^c$

We want to check if N is of the form $N = p^c$ where $p > 2$ is a prime number and $c \geq 1$ an integer. We fist fix c. Since $p > 2$, we have $c \leq L \equiv \log_2(N)$. Now for each fixed c we have

$$p = N^{\frac{1}{c}} = \exp \frac{\log N}{c}. \tag{8.6}$$

The logarithm and the exponential function can be calculated using $O(L^2)$ operations. To test every possible c, we need in total $O(L^3)$ steps.

8.4 Quantum part

8.4.1 quantum Fourier transform

The **quantum Fourier transform** is a procedure for calculating the **discrete Fourier transform**. Acting on the the orthonormal basis $|0\rangle, ..., |n-1\rangle$ we write it as

$$|j\rangle \to \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} e^{2\pi ijk/n} |k\rangle. \tag{8.7}$$

Thus, the action of the quantum Fourier transform on an arbitrary state is

$$\sum_{j=0}^{n-1} x_j |j\rangle \to \sum_{k=0}^{n-1} y_k |k\rangle \tag{8.8}$$

where

$$y_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} x_j e^{2\pi ijk/n}. \tag{8.9}$$

The quantum Fourier transform is a unitary transform and thus realizable as a quantum circuit. On a quantum computer, the quantum Fourier transform can be realized with $O(n^2)$ gates. In contrast, the fastest classical algorithm for a discrete Fourier transform, the Fast Fourier Transform, needs $O(n2^n)$ gates. Unfortunately, the amplitudes in a quantum computer are not directly accessible and therefore the use of the quantum Fourier transform is highly limited.

8.4.2 Phase estimation

Given is a unitary operator U with eigenvector $|u\rangle$ and corresponding eigenvalue $e^{2\pi i\phi}$, where the phase $\phi \in [0,1)$ is unknown. The goal of the phase estimation is to estimate ϕ. The procedure needs two oracles, also called black boxes. One preparing the state $|u\rangle$ and one performing a controlled-U^{2^j} operation, for integers $j \geq 0$.
The procedure uses two registers.

Register	initial state	bits
1	$\|0\rangle$	t qubits
2	$\|u\rangle$	as many qubits as necessary

The number of bits t of register 1 is determined by the number of digits of desired accuracy n for the approximation of ϕ and the probability of success $p_{\text{success}} = 1-\epsilon$ of our phase estimation procedure. For given n and ϵ one needs

$$t = n + \lceil \log\left(2 + \frac{1}{2\epsilon}\right) \rceil \tag{8.10}$$

The phase estimation procedure is performed in two stages. We first apply the circuit shown in Figure 8.1. This transforms the state $|0\rangle|u\rangle$ into

$$\begin{aligned} |0\rangle|u\rangle \quad &\rightarrow \quad \frac{1}{\sqrt{2^t}}\left(|0\rangle + e^{2\pi i 2^{t-1}\phi}|1\rangle\right)\left(|0\rangle + e^{2\pi i 2^{t-2}\phi}|1\rangle\right) \ldots \left(|0\rangle + e^{2\pi i 2^0\phi}|1\rangle\right)|u\rangle \\ &= \quad \frac{1}{\sqrt{2^t}}\sum_{j=0}^{2^t-1} e^{2\pi i\phi k}|k\rangle|u\rangle \end{aligned} \tag{8.11}$$

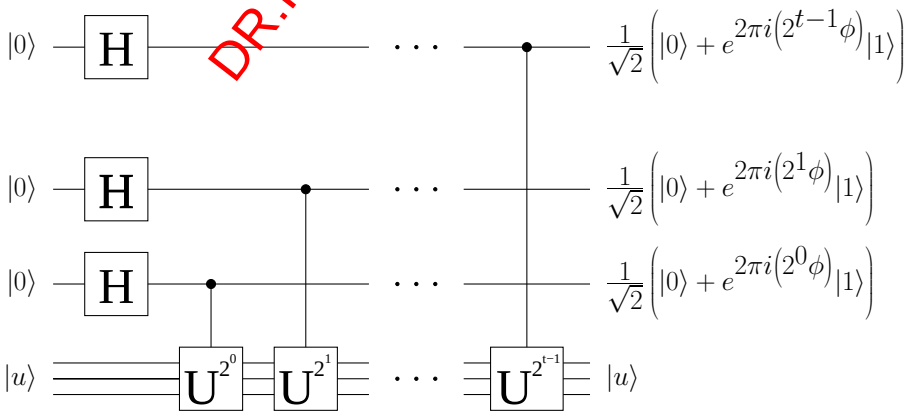


Figure 8.1: First stage of phase estimation.

Secondly, we apply the inverse Fourier transform

$$\sum_{j=0}^{2^t-1} e^{2\pi i\phi k}|k\rangle|u\rangle \rightarrow |\widetilde{\phi}_{t,u}\rangle|u\rangle. \tag{8.12}$$

The state $|\widetilde{\phi}_{t,u}\rangle$ is peaked around the value ϕ and for $t \rightarrow \infty$ the state converges to a δ-function.
Thirdly, we measure the first register, which gives us a value close to ϕ. The measured value has a probability of $1-\epsilon$ to be in a range $\phi \pm 2^n$ around ϕ. For the example $t = 4$, Figure 8.2 shows the probabilities of measuring the state j in the first register after applying the inverse quantum Fourier transform.

8.4.3 Performance

We want to derive expression 8.10. Let $b \in \{0, 2^t - 1\}$ be the integer such that $b/2^t = 0.b_1b_2...b_t$ is the best t bit approximation to ϕ. Thus, the difference $d = \phi - b/2^t$ satisfies $0 \leq d \leq 2^{-t}$.
The inverse quantum Fourier transform of equation 8.12 is given by

$$\frac{1}{\sqrt{2^t}}\sum_{k=0}^{2^t-1} e^{2\pi i\phi k}|k\rangle \rightarrow \frac{1}{\sqrt{2^t}}\sum_{k,l=0}^{2^t-1} e^{2\pi i\phi k}e^{-2\pi ikl/2^t}|l\rangle. \tag{8.13}$$

The amplitude of the state $|l\rangle$ is

$$\alpha_l = \frac{1}{2^t}\sum_{k=0}^{2^t-1}\left(e^{2\pi i\left(\phi-l/2^t\right)}\right)^k = \frac{1}{2^t}\left(\frac{1-e^{2\pi i\left(\phi 2^t-l\right)}}{1-e^{2\pi i(\phi-l/2^t)}}\right) \tag{8.14}$$

where we used the geometric sum.
Suppose the outcome of the measurement is m. We want to bound the probability of $|m-b| > \Delta$, where Δ is a positive integer characterizing the tolerance.

$$p\left(|m-b|>\Delta\right) = \sum_{\substack{-2^{t-1}<(b+l \mod N)\leq-(\Delta+1)\\(\Delta+1)\leq(b+l \mod N)\leq 2^{t-1}}} |\alpha_{b+l \mod N}|^2 = \sum_{\substack{-2^{t-1}<l\leq-(\Delta+1)\\(\Delta+1)\leq l\leq 2^{t-1}}} |\alpha_l|^2 \tag{8.15}$$

We can estimate $|\alpha_l|$ by

$$|\alpha_l| \leq \frac{2}{2^t\left|1-e^{2\pi i(d-l/2^t)}\right|} \leq \frac{2}{2^{t+1}\left(d-l/2^t\right)} \leq \frac{1}{l-2^t d} \tag{8.16}$$

Starting form expression 8.14, we use for the nominator that for any real θ, $\left|1-e^{i\theta}\right| \leq 2$ and for the denominator that $\left|1-e^{i\theta}\right| \geq 2|\theta|/\pi$ for $\theta \in [-\pi, \pi]$.

But, for $l \in (-2^{t-1}, 2^{t-1}]$, we have $2\pi\left(d - l/2^t\right) \in [-\pi, \pi]$. Using the fact that $0 \leq d2^t \leq 1$, one gets

$$\begin{aligned} p\left(|m-b| > \Delta\right) &\leq \frac{1}{4} \sum_{l=-2^{t-1}+1}^{-(\Delta+1)} \frac{1}{l^2} + \sum_{l=(e+1)}^{2^{t-1}} \frac{1}{(l-1)^2} \leq \frac{1}{2} \sum_{l=\Delta}^{2^{t-1}-1} \frac{1}{l^2} \\ &\leq \int_{l=\Delta-1}^{2^{t-1}-1} dl \frac{1}{l^2} = \frac{1}{2\left(\Delta-1\right)} \end{aligned} \tag{8.17}$$

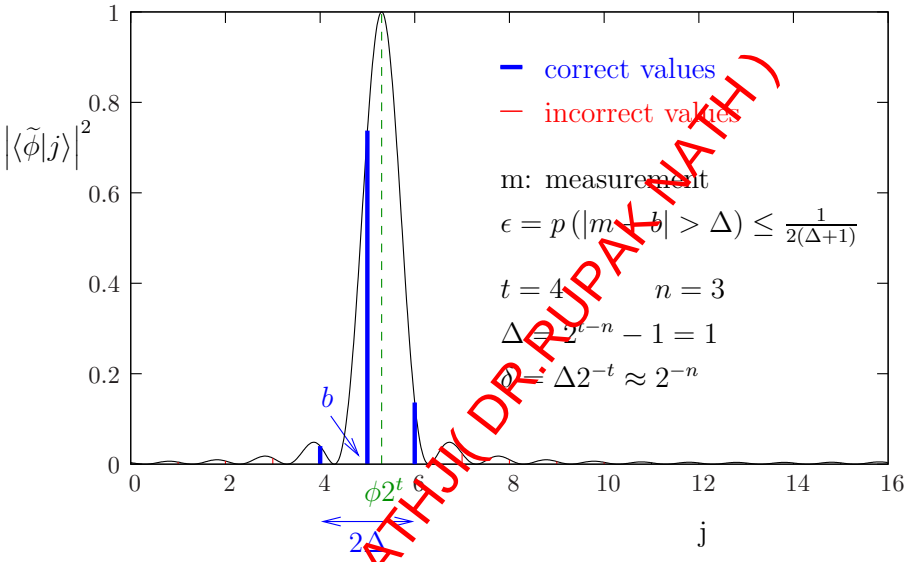


Figure 8.2: Measurement probabilities of the state $|\widetilde{\phi}\rangle$

We can identify $p\left(|m-b| > \Delta\right)$ with the probability that our algorithm fails, which we defined by ϵ. If we want to approximate ϕ with accuracy $\delta \approx 2^{-n}$ we choose our tolerance $\Delta = 2^{t-n} - 1$. This yields expression 8.10.

8.4.4 SUMMARY PHASE ESTIMATION

1. initial state $|0\rangle|u\rangle$
2. Hadamard gates $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|u\rangle$
3. apply black box $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle U^j|u\rangle$

$$= \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} e^{2\pi i j \phi_u} |j\rangle |u\rangle$$

4. apply inverse Fourier transform $\rightarrow |\widetilde{\phi_u}\rangle |u\rangle$
5. measure first register $\rightarrow \phi_u \pm \delta$ with $\delta \approx 2^{-n}$

8.4.5 ORDER-FINDING

Given two positive integers x and N , with $x < N$ and $\gcd(x, N) = 1$, we want to find the order of x modulo N, which is the least positive integer r, such that $x^r = 1 \pmod N$.
The order-finding problem is a phase estimation applied to the operator

$$U_{x,N}|y\rangle = |xy \mod N\rangle. \tag{8.18}$$

This operator is unitary. It has an inverse given by $U_{x^{-1},N}$, where x^{-1} is the inverse of $x \mod N$. This inverse exists and is unique since x and N are co-prime. Thus, $U_{x,N}$ maps one-to-one. We have that $\langle U_{x,N} z | U_{x,N} y\rangle = \langle xz \mod N | xy \mod N\rangle = \langle z|y\rangle$, which shows that $U_{x,N}$ is unitary.
The states defined by

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left(\frac{-2\pi s k}{r}\right) |x^k \mod N\rangle \tag{8.19}$$

where $0 < s \leq r$, are eigenstates of the operator $U_{x,N}$. The corresponding eigenvalues are given by $\exp(2\pi i s/r)$.
To use the phase estimation procedure, we should be able to prepare the eigenstate $|u_s\rangle$. This is not possible since we need to know the order r to construct such a state.
The phases of the above eigenstates are s/r, with different s for different eigenstates, but we are only interested in the denominator r. Therefore it is sufficient to choose a state in $\text{span}\{u_1, u_2, ...u_{r-1}\}$. (To apply the continued fraction algorithm explained in section 8.4.6, we need additionally $\gcd(r, s) = 1$, which means that the fraction s/r must be irreducible.)
The state $|1\rangle$ is an excellent candidate. It can easily be prepared and it is the equally weighted superposition of the above eigenstates

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle = |1\rangle. \tag{8.20}$$

The controlled -U^{2^j} operation can be represented as

$$\begin{aligned} |z\rangle|y\rangle &\rightarrow |z\rangle U^{z_t 2^{t-1}} ... U^{z_1 2^0}|y\rangle \\ &= |z\rangle|x^{z_t 2^{t-1}} ... x^{z_1 2^0} y \mod N\rangle \\ &= |z\rangle|x^z y \mod N\rangle. \end{aligned} \tag{8.21}$$

The operation $x^z y \mod N$ can be calculated classically in $O(L^3)$ steps.

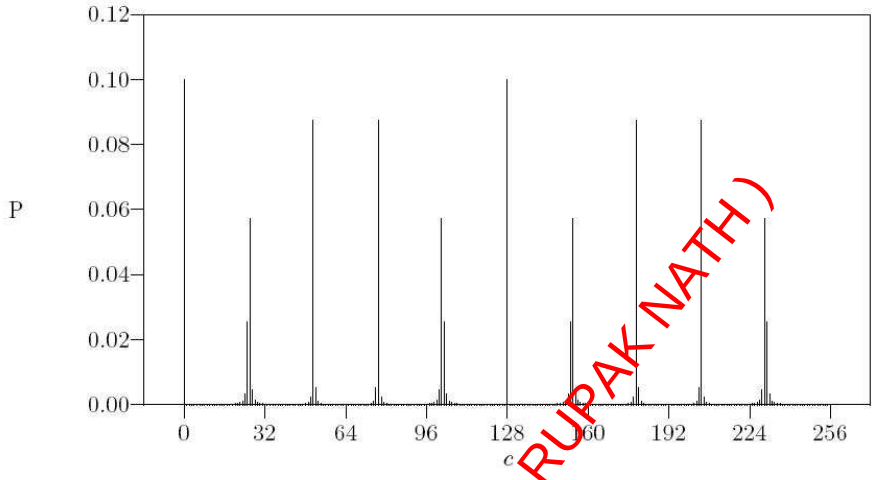


Figure 8.3: Observing probabilities of the order-finding. Range $[0, 256)$, Period $r = 10$, Distance between peaks $2t/r$.

The eigenvalues of the phase-estimation operator are $\exp(2\pi i s/r)$. Thus, the measurement of the phase gives us a value $s/r \pm \delta$, which is close to the fraction s/r. Extracting r form $s/r \pm \delta$ can be done using the continued fraction expansion.

8.4.6 Continued fraction algorithm

A finite continued fraction is an expression of the form

$$[a_0, ..., a_M] \equiv a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots + \cfrac{1}{a_M}}}}, \tag{8.22}$$

where $a_0, ..., a_M$ are positive integers. The ***m*th convergent** to this expression is defined as $[a_0, ..., a_m]$ for $0 \leq m \leq M$.
Any rational number x can be represented as a continued fraction by the following procedure. Let $a_0 = \lfloor x \rfloor$ and $x = a_0 + \xi_0$ for some $0 \leq \xi_0 < 1$. If $\xi_0 \neq 0$ then $a_1 = \lfloor 1/\xi_0 \rfloor$ and $1/\xi_0 = a_1 + \xi_1$ some $0 \leq \xi_1 < 1$. For rational numbers this procedure always terminates and we get $[a_0, ..., a_M]$. If we impose $a_M > 1$ we get

a unique representation. In the same way irrational numbers can be represented by infinite continued fractions.
For the sequence $a_0, ..., a_M$, the mth convergent $[a_0, ..., a_m] = p_m/q_m$ can be calculated inductively by $p_0 = a_0, q_0 = 1$ and $p_1 = 1 + a_0a_1, q_1 = a_1$, and for $2 \leq m \leq M$

$$\begin{aligned} p_m &= a_m p_{m-1} + p_{m-2} \\ q_m &= a_m q_{m-1} + q_{m-2}. \end{aligned} \tag{8.23}$$

Using $[a_0, ..., a_m] = [a_0, ..., a_{m-2}, a_{m-1} + 1/a_m]$ and induction in m proves this statement.
From the definition of p_m and q_m, it follows that p_m and q_m are increasing. $p_m = a_m p_{m-1} + p_{m-2} \geq 2p_{m-2}$ and similarly $q_m \geq 2q_{m-2}$. Thus, $p_m, q_m \geq 2^{\lfloor m/2 \rfloor}$ and $p, q \geq 2^{\lfloor M/2 \rfloor}$. The number of values M of the continued fraction expansion for a rational number $x = p/q$ is therefore of order $O(L)$ where L is the number of bits of $\max(p, q)$. Since round down and inversion are of order $O(L^2)$, the continued fraction $[a_0, ..., a_M]$ can be calculated in $O(L^3)$ operations.

Theorem 8.4.1. *Suppose that p/q is a rational number such that*

$$\left| \frac{p}{q} - x \right| \leq \frac{1}{2q^2} \tag{8.24}$$

Then p/q is a convergent of the continued fraction for x. Thus, it can be computed in $O(L^3)$ steps.

A proof of this theorem can be found in [3].
If we choose $t = 2L + 1 + \lceil \log(2 + 1/(2\epsilon)) \rceil$ and measure the first register, we get a value c/q in the range $[s/r - \delta, s/r + \delta]$, where $q = 2^t$ and $\delta \approx 2^{-2L-1}$. This means that c/q is equal to s/r up to $2L + 1$ bit. Thus,

$$\left| \frac{s}{r} - \frac{c}{q} \right| \leq 2^{-2L-1} \leq \frac{1}{2N^2} \leq \frac{1}{2r^2} \tag{8.25}$$

and we can use Theorem 8.4.1 to compute the continued fraction of s/r.
This continued fraction is unique. Suppose there are two different fractions $\frac{s}{r}$ and $\frac{s'}{r'}$. Then $\left|\frac{s'}{r'} - \frac{s}{r}\right| = \left|\frac{s'}{r'} - \frac{c}{q} + \frac{c}{q} - \frac{s}{r}\right| \leq \left|\frac{s'}{r'} - \frac{c}{q}\right| + \left|\frac{c}{q} - \frac{s}{r}\right| \leq \frac{1}{N^2}$. Assuming $s'/r' \geq s/r$ and using $r < N$, it follows that $\frac{s'}{r'} - \frac{s}{r} \leq \frac{1}{N^2}$, thus $s'r - r's = 0$ and $s'/r' = s/r$.
Form the continued fraction s and r can be calculated straightforwardly be inversion and addition.

The phase estimation procedure fails if the measurement projects to a state $|\widetilde{s/r}\rangle$ where s/r is not irreducible.
To solve this problem, the phase estimation procedure is run twice. Assume we obtain obtaining (r'_1, s'_1) and (r'_2, s'_2), which differ form the intended values (r, s_1) and (r, s_2) by $r'_1 n = r$ and $s'_1 n = s_1$, $r'_2 m = r$ and $s'_2 m = s_2$ respectively, $n, m \in \mathbb{N}$. If s_1 and s_2 have no common factor, then $\gcd(m, n) = 1$. It follows that

$$\begin{aligned} \operatorname{lcm}(r'_1, r'_2) &= \frac{r'_1 r'_2}{\gcd(r'_1, r'_2)} = \frac{r^2}{mn\gcd(r'_1, r'_2)} = \frac{r^2}{\gcd(mnr'_1, mnr'_2)} \\ &= \frac{r^2}{\gcd(mr, nr)} = \frac{r^2}{r\gcd(m, n)} = r. \end{aligned} \tag{8.26}$$

The probability that s_1 and s_2 have no common factor is given by

$$1 - \sum_{q \text{ prime}} p(q|s_1)\, p(q|s_2) \geq 1 - \sum_{q \text{ prime}} \frac{1}{q^2} \geq \frac{1}{4}, \tag{8.27}$$

where $p(p|s)$ is the probability of q dividing s.
Thus, running the phase estimation procedure several times, it is likely that we get a pair (s_1, s_2) such that $\gcd(s_1, s_2) = 1$, form which we can calculate r.

8.4.7 SUMMARY ORDER-FINDING

1. initial state $|0\rangle|1\rangle$
2. create superposition $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|1\rangle$
3. apply $U_{x,N}$ $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|x^j \mod N\rangle$
   $= \frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} e^{2\pi i s j/r} |j\rangle|u_s\rangle$
4. apply inverse Fourier transform $\rightarrow \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |\widetilde{s/r}\rangle u_s\rangle$
5. measure first register $\rightarrow s/r \pm \delta$
6. apply cont. frac. algorithm $\rightarrow r$

8.5 Related problems

8.5.1 Period-finding

Given a periodic function $f(x+r) = f(x)$ for $0 < r < 2^L$, and a black box U which performs a unitary transformation $U|x\rangle|y\rangle = |x\rangle|f(x) \oplus y\rangle$ ($\oplus$ denotes bitwise addition modulo 2), we want to find the period r. This can be done with one use of U and $O(L^2)$ operations.

Procedure:

1. initial state $|0\rangle|0\rangle$
2. create superposition $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{x=0}^{2^t-1} |x\rangle|0\rangle$
3. apply U $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{x=0}^{2^t-1} |x\rangle|f(x)\rangle$
   $\approx \frac{1}{\sqrt{r2^t}} \sum_{l=0}^{r-1} \sum_{x=0}^{2^t-1} e^{2\pi i l x/r} |x\rangle|\hat{f}(l)\rangle$
4. apply inverse Fourier transform $\rightarrow \frac{1}{\sqrt{r}} \sum_{l=0}^{r-1} |\widetilde{l/r}\rangle|\hat{f}(l)\rangle$
5. measure first register $\rightarrow l/r \pm \delta$
6. apply cont. frac. algorithm $\rightarrow r$

The approximate equality in Step 3 comes from the fact that 2^t may not be a multiple of r. The order finding problem is a problem of period-finding applied to the function $f(k) = x^k \mod N$.

8.5.2 Discrete logarithm

Given a prime number p, a generator a of $\mathbb{Z}_p^*$ and $b \in \mathbb{Z}_p^*$, we want to find s such that $a^s = b \pmod N$. To solve this problem we consider the function $f(x_1, x_2) = a^{sx_1+x_2} \mod p = b^{x1}a^{x_2} \mod p$, where x_1, x_2 are integers. The function f is periodic since $f(x_1+l, x_2-ls) = f(x_1, x_2)$ for all $l \in \mathbb{Z}$. Thus, the period is $(1, -s)$. The black box U transforms $U|x_1\rangle|x_2\rangle|y\rangle = |x_1\rangle|x_2\rangle|f(x_1, f_2) \oplus y\rangle$. The discrete logarithm can be calculated by one use of U and $O(\lceil \log r \rceil^2)$ operations.

Procedure:

1. initial state $|0\rangle|0\rangle|0\rangle$

2. create superpos. $\rightarrow \frac{1}{2^t}\sum_{x_1=0}^{2^t-1}\sum_{x_2=0}^{2^t-1}|x_1\rangle|x_2\rangle|0\rangle$

3. apply U $\rightarrow \frac{1}{2^t}\sum_{x_1=0}^{2^t-1}\sum_{x_2=0}^{2^t-1}|x_1\rangle|x_2\rangle|f(x_1,x_2)\rangle U$

$$\approx \frac{1}{\sqrt{r}2^t}\sum_{l=0}^{p-2}\sum_{x_1=0}^{2^t-1}\sum_{x_2=0}^{2^t-1}e^{2\pi i(slx_1+lx_2)/r}|x_1\rangle|x_2\rangle|\hat{f}(ls,l)\rangle$$

$$= \frac{1}{\sqrt{r}2^t}\sum_{l=0}^{p-2}\left[\sum_{x_2=0}^{2^t-1}e^{2\pi i(slx_1)/r}|x_1\rangle\right]\left[\sum_{x_1=0}^{2^t-1}e^{2\pi i(lx_2)/r}|x_2\rangle\right]|\hat{f}(ls,l)\rangle$$

4. inv. Fourier transf. $\rightarrow \frac{1}{\sqrt{r}}\sum_{l=0}^{p-2}|\widetilde{sl/r}\rangle|\widetilde{l/r}\rangle|\hat{f}(l)\rangle$

5. measurement $\rightarrow (sl/r \pm \delta, l/r \pm \delta)$

6. cont. frac. alg. $\rightarrow s$

Similarly to the factoring problem, the discrete logarithms is apparently difficult to compute, while the inverse problem of discrete exponentiation is not. This asymmetry is exploited to construct cryptographic systems like the ElGamal encryption, the Diffie-Hellman key exchange, or the Digital Signature Algorithm.

Bibliography

[1] R. L. Rivest, A. Shamir, and L. M. Adelman, Tech. Rep. MIT/LCS/TM-82 (1977), URL `citeseer.ist.psu.edu/rivest78method.html`.

[2] H. W. L. Von A. K. Lenstra, *The Development of the Number Field Sieve, Lecture Notes in Math. (1993) 1554* (Springer-Verlag, 1993).

[3] M. M. Nielsen and C. I. L., *Quantum Computation and Quantum Information* (Cambridge University Press, 2001).

[4] A. Ekert and R. Jozsa, *Shor's quantum algorithm for factorising numbers*, Review of Modern Physics pp. 733–753 (1996).

[5] P. W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, Soc. Ind. Appl. Math. J. Comp. pp. 1484–1509 (1997).

[6] E. Gerjuoy, *Shor's factoring algorithm and modern cryptography. an illustration of the capabilities inherent in quantum computers*, American Journal of Physics **73**, 521 (2005).

Chapter 9

Topologically-protected Quantum Computing

Andreas Gölzer
supervisor: Sebastian Huber

We look at 2-level systems and the basic mathematics of qubits, before introducing the Kitaev model, a model that uses the topological properties of tori to realize quantum storage.

9.1 Introduction

One of the major obstacles in realizing quantum computers is quantum decoherence. We have seen in previous chapters that fault-tolerant quantum computing is possible using error-correcting algorithms like Shor's algorithm, however these approaches have their problems, and work only if the gates are almost ideal. So we ask ourselves whether errors could instead be corrected directly on the physical level.

In classical storage this is nothing extraordinary. If we consider the magnetic storage in a hard drive, while individual spins may fluctuate due to thermal noise, the interaction with their neighbors provides fault tolerance in two ways: For one, if a spin is flipped, the measured average over the many spins forming a bit will hardly differ, and also the ferromagnetic interaction will force it to flip up eventually.

For quantum storage, achieving fault-tolerance is not that easy, but at least in theory possible. The Kitaev model presented in this chapter provides for the correction of local errors by storing a qubit in the degenerate ground state of a toric system.

9.2 Qubit

Before we start with the Kitaev model, we will discuss the quantum mechanics of two-level systems to understand decoherence. The qubit is a basic unit of quantum information, and similar to the classical case, has two states $|0\rangle$ and $|1\rangle$. However, unlike its classical equivalent it can be in a superposition of those two states.
What those two states are depends on the specific implementation. In principle every two levels of a nonharmonic quantum system can be used as a qubit. Qubits have been realized using a large variety of means, in this chapter we will consider the descriptive case of a spin system, where the two states are $|\uparrow\rangle$ and $|\downarrow\rangle$.

9.2.1 Pure states

In a two-level system the wave function is a superposition of the two basis states. However, we can demand it to be normalized, and we can also fix a global phase. After doing so, we can write the wave function as

$$|\psi\rangle = \cos(\theta)\,|\uparrow\rangle + e^{i\varphi}\sin(\theta)\,|\downarrow\rangle\,, \tag{9.1}$$

with two angles $\theta \in [0, \frac{\pi}{2}]$ and $\varphi \in [0, 2\pi]$. It is convenient to map the state vector of such a system to the Bloch sphere (Fig. 9.1)

$$S_x = \sin(2\theta)\cos(\varphi), \tag{9.2}$$
$$S_y = \sin(2\theta)\sin(\varphi), \tag{9.3}$$
$$S_z = \cos(2\theta). \tag{9.4}$$

The time evolution of that system is given by the Schrödinger equation

$$i\hbar\frac{\partial}{\partial t}\,|\psi\rangle = H\,|\psi\rangle\,, \tag{9.5}$$

where H is a 2×2 hermitian operator. In Matrix form, H consists of 4 complex numbers, however, the fact that H is hermitian places constraints on those elements, such that

$$H = \begin{pmatrix} a & c - id \\ c + id & b \end{pmatrix}. \tag{9.6}$$

Using the Pauli matrices, we can express H with just 4 real numbers, where $H = \frac{a+b}{2} + c\sigma^x + d\sigma^y + \frac{a-b}{2}\sigma^z$ with

$$\sigma^x = \begin{pmatrix} & 1 \\ 1 & \end{pmatrix}, \quad \sigma^y = \begin{pmatrix} & -i \\ i & \end{pmatrix}, \quad \sigma^z = \begin{pmatrix} 1 & \\ & -1 \end{pmatrix}. \tag{9.7}$$

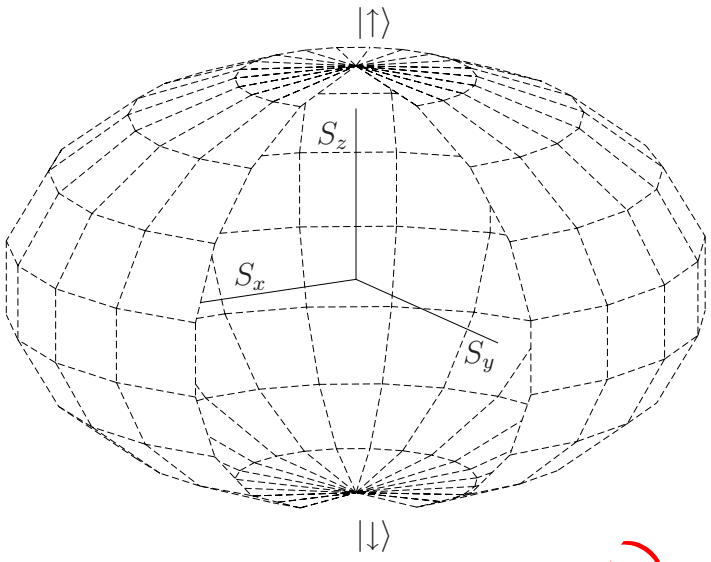


Figure 9.1: The Blochsphere contains all states of a two-level quantum system. Pure states live on the surface, while the interior is filled with mixed states.

The Pauli matrices are a convenient basis for calculations in 2-level systems. If we square a Pauli matrix, the result is the identity operator. Therefore their eigenvalues are ± 1. We identify $|\uparrow\rangle$ with the eigenstate to eigenvalue $+1$ of σ^z and similarly $|\downarrow\rangle$ with its eigenstate to eigenvalue -1.
The complete structure of the Pauli matrices can be expressed by $\sigma^j\sigma^k = \delta_{jk} + i\varepsilon_{jkl}\sigma^l$, or equivalently by the commutators and anticommutators

$$[\sigma^j, \sigma^k] = 2\varepsilon_{jkl}\sigma^l, \tag{9.8}$$

$$\{\sigma^j, \sigma^k\} = 2\delta_{jk}. \tag{9.9}$$

We use the notation of scalar products of Pauli matrices and three-dimensional vectors, and using that we can express the Hamiltonian,

$$H = E_0 - \vec{B}\vec{\sigma} = E_0 - B_x\sigma_x - B_y\sigma_y - B_z\sigma_z. \tag{9.10}$$

From that equation and the mapping in Eq. (9.2) one could derive the Bloch equation, Eq. (9.18), however the derivation is actually simpler if one considers mixed states.

9.2.2 Mixed states

The problem with the description of a qubit by pure states is that interaction with the environment is one of the major problems of present-day quantum computing.

We could achieve an approximation for that interaction by considering a larger Hilbert Space, $\mathcal{H} = \mathcal{H}_{\text{qubit}} \otimes \mathcal{H}_{\text{environment}}$, but then we would no longer have a simple 2-dimensional Hilbert space and would have to choose a basis for the environment. We will therefore use the density matrix formalism to describe the qubit.
In the density matrix formalism, the state of the system is described by a positive-definite matrix ρ with trace one:

$$\rho = \sum_j \rho_j \left|\psi_j\right\rangle \left\langle\psi_j\right| \tag{9.11}$$

The states ψ_j are pure and normalized, but not necessarily basis states; and the coefficients ρ_j are positive real numbers that add up to one. One can think of them as the probability of the system being prepared in the state ψ_j.
The expectation value of an operator, take for example σ^x, is defined by the trace of the product with the density matrix as in Eq. (9.11),

$$S_x = \left\langle\sigma^x\right\rangle_\rho = \text{Tr}\left(\rho\sigma_x\right) = \sum_j \rho_j \text{Tr}\left(\left|\psi_j\right\rangle \left\langle\psi_j\right| \sigma^x\right) = \sum_j \rho_j \text{Tr}\left(\left\langle\psi_j\right| \sigma^x \left|\psi_j\right\rangle\right) \tag{9.12}$$

$$= \sum_j \rho_j \left\langle\psi_j\right| \sigma^x \left|\psi_j\right\rangle = \sum_j \rho_j \left\langle\sigma^x\right\rangle_{\psi_j}. \tag{9.13}$$

We have used the fact we can swap the arguments of the trace, and that the trace of a number is just a number, in that case the expectation value if the system is in the pure state ψ_j.
If we calculate the spin vector in that way for pure states where $\rho = \left|\psi\right\rangle \left\langle\psi\right|$, then the mapping Eq. (9.2) is consistent with the new one for density matrices.
We are interested in where to map mixed states. One can show that the density matrix is completely determined by the spin vector:

$$\rho = \frac{1}{2}\left(1 + \vec{S}\vec{\sigma}\right) \tag{9.14}$$

If we calculate the determinant of ρ using that relation, we see that $\det(\rho) = (1 - \vec{S}^2)/4$. Since ρ is a positive definite matrix, the determinant is bigger than zero and therefore $\vec{S}^2 < 1$. Again we can map all states to the Bloch sphere, where pure states live on the surface and mixed states in the interior. One can furthermore show that the purity $r := \text{Tr}\left(\rho^2\right)$ satisfies $r = \frac{1}{2}\left(1 + \vec{S}^2\right)$, and therefore continuously decreases as the spin vector approaches the center of the sphere.
We are interested in the time evolution of the density matrix. By applying the Schrödinger equation, Eq. (9.5), and its complex conjugate to time derivative of

Eq. (9.11), we can derive the von Neumann equation

$$i\hbar\dot{\rho} = [H, \rho]. \tag{9.15}$$

It is interesting to note that while we got a different time evolution for the state vector if H contained a term proportional to unity, for density matrices this is not the case. It is a useful formula for Pauli matrices that

$$(\vec{a}\vec{\sigma})\left(\vec{b}\vec{\sigma}\right) = \vec{a}\vec{b} + i\left(\vec{a}\wedge\vec{b}\right)\vec{\sigma}. \tag{9.16}$$

Let us now use that equation to derive the time evolution for the spin vector. As we have seen before, we can write $H = -\vec{B}\vec{\sigma}$, and starting from the von Neumann equation, Eq. (9.15), and using Eq. (9.14) and Eq. (9.16) we find the Bloch equation,

$$\frac{i\hbar}{2}\dot{\vec{S}}\vec{\sigma} = i\hbar\dot{\rho} = [-\vec{B}\vec{\sigma}, \rho] = \frac{1}{2}[\vec{S}\vec{\sigma}, \vec{B}\vec{\sigma}] = i\left(\vec{S}\wedge\vec{B}\right)\vec{\sigma} \tag{9.17}$$

$$\overset{\sigma^j \text{ basis}}{\Longrightarrow} \dot{\vec{S}} = \frac{2}{\hbar}\vec{S}\wedge\vec{B}. \tag{9.18}$$

Using it, we no longer have to worry about the density matrix, and can describe the complete system with the spin vector.
We now assume a magnetic field $\vec{B} = B\vec{e}_z$. This removes the degeneracy of the energy eigenstate, with $|\uparrow\rangle$ now the state with the lowest energy and $|\downarrow\rangle$ the state with the highest energy. If we insert that field into the Bloch equation, we see that now the spin vector oscillates around the ground state,

$$\dot{\vec{S}} = \frac{2B}{\hbar}\begin{pmatrix} S_y \\ -S_x \\ 0 \end{pmatrix}. \tag{9.19}$$

Now we add decay and decoherence phenomenologically, thereby making sure the system will relax to the ground state in time, while simultaneously rotating around the ground state,

$$\dot{\vec{S}} = \frac{2B}{\hbar}\begin{pmatrix} S_y \\ -S_x \\ 0 \end{pmatrix} - \begin{pmatrix} \frac{S_x}{T_2} \\ \frac{S_y}{T_2} \\ \frac{S_z-1}{T_1} \end{pmatrix}. \tag{9.20}$$

The relaxation time T_1 is responsible for energy relaxation and describes how fast the system will return to the ground state. The decoherence time T_2 describes how fast the information in the spin vector perpendicular to the axis of the magnetic field is lost.

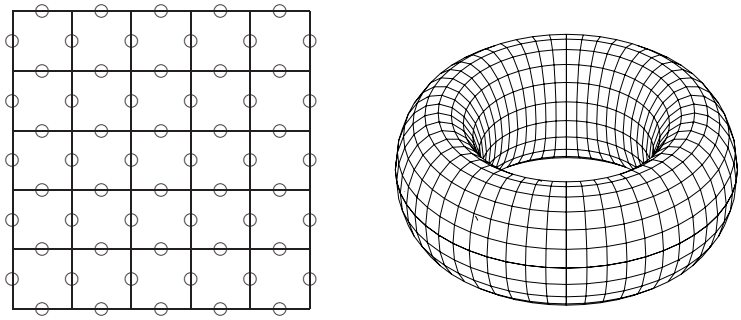


Figure 9.2: (left) The Kitaev model lattice: We place spins (marked by circles) on all connection lines between vertices. By connecting the left side to the right one, and the upper side to the lower one, we achieve the topology of a torus (right).

Those decay times are rather small in present-day qubit implementations and T_2 is typically in the range of about 100 ns [1], T_1 is in the same range for most qubit types, but can be significantly larger for well-isolated cold atom qubits.
Arbitrary operators on such a system can be realized by time-dependent magnetic fields. By tuning a time-dependent field to resonance with the oscillation around the ground state, one can use fields smaller than the one providing the energy difference between the two basis states.

9.3 Kitaev model

To get a model with longer decoherence times, we create one, in which the decoherence times of the qubit will depend on some macroscopic properties. In the Kitaev model the decoherence times will depend on the diameters of a torus.
The Kitaev model arises from theoretical considerations; it was proposed [2] by Kitaev to have the properties one wants to have from a qubit, such as the aforementioned decoherence times, and a stability against errors. There is no physical system that has those properties yet.
The Hilbert space $\mathcal{H}$ of the model is a two-dimensional square lattice on a torus (see Fig. 9.2) with a spin or qubit attached to each connection line of the lattice. On each spin we have Pauli matrices as operators, now with an additional index for the site they are operating on.

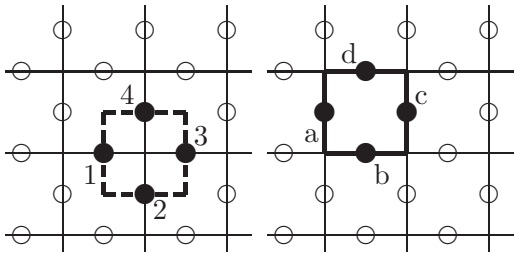


Figure 9.3: To the left a vertex operator $A_s = \prod_{j \in \text{star } s} \sigma_j^x = \sigma_1^x \sigma_2^x \sigma_3^x \sigma_4^x$, to the right a plaquette operator $B_p = \prod_{j \in \text{plaquette } p} \sigma_j^z = \sigma_a^z \sigma_b^z \sigma_c^z \sigma_d^z$. In this chapter, solid lines, which will always connect vertices, represent σ^z operators, while dashed lines, which will always connect centers of plaquettes, represent σ^x operators.

9.3.1 Kitaev Hamiltonian

On $\mathcal{H}$ we define (see Fig. 9.3) the vertex operators A_s, which flip the spins on all sites connecting to the vertex s, and the plaquette operators B_p, which measure all spins on the boundary of the plaquette p.
Out of vertex and plaquette operators (see Fig. 9.3) we construct the Hamiltonian

$$H = -\sum_{\text{vertices } s} A_s - \sum_{\text{plaquettes } p} B_p. \tag{9.21}$$

To derive some properties for those operators in the Hamiltonian, we first note that Pauli matrices on different sites commute, while they have the usual properties from Eq. (9.8) if they are on the same site. From their properties follow the properties of the vertex and plaquette operators:

- $A_s^2 = 1$, $B_p^2 = 1$: Since the Pauli matrices on different sites commute, we can rearrange the Pauli matrices in the product and get $A_s^2 = \prod_{j \in \text{star } s} \left(\sigma_j^x\right)^2 = 1$. The same holds for the plaquette operators. From that follows that the eigenvalues are ± 1.

- $\prod_s A_s = 1$, $\prod_p B_p = 1$: Due to the toric topology, those two products are finite, and each spin is part of exactly two vertex and plaquette operators. Therefore, for each spin both products contain just a squared Pauli matrix, which is one.

- $[A_s, A_t] = 0$, $[B_p, B_q] = 0$. Since Pauli matrices commute if they are not on the same site or if they are of the same type, vertex and plaquette operators commute with themselves.

- $[A_s, B_p] = 0$: If one looks at their definition in Fig. 9.3, one sees that vertex and plaquette operators have either none or two edges in common. In the first case, they commute since Pauli matrices commute if they are not on the same site. In the second case, we can swap all the Pauli matrices on different sites:

$$\begin{aligned}
[A_s, B_t] &\propto [\sigma_1^x \sigma_2^x, \sigma_1^z \sigma_2^z] && (9.22)\\
&= \sigma_1^x \sigma_2^x \sigma_1^z \sigma_2^z - \sigma_1^z \sigma_2^z \sigma_1^x \sigma_2^x && (9.23)\\
&= \underbrace{\{\sigma_1^x, \sigma_1^z\}}_{=0} \sigma_2^x \sigma_2^z - \sigma_1^z \sigma_1^x \sigma_2^x \sigma_2^z - \sigma_1^z \sigma_2^z \sigma_1^x \sigma_2^x && (9.24)\\
&= -\sigma_1^z \sigma_1^x \underbrace{\{\sigma_2^x, \sigma_2^z\}}_{=0} + \sigma_1^z \sigma_1^x \sigma_2^z \sigma_2^x - \sigma_1^z \sigma_2^z \sigma_1^x \sigma_2^x = 0 && (9.25)
\end{aligned}$$

We can generalize that last result: Every product of σ^z matrices commutes with another product of σ^x matrices, if the two products have an even number of sites in common. Should they have an odd number of sites in common, the two products anticommute.
This motivates the notation we used of connecting vertices with solid lines denoting σ^z operators. If we form a closed loop by always connecting vertices that way, the corresponding product of σ^z matrices will commute with all vertex operators. Similarly we can create so-called cuts, closed structures connecting plaquette centers, and operating with σ^x on the sites they cross, and such structures will commute with all plaquette operators. Furthermore all cuts will commute will all loops.

9.3.2 Ground state

Since the Hamiltonian, Eq. (9.21), is a sum of commuting operators, a basis exists where all vertex and plaquette operators are diagonal. So a state $|\xi\rangle$ which is the eigenstate to eigenvalue $+1$ of all vertex and plaquette operators exists. We will see that in fact four such states exist. We can construct one of those eigenstates as follows:

- Let us start with $|\text{all } \uparrow\rangle$, this is already an eigenstate to eigenvalue $+1$ of all B_p.

- Construct $|\phi_1\rangle = \frac{1}{\sqrt{2}}(A_1 + 1)\,|\text{all } \uparrow\rangle$, this is still an eigenstate to $+1$ of the B_p operators since $[B_p, A_1] = 0$. It is furthermore an Eigenstate to eigenvalue $+1$ of A_1 since $A_1(A_1 + 1) = (1 + A_1)$ because $A_1^2 = 1$.

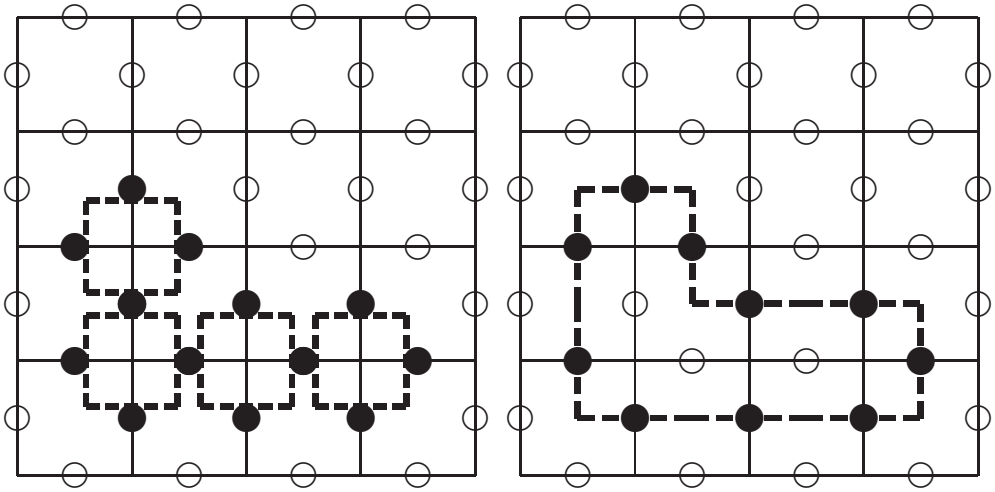


Figure 9.4: If we look at a product of neighboring vertex operators, the sites which are contained in two of those operators will be flipped twice, so the product leaves them alone. The remaining contour forms a cut, a product of σ^x along connection lines between plaquette centers.

- We can do the same for A_2: Construct $|\phi_{12}\rangle = \frac{1}{\sqrt{2}}(A_2+1)|\phi_1\rangle$, this is still an eigenstate to $+1$ of all B_p and A_1, and also of A_2.
- Repeat for the other vertices.

Following this procedure, we arrive at a ground state $|\xi\rangle$ and also get the operator $\mathcal{A}$ that projects any state to a state that is an eigenstate of all vertex operators (up to the normalization constant)

$$|\xi\rangle = \frac{1}{\sqrt{2^N}} \underbrace{\left[\prod_{\text{vertices } s} (A_s+1)\right]}_{=:\mathcal{A}} |\text{all } \uparrow\rangle\,. \tag{9.26}$$

We can check that $\mathcal{A}' = 2^{-N/2}\mathcal{A}$ is a projector by calculating its square,

$$(\mathcal{A}')^2 = 2^{-N}\mathcal{A}^2 = \frac{1}{4^N}\left[\prod_{\text{vertices } s} \underbrace{(A_s+1)^2}_{=A_s^2+2A_s+1}\right] = 2^{-N/2}\mathcal{A} = \mathcal{A}'. \tag{9.27}$$

To view the ground state in terms of spins, we transform the operator $\mathcal{A}$ into the sum of all 2^N possible combinations of vertex operators. If we now look at one such product, we see in Fig. 9.4 that if the product contains two neighboring vertex operators, the spin between them gets flipped twice and we can describe a product of vertex operators by its contour - which will be a cut as introduced

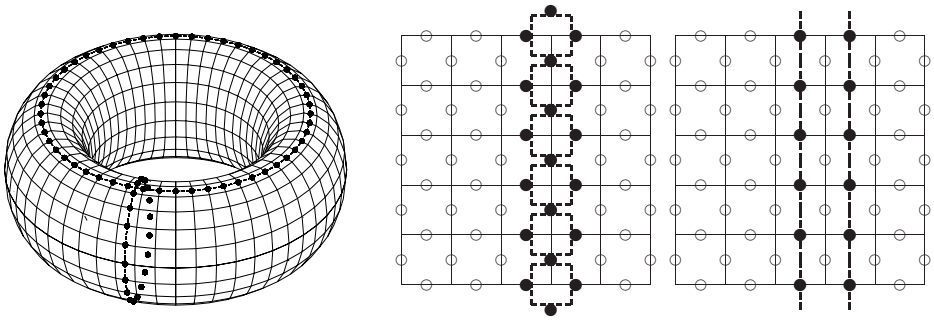
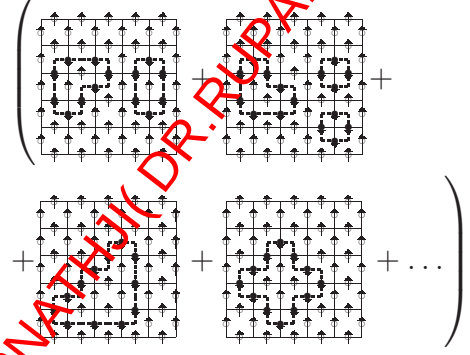


Figure 9.5: (left) The two non-contractible cuts on a torus. (middle and right) One cannot construct a single non-contractible cut as a product of vertex operators, only an even number.

before and, being that and also being just a product of vertex operators, commuting with all vertex and plaquette operators. In terms of spins, the ground state is a superposition of equal weights, where spins are flipped along every possible combination of cuts:

$$|\xi\rangle = \frac{1}{\sqrt{2^N}} \Bigg(+ + + \Bigg)$$



(9.28)

Ground state degeneracy

The construction of the ground state started with $|\text{all } \uparrow\rangle$, was that necessary? We required an eigenstate to all plaquette operators and can convince ourselves that we might as well have started with $A_1 |\text{all } \uparrow\rangle$ (or any combination of vertex or plaquette operators applied to $|\text{all } \uparrow\rangle$) and since $A_1\mathcal{A} = \mathcal{A}$ would have arrived at the same $|\xi\rangle$. But if we can find operators that commute with all plaquette and vertex operators, but not with all σ^x and σ^z, then the ground state is degenerate, and we can use those operators to reveal the other ground states $|\xi_j\rangle$.
On the torus, there are two non-contractible paths (see Fig. 9.5), let us consider a cut operator along such a path. This cut is no longer a product of vertex operators, if we try to construct products of vertex operators along the torus to create a non-contractible cut, we will always create an even number of such cuts. However, such a cut still commutes with all plaquette and vertex operators: Being

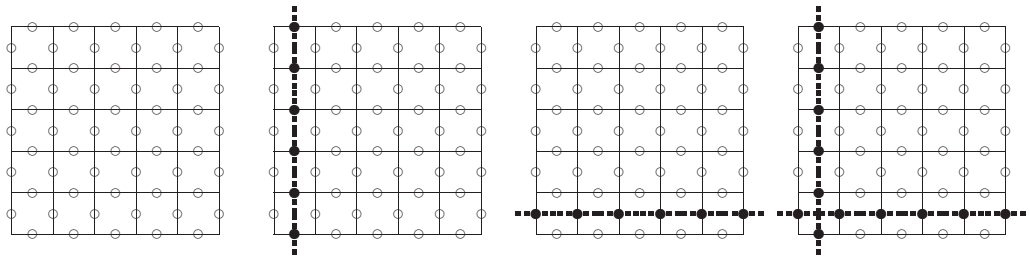


Figure 9.6: The four topological sectors of the ground states can be constructed by starting with $|\text{all } \uparrow\rangle$, applying $\mathcal{A}$, and then a combination of these non-contractible cut operators. From left to right: $|\xi_{\uparrow,\uparrow}\rangle = \mathcal{A}\,|\text{all } \uparrow\rangle$, $|\xi_{\downarrow,\uparrow}\rangle = X_1\mathcal{A}\,|\text{all } \uparrow\rangle$, $|\xi_{\uparrow,\downarrow}\rangle = X_2\mathcal{A}\,|\text{all } \uparrow\rangle$, $|\xi_{\downarrow,\downarrow}\rangle = X_1X_2\mathcal{A}\,|\text{all } \uparrow\rangle$.

a product of σ^x it commutes with all vertex operators; and like a contractible cut it has always an even number of sites in common with a plaquette operator, so if we swap those operators, we will anticommute different Pauli matrices an even number of times. We have found operators that commutes with all vertex and plaquette operators, yet are not a product of such operators.
These two operators, let us denote them X_1 and X_2, are a product of σ^x matrices and therefore $X_j^2 = 1$ and their eigenvalues are ± 1. This hints to them being a Pauli matrix on the ground state, which we will see in the next section. By Applying the four possible combinations of them to a ground state like the $|\xi\rangle$ constructed before, we can retrieve the other three ground states (see Fig. 9.6).

9.3.3 Operators on the ground state

Similarly to the non-contractible cuts one can construct non-contractible loops – products of σ^z along lines connecting vertices. Fig. 9.7 shows the relations between the non-contractible loops and cuts. Now, for each pair X_k, Z_k we have the relations $X^2 = 1$, $Z^2 = 1$ and $\{X, Z\} = 0$. These relations generate the structure of Pauli sigma matrices and require a system with at least two states to operate on.
To prove this, we choose $Y = -iXZ$. We immediately find $Y^2 = -XZXZ = X^2Z^2 = 1$. As for the products of different operators, let us as an example calculate $ZY = Z(-iXZ) = iZ^2X = iX$ and check that $YZ = (-iXZ)Z = -iX$. To show we have at least two states, take an eigenvector v of Z to eigenvalue α. $ZXv = -XZv = -\alpha Xv$, so Xv is an eigenvector of Z to $-\alpha$.
This justifies the construction in Eq. (9.6), we can view the degeneracy of the ground state as consisting of two qubits, which we can flip using the X operators and measure using the Z operators. For example $X_1\,|\xi_{\uparrow,\uparrow}\rangle = |\xi_{\downarrow,\uparrow}\rangle$ and $|\xi_{\downarrow,\uparrow}\rangle$ is

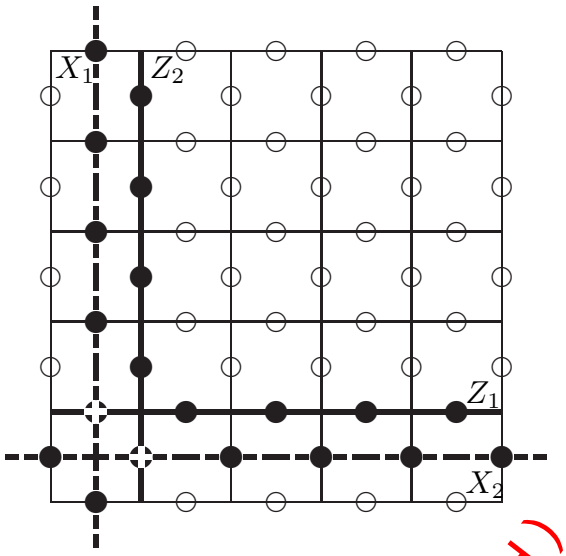


Figure 9.7: The Operators X_1, Z_1 and X_2, Z_2 operate on the ground state. If a spin is part of a cut and a loop, those two anticommute at that site. From that we get the relations $[X_1, X_2] = 0$, $[Z_1, Z_2] = 0$, $[X_1, Z_2] = 0$, $[X_2, Z_1] = 0$, $\{X_1, Z_1\} = 0$ and $\{X_2, Z_2\} = 0$.

an eigenstate to eigenvalue -1 of Z_1 and to +1 of Z_2. We know the structure of those operators, the question remains, can we realize them? For that we need to look at the excited states.

9.3.4 Excited states

First we note that the relations $\prod_s A_s = 1$ and $\prod_p B_p = 1$ prevent states where just one plaquette or vertex operator has eigenvalue -1, in all excited state pairs of plaquette operators or pairs of vertex operators have eigenvalue -1, and the minimal excitation energy is 4.

To create a state $|Z_{12}\rangle$, where A_1 and A_2 have eigenvalue -1, we look at its generation operator $S_c^z = \prod_{j \in c} \sigma_j^z$ where c is a z-like path from vertex 1 to vertex 2 (see Fig. 9.8). Note that while S_c^z depends on the specific path c, the state $|Z_{12}\rangle = S_c^z |\xi_k\rangle$ depends only on the homotopy class of that path and the topological sector k, that is, we may change the shape of the path by applying a product of plaquette operators to it, since $|\xi_k\rangle$ is an Eigenstate to eigenvalue +1 for all plaquette operators. Similarly we can create states $|X_{ab}\rangle$ where B_a and B_b have eigenvalue -1.

We can realize the operators X_k and Z_k by using the excitations. First we invest an energy of 4 and create a pair of particles, then we move one around the torus,

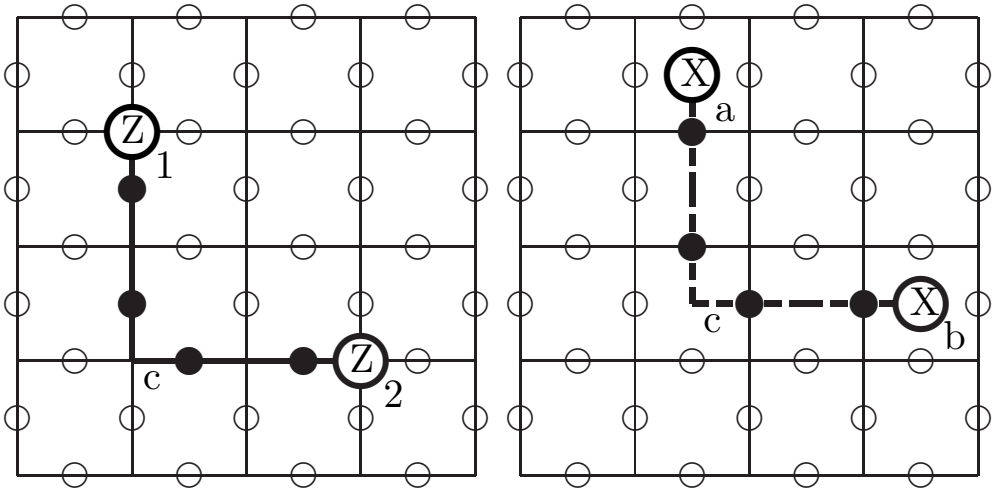


Figure 9.8: Excitation generation operators for an z-like particle (left) and an x-like particle (right).

and annihilate it with the other one, thereby regaining the energy. By doing this, depending on the direction around the torus and the type of particle created, we can realize all four operators X_1, Z_1 and X_2, Z_2.
With those operators, we can realize simple quantum gates that operate on the ground state. Unfortunately, only simple ones, namely the ones that correspond to identity or the sigma matrices, but not an arbitrary superposition of those.

9.3.5 Statistics

Depending of the homotopy class of the connection line, for any excitation with two particles at two fixed positions we have 4 states. That the connection lines are important is something not known from traditional fermions and bosons, so the particles must not necessarily fall in one of those categories.
If we consider only z-like particles, they are bosons, since the generation and movement operators commute, similarly if we have only x-like particles,

$$[S_l^z, S_{l'}^z] = 0, \tag{9.29}$$
$$[S_c^x, S_{c'}^x] = 0. \tag{9.30}$$

However, their mutual statistics show anyonic behavior. If we look at two pairs of particles, one x-like pair generated by S_s^x, one z-like pair generated by S_t^z, and move one z-like particle around one x-like particle and back to its original position by applying a loop operator $S_{t_l}^z$, we will see a change to the phase. The ground state is an eigenstate to $+1$ of the loop operator $S_{t_l}^z$, so we move that

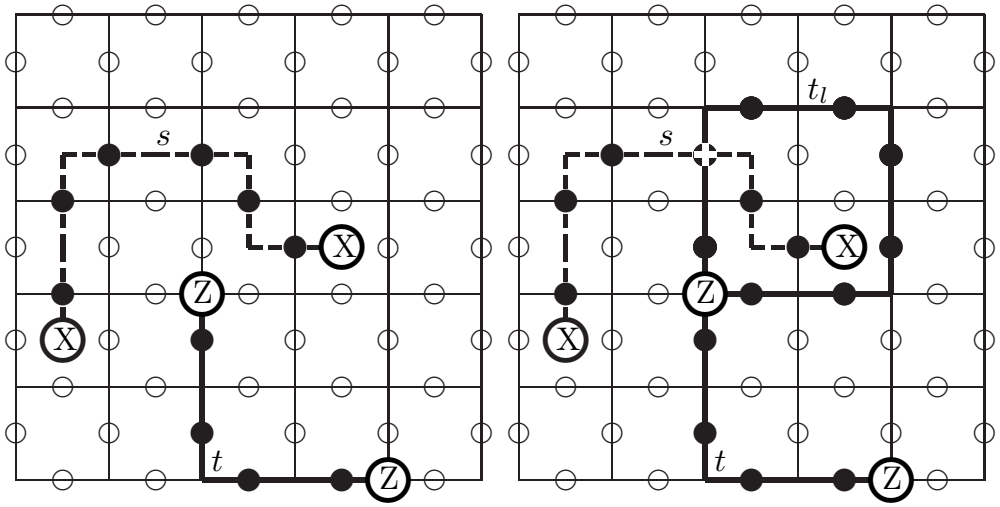


Figure 9.9: Moving an x-like particle around a z-like particle. The initial state is shown on the left side, the final state on the right.

operator to the right and have to anticommute it with S_s^x:

$$\begin{aligned} |\text{initial}\rangle &= S_t^z S_s^x |\xi\rangle \\ |\text{final}\rangle &= S_{t_l}^z |\text{initial}\rangle = S_{t_l}^z S_t^z S_s^x |\xi\rangle \\ &= S_t^z \left(\{S_{t_l}^z, S_s^x\} - S_s^x S_{t_l}^z \right) |\xi\rangle \\ &= - |\text{initial}\rangle \,. \end{aligned}$$

In the final state all the particles are at the same places, but the overall phase has changed. In a one-dimensional way, we can think of it as exchanging a x-like particle with a z-like particle twice, a process that does not change the phase for fermions and bosons. This means that the particles are neither one, but (abelian) anyons. Anyons are not possible in three-dimensional systems, they require topologically distinct ways of moving particles around each other. Anyons are deeply connected to the degeneracy of the ground state on a torus [3].

9.3.6 Errors

The system was invented to correct errors. So what happens if several spins are flipped? Since the sigma matrices and one form a basis of all 2×2 matrices, a general error can be described by an operator

$$E = \prod_{j \in I = \{\alpha_1, \cdots, \alpha_n\}} \vec{a}_j \vec{\sigma}_j \tag{9.31}$$

Using the relation $\sigma^y = i\sigma^x\sigma^z$ we can view that as a superposition of excited states as shown in previous sections. If we couple the system to a cold thermal

bath, we can expect those excitations to relax over time and will eventually reach the ground state again. As long as the error E does not contain a non-contractible loop or cut, that will be the same ground state as the initial one. The error E can only contain one of the X_j or Z_j operators if its support I has as least as many elements as needed to cross the smaller of the two toric circumferences. However, as long as the errors are random and the circumferences are large enough, the system is far better than that at error correction: The problem is then related to percolation, and as long as $|I| < p_c N$, where p_c is some threshold value and N the total number of sites, the error E will likely not contain any non-contractible loop or cut – the system is protected against almost any random error.
If we look at a perturbed system, the excitations shown in Fig. 9.8 are no longer eigenstates of the system. It is a convenient approximation to describe the dynamics of the excitations by applying the Schrödinger equation for some finite effective mass to the quasiparticles, where that mass might be different for x-like and z-like particles.
In the ground state, no real quasiparticles exist. Virtual pairs of quasiparticles however may be created, and those can tunnel around the torus. So the effective Hamiltonian will contain terms in Z_i and X_i, where the proportionality constants are $\exp(-\sqrt{2m_{x/z}\Delta E}L_{1/2})$. Therefore, the distortions decay exponentially with the torus size.

9.4 Concluding remarks

In this chapter we have presented a way to store two qubits in the degenerate ground state on a torus. We have realized operators on that qubit using excited states and seen how the system is stable against errors. By using tori of higher genus, one can store even more qubits in the ground state, so the system is scalable. However, there are some shortcomings of the model: We cannot create arbitrary gates with the excitations, and more importantly, if and to what extend it is realizable is not yet known.

Bibliography

[1] P. Bertet, I. Chiorescu, G. Burkard, K. Semba, C. Harmans, D. P. DiVincenzo, and J. E. Mooij, *Relaxation and dephasing in a flux-qubit* (2004), cond-mat/0412485.

[2] A. Kitaev, *Fault-tolerant quantum computation by anyons*, Annals of Physics **303**, 2 (2003).

[3] T. Einarsson, *Fractional statistics on a torus*, Phys. Rev. Lett. **64**, 1995 (1990).

Chapter 10

Implementing Topologically-Protected Quantum Computers

Markus Baden
supervisor: Fabian Hassler

We review the Ioffe proposal on the implementation of topologically-protected quantum computers. After a short introduction to the quantum dimer model, the mechanism is sketched that leads to dimer formation across Josephson junctions. The Josephson junctions are the building blocks of the implementation of topologically-protected quantum computing proposed by Ioffe et al. which we explain afterwards.

10.1 Introduction

Topologically-protected quantum computers use global properties of physicals system in order to achieve robust quantum computation. In particular, a system should show two features in order to be a candidate for a topologically-protected quantum computer. *i)* a gapped excitation spectrum between the two ground states, used as a computational basis, and the first excited states should exist. *ii)* the two ground states should lie in different topologically sectors of the Hilbert space describing the system.
The first property takes care that small perturbations do not introduce errors by exciting the system. The second property makes sure that the two states of the computational basis are only distinguishable by some global, topological, property

that can not be changed by local perturbations. Thus these local perturbation can not introduce errors.
Below, we will first describe the quantum-dimer model, a model system with these properties. Then, we will briefly discuss Josephson junctions and how dimers form across them. These Josephson junctions will be the building blocks of the Ioffe proposal discussed afterwards as an example of how topologically-protected computers might be implemented.

10.2 The Dimer Model

Following a concept suggested by Anderson [1], Rokshar and Kivelson (RK) analyzed a quantum hard-core gas composed of charged dimers [2]. One system where dimer formation is possible is a fully filled lattice with one electron per site. The energy of this system can be lowered if neighboring electrons form covalent bonds (singlets). These bound states of electrons are called dimers. Below, we will introduce dynamics by letting dimers repel or attract.
If one requires that each electron takes part in exactly one bond, the lattice is fully covered with dimers made up of two electrons. Since each electron takes part in only one bond, these dimers do not overlap and thus are called hard-core dimers.
It is possible to populate the lattice with dimers in different ways, each way is called a covering of the lattice. The detailed properties of the dimer model depend on the lattice chosen, e.g., whether it is a square or a triangular lattice. In particular, we will see that both properties *i)* and *ii)* are present on the triangular lattice, but not in the square lattice.
To start with, it is however convenient to work with the latter. The lattice is divided into squares with four lattice sites on the edges, so called plaquettes, as shown in Fig. 10.1. Depending on the covering, it is possible to have parallel dimers on some plaquettes. There exist two possibilities for being parallel, dimers can be either horizontally or vertically parallel.
Interesting dynamics are introduced in the dimer model by two competing terms. Letting parallel dimers either attract or repel gives rise to a potential term. A kinetic term is introduced by letting parallel dimers rotate, that is vertically parallel dimers can flip to horizontally parallel ones and vice versa. This leads to to the Hamiltonian

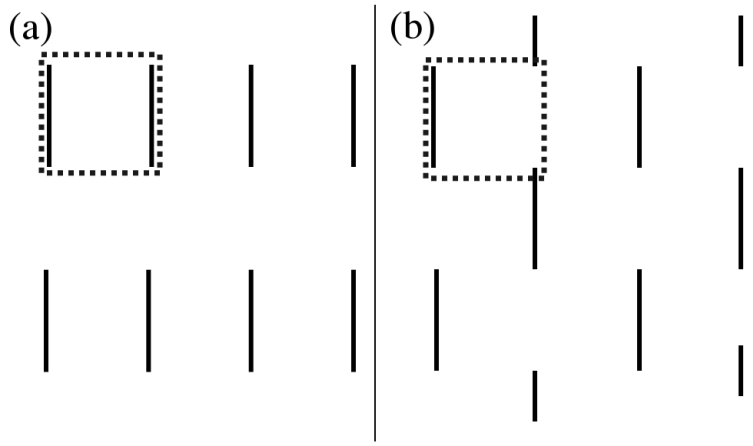


Figure 10.1: Coverings favored by competing dynamical terms of the quantum dimer model on the square lattice. Dashed squares are plaquettes. (a) Parallel covering favored by the kinetic term with all dimers being parallel. (b) Staggered covering favored by the potential term without any parallel dimers.

$$H = -t \sum_{\text{plaquettes}} \left[|=\rangle \langle \shortparallel| + |\shortparallel\rangle \langle =| \right] + v \sum_{\text{plaquettes}} \left[|=\rangle \langle =| + |\shortparallel\rangle \langle \shortparallel| \right]. \tag{10.1}$$

The kinetic parameter is $t \geq 0$ and $v \in \mathbb{R}$ is the potential parameter. For $v > 0$ dimers repel, for $v < 0$ they attract. From now on, we want to focus on the former case. Figure 10.1 shows the coverings that are ground states of the two different terms alone. The kinetic term favors a parallel covering where all plaquettes are flippable whereas the potential term favors a staggered covering where there are no plaquettes with parallel dimers on them.

Between the parallel phase for $v/t \to 0$ and the staggered phase for $v/t \to \infty$, we expect at least one transition. RK showed [2] that for $v/t = 1$, the so-called RK point, the ground state of the system is an equal superposition of all possible coverings

$$|\mathrm{RK}\rangle = \frac{1}{\sqrt{N_\mathrm{c}}} \sum_{c=1}^{N_\mathrm{c}} |c\rangle \tag{10.2}$$

where $|c\rangle$ represents a possible covering and N_c is the number of possible coverings.

We now sketch a proof that $|\mathrm{RK}\rangle$ is indeed the ground state. On the one hand, the total energy of the system has a lower bound which is seen by considering individual plaquettes. If there are parallel dimers on the plaquette, it has a potential energy of v and a kinetic energy of at best $-t$. The total energy E of the system thus satisfies $E \geq \min\{N_{\mathrm{p}}(v-t), 0\}$ where N_p is the number of flippable plaquettes. On the other hand, the energy of $|\mathrm{RK}\rangle$ is given by [3]

$$\langle \mathrm{RK}|H|\mathrm{RK}\rangle = (v-t)\,\langle n_{\square}\rangle = 0 \quad \text{at} \quad \frac{v}{t} = 1; \tag{10.3}$$

here

$$\langle n_{\square}\rangle = \sum_{c=1}^{N_{\mathrm{c}}} \sum_{\text{plaquettes}} \left[\langle c\,|\!=\!\rangle\,\langle\!=\!|\,c\rangle + \langle c\,|\,\shortparallel\,\rangle\,\langle\,\shortparallel\,|\,c\rangle\right]$$

is the expectation value of the number of flippable plaquettes.
The set of all possible covering $\{|c\rangle\}_{c=1}^{N_{\mathrm{c}}}$ forms an orthonormal basis [4]. The potential term in Eq. (10.3) is just by definition $v\,\langle n_{\square}\rangle$. For each covering with two horizontally parallel dimers on a plaquette there also exists the covering with two vertically parallel dimers on that plaquette and thus

$$-t\sum_{c=1}^{N_{\mathrm{c}}} \sum_{\text{plaquettes}} \left[\langle c\,|\!=\!\rangle\,\langle\,\shortparallel\,|\,c\rangle + \langle c\,|\,\shortparallel\,\rangle\,\langle\!=\!|\,c\rangle\right] = -t\,\langle n_{\square}\rangle\,.$$

Hence $|\mathrm{RK}\rangle$ saturates the lower bound for the energy at $v/t = 1$ and is one ground state of our system.
So far, we have only shown that $|\mathrm{RK}\rangle$ is the ground state for $v/t = 1$. Below we will use this state as a computational basis. In order for an implementation to be realizable, a range of possible values v/t has to exist for which states like $|\mathrm{RK}\rangle$ are ground states.
On the square lattice Rokshar and Kivelson argued that it is likely such a non-zero range of v/t exists [2]. However they showed that the excitation spectrum at the RK-point is gapless and thus the quantum dimer model on the square lattice does not show property *i)* we would like to have for our topologically-protected qubit.
In search for property *i)* in systems described by the quantum dimer model, Moessner and Sondhi did some numerical analysis and found strong indications that on the triangular lattice the quantum dimer model both exhibits a gapped excitation spectrum and that there is a non-zero range of v/t where ground states are given by state similar to $|\mathrm{RK}\rangle$ [3]. These findings where later confirmed by Ioffe *et al.* [5]. From now on, we therefore focus on the quantum dimer model on

the triangular lattice. The only difference to the square lattice is that there are three different plaquettes on which parallel dimers can occur [3].
The quantum dimer model on the triangular also shows different topological sectors (property *ii)*. Topological sectors of the Hilbert space are disconnected subspaces that are closed under the action of the Hamiltonian. In other words, just by applying the Hamiltonian it is not possible to leave a topological sector. By introducing cylindrical boundary conditions, the Hilbert space consists of two topologically separated sectors (see Fig. 10.2). The Hilbert space $\mathcal{H}$ separates into two space $\mathcal{H} = \mathcal{H}_{\text{even}} \oplus \mathcal{H}_{\text{odd}}$, where $\mathcal{H}_{\text{even/odd}}$ contains all coverings with even/odd dimer count along a given reference line γ going from one fixed boundary to the other [6].
This leads to a topological degeneracy of the ground state, that is $|\text{RK}\rangle$ is twofold degenerate on a cylindrical topology. One ground state is given by $|\text{RK}\rangle_{\text{even}}$, the equal superposition of all coverings with even dimer count along γ. The other by $|\text{RK}\rangle_{\text{odd}}$, the equal superposition of all coverings with odd dimer count along γ.
The subspaces $\mathcal{H}_{\text{even/odd}}$ are indeed topological sectors because the action of the Hamiltonian (10.1), i.e., flipping of two parallel dimers, cannot transform an even into an odd covering. By using the two RK-states as our computational basis, the system is thus a topologically-protected qubit.
Summarizing, we have seen that on the triangular lattice the quantum dimer model has a phase with the equal superposition state $|\text{RK}\rangle$ as ground state, that this ground state is protected by an energy gap, and that it is twofold degenerate due to the topology of periodic boundary conditions.

10.3 Implementing Dimers with Josephson Junctions

Brian D. Josephson first described the behaviour of two superconductors separated by a thin insulating layer [7]. These so-called Josephson junctions will be used below to implement dimers. All electrons are considered to be paired in Cooper pairs treated as single particles of charge $-2e$. By choosing the insulating layer sufficiently thin, there is a high probability that the Cooper pairs tunnel across the Josephson junction.
To see how dimers form across junction we analyze the circuit shown in Fig. 10.3(a) [8].
By tunneling, Cooper pairs can be brought on the Cooper-pair box via the junction without doing additional work to that equal to the electrostatic energy. Thus

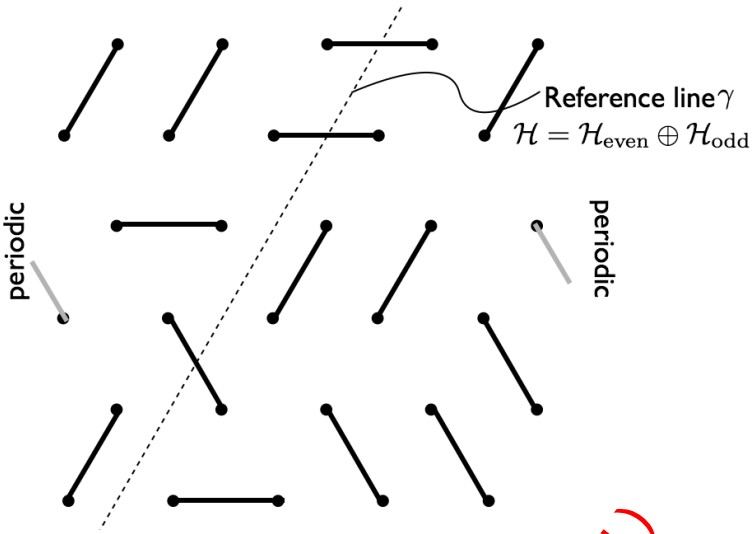


Figure 10.2: Cylindrical topology in the quantum dimer model on the triangular lattice. The Hilbert space separates into two subspace along a given reference line γ (dashed line). One subspace contains all coverings with even number of dimers along γ whereas the other contains all coverings with an odd number of dimers.

a charge imbalance is introduced in the system. At first the Cooper-pair box is neutral, so the charge on the island is $Q = 0$, the gate voltage V being fixed. Then some charges are brought on the Cooper-pair box via tunneling. This results in a flow of screening charges Q_0 around the circuit such that the energy of the system is minimized. In addition the charges on the Cooper-pair box will also rearrange. The electrostatic energy E_{el} of the system with charge imbalance is therefore given by

$$E_{\text{el}} = \frac{Q_0^2}{2C_0} + \frac{(Q_0 - Q)^2}{2C} - VQ_0, \tag{10.4}$$

where the screening charges $Q_0 \equiv Q_0(Q)$ are a function of the charge on the Cooper-pair box. Minimizing the energy with respect to Q_0 yields

$$\begin{aligned} \frac{\partial E_{\text{el}}}{\partial Q_0} &= \frac{Q_0}{C_0} + \frac{Q_0 - Q}{C} - V \equiv 0 \\ \Longrightarrow\ Q_0 &= \frac{C_0}{C_{\text{tot}}} Q + \frac{CC_0}{C_{\text{tot}}} V, \end{aligned}$$

where $C_{\text{tot}} = C_0 + C$ is the total capacitance of the system. By inserting Q_0 into Eq. (10.4) we get

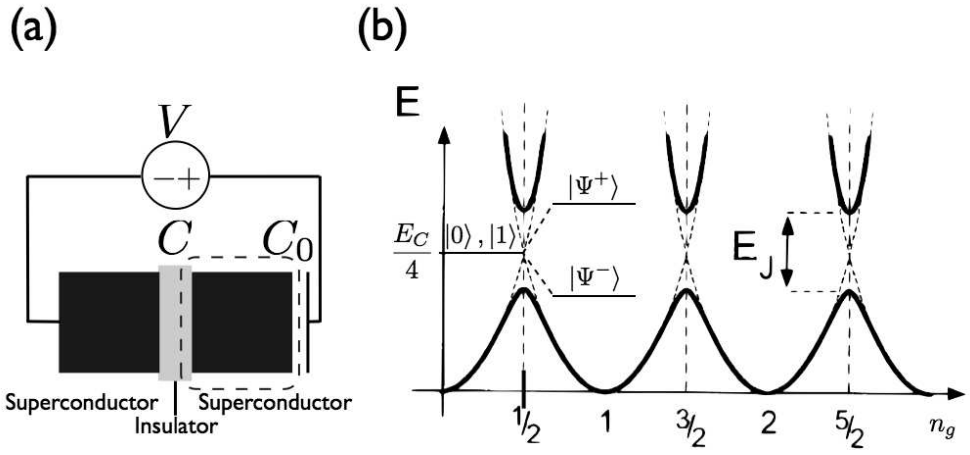


Figure 10.3: Cooper-pair box circuit. (a) Schematics of the circuit: One side of the Josephson junction is connected directly to a gate voltage V the other side is connected to that voltage via a capacitor C_0. This leads to a superconducting island (dashed line), referred to as a Cooper-pair box, which is separated from the environment by the insulator of the Josephson junction (acting as a capacitor C) and the insulator of the capacitor C_0. (b) Energy diagram as a function of the optimal charge n_g on the Cooper-pair box (solid lines). Tunneling leads to an avoided crossing of the electrostatic-energy parabolas (dashed lines). The degeneracy of the unperturbed ground states, e.g., $|0\rangle$ and $|1\rangle$ is lifted at crossing points. The new ground state $|\Psi^-\rangle$ is an equal superposition state representing a dimer across the junction.

$$E_{\text{el}} = \frac{(Q - C_0 V)^2}{2C_{\text{tot}}} - \frac{C_0 V^2}{2}$$

Since the Cooper-pair box is typically small, there is a large effect of putting one Cooper-pair more or less in the island so that the discrete nature of charge has to be taken into account, i.e., that the charge Q on the Cooper-pair box is quantized in units of $2e$. Since we are only interested in relative energies, we neglect the global energy shift $-C_0V^2/2$.

Going from the classical to the quantum picture, the charge on the Cooper-pair box is replaced by a charge operator

$$-Q \to -\hat{Q} = -2e\hat{n},$$

where $\hat{n} = \sum_n n \left|n\right\rangle \left\langle n\right|$ is a number operator with $n \in \mathbb{Z}$. This leads to the electrostatic Hamiltonian [9]

$$H_{\text{el}} = E_c \sum_n (n - n_g)^2 \, |n\rangle \langle n| \quad \text{with} \quad E_c = \frac{(2e)^2}{2C_{\text{tot}}}.$$

The external parameter $n_g = C_0 V/2e$ is controlled by the gate voltage V which is still being treated classically. It can be interpreted as the optimum charge it would require to have on the Cooper-pair box in order to get the minimum energy. Classically the charge on the Cooper-pair box n can take all values and thus the state with minimal energy is always $n = n_g$ and the minimal energy independent of n_g.

In the quantum picture n can only take integer values. The ground state energy of the system thus changes with n_g as shown in Fig. 10.3(b). For example in the range $n_g \in (-1/2, 1/2)$ the ground state is the $n = 0$ state with energy $E_{\text{el}} = E_c n_g^2$. At $n_g = 1/2$ the ground state becomes degenerate since both the $n = 0$ and the $n = 1$ state have the energy $E_c/4$.

To describe the Cooper-pair box more accurately, we have to take tunneling through the junction into account. This can be done by adding a Josephson Hamiltonian H_J to the system. The full model Hamiltonian is given by

$$H = H_{\text{el}} + H_J = E_C \sum_n (n - n_g)^2 \, |n\rangle \langle n| - \frac{E_J}{2} \sum_n \left[|n\rangle \langle n+1| + |n+1\rangle \langle n| \right],$$

where E_J is the Josephson energy characterizing the tunneling strength of the Josephson junction. The effect of tunneling is seen by treating it as a small perturbation and do first order degenerate perturbation theory. For simplicity we only consider tunneling of one Cooper pair at the time, so that there is for example no coupling between $n = 0$ and $n = 3$ or even higher n but only between $n = 0$ and $n = 1$. The two degenerate ground states at $n_g = 1/2$ are $|0\rangle$ (with $n = 0$) and $|1\rangle$ (with $n = 1$). First order perturbation theory at this point is equivalent to solving the eigenvalue problem

$$H_{\text{eff}}(a_0 \, |0\rangle + a_1 \, |1\rangle) = E_1(a_0 \, |0\rangle + a_1 \, |1\rangle),$$

with the effective Hamiltonian

$$\langle i|H_{\text{eff}}|j\rangle = \langle i|H_J|j\rangle \quad \text{with} \quad i, j = \{0, 1\}.$$

The calculation is analogous to that of electrons in a one dimensional periodic potential resulting in a band structure with a band gap. The splitting in our case is E_J (see Fig. 10.3(b)). In first order the ground state of the system is the equal superposition $|\Psi^-\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$.

One way to think about that state is that the Cooper pair is resonating between being on the island and not being on the island. This state is very similar to that of to electrons forming a singlet. So the $|\Psi^-\rangle$ state is a dimer across the junction. The process of tuning the external parameter n_g to a point where an equal superposition state of two charge states is the ground state is referred to as inducing charge frustration.

10.4 The Ioffe Proposal

Ioffe *et al.* proposed an implementation of a topologically-protected quantum computer based on an array of Josephson junctions [5]. The device is fabricated in such a way that the dynamics of the array is described by the quantum dimer model on the triangular lattice with cylindrical topology seen in Sec. 10.2. Dimer formation is reached by inducing charge frustration in a way similar to the one we have seen in Sec. 10.3.

Fig. 10.4(a) shows the basic arrangement of the Josephson junctions in the array. There are five experimental parameters depending upon the building process of the array that determine the properties of this system. The capacitive and Josephson energies of the junctions connecting short ends of the Y's on a hexagon, E_h^C and E_h^J, and the ones of the junctions connecting long ends at the links, E_l^C and E_l^J. The fifth parameter is the capacitive energy of the capacitor connected externally to each Y, E_Y^C.

We now want to outline how an effective low energy Hamiltonian similar to dimer Hamiltonian (see Eq. (10.1)) is reached. By choosing the capacitance C_l of the corresponding junctions to be large, the Y's around the hexagon are joined electronically together into one hexagonal array because, in order to minimize $E_h^C \sim C_h V_h^2/2$, there is no voltage drop across the junction.

Like in the previous section the system should be described in the charge basis. By setting the charging energy of the array to be large, the effect of additional Cooper pairs on the array is large and thus the charge basis is a good basis. The charging energy is given by $E_{\text{hex}} = E_Y^C/6 = Q_h^2/(12C_Y)$, six capacitors C_Y being connected in parallel. In order to get a large charging energy C_Y is chosen to be small.

By connecting a global gate voltage capacitive to the whole device (not shown in Fig. 10.4 (a)) and tuning it appropriately, only half a Cooper pair per hexagon is present on the device. In addition, charge frustration is induced by tuning the external parameter such that the state with one Cooper pair more on the island and the one with one Cooper pair less are degenerate. By treating tunneling

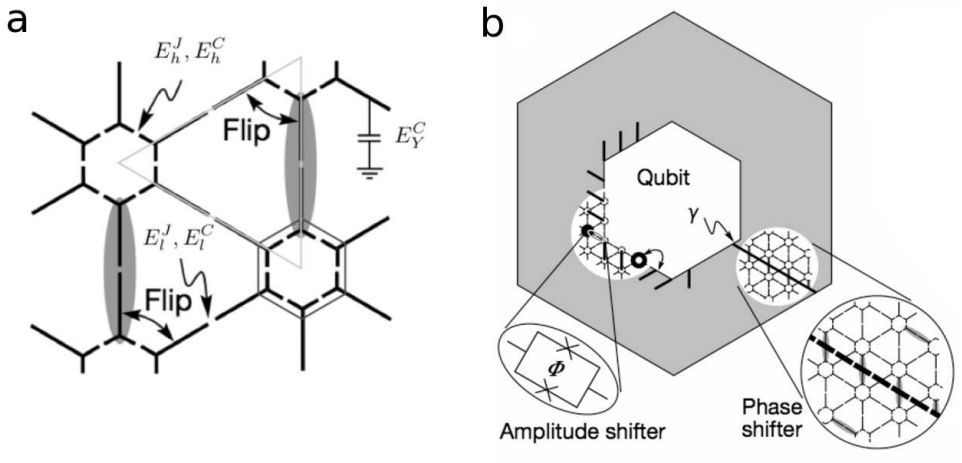


Figure 10.4: Josephson junction array proposed by Ioffe *et al.* (a) Basic arrangement. Each Y is a superconducting metal, the gaps in between are Josephson junctions. One type of Josephson junction connects long ends, another type short ends. Six Y's form a hexagonal shape and these hexagons are connected by the long ends of the Y's. Six of these long ends form a triangle and as we will see below this leads to a triangular dimer lattice. (b) Full device with cylindrical topology representing one qubit. One qubit operations are achieved by an amplitude and a phase shifter varying relative phase and amplitude of the equal superposition state of all coverings with odd resp. even dimers along a reference line γ.

through the link junction as perturbation, dimer formation across those link junctions is seen in first order. This process is in complete analogy to the one described in Sec. 10.3.
So far, our consideration have resulted in a triangular lattice fully filled with hard-core dimers. Dimer flipping is also inherent in our system but it is a process only seen in second order. Performing second order degenerate perturbation theory is equivalent to construct an effective Hamiltonian with matrix elements

$$\langle m|H_{\text{eff}}|n\rangle = \sum_{|l\rangle \notin \mathcal{H}_{\text{deg}}} -\frac{\langle m|\, H_J\, |l\rangle \, \langle l|\, H_J\, |n\rangle}{E_l^J},$$

where the intermediate state $|l\rangle$ does not lie in the degenerate subspace $\mathcal{H}_{\text{deg}}$ of the unperturbed states. In our model Hamiltonian (Eq. (10.1)) the dimer flipping amplitude $-t$ is given by the matrix element $-t = \langle \sqsupset |\, H\, | \, \shortmid \, \shortmid \rangle$. To see that this is indeed a process of second order, we note that both $|n\rangle = |\, \shortmid \, \shortmid \rangle$ and $\langle m| = \langle \sqsupset |$ are

superpositions of the unperturbed ground states and therefore lie in $\mathcal{H}_{\text{deg}}$. The intermediate state is the state where one of the Cooper pairs of the two vertically parallel dimers has tunneled through the junction on the hexagon. Applying the tunneling Hamiltonian H_J again results in the second Cooper pair to also tunnel through the junction on the hexagon. This state has some overlap with the horizontally parallel pair of dimers. Since tunneling through the junction on the hexagon is involved twice we conclude

$$t \propto \frac{1}{E_l^J} \sum_{|l\rangle \notin \mathcal{H}_{\text{deg}}} \langle \amalg | \, H_J \, |l\rangle \, \langle l| \, H_J \, |\mathbf{I}\ \mathbf{I}\rangle \propto \frac{(E_h^J)^2}{E_l^J}.$$

It can be shown [5] that dimer repulsion depends on the electrostatic properties and that the amplitude is given by

$$v \propto E_h^C \frac{C_l}{C_h} \left[\left(1 + \frac{C_Y}{C_h} \right) (1 + \frac{3C_Y}{C_h}) \right]^{-2}.$$

We have seen that an arrangement of Josephson arrays leads to system that can be described by the dimer Hamiltonian with tunable parameters v and t. In order to achieve cylindrical topology, devices as shown in Fig. 10.4 (b) are used.
Such a device has a twofold degenerate ground state of the $|\text{RK}\rangle$-type. This leads to a computational basis $\{|0\rangle, |1\rangle\}$ that is topologically protected.
The single qubit Hamiltonian representing all one qubit operations is

$$H_{\text{qubit}} = h_x \sigma_x + h_z \sigma_z.$$

The amplitude shifter $\propto h_x$ changes the relative amplitude of $|0\rangle$ and $|1\rangle$, e.g., rotating from $|1\rangle$ to $|0\rangle$ The phase shifter $\propto h_z$ shifts the relative phase of the two basis states, e.g. rotating from $|0\rangle + |1\rangle$ to $|0\rangle - |1\rangle$.
Since we chose $|0\rangle$ and $|1\rangle$ to be topologically protected, we cannot achieve these operations by simple means. For the amplitude shifter one introduces a switchable junction near the hole of the array with energy $\tilde{E}_l^J$. Breaking the dimer up at this junction and then moving the Cooper pair around the hole by M subsequent flips of the Cooper pair with a neighboring dimer before recombining the Cooper pair with the hole left behind is a M-th order process. Since this changes the dimer count along the reference line γ by one it takes an even covering to an odd covering. The amplitude shifter is

$$h_x \propto E_h^J \left(\frac{E_h^J}{\tilde{E}_l^J} \right)^M.$$

When one wants to perform a σ_x operation one changes $\tilde{E}_l^J$ in order to increase h_x. For the phase shifter a gated superconducting strip underneath the reference line is introduced which is connected to an external voltage. By turning that voltage on, the gated superconducting strip attracts the dimers above it. Since $|0\rangle$ and $|1\rangle$ have different dimer count along this reference line the relative energies are shifted leading to a shift in relative phase. For details and a scheme for implementing many qubit operations, we refer the reader to the original proposal [5].

10.5 Conclusion

We have shown that in principle a device could be realized that uses the equal superposition ground state of the quantum-dimer model as a topologically-protected computational basis. Whether the Ioffe proposal is indeed realizable or whether neglected terms lead to different behaviour of the system is not yet clear.
Thorough numerical studies by Albuquerque *et al.* [10] indicate that other terms neglected in the original Ioffe proposal are relevant for the description of the superconducting device. It is however an open question whether the model still exhibits the properties outlined in the last section if one takes the additional terms into account.
Another approach to answer the question, whether such devices are realizable, is to build the device. The basic techniques for manufacturing such devices are at hand and very recently a proof of concept of a device very similar to the one discussed above has been reported by Gladchenko *et al.*[11].
We conclude that promising use of topological protection in order to achieve robust quantum computation is not just a totally theoretical exercise but also a candidate for implementation. But in order to be implemented some problems in topologically-protected quantum computing have to be overcome, e.g., how exactly to carry out multi qubit operations.

Bibliography

[1] P. Anderson, *Resonating Valence Bonds: A New Kind of Insulator?*, Materials Research Bulletin **8**, 153 (1973).

[2] D. S. Rokhsar and S. A. Kivelson, *Superconductivity and the Quantum Hard-Core Dimer Gas*, Phys. Rev. Lett. **61**, 2376 (1988).

[3] R. Moessner and S. L. Sondhi, *Resonating Valence Bond Phase in the Triangular Lattice Quantum Dimer Model*, Phys. Rev. Lett. **86**, 1881 (2001).

[4] R. Moessner and S. L. Sondhi, *Resonating Valence Bond Liquid Physics on the Triangular Lattice*, Progress of Theoretical Physics Supplement **145**, 37 (2002), (cond-mat/0205029).

[5] L. B. Ioffe, M. V. Feigel'man, A. Ioselevich, D. Ivanov, M. Troyer, and G. Blatter, *Topologically Protected Quantum Bits from Josephson Junction Arrays*, Nature **415**, 503 (2002).

[6] A. Ralko, M. Ferrero, F. Becca, D. Ivanov, and F. Mila, *Zero-Temperature Properties of the Quantum Dimer Model on the Triangular Lattice*, Phys. Rev. B **71**, 224109 (2005).

[7] B. D. Josephson, *PossIble New Effects in Superconductive Tunnelling*, Physics Letters **1**, 251 (1962).

[8] M. Büttiker, *Zero-Current Persistent Potential Drop Across Small-Capacitance Josephson Junctions*, Phys. Rev. B **36**, 3548 (1987).

[9] V. Bouchiat, D. Vion, P. Joyez, D. Esteve, and M. H. Devoret, *Quantum Coherence with a Single Cooper Pair*, Physica Scripta Volume T **76**, 165 (1998).

[10] A. F. Albuquerque, H. G. Katzgraber, M. Troyer, and G. Blatter, *Engineering Exotic Phases for Topologically-Protected Quantum Computation by Emulating Quantum Dimer Models*, arxiv:0708.0191 (2007).

[11] S. Gladchenko, D. Olaya, E. Dupont-Ferrier, B. Douot, L. B. Ioffe, and M. E. Gershenson, *Superconducting Nanocircuits for Topologically Protected Qubits*, arxiv:0802.2295 (2008).

Chapter 11

The One-Way Quantum Computer

Gregor Seiler
supervisor: Evgeny Kozik

In this report the cluster state is introduced as a universal substrate for one-way quantum computation by single-qubit measurements only. The one-way quantum computer QC_C is described as a simulator of quantum logic networks. Later it is shown that the QC_C is not actually described best by this approach and that astonishing features of the QC_C are only getting visible by looking beyond the network description.

11.1 Introduction

The one-way quantum computer which is also called cluster state quantum computer or QC_C in short is a model of a universal quantum computer. Therefore this report can be seen as a follow-up to the report written by Pascal Steger, who has introduced the two more commonly known models "quantum turing machine" and "quantum network model". The QC_C is quite a new development since the paper first describing the model was published by Robert Raussendorf and Hans J. Briegel only in 2001 [1].
A specific highly entangled quantum state called *cluster state* lies at the heart of the QC_C. The cluster state is described in every detail in [2]. This quantum state allows for quantum computation by *single-qubit* measurements only. The central role of measurement is why the QC_C is said to be one-way. This is because the randomness of the measurement outcomes makes the QC_C inherently

irreversible, whereas the quantum network model for instance is reversible. Here reversible means that after a computation is finished on the quantum network model, which means that a known quantum input state was transformed to a quantum output state, this computation can always be reversed such that the input state is reproduced. This is not possible with the QC_C, hence it is one-way.

11.2 The Cluster State

On one hand the cluster state is a specific highly entangled pure multi-qubit quantum state going to be defined shortly. On the other hand in addition to the qubits being in the quantum state referred to by the cluster state, if one says that a set of qubits are in the cluster state, one also implies that those qubits are located on a connected cluster $C \subset \mathbb{Z}^2$. Nevertheless throughout this report it is clear from the context what aspect of the cluster state is meant. The creation of the cluster state is efficient, that is the time needed to bring a cluster of qubits into the cluster state is independent of the number of qubits. In 2005 a group around P. Walther has managed to experimentally realize such a state by using the polarization states of four photons as qubits [3].
The cluster state $|\Phi\rangle_C$ of a cluster of qubits C can be defined as follows:

$$|\Phi\rangle_C := \bigotimes_{a\in C} \Big(|0\rangle_a \bigotimes_{\gamma\in\Gamma} \sigma_z^{(a+\gamma)} + |1\rangle_a \Big),$$

where $\{|0\rangle, |1\rangle\}$ is a computational basis with $|0\rangle$ and $|1\rangle$ being the eigenvalues of the phase flip operator σ_z. $\sigma_z^{(a)}$ is a Pauli operator acting on the qubit a with $\sigma_z^{(a+\gamma)} \equiv 1$ if $a+\gamma \notin C$, and $\Gamma = \left\{ \binom{1}{0}, \binom{0}{1} \right\}$. This formula can be evaluated for every cluster of qubits.

Example: Five qubits in a row, i.e. $C_5 = \{1,2,3,4,5\}\binom{1}{0}$.

$$|\Phi\rangle_{C_5} = |+,0,-,0,-\rangle - |+,0,+,1,+\rangle - |-,1,+,0,-\rangle + |-,1,-,1,+\rangle,$$

with $|\pm\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle \pm |1\rangle \big)$.
The cluster state can be realized by first preparing the state $\bigotimes_{a\in C} |+\rangle_a$ and then applying the controlled phase gate

$$S^{ab} = |0\rangle_a\langle 0| \otimes \mathrm{id}^{(b)} + |1\rangle_a\langle 1| \otimes \sigma_z^{(b)}$$

between neighboring qubits a, b. So, we have the alternative definition of the cluster state

$$|\Phi\rangle_C = S\Big(\bigotimes_{a\in C} |+\rangle_a \Big),$$

where $S = \prod_{a\in C,\gamma\in\Gamma} S^{a,a+\gamma}$. This is illustrated in figure 11.1. The unitary transformation S is generated by the Ising-type Hamiltonian

$$\mathcal{H} = \hbar g(t) \sum_{a\in C,\gamma\in\Gamma} \frac{1+\sigma_z^{(a)}}{2}\frac{1-\sigma_z^{(a+\gamma)}}{2},$$

where, again, $\Gamma = \left\{\binom{1}{0}, \binom{0}{1}\right\}$. The strength $g(t)$ of the next-neighbor interaction described by this Hamiltonian must be switchable between zero and a non-zero value, so that the interaction can be switched on for an appropriately chosen amount of time T such that $\int_0^T g(t)\mathrm{d}t = \pi$ and S is generated.
For theoretical considerations a third definition of the cluster state is best-suited.

$$\underbrace{\left(\sigma_x^{(a)} \bigotimes_{\gamma\in\Gamma\cup-\Gamma} \sigma_z^{(a+\gamma)}\right)}_{=:K^{(a)},\text{ stabilizers}} |\Phi\rangle_C = \pm\,|\Phi\rangle_C \quad \forall a \in C,$$

The eigenvalues depend on the occupation pattern of the cluster, but they are not important for the description of the QC_C. Here the state is completely defined by the set of eigenvalue equations, because the operators $K^{(a)}, a \in C$, called *stabilizers*, form a complete set of $|C|$ independent and commuting observables. All proofs of the claims made in this report use this definition.

11.3 Big Picture of the QCC

A computation on the QC_C works as follows. First, the qubits of the QC_C have to be brought into the cluster state. This process can be seen as an *initialization* of the QC_C, which is completely independent of the algorithm and the input to be processed with the QC_C, i.e. independent of the computational problem. Then, information is put in, processed and read out by single-qubit projective measurements only. By this a state $U\left(\bigotimes_{i=1}^n |+\rangle_i\right)$ with U an arbitrary unitary is generated on a particular set of n *output qubits* C_O. Those qubits are read out by measuring them in the σ_z-eigenbasis. An analogous set of *input qubits* is not needed, because the input state of the computational problem can be accounted for in the unitary transformation U by prepending the actual computation with a transformation that transforms the state $\bigotimes_{i=1}^n |+\rangle_i$ into the input state. One can say that the cluster state serves as a universal substrate for quantum computation by single-qubit measurements only. The directions of the measurements of every single qubit are adjusted according to the computational problem and according to results of measurements of other qubits. So, the computational problem is somewhat encoded into the directions of the single-qubit measurements. The

dependence of measurement directions on other measurement results makes it impossible to measure the qubits simultaneously or in any order, although the single-qubit measurements commute pairwise. It is very surprising that deterministic computation is possible by only using measurements whose outcomes are totally random. The dependency of measurement directions on other results is what makes this possible. This is going to be explained in more detail later.
An *implementation* of an algorithm on the QC_C consists of a temporal order in which the qubits are to be measured and a *measurement pattern*. A measurement pattern is an efficient classical algorithm which computes the directions in which the qubits have to be measured from previous measurement results such that the computational problem is solved on the QC_C if the qubits are measured in the way given by the measurement pattern.
In figure 11.2 a cluster of qubits is shown with which a computation by single-qubit measurements is performed. The circles and arrows denote measurements in the σ_z-basis and in the x-y-plane respectively.

11.4 Simulation of Quantum Logic Networks

Now the central question is, how can one find an implementation capable of solving a particular computational problem on the QC_C, that is, for instance, how can one implement Shor's algorithm on the QC_C.
For most quantum algorithms, and in particular for Shor's and Grover's algorithms, quantum logic network formulations exist. Therefore, we can reduce the problem of directly finding implementations of algorithms to the problem of finding implementations of a universal set of gates and to the problem of being able to compose those gates in arbitrary ways. This is called *simulation of quantum logic networks* on the QC_C for obvious reasons. Nevertheless the implementations of algorithms gained by this method are not the most efficient, but in fact there is no known other systematic way to find implementations. With being able to deduce measurement orders and patterns from arbitrary quantum logic networks, one not only solved the problem of implementing algorithms on the QC_C, but one also proved the QC_C to be universal. We will first look at implementing the cNot gate and general rotation gates. Afterwards we will see how these gate implementations can be composed. The forthcoming gate-implementations are analytically proven for arbitrary inputs in [4].
By simulation of quantum logic networks a close connection between the QC_C and the network model is established. Thus it is easy to confuse the two models. The following table summarizes the two models and clarifies the differences between

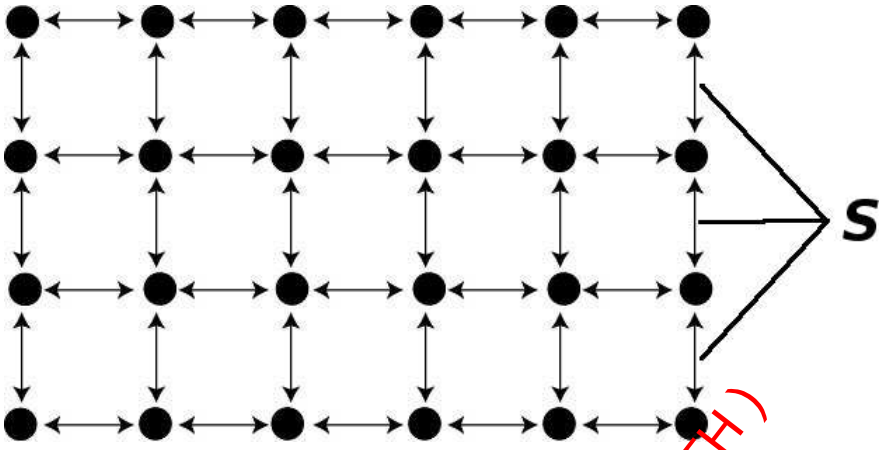


Figure 11.1: Phasegate S generated by Ising-type Hamiltonian

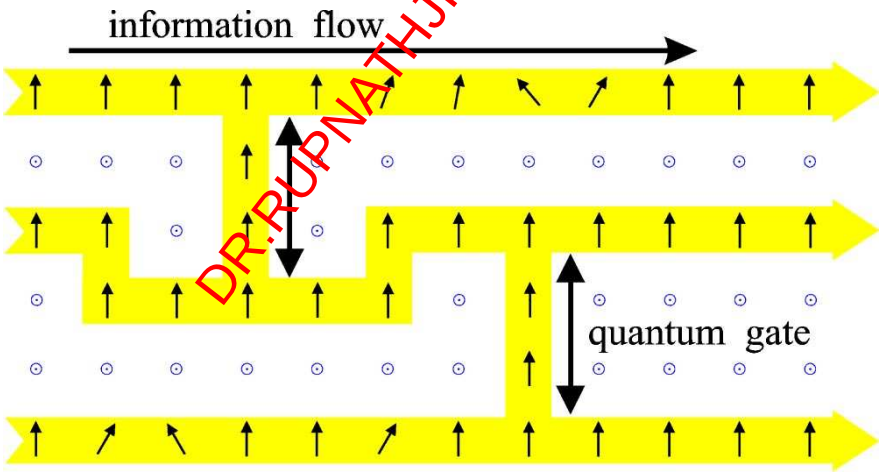


Figure 11.2: The Big Picture of the QC_C. Circles and arrows denote measurements in the σ_z-basis and in the x-y-plane respectively.

them.

Network Model	**QC_C**
Register of qubits in **undefined state**	**Cluster State**
Input of Information by preparation of particular register-state	
Unitary evolution of register-state decomposed into gates realized by **controlled interactions** between qubits	Computation realized by a sequence of **single-qubit measurements**
Output by read out of register, i.e. **measurement** of qubits	

The main difference between the QC_C and the network model is that in the network model one tries to directly implement gates by controlled interactions between qubits, whereas in the QC_C there are only single-qubit measurements. The simulation of quantum logic networks is only a method to find the directions in which those measurements have to be performed. If one is able to find measurement directions capable of implementing an algorithm directly, then the hole concept of gates is not needed in the QC_C. Gates are therefore not constituent elements of the QC_C. It is interesting to know that both models are equally efficient since they can simulate each other efficiently.
Although it was claimed that input qubits are not needed, they are introduced in the explanations of the gate implementations for didactic reasons. This makes it easier to understand gate composition, and we will later get rid of them again.

11.4.1 The cNot Gate

The cNot gate can be implemented on a cluster consisting of 15 qubits as is shown in figure 11.3. Qubits $1, 8$ and $7, 15$ are the input and output qubits respectively. First the input state $|c, t\rangle$ has to be prepared on the qubits $1, 8$ and all other qubits have to be brought into the state $|+\rangle$. Then the cluster has to be entangled by applying the unitary S. Note that by this procedure no cluster state in produced, but of course a related state is. After this the input can be

processed by measuring all qubits in bases as depicted in the figure. The following gate is thus realized, i.e. the following state is realized on the output qubits:

$$|\text{out}\rangle_{7,15} = U_\Sigma CNOT(c,t)\,|c,t\rangle\,,$$

with U_Σ a random *byproduct operator* in the Pauli group. The exact form of this operator depends on the measurement results, i.e. this operator is known if one keeps track of the measurement results. The byproduct operator reduces the problem of coping with measurement randomness to the problem of coping with this operator.

11.4.2 The General Rotation Gate

General rotation gates can be implemented on a cluster consisting of 5 qubits in a row, where qubit 1 is the input and qubit 5 is the output qubit (see figure 11.4). After input state preparation and entanglement the following measurements have to be performed.

1. Measurement of qubit 1 in $\mathcal{B}_1(0)$,
2. Measurement of qubit 2 in $\mathcal{B}_2(-\xi(-1)^{s_1})$,
3. Measurement of qubit 3 in $\mathcal{B}_3(-\eta(-1)^{s_2})$,
4. Measurement of qubit 4 in $\mathcal{B}_4(-\zeta(-1)^{s_1+s_2})$,

where

$$\mathcal{B}_i(\varphi_i) = \left\{\frac{|0\rangle_i + e^{i\varphi_i}\,|1\rangle_i}{\sqrt{2}}, \frac{|0\rangle_i - e^{i\varphi_i}\,|1\rangle_i}{\sqrt{2}}\right\}.$$

$s_i \in \{0,1\}$ denotes the result of the measurement of qubit i with $s_i = 0$ meaning that the qubit was projected into the first state of the basis. ξ, η, ζ are Euler-angles. Dependent on the measurement results, the following gate is thus realized:

$$|\text{out}\rangle_5 = U_\Sigma U_{rot}(\xi,\eta,\zeta)\,|in\rangle\,.$$

Example: The identity gate.
The identity gate is a simple special case of the rotation gates with $\xi, \eta, \zeta = 0$. So all four qubits have to be measured in the basis $\mathcal{B}(0) = \{|+\rangle\,, |-\rangle\}$, which is the eigenbasis of the σ_x-operator. We consider the case where the input state $|\text{in}\rangle = |+\rangle$ is prepared on qubit 1. So after the entanglement the state realized is the cluster state as was shown in the previous example

$$|\Phi\rangle_{C_5} = |+,0,-,0,-\rangle - |+,0,+,1,+\rangle - |-,1,+,0,-\rangle + |-,1,-,1,+\rangle\,.$$

We assume that the four measurements have outcomes $0, 1, 0, 1$, so the effect of those measurements is the projection, modulo norm factor, $|+\rangle_1\langle+| \otimes |-\rangle_2\langle-| \otimes |+\rangle_3\langle+| \otimes |-\rangle_4\langle-|$. Of course every other combination would be possible too, but with these measurement outcomes the random byproduct operator becomes the identity operator. Therefor we expect the fifth qubit to be in the state $|+\rangle$ after the measurements. The calculation goes as follows

$$\begin{aligned}
|+\rangle_1\langle+|\,|\Phi\rangle_{C_5} &= |+,0,-,0,-\rangle - |+,0,+,1,+\rangle \\
|-\rangle_2\langle-|\,|+\rangle_1\langle+|\,|\Phi\rangle_{C_5} &= |+,-,-,0,-\rangle - |+,-,+,1,+\rangle \\
|+\rangle_3\langle+|\,|-\rangle_2\langle-|\,|+\rangle_1\langle+|\,|\Phi\rangle_{C_5} &= -|+,-,+,1,+\rangle \\
|-\rangle_4\langle-|\,|+\rangle_3\langle+|\,|-\rangle_2\langle-|\,|+\rangle_1\langle+|\,|\Phi\rangle_{C_5} &= |+,-,+,-,+\rangle
\end{aligned}$$

Indeed, the fifth qubit is in the state $|+\rangle$ as expected. The identity gate can be seen as quantum teleportation and one can say that quantum information flows through the cluster during a computation (see figure 11.2).

11.4.3 Composition of Gates

In order to do something useful with the gate implementations described before, it is absolutely necessary to be able to compose those gate implementations. The 2-dimensional geometry of the cluster state and of the individual gate simulations allows for composing gates by overlapping input- and output-qubits. What would obviously work is to simulate a quantum logic network gate by gate. One can start on one side of the cluster by preparing a set of qubits in the input state of the computation. By entangling the qubits needed for the first gates and then measuring those qubits, those gates can be implemented. Afterwards one can proceed from the output qubits of the first gates by using those output qubits as input qubits for the next gates, entangling qubits needed for these gates, measuring them and so on. This is certainly not the best way to operate the QC_C and it would be very difficult to selectively entangle only some specific qubits in the lab. Fortunately there is a much better way. The entanglement of a gate in this gate-by-gate description commutes with all previous measurements, because those operations act on disjoint sets of qubits. Therefore the whole cluster can be entangled once and then all the measurements can be performed subsequently. This is graphically illustrated in figure 11.5. We got rid of the input qubits now. What remains is that a cluster normally has not the exact geometry needed for a particular quantum logic network. This is no problem because superfluous qubits can be removed from the cluster state by measuring them in the σ_z-eigenbasis. This means that if some qubits of a cluster state are measured in this basis then

they are of course not entangled with the other qubits anymore and the other qubits are still in a cluster state as if only those had been entangled.

11.4.4 Coping with Measurement Randomness

As explained so far, the simulation of a quantum logic network with gates $g_1, g_2, \cdots, g_n$ for the unitaries $U_1, U_2, \cdots, U_n$ realizes

$$(U_{\Sigma,n}U_n)(U_{\Sigma,n-1}U_{n-1})\cdots(U_{\Sigma,1}U_1)$$

instead of

$$(U_{\Sigma,n}U_{\Sigma,n-1}\cdots U_{\Sigma,1})(U_nU_{n-1}\cdots U_1) = U_\Sigma U,$$

which is not equivalent, since the byproduct operators in general do not commute with the gates. In the second equation byproduct operators can easily be coped with, because one can invert them by interpreting the read-outs adequately as long as their exact form is known, which is the case if one kept track of the measurement results.
The byproduct operators can be propagated through the network in a controlled way, but this modifies the gates:

$$\begin{aligned}&(V_{\Sigma,n}V_n)(V_{\Sigma,n-1}V_{n-1})\cdots(V_{\Sigma,1}V_1)\\ &\quad = (V'_{\Sigma,n}V'_{\Sigma,n-1}\cdots V'_{\Sigma,1})(V'_nV'_{n-1}\cdots V'_1),\end{aligned}$$

that is $V'_i \neq V_i$. For deterministic computation, V_i must be deduced from V'_i, such that $V'_i = U_i\ \forall i$. In fact, this is the only reason for measurement result dependency in the QC_C. One needs the following commutation relations:

- For the cNot gate:

$$\begin{aligned}CNOT(c,t)\sigma_x^{(t)} &= \sigma_x^{(t)}CNOT(c,t),\\ CNOT(c,t)\sigma_x^{(c)} &= \sigma_x^{(c)}\sigma_x^{(t)}CNOT(c,t),\\ CNOT(c,t)\sigma_z^{(t)} &= \sigma_z^{(c)}\sigma_z^{(t)}CNOT(c,t),\\ CNOT(c,t)\sigma_z^{(c)} &= \sigma_z^{(c)}CNOT(c,t).\end{aligned}$$

- For general rotations:

$$\begin{aligned}U_{rot}(\xi,\eta,\zeta)\sigma_x &= \sigma_xU_{rot}(\xi,-\eta,\zeta),\\ U_{rot}(\xi,\eta,\zeta)\sigma_z &= \sigma_zU_{rot}(-\xi,\eta,-\zeta).\end{aligned}$$

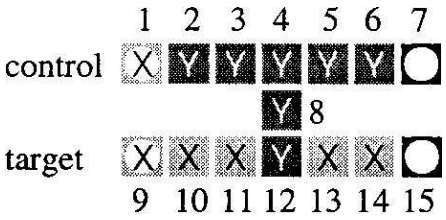


Figure 11.3: The cNot gate

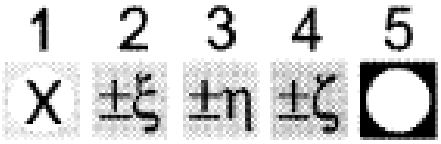


Figure 11.4: General rotations

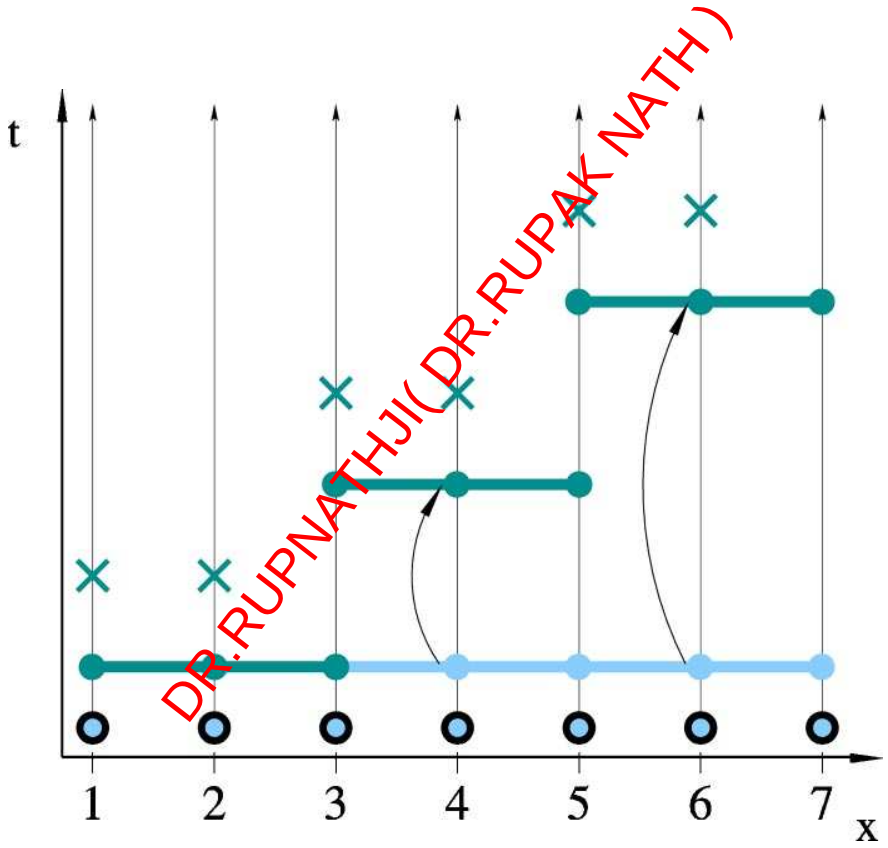


Figure 11.5: It suffices to entangle all qubits once, because entanglement commutes with the measurement of qubits belonging to earlier gates. The crosses denote one-qubit measurements and the lines between adjacent qubits denote the entanglement operations.

It can be seen that measurement directions of qubits belonging to cNot-gates do never depend on other measurement results, because a cNot-gate does not change when it is interchanged with a Pauli operator. The explanation of the QC_C as a simulator of quantum logic networks is complete now. In figure 11.3 an implementation for the quantum adder is shown to give an impression how a useful implementation looks like.

11.5 A New Computational Model

Till now, the QC_C was described in terms of the computational model of the network model. It was already stated that the QC_C is not described best by the network model and in particular that gates are not constituent elements of the QC_C. But there is a lot more to it which gets clear if one asks the question of what can be changed of an implementation deduced from a quantum logic network without changing the computational problem implemented. Firstly it is easy to realize that all those qubits whose measurement directions do not depend on other measurement results can be measured simultaneously at the very beginning of the computation and independently of the positions of the corresponding gates in the quantum logic network. In particular, all qubits belonging to cNot gates can be measured among the first. This shows that the temporal order of measurements induced by the logic network is not the most efficient. The so-called output qubits of the computation can be measured among the first because those qubits are always measured in the σ_z-eigenbasis. We got rid of input qubits already, so the distinction between input- measurement- and output-qubits is actually not appropriate. But if one measures the output qubits among the first, then the whole concept of quantum input and output does not make sense anymore because there is never, at any time, a set of qubits which represents the input or the output state of the computation. Further, the QC_C does not process information at the quantum level because the quantum correlations of the cluster state which are measured at run-time are not specific to the computational problem. The QC_C has changed what is believed to be required for quantum computation, because of those astonishing features.

So a good plan to find implementations which naturally emerges now is to keep the measurement pattern induced by a logic network, but to optimize the temporal order, so that every qubit is measured as early as possible. The computation model that thereby evolves goes beyond the scope of this report. The interested reader is refered to [5] and to [6] where this model is discussed.

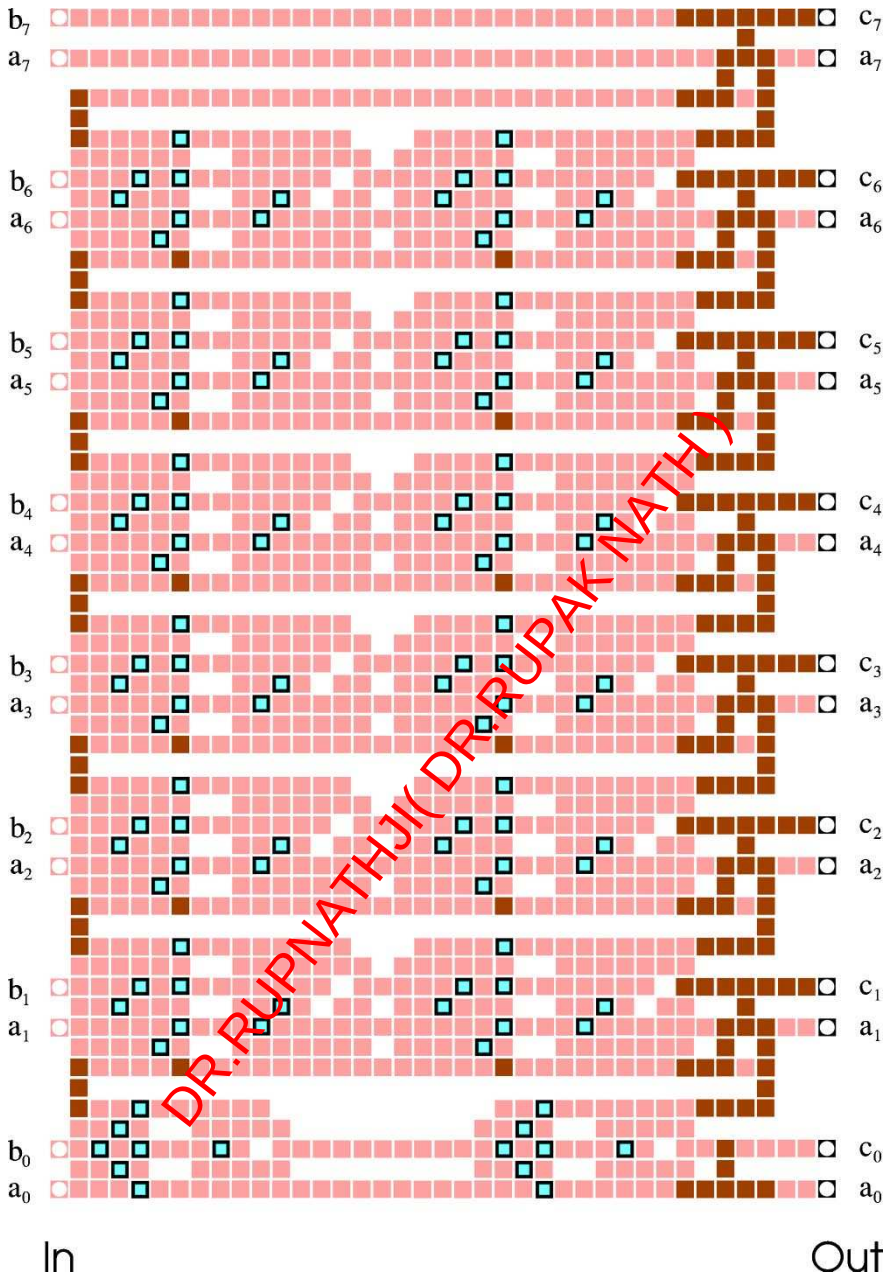


Figure 11.6: Example Implementation: Quantum Adder. White, light gray and dark gray squares denote measurements in the σ_z-, σ_x- and σ_y-basis respectively. Framed squares denote measurements which depend on other results.

Bibliography

[1] R. Raussendorf and H. J. Briegel, *A one-way quantum computer*, Phys. Rev. Letters **86**, 5188 (2001).

[2] H. J. Briegel and R. Raussendorf, *Persistent entanglement in arrays of interacting particles*, Phys. Rev. Letters **86**, 910 (2001).

[3] P. Walther, K. J. Resch, T. Rudolph, E. Schenck, H. Weinfurter, V. Vedral, M. Aspelmeyer, and A. Zeilinger, *Experimental one-way quantum computing*, Nature **434**, 169 (2005).

[4] R. Raussendorf and H. J. Briegel, *Measurement-based quantum computation on cluster states*, Phys. Rev. A **68**, 22312 (2003).

[5] R. Raussendorf and H. J. Briegel, *Computational model for the one-way quantum computer: Concepts and summary*, in *Quantum Information Processing*, edited by G. Leuchs and T. Beth (WILEY-VCH, Weinheim, 2003).

[6] R. Raussendorf and H. J. Briegel, *Computational model underlying the one-way quantum computer*, Quant. Inform. Comp. **2**, 344 (2002).

Chapter 12

Quantum Communication Complexity

Yves Delley
supervisor: Lode Pollet

In the first part communication complexity is introduced. Based on a survey done by Gilles Brassard in [1], different models for quantum communication complexity are presented and their application is shown with simple examples. In the second part a remarkable result achieved by Kavinsky et al. [2] is presented in more detail: They exhibit a problem which can be solved exponentially better using quantum communication than with classical communication. We further explore the connection of that result with quantum cryptography.

12.1 Motivation

In the field of quantum communication complexity one tries to quantify the power of quantum information for communication purposes. Naturally, we would like to know if quantum communication is better than classical communication. Holevo's theorem [4] might lead to the assumption that this is not the case. It states that by the transmission of n quantum bits no more than n bits of expected classical information can be communicated between unentangled parties. If the parties additionally share an entangled quantum state this number can only be doubled [4].

However there are cases where quantum communication can achieve up to an exponential advantage against classical communication. In *computation* there is

the famous result by Shor [5], which is believed to provide an exponential speed-up over any classical algorithm. However, it still remains to be proved if really no better classical algorithm exists. Contrarily, the exponential separation between classical and quantum communication has been proved!

12.2 Basics

12.2.1 Classical communication complexity

The first communication complexity model was introduced by Andrew Yao in 1979 in [6]. In his setting there are two distant parties, *Alice* and *Bob*. They want to compute the value of a function $f : A \times B \longrightarrow Z$ on a pair of inputs $a \in A$ and $b \in B$, where A,B and Z are arbitrary finite sets. However a is only known to Alice and b only to Bob. The question is how much information they have to transmit to each other, until one of them can compute the output value. This model of communication is thus similar to one of computation, but we are only interested in the amount of communication needed.
The notion of *algorithm* is replaced by the notion of *protocol*. Alice and Bob take turns to send a message (bits, or strings of bits) between themselves according to some protocol T. The protocol consists of a list of functions describing what message to send in which turn:

Turn	Alice's messages	Bob's messages
1	$M_1 = f_1(a)$	-
2	-	$M_2 = f_2(b, M_1)$
3	$M_3 = f_3(a, M_1, M_2)$	-
$\vdots$	$\vdots$	$\vdots$

The protocol is finished when the output can be determined by one party using his own input and all the previous messages, and both parties know about this fact. [1]
The *cost* of a protocol T on a given input (a, b) is defined as the number of bits sent by Alice and Bob together until the output is determined. Let us denote this cost by $\mathcal{C}_T(a, b)$. The total cost of a protocol T is the worst case cost of T

[1] We usually do not care whether Alice or Bob computes the output. For a problem giving interesting results the output has to be much smaller than the input, so it's transmission can usually be neglected. If the output were of the size of the input, then there cannot be any protocol much more efficient than just sending the input from one party to the other, which is the trivial upper bound.

over all inputs. The (deterministic) *communication complexity* of a function f is then defined as the minimum cost over all protocols that compute f:

$$D(f) = \min_{T \in \mathcal{T}_f} \left(\max_{(a,b) \in A \times B} \mathcal{C}_T(a,b) \right) \tag{12.1}$$

$\mathcal{T}_f$ denotes the set of protocols that compute f.
If we have a family of functions or problems with varying input sizes $n := \frac{1}{2}\log(|A|\,|B|)$, we can regard the communication complexity as a function of n and investigate it's asymptotic behaviour.

Example 1. *A well studied communication problem is the* equality function*: Let A,B be the set of n-bit strings.*

$$\begin{array}{rccl} EQ: & \{0,1\}^n \times \{0,1\}^n & \longrightarrow & \{0,1\} \\ & (a,b) & \longmapsto & EQ(a,b) := \begin{cases} 1 & a = b \\ 0 & else \end{cases} \end{array}$$

One can show that the deterministic communication complexity of this problem is linear in n:

$$D(EQ) = n$$

Probabilistic classical communication complexity

Similar as in the field of computation, interesting results can be achieved if we add randomness. In communication, we achieve this by allowing the parties to flip coins to decide what message to send. This is represented in the protocol by the random variables r_A and r_B:

Turn	Alice's messages	Bob's messages
1	$M_1 = f_1(a, r_A)$	-
2	-	$M_2 = f_2(b, r_B, M_1)$
3	$M_3 = f_3(a, r_A, M_1, M_2)$	-
$\vdots$	$\vdots$	$\vdots$

In this case the output and the number of bits transmitted become random variables too. So the question is how to define the cost of such a probabilistic protocol.

Bounded-error communication complexity

One usually tolerates a probabilistic protocol to *fail* computing f correctly in a small ε-fraction of all coin flips, $\varepsilon > 0$. Such a protocol is then said to compute f to ε precision, and we denote the set of all such protocols by $\mathcal{T}_f^\varepsilon$. The communication complexity of a problem, if a small error probability is tolerated, is then called the *bounded-error* communication complexity. We can further distinguish two different variants of this definition, either taking the full worst case cost, or taking the *expected* cost over all coin tosses, denoted by $\mathbb{E}_{r_A,r_B}[\mathcal{C}_T(a,b,r_A,r_B)]$. [2]

$$C_\varepsilon(f) = \min_{T\in\mathcal{T}_f^\varepsilon}\left(\max_{(a,b),r_A,r_B}\mathcal{C}_T(a,b,r_A,r_B)\right) \tag{12.2}$$

$$C_\varepsilon^E(f) = \min_{T\in\mathcal{T}_f^\varepsilon}\left(\max_{(a,b)}\mathbb{E}_{r_A,r_B}\left[\mathcal{C}_T(a,b,r_A,r_B)\right]\right) \tag{12.3}$$

The asymptotic bounded-error communication complexity is independent of ε as by repeating such a protocol k times, the error probability can be lowered to any value, without changing the asymptotic complexity since constant factors are irrelevant.

Zero-error communication complexity

If measuring the *expected* cost in the above definition, we could even demand the correct answer all the time. In this case it is called the *zero-error* communication complexity:

$$C_0^E(f) = \min_{T\in\mathcal{T}_f^0}\left(\max_{(a,b)}\mathbb{E}_{r_A,r_B}\left[\mathcal{C}_T(a,b,r_A,r_B)\right]\right) \tag{12.4}$$

Public vs. private coins

In both models above, the random variables r_A and r_B are only known to a single party. They are *private* random variables. Another possibility would be to allow for one random variable r, which is *publicly* available and set before the inputs are given to the parties. For complexity purposes however, it has been shown by Ilan Newman [7] that they are equivalent. The main idea is that there is a small set of bit strings which exhibit enough randomness to run a random protocol with only a small increase in error probability. This set can be shared beforehand. Alice and Bob only have to agree on which element of the set to use. This set is of the order of n elements, so that the choice can

[2] In the literature it is often not stated which definition is used.

be communicated efficiently using only $\log(n)$ bits. This way n public random bits can be simulated by $\log(n)$ private random bits. So for any communication problem with asymptotic bounded-error communication complexity of at least $\log n$ when using public coins, the communication complexity is the same when using private coins.

Example 2.

Theorem 1. *$C_\varepsilon(EQ) = O(1)$ in the presence of public coins.*
Let the public coin r reside in $\{0,1\}^n$. The protocol achieving this upper bound consists of comparing the parity of the dot-product of the inputs with the random variable r.

1. *Alice computes $z = \bigoplus_i (r_i \cdot a_i) \in \{0,1\}$*
2. *Alice transmits z to Bob.*
3. *Bob compares z to $\bigoplus_i (r_i \cdot b_i)$*
4. *If they are not equal, he concludes $a \neq b$, otherwise he answers $a = b$*

If indeed $a = b$, Bob will always answer correctly. On the other hand if $a \neq b$ his chance of answering correctly is exactly $1/2$. For the parities to disagree, the inputs have to disagree on an odd number of sites i where r_i is one. Mathematically:

$$|\{i|a_i \neq b_i\} \cap \{i|r_i = 1\}| \quad \text{is odd}$$

Since for each r_i the probability of being one is exactly $1/2$, with each site considered additionally, the switch between agreement and disagreement of the parities is exactly $1/2$. This shows that this protocol achieves a bounded error probability using a constant amount of transmitted bits (just one), independent of the input size. Using the result of the preceding section we can also conclude:

Theorem 2. *$C_\varepsilon(EQ) = \Omega(\log(n))$ in absence of public coins.*

12.2.2 Functions, promises and relations

Computing a *total* function on distributed inputs is not the only possible task requiring communication. Often a *promise* is added to the function. A promise P is a subset of all inputs in $A \times B$. Alice and Bob are given inputs only from P. Sometimes this is also called a *partial* function evaluation.
An even more general description of a distributed problem can be given by defining a relation $R \in A \times B \times X$, requiring that the output of the protocol has to be chosen such that (a, b, x) resides in R. This allows to specify multiple valid answers for some inputs.

12.2.3 Quantum communication complexity

Yao also introduced the first *quantum* communication complexity model [8], where Alice and Bob are allowed to exchange qubits rather than bits. Analogous to the classical case, we can define different quantum communication complexities. $Q_\varepsilon(P)$,$Q_\varepsilon^E(P)$ for the bounded-error complexity, and $Q_0^E(P)$ for the zero-error complexity.

12.3 Exponential separation for one-way quantum communication complexity

In this section we are going to present the very recent and interesting result by Gavinsky et al. [2]. They were the first to show an asymptotically exponential separation between the classical and the quantum communication complexity of a one-way communication problem.

12.3.1 Description of the setup

The problem that they investigated is a variant of the so called *boolean hidden matching problem*, which will be introduced in more detail in the next section. It is basically a binary function evaluation with a input promise. Communication is restricted to one-way in this problem, where Alice sends just one message to Bob who then has to produce the output. The separation is established between the bounded-error classical communication complexity in presence of public coins and the bounded-error quantum communication complexity:

Theorem 3.

$$C_\varepsilon(BHMP) \in \Theta(\sqrt{n/\alpha}) \qquad \text{(Classical tight bound)}$$
$$Q_\varepsilon(BHMP) \in O(\log n/\alpha) \qquad \text{(Quantum upper bound)}$$

12.3.2 The boolean hidden matching problem

Alice's input is a n-bit string $x \in \{0,1\}^n$. Bob's receives a sequence M of $k = \alpha n$ disjoint pairs $(i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k) \in \{1, \ldots, n\} \times \{1, \ldots, n\}$ of indices into Alice's string, and a k-bit string w. $\alpha \in (0, 1/2]$ is a fixed parameter. Bob's sequence of pairs is called an α-*matching* as it matches Alice's bits to pairs. For $\alpha < 1/2$ it is a *partial* matching, as not all bits end up in a pair, whereas for $\alpha = 1/2$ it is *total* matching. We can construct an αn-bit sequence z out of those

two inputs by XOR'ing together the two bits referenced in each pair of Bob's matching $z_l := x_{i_l} \oplus x_{j_l}$. Regard x and z as vectors in a vector space over $\mathbb{F}_2$, the binary algebra. Each bit in z is then just a linear superposition of two bits of x. M can thus be regarded as a matrix in ${\mathbb{F}_2}^{k \times n}$ and z can be expressed as $z = Mx$. For an illustration of this, see Figure 12.1.
The promise on the input is that there exists a bit b satisfying $w_l = b \oplus z_l$ for all $l \in \{1, \ldots, k\}$. In other words w is either equal to z or to it's binary complement. Bob's task is now to find this bit b. To achieve this it suffices for him to learn any one of the k bits in z. Then using $w_l \oplus z_l = (b \oplus z_l) \oplus z_l = b \oplus (z_l \oplus z_l) = b \oplus 0 = b$ he can discover the bit b. However, any bit x_i on it's own does not reveal any bit of z. Only if information on both bits of a pair is simultaneously available, Bob can induce a value on z.

12.4 Application in Cryptography: Key expansion in the bounded storage model

Key expansion is a technique where, if two parties *Alice* and *Amanda* share a secret key, they can expand this key to a bigger one. In the bounded storage model, the adversary, *Bob*, has only bounded storage. In that context, the exponential separation between the classical and the quantum communication model translates in a failure of a classically secure protocol against quantum attacks.
Assume Alice and Amanda share a secret key M, which can be represented as an α-matching. If they want to expand their shared key, they could try the following: Alice produces a large random bit-string x uniformly distributed over $\{0,1\}^n$. Alice sends then this bit-string over a *public* channel to Amanda. The idea is, that since Bob's storage is limited, he cannot store all of this x, but must discard some bits. Alice and Amanda then use this bit-stream to generate $z = Mx$ and expand their shared secret M to their new secret (z, M).
According to the results presented in this paper, if Bob cannot store more than $\sqrt{n/\alpha}$ classical bits, even if he discovers M afterward, he cannot determine z. Thus z is a real expansion of M, in that learning M is not enough to know the whole secret. However, if Bob has quantum storage, then already a $\log(n)/\alpha$ qubit register is enough to keep all the *essential* information on x. If he later discovers M, he can then discover a random bit of z.
The important point here is proving a key-expansion scheme is secure classically, is not sufficient. Even if the scheme works without quantum information, the adversary can profit from quantum memory.
The important point here is if you have a (classical) key-expansion scheme, it's

not sufficient to prove it's classical security, it still could be insecure against quantum adversaries!

12.5 Proof outlines of the bounds

12.5.1 Quantum upper bound

Using a simple protocol of only of the order of $\log_2 n$ sent qubits, Alice can guarantee Bob a constant probability to learn one bit of z.
Alice sends a uniform superposition of her bits in a $\log_2 n$ qubit register:

$$|\psi\rangle = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} (-1)^{x_i} |i\rangle \tag{12.5}$$

Bob applies subsequently for each of his pairs the projection operator onto the subspace of the two bits belonging to the pair l:

$$|i_l\rangle \langle i_l| + |j_l\rangle \langle j_l| \tag{12.6}$$

For either pair there is a chance that the measurement will project the initial state onto that two-dimensional subspace. He knows if this happened by the outcome of the measurement.

$$|\psi\rangle = \frac{1}{\sqrt{2}} \left((-1)^{x_{i_k}} |i_k\rangle + (-1)^{x_{j_k}} |j_k\rangle \right) \tag{12.7}$$

Bob measures this state in the $|\pm\rangle$ base to find z_k:

$$\begin{aligned} \langle\psi| \frac{1}{\sqrt{2}} (|i_k\rangle - |j_k\rangle) \frac{1}{\sqrt{2}} (\langle i_k| - \langle j_k|) |\psi\rangle &= \frac{1}{4} \left((-1)^{x_{i_k}} - (-1)^{x_{j_k}} \right)^2 \\ &= x_{i_k} \oplus x_{j_k} = z_k \end{aligned} \tag{12.8}$$

The only way for this protocol to fail is if each time applying the projection operator 12.6 doesn't measure the state in this space. Then the state $|\psi\rangle$ collapsed to the subspace corresponding to the bits which are not part of any pair of Bob's matching. Since this subspace has dimension $n - 2k = (1 - 2\alpha)n$ and the initial state $|\psi\rangle$ is uniformly distributed over the entire space of dimension n, the probability of finding the state in any of his pairs is exactly 2α which is non-vanishing and independent of n. By repeated application of this procedure a constant number of times, Bob can achieve any required error bound. This concludes the prove of the upper bound of the asymptotic quantum communication complexity of this problem: $Q_\epsilon(HBMP) \in O(\log(n)/\alpha)$.

12.5.2 The classical upper bound

Using the birthday paradox we can show that if Alice chooses of the order of $\sqrt{n/\alpha}$ bits of her input, then with constant probability Bob will end up with both bits of at least one of his pairs. Contrary to the normal birthday paradox, we have here two different types of "birthdays", those which belong to a pair and those which don't. This renders the derivation of an upper bound a bit more difficult. The proof of the quoted upper bound is given in the Appendix 12.6.

12.5.3 Classical lower bound

The proof of the classical lower bound is much more complicated than the bounds done before. The complication comes from the fact that we have to prove that there is no protocol at all which achieves a better asymptotic complexity than the one we are proposing. However by Yao's principle [9], we can constrain ourselves to *deterministic* protocols applied onto some difficult input distribution.

Yao's principle

Recall the definition of the bounded error communication complexity given in section 12.2:

$$C_\varepsilon(f) = \min_{T \in \mathcal{T}_f^\varepsilon} \left(\max_{(a,b),r_A,r_B} \mathcal{C}_T(a,b,r_A,r_B) \right)$$

To provide a lower bound on this communication complexity, we need to establish that the worst case cost of any protocol is at least as high as our proposition. Yao's principle states that the worst case performance of any randomized procedure is always worse than the best deterministic algorithm for a given input *distribution*. Mathematically:

Theorem 4. *Let T be an arbitrary* probabilistic *bounded-error protocol that solves the problem P, and let I be an arbitrary probability distribution of input variables (a,b). Denote by $\mathcal{T}_P^0$ the set of* deterministic *protocols solving the problem P. Then the following inequality holds:*

$$\max_{(a,b)\in A\times B} \left(\mathcal{C}_T(a,b)\right) \geq \min_{S\in\mathcal{T}_P^0} \mathbb{E}_{(a,b)\ \textit{distributed according to } I} \left(\mathcal{C}_S(a,b)\right)$$

Thus if we find a hard enough input distribution I for which we can prove a lower bound for the expected cost for *all deterministic* protocols S, then this translates into a lower bound for the worst case cost of all *probabilistic* protocols, and then in turn into a lower bound for the probabilistic communication complexity.
Yao's principle is based on Von Neumanns *minimax principle*.

The main theorem

Suppose Alice follows a deterministic protocol which transmits c bits to Bob. The message then induces a set $A \subset \{0,1\}^n$, which contains all the input values that yield the same message. The cardinality of this set is on average 2^{n-c}. Assuming uniform distribution of the random variable x over this set A, Bob's knowledge of M leaves him with a probability distribution p_M:

$$p_M(z) = \frac{|\{x \in A | Mx = z\}|}{|A|} \tag{12.9}$$

The main result presented by Gavinsky et Al. in [2] is the following theorem, which basically states that if much less than $\sqrt{n/\alpha}$ bits are transmitted, this probability distribution is essentially uniform.

Theorem 5. *Let x be uniformly distributed over a set $A \subset \{0,1\}^n$ of size $|A| \geq 2^{n-c}$ for some $c \geq 1$, and let M be uniformly distributed over the set $\mathcal{M}_{\alpha n}$ of all α-matchings, for some $\alpha \in (0, 1/4]$. Denote with U the uniform probability distribution over $\{0,1\}^k$. There exists a universal constant $\gamma \geq 0$ (independent of n, c, and α), such that for all $\varepsilon > 0$: if $c \leq \gamma\varepsilon\sqrt{n/\alpha}$ then*

$$\mathbb{E}_M \left[\|p_M - U\|_{tvd} \right] \leq \varepsilon$$

The total variational distance $\|\cdot\|_{tvd}$ appearing in this theorem is defined by

$$\|p - q\|_{tvd} := \sum_{x \in X} \frac{|p(x) - q(x)|}{|X|} \tag{12.10}$$

where p and q are probability distributions over X. It is useful as it is an upper bound on the probability of success of determining if a value x originates from one or the other probability distribution. We won't proof this theorem, the concerning reader is requested to consult [2] for that.

Putting it together

Using Yao's principle we constrain us to *deterministic* protocols applied onto a hard input distribution. As input distribution we will choose the uniform distribution of x over $\{0,1\}^n$, of M over $\mathcal{M}_{\alpha n}$ and w equally distributed over $z = z(M, x)$ and $\overline{z}$.
A classical protocol using $C = c - \log_2(1/\varepsilon)$ divides the a priori distribution of x for Bob into a uniform distribution over one of 2^C distinct sets $A_1, \ldots, A_{2^C}$ of input values, depending on the message produced by the protocol. As they

together cover all $\{0,1\}^n$ the different sets have on average the size 2^{n-C}. Moreover for every l there cannot be more than a 2^{-l} fraction of all x in sets of size smaller than 2^{n-C-l}. Set $-l = -\log_2(1/\varepsilon) = \log_2(\varepsilon)$ and we easily see that with probability $1-\varepsilon$ the message sent by Alice corresponds to a set A of size at least $2^{n-C-log_2(1/\varepsilon)} = 2^{n-c}$ In that case we can apply the theorem 5. So the expected total variational distance of the induced distribution on z, $Z = MX$ is smaller than ε. Using Markov's inequality we can conclude that at least for $1-\sqrt{\varepsilon}$ of all M, $\|Z-U\|_{tvd} \leq \sqrt{\varepsilon}$ holds. So most of the time his induced distribution on Z will be essentially indistinguishable from the uniform distribution. But his task is to determine if the w he has been given was taken from Z or from $\overline{Z}$. The best probability he can do this is bounded by the total variational distance between the two distributions:

$$\left\|Z-\overline{Z}\right\|_{tvd} \leq \|Z-U\|_{tvd} + \left\|\overline{Z}-U\right\|_{tvd} = 2\|Z-U\|_{tvd} \leq 2\sqrt{\varepsilon} \tag{12.11}$$

Counting up we see that Bob's advantage over random guessing is less than $\varepsilon + \sqrt{\varepsilon} + \sqrt{\varepsilon}/2$. If $C/\sqrt{n/\alpha} < \delta$ and solving $\delta = \gamma\varepsilon - \log(1/\varepsilon)$ for ε we will fulfill the requirements of the derivation above. Since $\gamma\varepsilon - \log(1/\varepsilon)$ is monotonically increasing in ε, this means that if the transmitted number of bits C scales slower than $\sqrt{n/\alpha}$, we can put an upper bound on Bob's advantage which converges toward zero. Thus this proves the lower bound quoted in the beginning:

$$Q_\varepsilon(BHMP) \in \Omega(\sqrt(n/\alpha))$$

12.6 Appendix: Proof of the classical upper bound

12.6.1 The upper bound for the total matching

We start with a simplified problem: What is the chance of hitting a pair of a *total* matching of k pairs, when sending i bits? We calculate the complement of this probability. Then it is an easy distribution task. Let Alice choose her bits sequentially. After she has transmitted j bits without hitting a pair twice, the probability that the $(j+1)$th bit does hit one of the previous j pairs is given by $j/(2k-j)$. Alice has $2k-j$ bits to choose from, but only j of them belong to a pair already hit. So we arrive at

$$\overline{p}(i,k) = \prod_{j=0}^{i-1}\left(1-\frac{j}{2k-j}\right) = \prod_{j=0}^{i-1}\frac{k-j}{k-j/2} \tag{12.12}$$

Note that $\overline{p}(i,k)$ is positive and smaller than 1, as expected for a probability. Furthermore it is monotonically decreasing in i.
We intend to establish an asymptotic upper bound on the minimum number i of bits required to achieve a certain probability of getting both bits of one pair.

Theorem 6. *Let $\varepsilon > 0$ and $\beta \geq 0$. Then we can find a N so that for all $k > N$ the following inequality holds:*

$$\forall\, i \geq \beta\sqrt{k}: \qquad \overline{p}(i,k) < \mathrm{e}^{-\frac{\beta^2}{4}} + \varepsilon$$

Similar to the genuine birthday paradox, the minimum number of bits to send scales with the square root of the number of total bits.
Proof: Let ε and β be given. Because $\overline{p}$ is monotonous it is sufficient to show the inequality for $i = \beta\sqrt{k}$, for all large enough k.

$$\overline{p}(i,k) = \prod_{j=0}^{i-1} \frac{k-j}{k-j/2} \tag{12.13}$$

$$\leq \prod_{j=0}^{i-1} \frac{\mathrm{e}^{-j/k}}{\mathrm{e}^{-\frac{j}{2k}} - \frac{j^2}{8k^2}} \tag{12.14}$$

In the first inequality we used the convexity of the exponential and of it's derivative. Factoring the exponential out of the denominator and reducing leads to:

$$\overline{p}(i,k) \leq \prod_{j=0}^{i-1} \mathrm{e}^{-\frac{j}{2k}} \left(1 - \frac{j^2}{8k^2}\mathrm{e}^{\frac{j}{2k}}\right)^{-1} \tag{12.15}$$

Replacing j by i inside the bracket decreases the magnitude of the bracket itself and so increases the total term:

$$\overline{p}(i,k) \leq \mathrm{e}^{-\frac{i\cdot(i-1)}{4k}} \left(1 - \frac{i^2}{8k^2}\mathrm{e}^{i/(2k)}\right)^{-i} \tag{12.16}$$

Replace i by $\beta\sqrt{k}$, to show the dependency on k:

$$\overline{p}(i,k) \leq \mathrm{e}^{-\frac{\beta^2}{4}} \mathrm{e}^{\frac{\beta}{\sqrt{k}}\left(\frac{1}{4} - k\log\left(1 - \frac{\beta^2}{8k}\mathrm{e}^{\beta/(2\sqrt{k})}\right)\right)} \tag{12.17}$$

As the argument of the logarithm approaches 1 with increasing k we replace it by a linear function. $log(1-x) \leq -2x$ holds for small enough $x > 0$. We can ensure that the argument to the logarithm is small enough by choosing N appropriately. With this lower bound on the logarithm we properly establish an upper bound to the second exponential.

$$\overline{p}(i,k) \leq \mathrm{e}^{-\frac{\beta^2}{4}} \mathrm{e}^{\frac{\beta}{\sqrt{k}}\left(\frac{1}{4} + 2\beta^2\mathrm{e}^{\beta/(2\sqrt{k})}\right)} \tag{12.18}$$

With increasing k, the bracket converges to a finite value, but the whole exponent converges to zero. Therefore the second exponential converges to one. As the first exponential is constant, we can replace the second exponential by the addition of an arbitrarily small ε, if we choose N big enough.

$$\overline{p}(i,k) \leq \mathrm{e}^{-\frac{\beta^2}{4}} + \varepsilon \tag{12.19}$$

12.6.2 The upper bound for the partial matching

Assume Alice sends c arbitrarily chosen bits. The probability that exactly i of them are part of a pair of Bob's α-matching M is given by:

$$q(i,c,k,n) := \frac{\binom{2k}{i}\binom{n-2k}{c-i}}{\binom{n}{c}} = \frac{\binom{2\alpha n}{i}\binom{(1-2\alpha)n}{c-i}}{\binom{n}{c}}$$

For $c/n \ll 1$ the chance of any bit hitting a pair becomes independent of the distribution of the previous bits and then the formula should approach $\binom{c}{i}\alpha^i(1-\alpha)^{c-i}$, the binomial distribution. The chance for Bob of not receiving both bits of any of his pairs is given by averaging the corresponding probability for the total matching, weighted by the distribution above:

$$\overline{p}(c,\alpha,n) = \sum_{i=0}^{c} \overline{p}(i,\alpha n) q(i,c,\alpha n,n)$$

Again replace i by $\beta\sqrt{k}$:

$$\overline{p}(c,\alpha,n) = \sum_{\beta\in\{0,1/\sqrt{k},\ldots,c/\sqrt{k}\}} \overline{p}(\beta\sqrt{k},\alpha n) q(\beta\sqrt{k},c,\alpha n,n)$$

Now assuming $c = \sqrt{n/\alpha} = \sqrt{k}/\alpha$ we arrive at $i/c = \alpha\beta$. As shown in the previous section for big enough k $\overline{p}$ as a function of β and k is bounded from above independent of k. Furthermore q converges to the binomial-distribution, which in turn narrows with increasing $c \propto \sqrt{k}$ to a δ-distribution centered on $i/c = \alpha$.
Thus we have:

$$\lim_{n\to\infty} \overline{p}(\sqrt{n/\alpha},\alpha,n) = \lim_{n\to\infty} \overline{p}(\beta=1,k=\alpha n) \leq \mathrm{e}^{-1/4} \leq 1$$

This concludes the proof of the classical upper bound.

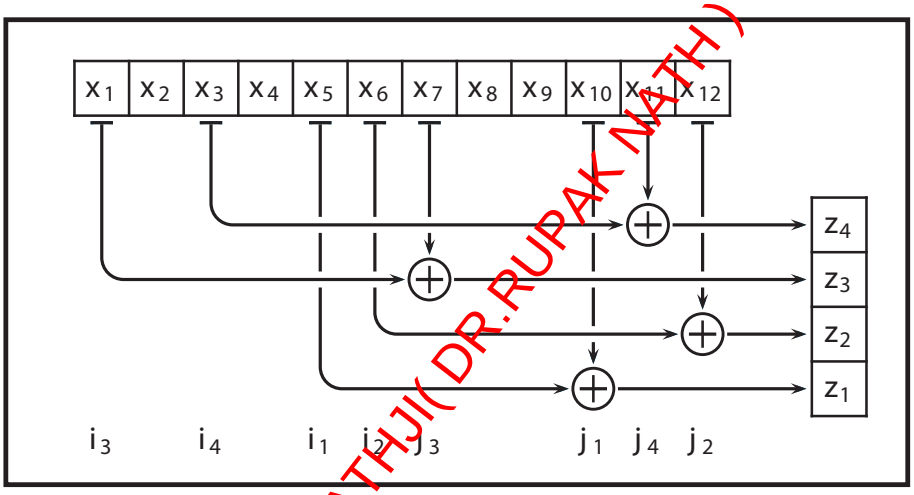


Figure 12.1: Illustration of the effect of an α-matching on x, with $n = 12$ and $\alpha = 1/3$

Bibliography

[1] G. Brassard, *Quantum Communication Complexity (A Survey)* (2001), URL `arXiv:quant-ph/0101005v1`.

[2] D. Gavinsky, J. Kempe, I. Kerenidis, R. Raz, and R. de Wolf, in *Proceedings of the 39th Annual ACM Symposium on Theory of Computing* (2008), vol. 39, pp. 516–525, URL `arXiv:quant-ph/0611209v3`.

[3] E. Kushilevitz and N. Nisan, *Communication Complexity* (Cambridge University Press, 1997).

[4] A. S. Holevo, in *Problemy Peredachi Informatsii* (1973), vol. 9(3), pp. 3–11, english translation in *Problems of Information Transmission* (1973), vol. 9, pp. 177-183.

[5] P. W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM J.SCI.STATIST.COMPUT. **26**, 1484 (1997), URL `arXiv.org:quant-ph/9508027`.

[6] A. C.-C. Yao, in *STOC '79: Proceedings of the eleventh annual ACM symposium on Theory of computing* (ACM, New York, NY, USA, 1979), pp. 209–213.

[7] I. Newman, *Private vs. common random bits in communication complexity*, Inf. Process. Lett. **39**, 67 (1991), ISSN 0020-0190.

[8] A. C.-C. Yao, in *Proceedings of the 34th annual IEEE symposium on Foundation of computer science* (1993), pp. 352–361.

[9] A. C.-C. Yao, in *Proc. of 18th IEEE FOCS* (1977), pp. 222–227.

Chapter 13

Coin Flipping

Philippe Labouchere
supervisor: Roger Colbeck

We review coin flipping starting from Blum's idea [1] in the classical case and focus on the quantum version of this two-party cryptographic task. Classically, coin tossing is impossible [2]. Quantum mechanically, we distinguish two kinds of coin flipping: weak coin tossing, which is asymptotically possible [3], meaning that the bias tends to zero as the number of rounds between Alice and Bob tends to infinity, and strong coing tossing, which is also impossible [2]. Protocols have been found [4, 5, 6, 7] which allow two players - Alice and Bob - to perform non-ideal strong and weak quantum coin tossing, i.e. the party's bias is non-zero.

13.1 Classical coin flipping

In 1981, Blum [1] introduced the following cryptographic problem: Alice and Bob, who have just divorced, are speaking over the phone to decide who will get the car; they both agree to find this out by tossing a coin. The question lies here: does it exist a protocol that allows them to decide on a winner in such a way that both parties feel secure that the other cannot fix the outcome? In classical cryptography, the answer is no [2]. However, two-party protocols whose security rely upon assumptions about the complexity of a computational task do exist (such as the the factorization of a large number into a product of two prime numbers, which is a common example of a secure one-way function). They are nevertheless threatened by quantum computation and are thus called insecure

from an information theoretic point of view. In addition, coin flipping can be done through trusted intermediaries.

13.2 Distance measures for quantum information

Before we investigate further our study about coin flipping, we introduce some definitions that will be useful in the rest of the report.

13.2.1 Classical case

1. Trace distance

   Mathematically, the trace distance is classically defined as

   $$D(p_x, qx) = \frac{1}{2} \sum_x |p_x - q_x| \tag{13.1}$$

   where $\{p_x\}$ and $\{q_x\}$ are probability distributions. Intuitively, it quantifies the 'closeness' or 'distinguishability' of two probability ditributions.

2. Fidelity

   Mathematically, the fidelity is defined as

   $$F(p_x, q_x) = \sum_x \sqrt{p_x \cdot q_x} \tag{13.2}$$

   where $\{p_x\}$ and $\{q_x\}$ are again probability distributions. Intuitively, it corresponds to the inner product between vectors with components $\sqrt{p_x}$ and $\sqrt{q_x}$ which lie on the unit sphere ($1 = \sum_x \left(\sqrt{p_x}\right)^2 = \sum_x \left(\sqrt{q_x}\right)^2$).

13.2.2 Quantum case

1. Trace distance

   The overall question we are trying to answer is how close are two quantum states? From the above definition (13.1), we can write

   $$D(\rho, \sigma) = \frac{1}{2} \operatorname{Tr} |\rho - \sigma| \tag{13.3}$$

where $|\mathrm{A}| = \sqrt{\mathrm{A}^\dagger \cdot \mathrm{A}}$. Note that the trace distance is invariant under unitary transformation U, i.e. $D(U\rho U^\dagger, U\sigma U^\dagger) = D(\rho, \sigma)$

Next, we will write down two theorems whose proofs can be found in [8]:

- Let $p_m = \mathrm{Tr}(\rho E_m)$ and $q_m = \mathrm{Tr}(\sigma E_m) \Rightarrow D(\rho, \sigma) = \max_{\{E_m\}} \; D(p_m, q_m)$

where $\{E_m\}$ is a Positive Operator-Valued Measurement (POVM); it is a set of positive operators (i.e. Hermitian operators with non-negative eigenvalues) that satisfy the completeness relation $\sum_m E_m = I$. This result states that if two density operators ρ and σ are close in trace distance, then any measurement performed on those quantum states (described by density operators) will give rise to probability distribution which are close together in the classsical sense of trace distance. Therefore, trace distance between two quantum states can be seen as an upper bound on trace distance between probability distributions arising from a measurement performed on those quantum states. In fact, trace distance is the maximal probability of distinguishing ρ and σ.

- No physical process ever increases the distance between two quantum states; in effect, trace-preserving quantum operations are contractive.

2. Fidelity

Proceeding analogically as before, we can extend (13.2) to the quantum case:

$$F(\rho, \sigma) = \mathrm{Tr} \sqrt{\sqrt{\rho} \cdot \sigma \cdot \sqrt{\rho}} \tag{13.4}$$

Again, we state two theorems without giving their proofs (cf [8]):

- *Uhlmann's theorem*: $F(\rho, \sigma) = \max_{|\psi\rangle, |\phi\rangle} \; |\langle\psi|\phi\rangle|$
  where $|\psi\rangle$ and $|\phi\rangle$ are purifications[1] of ρ resp. σ on a quantum system.

This theorem has interesting consequences, namely that the fidelity is symmetric in its inputs and that it is bounded between 0 and 1: $0 \leq F(\rho, \sigma) \leq 1$ Indeed, when ρ and σ have support on orthogonal subspaces, $F(\rho, \sigma) = 0$, so quantum states are perfectly distinguishable.

- Let $p_m = \mathrm{Tr}(\rho E_m)$ and $q_m = \mathrm{Tr}(\sigma E_m) \Rightarrow F(\rho, \sigma) = \min_{\{E_m\}} F(p_m, q_m)$

[1] any pure state $|\psi\rangle$ of a larger system that gives ρ if a part of the system is traced out is called a **purification of** ρ

This similarity in theorems for trace distance and fidelity leads us to point out that the latter can in some sense be seen as an 'upside-down' version of the former. In fact, fidelity decreases as two states become more distinguishable.

The fidelity also has a few useful properties. Firstly, the fidelity of a pure state $|\psi\rangle$ and a mixed state ρ is given by the overlap between them, i.e. $F(|\psi\rangle, \sigma) = \sqrt{\langle\psi| \rho |\psi\rangle}$.

Then, as for trace distance, it remains invariant under unitary transformation, i.e. $F(U\rho U^\dagger, U\sigma U^\dagger) = F(\rho, \sigma)$.

Now, what are the relationships between trace distance and fidelity? For pure states, both measures are completely equivalent and we can write $D(|\psi\rangle, |\phi\rangle) = \sqrt{1 - F(|\psi\rangle, |\phi\rangle)^2}$.
For any quantum states, we can bound trace distance on both sides by fidelity $1 - F(\rho, \sigma) \leq D(\rho, \sigma) \leq \sqrt{1 - F(\rho, \sigma)^2}$.

13.3 Quantum coin flipping

Having addressed the classical case, the following question then naturally arises: can we replace the computational assumptions of the classical case by information-theoretic security (i.e. security which does not rely on complexity assumptions and is considered as 'supreme' security) in the quantum case for coin flipping?
We foremost need to introduce the concept of bit commitment. A general quantum bit commitment scheme involves a sender - Alice - and a receiver - Bob. Ideally, the situation is as follows: Alice would like to be committed towards Bob, i.e. she wishes to provide Bob with a piece of evidence that she has a bit b in mind ($b = 0$ or 1) and that she cannot change it. Meanwhile, Bob should not be able to tell from that evidence what b is. At a later time, however, it must be possible for Alice to open the commitment, i.e. she must be able to show Bob which bit she has committed to, and convince him that it is indeed the genuine bit that she had in mind when she committed.
Mayers [9] and later Lo and Chau [2] showed that *all* proposed quantum bit commitment schemes following the laws of quantum physics are, in fact, insecure. The reason of the insecurity of ideal quantum bit commitment (with zero bias ϵ) is that Alice can always cheat successfully by using an EPR-type of attack and delaying her measurement until she opens her commitment [10, 9] (in effect, the existence of quantum computers would make the use of reversible unitary transformations possible, which allows her to postpone the measurement at

the end of the communication between her and Bob; cf [3] for a more in depth analysis).
Then why is bit commitment relevant in our discussion? Spekkens and Rudolph [6] have shown that a large set of bit-commitment based (strong) coin tossing protocols achieve the best trade-off when considering security vs. cheating. As we will see later, this means that the bias is optimal for many bit-commitment based protocols.
It is also essential that secure bit commitment protocol can be used trivially to implement a secure coin tossing protocol but the converse is *not* true [10, 11] (Alice chooses a bit b and commits it to Bob who then in turn tells Alice his guess for her bit; Alice then opens her commitment to see if Bob has guessed correctly).
Actually, bit commitment is a powerful primitive from which *all* other two-party secure computation protocols can be constructed [12]; its impossibility means that all other universal primitives for two-party secure computation must also be unrealizable using quantum information [2].
Lo and Chau [2] proved two lemmas that assert the impossibility of ideal quantum coin tossing if they initially do not share any entangled quantum state (following the impossibility of ideal bit commitment; cf [11]).

13.3.1 Definitions [4, 5, 13]

A **coin flipping protocol with bias** ϵ is one where Alice and Bob communicate and decide on a bit value $b \in \{0,1\}$ such that

- if both Alice and Bob are honest (ideal case), then $Prob(b = 0) = Prob(b = 1) = \frac{1}{2}$;
- if one of them is dishonest, then, for *any* strategy of the dishonest player, $Prob(b = 0) \leq \frac{1}{2} + \epsilon$, $Prob(b = 1) \leq \frac{1}{2} + \epsilon$.

The bias ϵ is the difference between their probability of winning and $\frac{1}{2}$, i.e. $\epsilon_{max} = (Prob_A, Prob_B) - \frac{1}{2}$. The goal for cryptographers has been to find such protocols with ϵ_{max} as small as possible.
Simply put, coin tossing is a two-party communication protocol that begins with a completely uncorrelated initial state and ends with each of the participants outputting a single bit. It has following requirements:

1. when both players are honest, Alice's output is uniformly random and equal to Bob's output;

2. if Alice is honest but Bob cheats (does not follow the protocol), then no matter what Bob does, the probability that Alice wins (outputs $b = 1$) is no greater than $Prob_B$;

3. similarly, if Bob is honest but Alice cheats, then the maximum probability for Bob to declare Alice winner is $Prob_A$.

We distinguish two kinds of coin flipping:

Strong coin flipping describes the following situation: Alice and Bob engage is some number of rounds of communication, at the end of which each infers the outcome of the protocol to be either 0, 1 or fail. If both are honest, then they agree on the outcome and find it to be 0 or 1 with equal probability. If one of the party X is dishonest, X cannot increase the probability of his/her opponent obtaining the outcome b to greater than $\frac{1}{2} + \epsilon_X^b$, for either $b = 0$ or $b = 1$. All the parameters ϵ_A^0, ϵ_A^1, ϵ_B^0 and ϵ_B^1 which specify the degree to which the protocol resists biasing, must each be strictly less than $\frac{1}{2}$.
A strong flipping protocol, as it name specifies, imposes the constraint that neither party can fix the outcome of the bit to be 0 or 1. This happens when Alice and Bob do not know which outcome their opponent favors; there are no a priori desired outcomes and the main task is to prevent either player from biasing the coin's outcome in either direction.

Weak coin flipping is a special case of strong flipping, namely the situation where there are no constraints on ϵ_A^0 nor on ϵ_B^1. The parameters ϵ_A^1 and ϵ_B^0 must be strictly less than $\frac{1}{2}$ and specify the bias-resistance of the protocol.
A weak coin tossing protocol is 'looser' than a strong coin tossing one in the sense that Alice cannot fix the outcome to be 1 and Bob cannot fix the outcome to be 0; such a protocol formalizes the 'real-life' situation where Alice wins if the outcome is 1 and Bob wins if the ouctome is 0: both have an a priori desired coin outcome.

13.3.2 Coin tossing protocols

Caveat: we will only describe the best known protocols; note that they do not necessarily achieve the lowest bounds found theoretically (for which protocols are still missing).

For the sake of clarity, we will again expose the inital situation: we are dealing with two distrustful parties wo are far apart from each other and no trustwor-

thy third person (who could flip the coin for them) is present; the only available resource to transmit information is a communication channel and they initially do not share any resources, but have access to trusted laboratories containing trusted error-free apparatus for creating and manipulating quantum states. According to quantum mechanics, every system is associated with a Hilbert space; in our case: $\mathcal{H}^A$, $\mathcal{H}^B$ and $\mathcal{H}^M$, where the indices A, B and M refer to Alice's and Bob's (private) systems and to the quantum channel between them (which can be seen as a mailbox) respectively. Observables of each system are described by operators on a specific system.

Strong coin tossing Ambainis [4] and Döscher and Keyl [5] have independently proposed protocols based on bit commitment which achieve a bias of $\epsilon = 0.25$. The fact that is based on bit commitment is not necessarily a generality; for example, Colbeck [14] proposed an entangled-based coin-flipping protocol which also achieves a bias of $\epsilon = 0.25$. Using his formalism, Kitaev (cf below) found an optimal protocol with a smaller bias of $\epsilon = \frac{1}{\sqrt{2}} - \frac{1}{2} \approx 0.21$
Here, we will look at Ambainis' protocol. Let's consider the following states:

$$|\phi_{b,x}\rangle = \begin{cases} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) & \text{if } b = 0, x = 0 \\ \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) & \text{if } b = 0, x = 1 \\ \frac{1}{\sqrt{2}}(|0\rangle + |2\rangle) & \text{if } b = 1, x = 0 \\ \frac{1}{\sqrt{2}}(|0\rangle - |2\rangle) & \text{if } b = 1, x = 1 \end{cases} \tag{13.5}$$

Alice and Bob proceed as follows (for each step of the protocol):

1. Alice picks a random bit value $b \in \{0,1\}$ and $x \in \{0,1\}$ and sends $|\phi_{b,x}\rangle$ to Bob.

2. Bob picks a random bit value $b' \in \{0,1\}$ and sends it to Alice.

3. Alice sends b and x to Bob; he then checks if the state he received from Alice in step (1) is really $|\phi_{b,x}\rangle$ by measuring it in a basis consisting of $|\phi_{b,x}\rangle$ and two vectors orthogonal to it.

   - If the outcome of the measurement is not $|\phi_{b,x}\rangle$, Bob has caught Alice cheating and he aborts the communication.
   - If the outcome of the measurement is $|\phi_{b,x}\rangle$, the result of the coin flip is $b \oplus b'$.

Ambainis distinguishes two different cases:

1. Alice is honest but Bob cheats → if $b = 0$, Alice sends either $|\phi_{0,0}\rangle$ or $|\phi_{0,1}\rangle$, each with 50% probability; if $b = 1$, Alice sends either $|\phi_{1,0}\rangle$ or $|\phi_{1,1}\rangle$, each with 50% probability. Therefore, the density matrices of these two mixed states are

$$\rho_{b=0} = \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad \rho_{b=1} = \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} \end{pmatrix}$$

and the trace distance between these two density matrices $D(\rho_{b=0}, \rho_{b=1}) = \frac{1}{2}$. From Helstrom [15], the probability that Bob achieves $b = b^{'}$ is at most $\frac{1}{2} + \frac{2 \cdot D(\rho_{b=0}, \rho_{b=1})}{4} = \frac{3}{4}$.

2. Bob is honest but Alice cheats → this case requires a more strenuous mathematical treatment where we 'symmetrize' Alice's strategy so that the task becomes easier (indeed, neither of the parties favors one outcome or the other). In this perspective, we will follow Ambainis' idea and prove three lemmas.

Lemma 1 *There is a strategy for a dishonest Alice where the state sent by her in step (1) has a density matrix of the form*

$$\rho^{'} = \begin{pmatrix} 1 - \delta_1 - \delta_2 & 0 & 0 \\ 0 & \delta_1 & 0 \\ 0 & 0 & \delta_2 \end{pmatrix} \tag{13.6}$$

for some δ_1 and δ_2 and Alice achieves $b = b^{'}$ with the same probability.

Proof Let's consider the following transformations:

$$U_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad U_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$U_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \quad U_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

Assume that Alice, before sending the state $|\psi\rangle$ to Bob in round (1), applies U_i to the state and then replaces later on in round (3) each description of $|\phi_{b,x}\rangle$ by $U_i |\phi_{b,x}\rangle$. As a consequence, Alice achieves the outcome 0 and 1 and gets caught with the same probability as before because for all $i \in \{0,1,2,3\}$, $b \in \{0,1\}$, $x \in \{0,1\}$, $U_i |\phi_{b,x}\rangle$ is either $U_i |\phi_{b,0}\rangle$ or $U_i |\phi_{b,1}\rangle$ AND because for any $|\psi\rangle$, the inner

product between $U_i |\psi\rangle$ and $U_i |\phi_{b,x}\rangle$ is the same as the one between $|\psi\rangle$ and $|\phi_{b,x}\rangle$.

Probabilities of obtaining 0, 1 and getting caught also stay the same if Alice picks a random $i \in \{0,1,2,3\}$ and then applies U_i to both the state sent in step (1) and the description sent in step (3). In this case, the density matrix of the state sent by Alice in round (1) is $\rho' = \frac{1}{4}\left(U_0\rho U_0^\dagger + U_1\rho U_1^\dagger + U_2\rho U_2^\dagger + U_3\rho U_3^\dagger\right)$. For every j, $k \in \{1,2,3\}$ with $j \neq k$, $\left(U_i\rho U_i^\dagger\right)_{jk}$ is equal to ρ_{jk} for two $i \in \{0,1,2,3\}$ and to $-\rho_{jk}$ for the other two i. Thus, $\rho'_{jk} = 0 \quad \forall j \neq k$, i.e. ρ' is of the form (13.6). **QED**

Lemma 2 *For a 'symmetrized' strategy of Alice, the probability that Alice convinces Bob that $b = 0$ is at most $F(\rho', \rho_0)^2$.*

Proof Let $|\psi\rangle = \sum_i a_i |i\rangle |\psi_i\rangle$ be the purification of ρ' chosen by Alice if she wants to convince Bob that $b = 0$. For every $|\psi_i\rangle$, Alice sends to Bob a description of a state $|\psi'_i\rangle$ which is one of $|\phi_{b,x}\rangle$, $b \in \{0,1\}$ and $x \in \{0,1\}$. Alice is trying to convince Bob that $b = 0$. As a matter of fact, we can assume that she always sends to Bob a description of $|\phi_{0,0}\rangle$ or $|\phi_{0,1}\rangle$ (replacing a description of $|\phi_{1,x}\rangle$ by a description of $|\phi_{0,x}\rangle$ can only increase the probability of Bob accepting $b = 0$, altough it may simultaneously increase the probability of Alice being caught cheating). We pair up each state $|\psi_i\rangle$ with the state $|\psi_j\rangle \equiv U_1 |\psi_i\rangle$ and each state $U_2 |\psi_i\rangle$ with $U_3 |\psi_i\rangle \equiv U_1 U_2 |\psi_i\rangle$. Our 'symmetrization' guarantees that

- if $|\psi_i\rangle$ and $|\psi_j\rangle$ are the two states in one pair, then $a_i = a_j$,
- if one of the states in a pair has $|\psi'_i\rangle = |\phi_{0,0}\rangle$, the other has $|\psi'_j\rangle = U_1 |\phi_{0,0}\rangle = |\phi_{0,1}\rangle$, and conversely,
- $\langle\psi_i|\psi'_i\rangle = \langle\psi_j|\psi'_j\rangle$.

Therefore, we can write the above purification $|\psi\rangle$ as

$$|\psi\rangle = \sum_i a_i \left[\frac{1}{\sqrt{2}}\left(|i,0\rangle |\psi_{i,0}\rangle + |i,1\rangle |\psi_{i,1}\rangle\right)\right] \tag{13.7}$$

with $|\psi'_{i,0}\rangle = |\phi_{0,0}\rangle$ and $|\psi'_{i,1}\rangle = |\phi_{0,1}\rangle$. The probability that Bob accepts $|\psi_{i,x}\rangle$ as $|\psi'_{i,x}\rangle$ is $\langle\psi_{i,x}|\psi'_{i,x}\rangle^2$. The total probability of Bob accepting is

$$\sum_i \frac{1}{2} |a_i|^2 \left(\left|\langle\psi_{i,0}|\psi'_{i,0}\rangle\right|^2 + \left|\langle\psi_{i,1}|\psi'_{i,1}\rangle\right|^2\right) \quad . \tag{13.8}$$

Notice that, because of 'symmetrization', $\langle\psi_{i,0}|\psi_{i,0}^{'}\rangle = \langle\psi_{i,1}|\psi_{i,1}^{'}\rangle$. Therefore, if we define $|\varphi_i\rangle = \frac{1}{\sqrt{2}}\big(\,|i,0\rangle\,|\psi_{i,0}\rangle + |i,1\rangle\,|\psi_{i,1}\rangle\,\big)$ and $|\varphi_i^{'}\rangle = \frac{1}{\sqrt{2}}\big(\,|i,0\rangle\,|\psi_{i,0}^{'}\rangle + |i,1\rangle\,|\psi_{i,1}^{'}\rangle\,\big)$, we have $\langle\varphi_i|\varphi_i^{'}\rangle = \langle\psi_{i,0}|\psi_{i,0}^{'}\rangle = \langle\psi_{i,1}|\psi_{i,1}^{'}\rangle$. This means that (13.8) is equal to

$$\sum_i |a_i|^2 \left|\langle\varphi_i|\varphi_i^{'}\rangle\right|^2 \qquad .$$

Let ρ_i by a mixed state which is $|\psi_{i,0}\rangle$ with 50% probability and $|\psi_{i,1}\rangle$ with 50% probability. Then, $\rho^{'} = \sum_i |a_i|^2 \rho_i$. Since $|\varphi_i\rangle$ and $|\varphi_i^{'}\rangle$ are purifications of ρ_i and ρ_0, we have $\left|\langle\varphi_i|\varphi_i^{'}\rangle\right|^2 \leq F(\rho_i,\rho_0)^2$ (from Uhlmann's theorem [8]) and

$$\sum_i |a_i|^2 \left|\langle\varphi_i|\varphi_i^{'}\rangle\right|^2 \leq \sum_i |a_i|^2 F(\rho_i,\rho_0)^2 \qquad .$$

By concavity of the fidelity [8], we get

$$\sum_i |a_i|^2 F(\rho_i,\rho_0) \leq F\left(\sum_i |a_i|^2 \rho_i, \rho_0\right)^2 = F(\rho^{'},\rho_0)^2 \qquad \textbf{QED}$$

Lemma 3 *The probability that Alice achieves $b\oplus b^{'} = 0$ (or equivalently $b\oplus b^{'} = 1$) is at most $\frac{1}{2}\big(F(\rho^{'},\rho_0)^2 + F(\rho^{'},\rho_1)^2\big)$.*

Proof With 50% probablity, Bob's bit is $b^{'} = 0$. Then, to achieve $b\oplus b^{'} = 0$, Alice needs to convince him that $b = 0$. By Lemma 2, she succeeds with probability at most $F(\rho^{'},\rho_0)^2$. With 50% probability, Bob's bit is $b^{'} = 1$. Then, Alice needs to convince Bob that $b = 1$ and she can do that with probability $F(\rho^{'},\rho_1)^2$. The overall probablity that Alice succeeds is $\frac{1}{2}\big(F(\rho^{'},\rho_0)^2 + F(\rho^{'},\rho_1)^2\big)$. **QED**

Using definition (13.4) of the fidelity, we get

$$F(\rho^{'},\rho_0)^2 = \left[\mathrm{Tr}\sqrt{\sqrt{\rho^{'}}\rho_0\sqrt{\rho^{'}}}\right]^2 = \left(\frac{1}{\sqrt{2}}\sqrt{1-\delta_1-\delta_2} + \frac{1}{\sqrt{2}}\sqrt{\delta_1}\right)^2 \qquad .$$

Similarly, $F(\rho^{'},\rho_1) = \left(\frac{1}{\sqrt{2}}\sqrt{1-\delta_1-\delta_2} + \frac{1}{\sqrt{2}}\sqrt{\delta_2}\right)^2$. Therefore,

$$\begin{aligned}
&\tfrac{1}{2}\big(F(\rho^{'},\rho_0)^2 + F(\rho^{'},\rho_1)^2\big) \\
&= \left[\tfrac{1}{\sqrt{2}}\left(\sqrt{1-\delta_1-\delta_2}+\sqrt{\delta_1}\right)\right]^2 + \left[\tfrac{1}{\sqrt{2}}\left(\sqrt{1-\delta_1-\delta_2}+\sqrt{\delta_2}\right)\right]^2 \\
&= \tfrac{1}{2}\Big((1-\delta_1-\delta_2) + \tfrac{\delta_1}{2} + \tfrac{\delta_2}{2} + \sqrt{1-\delta_1-\delta_2}\left(\sqrt{\delta_1}+\sqrt{\delta_2}\right)\Big) \qquad . \quad (13.9)
\end{aligned}$$

Let $\delta = \frac{\delta_1+\delta_2}{2}$; the convexity of the square root implies that $\sqrt{\delta_1}+\sqrt{\delta_2} \leq 2\sqrt{\delta}$ and (13.9) is at most $\frac{1}{2}\left(1-\delta+2\sqrt{\delta(1-2\delta)}\right)$.

Taking the derivative of this expression shows that it is maximized by $\delta = \frac{1}{6}$. Then, it is equal to $\frac{1}{2}\left(1 - \frac{1}{6} + \frac{4}{6}\right) = \frac{3}{4}$. **QED**

Weak coin tossing Prior to the works of Kitaev and Mochon [3], Spekkens and Rudolph [13] found a protocol with a bias $\epsilon = \frac{1}{2} - \frac{1}{\sqrt{2}} \approx 0.21$ but later on, a lower bias have been achieved by Mochon, who was, year after year, able to decrease the lowest attainable bias [16]. Indeed, in his recent paper [3], he proves that the lowest attainable bound for weak coin tossing is actually 0. He extensively introduces Kitaev's formalism, from which we give a brief introduction below (we will not cover this subject in depth as it lies outside the scope of this report). An important result also was derived by Ambainis [4] who has shown that a weak coin tossing protocol with bias $\epsilon > 0$ must have a number of rounds that grows as $\Omega(log\ log\frac{1}{\epsilon})$, making the Mochon's protocol asymptotically feasible (as the number of rounds tends to infinity as the bias ϵ tends to zero).

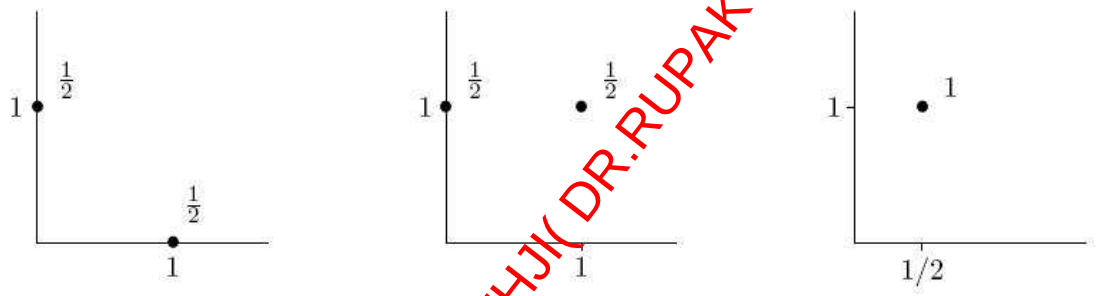


Figure 13.1: The sequence corresponds to a protocol with $Prob_A = 1$ and $Prob_B = \frac{1}{2}$, that is a protocol where Alice flips a coin and announces the outcome. Numbers outside the axes label location and numbers inside the axes label probability.

The formalism first introduced by Kitaev in 2003 (enhanced in 2004) can be described as a sequence of configurations, each of which consists of a number of marked points on a plane; these points can only take positive coordinate values and each point carries a probability, as can be seen from Figure 13.1. Two successive configurations can only differ by points on a single vertical or horizontal line. The rule is that the total probability on the line must be conserved (though the total number of points can change) and that for every $\lambda \in (0, \infty)$, we must satisfy $\sum_z \frac{\lambda \cdot z}{\lambda + z} p_z \quad \leq \quad \sum_{z'} \frac{\lambda \cdot z'}{\lambda + z'} p_{z'}$ where the left hand side is a sum over points before the transition and the right hand side is a sum over points after the transition. The variable z is respectively the x coordinate for transitions occurring

on a horizontal line or the y coordinate for transitions occurring on a vertical line. The numbers p_z are the probabilities associated to each point. If the final point is located at (x, y) then the resulting protocol will satisfy $Prob_A \leq y$ and $Prob_B \leq x$, and hence the bias ϵ is bounded by $\max_{(x,y)} -\frac{1}{2}$. Mochon [3] named these sequences 'point games' and they have the remarkable property that they are completely equivalent to standard protocols described by unitaries (existence of mappings from point games to standard protocols and vice-versa, even if finding easy-to-implement protocols with a small bias remains an open problem). By optimizing point games, Kitaev was able to find an optimal protocol. To achieve zero bias in coin flipping, similar constructions can be used with an infinite number of steps. Indeed, a protocol can be build such that

$$Prob_A = Prob_B = \frac{k+1}{2k+1} \qquad \forall k \geq 0 \tag{13.10}$$

The limit $k \to \infty$ achieves arbitrarily small bias.

13.3.3 Relativistic protocols

These protocols are very interesting since they offer the possiblity to perform strong coin tossing with zero bias ϵ. How can that be? The procedure is given by Colbeck [14] from an original idea by Kent [7], who proposed to use a variant of the 'standard' cryptographic scenario (which involves sequentially exchanged messages) in which each party controls two separated site. It runs as follows (please see Figure 13.2):

1. At time t_0, a trusted intermediary of Alice: A_1, sends a bit $b \in \{0,1\}$ to a trusted intermediary of Bob: B_1, choosing b randomly from a uniform distribtion.

2. Then, a second trusted intermediary of Bob: B_2, sends a bit b' to another trusted intermediary of Alice: A_2, also choosing b' randomly from a uniform distribution;

3. B_1 checks that his received message arrived before time t_0+D and so does A_2 in the same manner. If this is not the case, they abort the communication;

4. The disconnected agents of Alice communicate with one another, as do those of Bob. Alice and Bob can then compute the coin toss outcome $b \oplus b'$.

The impossibility of sending signals faster than the speed of light prevents either party from cheating in such a protocol. This is where relativity is important

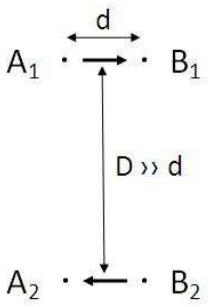


Figure 13.2: The impossibility of superluminal signalling means that information can be completely concealed from one party, at least for the light travel time. The distance D can be seen as the distance between adversary agents' laboratories, whereas d symbolizes the distance between one of Alice's agent and one of Bob's.

when we are looking at the security of the protocol for it has the advantage that perfectly concealing and perfectly binding communications between Alice and Bob can take place.
Note that the protocol is classical: it does not require the transmission or processing of quantum information. Nothing in it prevents either party from using quantum informations transmissions. Nevertheless, Kent [7] proves that it is secure in both classical and quantum cases.

13.4 Conclusion

In this report, we have seen that ideal non-relativistic classical and quantum protocol based on bit-commitment are impossible because of the Mayers-Lo-Chau no-go theorem [10, 9] which forbids unconditionally secure bit commitment. The following table summarizes the results for strong and weak quantum coin tossing:

Bias $\epsilon =$	Best protocol	Best lower bound
Strong coin flipping	0.25 [4]	$\frac{1}{\sqrt{2}} - \frac{1}{2} \approx 0.21...$ (Kitaev)
Weak coin flipping	0 [3]	$\epsilon > 0$, $\Omega(\log\log\frac{1}{\epsilon})$ rounds required [4]

The bounds for strong coin flipping are quite close and different protocols based on bit commitment [4, 5, 7] or entanglement [14] achieve the same lower bias of $\epsilon = \frac{1}{4}$. For weak coin tossing, a protocol with arbitrarily small bias has to

be found using Kitaev's formalism [3]. However, Ambainis [4] gives a general lower bound on the number of rounds needed to achieve a bias ϵ. Until now, the best known weak coin-flipping protocol - given by Mochon [16] - asymptotically achieves a bias of $\epsilon = \frac{1}{6}$. Relativistic protocols proposed by Kent [7] offer the possibility to run secure protocols for relativity is introduced as a means to guarantee the security of the protocol, which relies on the fact that the velocity of the signal is smaller than the light velocity.

Bibliography

[1] M. Blum, *Coin flipping by telephone. A protocol for solving impossible problems*, SIGACT News **15**, 23 (1981).

[2] H.-K. Lo and H. F. Chau, *Why quantum bit commitment and ideal quantum coin tossing are impossible*, Phys. Rev. Lett. **78**, 3410 (1997).

[3] C. Mochon, *Quantum weak coin flipping with arbitrarily small bias* (2007), (arxiv:0711.4114).

[4] A. Ambainis, *A New Protocol and Lower Bounds for Quantum Coin Flipping*, Journal of Computer and System Sciences **68**, 398 (2002).

[5] C. Döescher and M. Keyl, *An introduction to quantum coin-tossing* (2002), (quant-ph/0206088).

[6] R. W. Spekkens and T. Rudolph, *Degrees of concealment and bindingness in quantum bit commitment protocols*, Phys. Rev. A **65**, 012310 (2002).

[7] A. Kent, *Unconditionally secure bit commitment*, Phys. Rev. Lett. **83**, 1447 (1999).

[8] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2005).

[9] D. Mayers, *Unconditionally secure quantum bit commitment is impossible*, Phys. Rev. Lett. **78**, 3414 (1997).

[10] H.-K. Lo, *Insecurity of quantum secure computations*, Phys. Rev. A **56**, 1154 (1997).

[11] A. Kent, *Coin tossing is strictly weaker than bit commitment*, Phys. Rev. Lett. **83**, 5382 (1999).

[12] A. C. Yao, *Security of quantum protocols against coherent measurements*, 27th Symposium on Theory of Computing, ACM Press **29**, 67 (1995).

[13] R. W. Spekkens and T. Rudolph, *Quantum protocol for cheat-sensitive weak coin flipping*, Phys. Rev. Lett. **89**, 227901 (2002).

[14] R. A. Colbeck, Ph.D. thesis, University of Cambridge (2007).

[15] C. Helstrom, *Quantum Detection and Estimation Theory*, Journal of Statistical Physics **1**, 231 (1976).

[16] C. Mochon, *Quantum Weak Coin-Flipping with Bias of 0.192* (2004), (arxiv:quant-ph/0403193).

Chapter 14

Non-Abelian quantum computing

Fabio Pedrocchi
supervisor: Sergei Isakov

Although there exist quantum correcting protocols, decoherence is always a big problem of quantum computation. Such correcting codes require in effect that a quantum computer is able to perform between 10^4 and 10^6 operations perfectly before an error occurs. Such a big precision is very difficult (or even perhaps impossible) to achieve. The aim of this chapter is to describe a model for a topological quantum computation which is protected against local perturbations from the environment. This model relies on topological phase of matter whose quasiparticle excitations follow an interesting statistics. These quasiparticles, so-called anyons, which we might find for example in quantum Hall states, doesn't follow in general fermionic or bosonic statistics. We will describe how to encode qubits with anyons in a non-local way and how to manipulate the qubits also in a non-local way by exchanging anyons. This non-local encoding and manipulation of the quantum information leads to immunity against local perturbations from the environment and thus immunity against decoherence.

14.1 Introduction

As we have seen in the previous chapters, one of the big problems with quantum computation is decoherence, i.e. local interactions with the environment. One way to overcome this problem is to "hide" the quantum information in topologically degenerate states, i.e. states which cannot be distinguished by local measurements. We will actually see that one promising possibility to do this is to use particular quantum Hall states whose quasiparticle excitations follow a statistics that isn't in general fermionic or bosonic. We will see how to use these quasiparticles, called (non-abelian) anyons, to encode quantum information in a non-local way and then how to manipulate them using non-local operations. There are other, less realistic models than quantum Hall states, that have non-abelian excitations [1], [2], [3]. We won't consider these models in this chapter.
In the first section we will thus review the basic properties of the quantum Hall effect from a phenomenological point of view. In the second section we will introduce the concept of anyons and describe in detail some of their properties that are useful for quantum computation. In the last section, we will consider a precise model of anyons, namely the Fibonacci model, which is universal for quantum computation. We will describe in detail how we can build a universal set of gates by braiding Fibonacci anyons.

14.2 Quantum Hall effect and topological quantum computation

First, I want to give a phenomenological reminder of the quantum Hall effect (QHE) [5], [6]. Let's consider a gas of electrons trapped in a two dimensional plane. Then let's apply a strong magnetic field perpendicular to it and let's cool the system down to a low temperature ($\approx 1mK$). Under such conditions, the resistivity of the system presents an interesting behavior. It turns out that $\rho_{xx} = 0$ and $\rho_{xy} = \frac{1}{\nu}\frac{e^2}{h}$ (i.e. ρ_{xy} takes quantized values), where ν is called *filling factor*. If ν is an integer, we speak of integer QHE and if ν is a fraction, we speak of fractional QHE. In Fig. (14.1), we see that the plot of R_{xy} as a function of the external magnetic field presents plateaus where the value of R_{xy} is constant and on the bottom plot we can see that the corresponding longitudinal resistivity vanishes.
Let's now consider the $\nu = 1/3$ quantum Hall state. The main idea is that the ground state (or vacuum state), i.e. the state where no quasiparticles are present at all, can be described by an incompressible fluid, i.e. there is an energy gap

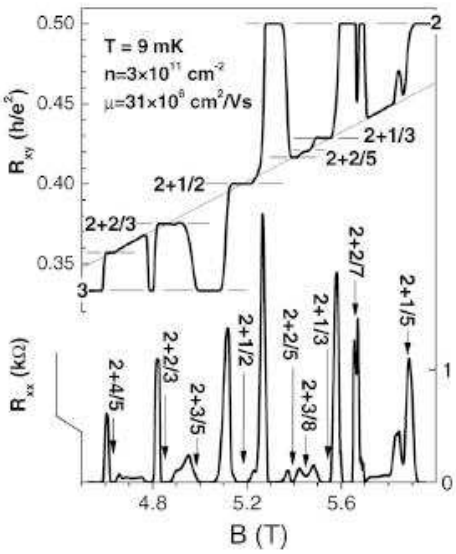


Figure 14.1: Top: Plot of the transverse resistivity as a function of the external magnetic field. Bottom: Plot the longitudinal resistivity as a function of the external magnetic field. This figure is taken from [9].

between the ground state and all the excited states. A very remarkable property is that such a ground state has quasiparticle excitations which follow a statistics which isn't in general fermionic or bosonic. We call such quasiparticles *abelian anyons* (more details on this in the next section). Basically the exchange of two abelian anyons produces a phase $e^{i\theta}$ in front of the wave function. This phase isn't in general ± 1 [7]. Another very interesting fact is that the ground state of such a system is degenerated when we put it on a non-trivial surface. On a sphere the vacuum state exhibits a one-fold degeneracy (i.e. the ground state is unique), on a torus a 3-fold degeneracy and on a surface with genus N a 3^N-fold degeneracy. We can understand this by considering a process in which a particle-antiparticle pair is created, one particle winds around the torus along a closed loop and then the pair annihilates. If we consider two operators A and B describing this process and corresponding to the two basic loops on a torus (note that these operators preserve the ground state), then the operator $ABA^{-1}B^{-1}$ is topologically equivalent to a process where one particle winds around the other one and thus the effect of this process is in general a non-trivial phase $e^{i\alpha}$. This induces that A and B don't commute and thus that they act on a degenerate space (for more details see [8]). This degeneracy depends only on the topology of the surface and thus no local measurements can distinguish the different ground states. We can now understand that it would be a remarkable place to put quantum information if

we could use the topological degrees of freedom to encode qubits. Although we can't build a universal quantum computer by using abelian-anyons they could be useful to store quantum information [9]. As you can foresee from the title of this chapter, we won't consider abelian anyons but non-abelian anyons, one reason for this is that non-abelian anyons present much more powerful properties in order to build a topological quantum computer. We will see that we can store quantum information using non-abelian anyons and also *manipulate* it in a non-local way by exchanging the quasiparticles.

To illustrate this, let's consider the $\nu = \frac{5}{2}$ quantum Hall state. Moore and Read proposed in 1991 [10] that this quantum Hall state exhibits quasiparticle excitations which are non-abelian anyons. Another interesting property is that these non-abelian anyons have fractional charge, namely $\frac{e}{4}$. In the next section we will see in detail what a non-abelian anyon is. One condition to find non-abelian anyons is the presence of a set of degenerate states. When two such quasiparticles are exchanged this induces a unitary transformation on this set of degenerate states. The $\nu = \frac{5}{2}$ state with $2n$ quasiparticles exhibits a 2^{n-1}-fold degeneracy (which is required for non-abelian statistics) and this is a topological degeneracy if the quasiparticles are kept sufficiently far apart, i.e. no local measurements can distinguish the different states [11]. However, if the quasiparticles are brought close to one another the degeneracy is broken.

The basic idea of our topological quantum computation is thus to encode qubits using different distant quasiparticles. A qubit is thus a non-local entity composed by well separated quasiparticles. We will see that the different states of the qubit correspond to different values of an internal quantum number (q-spin) of this set of quasiparticles. The next step is to find a method to manipulate the qubits with operations insensible to local perturbations from the environment. We will see that exchanging two quasiparticles is such a global operation which induces a unitary transformation on the set of degenerate states. Since the storage and the manipulation of the information is topologically protected, our computation is immune against decoherence.

14.3 Anyons and braid statistics

As we have seen in the previous section, the basic ingredient for this model of topological quantum computation is anyons. I thus want to introduce the concept of anyons in this section and present in detail the properties of anyons which are useful for quantum computation.

As you know from quantum mechanics, in 3+1 dimensions (3 spatial and 1 time

dimension) there are two types of particles, namely: bosons and fermions. If we consider N indistinguishable particles there are two different behaviors under particles exchange:

$$\begin{aligned}\Psi(1,...,j,...,i,...,N) &= +\Psi(1,...,i,...,j,...,N) \text{ for Bosons} \\ \Psi(1,...,j,...,i,...,N) &= -\Psi(1,...,i,...,j,...,N) \text{ for Fermions,}\end{aligned}$$

where Ψ is the wave function of this set of indistinguishable particles.
As you already know, the two prefactors (± 1) correspond to the two 1-dimensional irreducible representations of the permutation group of N elements (S_N). One could now ask if there is any other type of particles which doesn't obey bosonic or fermionic statistics. The remarkable answer is yes! Although the concept of anyons was initially introduced as a mathematical construction in which any phase factor $e^{i\alpha}$ could result from a counterclockwise exchange of two particles, this isn't only a mathematical concept since we might find them in some quantum Hall states. To summarize, we are interested in particles which can get any possible phase α under particles exchange, i.e.

$$\Psi(r_1, r_2) \rightarrow e^{i\alpha}\Psi(r_1, r_2).$$

It is relatively simple to prove that we can only have anyons in a 2+1 dimensional system and not in a 3+1 dimensional one. Through lack of space I will not prove this here, but the basic idea behind this is topological: the configuration space of N indistinguishable particles in 3+1 dimensions is simply connected and this isn't the case in 2+1 dimensions. This implies that in 2+1 dimensions a loop (exchanging two times two particles is the same as taking one particle around the other one) cannot be shrunk to a point as in 3+1 dimensions. The phase corresponding to two successive particles exchanges should thus not be necessarily 0 or π (see Ref.[12] for more information about this).
In the same sense that there is a link between bosons, fermions and the permutation group S_N, there is a link between anyons and the so called braid group B_N. To understand this we can use Feynman's path integral formalism (the following discussion comes from [12]). Let's consider two points (x_1, t_1) and (x_N, t_N) in the configuration space of N indistinguishable particles, then the transition amplitude is given by

$$\langle x_N, t_N | x_1, t_1 \rangle \propto \sum_{all\ paths} \exp\left(\frac{iS}{\hbar}\right) = \sum_{all\ paths} \exp\left(\frac{i}{\hbar}\int_{t_1}^{t_N} L\, dt\right), \qquad (14.1)$$

where S is the trajectory's classical action and L the corresponding Lagrangian. Let's now consider closed paths (i.e $x_1 = x_N$). We are interesting in the proportionality constant of Eq.(14.1). There is no reason why this factor should be the

same for two paths which aren't homotopic. We can thus rewrite equation (14.1) as

$$\langle x_1, t_1 | x_N, t_N \rangle = \sum_{all\ closed\ paths} \chi(\gamma) \sum_{q(t)\in\gamma} \exp\left(\frac{i}{\hbar} L(q,\dot{q})\, dt\right).$$

Now remembering that we assign the same amplitude for the time development of a trajectory between t_0 and t_1 followed by the time development between t_1 and t_2 as we assign for the time development between t_0 and t_2, we find :

$$\chi(\gamma_1\gamma_2) = \chi(\gamma_1)\chi(\gamma_2) \tag{14.2}$$

From Eq.(14.2) we conclude that $\chi(\gamma)$ is a 1-dimensional representation of the homotopy group π_1 of the configuration space of N indistinguishable particles. Our mathematician friends found the following very important results (it is remarkable to note that they found it at the same time than the physicists were interested in this problem!):

$$\begin{aligned} \pi_1((R^d - \Delta)/S_N) &= B_N \ for\ d = 2 \\ &= S_N \ \ for\ d \geq 3, \end{aligned}$$

where B_N is the braid group, $(R^d - \Delta)/S_N$ is the configuration space of N indistinguishable particles in d dimensions and $\Delta = ((x_1, ..., x_N) \in (R^d)^N : x_i = x_j$ for at least one pair $i \neq j)$. Thus we see that in 2+1 dimensions the exchange of particles can lead to a factor χ which is a 1-dimensional representation of the braid group. The braid group is a generalization of the permutation group. From an algebraic point of view it is generated by elements σ_i (σ_i interchanges element i and $i+1$) which satisfy:

$$\begin{aligned} \sigma_i\sigma_{i+1}\sigma_i &= \sigma_{i+1}\sigma_i\sigma_{i+1} \text{ for } i = 1, 2, ..., N-2, \\ \sigma_i\sigma_j &= \sigma_j\sigma_i \ \ for\ |i-j| \geq 2. \end{aligned}$$

An element of the braid group can be visualized by thinking of the world lines of the anyons with the time along the z-axis (Fig.(14.2)). The most important difference between S_N and B_N is that in B_N $\sigma_i \neq \sigma_i^{-1}$. This implies that the phase we get when we exchange two particles in 2 dimensions can take **any** value $e^{i\alpha}$ and not just ± 1, that's why we call these particles abelian **any**ons. We add the word *abelian* to remind that they correspond to *abelian* representations of the braid group.

As I mentioned in the previous section, there exists another type of anyons: *non-abelian* anyons. They correspond to higher dimensional representations of the

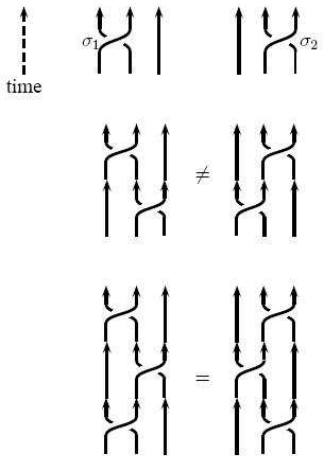


Figure 14.2: Top:σ_1, σ_2. Middle:the group isn't abelian: $\sigma_2\sigma_1 \neq \sigma_1\sigma_2$. Bottom:$\sigma_i\sigma_{i+1}\sigma_i = \sigma_{i+1}\sigma_i\sigma_{i+1}$. This picture is taken from [5].

braid group. To have a higher dimensional representation of the braid group we have to consider a set of g degenerate states, $\psi_\alpha, \alpha = 1, 2, ..., g$, of quasiparticles at fixed positions. Then the effect of braiding two particles (e.g. σ_1) is represented by a $g \times g$ unitary matrix:

$$\Psi = \alpha_1\psi_1 + ... + \alpha_g\psi_g \equiv \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ ... \\ \alpha_g \end{pmatrix} \rightarrow \Psi' \equiv \rho(\sigma_1) \cdot \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ ... \\ \alpha_g \end{pmatrix}.$$

If the braid group representation is non-abelian, i.e. $[\rho(\sigma_i), \rho(\sigma_j)] \neq 0$ for some i and j, we call the corresponding anyons, *non-abelian anyons*.
The very important result is that the matrix ρ does only depend on the topology of the braid. In [13] it is shown that in quantum Hall states the abelian part of the statistics can be perturbed by the geometry of the paths, however the non-abelian part (the one important for quantum computation) depends only on the **topology** of the braid. If this wasn't the case, this model of quantum computation wouldn't be robust against decoherence. Actually, if this wasn't the case, any perturbation from the environment (e.g. quasiparticles being scattered by photons) which change the geometry of the path of the exchanged anyons would also influence the corresponding unitary operation leading to a sensibility to local perturbations from the environment. As I already mentioned, in this model a qubit is a non-local entity composed by well separated anyons (we will see the details in the next section), thus since the encoding of the quantum information is non-local and since the manipulation of this information by exchanging the

anyons is also non-local (as we have just seen), our model for a topological quantum computation is insensible to decoherence.
Let's now consider some basic and important properties of anyons. We assign to every type of anyons a label $(a, b, c...)$, called q-spin or topological charge. The label is a quantum number which cannot be changed by any local physical process. The possibility that a particle a and a particle b fuse to a particle c when they are 'brought together' is described by the fusion rule:

$$\phi_a \times \phi_b = \sum_c N_{ab}^c \phi_c. \tag{14.3}$$

By fusion of two particles a and b we mean that we consider these two anyons as a bound object. The composite object behaves exactly as a single anyon c. An analogy with the ordinary spin makes this relation more clear. If we consider two electrons of spin $\frac{1}{2}$ the addition rule is:

$$\frac{1}{2} + \frac{1}{2} = 1 + 0.$$

Thus two fermions of spin $\frac{1}{2}$ can either fuse to a boson of spin 1 or a boson of spin 0 (i.e. the corresponding composite object behaves either as a boson with spin 0 or as a boson with spin 1). This is only an analogy but it makes things more clear. It is very important to note that in a model of abelian anyons, any two particles fuse in a unique way. However, in a non-abelian anyon model there is at least one pair a, b with

$$\sum_c N_{ab}^c \geq 2.$$

The different fusion channels of non-abelian anyons is one way to account for the degeneracy of the multi-particle states. Let's consider, for example, a model of non-abelian anyons (called *Fibonacci anyons*, see the next section) with two particles τ and 1 and the following fusion rule:

$$\tau \times \tau = 1 + \tau, \tag{14.4}$$

where 1 is the trivial particle, also called vacuum because it is equivalent to having no particle at all. If we represent one anyon by $\bullet$, we can represent the three basis states of the Hilbert space as

$$|((\bullet,\bullet)_\tau,\bullet)_\tau\rangle \quad |((\bullet,\bullet)_1,\bullet)_\tau\rangle \quad |((\bullet,\bullet)_\tau,\bullet)_1\rangle\,, \tag{14.5}$$

where $(\bullet,\bullet)_a$ means that the two anyons fuse to a third anyon a. Thus we see

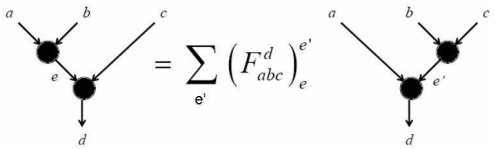


Figure 14.3: The associativity of the fusion rule implies that we can go from one basis to the other with the matrix F. This figure is taken from [8].

that the fusion rule (14.4) accounts for the degeneracy of the state (keeping in mind that we always consider that the quasiparticles are kept sufficiently far apart in the corresponding quantum Hall state, if this isn't the case the degeneracy is broken).
An important property of the fusion rule is that it is associative, i.e.

$$(a \times b) \times c = a \times (b \times c).$$

Intuitively we can understand this associativity as follows. I didn't mention this, but the q-spin describes the total charge carried by an anyon. The charge of a system of three anyons is an intrinsic property of these three particles and doesn't depend on the way they are fused. Thus in the previous anyon model (14.4) we could have fused the two rightmost anyons first. The basis states would then look like this:

$$|(\bullet,(\bullet,\bullet)_\tau)_\tau\rangle \quad |(\bullet,(\bullet,\bullet)_1)_\tau\rangle \quad |(\bullet,(\bullet,\bullet)_\tau)_1\rangle\,. \tag{14.6}$$

The basis change between (14.5) and (14.6) is parametrised by a matrix F:

$$|(\bullet,(\bullet,\bullet)_i)_k\rangle = \sum_j [F_k^{\tau\tau\tau}]_{ij}\,|((\bullet,\bullet)_j,\bullet)_k\rangle\,. \tag{14.7}$$

In general, if we consider three anyons a, b, c which fuse to a third anyon d and with fusion rule $a \times b = e_1 + e_2 + \ldots$ and $b \times c = e_1' + e_2' + \ldots$ then the basis change between $|((a,b)_{e_1},c)_d\rangle$ $|((a,b)_{e_2},c)_d\rangle$, ... and $\left|(a,(b,c)_{e_1'})_d\right\rangle$ $\left|(a,(b,c)_{e_2'})_d\right\rangle$, ... can be represented by Fig.(14.3).
Finally, I have to introduce the R-matrix in order to fully describe the anyons properties that are important for quantum computation. We describe by R^c_{ab} the phase acquired when we (counterclockwise) exchange two anyons a and b which fuse to a third anyon c. Actually the effect of braiding two anyons doesn't affect the overall q-spin of the composite object. To explain this, we have to mention that every anyon carries another quantum number called the topological spin (please do not confuse with the q-spin) and which basically describes the effect of rotating a particle by 2π (for more details see [8] or [5]):

$$U(2\pi) = e^{-i2\pi S}, \tag{14.8}$$

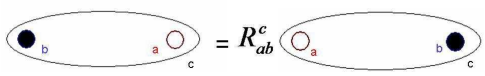


Figure 14.4: The effect of R^c_{ab} doesn't change the overall q-spin.

where S is the topological spin of the anyon. Intuitively we can understand that exchanging particles a and b two times is the same as rotating particle c by 2π up to some corrections (actually when we rotate particle c by 2π, we also rotate particles a and b by 2π). The effect of such a rotation is thus

$$(R^c_{ab})^2 = e^{-2\pi i S_c} e^{2\pi i S_a} e^{2\pi i S_b}, \tag{14.9}$$

where $(R^c_{ab})^2$ is called monodromy operator.
Thus if we consider two particles a and b with fusion rule $a \times b = c_1 + c_2 + ...$ the monodromy operator doesn't change the overall q-spin (see Fig.(14.4)):

$$\begin{pmatrix} |c_1\rangle \\ |c_2\rangle \\ ... \\ ... \end{pmatrix} \rightarrow \begin{pmatrix} e^{-2\pi i S_{c_1}} e^{2\pi i S_a} e^{2\pi S_b} & 0 & 0 & 0 \\ 0 & e^{-2\pi i S_{c_2}} e^{2\pi i S_a} e^{2\pi S_b} & 0 & 0 \\ . & & . & . \\ . & & . & . \end{pmatrix} \cdot \begin{pmatrix} |c_1\rangle \\ |c_2\rangle \\ ... \\ ... \end{pmatrix}. \tag{14.10}$$

We conclude that R^c_{ab} is only a phase factor.
Until now we have introduced the basic properties of anyons which will be of major importance for the next section. In some sense, the next section is an application of what we have just seen in the context of (topological) quantum computation.

14.4 Quantum computation with Fibonacci anyons

14.4.1 Fibonacci model and structure of the Hilbert space

In this section we focus on a specific model of anyons which can be used for universal topological quantum computation [8], [14]. The Fibonacci model is pretty simple, there are 2 particles, the trivial 1 and the nontrivial τ, which satisfy the following fusion rule:

$$\tau \times \tau = 1 + \tau.$$

The reasons why we focus on these Fibonacci anyons is because we hope to find them in the $\nu = \frac{12}{5}$ quantum Hall state [15] and because it allows us to achieve universal topological quantum computation. The first question you may ask is: why do we call these particles Fibonacci anyons? The answer is quite simple, the dimension of the Hilbert space of N Fibonacci anyons is the $(N+1)$-th Fibonacci number. Let's take the example of two particles. In this case the Hilbert space is 2-dimensional (2 is the 3rd Fibonacci number) with basis

$$|(\bullet,\bullet)_1\rangle \quad and \quad |(\bullet,\bullet)_\tau\rangle\,.$$

For three particles, we have three basis-states (3=4th Fibonacci number), namely

$$|((\bullet,\bullet)_1,\bullet)_\tau\rangle\,, \quad |((\bullet,\bullet)_\tau,\bullet)_\tau\rangle \quad and \quad |((\bullet,\bullet)_\tau,\bullet)_1\rangle\,. \tag{14.11}$$

We can generalize this to N anyons counting the different paths in a fusion diagram and see that the dimension of the Hilbert space grows as the Fibonacci sequence [16].
In order to find the effect of σ_1 (braiding the first and the second particles) or σ_2 (braiding the second and the third one) we have first to find the F-matrix (defined in Eq.(14.7)). The way to find it is to use a consistency equation called *pentagon equation*. In Fig.(14.5) we consider four particles which fuse to a fifth

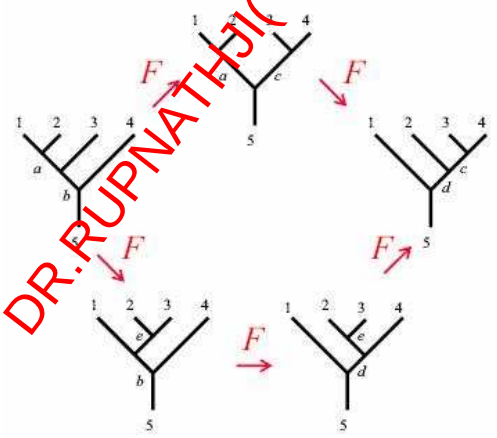


Figure 14.5: We can move from one basis to the other one by using the F-matrix. Following the upper or the lower path should give the same result. This figure is taken from [8].

particle. The different basis-states are represented by 'trees'. We can go from one basis to another one by applying the F-matrix. Following the upper or the

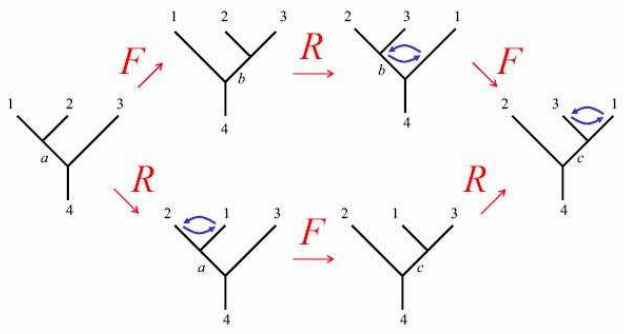


Figure 14.6: Following the upper or the lower path should give the same result. This figure is taken from [8].

lower path should lead to the same result:

$$|(((1,2)_a,3)_b,4)_5\rangle = \sum_{c,d} |(1,(2,(3,4)_c)_d)_5\rangle\,(F^5_{12c})^d_a (F^5_{a34})^c_b \tag{14.12}$$

$$|(((1,2)_a,3)_b,4)_5\rangle = \sum_{c,d,e} |(1,(2,(3,4)_c)_d)_5\rangle\,(F^d_{234})^c_e (F^5_{1e4})^d_b (F^b_{123})^e_a \tag{14.13}$$

Equating Eq.(14.12) and Eq.(14.13) we finally find the *pentagon equation*:

$$(F^5_{12c})^d_a (F^5_{a34})^c_b = \sum_e (F^d_{234})^c_e (F^5_{1e4})^d_b (F^b_{123})^e_a \tag{14.14}$$

For the Fibonacci theory there are only two F matrices that we have to consider, namely the 1×1 $F^1_{\tau\tau\tau}$ and the 2×2 $F^\tau_{\tau\tau\tau}$. Since there is only one state with total q-spin 1, $(F^1_{\tau\tau\tau})^b_a = \delta^\tau_a \delta^b_\tau$ and then using the pentagon equation Eq.(14.14) we find $F^\tau_{\tau\tau\tau}$

$$F = \begin{pmatrix} \tau & \sqrt{\tau} \\ \sqrt{\tau} & -\tau \end{pmatrix},$$

where $\tau = \frac{\sqrt{5}-1}{2}$.

Now the next step is to find the R-matrix. In order to do this, we will use another consistency equation, called *hexagon equation*. In Fig.(14.6), we consider three particles fusing to a fourth one. The different basis states of this system are represented by 'trees'. As before we can follow either the upper or the lower path:

$$|((1,2)_a,3)_4\rangle = \sum_{b,c} |(2,(3,1)_c)_4\rangle\,(F^4_{231})^c_b R^4_{1b} (F^4_{123})^b_a \tag{14.15}$$

$$|((1,2)_a,3)_4\rangle = \sum_{c} |(2,(3,1)_c)_4\rangle\,R^c_{13} (F^4_{123})^c_a R^a_{12} \tag{14.16}$$

Figure 14.7: Representation of σ_1 and σ_2. This figure is taken from [5].

Equating Eq.(14.15) and Eq.(14.16) we find the *hexagon equation*:

$$R^c_{13}(F^4_{213})^c_a R^a_{12} = \sum_b (F^4_{231})^c_b R^4_{1b} (F^4_{123})^b_a .$$

Using our above result for F we find

$$R_{counterclockwise} = \begin{pmatrix} e^{-\frac{4\pi i}{5}} & 0 \\ 0 & -e^{-\frac{2\pi i}{5}} \end{pmatrix} . \tag{14.17}$$

The conjugated matrix is also a solution and corresponds to the clockwise exchange.

We have now all the ingredients to find which unitary operation corresponds to which braid. Let's rewrite the three basis state of the Hilbert space for our three Fibonacci anyons in a way which will be practical for computation's theory:

$$|0\rangle := |((\bullet,\bullet)_1,\bullet)_\tau\rangle\,, \quad |1\rangle := |((\bullet,\bullet)_\tau,\bullet)_\tau\rangle \quad and \quad |N\rangle := |((\bullet,\bullet)_\tau,\bullet)_1\rangle\,. \tag{14.18}$$

From Eq.(14.17) it is easy to find:

$$\begin{pmatrix} |0\rangle \\ |1\rangle \\ |N\rangle \end{pmatrix} \to \underbrace{\begin{pmatrix} e^{-\frac{4\pi i}{5}} & 0 & 0 \\ 0 & -e^{-\frac{2\pi i}{5}} & 0 \\ 0 & 0 & -e^{-\frac{2\pi i}{5}} \end{pmatrix}}_{\rho(\sigma_1)} \cdot \begin{pmatrix} |0\rangle \\ |1\rangle \\ |N\rangle \end{pmatrix} .$$

At this point it is useful to show how we can find the unitary matrix corresponding to σ_2. It is more complicated but I find this interesting. We first have to change the basis to see how the two rightmost particles fuse. Then we apply R and then we undo the basis change, i.e. $\rho(\sigma_2) = FRF^{-1}$. From our previous results for F and R we find (here $F^{\tau\tau\tau}_\tau = F$) :

$$\begin{aligned} |0\rangle &= F_{11}\,|(\bullet,(\bullet,\bullet)_1)_\tau\rangle + F_{21}\,|(\bullet,(\bullet,\bullet)_\tau)_\tau\rangle \\ (Rotation) \Rightarrow &\quad e^{\frac{-4\pi i}{5}} F_{11}\,|(\bullet,(\bullet,\bullet)_1)_\tau\rangle - e^{\frac{-2\pi i}{5}} F_{21}\,|(\bullet,(\bullet,\bullet)_\tau)_\tau\rangle \\ \Rightarrow \rho(\sigma_2)\,|0\rangle &= ([F^{-1}]_{11} e^{\frac{-4\pi i}{5}} F_{11} - [F^{-1}]_{12} e^{\frac{-2\pi i}{5}} F_{21})\,|0\rangle + \\ &+ ([F^{-1}]_{21} e^{\frac{-4\pi i}{5}} F_{11} - [F^{-1}]_{\tau\tau} e^{\frac{-2\pi i}{5}} F_{21})\,|1\rangle \\ &= -\tau e^{\frac{-\pi i}{5}}\,|0\rangle - i\sqrt{\tau} e^{\frac{-i\pi}{10}}\,|1\rangle\,. \end{aligned}$$

We can perform exactly the same kind of calculations for $|1\rangle$ and $|N\rangle$ and finally find:

$$\rho(\sigma_2) = \begin{pmatrix} -\tau e^{\frac{-\pi i}{5}} & -\sqrt{\tau} i e^{\frac{-i\pi}{10}} & 0 \\ -\sqrt{\tau} i e^{\frac{-i\pi}{10}} & -\tau & 0 \\ 0 & 0 & -e^{\frac{-2\pi i}{5}} \end{pmatrix}.$$

Since σ_1, σ_2, σ_1^{-1} and σ_2^{-1} generate all possible braids, the corresponding unitary operations are generated by $\rho(\sigma_1)$, $\rho(\sigma_2)$, $\rho(\sigma_1)^{-1}$ and $\rho(\sigma_2)^{-1}$.

14.4.2 Computation with Fibonacci anyons

In order to perform quantum computation, we first need to construct qubits. As I already mentioned, we construct qubits with anyons which are far away from each other and the different states of the qubit are given by the value of the q-spin of the set of these quasiparticles (see Eq.(14.18)). To summarize we delocalise the qubit on the whole system. The very first idea we could have is to use 2 anyons, i.e. we could define the two orthogonal states of the qubit as

$$|(\bullet,\bullet)_1\rangle \quad and \quad |(\bullet,\bullet)_\tau\rangle\,.$$

However this isn't a good idea. The reason for this comes from our previous discussion, where we concluded that the exchange of two particles cannot change the overall q-spin of the two particles. This implies that it isn't possible to make a transition from $|(\bullet,\bullet)_1\rangle$ to $|(\bullet,\bullet)_\tau\rangle$, which means that we couldn't perform arbitrary qubit rotations. This is naturally not convenient for quantum computation. The idea to remedy this problem is to use the three Fibonacci states (14.18). $|0\rangle$ and $|1\rangle$ are the two orthogonal qubit states and $|N\rangle$ is a "noncomputational" state. When we start braiding the particles we have to be sure that we don't induce any transition from the computational states to the noncomputational state. Such transitions are called **leakage errors**.

Having now introduce all the important properties of the Fibonacci model, we can basically understand how we can perform a quantum computation, i.e. initialization of a qubit, unitary evolution and measurement. In Fig.(14.8), you can see the basic scheme of our quantum computation model (please note that on all the following pictures every 0 should be replaced by 1 and every 1 should be replaced by τ). One possibility to initialize the qubit is to create two particle-antiparticle pairs from the vacuum. Since they are created from the vacuum their total q-spin is 1. To construct the qubit we take three of the four particles and ignore the fourth one. In Fig.(14.8) the qubit is initialized in state $|0\rangle$. As I

Figure 14.8: First we create particle-antiparticle pairs from the vacuum, then we braid them and finally they are fused together. This figure is taken from [14].

already mentioned, in order to have a degenerate quantum Hall state the quasiparticles have to be kept far apart from each other. When the initialization of the qubit has been done, we can perform a unitary evolution of the system by braiding the quasiparticles. Let me remind you that braiding two quasiparticles induce a unitary operation on the set of degenerate states. Finally, to measure the state of the qubit after the unitary evolution, we can try to annihilate the bottom particle anti-particle pair from Fig.(14.8) and look if they annihilate. If this is the case this means that their overall q-spin is 1, however if this isn't the case their overall q-spin is τ. Thus in the first case the qubit is in state $|0\rangle$ and in the second case the qubit is in state $|1\rangle$.
In the following two subsections, I will focus on the unitary evolution and present how we can build single-qubit rotations by braiding Fibonacci anyons and then how we can build two-qubit gates. Doing this we will thus prove that the Fibonacci model is universal for quantum computation since:

Any multiple qubit logic gate may be composed from the CNOT-gate and single-qubit rotations [17].

Single-qubit rotations

The aim of this section is to show how we can construct single-qubit rotations up to any desired accuracy by braiding Fibonacci anyons. We will follow [14] and [18]. The first thing to note is that σ_1, σ_1^{-1}, σ_2, σ_2^{-1} generate all possible braids, thus $\rho(\sigma_1)$, $\rho(\sigma_2)$ and their inverse generate the corresponding unitary operations. Fortunately $\rho(\sigma_1)$ and $\rho(\sigma_2)$ are block-diagonal matrices, which means that they don't induce transitions from the computational states into the non-computational state. Now before going further, I have to mention the following

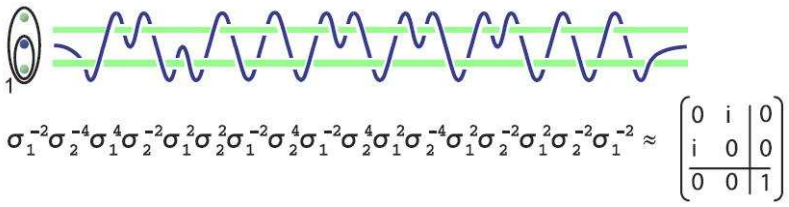


Figure 14.9: Approximation of a iX-gate to a distance of about 10^{-3} (X is the NOT-gate). This figure is taken from [18].

important "theorem":

There exists a braid that corresponds to a unitary operation arbitrarily close to any desired single-qubit rotations in the Fibonacci theory [5].

This isn't a trivial statement, for example the set of unitary operations induced by braiding anyons in the $\nu = \frac{5}{2}$ quantum Hall state isn't rich enough for all operations needed for a universal computation. In what follows we will always consider that two particles of the qubit are maintained fixed and the other one moves around them. One method to find which braid corresponds to a given unitary operation within a given accuracy (the accuracy or the distance ϵ between two matrices U and V is defined as $\epsilon = ||U - V||$, where $||O||$ is the square-root of the biggest eigenvalue of the matrix O^+O) is brute-force search (see Fig.(14.9)). That is by investigating all possible braids up to a certain length (e.g. 46 in reference [18]) and by determining which of these approximates the gate up to the desired accuracy. We can implement this procedure on a classical computer for example. Note that the length of the braid grows typically as $|\log \epsilon|$, ϵ being the allowed error. Although this method works very well for short braids, it becomes very unpractical when we require high precision. Actually the number of different braids grows exponentially in the length of the braid. Fortunately Solovay and Kitaev found a very powerful recursive algorithm to overcome this problem (for the details of this algorithm see [19]). **It is an iterative algorithm which allows one to put together many short braids to efficiently construct a long braid arbitrarily close to any desired target unitary operation** (quotation from [5]). The length of the obtained new braid scales as $|\log \epsilon|^c$, where ϵ is the required accuracy and $c \approx 4$ (for more details see [19]). In Fig.(14.10) you can see the first step of the Solovay-Kitaev algorithm [14]. This first step of the algorithm leads to an accuracy of about 10^{-4}, i.e. ten times better than our previous brute-search method (Fig.(14.9)).

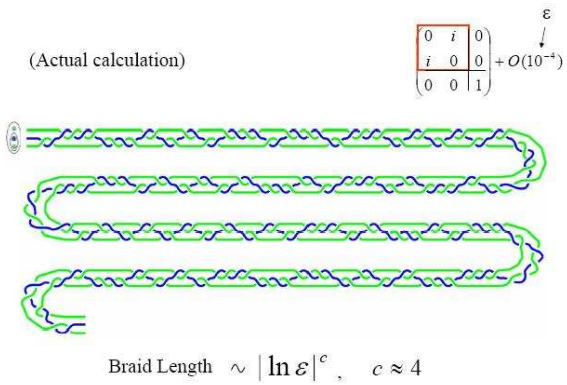


Figure 14.10: iX-gate approximation with Solovay-Kitaev algorithm (X is the NOT-gate). This figure is taken from [16].

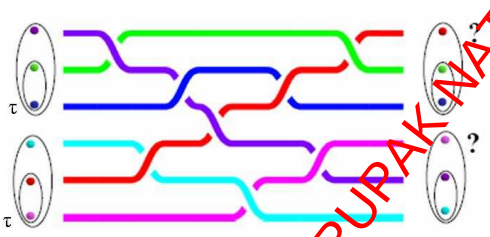


Figure 14.11: The effect of braiding particles from one qubit around particles from another qubit can lead to leakage errors. This figure is taken from [16].

14.4.3 Two-qubit gates

Knowing from the previous subsection that building single-qubit rotations is a manageable task, now we have to show that it is also possible to construct two-qubit gates in a simple way and thus to achieve universal quantum computation. Let's now consider two qubits, each formed by three Fibonacci anyons. Two-qubit gates are formed by braiding the anyons of one qubit with the anyons of the other qubit (Fig.(14.11)). Here we have to take two major problems into account, the first one is that brute-force search is unfeasible in this case because the space of special unitary operations for six particles has 87 free parameters (for more details see [14]) which is much more than the three free parameters in the case of a single qubit (three Fibonacci anyons). The other problem is that we will exchange particles from one qubit with particles from another qubit and this can lead in general to leakage errors. However we will see that we can solve these problems by reducing the procedure of finding two-qubit gates to the problem of finding braids of three particles (we have described this case in the last section). Here I will also follow [14] and [18]. Let's consider the situation depicted in Fig.(14.12).

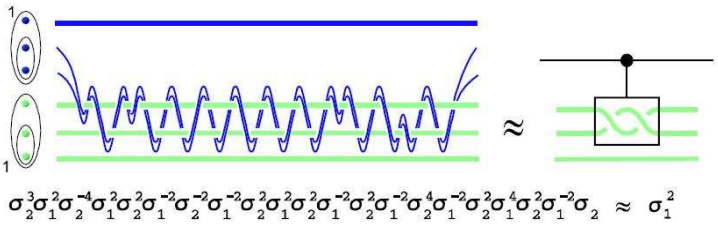


Figure 14.12: The effect of braiding the two anyons from the control qubit around the two fixed anyons from the target-qubit has the same effect has braiding two times the anyons from the target-qubit. This figure is taken from [18].

The upper qubit is the control-qubit and the lower one is the target-qubit. The first useful process that we can easily build is to move a pair of particles from the conrtol-qubit around the particles of the target-qubit. The two particles of the control-qubit, which are used to braid, can fuse either to the trivial particle 1 or to the nontrivial τ. We can thus consider this particles pair as a single anyon which is either in state 1 or τ. If it is in state 1 then moving it has no effect (remember that the 1 particle is similar to the vacuum). However, if it is in the sate τ, braiding it produces a nontrivial unitary operation. In Fig.(14.12), you can see that the braiding of the two anyons from the control-qubit approximates (up to a distance of $\epsilon \approx 2.3 \times 10^{-3}$) an operation where two particles from the target-qubit are braid two times (you can easily compute the corresponding matrix multiplication). Remembering the notation from Eq.(14.18), we see that if the control-qubit is in state $|0\rangle$, then this braiding pattern has no effect and if it is in state $|1\rangle$, then this braiding pattern is equivalent to the unitary operation σ_2^2 acting on the target-qubit (be aware that the σ_1^2 operation acting on the three particles of the braid corresponds to a σ_2^2 operation acting on the target-qubit). It is now very clear that this is a controlled-σ_2^2 gate and this actually corresponds to a rotation of the target-qubit through an angle of $\frac{6\pi}{5}$. In the limit where $\epsilon \to 0$, i.e. in the limit where we perfectly approximate the unitary operation, there aren't any leakage errors. Although this isn't as CNOT-gate, the set composed by this controlled two-qubit gate and all the single-qubit rotations is already universal for quantum computation. However, as I told you before, I want to show how we can build a CNOT-gate. It is very useful to be able to directly implement a CNOT-gate in quantum computation. Unfortunately with the above procedure, it is only possible to construct controlled rotation of the target qubit through an angle $\frac{6m\pi}{5}$ where m is an even number (for more details see [14]). To construct a NOT-gate (controlled π rotation up to single-qubit rotations, see [17]), we introduce a new procedure called injection. As you can understand from its

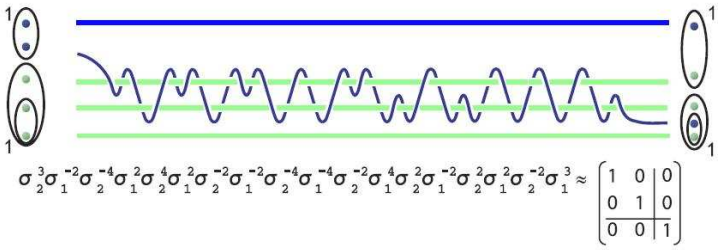


Figure 14.13: Injection of an anyon from the control-qubit into target-qubit. This figure is taken from [18].

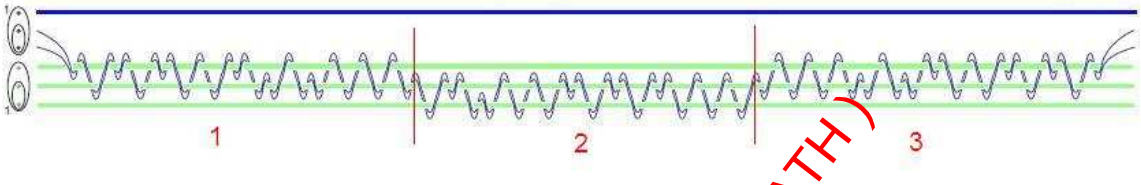


Figure 14.14: Approximation of a CNOT-gate. First we inject a particle into the target-qubit, then we apply an iX-gate on the injected target-qubit and finally we extract the control pair from the target-qubit. This figure is taken from [14] (I modified it by adding the vertical lines and the numbers 1,2 and 3).

name, it corresponds to an injection of a particle from the control-qubit into the target-qubit. At this stage we will consider braids where the particle from the control-qubit doesn't reach its initial position after it has been braid (this wasn't the case before, in Fig.(14.12) the initial position of the particles is the same as the final position). This procedure is depicted in Fig.(14.13) where we see one anyon from the control-qubit being braid around two static particles from the target-qubit and where the final position of the anyons isn't the same as the initial position. This operation approximates the identity operation up to a distance of about $\epsilon \approx 1.5 \times 10^{-3}$. In the limit $\epsilon \to 0$, the effect of this braiding pattern is to interchange one particle from the control-qubit with one particle from the target-qubit without affecting both qubits by any unitary operation, i.e. no q-spin of the system is changed by this procedure. After the injection we can use the braiding pattern from Fig.(14.9) to implement an iX-gate which acts on the target-qubit. The braid depicted in Fig.(14.14) corresponds to a controlled-iX gate up to an accuracy of about $\epsilon \approx 10^{-3}$. As you can see, the braid is divided in three parts. The left part is the injection of the control pair in the target-qubit, the second part corresponds to the application of the iX-gate (see Fig.(14.9)) on the target qubit and the last part of this braid corresponds to the extraction of the injected particle. This last operation is simply the inverse

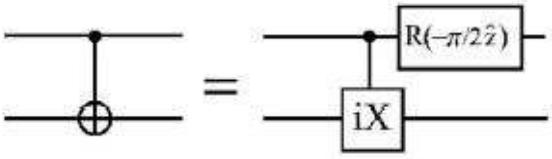


Figure 14.15: A CNOT-gate is equivalent to a controlled-iX gate followed by a single-qubit rotation applied on the control-qubit. This figure is taken from [14].

of the injection. As in the previous example, if the control pair is in state $|0\rangle$, then this braiding pattern corresponds to the identity, however if it is in state $|1\rangle$, then an iX-gate is applied to the target-qubit. Finally, I have to mention that a CNOT-gate is actually a controlled-iX gate followed by a single-qubit rotation applied on the control-qubit, as shown in Fig.(14.15). We thus conclude from this section that constructing a CNOT-gate is as manageable as constructing single-qubit rotations. To finish I want to emphasize that it is always possible to apply the Solovay-Kitaev algorithm to improve the accuracy of the braid from Fig.(14.14).

14.5 Conclusion

In this chapter we described a model for a topological quantum computation which is protected against decoherence. The basic idea is to encode qubits non-locally using multiple non-abelian anyons distant from each other in a quantum Hall state. We have focused on Fibonacci anyons which are universal for quantum computation and which we hope to detect in the $\nu = \frac{12}{5}$ quantum Hall state. We have seen how it is possible to perform any unitary operations by exchanging anyons and that these operations are non-local (at least the non-abelian part). Since the encoding and the manipulation of the quantum information in this model are non-local, this computation is fault-tolerant and is one of the most promising way in order to build a robust quantum computer. To finish, I want to mention that some interference experiments have been proposed in order to detect the non-abelian statistics of quasiparticles in the the $\nu = \frac{5}{2}$ quantum Hall state. It has also been proposed how we could use such interference experiments to implement a NOT-gate using non-abelian anyons, if you are interested in this you can look at [5] or [20].

Bibliography

[1] B. Doucot, L. B. Ioffe, and J. Vidal, *Discrete non-Abelian gauge theories in Josephson-junction arrays and quantum computation*, Phys. Rev. B **69**, 214501 (2004).

[2] M. A. Levin and X.-G. Wen, *String-net condensation: A physical mechanism for topological phases*, Phys. Rev. B **71**, 045110 (2005).

[3] P. Fendley and E. Fradkin, *Realizing non-Abelian statistics*, Phys. Rev. B **72**, 024412 (2005).

[4] J. S. Xia, W. Pan, N. S. Adams, H. L. Sullivan, D. C. Tsui, L. N. Pfeiffer, K. W. Baldwin, and K. W. West, *Electron Correlation in the Second Landau Level: A Competition Between Many Nearly Degenerate Quantum Phases*, Phys. Rev. Lett. **93**, 17 (2004).

[5] S. Das Sarma, M. Freedman, C. Nayak, S. H. Simon, and A. Stern, *Non-Abelian Anyons and Topological Quantum Computation* (2007), arxiv:0707.1889v1[cond-mat.str-el].

[6] R. Prange and S. M. Girvin, *The Quantum Hall effect* (Springer-Verlag, New-York, 1990).

[7] D. Arovas, J. R. Schrieffer, and F. Wilczek, *Fractional Statistics and the Quantum Hall Effect*, Phys. Rev. Lett. **53**, 722 (1984).

[8] J. Preskill, *Topological Quantum Computation*, Lecture notes (2000), chapter 9 of "Quantum Computation" available online: www.theory.caltech.edu/people/preskill/ph229.

[9] A. Kitaev, *Fault-tolerant quantum computation by anyons* (1997), arxiv:quant-ph/9707021.

[10] G. Moore and N. Read, *Nonabelions in the fractional quantum Hall effect*, Nucl. Phys. B **360**, 362 (1991).

[11] C. Nayak and F. Wilczek, *2n-quasihole states realize 2^{n-1}-dimensional spinor braiding statistics in paired quantum Hall states*, Nucl. Phys. B **479**, 529 (1996).

[12] A. Khare, *Fractional Statistics and Quantum Theory* (World Scientific, Singapore, 1997).

[13] S. H. Simon, *Effect of Landau Level Mixing on Braiding Statistics*, Phys. Rev. Lett. **100**, 116803 (2008).

[14] L. Hormozi, G. Zikos, N. E. Bonesteel, and S. H. Simon, *Topological quantum compiling*, Pys. Rev. B **75**, 165310 (2007).

[15] N. Read and E. Rezayi, *Beyond paired quantum hall states: Parafermions and incompressible states in the first excited Landau level*, Phys. Rev. B **59**, 8084 (1999).

[16] N. E. Bonesteel, *Braid Topologies for Quantum Computation* (2004), talk given at KITP, Santa Barbara, available online: http://online.kitp.ucsb.edu/online/qubit03/bonesteel.

[17] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge U.K., 2000).

[18] N. E. Bonesteel, L. Hormozi, and G. Zikos, *Braid Topologies for Quantum Computation*, Phys. Rev. Lett. **95**, 140503 (2005).

[19] C. M. Dawson and M. A. Nielsen, *The solovay-kitaev algorithm* (2005), arxiv:quant-ph/0505030v2.

[20] P. Bonderson, A. Kitaev, and K. Shtengel, *Detecting Non-Abelian Statistics in the $\nu = \frac{5}{2}$ Quantum Hall State*, Phys. Rev. Lett. **96**, 016803 (2006).