QUANTUM COMPUTATION*

Dorit Aharonov
Departments of Physics and Computer Science,
The Hebrew University, Jerusalem, Israel

December 15, 1998

Abstract

In the last few years, theoretical study of quantum systems serving as computational devices has achieved tremendous progress. We now have strong theoretical evidence that quantum computers, if built, might be used as a dramatically powerful computational tool, capable of performing tasks which seem intractable for classical computers. This review is about to tell the story of theoretical quantum computation. I left out the developing topic of experimental realizations of the model, and neglected other closely related topics which are quantum information and quantum communication. As a result of narrowing the scope of this paper, I hope it has gained the benefit of being an almost self contained introduction to the exciting field of quantum computation.

The review begins with background on theoretical computer science, Turing machines and Boolean circuits. In light of these models, I define quantum computers, and discuss the issue of universal quantum gates. Quantum algorithms, including Shor's factorization algorithm and Grover's algorithm for searching databases, are explained. I will devote much attention to understanding what the origins of the quantum computational power are, and what the limits of this power are. Finally, I describe the recent theoretical results which show that quantum computers maintain their complexity power even in the presence of noise, inaccuracies and finite precision. This question cannot be separated from that of quantum complexity, because any realistic model will inevitably be subject to such inaccuracies. I tried to put all results in their context, asking what the implications to other issues in computer science and physics are. In the end of this review I make these connections explicit, discussing the possible implications of quantum computation on fundamental physical questions, such as the transition from quantum to classical physics.

1 Overview

Since ancient times, humanity has been seeking tools to help us perform tasks which involve calculations. Such are computing the area of a land, computing the stresses on

*To appear in *Annual Reviews of Computational Physics* VI, Edited by Dietrich Stauffer, World Scientific, 1998

rods in bridges, or finding the shortest route from one place to another. A common feature of all these tasks is their structure:

Input ——> Computation ——> Output

The computation part of the process is inevitably performed by a dynamical physical system, evolving in time. In this sense, the question of what can be computed, is intermingled with the physical question of which systems can be physically realized. If one wants to perform a certain computation task, one should seek the appropriate physical system, such that the evolution in time of the system corresponds to the desired computation process. If such a system is initialized according to the input, its final state will correspond to the desired output.

A very nice such example was invented by Gaudí, a great Spanish architect, who lived around the turn of the century. His design of the holy family church, (*la sagrada familia*) in Barcelona is a masterpiece of art, and is still in the process of building, after almost a hundred years. The church resembles a sand palace, with a tremendous complexity of delicate thin but tall towers and arcs. Since the plan of the church was so complicated, towers and arcs emerging from unexpected places, leaning on other arcs and towers, it is practically impossible to solve the set of equations which corresponds to the requirement of equilibrium in this complex. Instead of solving this impossible task, Gaudí thought of the following ingenious idea: For each arc he desired in his complex, he took a rope, of length proportional to the length of the arc. He tied the edges of one rope to the middle of some other rope, or where the arcs were supposed to lean on each other. Then he just tied the edges of the ropes corresponding to the lowest arcs, to the ceiling. All the computation was instantaneously done by gravity! The set of arcs arranged itself such that the whole complex is in equilibrium, but upside down. Everything was there, the angles between the different arcs, the radii of the arcs. Putting a mirror under the whole thing, he could simply see the design of the whole church! [102].

Many examples of analog computers exist, which were invented to solve one complicated task. Such are the differential analyzer invented by Lord Kelvin in 1870[120], which uses friction, wheels, and pressure to draw the solution of an input differential equations. The spaghetti sort is another example, and there are many more[194]. Are these systems "computers"? We do not want to construct and build a completely different machine for each task that we have to compute. We would rather have a general purpose machine, which is "universal". A mathematical model for a "universal" computer was defined long before the invention of computers and is called the Turing machine[188]. Let me describe this model briefly. A Turing machine consists of an infinite tape, a head that reads and writes on the tape, a machine with finitely many possible states, and a transition function δ. Given what the head reads at time t, and the machine's state at time t, δ determines what the head will write, to which direction it will move and what will be the new machine's state at time $t+1$. The Turing machine model seems to capture the entire concept of computability, according to the following thesis[62]:

Church Turing Thesis: A Turing machine can compute any function computable by a reasonable physical device

What does "reasonable physical device" mean? This thesis is a physical statement, and as such it cannot be proven. But one knows a physically unreasonable device when one sees it. Up till now there are no candidates for counterexamples to this thesis (but see Ref. [103]). All physical systems, (including quantum systems), seem to have a simulation by a Turing Machine.

It is an astonishing fact that there are families of functions which cannot be computed. In fact, most of the functions cannot be computed. There are trivial reasons for this: There are more functions than there are ways to compute them. The reason for this is that the set of Turing machines is countable, where as the set of *families* of functions is not. In spite of the simplicity of this argument (which can be formalized using the *diagonal argument*) this observation came as a complete surprise in the 1930's when it was first discovered. The subject of computability of functions is a cornerstone in computational complexity. However, in the theory of computation, we are interested not only in the question of which functions can be computed, but mainly in the *cost* of computing these functions. The cost, or *computational complexity*, is measured naturally by the physical resources invested in order to solve the problem, such as time, space, energy, etc. A fundamental question in computation complexity is how the cost function behaves as a function of the input size, n, and in particular whether it is exponential or polynomial in n. In computer science problems which can only be solved in exponential cost are regarded intractable, and any of the readers who has ever tried to perform an exponentially slow simulation will appreciate this characterization. The class of tractable problems constitutes of those problems which have polynomial solutions.

It is worthwhile to reconsider what it means to *solve* a problem. One of the most important conceptual breakthroughs in modern mathematics was the understanding[164] that sometimes it is advantageous to relax the requirements that a solution be always correct, and allow some (negligible) probability for an error. This gave rise to much more rapid solutions to different problems, which make use of random coin flips, such as the Miller-Rabin randomized algorithm to test whether an integer is prime or not[73]. Here is a simple example of the advantage of probabilistic algorithms:

> we have access to a database of N bits, and we are told that they are either all equal, ("constant") or half are 0 and half are 1 ("balanced"). We are asked to distinguish between the two cases.

A deterministic algorithm will have to observe $N/2+1$ bits in order to always give a correct answer. To solve this problem probabilistically, toss a random i between 1 to N, observe the i'th bit, and repeat this experiment k times. If two different bits are found, the answer is "balanced", and if all bits are equal, the answer is "constant". Of course, there is a chance that we are wrong when declaring "constant", but this chance can be made arbitrarily small. The probability for an error equals the chance of tossing a fair coin k times and getting always 0, and it decreases exponentially with k. For example, in order for the error probability to be less than 10^{-10}, $k = 100$ suffices. In general, for any desired confidence, a constant k will do. This is a very helpful shortcut if N is very large. Hence, if we allow negligible probability of error, we can do much better!

The class of tractable problems is now considered as those problems solvable with a negligible probability for error in polynomial time. These solutions will be computed by a probabilistic Turing machine, which is defined exactly as a deterministic Turing machine, except that the transition function can change the configuration in one of several possible ways, randomly. The modern Church thesis refines the Church thesis and asserts that the probabilistic Turing machine captures the entire concept of computational complexity:

> **The modern Church thesis**: A probabilistic Turing machine can simulate any reasonable physical device in polynomial cost.

It is worthwhile considering a few models which might seem to contradict this thesis at first sight. One such model is the DNA computer which enables a solution of NP-complete problems (these are hard problems to be defined later) in polynomial time[4, 140]. However, the cost of the solution is exponential because the number of molecules in the system grows exponentially with the size of the computation. Vergis et al[194] suggested a machine which seems to be able to solve instantaneously an NP-complete problem using a construction of rods and balls, which is designed such that the structure moves according to the solution to the problem. A careful consideration[178] reveals that though we tend to think of rigid rods as transferring the motion instantaneously, there will be a time delay in the rods, which will accumulate and cause an exponential overall delay. Shamir[170] showed how to factorize an integer in polynomial time *and* space, but using another physical resource exponentially, namely precision. In fact, J. Simon showed that extremely hard problems (The class of problems called Polynomial space, which are harder than NP problems) can be solved with polynomial cost in time and space[176], but with exponential precision. Hence all these suggestions for computational models do not provide counterexamples for the modern Church thesis, since they require exponential physical resources. However, note that all the suggestions mentioned above rely on classical physics.

In the early 80's Benioff[27, 28] and Feynman[94] started to discuss the question of whether computation can be done in the scale of quantum physics. In classical computers, the elementary information unit is a *bit*, i.e. a value which is either 0 or 1. The quantum analog of a bit would be a two state particle, called a quantum bit or a **qubit**. A two state quantum system is described by a unit vector in the Hilbert space C^2, where C are the complex numbers. One of the two states will be denoted by $|0\rangle$, and corresponds to the vector $(1,0)$. The other state, which is orthogonal to the first one, will be denoted by $|1\rangle = (0,1)$. These two states constitute an orthogonal basis to the Hilbert space. To build a computer, we need to compose a large number of these two state particles. When n such qubits are composed to one system, their Hilbert space is the tensor product of

n spaces: $C^2 \otimes C^2 \otimes \cdots \otimes C^2$. To understand this space better, it is best to think of it as the space spanned by its basis. As the natural basis for this space, we take the basis consisting of 2^n vectors, which is sometimes called the computational basis:

$$
\begin{aligned}
&|0\rangle \otimes |0\rangle \otimes \cdots \otimes |0\rangle \\
&|0\rangle \otimes |0\rangle \otimes \cdots \otimes |1\rangle \\
&\quad \vdots \\
&|1\rangle \otimes |1\rangle \otimes \cdots \otimes |1\rangle.
\end{aligned}
\tag{1}
$$

Naturally classical strings of bits will correspond to quantum states:

$$
i_1 i_2 ... i_n \longleftrightarrow |i_1\rangle \otimes |i_2\rangle \otimes \cdots \otimes |i_n\rangle \equiv |i_1 .... i_n\rangle \tag{2}
$$

How can one perform computation using qubits? Suppose, e.g., that we want to compute the function $f : i_1 i_2 ... i_n \longmapsto f(i_1, ...i_n)$, from n bits to n bits. We would like the system to evolve according to the time evolution operator U:

$$
|i_1 i_2 ... i_n\rangle \longmapsto U|i_1 i_2 ... i_n\rangle = |f(i_1, ...i_n)\rangle. \tag{3}
$$

We therefore have to find the Hamiltonian $\mathcal{H}$ which generates this evolution according to Schrödinger's equation: $i\hbar \frac{d}{dt}|\Psi(t)\rangle = \mathcal{H}|\Psi(t)\rangle$. This means that we have to solve for $\mathcal{H}$ given the desired U:

$$
|\Psi_f\rangle = \exp\left(-\frac{i}{\hbar}\int \mathcal{H} dt\right)|\Psi_0\rangle = U|\Psi_0\rangle \tag{4}
$$

A solution for $\mathcal{H}$ always exists, as long as the linear operator U is unitary. It is important to pay attention to the unitarity restriction. Note that the quantum analog of a classical operation will be unitary only if f is one-to-one, or reversible. Hence, reversible classical function can be implemented by a physical Hamiltonian. Researchers investigated the question of reversible classical functions in connection with completely different problems, e.g. the problem of whether computation can be done without generating heat (which is inevitable in irreversible operations) and as a solution to the "maxwell demon" paradox[136, 30, 31, 121]. It turns out that any classical function can be represented as a reversible function[137, 29] on a few more bits, and the computation of f can be made reversible without losing much in efficiency. Moreover, if f can be computed classically by polynomially many elementary reversible steps, the corresponding U is also decomposable into a sequence of polynomially many elementary unitary operations. We see that quantum systems can imitate all computations which can be done by classical systems, and do not lose much in efficiency.

Quantum computation is interesting not because it can imitate classical computation, but because it can probably do much more. In a seminal paper[93], Feynman pointed out the fact that quantum systems of n particles seem exponentially hard to simulate by classical devices. In other words, quantum systems do not seem to obey the modern Church thesis, i.e. they do not seem to be polynomially equivalent to classical systems! If quantum systems are hard to simulate, then quantum systems, harnessed as computational devices, might be dramatically more powerful than other computational devices.

Where can the "quantumness" of the particles be used? When I described how quantum systems imitate classical computation, the quantum particles were either in the state $|0\rangle$ or $|1\rangle$. However, quantum theory asserts that a quantum system, like Schrödinger's cat, need not be in one of the basis states $|0\rangle$ and $|1\rangle$, but can also be in a *linear superposition* of those. Such a superposition can be written as:

$$c_0|0\rangle + c_1|1\rangle \tag{5}$$

where c_0, c_1 are complex numbers and $|c_0|^2+|c_1|^2 = 1$. The wave function, or superposition, of n such quantum bits, can be in a superposition of all of the 2^n possible basis states! Consider for example the following state of 3 particles, known as the GHZ state[108]:

$$\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle) \tag{6}$$

What is the superposition describing the first qubit? The answer is that there is no such superposition. Each one of the 3 qubits does not have a state of its own; the state of the system is not a tensor product of the states of each particle, but is some superposition which describes quantum correlations between these particles. Such particles are said to be quantumly *entangled*. The Einstein Podolski Rosen paradox[89], and Bell inequalities[25, 26, 68, 108], correspond to this puzzling quantum feature by which a quantum particle does not have a state of its own. Because of the entanglement or quantum correlations between the n quantum particles, the state of the system cannot be specified by simply describing the state of each of the n particles. Instead, the state of n quantum bits is a complicated superposition of all 2^n basis states, so 2^n complex coefficients are needed in order to describe it. This exponentiality of the Hilbert space is a crucial ingredient in quantum computation. To gain more understanding of the advantages of the exponentiality of the space, consider the following superposition of n quantum bits.

$$\frac{1}{\sqrt{2^n}} \sum_{i_1,i_2,...,i_n=0}^{1} |i_1, i_2, ..., i_n\rangle \tag{7}$$

This is a uniform superposition of all possible basis states of n qubits. If we now apply the unitary operation which computes f, from equation 3, to this state, we will get, simply from linearity of quantum mechanics:

$$\frac{1}{\sqrt{2^n}} \sum_{i_1,i_2,...,i_n=0}^{1} |i_1, i_2, ..., i_n\rangle \longmapsto \frac{1}{\sqrt{2^n}} \sum_{i_1,i_2,...,i_n=0}^{1} |f(i_1, i_2, ..., i_n)\rangle. \tag{8}$$

Applying U once computes f simultaneously on all the 2^n possible inputs i, which is an enormous power of parallelism!

It is tempting to think that exponential parallelism immediately implies exponential computational power, but this is not the case. In fact, classical computations can be viewed as having exponential parallelism as well– we will devote much attention to this later on. The problem lies in the question of how to extract the exponential information out of the system. In quantum computation, in order to extract quantum information one has to *observe* the system. The measurement process causes the famous *collapse of the wave function*. In a nutshell, this means that after the measurement the state is projected to only one of the exponentially many possible states, so that the exponential amount of information which has been computed is completely lost! In order to gain advantage of exponential parallelism, one needs to combine it with another quantum feature, known as interference. Interference allows the exponentially many computations done in parallel to cancel each other, just like destructive interference of waves or light. The goal is to arrange the cancelation such that only the computations which we are interested in remain, and all the rest cancel out. The combination of exponential parallelism and interference is what makes quantum computation powerful, and plays an important role in quantum algorithms.

A quantum algorithm is a sequence of elementary unitary steps, which manipulate the initial quantum state $|i\rangle$ (for an input i) such that a measurement of the final state of the system yields the correct output. The first quantum algorithm which combines interference and exponentiality to solve a problem faster than classical computers, was discovered by Deutsch and Jozsa[80]. This algorithm addresses the problem we have encountered before in connection with probabilistic algorithms: Distinguish between "constant" and "balanced" databases. The quantum algorithm solves this problem *exactly*, in polynomial cost. As we have seen, classical computers cannot do this, and must release the restriction of exactness. Deutsch and Jozsa made use of the most powerful tool in quantum algorithms, the *Fourier transform*, which indeed manifests interference and exponentiality. Simon's algorithm[177] uses similar techniques, and was the seed for the most important quantum algorithm known today: Shor's algorithm.

Shor's algorithm (1994) is a polynomial quantum algorithm for factoring integers, and for finding the logarithm over a finite field[172]. For both problems, the best known classical algorithms are exponential. However, there is no proof that classical efficient algorithms do not exist. Shor's result is regarded as extremely important both theoretically and practically, mainly due to the fact that the assumption that factorization is hard lies in the heart of the RSA cryptographic system [166, 73]. A cryptosystem is supposed to be a secure way to transform information such that an eavesdropper will not be able to learn in reasonable time significant information about the message sent. The RSA cryptosystem is used very heavily: The CIA uses it, the security embedded into Netscape and the Explorer Web browsers is based on RSA, banks use RSA for internal security as well as securing external connections. However, RSA can be cracked by any one who has an efficient algorithm for factoring. It is therefore understandable why the publication of the factorization algorithm caused a rush of excitement all over the world.

It is important that the quantum computation power does not rely on unreasonable precision but a polynomial amount of precision in the computational elements is enough[38]. This means that the new model requires physically reasonable resources, in terms of time, space, and precision, but yet it is (possibly) exponentially stronger than the ordinary model of probabilistic Turing machine. As such, it is the only model which really threatens the modern Church thesis.

There are a few major developing directions of research in the area of quantum computation. In 1995 Grover[110] discovered an algorithm which searches an unsorted database of N items and finds a specific item in $\sqrt{N}$ time steps. This result is surprising, because intuitively, one cannot search the database without going through all the items. Grover's solution is quadratically better than any possible classical algorithms, and was followed by numerous extensions and applications[44, 111, 112, 87, 47, 48], all achieving polynomial advantage over classical algorithms. A promising new branch in quantum complexity theory is the study of a class of problems which is the quantum analog of the complexity class NP[126]. Another interesting direction in quantum computation is concerned with quantum computers simulating efficiently other physical systems such as many body Fermi systems[203, 1, 197, 42]. This direction pursues the original suggestion by Feynman[93], who noticed that quantum systems are hard to simulate by classical devices. An important direction of investigation is the search for a different, perhaps stronger, quantum computation model. For example, consider the introduction of slight non-linearities into quantum mechanics. This is completely hypothetical, as all experiments verify the linearity of quantum mechanics. However, such slight non linearities would imply extremely strong quantum algorithms[2]. A very interesting quantum computation model which is based on anyons, and uses non-local features of quantum mechanics, was suggested by Kitaev[125]. A possibly much stronger model, based on quantum field theory, was sketched recently by Freedman, but it has not been rigorously defined yet[97]. One other direction is oracle results in quantum complexity. This direction compares quantum complexity power and classical complexity power when the two models are allowed to have access to an oracle, i.e. a black box which can compute a certain (possibly difficult) function in one step [38, 36, 40, 41]. In fact, the result of Bernstein and Vazirani[38] from 1993 demonstrating a superpolynomial gap between quantum and classical computational comlexity with an access to a certain oracle initialized the sequence of results leading to the Shor's algorithm. An important recent result[23] in quantum complexity shows that quantum computers have no more than polynomial advantage in terms of number of accesses to the inputs. As of now, we are very far from understanding the computational power of quantum systems. In particular, it is not known whether quantum systems can efficiently solve NP complete problems or not.

Quantum information theory, a subject which is intermingled with quantum computation, provides a bunch of quantum magic tricks, which might be used to construct more powerful quantum algorithms. Probably the first "quantum pearl" that one encounters in quantum mechanics is the Einstein Podolsky Rosen paradox, which, as is best explained by Bell's inequalities, establishes the existence of correlations between quantum particles, which are stronger than any classical model can explain. Another "quantum pearl" which builds on quantum entanglement, is teleportation[34]. This is an amazing quantum recipe which enables two parties (Alice and Bob) which are far apart, to transfer an unknown quantum state of a particle in Alice's hands onto a particle in Bob's hand, without sending the actual particle. This can be done if Alice and Bob share a pair of particles which interacted in the past and therefore are quantumly entangled. Such quantum effects already serve as ingredients in different computation and communication tasks. Entanglement can be used, for example, in order to gain advantage in communication. If two parties, Alice and Bob, want to communicate, they can save bits of communication if they share entangled pairs of qubits[69, 51, 70, 14]. Teleportation can be viewed as a quantum computation[49], and beautiful connections were drawn[35] between teleportation and quantum algorithms which are used to correct quantum noise. All these are uses of quantum effects in quantum computation. However, I believe that the full potential of quantum mechanics in the context of complexity and algorithmic problems is yet to be revealed.

Despite the impressive progress in quantum computation, a menacing question still remained. Quantum information is extremely fragile, due to inevitable interactions between the system and its environment. These interactions cause the system to lose part of its quantum nature, a process called *decoherence*[184, 205]. In addition, quantum elementary operations (called *gates*) will inevitably suffer from inaccuracies. Will physical realizations of the model of quantum computation still be as powerful as the ideal model? In classical computation, it was already shown by von-Neumann[153] how to compute when the elements of the computation are faulty, using redundant information. Indeed, nowadays error corrections are seldom used in computers because of extremely high reliability of the elements, but quantum elements are much more fragile, and it is almost certain that quantum error corrections will be necessary in future quantum computers. It was shown that if the errors are not corrected during quantum computation, they soon accumulate and ruin the entire computation[57, 58, 17, 149]. Hence, a method to correct the effect of quantum noise is necessary. Physicists were pessimistic about the question of whether such a correction method exists[135, 189]. The reason is that quantum information in general cannot be cloned[83, 200, 20], and so the information cannot be simply protected by redundancy, as is done classically. Another problem is that in contrast to the discreteness of digital computers, a quantum system can be in a superposition of eigenstates with continuous coefficients. Since the range of allowed coefficients is continuous, it seems impossible to distinguish between bona fide information and information which has been contaminated.

As opposed to the physical intuition, it turns out that clever techniques enable quantum information to be protected. The conceptual breakthrough in quantum error corrections was the understanding that quantum errors, which are continuous, can be viewed as a discrete process in which one out of four quantum operations occurs. Moreover, these errors can be viewed as classical errors, called bit flips, and quantum errors, called phase flips.

Bit flip errors can be corrected using classical error correction techniques. Fortunately, phase flips transform to bit flips, using the familiar Fourier transform. This understanding allowed using classical error correction codes techniques in the quantum setting. Shor was the first to present a scheme that reduces the affect of noise and inaccuracies, building on the discretization of errors[173]. As in classical error correcting codes, quantum states of k qubits are *encoded* on states of more qubits. Spreading the state of a few qubits on more qubits, allows correction of the information, if part of it has been contaminated. These ideas were extended [53, 180] to show that a quantum state of k qubits can be encoded on n qubits, such that if the n qubits are sent through a noisy channel, the original state of the k qubits can be recovered. k/n tends asymptotically to a constant *transmission rate* which is non zero. This is analogous to Shannon's result from noisy classical communication[171]. Many different examples of quantum error correcting codes followed[181, 134, 59, 131, 165, 138], and a group theoretical framework for most quantum codes was established[55, 54, 106].

Resilient quantum computation is more complicated than simply protecting quantum information which is sent through a noisy quantum channel. Naturally, to protect the information we would compute on encoded states. There are two problems with noisy computation on encoded states. The first is that the error correction is done with faulty gates, which cause errors themselves[19]. We should be careful that the error correction does not cause more harm than it helps. The second problem is that when computing on encoded states, qubits interact with each other through the gates, and this way errors can *propagate* through the gates, from one qubit to another. The error can spread in this way to the entire set of qubits very quickly. In order to deal with these problems, the idea is to perform computation and error correction in a *distributed manner*, such that each qubit can effect only a small number of other qubits. Kitaev[124] showed how to perform the computation of error correction with faulty gates. Shor discovered[174] how to perform a general computation in the presence of noise, under the unphysical assumption that the noise decreases (slowly) with the size of the computation. A more physically reasonable assumption would be that the devices used in the laboratory have a constant amount of noise, independent of the size of the computation. To achieve fault tolerance against such noise, we apply a concatenation of Shor's scheme. We encode the state once, and then encode the encoded state, and so on for for several levels. This technique enabled the proof of the *threshold theorem*[127, 128, 107, 5, 125, 162], which asserts that it is possible to perform resilient quantum computation for as long as we wish, if the noise is smaller than a certain *threshold*. Decoherence and imprecision are therefore no longer considered insurmountable obstacles to realizing a quantum computation.

In accord with these theoretical optimistic results, attempts at implementations of quantum circuits are now being carried out all over the world. Unfortunately, the progress in this direction is much slower than the impressive pace in which theoretical quantum computation has progressed. The reason is that handling quantum systems experimentally is extremely difficult. Entanglement is a necessary ingredient in quantum computers, but experimentally, it is a fragile property which is difficult to create and preserve[65].

So far, entangled pairs of photons were created successfully[133, 185], and entanglement features such as violation of Bell inequalities were demonstrated [10, 11]. Even entangled pairs of atoms were created[114]. However quantum computation is advantageous only when macroscopically many particles are entangled[118, 6], a task which seems impossible as of now. Promising experimental developments come from the closely related subject of quantum cryptography[50, 34, 46]. Quantum communication was successfully tested[116, 147]. Bouwmeester *et. al.* have recently reported on experimental realization of quantum teleportation[43] . Suggestions for implementations of quantum computation [63, 74, 104, 142, 85, 117, 37, 145, 117, 160, 163, 182] include quantum dots, cold trapped ions and nuclear magnetic resonance, and some of these suggestions were already implemented [150, 187, 147, 104, 75]. Unfortunately, these implementations were so far limited to three qubits. With three qubits it is possible to perform partial error correction, and successful implementation of error correction of phases using NMR was reported[76, 60]. Using nuclear magnetic resonance techniques, a quantum algorithm was implemented which achieves proven advantage over classical algorithms[61]. It should be noted, however, that all these suggestions for implementation suffer from severe problems. In nuclear magnetic resonance the signal-to-noise ratio decays exponentially with the number of qubits[195], though a theoretical solution to this problem was given recently[168]. Other implementations do not allow parallel operations, which are necessary for error resilience[6]. In all the above systems controlling thousands of qubits seems hopeless at present. Never the less, the experimental successes encourage our hope that the ambitious task of realizing quantum computation might be possible.

The exciting developments in quantum computation give rise to deep new open questions in both the fields of computer science and physics. In particular, computational complexity questions shed new light on old questions in fundamental quantum physics such as the transition from quantum to classical physics, and the measurement process. I shall discuss these interesting topics at the end of the paper.

We will start with a survey of the important concepts connected to computation, in section 2. The model of quantum computation is defined in section 3. Section 4 discusses elementary quantum operations. Section 5 describes basic quantum algorithms by Deutsch and Jozsa's and by Simon. Shor's factorization algorithm is presented in section 6, while Fourier transforms are discussed separately in section 7, together with an alternative factorization algorithm by Kitaev. Grover's database search and variants are explained in section 8. Section 9 discusses the origins for the power of quantum computation, while section 10 discusses weaknesses of quantum computers. Sections 11, 12 and 13 are devoted to noise, error correction and fault tolerant computation. In Section 14 I conclude with a few remarks of a philosophical flavor.

2 What is a Computer?

Let us discuss now the basic notions of computational complexity theory: Turing machines, Boolean circuits, computability and computational complexity. The important complexity classes P and NP are also defined in this section. For more background, consult [73, 157]. We begin with the definition of a Turing machine:

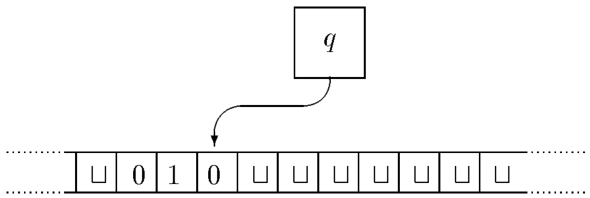
Definition 1 *A Turing machine (TM) is a triplet* $M = (\Sigma, K, \delta)$*.*

1. $\Sigma = \{\sqcup, 0, 1, ...\}$ *is a finite set of symbols which we call the alphabet.* $\sqcup$ *is a special symbol called the blank symbol.*

2. K *is a finite set of "machine states", with two special states:* $s \in K$ *the initial state and* $h \in K$ *the final state.*

3. *A transition function* $\delta : K \times \Sigma \longmapsto K \times \Sigma \times \{-1, 0, 1\}$

The machine works as follows: the tape has a head which can read and write on the tape during the computation. The tape is thus used as working space, or memory. The computation starts with an input of n symbols written in positions $[1, ...n]$ on the tape, all symbols except these n symbols are blank ($\sqcup$), the head is initially at position 1, and the state is initially s. Each time step, the machine evolves according to the transition function δ in the following way. If the current state of the machine is q and the symbol in the current place of the tape is σ, and $\delta(q, \sigma) = (q', \sigma', \epsilon)$, then the machine state is changed to q', the symbol under the head is replaced by σ' and the tape head moves one step in direction ϵ. (if $\epsilon = 0$ the head doesn't move). Here is a schematic description of a Turing machine:



Note that the operation of the Turing machine is local: It depends only on the current state of the machine and the symbol written in the current position of the tape. Thus the operation of the machine is a sequence of *elementary steps* which require a constant amount of effort. If the machine gets to "final state", h, we say that the machine has "halted". What is written at that time on the tape should contain the output. (Typically, the output will be given in the form "yes" or "no".) One can easily construct examples in which the machine never halts on a given input, for example by entering into an infinite loop.

According to the definition above, there are many possible Turing machines, each designed to compute a specific task, according to the transition function. However, there exists one Turing machine, U which when presented with an input, it interprets this input as a description of another Turing machine, M, concatenated with the description of the input to M, call it x. U will simulate efficiently the behavior of M when presented with the input x, and we write $U(M,x) = M(x)$. This U is a called a *universal* Turing machine. More precisely, the description of M should be given with some fixed notation. Without loss of generality, all the symbols and states of M can be given numbers from 1 to $|K|+|\Sigma|$. The description of M should contain $|K|$, $|\Sigma|$ and the transition function, which will be described by a set of rules (which is finite) of the form $((q,\sigma)(q',\sigma',\epsilon))$. For this, U's set of symbols will contain the symbols $''($" and $'')''$ apart from $\sqcup, 0, 1$. U will contain a few machine states, such as: "q_1: now reading input", "q_2: looking for an appropriate rule to apply" and so on. I will not go through the details, but it is convincing that with such a finite set of states, U can simulate the operation of any M on any input x, because the entire set of rules of the transition function is written on the tape.

The existence of a universal Turing machine leads naturally to the deep and beautiful subject of *non-computability.* A function is non-computable if it cannot be computed by a Turing machine, i.e. there is no Turing machine which for any given input, halts and outputs the correct answer. The most famous example is the HALTING problem. The problem is this: Given a description of a Turing machine M and its input x, will M halt on x?

Theorem 1 *There is no Turing machine that solves the HALTING problem on all inputs (M,x).*

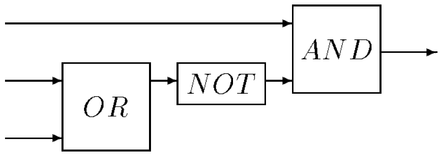
Proof: The proof of this theorem is conceptually puzzling. It uses the so called diagonal argument. Assume that H is a Turing machine, such that $H(M,x)$ is $''yes''$ if $M(x)$ halts and $''no''$ otherwise. Modify H to obtain $\tilde{H}$, such that

$$\begin{aligned} H(M,M) ='' \text{yes}'' &\longmapsto \tilde{H}(M) \text{ enters an infinite loop.} \\ H(M,M) ='' \text{no}'' &\longmapsto \tilde{H}(M) ='' \text{yes}''. \end{aligned}$$

The modification is done easily by replacing a few rules in the transition function of H. A rule which writes "yes" on the tape and causes H to halt is replaced by a rule that takes the machine into an infinite loop. A rule which writes "no" on the tape and causes H to halt is replaced by a rule that writes "yes" on the tape and than halts H. This way, $\tilde{H}$ is a "twisted" version of H. Now, does $\tilde{H}(\tilde{H})$ halt or not? We obtain a contradiction in both ways. Suppose it does halt. This means that $H(\tilde{H},\tilde{H}) ='' no''$ so $\tilde{H}(\tilde{H})$ does not halt! If $\tilde{H}(\tilde{H})$ does not halt, this means $H(\tilde{H},\tilde{H}) ='' yes''$ so $\tilde{H}(\tilde{H})$ does halt! ∎

This beautiful proof shows that there are functions which cannot be computed. The Turing machine is actually used to *define* which functions are computable and which are not.

It is sometimes more convenient to use another universal model, which is polynomially equivalent to Turing machines, called the Boolean circuit model. We will use the quantum analog of this model throughout this review. A Boolean circuit is a directed acyclic graph, with nodes which are associated with Boolean functions. These nodes are sometimes called *logical gates*. A node with n input wires and m output wires is associated with a function $f : \{0,1\}^n \longmapsto \{0,1\}^m$. Here is a simple example:



Given some string of bits as input, the wires carry the values of the bits, until a node is reached. The node computes a logical function of the bits (this function can be NOT, OR, AND, etc.) The output wires of the node, carry the output bits to the next node, until the computation ends at the output wires. The input wires can carry *constants* which do not vary with the different inputs to the circuit, but are part of the hardware of the circuit. In a Turing machine the transition function is local, so the operation is a sequence of elementary steps. In the circuit model the same requirement translates to the fact that the gates are local, i.e. that the number of wires which each node operates on is bounded above by a constant.

To measure the cost of the computation we can use different parameters: S, the number of gates in the circuit, or T, the time, or *depth* of the circuit. In this review, we will mainly be considered with S, the number of gates. We will be interested in the behavior of the cost, S, as a function of the size of the input, i.e. the number of wires input to the circuit, which we will usually denote by n. To find the cost function $S(n)$, we will look at a function f as a family of functions $\{f_n\}_{n=1}^{\infty}$, computed by a family of circuits $\{C_n\}_{n=1}^{\infty}$, each operating on n input bits; $S(n)$ will be the size of the circuit C_n.

I would like to remark here on an important distinction between the model of Turing machines and that of circuits. A lot of information can get into the circuit through the hardware. If we do not specify how long it takes to design the hardware, such circuits can compute even non-computable functions. This can be easily seen by an example. Define the circuit C_n to be a very simple circuit, which outputs a constant bit regardless of the n input bits. This constant bit will be 0 or 1 according to whether the n'th Turing machine, M_n (ordered according to the numerical description of Turing machines) halts on the input M_n or not. The family of circuits $\{C_n\}_{n=1}^{\infty}$ computes the non-computable HALTING problem with all the circuits having only one gate! This unreasonable computational power of circuits is due to the fact that we haven't specified who constructs the hardware of the circuit. We want to avoid such absurdity and concentrate on interesting and realistic cases. We will therefore require that the hardware of the circuits which compute $\{f_n\}_{n=1}^{\infty}$ can be designed with polynomial cost by a Turing machine. The Turing machine is given as an input the integer n, and outputs the specification of the circuit C_n. This model is called the "uniform circuit model", as opposed to the "non uniform" one, which is too strong. The models of uniform Boolean circuits and Turing machines are polynomially equivalent. This means that given a Turing machine which computes in polynomial time $f(x)$, there is a family of polynomial circuits $\{C_n\}_{n=0}^{\infty}$, specified by a polynomial Turing machine, such that C_n computes f_n. This correspondence is true also in reverse order, i.e. given the family of circuits there is a Turing machine that simulates them. Therefore the complexity of a computation does not depend (except for polynomial factors) on the model used. From now on, we will work only in the uniform circuit model.

One of the main questions in this review is whether the cost of the computation grows like a polynomial in n or an exponential in n. This distinction might seem arbitrary, but is better understood in the context of the complexity classes P and NP. The complexity class P is essentially the class of "easy" problems, which can be solved with polynomial cost:

Definition 2 : Complexity class P
$f=\{f_n\}_{n=1}^{\infty}\in P$ if there exists a uniform family of circuits $\{C_n\}_{n=1}^{\infty}$ of poly(n) size, where C_n computes the function $f_n(x)$ for all $x\in\{0,1\}^n$.

The class of *Non-deterministic Polynomial time* (in short, NP) is a class of much harder problems. For a problem to be in NP, we do not require that there exists a polynomial algorithm that solves it. We merely require that there exists an algorithm which can verify that a solution is correct in polynomial time. Another way to view this is that the algorithm is provided with the input for the problem and a *hint*, but the hint may be misleading. The algorithm should solve the problem in polynomial time when the hint is good, but it should not be mislead by bad hints. In the formal definition which follows, y plays the role of the hint.

Definition 3 : Complexity class NP
$f=\{f_n\}_{n=1}^{\infty}\in NP$ if there exists a uniform family of circuits, $\{C_n\}_{n=1}^{\infty}$, of poly(n) size, such that

If x satisfies $f_n(x)=''yes''\longmapsto$ there exists a string y of poly(n) *size such that $C_n(x,y)=1$,*

If x satisfies $f_n(x)=''no''$ there is no such y, i.e. for all $y's$, $C_n(x,y)=''no''$.

To understand this formal definition better, let us consider the following example for an NP problem which is called *satisfiability*:

> **Input**: A formula of n Boolean variables, $X_1,...X_n$, of the form
>
> $$g(X_1,...X_n)=(X_i\cup\neg X_j\cup X_k)\bigcap(X_m\cup\neg X_i)....$$
>
> which is the logical AND of poly(n) clauses, each clause is the logical OR of poly(n) variables or their negation.
>
> **Output**: $f(g)=1$ if there exists a satisfying assignment of the variables $X_1,...X_n$ so that $g(X_1,...X_n)$ is true. Else, $f(g)=0$.

To see that satisfiability is in NP, define the circuit C_n to get as input the specification of the formula g and a possible assignment $X_1, ...X_n$. The circuit will output $C_n(g, X_1, ...X_n) = g(X_1, ...X_n)$. It is easy to see that these circuits satisfy the requirements of the definition of NP problems. However, nobody knows how to build a polynomial circuit which gets g as an input, and finds whether a satisfying assignment exists. It seems impossible to find a satisfying assignment without literally checking all 2^n possibilities. Hence satisfiability is not known to be in P.

Satisfiability belongs to a very important subclass of NP, namely the NP *complete* problems. These are the hardest problems in NP, in the sense that if we know how to solve an NP-complete problem efficiently, we can solve any problem in NP with only polynomial slowdown. In other words, a problem f is NP-complete if any NP problem can be *reduced* to f in polynomial time. Garey and Johnson[101] give hundreds of examples of NP-complete problems, all of which are *reducible* one to another with polynomial slowdown, and therefore they are all equivalently hard. As of now, the best known algorithm for any NP-complete problem is exponential, and the widely believed conjecture is that there is no polynomial algorithm, i.e. $P \neq NP$. Perhaps the most important open question in complexity theory today, is proving this conjecture.

Another interesting class consists of those problems solvable with negligible probability for error in polynomial time by a probabilistic Turing machine. This machine is defined exactly as deterministic TM, except that the transition function can change the configuration in one of several possible ways, randomly. Equivalently, we can define *randomized circuits*, which are Boolean circuits with the advantage that apart from the input of n bits, they also get as input random bits which they can use as random coin flips. The class of problems solvable by uniform polynomial randomized circuits with bounded error probability is called BPP (*bounded probability polynomial*):

Definition 4 *$f = \{f_n\}_{n=1}^{\infty} \in BPP$ if there exists a family of uniform randomized circuits, $\{C_n\}_{n=1}^{\infty}$, of poly(n) size such that $\forall x \in \{0,1\}^n$, probability($C_n(x,y) = f_n(x)) \geq 2/3$, where the probability is measured with respect to a uniformly random y.*

Until the appearance of quantum computers, the modern Church thesis which asserts that a probabilistic Turing machine, or equivalently randomized uniform circuits, can simulate any reasonable physical device in polynomial time, held with no counterexamples. The quantum model, which I will define in the next chapter, is the only model which seems to be qualitatively different from all the others. We can define the quantum complexity classes:

Definition 5 *The complexity classes QP and BQP are defined like P and BPP, respectively, only with quantum circuits.*

It is known that $P \subseteq QP$ and $BPP \subseteq BQP$, as we will see very soon.

3 The Model of Quantum Computation

Deutsch was the first to define a rigorous model of quantum computation, first of quantum Turing machines[78] and then of quantum circuits[79]. I will describe first the model of quantum circuits, which is much simpler. At the end of the chapter, I present the model of quantum Turing machines, for completeness. For background on basic quantum mechanics such as Hilbert spaces, Schrödinger equation and measurements I recommend to consult the books by Sakurai[167], and by Cohen-Tanoudji[71]. As for more advanced material, the book by Peres[161] would be a good reference. However, I will give here all the necessary definitions.

A quantum circuit is a system built of two state quantum particles, called qubits. We will work with n qubits, the state of which is a unit vector in the complex Hilbert space $\mathcal{C}^2 \otimes \mathcal{C}^2 \otimes \cdots \otimes \mathcal{C}^2$. As the natural basis for this space, we take the basis consisting of 2^n vectors:

$$\begin{aligned} &|0\rangle \otimes |0\rangle \otimes \cdots \otimes |0\rangle \\ &|0\rangle \otimes |0\rangle \otimes \cdots \otimes |1\rangle \\ &\quad\vdots \\ &|1\rangle \otimes |1\rangle \otimes \cdots \otimes |1\rangle. \end{aligned} \tag{9}$$

For brevity, we will sometimes omit the tensor product, and denote

$$|i_1\rangle \otimes |i_2\rangle \otimes \cdots \otimes |i_n\rangle = |i_1, i_2, ..., i_n\rangle \equiv |i\rangle \tag{10}$$

where $i_1, i_2, ..., i_n$ is the binary representation of the integer i, a number between 0 and $2^n - 1$. This is an important step, as this representation allows us to use our quantum system to encode integers. This is where the quantum system starts being a computer. The general state which describes this system is a complex unit vector in the Hilbert space, sometimes called the *superposition:*

$$\sum_{i=0}^{2^n-1} c_i|i\rangle \tag{11}$$

where $\sum_i |c_i|^2 = 1$. The initial state will correspond to the "input" for the computation. Let us agree that for an input string i, the initial state of the system will be $|i\rangle$:

$$i \longmapsto |i\rangle \tag{12}$$

We will then perform "elementary operations" on the system. These operations will correspond to the computational steps in the computation, just like logical gates are the elementary steps in classical computers. In the meantime we will assume that all the operations are performed on an isolated system, so the evolution can always be described by a unitary matrix operating on the state of the system. Recall that a unitary matrix satisfies $UU^\dagger = I$, where $U^\dagger$ is the transposed complex conjugate of U.

Definition 6 *A quantum gate on k qubits is a unitary matrix U of dimensions $2^k \times 2^k$.*

Here is an example of a simple quantum gate, operating on one qubit.

$$NOT = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{13}$$

Recalling that in our notation $|0\rangle = (1,0)$ and $|1\rangle = (0,1)$, we have that $NOT|0\rangle = |1\rangle$ and $NOT|1\rangle = |0\rangle$. Hence, this gate flips the bit, and thus it is justified to call this gate the NOT gate. The NOT gate can operate on superpositions as well. From linearity of the operation,

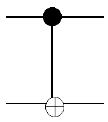
$$NOT(c_0|0\rangle + c_1|1\rangle) = c_0|1\rangle + c_1|0\rangle.$$

This linearity is responsible for the quantum parallelism (see Margolus[148]) which we will encounter in all powerful quantum algorithms. When the NOT gate operates on the first qubit in a system of n qubits, in the state $\sum_i c_i|i_1 i_2 ... i_n\rangle$ this state transforms to $\sum_i c_i(NOT|i_1\rangle)|i_2...i_n\rangle = \sum_i c_i|\neg i_1 i_2 ... i_n\rangle$. Formally, the time evolution of the system is described by a unitary matrix, which is a tensor product of the gate operating on the first qubit and the identity matrix I operating on the rest of the qubits.

Another important quantum gate is the *controlled NOT* gate acting on two qubits, which computes the classical function: $(a,b) \longmapsto (a, a \oplus b)$ where $a \oplus b = (a+b)$ mod 2 and $a, b \in 0, 1$. This function can be represented by the matrix operating on all 4 configurations of 2 bits:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{14}$$

The above matrix, as all matrices in this review, is written in the computational basis in lexicographic order. This gate is also called the *exclusive or* or XOR gate (On its importance see [86].) The XOR gate applies a NOT on the second bit, called the *target* bit, conditioned that the first *control* bit is 1. If a black circle denotes the bit we condition upon, we can denote the XOR gate by:



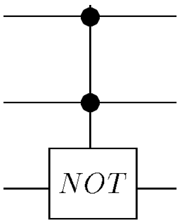
In the same way, all classical Boolean functions can be transformed to quantum gates. The matrix representing a classical gate which computes a reversible function, (in particular the number of inputs to the gate equals the number of outputs) is a permutation on all the possible classical strings. Such a permutation is easily seen to be unitary. Of course, not all functions are reversible, but they can easily be converted to reversible functions, by writing down the input bits instead of erasing them. For a function f,from n bits to m bits, we get the reversible function from $m+n$ bits to $m+n$ bits:

$$\begin{array}{c} f : i \longmapsto f(i) \\ \Downarrow \\ f_r : (i,j) \longmapsto (i, f(i) \oplus j). \end{array} \tag{15}$$

Applying this method, for example, to the logical AND gate, $(a,b) \longmapsto ab$ it will become the known Toffoli gate[186] $(a,b,c) \longmapsto (a,b,c \oplus ab)$, which is described by the unitary matrix on three qubits:

$$T = \begin{pmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & 0 & 1 \\ & & & & & & 1 & 0 \end{pmatrix} \tag{16}$$

The Toffoli gate applies NOT on the last bit, conditioned that the other bits are 1, so we can describe it by the following diagram:



Quantum gates can perform more complicated tasks than simply computing classical functions. An example of such a quantum gate, which is not a classical gate in disguise, is a gate which applies a general rotation on one qubit:

$$G_{\theta,\phi} = \begin{pmatrix} \cos(\theta) & \sin(\theta)e^{i\phi} \\ -\sin(\theta)e^{-i\phi} & \cos(\theta) \end{pmatrix} \tag{17}$$

To perform a quantum computation, we apply a sequence of elementary quantum gates on the qubits in our system. Suppose now, that we have applied all the quantum gates in our algorithm, and the computation has come to an end. The state which was initially a basis state has been *rotated* to the state $|\alpha\rangle \in C^{2^n}$. We now want to extract the output from this state. This is done by the process of *measurement*. The notion of measurement in quantum mechanics is puzzling. For example, consider a measurement of a qubit in the state $|\alpha\rangle = c_0|0\rangle + c_1|1\rangle$. This qubit is neither in the state $|0\rangle$ nor in $|1\rangle$. Yet, the *measurement postulate* asserts that when the state of this qubit is observed, it must decide on one of the two possibilities. This decision is made non-deterministically. The classical outcome of the measurement would be 0 with probability $|c_0|^2$ and 1 with probability $|c_1|^2$. After the measurement, the state of the qubit is either $|0\rangle$ or $|1\rangle$, in consistency with the classical outcome of the measurement. Geometrically, this process can be interpreted as a projection of the state on one of the two orthogonal subspaces, S_0 and S_1, where $S_0 = \text{span}\{|0\rangle\}$ and $S_1 = \text{span}\{|1\rangle\}$, and a measurement of the state of the qubit $|\alpha\rangle$ is actually an observation in which of the subspaces the state is, in spite of the fact that the state might be in neither. The probability that the decision is S_0 is the norm squared of the projection of $|\alpha\rangle$ on S_0, and likewise for 1. Due to the fact that the norm of $|\alpha\rangle$ is one, these probabilities add up to one. After the measurement $|\alpha\rangle$ is projected to the space S_0 if the answer is 0, and to the space S_1 if the answer is 1. This projection is the famous *collapse* of the wave function. Now what if we measure a qubit in a system of n qubits? Again, we project the state onto one of two subspaces, S_0 and S_1, where S_a is the subspace spanned by all basis states in which the measured qubit is a. The rule is that if the measured superposition is $\sum_i c_i|i_1, ...i_n\rangle$, a measurement of the first qubit will give the outcome 0 with probability $\text{Prob}(0) = \sum_{\mathrm{i_2,...i_n}} |\mathrm{c_{0,i_2,...i_n}}|^2$, and the superposition will collapse to

$$\frac{1}{\text{Prob}(0)} \sum_{i_2,...i_n} c_{0,i_2,...i_n}|0, i_2, ...i_n\rangle,$$

and likewise with 1. Here is a simple example: Given the state of two qubits:

$$\frac{1}{\sqrt{3}}(|00\rangle + |01\rangle - |11\rangle),$$

the probability to measure 0 in the left qubit is 2/3, and the probability to measure 1 is 1/3. After measuring the left qubit, the state has collapsed to $\frac{1}{\sqrt{2}}(|00\rangle + |01\rangle)$ with probability $\Pr(0) = 2/3$ and to $-|11\rangle$ with probability $\Pr(1) = 1/3$. Thus, the resulting state depends on the outcome of the measurement. After the collapse, the projected state is renormalized back to 1.

We can now summarize the definition of the model of quantum circuits. A quantum circuit is a directed acyclic graph, where each node in the graph is associated a quantum gate. This is exactly the definition from section 2 of classical Boolean circuits, except that the gates are quantum. The input for the circuit is a basis state, which evolves in time according to the operation of the quantum gate. At the end of the computation we apply measurements on the output qubits (The order does not matter). The string of classical outcome bits is the classical output of the quantum computation. This output is in general probabilistic. This concludes the definition of the model.

Let us now build a repertoire of quantum computations step by step. We have seen that classical gates can be implemented quantumly, by making the computation reversible. More explicitly,

Lemma 1 *Let f be a function from n bits to m bits, computed by a Boolean circuit C of size S. There exists a quantum circuit Q which computes the unitary transformation on $n+m$ qubits: $|0^b, i, j\rangle \longmapsto |0^b, i, f(i) \oplus j\rangle$. b and the size of Q are linear in S.*

Proof: Replace each gate in C by its reversible extension, according to equation 15. We will add b extra bits for this purpose. The input for this circuit is thus $(0^b, i)$. The modified C, denoted by $\tilde{C}$, can be viewed as a quantum circuit since all its nodes correspond to unitary matrices. The function that it computes is still not the required function, because the input i is not necessarily part of the output as it should be. To solve this problem, we add to $\tilde{C}$ m extra wires, or qubits. The input to these wires is 0. At the end of the sequence of gates of $\tilde{C}$, we copy the m "result" qubits in $\tilde{C}$ on these m blank qubits by applying m CNOT gates. We now apply in reverse order the reversed gates of all the gates applied so far, except the $CNOT$ gates. This will reverse all operations, and retain the input $(0^b, i)$, while the m last qubits contain the desired $f(i)$. ∎

The state of the system is always a basis state during the computation which is described in the proof. Hence measurements of the final state will yield exactly the expected result. This shows that any computation which can be done classically can also be done quantumly with the same efficiency, i.e. the same order of number of gates. We have shown:

Theorem 2 $P \subseteq QP$

In the process of conversion to reversible operations, each gate is replaced by a gate operating on more qubits. This means that making circuits reversible costs in adding a linear number of extra qubits. In [32], Bennett used a nice pebbling argument, to show that the space cost can be decreased to a logarithmic factor with only a minor cost in time: $T \longmapsto T^{1+\epsilon}$. Thus the above conversion to quantum circuit can be made very efficient.

To implement classical computation we must also show how to implement probabilistic algorithms. For this we need a quantum subroutine that generates a random bit. This is done easily by measurements. We define the Hadamard gate which acts on one qubit. It is an extremely useful gate in quantum algorithms.

$$H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \tag{18}$$

Applying this gate on a qubit in the state $|0\rangle$ or $|1\rangle$, we get a superposition: $\frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$. A measurement of this qubit yields a random bit. Any classical circuit that uses random bits can be converted to a quantum circuit by replacing the gates with reversible gates and adding the "quantum random bit" subroutine when needed. Note that here we allow measuring in the middle of the computation. This shows that:

Theorem 3 $BPP \subseteq BQP$

The repertoire of classical algorithms can therefore be simulated efficiently by quantum computers. But quantum systems feature characteristics which are far more interesting. We will encounter these possibilities when we discuss quantum algorithms.

Let me define here also the model of quantum Turing Machine[78, 38, 179] (QTM) which is the quantum analog of the classical TM. The difference is that all the degrees of freedom become quantum: Each cell in the tape, the state of the machine, and the reading head of the tape can all be in linear superpositions of their different possible classical states.

Definition 7 *A quantum Turing machine is specified by the following items:*

1. *A finite alphabet* $\Sigma = \{\sqcup, 0, 1...\}$ *where* $\sqcup$ *is the blank symbol.*
2. *A finite set* $K = \{q_0, ...q_s\}$ *of "machine states", with* $h, s \in K$ *two special states.*
3. *A transition function* $\delta : Q \times \Sigma \times Q \times \Sigma \times \{-1, 0, 1\} \longmapsto \mathcal{C}$

As in classical TM, the tape is associated a head that reads and writes on that tape. A classical configuration, c, of the Turing machine is specified by the head's position, the contents of the tape and the machine's state. The Hilbert space of the QTM is defined as the vector space, spanned by all possible classical configurations $\{|c\rangle\}$. The dimension of this space is infinite. The computation starts with the QTM in a basis state $|c\rangle$, which corresponds to the following classical configuration: An input of n symbols is written in positions $1, .., n$ on the tape, all symbols except these n symbols are blank ($\sqcup$,) and the head is at position 1. Each time step, the machine evolves according to an infinite unitary matrix which is defined in the following way. $U_{c,c'}$, the probability amplitude to transform from configuration c to c' is determined by the transition function δ. If in c, the state of the machine is q and the symbol in the current place of the tape head is σ then $\delta(q, \sigma, q', \sigma', \epsilon)$ is the probability amplitude to go from c to c', where c' is equal to c everywhere except locally. The machine state in c', q, is changed to q', the symbol under the head is changed to σ' and the tape head moves one step in direction ϵ. Note that the operation of the Turing machine is local, i.e. it depends only on the current state of the machine and the symbol now read by the tape. Unitarity of infinite matrices is not easy to check, and conditions for unitarity were given by Bernstein and Vazirani[38].

In my opinion, the QTM model is less appealing than the model of quantum circuits, for a few reasons. First, QTMs involve infinite unitary matrices. Second, it seems very unlikely that a physical quantum computer, will resemble this model, because the head, or the apparatus executing the quantum operations, is most likely to be classical in its position and state. Another point is that the QTM model is a sequential model, which means that it is able to apply only one operation at each time step. Aharonov and Ben-Or showed[6] that a sequential model is fundamentally incapable of operating fault tolerantly in the presence of noise. Above all, constructing algorithms is much simpler in the circuit model. For these reasons I will restrict this review to quantum circuits. The model of quantum circuits, just like that of classical circuits, has a "uniform" and "non-uniform" versions. Again, we will restrict ourselves to the uniform model, i.e. quantum circuits which can be designed in polynomial time on a classical Turing Machine. Yao[202] showed that uniform quantum circuits are polynomially equivalent to quantum Turing machines, by a proof which is surprisingly complicated. This proof enables us the freedom of choosing whichever model is more convenient for us.

Another model worth mentioning in this context is the quantum cellular automaton[148, 196, 88, 77]. This model resembles quantum circuits, but is different in the fact that the operations are homogeneous, or periodic, in space and in time. The definition of this model is subtle and, unlike the case of quantum circuits, it is not trivial to decide whether a given quantum cellular automaton obeys the rules of quantum mechanics or not[88]. Another interesting quantum model is that of a finite state quantum automaton, which is similar to a quantum Turing machine except it can only read and not write, so it has no memory. It is therefore a very limited model. In this model Watrous[132] showed an interesting algorithm which uses interference, and is able to compute a function which cannot be computed in the analogous classical model.

4 Universal Quantum Gates

What kind of elementary gates can be used in a quantum computation program? We would like to write our program using elementary steps: i.e., the algorithm should be a sequence of steps, each potentially implementable in the laboratory. It seems that achieving controlled interactions between a large number of qubits in one elementary step is extremely difficult. Therefore it is reasonable to require an "elementary gate" to operate on a small number of qubits, (independent of n which can be very large.) We want our computer to be able to compute any function. The set of elementary gates used should thus be *universal*. For classical reversible computation, there exists a single universal gate[96, 186], called the Toffoli gate, which we have already encountered. This gate computes the function

$$a, b, c \longmapsto a, b, ab \oplus c.$$

The claim is that any reversible function can be represented as a concatenation of the Toffoli gate on different inputs. For example, to construct the logical AND gate on a, b, we simply input $c = 0$, and the last bit will contain $ab \oplus 0 = AND(a, b)$. To implement the NOT gate on the third bit we set the first two bits to be equal to 1. We now have what is well known to be a universal set of gates, The NOT and AND gates. In the quantum case, the notion of universality is slightly more complicated, because operations are continuous. We need not require that all operations are achieved exactly, but a very good approximation suffices. The notion of approximation is very important in quantum computation. Frequently operations are approximated instead of achieved exactly, without significantly damaging the correctness of the computation.

Definition 8 Approximation:

A unitary matrix U is said to be approximated to within ϵ by a unitary matrix U' if $|U - U'| \leq \epsilon$.

The norm we use is the one induced by the Euclidean norm on vectors in the Hilbert space.

Note that unitary transformations can be thought of as rigid rotations of the Hilbert space. This means that angles between vectors are preserved during the computation.The result of using U' instead of U, where $|U - U'| \leq \epsilon$, is that the state is tilted by an angle of order ϵ from the correct state. However this angle does not grow during the computation, because the rotation is rigid. The state always remains within ϵ angle from the correct state. Therefore the overall error in the entire computation is additive: it is at most the sum of the errors in all the gates. This shows that the accuracy to which the gates should be approximated is not very large. If S gates are used in the circuit, it suffices to approximate each gate to within $O(\frac{1}{S})$, in order that the computation is correct with constant probability[38].

We can now define the notion of universal gates, which approximate any possible quantum operation:

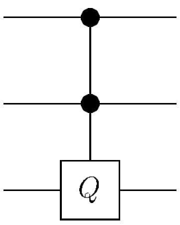
Definition 9 Universal Set of Gates:

A set of quantum gates, $\mathcal{G}$, is called universal if for any ϵ and any unitary matrix U on any number of bits, U can be approximated to within $\epsilon > 0$ by a sequence of gates from $\mathcal{G}$. In other words, the subgroup generated by $\mathcal{G}$ is dense in the group of unitary operators, $U(n)$, for all n.

Deutsch was the first to show a universal elementary gate, which operates on three qubits[79]. Bernstein and Vazirani[38] gave another proof of universality in terms of QTM. It was then shown by DiVincenzo that two-qubit gates are universal[84]. This is an important result, since it seems impossible to control interactions between three particles, whereas two particle interactions are likely to be much easier to implement. It was a surprising achievement, since in reversible classical computation, which is a special case of quantum computation, there is no set of two bit gates which is universal. Note that one qubit gate is certainly not enough to construct all operations. Barenco[13] and Deutsch *et.al*[81] showed that almost any two-bit gate is universal (See also Lloyd [141, 143]). An improvement of DiVincenzo's result was achieved later by Barenco *et.al*[16], where it was shown that the classical controlled not gate, together with all one-qubit gates construct a universal set as well! In fact, one 1-qubit gate and the controlled not gate will do. This is perhaps the simplest and most economic set constructed so far. Implementation of one qubit gates are feasible, and experimentalists have already implemented a controlled not gate [187]. However, there are other possible sets of gates. Adleman, *et. al.*[3] and Solovay[179] suggested a set of gates, where all entries of the matrices are $\pm\frac{3}{5}$ and $\pm\frac{4}{5}$ and ± 1. Other universal sets of gates were suggested in connection with fault tolerant quantum computation[174, 5, 128].

Why do we need so many possible universal sets to choose from? Universal sets of gates are our computer languages. At the lowest level, we need *quantum assembly*, the machine language by which everything will be implemented. For this purpose, we will use the set which consists of the easiest gates to implement in the laboratory. Probably, the set of one and two qubit gates will be most appropriate. Another incentive is analyzing the complexity power of quantum computers. For this the set suggested by Solovay and by Adleman *et. al.* seems more appropriate. (Fortnow recently reported on bounds using this set[95]). We will see that for error correction purposes, we will need a completely different universal set of gates. An important question should arise here. If our computer is built using one set, how can we design algorithms using another set, and analyze the computational power using a third set? The answer is that since they are all universal sets, there is a way to *translate* between all these languages. A gate from one set can be approximated by a sequence of gates from another set. It turns out that in all the universal sets described here, the approximation to within ϵ of an operation on k qubits takes poly$(\log(\frac{1}{\epsilon}), 2^{k})$ gates from the set. As long as the gates are local (i.e k is constant) the translation between different universal sets is efficient.

Now that the concept of a universal set of gates is understood, I would like to present an example of a simple universal set of gates. It relies on the proof of Deutsch's universal gate. The idea underlying Deutsch's universal gate is that Reversible computation is a special case of quantum computation. It is therefore natural that universal quantum computation can be achieved by generalizing universal reversible computation. Deutsch showed how to generalize Toffoli's gate so that it becomes a universal gate for quantum computation:



The NOT matrix in the original Toffoli gate (see equation 16) is replaced by another unitary matrix on one qubit, Q, such that Q^n can approximate any $2 \otimes 2$ matrix. I will present here a modification of Deutsch's proof, using two gates of the above form. Define:

$$U = \begin{pmatrix} \cos(2\pi\alpha) & \sin(2\pi\alpha) \\ -\sin(2\pi\alpha) & \cos(2\pi\alpha) \end{pmatrix}, W = \begin{pmatrix} 1 & 0 \\ 0 & e^{i2\pi\alpha} \end{pmatrix}. \tag{19}$$

We have freedom in choosing α, except we require that the sequence
$\alpha \bmod 1, 2\alpha \bmod 1, 3\alpha \bmod 1, ...$ hits the ϵ-neighborhood of any number in $[0, 1]$, within poly$(\frac{1}{\epsilon})$ steps. Clearly, α should be irrational, but not all irrational numbers satisfy this property. It is not very difficult to see that an irrational root of a polynomial of degree 2

satisfies the required property. Let U_3 (W_3) be the generalized Toffoli gate with U (W) playing the role of the conditioned matrix, Q, respectively. Then

Theorem 4 *$\{U_3, W_3\}$ is a universal set of quantum gates.*

Proof: First, note that according to the choice of α, U approximates any rotation in the real plane, and W approximates any rotation in the complex plane. Given an 8×8 unitary matrix U, let us denote its 8 eigenvectors as $|\psi_j\rangle$ with corresponding eigenvalues $e^{i\theta_j}$. U is determined by $U|\psi_j\rangle = e^{i\theta_j}|\psi_j\rangle$. Define:

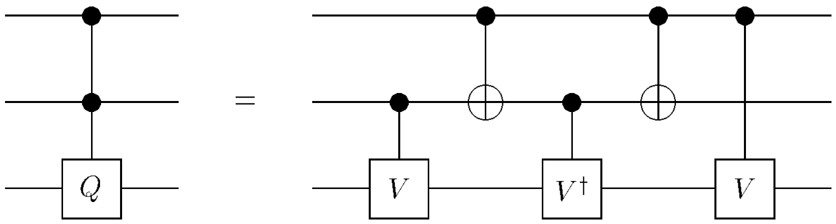
$$U_k|\psi_j\rangle = \begin{cases} |\psi_j\rangle & \text{if } k \neq j \\ e^{i\theta_k}|\psi_k\rangle & \text{if } k = j \end{cases} \tag{20}$$

Then $U = U_7 U_6 .... U_0$. U_k can be achieved by first taking $|\psi_k\rangle$ to $|111\rangle$, by a transformation which we will denote by R. After R we apply W the correct number of times to approximate $|111\rangle \longmapsto e^{i\theta_k}|111\rangle$ and then we take $|111\rangle$ to $|\psi_k\rangle$ by applying the reverse transformation of R, R^{-1}.

It is left to show how to apply R, i.e. how to take a general state $|\psi\rangle = \sum_{i=0}^{7} c_i|i\rangle$ to $|111\rangle$. For this, note that U_3^n can approximate the Toffoli gate, and therefore can approximate all permutations on basis states. To apply $|\psi\rangle \longmapsto |111\rangle$, first turn the coefficient on the coordinate $|110\rangle$ to 0. This is done by applying W an appropriate number of times so that the phase in the coefficient of $|110\rangle$ will equal that of $|111\rangle$. The coefficients now become $c_6 = r_6 e^{i\phi}, c_7 = r_7 e^{i\phi}$. Let θ be such that $r_6 = r\sin(\theta), \mathrm{r}_7 = \mathrm{r}\cos\theta$. Now apply U an appropriate number of times to approximate a rotation by $-\theta$. This will transform all the weight of $|110\rangle$ to $|111\rangle$. In the same way we transform the weight from all coordinates to $|111\rangle$, using permutations between coordinates. This achieves $|\psi\rangle \longmapsto |111\rangle$, i.e. the transformation R. R^{-1} is constructed in the same way.

We have shown that all three qubit operations can be approximated. For operations on more qubits, note that the group generated by $\{U_m, W_m\}$ is dense in all operations on m bits, by the same reasoning. To create U_m (W_m) from U_3 (W_3) use recursion: compute the logical AND of the first two bits by a Toffoli gate writing it on an extra bit, and then apply U_{m-1} (W_{m-1}). The reader can verify that the approximation is polynomially fast, i.e. for fixed m, any unitary matrix on m qubits can be approximated to within ϵ by $poly(\frac{1}{\epsilon})$ applications of the gates U_3 and W_3. $\Box$

The generalized Toffoli gates operate on three qubits. Barenco *et. al.*[16] show an explicit sequence of two bit gates which constructs any matrix on three qubits, of the form of a generalized Toffoli gate:



where $V=\sqrt{Q}$. Thus, two bit gates are universal. $\square$

It was further shown[16] that one-qubit matrix conditioned on one other qubit can be expressed as a sequence of one-qubit matrices and $CNOT$s. So the generalized Toffoli gate of Deutsch can be written as a finite sequence of one-qubit gates and $CNOT$s. This shows that $\{One-qubit\ gates, CNOT\}$ is universal.

The description above shows how to approximate unitary matrices using poly($\frac{1}{\epsilon}$) gates from the universal set. In fact, an exponentially faster approximation is possible due to a theorem by Kitaev [122], which was also proved by Solovay[179]:

Theorem 5 *Let the matrices $U_1,...U_r \in SU(n)$ generate a dense subset in $SU(n)$. Then any matrix $U \in SU(n)$ can be approximated to within ϵ by a product of poly*($\log(\frac{1}{\epsilon})$) *matrices from $U_1,...U_r,U_1^\dagger,...U_r^\dagger$.*

$SU(n)$ is the set of $n\times n$ unitary matrices with determinant 1. Given a universal quantum set, we can easily convert it to a set in $SU(n)$ by multiplying each matrix with an overall complex scalar of absulute value 1, namely a phase. This overall phase does not effect the result of any measurement, so any gate can be multiplied by a phase without affecting the computation. We thus have:

Corollary 1 *The approximation rate of any universal set of quantum gates is exponential.*

The idea of the proof of the theorem is to construct finer and finer nets of points in $SU(n)$. The $2k$'th net is constructed by taking commutators of points from the k'th net. Each point in the k'th net is a product of a linear (in k) number of gates from the set of gates. It turns out that the distance between two adjacent points in the net decreases exponentially with k. ∎

Having chosen the set of gates to write algorithms with, actually writing the algorithm in this assembler-like language seems like a very tedious task! Just like higher languages in ordinary computer programming, it is desirable that quantum operations which are commonly used can be treated as black boxes, without rewriting them from the beginning with elementary gates. Steps in this direction were made by [16, 15, 24, 155, 193].

5 Quantum Algorithms

The first and simplest quantum algorithm which achieves advantage over classical algorithms was presented by Deutsch and Jozsa[79]. Deutsch and Jozsa's algorithm addresses a problem which we have encountered before, in the context of probabilistic algorithms.

> f is a Boolean function from $\{1, N\}$ to $\{0, 1\}$. Assume $N = 2^n$ for some integer n. We are promised that $f(i)$ are either all equal to 0, ("constant") or half are 0 and half are 1 ("balanced"). We are asked to distinguish between the two cases.

The question is presented in the *oracle* setting. This means that the circuit does not get $f(1), ...f(N)$ as input. Instead, the circuit has access to an *oracle* for f. A *query* to the oracle is a gate with n input wires carrying an integer $i \in \{1, n\}$ in bit representation. The output from the oracle gate is $f(i)$. A quantum query to the oracle means applying the unitary transformation $|i\rangle|j\rangle \longmapsto |i\rangle|j \oplus f(i)\rangle$. The cost is measured by the number of queries to the oracle. A classical algorithm that solves this question exactly will need $O(N)$ queries. The quantum algorithm of Deutsch and Jozsa solves the problem exactly, with merely one quantum query! The algorithm makes use of a transformation known as the discrete Fourier transform over the group Z_2^n.

$$|i\rangle \stackrel{ForiuerTransform}{\longrightarrow} \frac{1}{\sqrt{N}} \sum_j (-1)^{i \cdot j} |j\rangle \tag{21}$$

where i, j are strings of length n, and $i \cdot j = \sum_{k=1}^n i_k j_k \ mod\ 2$, the inner product of i and j modulo 2. Meanwhile, we need only one easily verified fact about the Fourier transform over Z_2^n: To apply this transformation on n qubits, we simply apply the Hadamard transform H from equation 18 on each of the n qubits. Note also that the reversed Fourier transform, FT^{-1} is equal to the FT. We now turn to solve Deutsch and Jozsa's problem. We will work with two registers, one will hold a number between 1 to N and therefore will consist of n qubits, and the other register will consist of one qubit that will carry the value of the function.

Deutsch and Jozsa's Algorithm

$|0^n\rangle \otimes |1\rangle$

Apply Fourier transform on first register.
Apply H on last qubit

$$\Downarrow$$

$\frac{1}{\sqrt{N}} \sum_{i=1}^{N} |i> \otimes (\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle)$

Call oracle, $|i\rangle|j\rangle \longmapsto |i\rangle|j \oplus f(i)\rangle$.

$$\Downarrow$$

$\frac{1}{\sqrt{N}} \sum_{i=1}^{N} (-1)^{f(i)}|i> \otimes (\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle)$

Apply reversed Fourier transform on first register

$$\Downarrow$$

$|\psi\rangle \otimes (\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle)$

Measure first register

$$\Downarrow$$

If outcome equals 0^n, output "constant"
Else, output "balanced"

To see why this algorithm indeed works, let us denote by $|\psi_c\rangle$ the vector $|\psi\rangle$ in the case "constant", and $|\psi_b\rangle$ the vector $|\psi\rangle$ in the case "balanced". Note that if $f(i)$ is constant, the second Fourier transform merely undoes the first Fourier transform, so $|\psi_c\rangle = |0^n\rangle$. On the other hand, if $f(i)$ is balanced, the vector

$$\frac{1}{\sqrt{N}} \sum_{i=1}^{N} (-1)^{f(i)}|i>$$

is orthogonal to

$$\frac{1}{\sqrt{N}}\sum_{i=1}^{N}|i> .$$

Since unitary operations preserve angles between vectors, $|\psi_b\rangle$ is orthogonal to $|\psi_c\rangle$. Hence the probability to measure 0^n in the "balanced" case is zero. Hence, the algorithm gives the correct answer with probability 1. This algorithm shows the advantage of exact quantum complexity over exact classical complexity. However, when the restriction to exact solution is released, this advantage is gone. A classical probabilistic machine can solve the problem using a constant number of queries - though not by one query! (This was shown in the overview).

Let me remark that discussing exact solutions is problematic in the context of quantum algorithms, because of the continuous characteristics of quantum operators. Almost all quantum computations cannot be achieved exactly, when using a finite universal set of gates; the set of unitary operations is continuous, while the set of achievable operations using a finite universal set of gates is countable. Moreover, the notion of exact quantum algorithms is not robust, because the set of problems that have exact solution depend very strongly on the universal set of gates. The function AND, for example,cannot be computed exactly by Deutsch's universal machine!

In the next algorithm, due to Simon, the exponential advantage is achieved even without requiring exact solutions. The problem can be specified as follows:

Simon's Problem:

f is a function from $\{1, N\}$ to $\{1, N\}$, where $N = 2^n$. We are promised that one of two cases occurs:

Either all $f(i)$ are different, i.e. f is "one to one",

or

f satisfies that $\exists s, f(i) = f(j)$ if and only if $i = j$ or $i = j \oplus s$, i.e f is "two to one".

We are asked to distinguish between the two cases.

Here a classical computer will need order of $O(N)$ queries, even when an error is allowed. Simon's quantum algorithm can solve this question with the expected number of queries being $O(\log(\mathrm{N}))$. (In fact, Brassard *et.al.* improved this result from expected $O(\log(\mathrm{N}))$ queries to worst case $O(\log(\mathrm{N}))$ queries[48].)

We will work with two registers of n qubits; both will hold an integer between 1 to N. The first register will carry numbers in the range of the function. The second register will carry the value of the function.

Simon's Algorithm

$|0^n\rangle \otimes |0^n\rangle$

Apply Fourier transform on first register.

$$\Downarrow$$

$\frac{1}{\sqrt{N}} \sum_{i=1}^{N} |i > \otimes |0^n\rangle$

Call oracle

$$\Downarrow$$

$\frac{1}{\sqrt{N}} \sum_{i=1}^{N} |i > \otimes |f(i)\rangle$

Apply Fourier transform on first register.

$$\Downarrow$$

$\frac{1}{N} \sum_{k=1}^{N} |k > \otimes \sum_{i=1}^{N} (-1)^{i \cdot k} |f(i)\rangle$

Measure first register. Let k_1 be the outcome.
Repeat the previous steps cn times to get k_1, k_2,..., k_{cn}.

$$\Downarrow$$

Apply Gauss elimination to find a non-trivial solution for s in the set of equations:

$$\begin{aligned} k_1 \cdot s &= 0 \; mod \; 2 \\ k_2 \cdot s &= 0 \; mod \; 2 \\ &\vdots \\ k_{cn} \cdot s &= 0 \; mod \; 2 \end{aligned}$$

$$\Downarrow$$

If found, output "two to one". If not, declare "one to one".

Proof of correctness: To see why this algorithm works, let us analyze the probability to measure $k_1 = k$, in the two cases. In the case of "one to one", the probability to measure

$k_1 = k$ is independent of k:

$$\mathrm{Prob}(\mathrm{k}_1 = \mathrm{k}) = \sum_{\mathrm{i}} \left| \frac{(-1)^{\mathrm{i}\cdot\mathrm{k}}}{\mathrm{N}} \right|^2 = \frac{1}{\mathrm{N}}. \tag{22}$$

The above formula is derived by computing the squared norm of the projection of the measured vector on $|k\rangle \otimes |f(i)\rangle$ and summing over all possible $f(i)$. If we do the same thing in the "two to one" case, the projection on $|k\rangle \otimes |f(i)\rangle$ will consist of two terms: one comes from i and the other from $i \oplus s$, since $f(i) = f(i \oplus s)$. Hence, in the following sum we divide by 2 to correct for the fact that every term is counted twice. In the case "two to one", we derive:

$$\mathrm{Prob}(\mathrm{k}_1 = \mathrm{k}) = \frac{1}{2} \sum_{\mathrm{i}} \frac{1}{\mathrm{N}^2} |(-1)^{\mathrm{i}\cdot\mathrm{k}} + (-1)^{(\mathrm{i}\oplus\mathrm{s})\cdot\mathrm{k}}|^2 = \begin{cases} \frac{2}{N} & \text{if } k \cdot s = 0 \ mod\ 2 \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

So we will only measure k which is orthogonal to s. In order to distinguish between the cases, we repeat the experiment many times, and observe whether the space spanned by the random vectors is the whole space or a subspace. If we perform a large enough number of trials, we can be almost sure that in the "one to one" case, the vectors will span the whole space. Hence finding a non trivial solution will mean that we are in the "two to one" case. A more precise argument follows. Let V be a vector space of dimension n over Z_2. Let $S \subset V$ be the subspace spanned by the vectors, $k_1, \ldots k_t$, which were measured at the first t trials. If S is not equal to V, a random vector k_{t+1} from V will be in S with probability at most $\frac{1}{2}$. Hence, with probability greater than half, the dimension of $\mathrm{span}\{S, k_{t+1}\}$ is larger than that of S. By Chernoff's law[56], the probability the vectors will not span the whole space after cn trials is exponentially small in n. $\Box$

This algorithm is exponentially more efficient than any randomized classical algorithm! This seems like an extremely strong result, but it is very important to notice here that the problem is stated in the oracle setting and that the algorithm does not apply for any oracle, but only on oracles from a restricted set: either "balanced" or "constant" functions. This restriction is called in complexity theory a "promise" to the algorithm: the algorithm is "promised" that the oracle is from some restricted subset. We will see later, in section 10, that without such a "promise", quantum computation and classical computation are polynomially equivalent in terms of number of queries to the oracle. This shows that in the absence of a promise, i.e. full range input, the quantum advantage is exhibited not in the number of accesses to the input, but in the way the information is processed. We will see an example for this in the next section, in Shor's factorization algorithm.

6 Shor's Algorithm for Factoring Integers

Shor's algorithm is the most important algorithmic result in quantum computation. The algorithm builds on ideas that already appear in Deutsch and Jozsa's algorithm and in

Simon's algorithm, and like these algorithms, the basic ingredient of the algorithm is the Fourier transform. The problem can be stated as follows:

Input: An integer N
Output: A non-trivial factor of N, if exists.

There is no proof that there is no polynomial classical factorization algorithm The problem is even not known to be NP-complete. However, factorization is regarded as hard, because many people have tried to solve it efficiently and failed. In 1994, Shor published a polynomial (in log(N)) quantum algorithm for solving this problem [172]. This result is regarded as extremely important both theoretically and practically, although there is no proof that a classical algorithm does not exist. The reason for the importance of this algorithm is mainly the fact that the security of the RSA cryptosystem, which is so widely used, is based on the assumed hardness of factoring integers. Before explaining the algorithm, I would like to explain here in short how this cryptosystem works.

A cryptosystem is a secure way to transform information such that an eavesdropper will not have any information about the message sent. In the RSA method, the receiver, Bob, who will get the message, sends first a public key to Alice. Alice uses this key to encode her message, and sends it to Bob. Bob is the only one who can encode the message, assuming factoring is hard.

The RSA cryptosystem

Alice		**Bob**
	N, E $\longleftarrow$	P, Q *large primes. Set* $N = PQ$. E *coprime with* $P-1, Q-1$
Message M	$M^E mod\ N$ $\longrightarrow$	
		Computes $E^{-1} mod (P-1)(Q-1)$, *Computes* $(M^E)^{E^{-1}} modN = M$

The key is chosen as follows: Bob chooses two large primes P and Q. He then computes $N = PQ$, and also picks an integer co-prime to $(P-1)(Q-1) = \phi(N)$, the number of co-primes to N smaller than N. Bob sends E and N to the sender, Alice, using a public domain (newspaper, phone...) The pair (E, N) is called Bob's *public key.* Bob keeps secret $D = E^{-1} mod(P-1)(Q-1)$, which he can compute easily knowing P and Q, using the extended Euclid's algorithm[73]. The pair (N, D) is called Bob's *secret key.* Alice computes her message, M, to the power of E, modulo N, and sends this number in a public channel to Bob. Note that Alice's computation is easy: taking a number Y to the power of X modulo N is done by writing X in binary representation: $X = x_1...x_n$. Then one can square (Y^{x_i}) i times to get $(Y^{x_i})^{2^i}$, add the results for all i and take the modulus over N. Bob decodes Alice's massage using his secret key by computing $(M^E)^D modN$.

Why does Bob get the correct message M? This follows from Fermat's little theorem and the Chinese remainder theorem which together imply[73] that for any M, $M^{k\phi(N)+1} = M \ mod \ N$. The security of this cryptosystem rests on the difficulty of factoring large numbers. If the eavesdropper has a factorization algorithm, he knows the factors P, Q, and he can simply play the role of Bob in the last step of the cryptographic protocol. The converse statement, which asserts that in order to crack RSA one must have a factoring algorithm, is not proven. However, all known methods to crack RSA can be polynomially converted to a factorization algorithm. Since factorization is assumed hard, classically, RSA is believed to be a secure cryptosystem to use. In order to use RSA securely, one should work with integers that are a few hundreds digits in length, since factoring smaller integers is still practical. Integers of up to 130 digits have been factorized by classical computers in no longer than a few weeks. Due to the fact that the only classical factorization algorithm is exponential, factorizing a number of twice the number of digits will take an eavesdropper not twice the time, but of the order of million years. If Alice and Bob work with numbers of the order of hundreds of digits, they are presumably secure against classical eavesdroppers.

Shor's algorithm provides a quantum efficient way to break the RSA cryptosystem. In fact, Shor presented a quantum algorithm not for factoring, but for a different problem:

> **Order modulo N:**
> **Input:** An integer N, and Y coprime to N
> **Output:** The order of Y, i.e. the minimal positive integer r such that $Y^r = 1 \ mod \ N$.

The problem of factorization can be polynomially reduced to the problem of finding the order modulo N, using results from number theory. I will not describe the reduction here; an explanation can be found in an excellent review on Shor's algorithm [90]). Instead, I will show a way[70] to crack RSA given an efficient algorithm to find the order modulo N: Suppose the message sent is M^E. Find the order r of M^E modulo N, r is also the order of M, since E is coprime to $(P-1)(Q-1) = \phi(N)$. It is easy to find efficiently the inverse of E, $D' = E^{-1}$ modulo r, using Euclid's algorithm. Then simply, $(M^E)^{D'} \equiv M \ mod \ N$, since $M^r \equiv \ 1 \ mod \ N$.

Let me now present Shor's beautiful algorithm for finding the order of Y, for any given Y, modulo N. The description follows[90]. In short, the idea of the algorithm is to create a state with periodicity r, and then apply Fourier transform over Z_Q, (the additive group of integers modulo Q), to reveal this periodicity. The Fourier transform over the group

Z_Q is defined as follows:

$$|a\rangle \longmapsto \frac{1}{\sqrt{Q}} \sum_{b=0}^{Q-1} e^{2\pi i ab/Q} |b\rangle = |\Psi_{Q,a}\rangle \tag{24}$$

The algorithm to compute this Fourier transform will be given in the next section, which is devoted entirely to Fourier transforms. Again we will work with two registers. The first will hold a number between 1 to Q. (Q will be fixed later: it is much larger than N, but still polynomial in N.) The second register will carry numbers between 1 to N. Hence the two registers will consist of $O(\log(\mathrm{N}))$ qubits

Shor's Algorithm

$|\vec{0}\rangle \otimes |\vec{0}\rangle$

Apply Fourier Transform over Z_Q on the first register

$$\Downarrow$$

$\frac{1}{\sqrt{Q}} \sum_{l=0}^{Q-1} |l\rangle \otimes |\vec{0}\rangle$

Call subroutine which computes $|l\rangle|d\rangle \longmapsto |l\rangle|d \oplus Y^l \ mod \ N\rangle$

$$\Downarrow$$

$\frac{1}{\sqrt{Q}} \sum_{l=0}^{Q-1} |l\rangle \otimes |Y^l mod N\rangle$

Measure second register.

$$\Downarrow$$

$\frac{1}{\sqrt{A}} \sum_{l=0|Y^l=Y^{l_0}}^{Q-1} |l\rangle \otimes |Y^{l_0}\rangle = \frac{1}{\sqrt{A}} \sum_{j=0}^{A-1} |jr + l_0\rangle \otimes |Y^{l_0}\rangle$

Apply Fourier Transform over Z_Q on the first register

$$\Downarrow$$

$\frac{1}{\sqrt{Q}} \sum_{k=0}^{Q-1} \left(\frac{1}{\sqrt{A}} \sum_{j=0}^{A-1} e^{2\pi i (jr+l_0)k/Q} \right) |k\rangle \otimes |Y^{l_0}\rangle$

$$\Downarrow$$

Measure first register. Let k_1 be the outcome.
Approximate the fraction $\frac{k_1}{Q}$ by a fraction with denominator smaller than N, using the (classical) method of continued fractions.
If the denominator d doesn't satisfy $Y^d = 1 mod \ N$, throw it away.
Else call the denominator r_1.

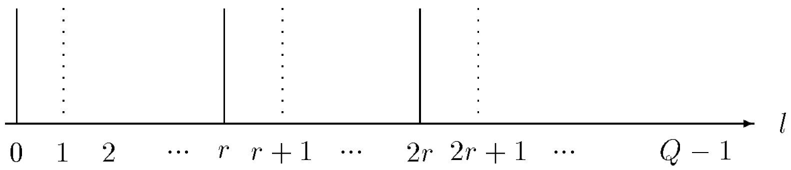
$$\Downarrow$$

Repeat all previous steps poly(log(N)) times to get r_1, r_2,..
Output the minimal r.

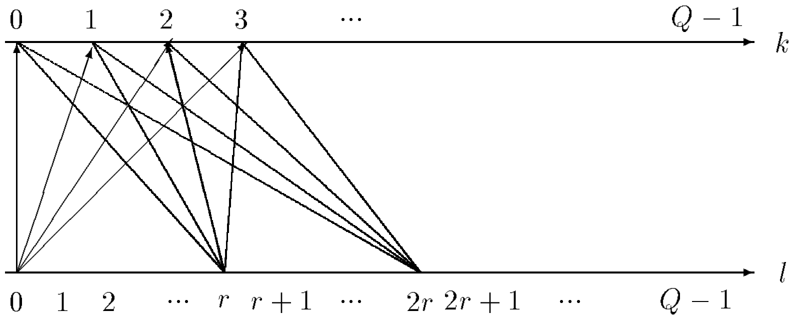
Let us now understand how this algorithm works. In the second step of the algorithm, all numbers between 0 and $Q-1$ are present in the superposition, with equal weights. In the third step of the algorithm, they are separated to sets, each has periodicity r. This is done as follows: there are r possible values written on the second register: $a \in \{Y^0, Y^1, ...Y^{r-1}\}$. The third state can thus be written as:

$$\frac{1}{\sqrt{Q}}\left((\sum_{l=0|Y^l=Y}^{Q-1} |l\rangle \otimes |Y\rangle) + (\sum_{l=0|Y^l=Y^2}^{Q-1} |l\rangle \otimes |Y^2\rangle) + ... + (\sum_{l=0|Y^l=Y^r}^{Q-1} |l\rangle \otimes |Y^r = 1\rangle) \right)$$

Note that the values l that give $Y^l = a$ have periodicity r: If the smallest such l is l_0, then $l = l_0 + r, l_0 + 2r, ..$ will also give $Y^l = a$. Hence each term in the brackets has periodicity r. Each set of l's, with periodicity r, is attached to a different state of the second register. Before the computation of Y^l, all l's appeared equally in the superposition. Writing down the Y^l on the second register can be thought of as giving a different "color" to each periodic set in $[0, Q-1]$. Visually, this can be viewed as follows:



The measurement of the second register picks randomly one of these sets, and the state collapses to a superposition of l's with periodicity r, with an arbitrary shift l_0. Now, how to obtain the periodicity? The first idea that comes to mind is to measure the first register twice, in order to get two samples from the same periodic set, and somehow deduce r from these samples. However, the probability that the measurement of the second register yields the same shift in two runs of the algorithm, i.e. that the same periodic set is chosen twice, is exponentially small. How to gain information about the periodicity in the state without simply sampling it? This is done by the Fourier transform. To understand the operation of the Fourier transform, we use a diagram again:



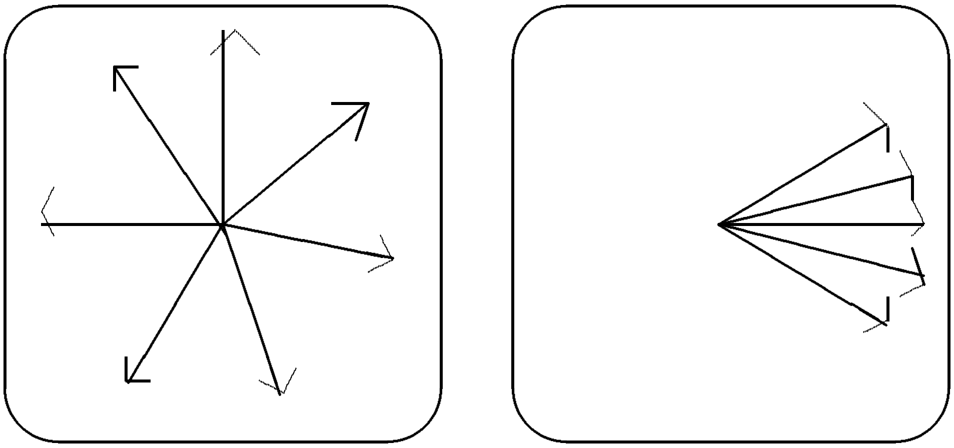
Each edge in the diagram indicates that there is some probability amplitude to transform from the bottom basis state to the upper one. We now measure the first register, to obtain k. To find the probability to measure each k, we need to sum up the weights coming from all the j's in the periodic set.

$$\mathrm{Prob(k)} = |\frac{1}{\sqrt{\mathrm{QA}}}\sum_{\mathrm{j=0}}^{\mathrm{A-1}} \mathrm{e}^{2\pi \mathrm{ik(jr+l_0)/Q}}|^2 = |\frac{1}{\sqrt{\mathrm{QA}}}\sum_{\mathrm{j=0}}^{\mathrm{A-1}} (\mathrm{e}^{2\pi \mathrm{ikr/Q}})^{\mathrm{j}}|^2 \tag{25}$$

Hence, in order to compute the probability to measure each k, we need to evaluate a geometrical series. Alternatively the geometric series is a sum over unit vectors in the complex plane.

Exact periodicity: Let us assume for a second *exact periodicity*, i.e. that r divides Q exactly. Then $A = Q/r$. In this case, the above geometrical series is equal to zero, unless $e^{2\pi ikr/Q} = 1$. Thus we measure with probability 1 only $k's$ such that $kr = 0 \, mod \, Q$. This is where destructive interference comes to play: only "good" k's, which satisfy $kr = 0 \, mod \, Q$, remain, and all the others cancel out. Why are such k's "good"? We can write $kr = mQ$, for some integer m, or $k/Q = m/r$. We know Q, and we know k since we have measured it. Therefore we can reduce the fraction k/Q. If m and r are coprime. the denominator will be exactly r which we are looking for! By the prime number theorem, there are approximately $n/log(n)$ numbers smaller than n and coprime with n, so since m is chosen randomly, repeating the experiment a large enough number of times we will with very high probability eventually get m coprime to r.

Imperfect periodicity: In the general case, r does not divide Q, and this means that the picture is less clear. "Bad" k's do not completely cancel out. We distinguish between two types of k's, for which the geometrical series of vectors in the complex plain looks as follows:



In the left case, all vectors point in different directions, and they tend to cancel each other. This will cause destructive interference, which will cause the amplitude of such $k's$ to be small. In the right case, all vectors point almost to the same direction. In this case there will be constructive interference of all the vectors. This happens when $e^{2\pi ikr/Q}$ is close to one, or when $kr\ mod\ Q$ is close to zero. This means that with high probability, we will measure only $k's$ which satisfy an *approximate* criterion $kr \approx 0\ mod\ Q$. In particular, consider k's which satisfy:

$$-r/2 \leq kr \text{ mod } Q \leq r/2 \tag{26}$$

There are exactly r values of k satisfying this requirement, because k runs from 0 to $Q-1$, therefore kr runs from 0 to $(Q-1)r$, and this set of integers contains exactly r multiples of Q. Note, that for such k's all the complex vectors lie in the upper half of the complex plane, so they are instructively interfering. Now the probability to measure such a k is bounded below, by choosing the largest exponent possible:

$$\begin{aligned}\mathrm{Prob(k)} &= |\frac{1}{\sqrt{\mathrm{QA}}}\sum_{\mathrm{j=0}}^{\mathrm{A-1}}(\mathrm{e}^{2\pi\mathrm{ikr/Q}})^{\mathrm{j}}|^2 \geq |\frac{1}{\sqrt{\mathrm{QA}}}\sum_{\mathrm{j=0}}^{\mathrm{A-1}}(\mathrm{e}^{\mathrm{i}\pi\mathrm{r/Q}})^{\mathrm{j}}|^2\\ &= \frac{1}{QA}|\frac{1-e^{\pi irA/Q}}{1-e^{i\pi r/Q}}|^2 = \frac{1}{QA}|\frac{sin(\frac{\pi rA}{2Q})}{sin(\frac{\pi r}{2Q})}|^2 \approx \frac{4}{\pi^2 r}\end{aligned}$$

Where the approximation is due to the fact that Q is chosen to be much larger than $N > r$, therefore the sinus in the enumerator is close to 1 with negligible correction of the order of r/Q. In the denominator we use the approximation $\sin(\mathrm{x}) \approx \mathrm{x}$ for small x, and the correction is again of the order of r/Q. The probability to measure any k which satisfies 26 is approximately $4/\pi^2$, since there are r such k's.

Why are such k's "good"? Given an integer k which satisfies the criterion 26, we can find r with reasonably high probability. Note that for "good" k's, there exists an integer m such that:

$$|\frac{k}{Q} - \frac{m}{r}| \leq \frac{1}{2Q}.$$

Remember that Q is chosen to be much larger than N, say $Q \geq N^2$. This means that $\frac{k}{Q}$, a fraction with denominator $\geq N^2$, can be approximated by $\frac{m}{r}$, a fraction with denominator smaller than N, to within $\frac{1}{N^2}$ There is only one fraction with such a small denominator that approximates a fraction so well with such large denominator. Given k/Q, the approximating fraction, $\frac{m}{r}$, can be found efficiently, using the method of continued fractions:

$$a = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \ldots}},$$

where a_i are all integers. Finding this fraction, the denominator will be r! Well, not precisely. Again, it might be the case that m and r are not coprime, and the number we find will be the denominator of the reduced fraction of $\frac{m}{r}$. In this case the number will fail the test $Y^r = 1$ which is included in Shor's algorithm, and it will be thrown away. Fortunately, the probability for m to be coprime to r is large enough: it is greater than $1/log(r)$. We repeat the experiment until this happens.

This concludes Shor's algorithm. In the next chapter we will see an alternative algorithm by Kitaev for finding the order modulo N.

7 Fourier Transforms

The ability to efficiently apply Fourier transforms over groups with exponentially many elements is unique to the quantum world. In fact, Fourier transforms are the *only* known tool in quantum computation which gives exponential advantage. For this reason it is worthwhile to devote a whole chapter for Fourier transforms. The Fourier transform is defined as follows. Denote the additive group of integers modulo Q by Z_Q. Let f be a function from the group Z_Q to the complex numbers:

$$f : a \longmapsto f(a) \in C \tag{27}$$

The Fourier transform of this function is another function from Z_Q to the complex numbers:

$$\hat{f} : a \longmapsto \hat{f}(a) = \frac{1}{\sqrt{Q}} \sum_{b \in Z_Q} e^{2\pi iab/Q} f(b) \in C \tag{28}$$

The straight forward way to compute the Q Fourier coefficients of the function, $\hat{f}(a)$ $\forall a$, will take $O(Q^2)$ time. When Q is a factor of 2, there is a way to shorten the trivial Fourier transform algorithm using recursion. This is called fast Fourier transform, or in short FFT, and it enables to compute the Fourier transform within $O(Q\log(\mathrm{Q}))$ time steps[73]. When Q is very large, this still is a very slow operation.

In the quantum world, a function from the Abelian group $G = Z_Q$ to the complex numbers $f : a \longmapsto f(a)$ can be represented by a superposition $|f\rangle = \sum_{a=0}^{Q-1} f(a)|a\rangle$ (perhaps normalized.) The Fourier transform of the function will be $|\hat{f}\rangle = \sum_{a=0}^{Q-1} \hat{f}(a)|a\rangle$. Note that in the quantum setting, the function on Q elements is represented compactly as a superposition on $log(Q)$ qubits. This compact representation allows in some cases to apply the transformation $|f\rangle \longmapsto |\hat{f}\rangle$ very efficiently, in only $O(log(Q))$ time steps. Indeed, measuring all the Fourier coefficients will still take time which is exponential in $\log(Q)$ simply because the number of coefficients is exponential. However, the actual transformation from a superposition to its Fourier transform will be very fast.

In order to apply the Fourier transformation on general states, it suffices to apply the following transformation on the basis states:

$$|a\rangle \longmapsto |\Psi_{Q,a}\rangle = \frac{1}{\sqrt{Q}} \sum_{b=0}^{Q-1} e^{2\pi iab/Q} |b\rangle. \tag{29}$$

We will first consider the special case of $Q = 2^m$, which is simpler than the general case, since classical techniques for fast Fourier transforms can be adopted[172, 51, 72, 80, 109] I will give here a nice description by Cleve *et. al.*[70]. Later I'll describe Kitaev's[123] more general quantum Fourier transform, for any Abelian group, which implies a beautiful alternative factorization algorithm.

Quantum fast Fourier transform. Let $Q = 2^m$. An integer $a \in \{0, 1, ..., 2^m - 1\}$ is represented in binary representation by $|a_1...a_m\rangle$, so $a = a_1 2^{m-1} + a_2 2^{m-2} + ... + a_{m-1} 2^1 + a_m$. Interestingly, the Fourier state in this case is not entangled, and can be written as a tensor product:

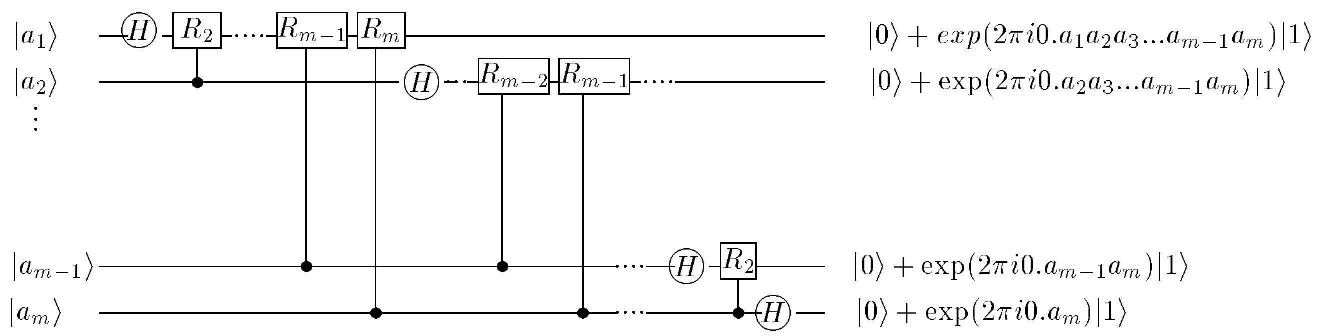
$$|\Psi_{Q,a}\rangle = \frac{1}{\sqrt{Q}} \sum_{b=0}^{Q-1} e^{2\pi i ab/Q} |b\rangle = \frac{1}{\sqrt{2^m}} (|0\rangle + e^{2\pi i 0.a_m}|1\rangle)(|0\rangle + e^{2\pi i 0.a_{m-1}a_m}|1\rangle)...(|0\rangle + e^{2\pi i 0.a_1...a_{m-1}a_m}|1\rangle) \tag{30}$$

We can see this by computing the coefficient of b in this formula. In fact, what matters is that the phases in the coefficient of b from both sides of the equality are equal (modulo 1). To see this, observe that the phase of $|b\rangle$ in the left term is $2^{-m}ab = 2^{-m}\sum_{i,j=1}^{m} a_i 2^{m-i} b_j 2^{m-j}$, which can be seen to be equal modulo 1 to $0.a_m \cdot b_1 + 0.a_{m-1}a_m \cdot b_2 + ... + 0.a_1...a_{m-1}a_m \cdot b_m$ which is the phase of $|b\rangle$ in the right term.

To apply the QFFT, we will need only two gates. The first is the Hadamard gate on one qubit. The second gate is a gate on two qubits, which applies a conditioned phase shift on one qubit, given that the other qubit is in state $|1\rangle$. R_k denotes the phase shift on one qubit by $e^{2\pi i/2^k}$.

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix} \quad , \quad H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \tag{31}$$

We will operate the following gate array:



We claim that this gate array implements the FT, except that the output is in reverse order of bits. To prove this, we show that each bit gains the phase it is supposed to gain, according to equation 30. The first H on the first bit a_1 produces the state on m qubits:

$$(|0\rangle + e^{2\pi i(0.a_1)}|1\rangle)|a_2...a_m\rangle$$

and the next R_2 makes it

$$(|0\rangle + e^{2\pi i(0.a_1a_2)}|1\rangle)|a_2...a_m\rangle,$$

and so on until the first qubit is in the correct state (of the last bit in equation 30):

$$(|0\rangle + e^{2\pi i(0.a_1a_2...a_m)}|1\rangle)|a_2...a_m\rangle.$$

In the same way the phases of the rest of the qubits are fixed, one by one. We now simply reverse the order of the bits to obtain the correct FT.

Note that the number of gates is $m(m-1)/2$ which is $O(log^2(Q))$. In fact, many of these gates can be omitted, because R_k can be exponentially close to one. omitting such gates we still obtain a very good approximation of the Fourier transform[72].

Kitaev's algorithm: Kitaev's algorithm[123] shows how to approximate efficiently the FT over the cyclic group Z_Q for any Q (a cyclic group is a group that is generated by one element). The generalization to any Abelian group is simple[123], but will not be described here. The sequence of operation is the following:

Fourier Transform a la Kitaev

$$|a\rangle \otimes |0\rangle \implies |a\rangle \otimes |\Psi_{Q,0}\rangle \implies |a\rangle \otimes |\Psi_{Q,a}\rangle \implies |0\rangle \otimes |\Psi_{Q,a}\rangle \implies |\Psi_{Q,a}\rangle \otimes |0\rangle$$

The most important and difficult step in this algorithm is the third step. Let us understand how to perform each of the other steps first:

1. $|0\rangle \longmapsto |\Psi_{Q,0}\rangle$ is actually a classical operation. We pick an integer between 1 and Q uniformly at random using a recursive procedure. Let $2^{n-1} < Q < 2^n$. Denote $Q_0 = 2^{n-1}$ and $Q_1 = Q - Q_0$. Apply the one qubit gate $|0\rangle \longmapsto \sqrt{\frac{Q_0}{Q}}|0\rangle + \sqrt{\frac{Q_1}{Q}}|1\rangle$. Now, conditioned on the first bit x, create on the last $n-1$ bits, the state $|\Psi_{Q_x,0}\rangle$ recursively.

2. $|a\rangle \otimes |\Psi_0\rangle \implies |a\rangle \otimes |\Psi_a\rangle$ is achieved by applying $|a,b\rangle \longmapsto e^{2\pi iab/Q}|a,b\rangle$.

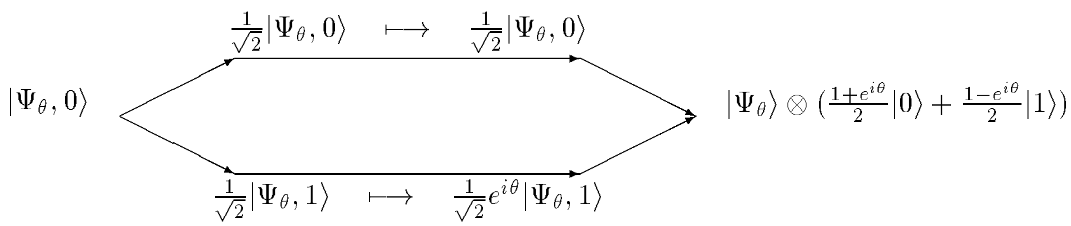
3. The third operation is, perhaps surprisingly, the most difficult part in the FT, and I will sketch the idea next.

4. The last operation is merely swapping the bits.

To apply the third step, we note that the vectors $|\Psi_{Q,a}\rangle$ are eigenvectors of the unitary operation $U: |g^m\rangle \longmapsto |g^{m+1}\rangle$, where g is the generator of the cyclic group, with eigenvalues $e^{-2\pi i a/Q}$ The operation $|a\rangle \otimes |\Psi_{Q,a}\rangle \implies |0\rangle \otimes |\Psi_{Q,a}\rangle$ is actually the reverse of computing the eigenvalue of an eigenvector. We need to be able to write down the eigenvalues of a given unitary matrix. Kitaev has proved the following lemma:

Lemma 2 *(Kitaev) Let U be a unitary matrix on n qubits such that $U, U^2, U^4 ... U^{2^n}$ can be applied efficiently. Let $|\Psi_\theta\rangle$ be U's eigenvectors with corresponding eigenvalues $e^{i\theta}$. Then the transformation $|\Psi_\theta\rangle \otimes |0\rangle \implies |\Psi_\theta\rangle \otimes |\theta\rangle$ can be approximated to exponential accuracy, efficiently.*

Proof: The idea that lies behind this theorem is *interference.* The eigenvalues are phases, and in order to gain information about a phase we need to compare it with some reference phase, just like what happens in an interferometer. The implementation of this idea in the setting of qubits is done by adding a control qubit. We proceed as follows. We apply the Hadamard transform H on the control qubit, which separates the state to two paths, one in which the control qubit is in state $|1\rangle$ and the other in which it is $|0\rangle$. Now U is applied on $|\Psi_\theta\rangle$, *conditioned* that the control qubit is 1. This adds a phase $e^{i\theta}$ on one of the paths, which can be compared to the reference path. Finally, the controlled qubit is rotated again by a Hadamard transform. The following diagram captures the idea schematically:



The control qubit is now in a state $|\beta\rangle = (\frac{1+e^{i\theta}}{2}|0\rangle + \frac{1-e^{i\theta}}{2}|1\rangle)$, which is a qubit biased according to the eigenvalue. If we measure this qubit, it behaves like a coin flip with bias $p = |1 - e^{i\theta}|^2/4 = \frac{1-\cos\theta}{2}$.

The idea is to create many control qubits, and measure all of them. This is like performing many independent coin tosses. We can deduce θ from the ratio between the number of times we got 1 and the number of times we got 0. For this, we will apply a classical algorithm on the outcomes of the measurements. However, there are two problems with this idea. One is that the outcome of the algorithm will be classical, while we want to create a unitary transformation which writes down the eigenvalues and can be applied on superpositions. We will deal with this problem later. A more severe problem is that the algorithm should find θ with exponential accuracy (polynomially many bits), since there are exponentially many eigenvalues. To achieve exponential accuracy in θ we need exponentially many coin tosses; By Chernoff's inequality[73], exponentially many coin tosses are required in order to achieve exponential accuracy in θ. Since we are limited to polynomial algorithms, we can only deduce θ with polynomial accuracy. The solution to this problem takes advantage of the fact that the powers of U can be applied efficiently. To deduce θ to higher accuracy, we slightly modify the interference scheme: instead of U, we apply U^2. This will generate another set of biased qubits, from which we can deduce 2θ with polynomial accuracy. The same thing can be done using $U^4, ..., U^{2^n}$, and this will generate n sets of $m = poly(n)$ biased qubits. From the outcomes of the measurements of the j'th set, we compute $2^j\theta$ with polynomial accuracy. It is easy to construct a polynomial classical algorithm that computes θ with exponential precision (which is what we need) from the polynomial approximations of θ,2θ,4θ,... $2^n\theta$.

It is left to show how the above computation can be made unitary. The idea is that it is not necessary to measure each set of qubits, in order to count the number of $1'$s. Instead of measuring these bits, we will apply a unitary transformation that counts the portion of 1's out of m and writes this portion down on a *counting* register. If we denote by $w(i)$ the number of $1'$s in a string i, or the *weight* of the string, then this transformation will be:

$$|i\rangle|0\rangle \longmapsto |i\rangle|w(i)/m\rangle. \tag{32}$$

The resulting state will look something like:

$$|\Psi\rangle \otimes \sum_i \sqrt{p^w(i)(1-p)^{m-w(i)}}|i\rangle|w(i)\rangle \tag{33}$$

with perhaps extra phases. Most of the weight in this state is concentrated on strings with approximately pm $1'$s, like in a Bernoulli experiment. For each set of control qubits, we obtain some portion, written on the counting register of that set. We denote the n portions by w_θ, $w_{2\theta}...w_{2^n\theta}$. We can now apply the unitary version of the classical algorithm which computes an exponentially close approximation of θ given the portions w. If we call this procedure T, we have:

$$|w_\theta\rangle|w_{2\theta}\rangle\cdots|w_{2^n\theta}\rangle|0\rangle \xrightarrow{T} |w_\theta\rangle|w_{2\theta}\rangle\cdots|w_{2^n\theta}\rangle|\theta\rangle \tag{34}$$

We now have θ written down on the last register. Let us denote by Q' the unitary operation which the algorithm applies so far. It is tempting to think that Q' is the desired transformation, $|\Psi_\theta\rangle\otimes|0\rangle \implies |\Psi_\theta\rangle\otimes|\theta\rangle$. This is not true. Actually, Q' is exponentially close to $|\Psi_\theta\rangle\otimes|0\rangle\otimes|0\rangle \stackrel{Q'}{\Longrightarrow} |\Psi_\theta\rangle\otimes|\theta\rangle\otimes|garbage_\theta\rangle$,

where the last register consists of all the control qubits and ancilla qubits which we have used during the computation. The reason for the fact that Q' is not exactly Q, is that in the classical coin tossing, there is an exponentially small probability to get result which is very far from the expected number of $1'$s, mp. This translates in equation 33 to the appearance, with exponentially small weight, of strings i which are very far from the expected number of $1'$s mp. We now want to ask, why do the garbage qubits matter.

These qubits carry information which is no longer needed, but never the less are entangled with the rest of the computer. The point is that their existence might prevent interference in future computation. We will develop tools to think about interference in section 9, but roughly, garbage has the same effect as interaction with the environment, which is known to cause decoherence. How to get rid of the garbage? The problem is that we cannot simply erase the garbage by setting all the garbage qubits to $|0\rangle$, because the transformation that takes a general state to $|0\rangle$ is not unitary. Fortunately, in our case there is a unitary transformation that erases the garbage. We do the following: We copy θ, which is written on the last register, on an extra register which is initialized in the state $|0\rangle$. The copying is done bit by bit, using polynomially many $CNOT$ gates. We now apply in reverse order the reverse of all transformations done so far in the algorithm, except for the $CNOT$ gates. The overall transformation is exponentially close to the following sequence of operations: apply Q, then copy θ and then apply Q^{-1}. This sequence of operation indeed achieves the desired transformation without garbage:

$$|\Psi_\theta\rangle \otimes |0\rangle \otimes |0\rangle \otimes |0\rangle \quad \stackrel{Q}{\Longrightarrow} \quad |\Psi_\theta\rangle \otimes |\theta\rangle \otimes |garbage_\theta\rangle \otimes |0\rangle \quad \stackrel{CNOT\ gates}{\Longrightarrow}$$
$$|\Psi_\theta\rangle \otimes |\theta\rangle \otimes |garbage_\theta\rangle \otimes |\theta\rangle \quad \stackrel{Q^{-1}}{\Longrightarrow} \quad |\Psi_\theta\rangle \otimes |0\rangle \otimes |0\rangle \otimes |\theta\rangle.$$

One can save many qubits by erasing garbage in the middle of the computation, when it is no longer needed, and using these erased qubits as register in the rest of the computation. A different proof of this lemma can be found in [70], where QFFT over Z_2^n is used. □

This concludes the Fourier transform algorithm. Kitaev's procedure of writing the eigenvalue down implies a very simple alternative factorization algorithm. The way an integer N is factorized is done again by finding the order of a number Y which is coprime to N. (Recall that the order of Y is the least r such that $Y^r = 1\ mod\ N$.) Consider the unitary transformation $U : |g\rangle \longmapsto |gY\ mod\ N\rangle$. The eigenvectors of U, $\{|\Psi\rangle\}$, are exactly the linear superpositions of all configurations in the subgroup $\{Y, Y^2, Y^3, ...Y^r\}$, or any coset of this subgroup, $\{gY, gY^2, gY^3, ...gY^r\}$, with appropriate phases:

$$U|\Psi_a\rangle = U(\sum_j e^{2\pi ija/r}|gY^j\rangle) = e^{-2\pi ia/r}(\sum_j e^{2\pi ija/r}|gY^j\rangle).$$

The eigenvalues of U hold information about r! The idea would be to apply Kitaev's lemma, write down $\theta = 2\pi a/r$ and deduce r from it.

We start with the basis state $|0\rangle$, which can be written as an equal superposition of all eigenvectors: $|0\rangle = \sum_a |\Psi_a\rangle$, as you can easily check. Applying Kitaev's lemma on the state $|0\rangle$ we get on the second register all eigenvalues written with uniform probability. We now measure this register, which carries an exponentially close approximation of $2\pi a/r$. We divide by 2π to get c, an exponentially good approximation of a/r. Now, using the method of continued fraction, like in Shor's algorithm, we find the closest fraction to c with denominator less than N. With high enough probability a and r are coprime, so we get r in the denominator. If not, the denominator does not satisfy $Y^r = 1 \; modN$, and we repeat the experiment again. Here is a summary of the algorithm:

Factorization a la Kitaev

$\sum_a |\Psi_a\rangle|0^n\rangle$
Apply Kitaev's transformation $|\Psi_a\rangle|0\rangle \longmapsto |\Psi_a\rangle|2\pi a/r\rangle$

$$\Downarrow$$

$\sum_a |\Psi_a\rangle|2\pi a/r\rangle$
Measure the second register. Classically compute r from the outcome.

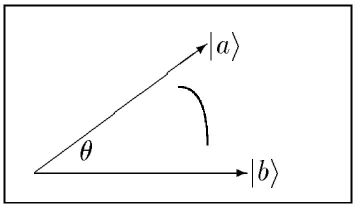
Factorization can be viewed as finding the order of elements in Abelian groups. Many people tried to generalize Shor's and Kitaev's algorithms to non-Abelian groups. It is conjectured that Fourier transforms over non-Abelian groups would be helpful tools, however they are much more complicated operations since the Fourier coefficients are complex *matrices*, and not complex numbers! Beals[22] made the first (and only) step in this direction by discovering an efficient quantum Fourier transform algorithm for the non-Abelian permutations group, S_n, building on the classical FFT over S_n[82, 67]. Beals was motivated by an old hard problem in computer science: Given two graphs, can we say whether they are isomorphic (i.e one is simply a permutation of the other) or not. This problem is not known to be $NP-$ complete, but the best known algorithm is exponential. It is still not known whether Beals' Fourier transform can be used for solving graph isomorphism. A very interesting open question is whether efficient quantum Fourier transforms can be done over any group, and can they be used to solve other problems.

8 Grover's Algorithm for Finding a Needle in a Haystack

Grover's algorithm is surprising and counter intuitive at first sight, though it achieves only a polynomial (quadratic) improvement over classical algorithms. It deals with the *database search problem.* Suppose you have access to an unsorted database of size N. You are looking for an item i which satisfies some property. It is easy to check whether the property is satisfied or not. How long will it take you to find such an item, if it exists? If you are using classical computation, obviously it can take you N steps. If you are using probabilistic classical computation, you can reduce it to $N/2$ expected steps. But if you are using a quantum computer, you can find the item in $O(\sqrt{N})$ steps! I will present here the algorithm which was found by Grover[110] in 1995. However, I will use here a different representation of the algorithm, which is mainly based on the geometrical interpretation by Boyer *et.al.* [44, 45].

The algorithm works as follows. Set log(N) = n, and let us define a function f : $\{0,1\}^n \longmapsto \{0,1\}$ where $f(i) = 0$ if the $i'th$ item does not satisfy the desired property, and $f(i) = 1$ in the case it does. Let t be the number of items such that $f(i) = 1$. For the moment, we assume that $t = 1$. The algorithm operates in the Hilbert space of n qubits. Its main part actually works in a subspace of dimension 2 of this space. This subspace is the one which is spanned by the two vectors:

$$|a\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{2^n-1} |i\rangle \quad , \quad |b\rangle = \frac{1}{\sqrt{N-1}} \sum_{i=0|f(i)=0}^{2^n-1} |i\rangle. \tag{35}$$



We begin by applying a FT on $|0\rangle$ which generates the uniform vector $|a\rangle$, using n Hadamard gates. We now want to rotate the vector in the two dimensional subspace spanned by $|a\rangle$ and $|b\rangle$, so that eventually we have large projection on the direction orthogonal to $|b\rangle$, which is exactly the item we want. The idea is that a rotation by the angle 2θ, is equivalent to two *reflections*, first with respect to $|a\rangle$, and then with respect to $|b\rangle$. We define a Boolean function $g(i)$ to be 0 only for $i = 0$, and 1 for the rest. A reflection around $|0\rangle$ is obtained by $R_0 : |i\rangle \longmapsto (-1)^{g(i)}|i\rangle$. A reflection around $|a\rangle$ is achieved by: $R_a = FT \circ R_0 \circ FT$. To reflect around $|b\rangle$, apply the transformation: $R_b : |i\rangle \longmapsto (-1)^{f(i)}|i\rangle$. A rotation by an angle 2θ is achieved by applying R_aR_b.

Grover's algorithm

Apply Fourier transform on $|0\rangle$ to get $|a\rangle$.
Apply R_aR_b $\sqrt{N}\pi/4$ times.
Measure all bits.

The crucial point is that θ satisfies $\cos(\theta) = \sqrt{\frac{\mathrm{N-1}}{\mathrm{N}}}$ so for large N, we have

$$\theta \approx \sin(\theta) = \frac{1}{\sqrt{\mathrm{N}}} \tag{36}$$

Therefore after $O(\sqrt{N})$ rotations, with high probability the measurement yields an item satisfying $f(i) = 1$. Note that this algorithm relies heavily on the assumption that the number of "good" items is one. If for example the number of "good" items is $t = 2$, we will have almost 0 probability to measure a "good" item, exactly when we expect this probability to be almost one! There are several ways to generalize this algorithm to the general case where the number of "good" items is not known. One is a known classical reduction[192]. Another generalization was suggested in [44]. This suggestion not only finds a "good" item regardless of what the number, t, of "good" items is, but also gives a good estimation of t. The idea is that the probability to measure a "good" item is a periodic function in the number of Grover's iteration, where this period depends on t in a well defined way. The period can be found using ideas similar to what is used in Shor's algorithm, by Fourier transforms. Grover's algorithm can be used to solve NP complete problems in time $\sqrt{2^n}$, instead of the classical 2^n, which simply goes over all the 2^n items in the database.

Grover's algorithm provides a quadratic advantage over any possible classical algorithm, which is optimal, due to Bennett *et.al.*[36, 44, 204], a result which I will discuss when dealing with quantum lower bounds in section 10. Let me now describe several variants on Grover's algorithm, all using Grover's iteration as the basic step. (These variants and others can be found in Refs. [47, 111, 87, 112, 48, 44] and [113].)

Estimating the median to a precision ϵ.[113, 111]

> f is a function from $\{1,..N\}$ to $\{1,..N\}$ where N is extremely large. We are given $\epsilon > 0$, We want to find the median M, where we allow a deviation by ϵ, i.e. the number of items smaller than M should be between $\frac{(1\pm\epsilon)N}{2}$. We also allow an exponentially small (in $1/\epsilon$) probability for an error.

We assume that N is very large, and so only polylog(N) operations are considered feasible. Classically, this means that the Median cannot be computed exactly but only estimated probabilistically. A classical probabilistic algorithm cannot do better than sample random elements $f(i)$, and compute their median. An error would occur if more than half the elements are chosen from the last $\frac{1+\epsilon}{2}$ items, or from the first $\frac{1-\epsilon}{2}$ items. For these events to have exponentially small probability, we need $O(\frac{1}{\epsilon^2})$ samples, by Chernoff's law[73]. The following quantum algorithm performs the task in $O(\frac{1}{\epsilon})$ steps.

The idea is to find M by binary search, starting with some value, M_0, as a guess. We will estimate up to precision ϵ, the number $|\eta|$ such that $(1+\eta)N/2$ items satisfy $f(i) > M_0$, This will take us $O(\frac{1}{\epsilon})$ steps. We can now continue the binary search of M, according to the η which we have found. Note that since we do not have information about the sign of η, a simple binary search will not do, but a slight modification will. Each step reduces the possible range of M by a factor of half, and thus the search will take polylog$(N)O(\frac{1}{\epsilon})$ steps. It is therefore enough to estimate $|\eta|$ in $O(\frac{1}{\epsilon})$ steps, given a guess for the median, M_0. Here is how it is done.

We define $f_0(i) = 1$ if $f(i) > M_0$, and $f_0(i) = 0$ if $f(i) \leq M_0$. Our basic iteration will be a rotation in the subspace spanned by two vectors:

$$|\alpha\rangle = \frac{1}{\sqrt{N}}\sum_{i=0}^{2^n-1}|i\rangle, \quad |\beta\rangle = \frac{1}{\sqrt{N}}\sum_{i=0}^{2^n}(-1)^{f_0(i)}|i\rangle \tag{37}$$

Let $|\gamma\rangle$ be a vector orthogonal to $|\beta\rangle$ in the two dimensional subspace. The angle between $|\alpha\rangle$ and $|\gamma\rangle$, is $\theta \approx \sin(\theta) = \eta$. Rotation by 2θ can be done like in Grover's algorithm. We start with $|\alpha\rangle$ and rotate by 2θ $\frac{1}{2\epsilon}$ times. The angle between our vector and $|\alpha\rangle$ is η/ϵ. We can now project on $|\alpha\rangle$ (by rotating $|\alpha\rangle$ to $|0\rangle$ and projecting on $|0\rangle$). The result is distributed like a coin flip with bias $\cos^2(\eta/\epsilon)$. We can repeat this experiment poly$(\frac{1}{\epsilon})$ number of times. This will allow us to estimate the bias $\cos^2(\eta/\epsilon)$ and from it $|\eta|/\epsilon$, up to a $1/4$, with exponentially small error probability. Thus we can estimate $|\eta|$ up to $\epsilon/4$ in $O(\frac{1}{\epsilon})$ time.

Estimating the mean to a precision ϵ.

> f is a function from $\{1,..N\}$ to $[-0.5, 0.5]$, where N is assumed to be very large. We are given $\epsilon > 0$, We want to estimate the mean M up to a precision ϵ.

Again, classically, this will take $O(\frac{1}{\epsilon^2})$, assuming that N is extremely large. Grover suggested a quantum algorithm to solve this problem in $O(\frac{1}{\epsilon})$ steps[111]. Instead of showing Grover's version, I will show a simple classical reduction[199] which allows solving the mean estimation problem given the median algorithm. The idea is that for Boolean functions the mean and median problems coincide. We write the real number $f(i)$, which is between -0.5 to 0.5 in its binary representation: $f(i) = 0.f_1(i)f_2(i)f_3(i).....$ up to $\log(\frac{2}{\epsilon})$ digits, where $f_j(i)$ is the j'th bit of $f(i)$. Hence, $f_j(i)$ are Boolean functions. We can denote by M_j the mean of f_j, which can be estimated by the median algorithm. The mean of f can be computed from $\frac{1}{N}\sum_i f(i) = \sum_j 2^{-j}(\frac{1}{N}\sum_i f_j(i)) = \sum_j 2^{-j}M_j$. Cutting the number of digits causes at most $\frac{\epsilon}{2}$ error in M. Each M_j will be estimated to precision $\epsilon/2$, and this will cause $\frac{\epsilon}{2}$ additional error all together.

Finding the minimum

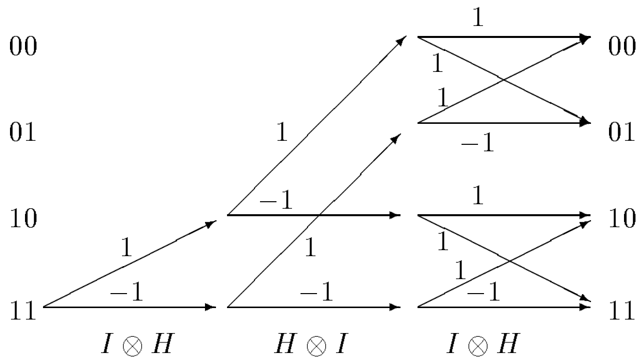
> f is a function from $\{1,..N\}$ to $\{1,..N\}$. We want to find i such that $f(i)$ is minimal.

Classically, this will take $O(N)$, if the database is not sorted. Durr and Hoyer[87] show a quantum algorithm which finds the minimum in $O(\sqrt{N})$. This is done by a binary search of the minimum: At each step j , we have a threshold θ_j. This defines a function: $f_j(i) = 1$ if $f(i) < \theta_j$, and $f_j(i) = 0$ otherwise. θ_0 is fixed to be $N/2$, i.e. in the middle of the interval $[1, ...N]$. Then we apply Grover's search, to find an i such that $f_0(i) = 1$. If we find such an i, we fix the new threshold, θ_1 to be $f(i)$. Else, we fix $\theta_1 = 3N/4$, i.e. in the middle of the interval $[N/2, ...N]$. We continue this binary search until the current interval has shrunk to the size of one number. This is the minimum.

Grover's iteration can be used to achieve a quadratic gap also between quantum and classical communication complexity[52], an issue which is beyond of the scope of this review.

9 What Gives Quantum Computers their (Possible) Extra Power

Let us ask ourselves why quantum computers can perform tasks which seem hard or impossible to do efficiently by classical machines. This is a delicate question which is still an issue of debate. One way to look at this question is using Feynman's path integrals. We will associate a diagram with a computation, in which the vertical axis will run over all 2^n possible classical configurations, and the horizontal axis will be time. Here is an example of such a diagram:



In this diagram, the state is initially $|11\rangle$. The operation H is applied thrice: First on the first bit, then on the second bit and then again on the first bit. The numbers near the edges indicate the probability amplitude to transform between configurations weights: -1 corresponds to $-\frac{1}{\sqrt{2}}$ and 1 corresponds to $\frac{1}{\sqrt{2}}$. Let us now compute the weight of each basis state in the final superposition. This weight is the sum of the weights of all paths leading from the initial configuration to the final one, where the weight of each path is the product of the weights on the edges of the path.

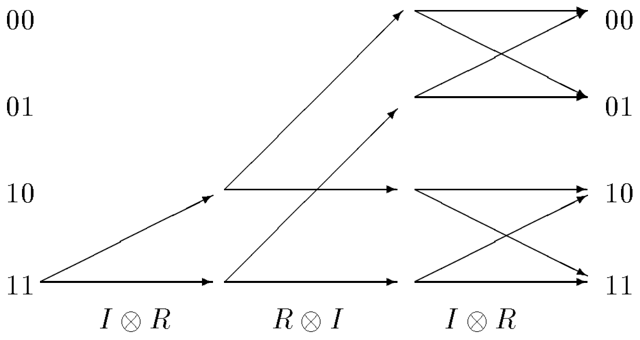
$$\text{Quantum}: \qquad \text{Prob(j)} = |\sum_{\text{d:i}\mapsto\text{j}} \text{w(d)}|^2 \tag{38}$$

One can see that in the above diagram the weights of 10 and 00 in the final superposition are zero, because the two paths leading to each one of these states cancel one another.

What can we learn from this diagram? In order to analyze this diagram, I would like to define a classical computation model, called *stochastic circuits* which can be associated with very similar diagrams. The comparison between the two models is quite instructive. The nodes in a stochastic circuit have an equal number of inputs and outputs, like nodes in a quantum circuit. Instead of unitary matrices, the nodes will be associated with stochastic matrices, which means that the entries of the matrices are positive reals, and the columns are probability distributions. Such matrices correspond to applying stochastic transformations on the bits, i.e. a string i transforms to string j with the probability which is equal to the matrix entry $R_{i,j}$. For example, let R be the stochastic matrix on one bit:

$$R = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} \tag{39}$$

This matrix takes any input to a uniformly random bit. Consider the probabilistic computation on two bits, where we apply R on the first bit, then on the second bit, and then again on the first bit. The diagram we get is:



where the weights of all edges are $\frac{1}{2}$. Just like in quantum computation, the probability for a configuration in the final state is computed by summing over the weights of all paths leading to that configuration, where the weight of each path is the product of the weights of the edges participating in the path.

$$\text{Stochastic}: \qquad \mathrm{Prob(j)} = \sum_{\mathrm{d:i \mapsto j}} \mathrm{Prob(d)} \tag{40}$$

In this diagram all the configurations in the final state have probability $\frac{1}{4}$.

We now have two models which are very similar. It can be easily seen that stochastic circuits are equivalent to probabilistic TM. This means that we can find the advantage of quantum computation over classical computation in the difference between quantum circuits and stochastic circuits. It is sometimes tempting to say that quantum computation is powerful because it has exponential parallelism. For n particles, the vertical axis will run over 2^n possible classical states. But this will also be true in the diagram of stochastic computation on n bits! The difference between quantum and classical computations is therefore more subtle.

To reduce the difference between the two models even further, it can be shown[38] that the complex numbers in quantum computation can be replaced with real numbers, without damaging the computational power. This is done by adding an extra qubit to the entire circuit, which will carry the information of whether we are working in the real or imaginary part of the numbers. The correspondence between the superpositions of the complex circuit to the real circuit will be:

$$\sum_i c_i|i\rangle \longmapsto \sum_i Re(c_i)|i,0\rangle + Im(c_i)|i,1\rangle \tag{41}$$

Hence quantum computers maintain their computational power even if they use only real valued unitary gates. There are two differences between these gates and stochastic gates. One is that stochastic gates have positive entries while real unitary gates have positive and negative entries. The other difference is that unitary gates preserve the L_2 norm of vectors, while stochastic gates preserve L_1 norm. The difference between the quantum and classical models can therefore be summarized in the following table:

Quantum	*Stochastic*
Negative + Positive	*Positive*
L_2 *Norm*	L_1 *Norm*

Why are negative numbers so important? The fact that weights can be negative allows different paths to cancel each other. We can have many non-zero paths leading to the same final configuration, all cancelling each other, causing destructive interference. This is exactly what happens in Deutsch and Jozsa's algorithm, Simon's algorithm and Shor's algorithm, where the paths that lead to "bad" strings in the last step of the algorithm are destructively interfering, and at the same time paths that lead to "good" strings are constructively interfering. In the probabilistic case, interference cannot occur. Paths do not *talk* to each other, there is no influence of one path on the other. Probabilistic computation has the power of exponentiality, but lacks the power of interference offered by computation that uses negative numbers. An exponential advantage in computational power of negative numbers is already familiar from classical complexity theory, when comparing Boolean circuits with monotone Boolean circuits[191].

There are other computational models which exhibit interference, such as optical computers. However, these models do not exhibit exponentiality. It is only the quantum model which combines the two features of exponential space which can be explored in polynomial time, together with the ability of interference. (See also [9].)

Another point of view of the origin of the power of quantum computation is quantum correlations, or *entanglement*. Two qubits are said to be entangled if their state is not in tensor product, for example the EPR pair $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. In a system of n qubits, the entanglement can be spread over macroscopic range, like in the state $\frac{1}{\sqrt{2}}(|0^n\rangle + |1^n\rangle)$, or it can be concentrated between pairs of particles like in the state $\bigotimes_{n/2} \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. It can be shown that quantum computational power exists only when the entanglement is spread over macroscopically many particles. If the entanglement is not macroscopically spread, the system can be easily simulated by a classical computer[6]. For the importance of entanglement see for example Jozsa's review[118]. This macroscopic spread of entanglement lies in the essence of another important topic, quantum error correcting codes, which we will encounter later.

10 What We Cannot Do with Quantum Computers

Now that we have all this repertoire of algorithms in our hands, it is tempting to try and solve everything on a quantum computer! Before doing that, it is worthwhile to understand the limitations of this model. The first thing to know is that this model cannot solve any question which is undecidable by a classical machine. This is simply due to the fact that anything that can be done in this model can be simulated on a classical machine by computing the coefficients of the superposition and writing them down. This will take an exponential amount of time, but finally will solve anything which can be done quantumly. Therefore the only difference between classical and quantum computation lies in the computational cost.

The trivial simulation of quantum computers by classical machines is exponential both in time and space. Bernstein and Vazirani[38] showed that classical Turing machines can simulate quantum computers in polynomial space, although still in exponential time:

Theorem 6 *(Bernstein, Vazirani) $BQP \subseteq Pspace$*

The theorem means that anything that can be done on a quantum machine can be done by a classical machine which uses only polynomial space. To prove this result, have another look on the Feynman path graph presented in Sec. 9. To compute the weight of one path, we need only polynomial space. We can run over all paths leading to the same configuration, computing the weight one by one, and adding them up. This will give the probability of one configuration. To compute the probability to measure 0, we add the probabilities of all the configurations with the result bit being 0. This again will take exponential time, but only polynomial space. ∎

Valiant improved this result[38] to show that BQP is contained in a complexity class which is weaker than $Pspace$, namely $P^{\#P}$, which I will not define here. It might still be that quantum computation is much less powerful, but we still do not have a proof for that. In particular, the relation between BQP and NP is not known yet.

We do understand a lot about the following question:

> Can quantum computation be much more efficient than classical computation in terms of number of accesses to the input?

Consider accessing the n input bits $X_1, ...X_n$, for a problem or a function via an oracle, i.e. by applying the unitary transformation:

$$|i\rangle|0\rangle \longmapsto |i\rangle|X_i\rangle \tag{42}$$

This unitary transformation corresponds to the classical operation of asking: "what is the i'th bit? " and getting the answer X_i. One might hope to make use of quantum parallelism, and query the oracle by the superposition $1/\sqrt{N}\sum_i |i\rangle|0\rangle \longmapsto 1/\sqrt{N}\sum_i |i\rangle|X_i\rangle$. In one query to the oracle, the algorithm can read all the N bits, so intuitively no quantum algorithm needs more than one query to the oracle. It turns out that this intuition is completely wrong. It can be shown, using the notion of von Neumann entropy (see [161]) that there are no more than $log(N)$ bits of information in the state $1/\sqrt{N}\sum_i |i\rangle|X_i\rangle$. Bennett *et.al.*[36] show that if the quantum algorithm is supposed to compute the OR of the oracle bits $X_1, ...X_n$, then at least $O(\sqrt{N})$ queries are needed. Note that OR is exactly the function computed by Grover's database search. Hence this gives a lower bound of $O(\sqrt{N})$ for database search, and shows that Grover's algorithm is optimal.

Theorem 7 *Any quantum algorithm that computes $OR(X_1...X_N)$ requires at least $O(\sqrt{N})$ steps.*

The idea of the proof is that if the number of the queries to the oracle is small, there exists at least one index i, such the algorithm will be almost indifferent to X_i, and so will not distinguish between the case of all bits 0 and the case that all bits are zero except $X_i = 1$. Since the function which the algorithm computes is OR, this is a contradiction.
Beals *et. al.*[23] recently generalized the above result building on classical results by Nisan and Szegedi[154]. Beals *et.al.* compare the minimal number of queries to the oracle which are needed in a quantum algorithm, with the minimal number of queries which are needed in a classical algorithm. Let us denote by $D(f)$ and $Q(f)$ the minimal number of queries in a classical and quantum algorithm respectively. Beals *et.al.*[23] show that $D(f)$ is at most polynomial in $Q(f)$.

Theorem 8 $D(f) = O(Q(f)^6)$

Beals *et. al.* use similar methods to give lower bounds on the time required to quantumly compute the functions MAJORITY, PARITY[92], OR and AND:

OR	$\Theta(\sqrt{N})$
AND	$\Theta(\sqrt{N})$
PARITY	$N/2$
MAJORITY	$\Theta(N)$

(Here $f = \Theta(g)$ means that f and g behave the same asymptotically.) The lower bounds are achieved by showing that the number of times the algorithm is required to access the input is large. This is intuitive, since these functions are very sensitive to their input bits. For example, the string 0^N satisfies $OR(0^N) = 0$, but flipping any bit will give $OR(0^{N-1}1) = 1$.

The meaning of these results, is that in terms of the number of accesses to the input, quantum algorithms have no more than polynomial advantage over classical algorithms[159]. This polynomial relation can give us a hint when looking for computational problems in which quantum algorithms may have an exponential advantage over classical algorithms. These problems will have the property that in a classical algorithm that solves them, the bottle neck is the information processing, while the number of accesses to the input can be very small. Factorization is exactly such a problem. $D(f)$ is $log(N)$, because the algorithm simply needs to read the number N in binary representation, but the classical information processing takes exponential in $log(N)$ steps. Shor's quantum algorithm enables an exponential speed up in the information processing. An opposite example is the database search. Here, the bottle neck in classical computation is not the information processing but simply the fact that the size of the input is very large. Indeed, in this case, quantum computers have only quadratic advantage over classical computers.

Now that we understand some of the limitations and advantages of the quantum model, let us go on to the subject of quantum noise.

11 Worries about Decoherence, Precision and Inaccuracies

Learning about the possibilities which lie in quantum computation gave rise to a lot of enthusiasm, but many physicist[135, 189, 57, 19] were at the same time very sceptic about the entire field. The reason was that all quantum algorithms achieve their advantage over classical algorithms when assuming that the gates and wires operate without any inaccuracies or errors. Unfortunately, in reality we cannot expect any system to be ideal. Quantum systems in particular tend to lose their quantum nature easily. Inaccuracies and errors may cause the damage to accumulate exponentially fast during the time of the computation[57, 58, 17, 19, 149]. In order to perform computations, one must be able to reduce the effects of inaccuracies and errors, and to correct the quantum state.

Let us try to understand the types of errors and inaccuracies that might occur in a quantum computer. The simplest problem is that the gates perform unitary operations which slightly deviate from the correct ones. Indeed, it was shown by Bernstein and Vazirani[38] that it suffices that the entries of the gates are precise only up to $1/n$, where n is the size of the computation. However, it is not reasonable to assume that inaccuracies decrease as $1/n$. What seems to be reasonable to assume is that the devices we will use in the laboratory have some finite precision, independent of the size of the computation. Errors, that might occur, will behave, presumably, according to the same law of constant probability for error per element per time step. Perhaps the most severe problem was that of *decoherence*[151, 184, 205, 156, 100]. Decoherence is the physical process, in which quantum system lose some of their quantum characteristics due to interactions with environment. Such interactions are inevitable because no system can be kept entirely isolated from the environment. The effect of entanglement with the environment can be viewed as if the environment applied a partial measurement on the system, which caused the wave function to collapse, with certain probability. This collapse of the wave function seems to be an irreversible process by definition. How can we correct a wave function which has collapsed?

In order to solve the problem of correcting the effects of noise, we have to give a formal description of the noise process. Observe that the most general quantum operation on a system is a unitary operation on the system and its environment. Noise, inaccuracies, and decoherence can all be described in this form. Formally, the model of noise is that in between the time steps, we will allow a "noise" operator to operate on the system and an environment. We will assume that the environment is renewed each time step, so there are no correlations between the noise processes at different times. Another crucial assumption is that the noise is *local*. This means that each qubit interacts with its own environment during the noise process, and that there are no interactions or correlations between these environments. In other words, the noise operator on n qubits, at each time step, can be written as a tensor product of n local noise operators, each operating on one qubit:

$$\mathcal{E} = \mathcal{E}_1 \otimes \mathcal{E}_2 \otimes \cdots \otimes \mathcal{E}_n.$$

If the qubits were correlated in the last time step by a quantum gate, the local noise operator operates on all the qubits participating in one gate together. This noise model assumes that correlations between errors on different qubits can only appear due to the qubits interacting through a gate. Otherwise, each qubit interacts with its own environment.

The most general noise operator on one qubit is a general unitary transformation on the qubit and its environment:

$$\begin{aligned} |e\rangle|0\rangle &\rightarrow |e_0\rangle|0\rangle + \ |e_0^b\rangle|1\rangle \\ |e\rangle|1\rangle &\rightarrow |e_1\rangle|1\rangle + \ |e_1^b\rangle|0\rangle \end{aligned} \tag{43}$$

When qubits interact via a gate, the most general noise operation would be a general unitary transformation on the qubit participating in the gate and their environments.

When dealing with noise, it is more convenient to use the language of density matrices, instead of vectors in the Hilbert space. I will define them here, so that I can explain the notion of "amount of noise" in the system, however they will rarely be used again later in this review. The density matrix describing a system in the state $|\alpha\rangle$ is $\rho = |\alpha\rangle\langle\alpha|$. The density matrix of part A of the system can be derived from ρ by tracing out, or integrating, the degrees of freedom which are not in A. The unitary operation on the environment and the system, which corresponds to quantum noise, can be viewed as a linear operator on the density matrix describing only the system. As a metric on density matrices we can use the fidelity[201], or the trace metric[8], where the exact definition does not matter now. Two quantum operations are said to be close if when operating on the same density matrix, they generate two close density matrices. We will say that the *noise rate* in the system is η if each of the local noise operators is within η distance from the identity map on density matrices.

We now want to find a way to compute fault tolerantly in the presence of noise rate η, where we do not want to assume any knowledge about the noise operators, except the noise rate. We will first concentrate on a simple special case, in which the computation consists of one time step which computes the identity operator on all qubits. This problem is actually equivalent to the problem of communicating with noisy channels. In order to understand the subtle points when trying to communicate with noisy channels, let us consider the classical analogous case. Classical information is presented by a string of bits instead of qubits, and the error model is simply that each bit flips with probability η.

Suppose Alice wants to send Bob a string of bits, and the channel which they use is noisy, with noise rate η, i.e. each bit flips with probability η. In order to protect information against noise, Alice can use redundancy. Instead of sending k bits, Alice will encode her bits on more bits, say n, such that Bob can apply some recovery operation to get the original k bits. The idea is that to first approximation, most of the bits will not be damaged, and the encoded bits, sometimes called the the *logical bits*, can be recovered. The simplest example of a classical code is the majority code, which encodes one logical bit on three bits.

$$0 \longmapsto 0_L = 000 \quad , \qquad 1 \longmapsto 1_L = 111$$

This classical code corrects one error, because if one bit has flipped, taking the majority vote of the three bits still recovers the logical bit. However, if more then one bit has flipped, the logical bit can no longer be recovered. If the probability for a bit flip is η, then the probability that the three bits cannot be recovered, i.e. the effective noise rate η_e, equals:

$$\eta_e = 3\eta^2(1-\eta) + \eta^3.$$

If we require that we gain some advantage in reliability by the code, then $\eta_e < \eta$ implies a *threshold* on η, which is $\eta < 0.5$. If η is above the threshold, using the code will only decrease the reliability.

The majority code becomes extremely non efficient when Alice wants to send long messages. If we require that Bob receives all the logical bits with high probability of being correct, Alice will have to use exponential redundancy for each bit. However, there are error correcting codes which map k bits to $m = O(k)$ bits, such that the probability for Bob to get the original message of k bits correct is high, even when k tends to infinity. A very useful class of error correcting codes are the *linear* codes, for which the mapping from k bits to n bits is linear, and the set of *code words*, i.e. the image of the mapping, is a linear subspace of F_2^m. A code is said to correct d errors if a recovery operation exists even if d bits have flipped. The *Hamming distance* between two strings is defined to be the number of coordinates by which the two strings differ. Being able to recover the string after d bit flips have occurred implies that the distance between two possible code words is at least $2d+1$, so that each word is corrected uniquely. For an introduction to the subject of classical error correcting codes, see van Lint[139].

We define a quantum code in a similar way. The state of k qubits is mapped into the state of m qubits. The term *logical state* is used for the original state of the k qubits. We say that such a code corrects d errors, if there exists a recovery operation such that if not more than d qubits were damaged, the logical state can still be recovered. It is important here that Bob has no control on the environment with which the qubits interacted during the noise process. Therefore we require that the recovery operation does not operate on the environment but merely on the m qubits carrying the message and perhaps some ancilla qubits. The image of the map in the Hilbert space of m qubits will be called a *quantum code*.

Let us now try to construct a quantum code. Suppose that Alice wants to send Bob a qubit in the state $c_0|0\rangle + c_1|1\rangle$. How can she encode the information? One way to do this is simply to send the classical information describing c_0 and c_1 up to the desired accuracy. We will not be interested in this way, because when Alice wants to send Bob a state of n qubits, the amount of classical bits that needs to be sent grows exponentially with n. We will want to encode qubits on qubits, to prevent this exponential overhead. The simplest idea that comes to mind is that Alice generates a few copies of the same state, and sends the following state to Bob:

$$c_0|0\rangle + c_1|1\rangle \longmapsto (c_0|0\rangle + c_1|1\rangle) \otimes (c_0|0\rangle + c_1|1\rangle) \otimes (c_0|0\rangle + c_1|1\rangle)\,.$$

Then Bob is supposed to apply some majority vote among the qubits. Unfortunately, a quantum majority vote among general quantum states is not a linear operation. Therefore, simple redundancy will not do. Let us try another quantum analog of the classical majority code:

$$c_0|0\rangle + c_1|1\rangle \longmapsto c_0|000\rangle + c_1|111\rangle$$

This code turns out to be another bad quantum code. It does not protect the quantum information even against one error. Consider for example, the local noise operator which operates on the first qubit in the encoded state $c_0|000\rangle + c_1|111\rangle$. It does nothing to that qubit, but it changes the state of the environment according to whether this bit is 0 or 1:

$$\begin{aligned} |0\rangle \otimes |e\rangle &\longmapsto |0\rangle \otimes |e_0\rangle \\ |1\rangle \otimes |e\rangle &\longmapsto |1\rangle \otimes |e_1\rangle \end{aligned} \tag{44}$$

Here $\langle e_0|e_1\rangle = 0$. Even though only an identity operation was applied on the first bit, the fact that the environment changed according to the state of this bit is equivalent to the environment *measuring* the state of the first qubit. This measurement is an irreversible process. After the noise operation, the environment is no longer in a tensor product with the state. Bob can only apply local operations on his system, and cannot control the environment. This means that the entanglement between the state of the first qubit, and the environment cannot be broken during the recovery operation; the coherence of the state is lost. A theorem due to Schumacher and Nielsen[169] formalizes this intuition. The claim is that if the reduced density matrix of the environment is different for different code words, then there is no unitary operation that operates on the system and recovers the logical state.

Theorem 9 *It is impossible to recover the logical state, if information about it has leaked to the environment via the noise process.*

This theorem underlines the main distinction between quantum error correcting codes and classical error correcting codes. Quantum codes try to *hide* information from the environment, In contrast, the protection of classical information from noise, is completely orthogonal to the question of hiding secrets. The theorem gives us insight as to the basic idea in quantum computation: The idea is to spread the quantum information over more than d qubits, in a non-local way, such that the environment which can access only a small number of qubits can gain no information about the quantum logical state, and this information will be protected. Now, that we have some intuition about the requirements from quantum codes, we can proceed to show how to construct such codes.

12 Correcting Quantum Noise

In order to succeed in correcting quantum noise, we need to consider more carefully the process of noise. The first and most crucial step is the discovery that quantum noise can be treated as discrete. In the quantum setting, we assume all qubits undergo a noise of size η. We want to replace this with the case in which a few qubits are completely damaged, but the rest of the qubits are completely fine. This can be done by rewriting the effect of a general noise operator. Let the state of m qubits be $|\alpha\rangle$. If the noise rate is η, we can develop the operation of a general noise operator operating on $|\alpha\rangle$ by orders of magnitude of η:

$$\begin{aligned}
&\mathcal{E}_1\mathcal{E}_2....\mathcal{E}_m|\alpha\rangle = \\
&(I_1+\eta\mathcal{E}'_1)(I_2+\eta\mathcal{E}'_2)...(I_m+\eta\mathcal{E}'_m)|\alpha\rangle = \\
&I_1I_2...I_m|\alpha\rangle + \eta\left(\mathcal{E}'_1I_2...I_m + ... + I_1I_2...I_{m-1}\mathcal{E}'_m\right)|\alpha\rangle + ... + \eta^m\left(\mathcal{E}'_1\mathcal{E}'_2...\mathcal{E}'_m\right)|\alpha\rangle.
\end{aligned} \tag{45}$$

The lower orders in η correspond to a small number of qubits being operated upon, and higher orders in η correspond to more qubits being contaminated. This way of writing the noise operator is the beginning of discretization of the quantum noise, because in each term a qubit is either damaged or not. For small η, we can neglect higher order terms and concentrate in the lower orders, where only one or two qubits are damaged out of m. A special case of this model is the probabilistic model, in which the local noise operator applies a certain operation with probability η and the identity operation with probability $(1-\eta)$. In this model, if the quantum system consists of m qubits, we can assume that with high probability only a few of the qubits went through some noise process. There are noise operators, such as amplitude damping, which do not obey this probabilistic behavior. However their description by equation (45) shows that we can treat them in the same discrete manner.

The second step is the discretization of the noise operation itself. The most general quantum operation on the k'th qubit and it's environment is described by:

$$|e\rangle|0_k\rangle \rightarrow |e_0\rangle|0_k\rangle + \ |e_0^b\rangle|1_k\rangle \\ |e\rangle|1_k\rangle \rightarrow |e_1\rangle|1_k\rangle + \ |e_1^b\rangle|0_k\rangle \tag{46}$$

This operation, applied on any logical state $c_0|0_L\rangle + c_1|1_L\rangle$, acts as the following operator:

$$(c_0|0_L\rangle + c_1|1_L\rangle) \rightarrow \left(|e_+\rangle\mathcal{I} + |e_-\rangle\sigma_z^k + |e_+^b\rangle\sigma_x^k - |e_-^b\rangle i\sigma_y^k\right)(c_0|0_L\rangle + c_1|1_L\rangle)\,, \tag{47}$$

Where σ_i^k are the Pauli operators acting on the k'th qubit:

$$\mathcal{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{48}$$

The environment states are defined as $|e_\pm\rangle = (|e_0\rangle \pm |e_1\rangle)/2$, $|e_\pm^b\rangle = (|e_0^b\rangle \pm |e_1^b\rangle)/2$. The most crucial observations, which enables to correct quantum errors, hide in equation 47. The first observation is that everything that can happen to a qubit is composed of four basic operations, so it is enough to correct for these four errors[35, 91, 129]. This resembles a discrete model more than a continuous one, and gives hope that such discrete errors can be corrected. The second crucial point is that the states of the environment which are entangled with the system after the operation of noise, are *independent* of $(c_0|0_L\rangle + c_1|1_L\rangle)$ and depend only on which of the four operations σ_i^k were applied. In particular, for any superposition of the logical states $|0_L\rangle, |1_L\rangle$, the operator will look the same. This suggests the following scheme of breaking the entanglement of the system with the environment. The idea is to measure which one of the four possible operators was applied. This is called the *syndrome* of the error. Measuring the syndrome will collapse the system to a state which is one of the following tensor products of the system and the environment:

$$\left(|e_+\rangle\mathcal{I} + |e_-\rangle\sigma_z^k + |e_+^b\rangle\sigma_x^k - |e_-^b\rangle i\sigma_y^k\right)(c_0|0_L\rangle + c_1|1_L\rangle) \stackrel{measure}{\longrightarrow} \left\{ \begin{array}{l} |e_+\rangle\mathcal{I}\Big(c_0|0_L\rangle + c_1|1_L\rangle\Big) \\ |e_-\rangle\sigma_z^k\Big(c_0|0_L\rangle + c_1|1_L\rangle\Big) \\ |e_+^b\rangle\sigma_x^k\Big(c_0|0_L\rangle + c_1|1_L\rangle\Big) \\ |e_-^b\rangle i\sigma_y^k\Big(c_0|0_L\rangle + c_1|1_L\rangle\Big) \end{array} \right. \tag{49}$$

After we know which of the operators had occurred, we can simply apply its reverse, and the state $c_0|0_L\rangle + c_1|1_L\rangle$ will be recovered. This reduces the problem of error correction to being able to detect which of the four operators had occurred. The operator σ_x corresponds to a *bit flip*, which is a classical error. This suggests the following idea: If the superposition of the encoded state, is a sum of strings $|i\rangle$ where the i's are strings from a classical code, then bit flips can be detected by applying classical techniques. Correcting the noise operator σ_z, which is a *phase flip*, seems harder, but an important observation is that $\sigma_z = H\sigma_x H$, where H is the Hadamard transform. Therefore, phase flips correspond to bit flips occurring in the Fourier transform of the state! If the Fourier transform of the state is also a superposition of strings in a classical code, this enables a correction of phase flips by correcting the bit flips in the Fourier transform basis. This idea was discovered by Calderbank and Shor[53] and Steane[180].

A simple version of the recipe they discovered for cooking a quantum code goes as follows. Let $C \subset F_2^m$ be a linear classical code, which corrects d errors, such that $C^\perp$, the set of all strings orthogonal over F_2 to all vectors in C, is strictly contained in C. We look at the cosets of $C^\perp$ in C, i.e. we partition C to non intersecting sets which are translations of $C^\perp$ of the form $C^\perp + v$. The set of vectors in C, with the identification of w with w' when $w - w' \in C^\perp$ is called $C/C^\perp$. For each $w \in C/C^\perp$ we associate a code word:

$$|w\rangle \longmapsto |w_L\rangle = \sum_{i \in C^\perp} |i + w\rangle \tag{50}$$

where we omit overall normalization factors. Note that all the strings which appear in the superposition are vectors in the code C. It is easy to check that the same is true for the Fourier transform over Z_2^m of the code words, which is achieved by applying the Hadamard gate, H, on each qubit:

$$H \otimes H \otimes .... \otimes H|w_L\rangle = \sum_{j \in C} (-1)^{w \cdot j} |j\rangle. \tag{51}$$

The error correction goes as follows. To detect bit flips, we apply the classical error correction according to the classical code C, on the states in equation (50). This operation, computes the syndrome (in parallel for all strings) and writes it on some ancilla qubits. Measuring the ancilla will collapse the state to a state with a specific syndrome, and we can compute according to the result of the measurement which qubits were affected by a bit flip, and apply NOT on those qubits. To detect phase flips we apply Fourier transform on the entire state, and correct bit flips classically according to the code C. Then we apply the reverse of the Fourier transform. This operation will correct phase flips. σ_y is a combination of a bit flip and a phase flip, and is corrected by the above sequence of error corrections[53].

The number of qubits which can be encoded by this code is the logarithm with base 2 of the dimension of the space spanned by the code words. To calculate this dimension, observe that the code words for different w's in $C/C^\perp$ are perpendicular. The dimension of the quantum code is equal to the number of different words in $C/C^\perp$, which is $2^{dim(C/C^\perp)}$. Hence the number of qubits which can be encoded by this quantum code is $dim(C/C^\perp)$.

Here is an example, due to Steane[180]. Steane's code encodes one qubit on seven qubits, and corrects one error. It is constructed from the classical code known as the

Hamming code, which is the subspace of F_2^7 spanned by the four vectors: $C = span\{1010101, 0110011, 0001111, 1111111\}$. $C^\perp$ is spanned by the three vectors: $1010101, 0110011, 0001111$. Since C is of dimension 4, and $C^\perp$ is of dimension 3, the number of qubits which we can encode is 1. The two code words are:

$$
\begin{aligned}
|0_L\rangle &= |0000000\rangle + |1010101\rangle + |0110011\rangle + |1100110\rangle \\
&\quad + |0001111\rangle + |1011010\rangle + |0111100\rangle + |1101001\rangle \\
|1_L\rangle &= |1111111\rangle + |0101010\rangle + |1001100\rangle + |0011001\rangle \\
&\quad + |1110000\rangle + |0100101\rangle + |1000011\rangle + |0010110\rangle
\end{aligned} \tag{52}
$$

Observe that the minimal Hamming distance between two words in C is 3, so one bit flip and one phase flip can be corrected.

One qubit cannot be encoded on less than 5 qubits, if we require that an error correction of one general error can be done. This was shown by Knill and Laflamme[129]. Such a code, called a perfect quantum code, was found by Bennett et al[35] and by Laflamme *et.al.* [134]. If we restrict the error, e.g. only bit flips or only phase flips occur than one qubit can be encoded on less than 5 qubits.

The theory of quantum error correcting codes has further developed. A group theoretical structure was discovered [54, 55, 105, 106, 129, 175], which most of the known quantum error correcting codes obey. Codes that obey this structure are called stabilizer codes[105, 106], and their group theoretical structure gives a recipe for constructing more quantum codes. Quantum codes are used for purposes of quantum communication with noisy channels, which is out of the scope of this review. For an overview on the subject of quantum communication consult Refs. [21, 158] and [144]. We now have the tools to deal with the question of quantum computation in the presence of noise, which I will discuss in the next section.

13 Fault Tolerant Computation

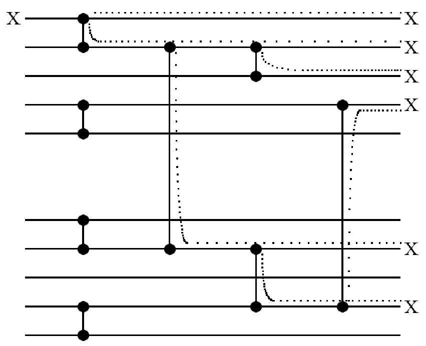
In order to protect quantum computation, the idea is that one should compute on encoded states. The entire operation will occur in the protected subspace, and every once in a while an error correction procedure will be applied, to ensure that errors do not accumulate. The original quantum circuit will be replaced by a quantum circuit which operates on encoded state. Suppose we use a quantum code which encodes one qubit into a block of 5 qubits. Then in the new circuit, each wire will be replaced by five wires, and the state of the new circuit will encode the state of the original circuit. In order to apply computation on encoded states, the original gates will be replaced by procedures which apply the corresponding operation. If Φ is the encoding, U is a quantum gate, then $\Phi(U)$ should be the "encoded gate" U, which preserves the encoding. In other words, the following diagram should be commutative:

$$\begin{tikzcd} \Phi(|\alpha\rangle) \arrow[r, "\Phi(U)"] & \Phi(U|\alpha\rangle) \\ |\alpha\rangle \arrow[u, "\Phi"] \arrow[r, "U"'] & U|\alpha\rangle \arrow[u, "\Phi"'] \end{tikzcd}$$

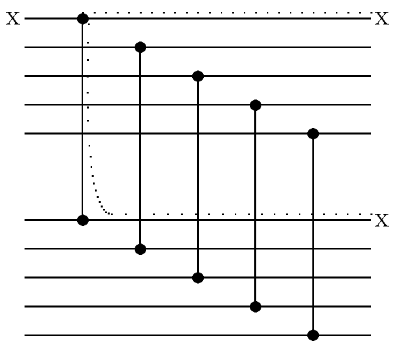
Hence, using a code Φ, which takes one qubit to m qubits, we replace a quantum circuit by another circuit which operates on encoded states, in this circuit

- 1 qubit $\longmapsto$ m qubits
- A gate $U \longmapsto \Phi(U)$
- Every few time steps, an error correction procedure is applied.

However, this naive scheme encounters deep problems. Since quantum gates create interactions between qubits, errors may propagate through the gates. Even a small number of errors might spread to more qubits than the error correction can recover. Moreover, we can no longer assume that the recovery operation is error free. The correction procedure might cause more damage than it recovers. Consider, for example, a code Φ that takes one qubit to 5 qubits. A gate on two qubits, U, is replaced in the encoded circuit by the encoded gate $\Phi(U)$ which operates on 10 qubits. Let us consider two scenarios:



a



b

In figure (a), the encoded gate is a gate array with large connectivity. An error which occurred in the first qubit, will propagate through the gates to five more qubits. At the end of the procedure, the number of damaged qubits is too large for any error correction to take care of. Such procedure will not tolerate even one error! In figure (b), we see an alternative way to implement $\Phi(U)$, in which the error cannot propagate to more than one qubit in each block. If the gate is encoded such that one error effects only one qubit in each block, we say that the encoded gate is implemented *distributively.* Such damage will be corrected during the error corrections. Of course, the error correction procedures should also be implemented in a distributed manner. Otherwise the errors generated during the correction procedure itself will contaminate the state.

Probably the simplest gate to implement distributively is the encoded NOT gate on Steane's code. The encoded NOT is simply achieved by applying a NOT gate bitwise on each qubit in the code. The implementation of the XOR gate is applied bitwise as well, and the network is the same as that in figure (b), only on 7 qubits instead of five. However, for other gates much more work needs to be done. Shor[174], showed a way to implement a universal set of gates in this way, where the implementation of some of the gates, and Toffoli's gate in particular, require some hard work and the use of additional "ancilla" or "working" qubits. Together with the set of universal encoded gates, one also needs an error correction procedure, an encoding procedure to be used in the beginning of the computation, and a decoding procedure to be used at the end. All these procedures should be implemented distributively, to prevent propagation of errors. A code which is accompanied by a set of universal gates, encoding, decoding and correction procedures, all implemented distributively, will be called a *quantum computation code.* Since Shor's suggestion, other computation codes were found[5, 128]. Gottesman[106] has generalized these results and showed how to construct a computation code from any stabilizer code.

Is the encoded circuit more reliable? The *effective noise rate*, η_e of the encoded circuit, is the probability for an encoded gate to suffer a number of errors which cannot be corrected. In the case of figure (b), one error is still recoverable, but two are not. The effective noise rate is thus the probability for two or more errors to occur in $U(\Phi)$. Let A denote the number of places in the implementation of $U(\Phi)$ where errors can occur. A stands for the *area* of $U(\Phi)$. The probability for more than d errors to occur can be bounded from above, using simple counting arguments:

$$\eta_e \leq \left(\begin{array}{c} A \\ d+1 \end{array} \right) \eta^{d+1} \tag{53}$$

We will refer to this bound as the *effective noise rate.* To make a computation of size n reliable, we need an effective noise rate of the order of $\frac{1}{n}$. Using a code with blocks of log(n) qubits, Shor[174] managed to show that the computation will be reliable, with polynomial cost. However, Shor had to assume that η is as small as $O(\frac{1}{\log^4(n)})$. This assumption is not physically reasonable , since η is a parameter of the system, independent of the computation size. The reader is urged to play with the parameters of equation 53 in order to be convinced that assuming η to be constant cannot lead to a polynomially small effective noise rate, as required.

Another idea, which was found independently by several groups [5, 128, 124, 107] was needed to close the gap, and to show that computation in the presence of constant noise rate and finite precision is possible. The idea is simple. Apply Shor's scheme recursively, gaining small improvement in the effective noise rate each level . Each circuit is replaced by a slightly more reliable circuit, which is replaced again by yet another circuit. If each level gains only a slight improvement from η to $\eta^{1+\epsilon}$, then the final circuit which is the one implemented in the laboratory, will have an effective noise rate exponentially smaller:

$$\eta \longmapsto \eta^{1+\epsilon} \longmapsto (\eta^{1+\epsilon})^{1+\epsilon} \ldots \longmapsto \eta^{(1+\epsilon)^r}$$

The number of levels the recursion should be applied to get a polynomially small effective noise rate is only $O(\log(\log(n)))$. The cost in time and space is thus only polylogarithmic. A similar concatanation scheme was used in the context of classical self correcting cellular automata[66, 98].

The requirement that the noise rate is improved from one level to the next imposes a threshold requirement on η:

$$\left(\begin{array}{c} A \\ d+1 \end{array} \right) \eta^{d+1} < \eta$$

If η satisfies the above requirement, fault tolerant computation can be achieved. This is known as the threshold result[5, 128, 122, 107]:

Theorem 10 Fault tolerance: *Quantum computation of any length can be applied efficiently with arbitrary level of confidence, if the noise rate is smaller than the threshold* η_c.

The *threshold* η_c, depends on the parameters of the computation code: A, the largest procedure's area, and d, the number of errors which the code can correct. Estimations[5, 128, 106, 107, 130, 162] of η_c are in the range between 10^{-4} and 10^{-6}. Presumably the correct threshold is much higher. The highest noise rate in which fault tolerance is possible is not known yet.

The rigorous proof of the threshold theorem is quite complicated. To gain some insight we can view the final r'th circuit as a multi scaled system, where computation and error correction are applied in many scales simultaneously. The largest procedures, computing on the largest (highest level) blocks, correspond to operations on the logical qubits, i.e. qubits in the original circuit. The smaller procedures, operating on smaller blocks, correspond to computation in lower levels. Note, that each level simulates the error corrections in the previous level, and adds error corrections in the current level. The final circuit, thus, includes error corrections of all the levels, where during the computation of error corrections of larger blocks smaller blocks of lower levels are being corrected. The lower the level, the more often error corrections of this level are applied, which is in correspondence with the fact that smaller blocks are more likely to be quickly damaged.

The actual system consists of $m = n\log^c(n)$ qubits (where n is the size of the original circuit), with a Hilbert space $\mathcal{H} = C^{2^m}$. In this Hilbert space we find a subspace, isomorphic to C^{2^n}, which is protected against noise. This subspace is a complicated multi-scaled construction, which is small in dimensions, compared to the Hilbert space of the system, but not negligible. The subspace is protected against noise for almost as long as we wish, and the quantum computation is done exactly in this protected subspace. The rate by which the state increases its distance from this subspace corresponds to the noise rate. The efficiency of the error correction determines the rate by which the distance from this subspace decreases. The threshold in the noise rate is the point where distance is decreases faster than it increases. In a sense, the situation can be viewed as the operation of a renormalization group, the change in the noise rate being the renormalization flow.

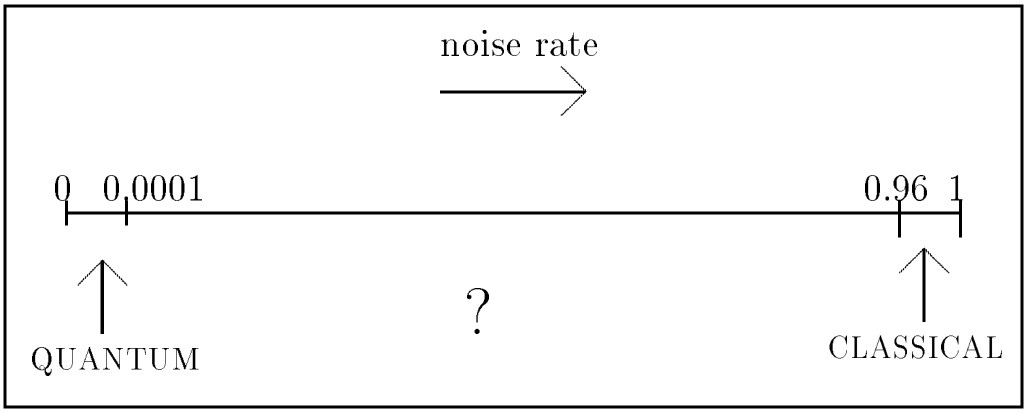
It should be noted that along the proof of fault tolerance, a few implicit assumptions were made [183]. The ancilla qubits that we need in the middle of the computation for error correction are assumed to be prepared in state $|0\rangle$ *when needed*, and not at the beginning of the computation. This requires the ability to cool part of the system constantly. It was shown by Aharonov *et. al.*[8] that if all operations are unitary, the system keeps warming (in the sense of getting more noise) with no way to cool, and the rate in which the system warms up is *exponential*. Fault tolerant quantum computation requires using non-unitary gates which enables to cool a qubit. This ability to cool qubits is used implicitly in all fault tolerant schemes. Another point which should be mentioned is that fault tolerant computation uses immense parallelism, i.e. there are many gates which are applied at the same time. Again, this implicit assumption is essential. If operation were sequential, fault tolerant computation would have been impossible, as was shown by Aharonov and Ben-Or[5]. However, with mass parallelism, constant supply of cold qubits and a noise rate which is smaller than η_c, it is possible to perform fault tolerant computation.

The fault tolerance result holds for the general local noise model, as defined before, and this includes probabilistic collapses, inaccuracies, systematic errors, decoherence, etc. One can compute fault tolerantly also with quantum circuits which are allowed to operate only on nearest neighbor qubits[5] (In this case the threshold η_c will be smaller, because the procedures are bigger when only nearest neighbor interactions are allowed.) In a sense, the question of noisy quantum computation is theoretically closed. But a question still ponders our minds: Are the assumptions on the noise correct? Dealing with non-local noise is an open and challenging problem.

14 Conclusions and Fundamental Questions

We cannot foresee which goals will be achieved, if quantum computers be the next step in the evolution of computation[115]. This question involves two directions of research. From the negative side, we are still very far from understanding the limitations of quantum computers as computation devices. It is possible that quantum Fourier transforms are the only real powerful tool in quantum computation. Up to now, this is the only tool which implies exponential advantage over classical algorithms. However, such a strong statement of the uniqueness of the Fourier transform is not known. Taking a more positive view, the goal is to find other techniques in addition to the Fourier transform. One of the main directions of research in quantum algorithms is finding an efficient solutions for a number of problems which are not known to be NP complete, but do not have a known efficient classical solution. Such is the problem of checking whether two graphs are isomorphic, known as *Graph Isomorphism.* Another important direction in quantum algorithms is finding algorithms that simulate quantum physical systems more efficiently. The field of quantum complexity is still in its infancy.

Hand in hand with the complexity questions, arise deep fundamental questions about quantum physics. The computational power of all classical systems seem to be equivalent, whereas quantum complexity, in light of the above results, seems inherently different. If it is true that quantum systems are exponentially better computation devices than classical systems, this can give rise to a new definition of quantum versus classical physics, and might lead to a change in the way we understand the transition from quantum to classical physics. The "phase diagram" of quantum versus classical behavior can be viewed as follows:



Changing the noise rate, the system transforms from quantum behavior to classical behavior. As was shown by Aharonov and Ben-Or[6], there is a constant η bounded away from 1 where the system cannot perform quantum computation at all. Fault tolerance shows that there is a constant η bounded away from 0 for which quantum systems exhibit their full quantum computation power. The regimes are characterized by the range of quantum entanglement, where in the quantum regime this range is macroscopic, and quantum computation is possible. On the right, "classical", range, entanglement is confined to microscopic clusters. A very interesting question is how does the transition between the two regimes occur. In [6] we gave indications to the fact that the transition is sharp and has many characteristics of a phase transition (and see also [119].) The order parameter corresponds to the range of entanglement, or to the size of entangled clusters of qubits. Unfortunately, we were unable yet to prove the existence of such a phase transition, presumably because of lack of the correct definition of an order parameter that quantifies "quantumness over large scales". Never the less I conjecture that the transition from macroscopic quantum behavior to macroscopic classical behavior occurs as a phase transition. The idea that the transition from quantum to classical physics is abrupt stands in contrast to the standard view of a gradual transition due to decoherence[205]. I believe that the flippant frontier between quantum and classical physics will be better understood if we gain better understanding of the transition from quantum to classical computational behavior.

An interesting conclusion of the threshold result is that one dimensional quantum systems can exhibit a non trivial phase transition at a critical noise rate η_c, below which the mixing time of the system is exponential, but above which the system mixes rapidly. This phase transition might be different from the transition from classical to quantum behavior, or it might be the same. This existence of a one dimensional phase transition is interesting because one dimensional phase transitions are rare, also in classical systems, though there exist several complicated examples[152, 99].

Perhaps a vague, but deeper, and more thought provoking question is that of the postulates of quantum mechanics. The possibility that the model will be realized will enable a thorough test of some of the more philosophical aspects of quantum theory, such as understanding the collapse of the wave function, the process of measurement, and other elements which are used as everyday tools in quantum algorithms. It might be that the realization of quantum computation will reveal the fact that what we understand in quantum physics is merely an approximation holding only for small number of particles, which we extrapolated to many particles. Such questions are appealing motivations for this extremely challenging task of realizing the quantum computation model physically. It seems that successes, and also failures, in achieving this ambitious task, will open new exciting paths and possibilities in both computer science and fundamental physics.

15 Acknowledgments

I am most grateful to Michael Ben Or who introduced me to this beautiful subject. We had a lot of fascinating discussions together on many of the things I presented. Noam Nisan taught me a lot simply by asking the right questions, with his clear point of view. It was a pleasure not to know the answers. It was great fun to argue with Avi Wigderson on quantum computation and other things. It is a special pleasure to thank my colleagues Peter Hoyer, Lidror Troyanski, Ran Raz and particularly Michael Nielsen. They all read the manuscript, corrected many errors, and had extremely helpful suggestions. Finally, I thank Ehud Friedgut for direct and indirect contributions to this review.

References

[1] Abrams D S and Lloyd S, Simulation of Many-Body Fermi Systems on a Universal Quantum Computer, *Phys.Rev.Lett.* **79** 2586–2589, 1997

[2] Abrams D S and Lloyd S, Non-Linear Quantum Mechanics implies Polynomial Time solution for NP-complete and #P problems, in *LANL e-print* quant-ph/9801041, http://xxx.lanl.gov (1998)

[3] Adleman L, Demarrais J and Huang M-D, Quantum Computability, *SIAM Journal of Computation* **26** 5 pp 1524–1540 October, 1997

[4] Adleman L, Molecular computation of solutions to combinatorial problems, *Science*, 266, 1021–1024, Nov. 11, 1994

[5] Aharonov D and Ben-Or M, Fault-Tolerant Quantum Computation with Constant Error, *Proc. of the 29th Annual ACM Symposium on Theory of Computing (STOC)* 1997

[6] Aharonov D and Ben-Or M, Polynomial Simulations of Decohered Quantum Computers *37th Annual Symposium on Foundations of Computer Science* (FOCS) pp 46–55, 1996

[7] Aharonov D, Kitaev A Yu and Nisan N, Quantum Circuits with Mixed States, *Proc. of the 30th Annual ACM Symposium on Theory of Computing (STOC)* 1998

[8] Aharonov D, Ben-Or M, Impagliazo R and Nisan N, Limitations of Noisy Reversible Computation, in *LANL e-print* quant-ph/9611028, http://xxx.lanl.gov (1996)

[9] Aharonov D, Beckman D, Chuang I and Nielsen M, What Makes Quantum Computers Powerful? preprint in http://wwwcas.phys.unm.edu∼mnielsen/science.html

[10] Aspect A, Dalibard J and Roger G, Experimental test of Bell's inequalities using time-varying analyzers, *Phys. Rev. Lett.* **49**, 1804–1807, 1982

[11] Aspect A, Testing Bell's inequalities, *Europhys. News.* **22**, 73–75, 1991

[12] Barahona F, in *J. Phys. A* vol. 15, (1982) 3241

[13] Barenco A A universal two-bit gate for quantum computation, *Proc. R. Soc. Lond. A* **449** 679–683, 1995

[14] Barenco A and Ekert A K Dense coding based on quantum entanglement, *J. Mod. Opt.* **42** 1253–1259, 1995

[15] Barenco A, Deutsch D, Ekert E and Jozsa R, Conditional quantum dynamics and quantum gates, *Phys. Rev. Lett.* **74** 4083–4086, 1995

[16] Barenco A, Bennett C H, Cleve R, DiVincenzo D P, Margolus N, Shor P, Sleator T, Smolin J A and Weinfurter H, Elementary gates for quantum computation, *Phys. Rev. A* **52**, 3457–3467, 1995

[17] Barenco A, Ekert A, Suominen K A and Torma P, Approximate quantum Fourier transform and decoherence, *Phys. Rev. A* **54**, 139–146, 1996

[18] Barenco A, Berthiaume A, Deutsch D, Ekert A, Jozsa R, and Macchiavello C, Stabilization of Quantum Computations by Symmetrization, *SIAM J. Comp.***26**,5, 1541–1557, 1997

[19] Barenco A, Brun T A, Schak R and Spiller T P, Effects of noise on quantum error correction algorithms, *Phys. Rev. A* **56** 1177–1188, 1997

[20] Barnum H, Caves C, Fuchs C A, Jozsa R and Schumacher B, Non commuting mixed states cannot be broadcast, *Phys. Rev. Lett.***76** 2818–2822, 1996

[21] Barnum H, Nielsen M and Schumacher B, in *Phys. Rev. A*, **57**,6, 1998, pp. 4153–4175

[22] Beals R, Quantum computation of Fourier transform over symmetric groups *Proc. of the 29th Annual ACM Symposium on Theory of Computing (STOC)* 1997

[23] Beals R, Buhrman H, Cleve R, Mosca M and de Wolf R, Quantum Lower Bounds by Polynomials, in *39th Annual Symposium on Foundations of Computer Science(FOCS)*, (1998)

[24] Beckman D, Chari A, Devabhaktuni S and Preskill J Efficient networks for quantum factoring, *Phys. Rev. A* **54**, 1034–1063, 1996

[25] Bell J S On the Einstein-Podolsky-Rosen paradox, *Physics* **1** 195–200, 1964

[26] Bell J S On the problem of hidden variables in quantum theory, Rev. Mod. Phys. **38** 447-52, 1966 *Speakable and unspeakable in quantum mechanics* 1987 (Cambridge University Press)

[27] Benioff P, The Computer as a Physical Systems: A Microscopic Quantum Mechanical Hamiltonian Model of Computers as Represented by Turing Machines, *J. Stat. Phys.* **22** 563–591, 1980

[28] Benioff P Quantum mechanical Hamiltonian models of Turing machines, *J. Stat. Phys.* **29** 515-546 1982

[29] Bennett C H, Logical reversibility of computation, *IBM J. Res. Develop.* **17** 525–532, 1973

[30] Bennett C H, The Thermodynamics of Computation - a Review, *International Journal of Theoretical Physics,* **21**, No. 12, p 905, 1982

[31] Bennett C H, Demons, engines and the second law, *Scientific American* **257** no. 5 (November) pp 88–96, 1987

[32] C.H. Bennett, Time/Space Trade-offs for Reversible Computation, *SIAM Journal of Computation*, **18**, 4, pp 766–776, 1989

[33] Bennett C H, Bessette F, Brassard G, Savail L and Smolin J Experimental quantum cryptography, *J. Cryptology* **5**, pp 3–28, 1992

[34] Bennett C H, Brassard G, Crépeau C, Jozsa R, Peres A and Wootters W K Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels, *Phys. Rev. Lett.* **70** 1895–1898, 1993

[35] Bennett C H, DiVincenzo D P, Smolin J A and Wootters W K Mixed state entanglement and quantum error correction, *Phys. Rev. A* **54** 3825, 1996

[36] Bennett C H, Bernstein E, Brassard G and Vazirani U Strengths and Weaknesses of quantum computing, *SIAM Journal of Computation* **26** 5 pp 1510–1523 October, 1997

[37] Berman G P, Doolen G D, Holm D D, Tsifrinovich V I Quantum computer on a class of one-dimensional Ising systems, *Phys. Lett.* **193** 444-450 1994

[38] Bernstein E and Vazirani U, 1993, Quantum complexity theory, *SIAM Journal of Computation* **26** 5 pp 1411–1473 October, 1997

[39] Berthiaume A, Deutsch D and Jozsa R, The stabilization of quantum computation, in *Proceedings of the Workshop on Physics and Computation, PhysComp 94* 60-62 Los Alamitos: IEEE Computer Society Press, 1994

[40] Berthiaume A and Brassard G, The quantum challenge to structural complexity theory, in *Proc. of the Seventh Annual Structure in Complexity Theory Conference* 1992 (IEEE Computer Society Press, Los Alamitos, CA) 132–137

[41] Berthiaume A and Brassard G, Oracle quantum computing, in *Proc. of the Workshop on Physics of Computation: PhysComp '92* (IEEE Computer Society Press, Los Alamitos, CA) 60–62, 1992

[42] Boghosian B M and Taylor W, Simulating quantum mechanics on a quantum computer in *Physica D*, 120 (1998) pp. 30-42

[43] Bouwmeester D, Pan J-W, Mattle K, Weinfurter H, Zeilinger A, Experimental quantum teleportation *Nature* **390**, 575-579 1997

[44] Boyer M, Brassard G, Hoyer P and Tapp A, Tight bounds on quantum searching, in *Fortsch.Phys.* 46, (1998) pp. 493-506

[45] Brassard G, Searching a quantum phone book, **Science 275** 627-628 1997

[46] Brassard G and Crepeau C, *SIGACT News* **27** 13-24 1996

[47] Brassard G, Hoyer P and Tapp A, Quantum Algorithm for the Collision Problem in *LANL e-print* quant-ph/9705002, http://xxx.lanl.gov (1997)

[48] Brassard G and Hoyer P, An Exact Quantum-Polynomail Algorithm for Simon's Problem *Proceedings of the 5th Israeli Symposium on Theory of Computing and Systems (ISTCS)*, 1997

[49] Brassard G, Teleportation as Quantum Computation, in *Physica D*, 120 (1998) 43-47

[50] Brassard G, The dawn of a new era for quantum cryptography: The experimental prototype is working!, Sigact News, **20**(4), 78–82, 1989

[51] Buhrman H, Cleve R and van Dam W, Quantum Entanglement and Communication Complexity, in *LANL e-print* quant-ph/9705033, http://xxx.lanl.gov (1997)

[52] Buhrman H, Cleve R and Wigderson A, Quantum vs. Classical Communication and Computation, in *Proc. of the 30th Annual ACM Symposium on Theory of Computing (STOC)* (1998)

[53] Calderbank A R and Shor P W, Good quantum error-correcting codes exist, *Phys. Rev. A* **54** 1098-1105, 1996

[54] Calderbank A R, Rains E M, Shor P W and Sloane N J A Quantum error correction and orthogonal geometry, *Phys. Rev. Lett.* **78** 405–408, 1997

[55] Calderbank A R, Rains E M, Shor P W and Sloane N J A, Quantum error correction via codes over $GF(4)$ in *LANL e-print* quant-ph/9608006, http://xxx.lanl.gov (1996), To appear in *IEEE Transactions on Information Theory.*

[56] Chernoff. See Feller W, An Introduction to Probability Theory and Its Applications, Wiley, New York,1957

[57] Chuang I L, Laflamme R, Shor P W and Zurek W H, Quantum computers, factoring, and decoherence, *Science* **270** 1633–1635, 1995

[58] Chuang I L, Laflamme R and Paz J P, Effects of Loss and Decoherence on a Simple Quantum Computer, in *LANL e-print* quant-ph/9602018, http://xxx.lanl.gov (1996)

[59] I.Chuang and W.C.D. Leung and Y. Yamamoto, Bosonic Quantum Codes for Amplitude Damping, in *Phys. Rev. A*, **56**, 2, (1997) pp. 1114-1125

[60] Chuang I L and Yamamoto Creation of a persistent qubit using error correction Phys. Rev. A **55**, 114–127, 1997

[61] Chuang I L ,Vandersypen L M K, Zhou X, Leung D W and Lloyd S, Experimental realization of a quantum algorithm, in *Nature*, 393, 143-146 (1998)

[62] Church A An unsolvable problem of elementary number theory, *Amer. J. Math.* **58** 345–363, 1936

[63] Cirac I J and Zoller P Quantum computations with cold trapped ions, *Phys. Rev. Let.*, 74: 4091-4094, 1995.

[64] Cirac J I, Pellizari T and Zoller P, Enforcing coherent evolution in dissipative quantum dynamics, *Science* **273**, 1207, 1996

[65] Cirac J I, Zoller P, Kimble H J and Mabuchi H Quantum state transfer and entanglement distribution among distant nodes of a quantum network, *Phys. Rev. Lett.* **78**, 3221, 1997

[66] Cirel'son (Tsirelson) B, Reliable storage of information in a system of unreliable components with local interactions. *Lecture notes in Mathematics***653** 15–30 ,1978

[67] Clausen M, Fast Generalized Fourier transforms, *Theoret. Comput. Sci.* **56** 55–63 1989

[68] Clauser J F, Holt R A, Horne M A and Shimony A Proposed experiment to test local hidden-variable theories, *Phys. Rev. Lett.* **23** 880–884, 1969

[69] Cleve R and Buhrman H, Substituting Quantum Entanglement for Communication in*Phys rev A*,**56** 2, (1997) pp. 1201-1204

[70] Cleve R, van Dam W, Nielsen M and Tapp A, Quantum Entanglement and the Communication Complexity of the Inner Product Function, in *LANL e-print* quant-ph/9708019, http://xxx.lanl.gov (1997)

[71] C. Cohen-Tanoudji, *Quantum Mechanics*, Wiley press, New York (1977)

[72] Coppersmith D, An approximate Fourier transform useful in quantum factoring, IBM Research Report RC 19642, 1994

[73] Cormen T, Leiserson C and Rivest R, *Introduction to Algorithms*, (pp 776–800 for FFT, 837–844 for primality test, 812 for extended Euclid algorithm, 834–836 for RSA cryptosystem) MIT press, 1990

[74] Cory D G, Fahmy A F, and Havel T F, Nuclear magnetic resonance spectroscopy: an experimentally accessible paradigm for quantum computing, in *Proc. of the 4th Workshop on Physics and Computation* (Complex Systems Institute, Boston, New England) 1996

[75] Cory D. G, Fahmy A F, and Havel T F *Proc. Nat. Acad. of Sciences of the U. S.*, 94:1634–1639, 1997.

[76] Cory D G, Mass W, Price M, Knill E, Laflamme R, Zurek W H, and Havel T F and Somaroo S S, Experimental quantum error correction in *Phys. Rev. Let.***81** 10, (1998) pp. 2152-2155

[77] van Dam W, A Universal Quantum Cellular Automaton, *Proceedings of the Fourth Workshop on Physics and Computation*, 1996

[78] Deutsch D, Quantum theory, the Church-Turing principle and the universal quantum computer, In *Proc. Roy. Soc. Lond.* A **400** 97-117, 1985

[79] Deutsch D, Quantum computational networks, In *Proc. Roy. Soc. Lond.* A **425** 73-90, 1989

[80] Deutsch D and Jozsa R, Rapid solution of problems by quantum computation, In *Proc. Roy. Soc. Lond* A **439** 553-558, 1992

[81] Deutsch D, Barenco A and Ekert A, Universality in quantum computation, In *Proc. R. Soc. Lond.* A **449** 669-677, 1995

[82] Diaconis P and Rockmore D, Efficient Computation of the Fourier transform on finite groups, *J. AMS* **3** No. 2, 297–332, 1990

[83] Dieks D, Communication by EPR Devices, Phys Lett A, 92(6) 271–272 1982

[84] DiVincenzo D P, Two-bit gates are universal for quantum computation, *Phys. Rev. A* **51** 1015-1022 1995

[85] DiVincenzo D P, Quantum computation, *Science* **270** 255-261 1995

[86] DiVincenzo D P, Quantum Gates and Circuits, in *Proceedings of the ITP Conference on Quantum Coherence and Decoherence*, December, (1996), Proc. R. Soc. London A

[87] Durr C and Hoyer P, A Quantum Algorithm for Finding the Minimum, in *LANL e-print* quant-ph/9607014, http://xxx.lanl.gov (1996)

[88] Durr C, LeThanh H and Santha M, A decision procedure for well formed linear quantum cellular automata, *Proceeding of the 37th IEEE Symposium on Foundations of Computer Science*, 38–45, 1996, and *Random Structures & Algorithms*, 1997

[89] Einstein A, Rosen N and Podolsky B, *Phys. Rev.* **47**, 777 1935

[90] Ekert A and Jozsa R Quantum computation and Shor's factoring algorithm, *Rev. Mod. Phys.* **68** 733 1996

[91] Ekert A and Macchiavello C 1996 Quantum error correction for communication, Phys. Rev. Lett. **77** 2585-2588

[92] Another proof of the parity lower bound, using interesting techniques, was found recently by E. Farhi, J. Goldstone, S. Gutmann, M. Sipser A Limit on the Speed of Quantum Computation in Determining Parity in *LANL e-print* quant-ph/9802045, http://xxx.lanl.gov (1998)

[93] Feynman R P Simulating physics with computers, In *Int. J. Theor. Phys.* **21** 467-488, 1982

[94] Feynman R P, Quantum mechanical computers, In *Found. of Phys.* **16** 507-531, 1986 see also Optics News February 1985, 11-20.

[95] Fortnow L and Rogers J, Complexity Limitations on quantum computation Technical report 97-003, DePaul University, School of Computer science, 1997

[96] Fredkin E and Toffoli T 1982 Conservative logic, Int. J. Theor. Phys. **21** 219-253

[97] Freedman M, Logic, P/NP and the quantum field computer, preprint, 1997

[98] P. Ga'cs, Self Correcting Two Dimensional Arrays, in *Randomness and Computation*,1989, edited by S. Micali, vol 5, in series "Advances in Computing Research", pages 240-241,246-248, series editor: F.P.Preparata

[99] P. Ga'cs, one dimensional self correcting array.

[100] Gardiner C W, Quantum Noise, Springer-Verlag, Berlin, 1991

[101] Garey M R and Johnson D S, Computers and Intractability, published by Freeman and Company, New York, 1979

[102] A. Gaudí. The set of ropes is presented in *la sagrada familia* in Barcelona, Spain. Pictures of *la sagrada familia* can be found in: http://futures.wharton.upenn.edu∼jonath22/Gaudi/eltemple.html

[103] R. Geroch and G. Hartle Computability and Physical theories in *Between Quantum and Cosmos* edited by Zurek and Van der Merwe and Miller, Princeton University Press, 1988, 549-566

[104] Gershenfeld N A and Chuang I L Bulk spin-resonance quantum computation, *Science*, 275:350–356, 1997.

[105] Gottesman D 1996 Class of quantum error-correcting codes saturating the quantum Hamming bound, Phys. Rev. A **54**, 1862-1868

[106] Gottesman D A theory of fault-tolerant quantum computation, in *Phys. Rev. A*,**57** 127–137

[107] Gottesman D, Evslin J, Kakade D and Preskill J, preprint (1996)

[108] Greenberger D M, Horne M A and Zeilinger A 1989 Going beyond Bell's theorem, in *Bell's theorem, quantum theory and conceptions of the universe*, Kafatos M, ed, (Kluwer Academic, Dordrecht) 73-76

[109] R. B. Griffiths and C. S. Niu 1996 Semi classical Fourier transform for quantum computation in *Phys. Rev. Lett.* ,76, pp. 3228–3231

[110] Grover L K, Quantum mechanics helps in searching for a needle in a haystack, Phys. Rev. Lett. **79**, 325-328 1997 and the original STOC paper: A fast quantum mechanical algorithm for database search *Proc. of the 28th Annual ACM Symposium on Theory of Computing (STOC)* 212–221, 1996

[111] Grover L K, A framework for fast quantum mechanical algorithms, in *LANL e-print* quant-ph/9711043, http://xxx.lanl.gov (1997)

[112] Grover L K, Quantum computers can search arbitrarily large databases by a single query in *Phys. Rev. Let.* **79** 23, 4709–4712, 1997

[113] Grover L K, A fast quantum mechanical algorithm for estimating the median, in *LANL e-print* quant-ph/9607024, http://xxx.lanl.gov (1997)

[114] Hagley E et. al, Generation of Einstein Podolsky Rosen pairs of atoms, *Phys. Rev. Lett,* **79**, 1–5, 1997

[115] Haroche S and Raimond J-M 1996 Quantum computing: dream or nightmare? Phys. Today August 51-52

[116] Hughes R J, Alde D M, Dyer P, Luther G G, Morgan G L ans Schauer M 1995 Quantum cryptography, Contemp. Phys. **36** 149-163

[117] A. J. Jones, M. Mosca and R. H. Hansen, Implementation of a Quantum Search Algorithm on a Nuclear Magnetic Resonance Quantum Computer, in *Nature* 393 (1998) 344-346, and see also A. J. Jones and M. Mosca, Implementation of a Quantum Algorithm to Solve Deutsch's Problem on a Nuclear Magnetic Resonance Quantum Computer, in *J. Chem. Phys.* 109 (1998) 1648-1653

[118] Jozsa R 1997 Entanglement and quantum computation, appearing in *Geometric issues in the foundations of science*, Huggett S *et. al.*, eds, (Oxford University Press)

[119] Khalfin L A and Tsirelson B S, Quantum/Classical Correspondence in the Light of Bell's Inequalities. *Foundations of physics,* **22** No. 7, 879–948 July 1992

[120] Lord Kelvin, differential analyzer (1870), presented in the Science Museum of Aarhus, Denmark (1998)

[121] Keyes R W Science **168**, 796, 1970

[122] Kitaev A Yu, Quantum Computations: Algorithms and Error Corrections, in *Russian Math. Serveys*, **52**:6, 1191-1249

[123] Kitaev A Yu, Quantum measurements and the Abelian stablizer problem, in *LANL e-print* quant-ph/9511026, http://xxx.lanl.gov (1995)

[124] Kitaev. A. Yu Quantum error correction with imperfect gates, *Quantum Communication, Computing, and Measurement*, eds: Hirota, Holevo and Caves, 181–188, Plenum Press, New York, 1997.

[125] Kitaev A Yu 1997 Fault-tolerant quantum computation by anyons, in *LANL e-print* quant-ph/9707021, http://xxx.lanl.gov (1997)

[126] A. Yu. Kitaev, private communication

[127] Knill E and Laflamme R, Concatenated quantum codes, in *LANL e-print* quant-ph/9608012, http://xxx.lanl.gov (1996)

[128] Knill E, Laflamme R, and Zurek W, Resilient quantum computation, *Science*, vol 279, p.342, 1998.

[129] Knill E and Laflamme R 1997 A theory of quantum error-correcting codes, Phys. Rev. A **55** 900-911

[130] Knill E, Laflamme R and Zurek W H 1997 Resilient quantum computation: error models and thresholds in *LANL e-print* quant-ph/9702058, http://xxx.lanl.gov (1997)

[131] E. Knill, Non-Binary Unitary Error Bases and Quantum Codes, in *LANL e-print* quant-ph/9608048, http://xxx.lanl.gov (1996)

[132] Kondacs A and Watrous J On the power of Quantum Finite State Automata *38th Annal Symposium on Foundations of Computer Science*,(FOCS) 1997

[133] Kwiat P G, Mattle K, Weinfurter H, Zeilinger A, Sergienko A and Shih Y 1995 New high-intensity source of polarization-entangled photon pairs Phys. Rev. Lett. **75**, 4337-4341

[134] Laflamme R, Miquel C, Paz J P and Zurek W H 1996 Perfect quantum error correcting code, Phys. Rev. Lett. **77**, 198-201

[135] Landauer R. Is quantum mechanics useful? *Phil. Trans. Roy. Soc. of London*, 353:367–376, 1995.

[136] Landauer R 1961 IBM J. Res. Dev. **5** 183, and 1970 IBM J. Res. Dev, volume 14, page 152.

[137] Lecerf Y 1963 Machines de Turing réversibles . Récursive insolubilité en $n \in N$ de l'equation $u = \theta^n u$, où θ est un isomorphisme de codes, C. R. Acad. Francaise Sci. **257**, 2597-2600

[138] Leung D W, Nielsen M A, Cuang I L and Yamamuto Y, Approximate quantum error correction can lead to better codes in *Phys. Rev. A*,**56**, 1, 2567–2573, 1997

[139] van Lint J H Coding Theory, Springer-Verlag, 1982

[140] Lipton R J, Using DNA to solve NP-complete problems. *Science*, **268** 542–545, Apr. 28, 1995

[141] Lloyd S, Universal quantum simulators, *Science*, 273:1073–1078, 1996.

[142] Lloyd S 1993 A potentially realisable quantum computer, Science **261** 1569; see also Science **263** 695 (1994).

[143] Lloyd S 1995 Almost any quantum logic gate is universal, Phys. Rev. Lett. **75**, 346-349

[144] Lloyd S 1997 The capacity of a noisy quantum channel, Phys. Rev. A **55** 1613-1622

[145] Loss D and DiVincenzo D P Quantum Computation with Quantum Dots, in *Phys. Rev. A*,**57**,1, pp. 120-126, 1997

[146] MacWilliams F J and Sloane N J A 1977 *The theory of error correcting codes*, (Elsevier Science, Amsterdam)

[147] Mattle K, Weinfurter H, Kwiat P G and Zeilinger A 1996 Dense coding in experimental quantum communication, Phys. Rev. Lett. **76**, 4656-4659.

[148] Margolus N 1990 Parallel Quantum Computation, in *Complexity, Entropy and the Physics of Information, Santa Fe Institute Studies in the Sciences of Complexity,* vol VIII p. 273 ed Zurek W H (Addison-Wesley)

[149] Miquel C, Paz J P and Perazzo 1996 Factoring in a dissipative quantum computer Phys. Rev. A **54** 2605-2613

Miquel C, Paz J P and Zurek W H 1997 Quantum computation with phase drift errors, Phys. Rev. Lett. **78** 3971-3974

[150] Monroe C, Meekhof D M, King B E, Itano W M and Wineland D J. Demonstration of a universal quantum logic gate, *Phys. Rev. Lett.*, 75:4714-4717, 1995.

[151] N. F. Mott, The wave Mechanics of α-Ray Tracks, in *Proc. Roy. Soc. London*, A126, 79-84, (1929), and in *Quantum Theory and Measurement*, edited by Wheeler J A and Zurek W H, Princeton Univ. Press, Princeton, NJ (1983)

[152] Mukamel D, private communication

[153] von Neumann, Probabilistic logic and the synthesis of reliable organisms from unreliable components, in *automata studies(Shanon,McCarthy eds)*, 1956

[154] Nisan N and Szegedy M, On the degree of Boolean functions as real polynomials, *Proc. of the 24th Annual ACM Symposium on Theory of Computing (STOC)* 1992

[155] Nielsen M A and Chuang I L 1997 Programmable quantum gate arrays, Phys. Rev. Lett. **79**, 321-324

[156] Palma G M, Suominen K-A & Ekert A K 1996 Quantum computers and dissipation, Proc. Roy. Soc. Lond. A **452** 567-584

[157] Papadimitriou C H, Computational Complexity, Addison-Wesley, 1994

[158] J. Mod. Opt. **41**, no 12 1994 Special issue: quantum communication

[159] See also in this context: Y. Ozhigov, Quantum computers cannot speed up iterated applications of a black box, in *LANL e-print* quant-ph/9712051, http://xxx.lanl.gov (1997)

[160] Pellizzari T, Gardiner S A, Cirac J I and Zoller P 1995 Decoherence, continuous observation, and quantum computing: A cavity QED model, Phys. Rev. Lett. **75** 3788-3791

[161] Peres A 1993 *Quantum theory: concepts and methods* (Kluwer Academic Press, Dordrecht)

[162] Preskill J 1997 Fault tolerant quantum computation, in *LANL e-print* quant-ph/9712048, http://xxx.lanl.gov (1997), to appear in *Introduction to Quantum Computation*, edited by H.-K. Lo, S. Popescu, and T. P. Spiller

[163] Privman V, Vagner I D and Kventsel G 1997 Quantum computation in quantum-Hall systems, in *Phys. Lett. A*, 239 (1998) 141-146

[164] Rabin M O, Probabilistic Algorithms *Algorithms and Complexity: New Directions and Recent Results*, pp. 21-39, Academic Press, 1976.

[165] Rains E, Hardin R H, Shor P W and Sloane N J A, A non additive quantum code, Phys.Rev.Lett. 79 953–954 1997

[166] Rivest R, Shamir A and Adleman L 1979 On digital signatures and public-key cryptosystems, MIT Laboratory for Computer Science, Technical Report, MIT/LCS/TR-212

[167] J.J.Saqurai Modern Quantum Mechanics, revised edition. Addison Wesley, 1994

[168] L. J. Schulman and U. Vazirani in *LANL e-print* quant-ph/9804060, http://xxx.lanl.gov (1998),

[169] Schumacher B W and Nielsen M A 1996 Quantum data processing and error correction Phys Rev A **54**, 2629

[170] Shamir A 1979 Factoring Numbers in O(log(n)) Arithmetic Steps, in *Information Processing Letters 8(1)* 28-31.

[171] Shannon C E 1948 A mathematical theory of communication Bell Syst. Tech. J. **27** 379; also p. 623

[172] Shor P W, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM J. Comp.*, **26**, No. 5, pp 1484–1509, October 1997

[173] Shor P W, Scheme for reducing decoherence in quantum computer memory, *Phys. Rev. A*, 52: 2493-2496, 1995.

[174] Shor P W, Fault tolerant quantum computation, In *Proceedings of the 37th Symposium on the Foundations of Computer Science*, pages 56–65, Los Alamitos, California, 1996. IEEE press. quant-ph/9605011.

[175] Shor P W and Laflamme R 1997 Quantum analog of the MacWilliams identities for classical coding theory, Phys. Rev. Lett. **78** 1600-1602

[176] Simon J On feasible numbers, in *Proc. of the 9th Annual ACM Symposium on Theory of Computing (STOC)* 195-207, 1977

[177] Simon D 1994 On the power of quantum computation, *SIAM J. Comp.*, **26**, No. 5, pp 1474–1483, October 1997

[178] Simon D 1998, private communication.

[179] R Solovay and A. C-C Yao, preprint, 1996

[180] Steane A, Multiple particle interference and quantum error correction, *Proc. Roy. Soc. of London A*, 452:2551-2577, 1996.

[181] Steane A M, Error correcting codes in quantum theory, Phys. Rev. Lett. **77** 793-797, 1996, Simple quantum error-correcting codes, Phys. Rev. A **54**, 4741-4751, 1996, Quantum Reed-Muller codes, submitted to IEEE Trans. Inf. Theory (preprint in *LANL e-print* quant-ph/9608026, http://xxx.lanl.gov) Active stabilization, quantum computation, and quantum state synthesis, Phys. Rev. Lett. **78**, 2252-2255, 1997

[182] Steane A M The ion trap quantum information processor, Appl. Phys. B **64** 623-642 1997

[183] Steane A M Space, time, parallelism and noise requirements for reliable quantum computing, in *Fortsch. Phys.* **46** (1998) 443-458

[184] Stern A, Aharonov Y and Imry Y, "Phase uncertainty and loss of interference: a general picture" Phys. Rev. A **41**, 3436 (1990). and "Dephasing of interference by a back reacting environment" in "Quantum coherence" ed. J. Anandan, World Scientific, 1990.

[185] W. Tittel, J. Brendel, B. Gisin, T. Herzog, H. Zbinden, N. Gisin Experimental demonstration of quantum-correlations over more than 10 kilometers, in *Phys. Rev. A*,**57**, 3229 (1998)

[186] Toffoli T 1980 Reversible computing, in *Automata, Languages and Programming*, Seventh Colloquium, Lecture Notes in Computer Science, Vol. 84, de Bakker J W and van Leeuwen J, eds, (Springer) 632-644

[187] Turchette Q A, Hood C J, Lange W, Mabushi H and Kimble H J 1995 Measurement of conditional phase shifts for quantum logic, Phys. Rev. Lett. **75** 4710-4713

[188] Turing A M 1936 On computable numbers, with an application to the Entschneidungsproblem, Proc. Lond. Math. Soc. Ser. 2 **42**, 230 ; see also Proc. Lond. Math. Soc. Ser. 2 **43**, 544

[189] Unruh W G, Maintaining coherence in quantum computers, *Phys. Rev. A*, 51:992–997, 1995.

[190] Valiant, unpublished

[191] Valiant L. G, Negation can be exponentially powerful. *Theoretical Computer Science,* 12(3):303-314, November 1980.

[192] Valiant L G and Vazirani V V. NP is as easy as detecting unique solutions. *Theoretical Computer Science,* 47(1):85-93, 1986

[193] Vedral V, Barenco A and Ekert A 1996 Quantum networks for elementary arithmetic operations, Phys. Rev. A **54** 147-153

[194] Vergis A, Steiglitz K and Dickinson B , "The Complexity of Analog Computation", Math. Comput. Simulation 28, pp. 91-113. 1986

[195] Warren W S, *Science*, 277:1688–1698, 1997.

[196] Watrous J, On one Dimensional quantum cellular automata, *Complex Systems* **5** (1), pp 19–30, 1991

[197] Wiesner S 1996 Simulations of many-body quantum systems by a quantum computer in *LANL e-print* quant-ph/9603028, http://xxx.lanl.gov (1996)

[198] Wheeler J A and Zurek W H, eds, 1983 *Quantum theory and measurement* (Princeton Univ. Press, Princeton, NJ)

[199] A. Wigderson, private communication

[200] Wootters W K and Zurek W H 1982 A single quantum cannot be cloned, Nature **299**, 802

[201] Wootters W K A Measure of the Distinguishability of Quantum States *Quantum Optics, General Relativity, and Measurement* eds: Marlan O. Scully and Pierre Meystre, 145–154, Plenum Press, New York, 1983

[202] Yao A C-C, Quantum circuit complexity, in *33th Annual Symposium on Foundations of Computer Science(FOCS)*, (1993) pp. 352–361

[203] Zalka C, Efficient simulation of quantum systems by quantum computers *Proc. Roy. Soc. of London A*, in press, in *LANL e-print* quant-ph/9603026, http://xxx.lanl.gov (1996)

[204] Zalka C, Grover's quantum searching algorithm is optimal, in *LANL e-print* quant-ph/9711070 http://xxx.lanl.gov (1997)

[205] Zurek W H, Decoherence and the transition from quantum to classical Physics Today 44(10), October, 1991 36–44.