Adiabatic quantum computing and adiabatic algorithm design

Filip Němec

2019
July

Abstract

The adiabatic theorem in quantum mechanics enables us to derive new algorithms in quantum computation (QC) and gives us a new perspective on QC itself. Using the adiabatic theorem we are able to find ground eigenstates of a potentially complicated Hamiltonian H_1 by constructing a time dependent Hamiltonian $H(t)$. If $H_0 = H(0)$ is suitably chosen, its eigenstates may be constructed easily; the adiabatic evolution then uncovers the eigenstates of the examined Hamiltonian H_1. We introduce the basic concepts of adiabatic quantum computation (AQC), examine Grover's problem and show the relation between AQC and the adiabatic theorem in quantum mechanics. We will also relate AQC to the standard circuit model.

Keywords: Adiabatic evolution, adiabatic theorem, circuit model, Grover's algorithm, quantum computing, qubit, unitary gates, universal quantum computing.

Contents

1 Introduction, basic concepts

While trying to solve various kinds of *problems* we are usually interested in their *solution (or solutions)*. Even though this might sound as a completely obvious fact it is convenient to elaborate the statement more profoundly as it contains ambiguities and some implicit assumptions.

Firstly, we need to say what a problem means to us. In everybody's intuition there is a clear meaning to the word *problem*; it could represent questions such as how much is $2 + 20$, how to get from the airport to a hotel, what structure should a company network have etc. In our intuitive perspective problems are connected with question words: how (many, much), what, which etc.; although we are mostly able to *agree* with other people on their meaning such intuition is subjective and can be misleading. It has been the aim of formalization to give exact meaning to such questions; from contemporary perspective, formalization is the key concept in mathematics [1] [2]. In this text we will deal with formal problems or with formalized versions of *problems*. For example the problem of travelling form an airport to a hotel can be formally described in the language of the graph theory and the task is given as an optimization problem. We can translate intuitive questions to a formalized ones, try to find a solution in the language governed by the chosen formalism and then translate the solution(s) to something we can grasp (e.g the most convenient path on a map). We would need to make the statement 'the most convenient' more precise so that we could somehow capture the convenience; it could be the shortest walking route or the shortest way in terms of time spent in public transport. Even these simple reformulation enable us to specify the convenience formally.

Secondly, it is desirable to refine what is meant by *solution(s)*. Using the previous example: we may express the solution of the "most suitable path from the airport to the hotel"-problem as a path on a map or as a set of instructions (starting from the airport, go..., turn left etc.). We implicitly expect that there *is* some solution to the problem. Sometimes the existence of a solution itself is an interesting question; then we talk about satisfiability problems or SAT-problems. On the other hand the existence of a solution to our transportation problem is not very helpful. We would appreciate a particular solution (we wanted to find the optimal route) or even more solutions that are nearly optimal so that we can choose the one we pick up by our subjective preferences. The precise form of an answer has to be know prior to any attempts to find a solution to a problem given as we might end up with an unwanted answer or we can get more information than needed and pay for such a fanciness by unacceptable amount of time.

Time and resources consumption have not been discussed yet. After having formalized a given problem and having specified the desired kind (form) of answer we would like to know the solution as soon as possible. This is not always feasible: some problems are considered as *hard* and their solution takes a huge effort. Time and space consumption are crucial factors while trying to solve problems. We believe that problems may be classified in terms of time or space required in a useful way; such a classification is the subject of complexity theory.

Imagine that you pass by two chess players and one of them asks for *the best move* [1] he can play. Not only you have to get some insight in the game but you also have to check reasonable moves and imagine what the opponent might play. According to our intuition this is a hard problem; complexity theory confirms our expectations [3]. Further,

[1] Defining what is the best chess move is not an easy task.

we are not able to check our solution easily: deciding whether the move we proposed is the best is extremely complicated even when we know it. The game of chess, respectively its generalised version on $n \times n$ board, has been theoretically investigated. It was shown that for the $n \times n$ version exponential time resources are reqired in order to find the best chess move and even to confirm whether a given move is the best one, see [3]. Such problems belong to a class called EXP.

The problems we want to solve are sometimes scalable: the game of chess can generalized in such way that we allow different board sizes (or even rectangular chess boards). We can ask how does the necessary effort (time consumption) vary while using $n \times n$ chess boards where n is considered to be a free parameter. In a similar way we can generalize Sudoku: instead of playing Sudoku on $9x9$ board we can pick any number n and play the game on $n \times n$ board. We would like to get an estimate on required resources in terms of n and we expect n to be of a considerable value. Such an approach might not appear to be useful for any practical purpose but for theoretical study of problems (or games). This is not true: consider the shortest route from the airport problem. We use graph theory to describe the possible routes; the obvious relation between the size of city considered and size of graph needed for describing of the city leads us to considerations like: how does the size of graph affect the time that our algorithm consumes while we search for our route? The algorithm should work independently of the size of the city we landed in but its run time may vary. If the algorithm consumed so much time we could not run it effectively we might not be very pleased in big cities. For example in London there are about 60,000 streets or roads within the 6 mile radius of Charing Cross [4]. When we need to estimate required resources for problems of such a great size we usually do not need an exact number of steps. Having a nice upper estimate depending on the size of our problem seems more useful in practise. For instance, Dijkstra's algorithm solves the shortest path in a graph in approximately $c_1n^2 + c_2n + de + k$ steps for some constants c_1, c_2, d and k where n is the number of vertices and e the number of edges [5]. We also know that the number of edges e is less then n^2. The estimate we have given is still somehow complicated: for big n's the contribution from linear terms will be minor and the expression will be lead by c_1n^2. A suitable constant c such that from a certain n_0 we have: for all $n \geq n_0$ the number of steps for the Dijkstra algorithm is less than or equal to cn^2, can be surely found. This expression is more straightforward and does not hide any piece of information. The steps we have just made are studied in asymptotics, a branch of mathematics we shall briefly examine right now.

1.1 Bachmann-Landau O-notation

Begining with the expression for number of steps required to perform Dijkstra's algorithm

$$c_1n^2 + c_2n + de + k, \tag{1}$$

where c_1, c_2, d and k are some positive constants, n is the number of vertices and e counts the edges, we would like to estimate the error caused by replacing (1) by

$$cn^2, \tag{2}$$

where c is a constant. Dividing (2) by (1) we obtain the relative error E:

$$E = \frac{c_1n^2 + c_2n + de + k}{cn^2} = \frac{c_1}{c} + \frac{c_2}{n} + \frac{de}{n^2} + \frac{k}{n^2}. \tag{3}$$

We can use our upper bound estimate for e:

$$e \leq c_3 n^2. \tag{4}$$

By choosing $c = (c_1 + c_3)$ we estimate the upper bound for the relative error (3) as follows

$$E(n) \leq \frac{c_1 + c_3}{c} + \frac{c_2}{n} + \frac{k}{n^2} = 1 + \frac{c_2}{n} + \frac{k}{n^2}, \quad \text{thus} \quad |E(n) - 1| \leq \frac{c_2}{n} + \frac{k}{n^2}, \tag{5}$$

where we consider $E(n)$ as a function of the size of our problem. The relative error $E(n)$ is bounded from above by 1 for all naturals n. It is a basic result of calculus that for any $\varepsilon > 0$ we can find a natural number n_0 such that for all integers n greater than n_0 we have:

$$|E(n) - 1| < \varepsilon. \tag{6}$$

Thus, provided that we choose n big enough, we are guaranteed that the relative error E can be made arbitrarily small. Elaborating equation (5) we find a dependency of n_0 on ε as follows

$$n_0 = \left\lceil \frac{1}{2\varepsilon}(\sqrt{4k\varepsilon + c_2^2} + c_2) \right\rceil, \tag{7}$$

where the $\lceil \quad \rceil$ brackets denote the ceiling function that returns the smallest integer greater or equal to its argument. From this perspective our statement about arbitrarily small relative error may be refined: there is a function $n_0(\varepsilon)$ depending on ε such that for all integers n greater than $n_0(\varepsilon)$ equation (6) holds. All the requirements of the well known limit notation were met. The crucial part required for making an assumption about the limit for $n \to +\infty$ was *the existence* of a suitable function $n_0(\varepsilon)$. Even though we started with a numerical estimate (5) we used it only to show the existence of a function $n_0(\varepsilon)$. Furthermore, we wrote all the constants explicitly so that we demonstrated how cumbersome such a task can be even for very simple limits. (Majority of readers knew the trivial answer instantly; we investigated the example in a great detail and thus the clumsiness could be accented.) The resulting limit of (5) obviously does not depend on the constants c_2 and k: after having changed them we could use function $n_0(\varepsilon)$ given by (7) again.

Yet there is another way of expressing limit behaviour of functions. On one hand it does not suppress so much information as limits but it still remains useful in practice, the Bachmann-Landau O-notation.

Definition 1.1. A function $f(n)$ where n are naturals is said to be $O(g(n))$ if there is a constant c not depending on n such that $|f(n)| \leq c|g(n)|$ for all naturals n. We write $f(n) = O(g(n))$.

Example 1.1. *Equation* (5) *says that* $|E(n) - 1| = O(\frac{c_2}{n} + \frac{k}{n^2}) = O(\frac{c_2}{n})$ *for all* $n > n_0$.

Note 1.1. (i) In fact, we can go back to the (1) and say that:

$$c_1 n^2 + c_2 n + de + k = O(n^2), \tag{8}$$

where we profited from the definition 1.1: we do not have to specify any constant c as we did in our previous steps.

(ii) The equality sign may be slightly misleading: being $O(g(n))$ for a function $g : \mathbb{N} \to \mathbb{R}$ is a relation on functions. It is reflexive and transitive (such relations are called pre-order). We shall check this assertion. Surely, $g(n) = O(g(n))$, for the required constant from the definition 1.1 is 1; thus big-O relation is a pre-order. If it was symmetric we could conclude that we arrived at a relation of equivalence. This is not the case and we would have to add more restrictions if wanted a relation of equivalence.

(iii) We can generalize the definition 1.1: only functions on the naturals considered: $f : \mathbb{N} \to \mathbb{R}$. We could have chosen any reasonable domain and range. The only requirements are: we need to have a notion of convergence in the domain and we need to be able to compare size of elements in the range and lastly, we should be able to multiply functions by numbers; thus the domain may be any topological space and the range might be any normed space.

Based on previous notes the definition of big-O symbol may be generalised.

Definition 1.2. Let M be a topological space, N be a normed space, f, g are functions from $M \to N$. Let a be any point of M and let A be a set of all sequences $(a_n)_{n\in\mathbb{N}}$ converging to x. Then we say that the function f is $O(g)$ as x approaches to a if there is a constant c not depending on a and not depending on sequence $(a_n)_{n\in\mathbb{N}}$ chosen such that $|f(a_n)| \leq c|g(a_n)|$ for all naturals n greater that a certain n_0 and the constant n_0 depends only on the sequence considered. We write $f = O(g)$ as $x \to a$.

We could equivalently say that $f = O(g)$ as $x \to a$ if there exists a neighbourhood U of the point $a \in M$ such that for all $x \in U$ the inequality $|f(x)| \leq c|g(x)|$ holds.

The big-O relation can be used to estimate upper bounds on the number of steps required to perform an algorithm; we will see the importance of such an estimate in chapters concerning complexity theory. From the equation (8) may be seen that the Dijstra's algorithm is comparable to an algorithm that needs n^2 steps for a problem of size n. That means that Dijkstra's algorithm is an algorithm that takes at most polynomial number of steps depending on the problem size. Such problems are called P problems. This class will be further elaborated in the section dedicated to complexity classes 1.2.

We shall return to our study of asymptotics. After having seen the notion of boundedness of a function by another, one may want to introduce the notion of dominance. It was showed that $c_0 + c_1 n + c_2 n^2 = O(n^2)$ as n tends to $+\infty$. But $c_0 + c_1 n + c_2 n^2 = O(n^3)$ also holds. We can exploit the obvious difference between the two expressions: $\frac{c_0+c_1 n+c_2 n^2}{n^2}$ tends to c_2 as n goes to $+\infty$, on the other hand $\frac{c_0+c_1 n+c_2 n^2}{n^3}$ tends to zero as n goes to $+\infty$. We see that $c_0 + c_1 n + c_2 n^2$ is asymptotically as large as $c_2 n^2$ and neither of $c_2 n^2$ and $c_0 + c_1 n + c_2 n^2$ grows much faster than the other. Though the limit of $c_0 + c_1 n + c_2 n^2$ is dominated by n^3 in a stronger manner: n^3 grows much faster. This phenomenon gives rise to a definition of small-o symbol [6].

Definition 1.3. We say that a function $f(n)$ where n are naturals is $o(g(n))$ if for any constant ε there is n_0 such that for all $n > n_0$ $|f(n)| < \varepsilon|g(n)|$ holds. We write $f(n) = o(g(n))$ as $n \to +\infty$.

Adequate generalisations of the definition 1.3 can be easily derived if needed. We shall elaborate the definitions 1.1 and 1.3 more thoroughly:

$$f(n) = O(g(n)) \text{ as } n \to +\infty \iff (\exists c > 0)(\exists n_0)(\forall n > n_0)(|f(n)| \leq cg(n)), \tag{9}$$
$$f(n) = o(g(n)) \text{ as } n \to +\infty \iff (\forall c > 0)(\exists n_0)(\forall n > n_0)(|f(n)| < cg(n)), \tag{10}$$

where both f and g are functions from $\mathbb{N}$ to $\mathbb{R}$.

Note 1.2. (i) If the function $g(n)$ from the definition 1.3 is not vanishing as n tends to $+\infty$ the condition that $f(n) = o(g(n))$ may be restated as follows: $f(n)$ is $o(g(n))$ if the limit $\lim_{n\to+\infty} \frac{f(n)}{g(n)} = 0$. This can be easily seen from expressions in ((ii)). Some authors use this limit condition as definition of being small-o [6].

(ii) The comparison of big-O and small-o symbols in the equations and reveals that small-o is more restrictive than big-O.

The aim of this section was to give a precise definition of basic asymptotic notation so that we can beneficiate our discussion of complexities of given problems. We have already seen big-O and small-O symbols. Yet there is another useful notion. While introducing the big-O symbol we pointed out that rather than equality big-O is to be conceived as the pre-order relation. In fact, we may refine the definition of big-O and gain an equivalence relation.

Definition 1.4. We say that two functions f, g from naturals $\mathbb{N}$ are asymptotically equivalent if for any $\varepsilon > 0$ there is an n_0 such that for all $n > n_0 \in \mathbb{N}$ $|\frac{f(n)}{g(n)} - 1| < \varepsilon$ holds. We write $f(n) \sim g(n)$ as $n \to +\infty$.

How can one possibly come up with such a definition? Recall that according to the note 1.2 the big-O relation is reflexive and transitive however it is not symmetric. The definition 1.4 of asymptotic equivalence is clearly symmetric. It fact, if $f \sim g$, then $f(n) = O(g(n))$ and $g(n) = O(f(n))$ [6].

The definition of the asymptotic equivalence may be further generalised if needed. The latter definition 1.4 is suitable for our purposes: we will be interested in comparing number of steps of algorithms or a programs and these are often given in terms of naturals. Imagine generalised Sudoku with $m^2 \times m^2$ board, city with m streets where we would like to optimize routes etc.

Before moving to the most common complexity classes we will summarize basic relations between asymptotic symbols defined in this section and give few examples [6].

Example 1.2.

$$\cos x = O(1) \qquad (x \to +\infty), \tag{11}$$
$$\cos x = O(1) \qquad (x \in \mathbb{R}), \tag{12}$$
$$x^2 = O(x) \qquad (x \to 0), \tag{13}$$
$$x^2 + x + 15 = O(x^2) \qquad (x \to +\infty), \tag{14}$$
$$x^2 + x + 15 = o(x^3) \qquad (x \to +\infty), \tag{15}$$
$$x^2 + x + 15 \sim (x^2) \qquad (x \to +\infty), \tag{16}$$
$$(\forall k > 0)\left(\Big(f(x) + g(x)\Big)^k = O\Big(f^k(x)\Big) + O\Big(g^k(x)\Big)\right) \qquad (x \in S), \tag{17}$$
$$n! \sim e^{-n} n^n \sqrt{2\pi n} \qquad (n \to +\infty), \tag{18}$$
$$\cos(x) = 1 + o(x) \qquad (x \to 0), \tag{19}$$
$$x + 1 \sim x \qquad (x \to +\infty), \tag{20}$$
$$f(x) = g(x)(1 + o(x)) \iff f(x) \sim g(x) \qquad (x \in S), \tag{21}$$

where we denote by S the domain of real functions f and g. Note that in (12) we used $(x \in \mathbb{R})$ to express that the relation holds for any $x \in \mathbb{R}$. We shall express this fact more precisely: $(\forall x \in \mathbb{R})\Big(\cos y = O(1) \text{ as } y \to x)\Big)$.

1.2 Introduction to Complexity Theory

One feels that some tasks are harder than others. The field of complexity theory studies how to describe the hardness formally and how to classify certain problems according to their hardness into classes. In the previous section we mentioned the basics of the asymptotic notation. In this section we will use some of it to describe the dependence of resources needed to perform a computation depending on a problem size.

The dependence just stated lies at the heart of the complexity theory. Recall our discussion of being able to pose problems and measure their size. It was mentioned how important are our questions: do we ask for a certain solution to a given problem, or do we need to know how many of them are there? It seems plausible that these discrepances affect the resources required to find an answer. Imagine for example two similar questions: (1) is a natural number n composite or prime and (2) what are the factors of a natural number n. Both questions are found relevant in certain contexts (e.g. cryptography) and both take different time to solve: in 2004 M. Agrawal, N. Kayal and N. Saxena presented an improved version [2] of algorithm deciding whether a given n-bit [3] number x is prime, requiring $O(\log^{\frac{15}{2}} n)$ steps [8]. On the other hand we are not aware of any algorithm factorizing an n-bit number x in polynomial number of steps [2].

We shall return to the discussion of a problem size: in the paragraph above we had to somehow capture that it is harder to factorize big integers than small ones. We did so by counting number of bits in their binary representation which is the same as counting the number of digits in their decimal representation up to a constant factor. Such a factor may be absorbed while using the big-O notation and the results mentioned are

[2] They had published the original algorithm in 2002 [7].

[3] By counting bits of a number we mean the number of bits in binary representation.

independent of numeral system used (it would be impractical if one had to rely on specific numeral system while describing results in the complexity theory). Finding a suitable description of a problem size might be quite complicated and we should be certain that while reformulating our problems using different notation (e.g. numeral systems etc.) the size scales at most up to a *reasonable* multiple (e.g. constant). Then, using the big-O notation together with a suitable set of functions we may forget about such factors as they do not play any role (e.g. polynomial overhead and set of all polynomial functions in a given variable).

As we said earlier: the theory of computational complexity studies how the resources needed to solve a given problem depends on its size. What do we mean by resources? The latter example of primality testing gave us the following relation: size of a number – number of steps required to perform the task. We might also ask how many calculations do we need to remember during our computation; then we arrive at the question of memory resources given as: size of a number – size of memory [4] (While considering the primality testing or any other problem, the choice of different number of tapes results in at most polynomial overhead. In order to have *useful* classes of problems they should rather be invariant under such overheads.)

Other types of resources such as time necessary to perform an algorithm [5] or energy needed to power a computer while computing may be discussed but the latter two: number of steps and size of memory are usually the most important [9].

The meaning of "an algorithm requires $O(f(n))$ steps to solve a problem of size n" should be further elaborated. Does that mean that a program needs to perform $O(f(n))$ step not depending on its input? Certainly, not always! It means that at most $O(f(n))$ steps has to be made to obtain a solution but there might be certain inputs such that the program ends sooner. We will return to the question of primality testing for a moment. We can try to decide whether number 1768790 is composite or prime. How many steps did our answer take? And how many steps would be needed if the number given had 30 digits and was even? Someone can pose a question whether the AKS algorithm [8] performs like that. Even numbers greater than 2 will not certainly be primes. If the algorithm did not recognize such even numbers as not primes we could upgrade it: we would have to say whether the last digit is even and this takes just a few steps. That means we could add this procedure in the beginning of the original program without affecting the asymptotic complexity and in the best case we would be able to get results after constant number of steps. Usually, we take the worst case scenario while dealing with problem complexities: we try to find upper bound for number of steps and/or memory resources not thinking about how relevant such scenarios are. People sometimes need to capture average complexity of a problem (e.g. in cryptography [10] or while examining search problems) but the topic is complex and some theory is needed in order to define average hardness of a problem [9].

[4] Our model of memory should not have any impact on the complexity. In general we consider memory a place to store data together with read and (re-)write operations. In the classical theory of computational complexity we consider a machine equipped with (for example magnetic) infinite tape together with read/write head and set of instructions (internal states) called Turing machine. (See [9] for further details.) The memory of such a device consist of the tape and read-write operations. It can be shown that the resulting complexities are independent of number of tapes etc. [9].

[5] We will see that time dependency on problem size is convenient for describing algorithm for adiabatic quantum computers.

When we discussed the hypothetical upgrade of AKS algorithm we saw that the procedure that added constant number of steps did not affect its asymptotic complexity as $O(\log^{\frac{15}{2}} n) = O(\log^{\frac{15}{2}} n + c)$, where c is a constant. From this example can be seen how the basic properties of big-O notation help us describe complexities: the big-O blurs constant factors that may be present due to implementation details [6]. But here comes an important question we have not discussed yet: how does an architecture of a machine performing an algorithm e.g. a computer affect the resulting complexities? Defining complexities that would be too sensitive to an architecture design would result in not very clear theory. That is the reason why A. Turing invented an ingenious model of computation that captures just enough to be able to compute everything we consider computable but not too powerful. Turning's model is still interesting to study in context of nowadays computers, even though one may consider its concept rather simple. The details may be found in C. Papadimitriou's great book Computational complexity: [9]. Basically, the Turing's machine seems like a primitive computer [7]. It consists of a control device containing finitely many states and a tape equipped with moving head. We imagine labels on squares on the tape and regard read-write operations as moves of the head square by square, writing, erasing etc. Each square can contain either a letter from a given alphabet or a special symbol (blank space, the begin of the tape...). Each step goes like this: the machine reads a symbol at current position, possibly rewrites it with a different one and according to the current state of the control device switches the internal state and possibly moves the read/write head one square left or right. At this time we accomplished one cycle: the machine works in discrete cycles that consist of basic steps described above. Although the device seems elementary, we can see that it may perform any algorithm [9], [2]. Turing machine gives us a model of computation to which all classical computers are somehow similar not taking their architecture design etc. in account: any classical computer may be *efficiently* simulated by a Turing machine described above. We could introduce multi-tape machines and discuss the effects of choosing different alphabets etc., thorough discussion is available in [9], [2]. For our purposes we will end up with conclusion: there exists a Universal Turing machine such that any other Turing machine may be *efficiently* simulated using the Universal one. Thus we have a simple yet powerful model of computation described by a machine called the Universal Turing machine (UTM).

We have just used word *efficiently*. What does that mean? Formally, by efficiently we mean: if a program with input of size n requires $O(f(n))$ steps on machine A, we need to perform $O(p(f(n)))$ steps while simulating machine A using machine B, where p is a chosen function (e.g. polynomial or logarithmic). In other words: the simulation takes at most p-more steps than the original calculation.

The word *efficiently* in the context above is often meant as 'at most polynomial'. From where does such selection of function p arise? Is there any reason why should be a program that takes $10000 + n$ steps considered more efficient than o program requiring $\lceil 1.000001^n \rceil$ [8] steps? For $n = 10$ the first algorithm needs 10010 steps but the second will do its work in 1 steps. On the other hand, as the size of a given problem increases we find that from certain point the exponential dominates the polynomial. Whether the problem of size for which the difference matters is of any use is not the point; we would like to capture some

[6] if we exclude trivial cases like $O(0)$

[7] Not to be confused by the word primitive: Turing machine is as powerful as any classical computer we can construct.

[8] By $\lceil x \rceil$ we mean the smallest integer greater or equal to x.

sense of *efficiency* not depending on input size. There is also another reason why we do not consider $\lceil 1.000001^n \rceil$ as a problematic case: in practice we do not arrive at algorithms requiring such a strange number of steps [2].

The idea of simulation leads us to an interesting classification of problems. Imagine that given algorithm require polynomial number of steps on a Turing machine A. But here comes the great advantage of the UTM: we can simulate the Turing machine A using the UTM with at most polynomial overhead. So that if we were given *any* realisation of UTM we would know that anything the Turing machine A performs, the UTM performs with at most polynomial overhead.

So here emanates an idea: we can assume all the algorithms requiring at most polynomial number of steps on some Turing machine at once, as a class of problems, and the concept of simulation and UTM guarantees that the UTM may perform any such task in polynomial time. Further, we can see that such a class is closed under composition of problems.

We denote the class of problems solvable using a Turing machine in at most polynomial time [9] by P. This class is closed under the same operations under which the big-O of polynomial is. When someone needs to verify a solution given by an algorithm it is possible to rerun the algorithm and verify the result in polynomial time. This seems obvious. But there is another way how to define a well behaved class of problems, the class of NP [10] problems: such a problem may have resource expensive algorithm, we only demand the possibility to be able to check the solution in at most polynomial time. Clearly, $\mathrm{P} \subset \mathrm{NP}$. Whether P equals NP is an open question and we believe that the equality does not hold [11].

[9] We sometimes use time – number of steps interchangeably even though there is a difference in meaning. For any practical purposes the two aspects seems to be closely related.

[10] NP stands for nondeterministic polynomial.

[11] For interesting overview what would that imply in physics and mathematics one can consult the essay Why Philosophers Should Care About Computational Complexity written by S. Aaronson's [11]. Papadimitriou's Computational Compliexity offers more profound logical analysis [9]

2 Quantum Computing

Historically, there have been several different approaches to computing. We may start with intuitive concepts of computation and devices suitable for making computations more feasible for humans such as mechanical computers such as slide rule; then one usually meant computable in the sense computable using paper and a pen (and possibly the slide rule). First theoretical approaches to computability and computing theory proposed different formal models of computation and different points of view on what may be computed. We present a brief list of first theoretical concepts: Gödel's general recursive functions, Church's lambda calculus or Turing's concept of computing machines nowadays called Turning machines together with the universal computing machine (UTM). In 1936 and 1937 Church and Turing proved that the latter approaches to computing coincide, i.e. any λ-computable function is effectively computable on a Turing machine and it is also an element of Gödel's general recursive functions etc. According to the Church-Turing thesis any function on natural numbers $\mathbb{N}$ is computable, if and only if it may be computed using a Turing machine, and by the latter equivalences of computation models we get equivalent statements after interchanging the Turing model with Church's or Gödel's. Computable functions on natural numbers are computable by an algorithm (this can be seen most easily from the Turing's concept). Such an approach may be limiting in the sense that one's calculation is restricted to only finitely many discrete steps concerning elementary operations. We can try to weaken (or even abandon if possible) both restrictions (i.e. discreteness of steps and finite number of steps) as they may seem too strict [12]. What if we were allowed to perform computation based on a physical processes. Such a definition clearly absorbs the latter one as any physical realisation of a Turing machine is a concrete realisation of the physical process required. The other inclusion is not decided yet but it is believed that it does not hold [2]. Computers based on the weakened definition are called analog computers.

For physicist the possibility of being able to effectively simulate evolution of a physical system has been an important question. Predictions of classical physics (i.e. not quantum) are based on ordinary and partial differential equations, finding extremas of certain functionals etc. The classical analog computers are not very suitable for such tasks as they require extremely precise setting and control over external conditions (i.e. baths, shielding, mechanical isolation etc.) in order to provide sufficiently reliable results. Moreover the setting of the initial data may be complicated and our inability to gather precise control over all components [13] may result in poor precision [2]. The idea of using *the real world* reappeared in Feynman's article Simulating Physics with Computers from 1982 [12]; here was the the question of simulating physics, as the title suggests, the main topic. The difference from the previous analog computers was that Feynman suggested performing simulations of quantum systems using other quantum systems: what we call nowadays a quantum computer (QC).

The key difference between classical analog computers and quantum computers is that classical analog computations are governed by the laws of classical physics and their quantum counterparts operate in the framework of the quantum physics. As Feynman pointed

[12] Both, discreteness and finality, are discussed in Aaronson's essay [11] or in [2].

[13] Such inability is in practise often summarized as a presence of noise. For example, one can try to perform analog addition as addition of two DC voltages. How precise are our power supplies? Is the voltage really DC, or are there some oscillations? And what about the presence of voltmeters etc.?

out [12], classical analog computers, similarly as classical Turing machines, are not able to perform calculations regarding the quantum world effectively: classical devices inherently work in the classical framework governed by the classical physics [14].

In contrast to macroscopic analog computers, quantum experiments (and possibly calculations) can be done with high precision. Further, exploring limits of such calculations seems interesting because we believe that *our world* is described by the laws of quantum mechanics and these laws should pose the only limits in questions of what can be computed and what can be *computed effectively*. We shall return to the second topic – the effectivity – later. For now, we briefly examine the Church-Turing thesis and revisit the computability with regard on the quantum computing. We mention two basic aspects of quantum computability.

In 1985 Deutch wrote an article Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer where he pointed out that any quantum computer based on circuit made of gates [15] can be simulated using a Turing machine. The simulation would suffer from great overhead but any calculation could be performed in finite number of steps provided that the quantum circuit has finite number of gates. Thus the set of computable functions remains the same after having introduced the universal circuit model of quantum computing. Why are the quantum computers interesting then? The criterion of computability seems too coarse: both classical approach to computability and quantum approach lead to the same set of computable functions, thereby we need to find *something* finer: and here comes the complexity theory. Computer scientist found several very interesting quantum algorithm such as Shor's algorithm for integer factorisation, quantum Fourier transform and Grover's search algorithm that outperform any known classical algorithms (i.e. algorithms not based on quantum computing) [2] [13] [11]. We shall examine known results comparing the computation resources of classical and quantum computers later; the key point is that complexity theory is fine enough to capture the difference between quantum and classical.

We promised to present two aspects of Church-Turing thesis in regard to quantum computing. The second point of view concerns physical ability to compute and asks whether there might be a physical motivation for a thesis like that of Church and Turing. This topic has been studied classically by Gandy in the article Church's Thesis and Principles for Mechanisms from 1980 [14] and in regard to the quantum mechanics by Holevo (Information-theoretical aspects of quantum measurement from 1973 [15]) and later in 1977 by Prugovečki [16] who discussed the quantum theory of information and our ability to gain information from a measurement. In 2012 Arrighi and Dowek discussed the quantum version of Church-Turing thesis in their article The physical Church-Turing thesis and the principles of quantum theory [17]. They stated a set of *reasonable physically motivated arguments* under which we may restate a quantum version of the classical Church-Turung thesis.

[14] This reasoning also implies further discussion of the Church-Turing thesis for it was stated in the classical setting.

[15] We shall return to the question of the universal quantum computer later. Here we, rather anachronistically, use the terms quantum circuit and quantum gate. For now it suffices to mention: the model of quantum computing may be realized as a sequence of quantum gates – unitary operators – together with the set of ancilla bits set to zero – bits that enable us to perform some classical irreversible logical operations in reversible way. Such a model was shown to be universal [13].

2.1 Qubit

We shall review the basic principles underlying the description of the simplest non-trivial example of a quantum object: a binary quantum object, quantum version of a bit, called qubit. In order to exploit the advantage of quantum computing at least a bit, we will go through the description of a qubit rather thoroughly. For anyone who is familiar with basics of the quantum mechanics the description: two state quantum particle suffices. Such a description does not uncover why the usage of qubits might add any computational advantage [10].

We distinguish three basic concepts of computation: classical deterministic, classical non-deterministic, classical probabilistic and quantum. Each of them has many models and each model can be realized in many particular ways. Models help us to understand basic properties and basic steps of computation. We have already encountered the Turing machine as the example of the classical deterministic approach [16], we mentioned the circuit model as one approach to the quantum computing and we will see adiabatic model later. To have at least one example of non-deterministic classical computer, consider non-deterministic Turing machine [2]. Probabilistic Turing machine chooses at each step the following one according to a probability distribution [2].

Not paying attention to concrete models the four concepts of computation are distinguished in the way they treat knowledge and certainty: deterministic bit is set as either 0 or 1. Classical probabilistic bit may be set to 0 with the probability p and to 1 with probability $1-p$ provided that $p \in [0,1]$ [17]. Is there any other way how to capture the nature of the probabilistic bit? We may consider the probabilistic bit b as a (probabilistic) vector of the form $b = \begin{pmatrix} p \\ 1-p \end{pmatrix}$ together with assumptions that: the 1-norm of b is equal to 1 [18] and p is non-negative. Any operation mapping such a vector to a vector of the same form should be taken into account as a legal operation on one non-deterministacal bit. The most general linear transformations of probabilistic vectors are governed by stochastic matrices (i.e. matrices whose columns add up to the unity).

What if we did not use the 1-norm and considered a 2-norm for example [19]? Than a bit may be described by a vector $b = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$, where $\alpha^2 + \beta^2 = 1$. Two immediate questions arises: what do we gain by such a construction and how to connect the numbers α and β to a relevant outcome. If we were to obtain a piece of information resembling a probability we should map α and β to two non-negative numbers. But we set them in such way that their squares add up to 1 exactly, so that we can postulate: the probability of seeing the 2-norm bit in state 0 is α^2 and for the state 1 it is β^2. Note that α and β could be real or complex. But now the construction collapsed to the previous one, one could think. Not in general: the set of transformations on a 2-norm bit is different than for 1-norm bit [20]. At this time we introduce the quantum bit (qubit) as a 2-norm bit in a 2 dimensional

[16] Other models are for example: matrix model, Post Machine, λ-calculus etc.

[17] Fuzzy logic is based on similar assumptions [2].

[18] $\|b\|_1 = 1$ and $\|\cdot\|_1$ is equal to the sum of absolute values of vector entries)

[19] Or any other p-norm...

[20] I we chose p-norm for $p \notin \{1,2\}$ the set of resulting linear transformations would be restricted severely. In fact only the 1-norm and 2-norm provide interesting transformations. [10]

complex [21] Hilbert space [22]. The set of allowed transformations on a qubit are unitary transformations [23].

We present a few examples of logical operations on a probabilistic bit (1-norm bit) and a qubit (2-norm bit). At first we begin with bit-flip operation. Let b^1 be 1-norm bit of the form $b^1 = \begin{pmatrix} p \\ 1-p \end{pmatrix}$ [24]. Bit flip operation is governed by a stochastic matrix $M_f = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$. We see that:

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} p \\ 1-p \end{pmatrix} = \begin{pmatrix} 1-p \\ p \end{pmatrix}, \tag{22}$$

thus we obtain the expected result.

Not only the matrix M_f providing the bit flip is stochastic but it is also a unitary matrix [25]: $M_f \cdot M_F^* = \mathbb{I}$ [26]. For a qubit (2-norm bit) $b^2 = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ we may write:

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix}, \tag{23}$$

thus for the bit-flip works for both p-norms, where $p = 1, 2$ [27].

The main advantage of the 2-norm concept can be summarized in a phrase: quantum interference. We shall see that the possibility of having negative (and even complex) amplitude coefficients together with the set of unitary transformations *almost* enables us to switch between probabilistic and deterministic. Consider a qubit b^2 initialised in the state $b^2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ in a given basis $\mathcal{B} = (|0\rangle, |1\rangle)$ [28] and a transformation H governed by the Hadamard matrix of the form $\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$ [2]. The transformation of b^2 may be written

[21] It is highly non-obvious why we would not be able to describe physics just by the real numbers. In fact, we are. And we can choose different fields associated to the *physical space*. The advantage of complex numbers are mainly two: the field is algebraically closed (this reason is rather aesthetic) and the resulting description seems easier than while using other fields.

[22] One can ask whether the linearity of the underlying space does not restrict us too much. There has been non-linear approaches to the quantum mechanics but the linear model seems to agree with experimental data well (and it does not seem unacceptable intuitively). If one did not want to lean himself just against phenomenology he may be interested in theoretical aspects of linearity in context of quantum physics, e.g. Gleason's theorem and related theoretical works.

[23] We do not consider any transformations but the linear ones as our space is framed by linearity: we work in a Hilbert space.

[24] In this section we will denote the corresponding norm by upper indices, e.g. b^1 for a 1-norm bit

[25] Matrices that are both unitary and stochastic are called unistochastic. Easy examples are permutation matrices. The task of writing a general unistochastic matrix is not trivial, one may see for example [18].

[26] The adjoint of M is written as M^*

[27] The bit-flip actually works for any reasonable p.

[28] We adopt standard physical notation for basis vectors written as so-called ket-vectors $|\cdot\rangle$. Linear functionals are called bra-vectors and are denoted by $\langle\cdot|$.

as follows:

$$b^2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad \text{(the initial state)} \tag{24}$$

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \qquad \text{(the first transformation)} \tag{25}$$

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \cdot \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad \text{(the transformation } H \text{ applied once again).} \tag{26}$$

If we used bra-ket notation instead, the steps 24 to 26 would proceed: $|0\rangle \mapsto \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ and then $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \mapsto |0\rangle$. Observe that if we performed a measurement of the qubit b^2 in the basis $\mathcal{B}$, we would always obtain the value 0 corresponding to the ket $|0\rangle$, thus we would have a deterministic bit. However after having applied the transformation H to b^2 we would have a vector $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ with coefficients $\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$. The measurement postulate of the quantum mechanics [13] or our postulate of measurement of a 2-norm bit both implies that the resulting probability of measuring 0 is $\frac{1}{2}$. The interesting feature of qubits is that the second application of H results in deterministic state again (which is impossible in classical probabilistic setup).

The word almost in: "*almost* enables us to switch between probabilistic and deterministic" was italicised; it is necessary to take care about the selection of measurement basis. If we chose a different basis after each application of H we could get different results.

2.2 Quantum Gates and the Circuit Model

In order to obtain interesting (quantum) computation results it is convenient to introduce systems of more than one qubit and describe the set of allowed operations properly. We shall proceed as follows: at first we describe a multiple-quibits system, then we introduce two common models of quantum computing: the circuit and the adiabatic approach. Even though we are mainly focused on the adiabatic quantum computing, skipping the circuit model would be restricting as there are many interesting results based on this model and a straightforward way how to prove them for the adiabatic model is to show equivalences of both schemas.

We shall begin with the description of a multiple-qubits system [29]. Each measurement of a set of n qubits should yield a result containing n classical bits for if we choose to measure one separate qubit we receive a classical bit of information and having more of them should not change our observations principally [30]. How many results may we get? It is 2^n and an easy exercise for $n = 2$ and $n = 3$ enable us to see what they are. We have a correspondence of 2-tuples of bits and basis vectors (representing certain qubits in ket-notation), see tables 1 and 2.

[29] One who is familiar with postulates of the quantum mechanics already know that we need to consider tensor product space. We decided to show a semi-intuitive way of how to invent this construction

[30] Such an argument may be misleading for no one can easily predict how nature works. It is our intuition and possibly our endeavours to achieve something aesthetic while stating such assumptions. On the other hand we did not find (experimentally) any situation where such an assumption had not worked; in the quantum mechanics we state it as a postulate [19] [20].

classical bit string	corresponding ket
00	$\|00\rangle$
01	$\|01\rangle$
10	$\|10\rangle$
11	$\|11\rangle$

Table 1: Two bit strings and corresponding basis vectors.

classical bit string	corresponding ket
000	$\|000\rangle$
001	$\|001\rangle$
010	$\|010\rangle$
011	$\|011\rangle$
100	$\|100\rangle$
101	$\|101\rangle$
110	$\|110\rangle$
111	$\|111\rangle$

Table 2: Three bit strings and corresponding basis vectors.

Tables 1 and 2 suggests that the basis elements can be chosen as tensor products of 1-qubit basis elements. A general n-qubit state x is a superposition of basis states satisfying the normalizing condition (2-norm). We may write x as follows:

$$x = \sum_{b_1\in\{0,1\}} \sum_{b_2\in\{0,1\}} \cdots \sum_{b_n\in\{0,1\}} \alpha_{b_1 b_2 \ldots b_n} |b_1\rangle \otimes |b_2\rangle \otimes \cdots \otimes |b_n\rangle \tag{27}$$

$$= \sum_{b_1 \ldots b_n \in \{0,1\}^n} \alpha_{b_1 b_2 \ldots b_n} |b_1 \ldots b_n\rangle, \tag{28}$$

where α's satisfy the normalising condition:

$$\sum_{b_1\in\{0,1\}} \sum_{b_2\in\{0,1\}} \cdots \sum_{b_n\in\{0,1\}} |\alpha_{b_1 b_2 \ldots b_n}|^2 = 1 \tag{29}$$

and where we suppressed the tensor product sign and shrunk all kets to one "tensored" ket.

Note that a qubit is a unit element of $\mathbb{C}^2$ Hilbert space and allowed transformations are governed by 2×2 unitary matrices. A set of n qubits is represented by a unit element of $\left(\mathbb{C}^2\right)^{\otimes n}$ and allowed transformations are again the unitary [31] matrices. Two questions arise immediately: are the elements of the tensor product space the most general objects we may obtain by joining multiple qubits? Unitarity of available transformations implies that any computation we perform may be reversed. On the other hand irreversible operations are performed frequently using classical logical gates (e.g. **and** or **nand** operations). Does that mean we are not able to perform classical logical operations on qubits? We shall answer both questions briefly in the following paragraphs.

A further uncertainty of multiple-qubits states may be introduced as follows [10]: imagine we are not sure what kind of state we have, we may have some candidates but none of us is sure about a particular choice. Say $\frac{1}{2}$ probability of a state $\frac{1}{\sqrt{2}}(|1\rangle + |0\rangle)$ and $\frac{1}{2}$ probability of $\frac{1}{\sqrt{2}}(|1\rangle - |0\rangle)$. Such states are called mixed states and they may be represented

[31] Unitary with respect to inner product induced by the inner products of $\mathbb{C}^2$. We show how to establish the inner product on tensor product space of $\mathcal{H}_1$ and $\mathcal{H}_2$ equipped with respective inner products $\langle\cdot|\cdot\rangle_1$ and $\langle\cdot|\cdot\rangle_2$. Let $|a_1\rangle, |b_1\rangle \in \mathcal{H}_1$ and $|a_2\rangle, |b_2\rangle \in \mathcal{H}_2$; we set $(|a_1\rangle \otimes |a_2\rangle, |b_1\rangle \otimes |b_2\rangle) = \langle a_1|b_1\rangle\langle a_2|b_2\rangle$. The the mapping $(\cdot,\cdot) : (\mathcal{H}_1 \otimes \mathcal{H}_1) \times (\mathcal{H}_1 \otimes \mathcal{H}_1) \to \mathbb{C}$ satisfies all properties of the inner product. Similar construction may be applied for any finite number of tensor products.

bit a	bit b	a **and** b	a **nand** b
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Table 3: Operation of two bit gates **and** and **nand**.

by so-called density matrices: convex combination of outer [32] products of given vectors. For the latter case it would be:

$$\rho = \frac{1}{2}\left(\frac{1}{\sqrt{2}}(|1\rangle + |0\rangle) \otimes \frac{1}{\sqrt{2}}(\langle 1| + \langle 0|)\right) + \frac{1}{2}\left(\frac{1}{\sqrt{2}}(|1\rangle - |0\rangle) \otimes \frac{1}{\sqrt{2}}(\langle 1| - \langle 0|)\right) \tag{30}$$

$$= \frac{1}{4}(|1\rangle + |0\rangle) \otimes (\langle 1| + \langle 0|) + \frac{1}{4}(|1\rangle - |0\rangle) \otimes (\langle 1| - \langle 0|) \tag{31}$$

$$= \frac{1}{4}(|1\rangle + |0\rangle)(\langle 1| + \langle 0|) + \frac{1}{4}(|1\rangle - |0\rangle)(\langle 1| - \langle 0|) \tag{32}$$

$$= \frac{1}{4}\left(|1\rangle\langle 1| + |1\rangle\langle 0| + |0\rangle\langle 1| + |0\rangle\langle 0| + |1\rangle\langle 1| - |1\rangle\langle 0| - |0\rangle\langle 1| + |0\rangle\langle 0|\right) \tag{33}$$

$$= \frac{1}{2}(|1\rangle\langle 1| + |0\rangle\langle 0|). \tag{34}$$

Same steps can be done in matrix notation:

$$\frac{1}{2}(|1\rangle + |0\rangle) \otimes (\langle 1| + \langle 0|) = \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \tag{35}$$

$$\frac{1}{2}(|1\rangle - |0\rangle) \otimes (\langle 1| - \langle 0|) = \frac{1}{2}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}. \tag{36}$$

Thus we write the resulting mixed state as a matrix ρ:

$$\rho = \frac{1}{2}\frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} + \frac{1}{2}\frac{1}{2}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \tag{37}$$

$$= \frac{1}{4}\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{38}$$

The density matrix formalism governing mixed states may be seen as another probability layer: it is like stacking classical probability over the underlying 2-norm (quantum) probability [33].

At this point we return to the question of reversibility. Even 2-bit irreversible classical gates such as **and** or **nand** may be constructed easily; their operation is described by the table 3. Clearly, such classical gates are not realizable as quantum gates. We would not be building a promising computation model if it were not able to capture even basic logical

[32] Sometimes also called dyadic product. It in fact is a tensor product but some caution is necessary: we take a ket-vector $|a\rangle$ from a Hilbert space $\mathcal{H}$ and tensorize it with its dual $\langle a|$ which yields: $|a\rangle \otimes \langle a|$. The tensor-product sign is often not mentioned explicitly: $|a\rangle \otimes \langle a| = |a\rangle\langle a|$.

[33] One may ask whether such a construction is the ultimate one – the most general one. Certainly, it is not, for one can choose a random $p \in [1, +\infty)$, pick p-norm probability and layer it on top of the mixed states. On the other hand, we are not aware of any use of such construction; thus we may conclude that for all we need, mixed states are general enough.

blocks. Yet here comes a fundamental information-theoretical observation: irreversible gates may be replaced by reversible ones without a significant overhead [2].

We shall try to safe the reversibility of the logical gates in the table 4. Surely, gates mapping a certain number of inputs, say n, to two the same number of outputs will be needed. The reversibility is equivalent with a statement: the mapping is a bijection. If we chose n to be 2 we would not be able to succeed, for it is needed to distinguish three zeros (in the case of **and**) and only a single bit may be attached. After having pasted an additional bit we could map uniquely 3 states. But there are 4 of them in total thus for $n = 2$ there is no prospect of fulfilling our requirements. Let n be 3, table 4 summarizes our possibilities. The control bit c is sometimes called an ancilla bit. At this time, we

bit c	bit a	bit b	conditional a **and** b	g_1	g_2
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0	0		
1	0	1	0		
1	1	0	0		
1	1	1	1		

Table 4: After having introduced an additional control bit c we analyse possible outcomes. We opt to perform a **and** b provided that c is set to 1. How many reversible gates are available? Note that the condition may be reversed, i.e. we may perform a **and** b if $c = 0$.

have no option but applying combinatorics. There has to be 4 zeros and 4 ones in the column named conditional a **and** b. We have already used 3 zeros, so as soon as the last one is placed there is no freedom (permuting ones does not yield a different result). That accounts for 4 possibilities. Then we need to fill two 2×4 blocks with zeros and ones in such way that the mapping described by the table 4 is one to one. The blocks can be actually filled uniquely up to a row permutation, see table 5. We may count 4! permutations for each of the the blocks. In total we are left with $4 \cdot 4! \cdot 4! = 2,304$ available gates.

The introduction of a control bit c was a major step forward. Even though we considered only 2 bit operation **and** it was necessary to introduce the control bit in order to preserve the reversibility. Another important discrepancy should be noted: we are left with bits g_1 and g_2 that may be considered as "garbage".

bit g_1	bit g_2
0	0
0	1
1	0
1	1

Table 5: Other choices of filling the rest space in table 4 are permutations of the rows above.

bit i_1	bit i_2	bit i_1	bit o_1	bit o_2	bit o_3
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

Table 6: Operation of the Toffoli gate. Input bits are denoted by i_1, i_2 and i_3, output bits are marked as o_1, o_2 and o_3.

Yet it would not be very convenient to have multiple possible definitions for reversible gates derived from their classical counterparts (e.g. 2,304 possibilities for **and**, see table 4). That is why a canonical way of deriving reversible gates from classical ones has been invented. The task – transform an irreversible classical gate to a reversible one – may be reforumlated as follows: we have to be able to derive the initial state after having performed an operation. The input could be just stored and glued together with the original result [2].

Classical irreversible gates on n bits may be generated by a very restrained set of elementary gates: identity **id**, 2-bit **nand** together with usage of ancilla bits, copying and ability to direct any pair of bits to any part of the circuit provides all gates in general. For the case of reversible computing we need to select the so-called Toffoli gate, a three-bit permutation gate [13]. Its operation is summarized in the table 6; the generality has to be proven carefully, see [13] or [2] for further details.

We shall proceed directly to discussion of quantum gates and circuits [34] skipping the classical non-deterministic models as they are not directly related to our study of quantum computing and their description is rather complicated. On the other hand interesting results may be proven within different non-deterministic frameworks, see for example [21].

A straightforward approach to any computation model is to present available operations, give rules how to combine them and describe how they relate (e.g. unitary transformations in quantum computing). Natural questions like: are such models closed under given operations (they should better be)? Is there any *minimal* set of operations that allows us to perform any operation? Are there more such sets? And how complicated it is to perform an arbitrary operation using only the elementary ones, in other words: do we have a bearable overhead under such restrictions (e.g. polynomial overhead)?

Firstly, we examine the circuit model of quantum computation. This approach has several reasons: it resembles classical reversible computation and possibly even the classical deterministic computation. Another aspect is that historically, this model has been theoretically examined profoundly as the first universal model of quantum computing; see Benioff's article The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines from 1980 [22], Deutsch's articles Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer from 1985 [23] and Quantum Computational Networks from 1989 [24]. We

[34] We have not introduced the term quantum circuit formally. It is basically a composition of quantum gates.

will proceed along the lines sketched in Nielsen's Quantum Computation and Quantum Information [13].

Nielsen shows that any unitary operation may be expressed using Hadamard gate H, phase gate S, $\pi/8$ gate T and **cnot** gate to arbitrary accuracy. These gates are expressed by matrices:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \tag{39}$$

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \tag{40}$$

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}, \tag{41}$$

$$\mathbf{cnot} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{42}$$

The construction proceeds as follows: it is shown that any unitary $n \times n$ matrix may be decomposed to a product of so-called two-level unitary matrices. These are characterised by the property that they act non-trivially on a subspace spanned by at most two basis vectors, i.e. they affect at most two vector components. Then it is shown that any two-level matrix can be expressed as composition of single-bit operations and **cnot** gate. Lastly, Nielsen shows that any single-qubit operation can be approximated by the gates H, S, T to arbitrary accuracy (only the last step is not exact).

We will show the first step in more theoretical way than Nielsen. Let U be unitary. Then we can apply a sequence of Givens rotations [25] resulting in decomposition of the form: $U = G_1 G_2 \dots G_k \mathbb{I}$ where k is at most $\frac{n(n-1)}{2}$ [35] and $\mathbb{I}$ denotes the $n \times n$ identity matrix.

We perform a rather shortened (but clear) proof of the second step. Any Givens rotation act on at most two basis vectors, say v_α and v_β, which results in change in at most two vector components (α'th and β'th). If we were able to swap basis vectors in such way that the α'th and β'th vectors sit next to each other and they are, say, placed as the first two basis vectors, then there would be a single one-quibit unitary operator $\tilde{U}_G$ (2×2 matrix) corresponding to the initial Givens rotation G ($n \times n$ matrix). $\tilde{U}_G$ may be obtained as a restriction of G to the space in which G acts non-trivially.

So the task is to swap basis vectors, perform a controlled single qubit operation and then swap all the vectors back. But swapping is exactly what **cnot** does. For any Givens rotation we may estimate the switching process by at most $4n$ operations (imagine, we interchange the last two basis vectors with the first two and then back again). At this time we require at most $\frac{n(n-1)}{2} \cdot 4n = O(n^3)$ transformations. The latter construction is described in Nielsen's monograph [13] in more detailed manner; on the other hand, the idea remains the same.

We are approaching to the last step: the approximation of an arbitrary single qubit unitary gate. An explicit construction will not be presented here because it is rather insightful and requires a lot of technical details. It can be found in Nielsen's book Quantum Computation and Quantum Information [13] as Solovay-Kitaev theorem or in [26]. The

[35] For the case $n = 2$ we do not need any decomposition.

important result is: we are able to create an universal quantum computer using a finite set of gates of the form (39), (40), (41) and (42) to an arbitrary accuracy with at most polynomial overhead in the number of qubits (in comparison to the direct calculations using any unitary circuit). Do the gates H, S, T and **cnot** from (39), (40), (41) and (42) form a *minimal* set? Certainly not for we may write $S = T^2$. On the other hand there is no difference in considering 3 or 4 gates from the theoretical point of view, so why not to consider 4 of them and gain some flexibility. The key property is that the set is finite and reasonably small.

3 Adiabatic Quantum Computation

We have already seen a model of the quantum computation called the circuit model. In this section we will examine a different approach to quantum computing that resembles classical analog computing is some way: the adiabatic quantum computing (AQC). It is based on a straightforward idea: a solution of a computationally hard problem may be encoded as a ground state of a Hamiltonian, say H_1. Then we might be able to find a Hamiltonian H_0 with known ground state and evolve H_0 to H_1. Provided that the evolution is slow enough and we do not change the (non)-degeneracy of the ground state during the evolution, we will be able to track the path of the lowest energy eigenstates and possibly obtain desired results. This idea is formally governed by adiabatic theorems of the quantum mechanics (there are several versions and each is suitable under different circumstances).

The design of encoding a result of a computation as a ground state appeared in 1988 [27] in the article Quantum Stochastic Optimization written by Apolloni, Carvalho, and de Falco [28]. Their approach was later considered as a branch of quantum annealing (QA) [27]. QA was understood as a quantum counterpart to the simulated annealing [29]. Later, in 1999, Brooke et al. published an article Quantum Annealing of a Disordered Magnet where they investigated an implementation of QA in experimental configurations of disordered quantum ferromagnets. Their experiments provided a step towards construction of devices enabling to perform QA and this approach became interesting from the perspective of the quantum computing [27]. In 2000 Farhi et. al. proposed a solution of SAT-3 problem based on quantum adiabatic algorithm: an algorithm that performs calculations based on adiabatic evolution of the ground state of a suitably chosen Hamiltonian [30]. The term adiabatic quantum computing was introduced [27] by van Dam, Mosca and Vazirani at Symposium on Foundations of Computer Science in 2001 [31].

The initial phase of research in algorithm design has become surrounded by the theoretical background. In 2007 Aharonov et. al. answered in their article Adiabatic quantum computation is equivalent to standard quantum computation [32] fundamental theoretical questions like: is there any difference in computation resources between the adiabatic approach and the circuit model? And how to define the model of computation known as the adiabatic computing? What steps are needed to transform an algorthm design and optimisation technique to a model of computation? We shall see answers to all of them in the following paragraphs.

In order to define AQC it is suitable to introduce a k-local Hamiltonian [32].

Definition 3.1. We say that a hermitian matrix H acting on a space of p-state particles (e.g. binary particles) that may be written as $H = \sum_{i=1}^{r} H_i$, where each H_i acts non-trivially on at most k particles, is a k-local Hamiltonian. We suppose that $p \geq 2$.

The adiabatic quantum computing is defined as follows [32].

Definition 3.2. A k-local adiabatic quantum computation is specified by two k-local Hamiltonians H_0 and H_1 acting on p-state particles and by a description of the adiabatic evolution. We pose additional conditions on the Hamiltonian: the ground state of H_0 is unique and it is a product state. The adiabatic evolution is described by a constant T: total runtime and time-evolution function $s(t) : [0, T] \to [0, 1]$, an increasing smooth-enough bijection.

We set the time dependent Hamiltonian $H(s)$ to be

$$H(s) = (1-s)H_0 + sH_1.$$

Both T and $s(t)$ have to be adjusted in such way that the final state of the adiabatic evolution generated by $H(s) = (1-s)H_0 + sH_1$ at time T, say $|\psi(T)\rangle$ is ε close in l_2 norm to the ground state of H_1, say $|\varphi\rangle$: $\||\varphi\rangle - |\psi(T)\rangle\| < \varepsilon$, where ε is a parameter from $(0, 1]$ describing the precision of our computation.

The output of the adiabatic computation is the evolved vector $|\psi(T)\rangle$.

The latter definition 3.2 shall be commented in a few notes:

Note 3.1. (i) As soon as an adiabatic evolution is given (i.e. the function $s(t)$ is known), ε can be counted (for example as an estimate on the computation error). Usually, in practise, one would like to perform a calculation and given a parameter $\varepsilon \in (0, 1]$ design the evolution function $s(t)$. On the other hand, the computation itself is defined using only the evolution function $s(t)$.

(ii) Even though the definition 3.2 supposes non-degenerate ground states of the Hamiltonian $H(s)$ we may suppress this assumption, for the adiabatic theorem of the quantum mechanics holds also for degenerate spectra (but it is necessary to track the path not of eigenvector but rather that of the whole eigenspace belonging to the lowest energy). On the other hand Aharonov showed in [32] that the more general case is not required in order to obtain a computation model of the equal strength as the quantum circuit model.

(iii) The definition 3.2 supposes that only the evolution of the ground state is tracked. In practise it may be useful to track the evolution of excited states (e.g. the glued trees problem). Although, strictly speaking, such a computation does not belong to the class AQC defined 3.2, it is generally considered as a variation on AQC.

(iv) The evolution prescribed by 3.2 is not the most general. In fact, we are aware of examples where we need to use a different type. Aharonov proposed introduction of an "intermediate catalyst" – a Hamiltonian $H_c(s)$ depending on $s \in [0, 1]$ that satisfies: $H_c(0) = H_c(1) = 0$ [32]. Having introduced the catalyst Hamiltonian, we may be able to guarantee non-degeneracy of the first two eigenvalues from the spectrum of $H(s) := (1-s)H_0 + sH_1 + H_c(s)$ despite the fact that the ground eigenvalues of $H(s) := (1-s)H_0 + sH_1$ would become degenerate during the evolution.

Farhi et. al. investigated the use of an intermediate catalyst in the article Quantum computation by adiabatic evolution [30] while solving the SAT-3 problem with use of the adiabatic evolution.

(v) An intuitive way how to understand the value of k may be that it somehow manages the interconnection between degrees of freedom of the given system. So that it may be compared to memory requirements from the classical computing theory. During the time there has been investigation of the value k and its relation to computation resources. In 2007 Aharonov et. al. showed that $k = 6$ suffices to create a universal AQC. Then it was shown that k has to be at least 2, see [33] and [34].

Further details concerning the history of our knowledge of the impact of k to computation resources are summarized in [27].

While introducing the quantum model of computation (adiabatic or circuit) we hope that it may be possible to perform some tasks faster that in classical computing. In order to capture a speed-up Rønnow et. al. introduced various "orders" of speed-up [35]:

(i) A so-called "provable" quantum speed-up is the case where there exists a proof that no classical algorithm can outperform a given quantum algorithm. The best known example is Grover's search algorithm [36], which, in the query complexity setting, exhibits a provable quadratic speed-up over the best possible classical algorithm [37].

(ii) A "strong" quantum speed-up was originally defined by Papageorgiou and Traub [38] by comparing a quantum algorithm against the performance of the best classical algorithm, whether such a classical algorithm is explicitly known or not. This aims to capture computational complexity considerations allowing for the existence of yet-to-be discovered classical algorithms. Unfortunately, the performance of the best possible classical algorithm is unknown for many interesting problems (for example, for factoring).

(iii) A "quantum speed-up" (unqualified, without adjectives) is a speed-up against the best available classical algorithm (for example, Shor's polynomial-time factoring algorithm [39]). Such a speed-up may be tentative in the sense that a better classical algorithm may eventually be found.

(iv) A "limited quantum speed-up" is a speed-up obtained when compared specifically with classical algorithms that "correspond" to the quantum algorithm in the sense that they implement the same algorithmic approach, but on classical hardware. This definition allows for the existence of other classical algorithms that are already better than the quantum algorithm. The notion of a limited quantum speed-up will turn out to be particularly useful in the context of StoqAQC [36].

3.1 Adiabatic Theorem in Quantum Mechanics

In 1916 Ehrenfest stated a hypothesis that the laws of quantum physics would allow only motions invariant under adiabatic perturbations [41]. Later, in 1928, Born and Fock investigated adiabatic evolution governed by operators with discrete spectra [42]. Their results were enriched by Kato, who stated a rigorous version of the adiabatic theorem holding for operators with non-discrete and possibly degenerate spectra [43] satisfying that as time t tends to infinity, operators "consolidate" [37].

Firstly, we shall comment so-called "approximate" or classical versions of the adiabatic theorem. The attribute "approximate" arises from the fact that additional assumptions have to be added in order to make the statements valid. On the other hand, after doing so, the criteria become complicated and cumbersome. For critique of the classical statements see, for example: [44], [45] or [46]. Later, we will point where might be the weak points and state more robust alternatives. It is important to note that the first versions are applicable in practise for AQC because because the time dependency of suitable Hamiltonians is

[36] StoqAQC is a subclass of AQC. StoqAQC uses Hamiltonians of a specific kind: not only they are k-local but their matrices have real non-positive off-diagonal elements in the computational basis [40].

[37] The requirement for a suitable "consolidation" has to be added [44]; otherwise it is possible to construct Hamiltonians with a so-called oscillatory driving terms satisfying all the conditions imposed by Kato violating the adiabatic evolution. For further comments on Kato's proof and for examples of oscillatory Hamiltonians, see [44].

"nice". Second reason why to start with "approximate" statements is that historically, they came first and the refined rigorous versions we usually derived from them.

After the Born's and Fock's formulation for Hamiltonians with discrete and non-degenerate spectra Messiah presented in his monograph Quantum Mechanics [19] a generalised version for Hamiltonians with possibly degenerate spectra. Except for the degeneracy, the both statements are analogous. Let $|\varepsilon_j(t)\rangle$, where j ranges possibly over all naturals including zero $\mathbb{N}_0$, be an instantaneous eigenstate of $H(t)$ with energy $\varepsilon_j(t)$, i.e.

$$H(t)|\varepsilon_j(t)\rangle = \varepsilon_j(t)|\varepsilon_j(t)\rangle. \tag{43}$$

Assume ordering of the eigenstates; for all $j : \varepsilon_j(t) \leq \varepsilon_{j+1}(t)$. When we initialize the system subjected to the time dependent Hamiltonian $H(t)$ in one of the eigenstates $|\varepsilon_j(0)\rangle$, the evolved state will follow the path [38] of instantaneous eigenstates $|\varepsilon_j(t)\rangle$ for all times $t \in [0, T]$, where T denotes the total time of the evolution, provided that [39] [19]

$$\max_{t\in[0,T]} \frac{|\langle\varepsilon_i|\partial_t\varepsilon_j\rangle|}{|\varepsilon_i - \varepsilon_j|} = \max_{t\in[0,T]} \frac{|\langle\varepsilon_i|\partial_t H|\varepsilon_j\rangle|}{|\varepsilon_i - \varepsilon_j|^2} \ll 1 \quad \forall j \neq i. \tag{44}$$

In 2009, Amin found a way how to change the assumptions (44) for the adiabatic theorem to hold generally [46]. The evolution of a physical systems in the quantum mechanics is governed by the Schödinger equation [19], [20]

$$i\frac{\partial|\psi(t)\rangle}{\partial t} = H(t)|\psi(t)\rangle, \tag{45}$$

where we set $\hbar = 1$. Amin showed that a suitable reparametrisation of the time scale t may lead to a criterion that guarantees the validity of the adiabatic theorem analogous to the Messiash's (44) [19]. Let s be from the interval $[0, 1]$. We may see that the Schrödinger equation (45) can be rewritten using $s = \frac{t}{T}$ as follows

$$\frac{i}{T}\frac{\partial|\tilde{\psi}(s)\rangle}{\partial s} = \tilde{H}(s)|\tilde{\psi}(s)\rangle, \tag{46}$$

where $\tilde{H}(s) = H(sT)$ and $|\tilde{\psi}(s)\rangle = |\psi(sT)\rangle$. The variable s is called dimensionless time and we will further omit tildes while expressing dependency on it unless someone may be led to confusion.

The adiabatic theorem holds provided that the $H(s)$ parametrized by the dimensionless time s does not explicitly depend on T and if

$$\max_{s\in[0,1]} \frac{|\langle\varepsilon_i(s)|\partial_s H(s)|\varepsilon_j(s)\rangle|}{|\varepsilon_i(s) - \varepsilon_j(s)|^2} \ll T \quad \forall j \neq i. \tag{47}$$

For the interpolating Hamiltonians of the form

$$H(s) = sH_0 + (1 - s)H_1, \tag{48}$$

suitable for the AQC, we are usually able to guarantee the above conditions if the time scale s grows monotonically and provided that the final Hamiltonian H_1 does not depend on the runtime T [40].

[38] up to global phase factor

[39] It is also necessary to add a requirement that the Hamiltonian does not oscillate in time [46].

[40] When it does, additional investigation has to be performed.

Consider an example with Hamiltonian [27]

$$H(t) = a\sigma^z + b\sin(\omega t)\sigma^x \tag{49}$$

After substituting $H(t)$ from the equation (49) above to (44), the condition reduces to

$$|b\omega| \ll a^2. \tag{50}$$

Although we can set a and b in such way that the condition (50) holds, when $\omega \approx 2a$ a resonance occurs and the system undergoes Rabi oscillations with period $\frac{\pi}{b}$. Thus, for the total runtime $T \gg \frac{\pi}{b}$ additional increase in the upper bound for T does not help in satisfying the adiabaticity [27]. On the other hand Amin commented the adiabatic condition and its relevance for $T \ll T_R$, where T_R denotes the Rabi period [46].

Let us examine the refined version of the adiabatic theorem (47) on the example of oscillatory Hamiltonian (49); after having changed the variables $t \mapsto s$ with $s = \frac{t}{T}$ we obtain:

$$H(s) = a\sigma^z + b\sin(\omega T s)\sigma^x \tag{51}$$

and the Hamiltonian $H(s)$ depends on the total runtime T explicitly; thus we are not able to use the adiabatic theorem.

Recall the adiabatic condition (47); if the operator $\partial_s H$ is bounded [41] we may set $B(s) = \|\partial_s H\|$. Then we arrive at a useful bound for the total runtime T, $\forall j \neq i$:

$$\max_{s\in[0,1]} \frac{|\langle\varepsilon_i(s)|\partial_s H(s)|\varepsilon_j(s)\rangle|}{|\varepsilon_i(s) - \varepsilon_j(s)|^2} < \max_{s\in[0,1]} \frac{|\langle\varepsilon_i(s)|B(s)\varepsilon_j(s)\rangle|}{|\varepsilon_i(s) - \varepsilon_j(s)|^2} \tag{52}$$

$$< \max_{s\in[0,1]} B(s)\frac{1}{|\varepsilon_i(s) - \varepsilon_j(s)|^2} \tag{53}$$

$$= \max_{s\in[0,1]} B(s)\frac{1}{\varepsilon_{ij}(s)^2} \ll T, \tag{54}$$

where $\varepsilon_{ij}(s)$ stands for $|\varepsilon_i(s) - \varepsilon_j(s)|$. An (ij)-th spectral gap may be defined as

$$\Delta_{ij} = \min_{s\in[0,1]} \varepsilon_{ij}(s). \tag{55}$$

With the use of (55) the equation (52) can be further estimated as follows:

$$\max_{s\in[0,1]} \frac{1}{\varepsilon_{ij}(s)^2} B(s) < \max_{s\in[0,1]} \frac{1}{\Delta_{ij}^2} B(s). \tag{56}$$

An estimate on the operator $\partial_s H$ was used together with the fact that T is a lot greater than the left-hand side (52). An inverse-square dependence may bee seen in (52) and (56); from here arise names for such bounds: "inverse square spectral gap" etc [27].

Various more precise bounds may be found; the are usually of the form $O\frac{1}{T^n}$ for $n \in \mathbb{N}$ [47] or [48]. We will not present a proof of any of the refined alternatives as (52) and (56) suffices for our purposes. For a deeper analysis and alternative statements of the "inverse spectral gap" rule see, for example [48].

[41] Up until now we have been implicitly presuming existence of required derivatives and in cases where operators have been derived boundedness.

We will conclude this section by adding two remarks: one concerning a structural aspect of the proof, second regarding a possibility of achieving an arbitrarily small error in the resulting state.

Ad 1: While introducing the adiabatic theorem, a note regarding a phase factor was added. We shall return to the discussion of the phase evolution in this remark. Let $H(s)$ be a Hamiltonian [42] with an eigenvector $\psi(s)$. Physically, for any parameter κ, a vector $|\phi(s)\rangle$ defined as $|\phi(s)\rangle = e^{i\kappa s}|\psi(s)\rangle$ is equally suitable eigenvector, for the phase factor is not of observable significance. One may easily see that unless $\kappa = 0$ it is not possible to make $\psi(s)$ and $\phi(s)$ arbitrarily close in l_2 norm as desired in the adiabatic theorem. Several solutions are available.

(i) One approach is based on switching the investigated objects – vectors with physically insignificant global phase – to objects without such an attribute – for example, rays or projections. Then the adiabatic theorem becomes a statement about evolution of projectors to the eigenstates. This approach was investigated by Kato's proof from 1950 [43] and later by various authors: for instance [49] and [50].

(ii) The other approach is based on elimination of the "rotational" phase factor. A clear investigation of a suitable selection of complex factors is presented in [47]. We will only point a lemma from [47] enabling us to regard evolution of eigenvectors:

Lemma 3.1. *Let $\psi(s)$, $0 \leq s \leq 1$, be a time-dependent unit vector in some Hilbert space H, such that $\psi(s)$ is a differentiable function of s. Then, there is another time-dependent unit vector $\phi(s)$ that is identical to $\psi(s)$ up to phase such that $\langle \frac{d\phi(s)}{ds}|\phi(s)\rangle = 0$ and $\phi(0) = \psi(0)$.*

Using the lemma 3.1, Ambainis and Regev fix a certain phase factor: $\langle \frac{d\phi(s)}{ds}|\phi(s)\rangle = 0$ and $\phi(0) = \psi(0)$; after having imposed these conditions the adiabatic theorem holds for vectors [47].

Ad 2.: In the criteria (52) and (56) we imposed rather imprecise conditions on the total runtime T: we were only advised to pick T *a lot greater than something.* Is there any way how to make the estimate how precise is the resulting state?

These questions were investigated in Nenciu's work [49] and later in [51]. We will not go through the derivation of statements presented in the latter articles nor will we state them precisely as they are rather complicated and some theoretical background is needed.

The major result of [51] will be summarized in the following lines. Assume for simplicity that we investigate conditions for the two lowest eigenvalues and suppose further that $\varepsilon_0(s) = 0$, i.e. the eigenvalue corresponding to the ground state is zero during the evolution. Let the phase be fixed as in the lemma 3.1: $\langle \frac{d\varepsilon_0(s)}{ds}|\varepsilon_0(s)\rangle = 0$. Under these conditions the theorem 3.2 holds [51] [27]

Theorem 3.2. *Assume that all derivatives of the Hamiltonian $H(s)$ vanish at $s = 0, 1$, and moreover that it satisfies the following Gevrey condition: there exist constants $C, R, \alpha > 0$ such that for all $k \geq 1$,*

$$\max_{s\in[0,1]} \|H^{(k)}(s)\| \leq CR^k \frac{(k!)^{1+\alpha}}{(k+1)^2}. \tag{57}$$

[42] H(s) is considered to be a reparametrization of a Hamiltonian $H(t)$ that uses a dimensionless time.

Then the adiabatic error is bounded as

$$\min_{\theta}\||\psi(1)\rangle - e^{i\theta}|\varepsilon_0(1)\rangle\| \leq c_1\frac{C}{\Delta}\exp\left(\left(-\frac{c_2\Delta^3 T}{C^2}\right)^{1/(1+\alpha)}\right), \tag{58}$$

where $c_1 = eR(8\pi^2/3)$ *and* $c_2 = \frac{3/4\pi^2}{4eR^2}$.

Thus, as long as $T \gg \frac{C^2}{\Delta^3}$ and as long as the conditions of the theorem 3.2 holds and with latter prerequisites, the adiabatic error is exponentially small in T [27].

Further comments on the estimates of error is available in [27].

3.2 Adiabatic solution for problems of the type $f(x) = 1$

Many computationally interesting problems may be transformed to investigation of functions and especially questions concerning functions of the the type:

$$f : \{0,1\}^n \to \{0,1\}. \tag{59}$$

An easy way how to transform almost any problem to the form (59) is the following. We may consider set of all potential solutions (call it S) [43] as a space through which we perform a search. Let n be a large enough constant so that there is an injection ι from S to $\{0,1\}^n$; set A to be the image of S under ι: $A = \iota(S)$. Then we define the function f from (59) according to rule:

$$f(\iota^{-1}(a)) = \begin{cases} 1 \text{ iff } a \in A \wedge \iota^{-1}(a) = s \in S \text{ satisfies the original problem,} \\ 0 \text{ otherwise.} \end{cases} \tag{60}$$

The function ι is used only as translation between zeros and ones and the original problem space S. Such an approach is usually not the most effective one unless ι has a special form. It may exhibit some symmetries etc. The main aim of introducing f in (59) is that it allows to study only a very specific classes of tasks, namely investigating multi-variable binary functions, and we know that all reasonable tasks can be represented by a suitable choice of f.

In the following subsections we will present a few examples of the use of AQC. We will outline basic ideas behind the algorithm design but leave the technical details out.

3.2.1 Grover's algorithm

Suppose we are given an unsorted list of N items and our task is to find an element satisfying certain criteria. It is guaranteed that there is only one match in the list. A matching function $f : \{0,1,\ldots N-1\} \to \{0,1\}$ [44] is introduced; the single match may be formally expressed as follows:

$$(\exists m \in \{0,1,\ldots N-1\})(f(m) = 1) \quad \wedge \tag{61}$$

$$(\forall l, m \in \{0,1,\ldots N-1\})(f(m) = 1 \quad \wedge \quad f(l) = 1 \implies l = m). \tag{62}$$

[43] The size of the set usually depends on the size of our problem and it should be finite.

[44] The function f could be rewritten in a way that matches the definition (59) exactly but it is not necessary.

Note that as long as we do not know precise definition of f, its complexity can not be analysed. Our task is thus to find an algorithm querying f the minimum number of times. In this "black-box" view f plays a role of an oracle [13] whose implementation details are omitted and whose complexity is not counted in the complexity of the algorithm designed.

Classically, it would be necessary to query f $\frac{N}{2}$ times on average and at most $N-1$ times. Assume uniformity of the distribution of marked items in search lists [45], then:

$$q := \sum_{i=0}^{N-1} \frac{1}{N}(i+1) = \frac{N(N-1)}{2N} = \frac{N-1}{2} = O(N), \tag{63}$$

where we sum over all elements in $\{0, 1, \ldots N-1\}$ and adjust the counter by adding one.

The Grover's algorithm was originally developed in the framework of the circuit model of quantum computing based on a discrete sequence of unitary gates and it is able to perform the same task using only $\sqrt{N}$ queries [36]. In 1998 Farhi et. al. investigated in [52] a different approach: they applied a time-independent Hamiltonian H on a suitably chosen initial state for a time T. Their resulting complexity (measured as time required to perfom a computation) is of order $T = O(\sqrt{N})$.

Later, in 2000, Farhi et. al. used an adiabatic evolution in order to evolve a ground state of an initial Hamiltonian H_0 to a ground state of the final Hamiltonian H_1 encoding the solution to the computation problem [30]. In contrast to the method of time-independent Hamiltonian [52] the resulting complexity was of the same order as the classical approach. We shall present their steps in the following paragraphs and then try to improve the complexity with the use of optimized time schedule [53].

Assume an unsorted list of $2^n = N$ items together with a "quantum matching oracle" F are given. The Hilbert space of consideration has dimension N and each basis element may be written as $|i\rangle$, where $i \in \{0, \ldots, N-1\}$. The marked state, say $|m\rangle$, is not known, but we are given an oracle F represented by a self-adjoint operator [46]:

$$F = \mathbb{I} - |m\rangle\langle m| = \sum_{i=0}^{N-1} f(i)|i\rangle\langle i|. \tag{64}$$

One can easily check that f from (64) has two eigenvalues: 0 corresponding to $|m\rangle$ [47] and 1 corresponding to the rest of basis vectors. When one compares possible outputs of F chosen for the purpose of the adiabatic approach (64) with (59), it is clear that the current F is chosen to be a negation of (59). Why was F defined in such a "strange" way? While introducing the AQC we specified that the result is encoded as a ground state of a certain Hamiltonian. And the "negated" oracular function of the form (64) matches this criterion exactly. On the other hand, if we used the straightforward way as in (59), the result would be encoded as an excited state. Thus we choose the final Hamiltonian H_1 to be:

$$H_1 = F = \mathbb{I} - |m\rangle\langle m|. \tag{65}$$

Are we aware of any candidate initial Hamiltonian? Not a priori. But we may try to use the uniformity of the list given and the fact that the final Hamiltonian is a projector.

[45] A probability that a random item is marked equals $\frac{1}{N}$.

[46] In the quantum mechanics, any physically measurable function f is represented by a self-adjoint operator F. The outcome of evaluation of a function (or rather a measurement) is a value from the spectrum of F; for more profound analysis, one can see, for example, [13], [20] or any book studying the quantum mechanics.

[47] not yet known

And here comes an important note: the following choice of the initial Hamiltonian is "reasonable" and the computation works, but we have not deduced its shape in any way, we just guessed it (with some insight). Let a uniform superposition vector $|\psi_0\rangle$ be defined as

$$|\psi_0\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle; \tag{66}$$

then, the initial Hamiltonian H_0 is set to be:

$$H_0 = \mathbb{I} - |\psi_0\rangle\langle\psi_0| = \mathbb{I} - \frac{1}{N} \sum_{i,j=0}^{N-1} |i\rangle\langle j|. \tag{67}$$

Along the lines of [30] we opt for a linear time evolution of the time-dependent Hamiltonian:

$$H(t) = \left(1 - \frac{t}{T}\right) H_0 + \frac{t}{T} H_1, \tag{68}$$

or, after having set $s = \frac{t}{T}$:

$$H(t) = (1 - s)H_0 + sH_1. \tag{69}$$

T acts as a parameter describing the length of the adiabatic evolution and s is chosen to be a rescaled time.

We prepare the system in the state $|\psi(0)\rangle = |\psi_0\rangle$ and after applying the Hamiltonian $H(t)$ (68) for time T we obtain an evolved vector $|\psi(T)\rangle$. Provided that the spectral gap $g(t)$ between the lowest eigenvalues of $H(t)$ is large enough, we may estimate T. The highest eigenvalue of $H(t)$ is 1 with degeneracy $(N - 2)$ for $t \in (0, T)$ and degeneracy $(N - 1)$ at times $t = 0$ and $t = T$. Then there are two lowest eigenvalues E_0 and E_1 with degeneracy 1 (for $t \in (0, T)$). Time dependency of spectrum of $H(s)$ is plotted in the figure 1.

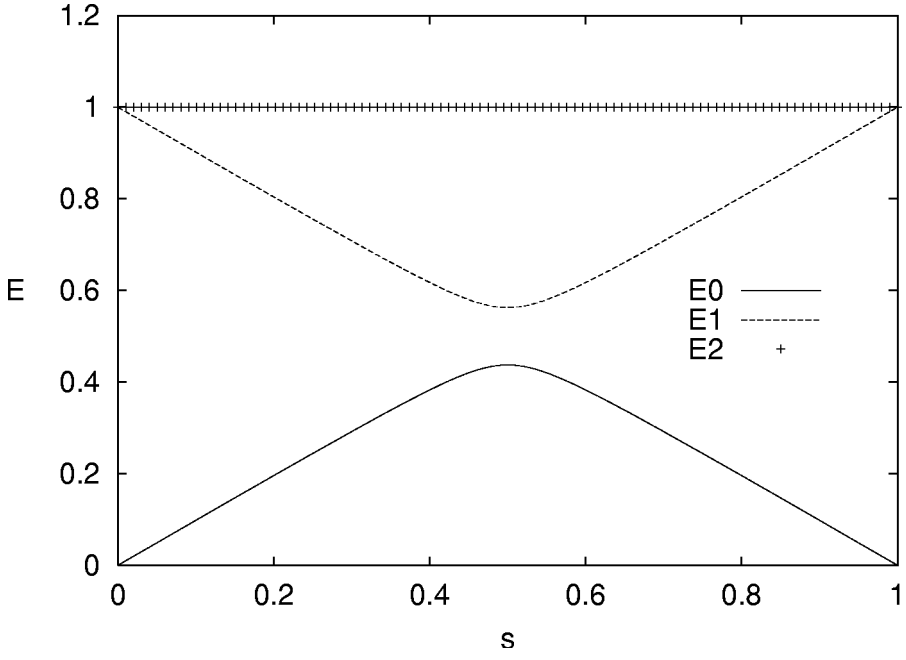

Figure 1: Eigenvalues of the time-dependent Hamiltonian $H(s)$ as a function of the dimensionless time s for $N = 64$. With the convention $\hbar = 1$, the energy is a dimensionless quantity. [53].

The two lowest eigenvalues are separated by a gap

$$g(s) = \sqrt{1 - 4\left(1 - \frac{1}{N}s(1-s)\right)}, \tag{70}$$

where we used the rescaled time s [30]. According to the adiabatic theorem [46] (47) we are guaranteed to obtain a reasonably close approximation of the final Hamiltonian H_1 ground state, provided that the spectral gap $g(s)$ is not small. In our case (70) $g(s)$ attains its minima of $g_{min} = \frac{1}{\sqrt{N}}$ at $s = \frac{1}{2}$.

The matrix element in the equation (52) may be expressed using:

$$\max_{0\leq t\leq T} \left\langle \psi_0(t) \left| \frac{dH}{dt} \right| \psi_1(t) \right\rangle = \left\langle \frac{dH}{dt} \right\rangle_{1,0} = \frac{ds}{dt} \left\langle \frac{dH}{ds} \right\rangle_{1,0} = \frac{1}{T} \left\langle \frac{dH}{ds} \right\rangle_{1,0}, \tag{71}$$

where $\psi_0(t)$ and $\psi_1(t)$ are eigenvectors corresponding to the eigenvalues E_0, E_1 respectively.

Note that we used dimensionless parametrisation and after a change of variables $t \mapsto s$ the Hamiltonian $H(s)$ does not explicitly depend on the total runtime T; we thus satisfied all the necessary conditions as stated in Amin's proof [46].

Equation (69) impies:

$$\frac{dH}{ds} = H_1 - H_0 \tag{72}$$

and thus $\left\langle \frac{dH}{ds} \right\rangle_{1,0}$ can be bounded as follows:

$$\frac{dH}{ds} = H_1 - H_0 \implies \left| \left\langle \frac{dH}{ds} \right\rangle_{1,0} \right| \leq 1. \tag{73}$$

Substituting the bound on $\left\langle \frac{dH}{ds} \right\rangle_{1,0}$ (74) and the formula for spectral gap $g(s)$ (70) into (47) we obtain

$$T \geq \frac{N}{\varepsilon} \tag{74}$$

which means that the time dependency of the adiabatic algorithm with Hamiltonian (68) is of order N. Hence we did not overcome the classical version of the search algorithm.

The linear time evolution proposed in (68) shall be reconsidered; a priori, there was not any reason why should a linear – a uniform – time evolution be used. The estimates we made restrict the speed of change of the Hamiltonian $H(s)$ in the same way for all times t although the spectral gap attains its minima only at $s = \frac{1}{2}$. What if we tried to account the non-uniformity of the spectral gap $g(s)$ (70) and look for an optimized time evolution?

The time interval $[0, T]$ can be split into smaller ones and the rate of the adiabatic evolution may be optimized on each of them separately. Note that if we chose a linear time evolution on each of the intervals separately, we would end up with a piecewise linear function. The more intervals we consider, the more suitable would the evolution rate be. In the limit case we arrive at a differential equation for $s(t)$:

$$\left| \frac{ds}{dt} \right| \leq \varepsilon \frac{g^2}{\left| \left\langle \frac{dH}{ds} \right\rangle_{1,0} \right|}, \tag{75}$$

for all times t. After substituting for $g(s)$ from the equation (70) and estimating $\left\langle \frac{dH}{ds} \right\rangle_{1,0}$ from above by 1 we obtain [48]

$$\frac{ds}{dt} = \varepsilon g^2(s(t)) = \varepsilon \left(1 - 4\frac{N-1}{N} s(t)\,(1 - s(t))\right), \tag{76}$$

where ε is a positive parameter smaller than one. Following the steps of [53] and integrating we are led to:

$$t(s) = \frac{1}{2\varepsilon} \frac{N}{\sqrt{N-1}} \left(\arctan\left(\sqrt{N-1}(2s-1)\right) + \arctan\sqrt{N-1} \right). \tag{77}$$

Having inverted $t(s)$ from the equation (77) we arrive at $s(t)$ as plotted in the figure 2 [53]. We may observe that the time dependency of $s(t)$ agrees with our intuition: its slope is the lowest around $s = \frac{1}{2}$, when the two lowest eigenvalues are close, and the highest at $t = 0$ or $t = 1$, when the eigenvalues are not very close.

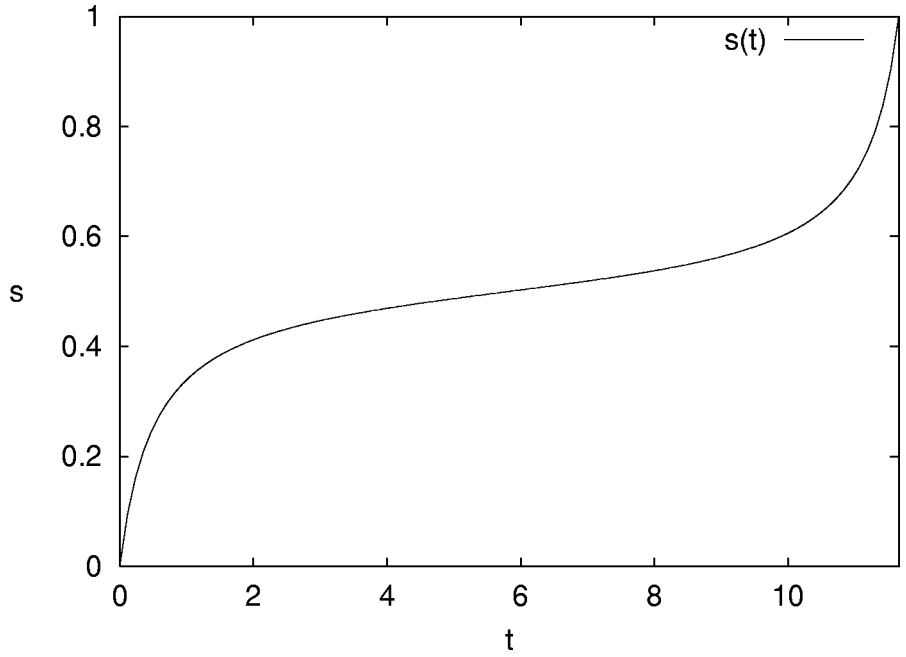

Figure 2: Dynamic evolution of the Hamiltonian that drives the initial ground state to the solution state: plot of the evolution function $s(t)$ for $N = 64$. The global adiabatic-evolution method of [36] would appear here as a straight line between $s(0) = 0$ and $s(1) = 1$. [53].

Setting $s = 1$ in the equation (77) leads to the total required time T

$$T = \frac{1}{\varepsilon} \frac{N}{\sqrt{N-1}} \arctan\sqrt{N-1}. \tag{78}$$

Estimating the equation (78) for N a lot greater than 1 we find an approximation on T

$$T = \frac{\pi}{2\varepsilon}\sqrt{N}, \tag{79}$$

[48] The absolute value on the left may be omitted due to monotonicity of $s(t)$.

which means that the optimized algorithm scales as $\sqrt{N}$. We thus improved the linear dependency on N (74) by switching from the linear schedule to non-linear one. According to [52] and [53] optimality was achieved.

Up until now only the case of single match has been considered. We will show that a straightforward extension to a case of multiple matches may be derived provided that the number of matches is known.

Let M be the number of matches; then the equation for the final Hamiltonian (65) transforms

$$H_1 = \mathbb{I} - \sum_{k \in S} |k\rangle\langle k|, \tag{80}$$

where S is the set of marked items – solutions to the search problem. The adiabatic is described by

$$H(s) = (1-s)H_0 + sH_1, \tag{81}$$

an equation analogous to (69). The multiple-marked-items case differs from the single-match case: the two lowest eigenvalues E_0, E_1 have degeneracy 1, eigenvalue $E_2 = 1$ is now $(N - M - 1)$ degenerated and lastly, there is a new $(M - 1)$ times degenerated eigenvalue E_3. The two eigenvalues E_0 and E_1 are separated by the spectral gap:

$$g(s) = \sqrt{1 - 4\left(1 - \frac{M}{N}\right) s(1-s)}. \tag{82}$$

The time dependency of the eigenvalues $E_0, \ldots, E_3$ is plotted in the figure 3.

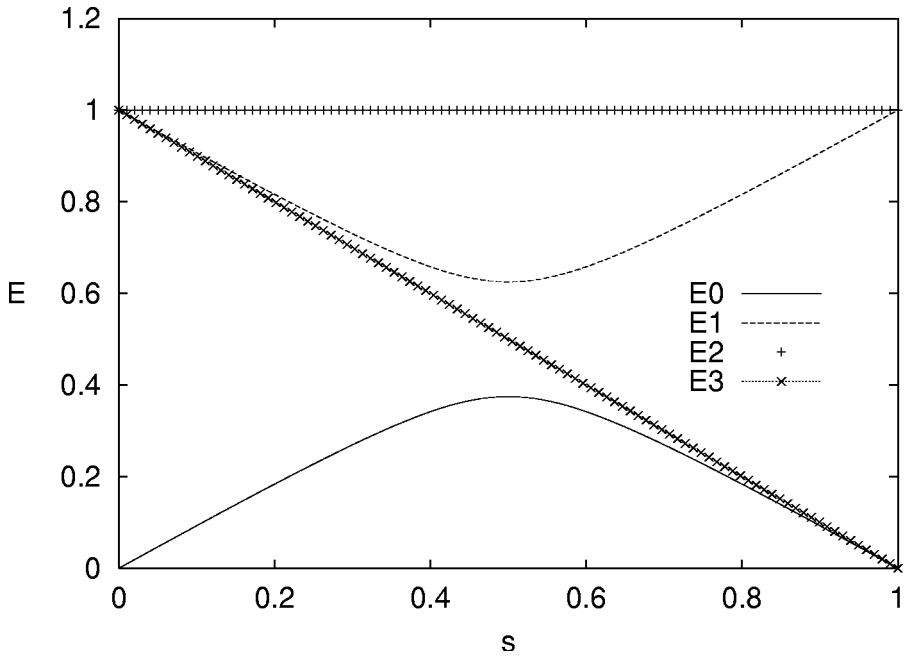

Figure 3: Eigenvalues of the time-dependent Hamiltonian $H(s)$ as a function of the reduced time s for $N = 64$ for the case of multiple marked items: $M = 4$. [53].

Clearly, the eigenvalue E_3 meets the E_2 at $t = 0$ and E_0 at $t = T$. Is the adiabatic theorem still applicable? Both H_0 and H_1 are invariant under permutation of any two solution states and under permutation of any two non-solution states. Thus the time dependent

Hamiltonian $H(s)$ (81) also exhibits the same symmetry. Due to this observation we see that the eigenstates belonging to E_0 and E_1 are uncoupled from the eigenstates belonging to E_2 and E_3. That is why we may use the same reasoning as in the single-marked-item case and by formal substitution $N \mapsto \frac{N}{M}$ in (70) we obtain the minimal spectral gap

$$g_{min} = \sqrt{\frac{N}{M}}. \tag{83}$$

The total time T required to perform the optimized adiabatic evolution 79 becomes

$$T = \frac{\pi}{2\varepsilon}\sqrt{\frac{N}{M}}. \tag{84}$$

3.3 Known Relations Between Different Models of Quantum Computing

We have summarized basic concepts of the adiabatic quantum computing in the previous subsections and presented the quantum circuit model in section 47. A straightforward questions arise immediately: how related are the two concepts? Has one model an advantage over the other?

In 2000, Farhi et. al. showed that the circuit model can efficiently simulate AQC [30]. Let $H(t)$ be a time dependent Hamiltonian of the form:

$$H(t) = (1 - \frac{t}{T})H_0 + \frac{t}{T}H_1, \tag{85}$$

We will find the time-evolution operator [49] of the physical system subjected to the Hamiltonian (85). The time evolution operator of a time-dependent Hamiltonian $H(t)$ may be written as

$$U(t_2, t_1) = \mathcal{T} \exp\left(-i \int_{t_1}^{t_2} dt H(t)\right), \tag{86}$$

where the symbol $\mathcal{T}$ is to be read as a time ordering symbol and where we set $\hbar = 1$ [20]. After having picked $t_1 = 0$ and $t_2 = T$ we arrive at a unitary operator $U(T, 0)$ describing the adiabatic evolution and due to the universality of the circuit model, we are guaranteed to be able to find a corresponding set of elementary unitary gates. Thus any adiabatic calculation may be performed using a circuit quantum computer. But how efficient is such a simulation?

An explicit construction of a sequence of operators converging to (86) is given by Farhi et. al. [30]. They assume a linear time schedule in the Hamiltonian (85) [50]. We will sketch the construction in the following lines.

A constant $M\mathbb{N}$ is picked and the time interval $[0, T]$ is split into M parts of the same length $\Delta = \frac{T}{M}$: $[0, T] = [0, \frac{T}{M}] \cup [\frac{T}{M}, \frac{2T}{M}] \cup \cdots \cup [\frac{(M-1)T}{M}, T]$. We approximate the time evolution on each interval separately; but instead of using time dependent Hamiltonian (85) we will assign to each interval a time-independent Hamiltonian H_m of the form:

$$H_m = H(m\Delta), \tag{87}$$

[49] Also called unitary propagator [20].

[50] If the schedule was non-linear, we could still perform a suitable change of variables resulting in linear schedule. This is guaranteed due to monotonicity of time schedules we imposed in the definition of AQC.

where m ranges through $\{1, 2, \ldots, M\}$. The m-th respective unitary propagator is then [51] of the form

$$U\left((m+1)\Delta, m\Delta\right) = \exp\left(-i\Delta H_m\right). \tag{88}$$

Thus, we replace $U(T,0)$ by $U_{app}^{(1)} = \prod_{m=1}^{M} U_m$ (we added a superscript 1 to denote the first step of the approximation). The task is to estimate the error introduced by such a discretization; van Dam, Mosca and Vazirani gave an estimate [31]:

$$\left\| U(T,0) - \prod_{m=1}^{M} U_m \right\| = \|U(T,0) - U_{app}^{(1)}\| = O\left(\sqrt{\frac{T \cdot \mathrm{poly(n)}}{M}}\right), \tag{89}$$

where n stands for the dimension of the underlying Hilbert space.

Right now we know that if M is large enough, the error incurred by discretization (89) can be made arbitrarily small; the important thing to note is that M has to grow as a polynomial times total runtime T.

Each individual term U_m may be approximated using only H_0 and H_1: due to the Baker-Cambpell-Hausdorff formula [20] we may write

$$U_m \mapsto \tilde{U}_m = \exp\left(-i\Delta\left(1 - \frac{m\Delta}{T}\right)H_0\right)\exp\left(-i\Delta\frac{m\Delta}{T}H_1\right), \tag{90}$$

which introduces an error proportional to the neglected leading commutator [52].

$$\|U_m - \tilde{U}_m\| = O\left(\frac{T^2}{M^2}\|H_0 H_1\|\right). \tag{91}$$

We set U_{app}^2 to be $U_{app}^{(2)} = \prod_{m=1}^{M} \tilde{U}_m$; the total error introduced by the latter sequence of approximations is [31]:

$$\left\| U(T,0) - \prod_{m=1}^{M} \tilde{U}_m \right\| = \|U(T,0) - U_{app}^{(1)}\| = O\left(\frac{T^2 \cdot \mathrm{poly(n)}}{M}\right). \tag{92}$$

We are thus able to begin with a time dependent Hamiltonian $H(t)$ (85) and find a sequence of M unitary operators $\tilde{U}_m$; if M scales faster than $T^2 \cdot \mathrm{poly}(n)$, we the error (92) can be made arbitrarily small. It is important to note that such M can still be chosen as a polynomial in T and n, thus the simulation of AQC using gates takes at most polynomial overhead.

The other way – showing that AQC can efficiently simulate the circuit model is rather complicated and requires few ingenious steps. The key fact is that it is possible to efficiently simulate the circuit model using AQC; a complete proof is described in detail in [32] by Aharonov et. al. A short but comprehensive review is available in [27].

As we have already seen, the circuit model is universal. The latter assertions lead us to a conclusion: AQC is up to a polynomial overhead equivalent to the circuit model and thus AQC is also universal.

[51] The time-independence of each H_m enables us to omit the time-ordering symbol $\mathcal{T}$.

[52] The Baker-Cambpell-Hausdorff formula states that: the solution Z of the equation $e^X e^Y = e^Z$, where X, Y, Z come from a possibly non-commutative algebra, may be expressed as $Z = X + Y + \frac{1}{2}[X,Y] + \ldots$ [20]. In other words, the formula expresses $Z = \log\left(e^X e^Y\right)$ as a series composed of X, Y and their repeated commutators. The question of convergence of such series (unless we restrict X and Y, they are rather formal) has to be discussed; for further details see [20] and for comments on our use in the proof see [31].

Yet there are other proposal how to perform quantum computations. We shall briefly present two other computation approaches, namely the Topological quantum computing [54] and measurement based quantum computing [55].

Ad measurement based QC: in the framework of measurement based QC a given fixed entangled state is our starting point. We then apply a sequence of measurements while keeping track of individual outcomes (outcomes are of the form of classical data). We can possibly use previous measurements to decide what to perform in the next step (e.g. what basis is chosen to perform a measurement). Such an approach differs principally from the models we have already seen: it is inherently irreversible.

Two principal schemes of measurement base computation have been developed: teleportation quantum computation [56] and so-called "one-way quantum computer" (1WQC) [57]. The relationship between the two designs is discussed in [58].

Even though it may, at first sight, seem that the inherent irreversibility disqualifies the measurement based QC from competition against the reversible models (e.g. AQC or circuit model), it is not the case. In 1999, Gottesman and Chuang showed that the measurement based approach is able to perform universal quantum computation [56].

Ad topological QC: the topological quantum computer is a theoretical computation model based on so-called braid gates [54]. It uses two-dimensional quasi-particles called anyons. Their world lines form braids [53] in a three-dimensional spacetime (we have two spatial and one temporal dimension). These braids are used as equivalents of logic gates and make up a computer [54].

Anyons are quasi-particles living in a two-dimensional space. In a three-dimensional space, we classify indistinguishable particles as fermions (obeying Fermi-Dirac statistics) and bosons (obeying Bose-Einstein statistics). Say, we have two particles $|\psi_1\rangle|\psi_2\rangle$ and that we exchange them; there should not be any observable difference, for our numbering of indistinguishable particles should not change any physical observables. Thus, $|\psi_1\rangle|\psi_2\rangle = e^{i\phi}|\psi_2\rangle|\psi_1\rangle$, where ϕ ranges in $[0, 2\pi)$. For fermions we have $\phi_f = \pi$ (i.e. $e^{i\phi_f} = -1$) whereas for boson $\phi_b = 0$ (i.e. $e^{i\phi_b} = 1$) [20].

In contrast, for two-dimensional space there need not to be such bifurcation: anyons, being interchanged, underlie a phase shift of a different value.

In 2002, Fredman, Kitaev and Larsen showed that topological quantum computing forms a model of universal quantum computer [54].

[53] Algebraically, braids were studied by Artin, see e.g. Artin's Algebra [59].

4 Further steps

Our aim was to present a brief introduction into the area of quantum computing, governing especially the adiabatic quantum computing. We believe that while discussing quantum computers, it is useful to begin with basic principles of computing, namely with classical computing theory, computability, complexity theory; and then move towards different classical computation models. We presented inherent differences in the treatment of certainty and knowledge of classical deterministic computers (Turing machines) and classical probabilistic computers. Such an approach enables us to introduce quantum computing as a model of computation leaning against the laws of quantum mechanics – a framework where information an knowledge obeys quite a different rules than we are used to observe in the macroscopic world. While comparing the three models we uncover principal differences and find possible advantages or disadvantages of each model. And here comes a motivation for further development of this article: it is our great desire to enrich the latter paragraphs by adding more theoretical details regarding computation theory (either classical or quantum) and by presenting further examples exploiting differences between particular models of computation.

Being an introductory text, this article does not provide much technical details nor does it examine the latter topics profoundly. We would be pleased to develop the major themes further; namely the discussion of complexity theory in regard to the quantum computing, theory of information, algorithm design, error correction, physical realisations etc. In our opinion, the world of quantum physics and quantum computing differs form classical subjects of physics, for it involves unimaginable wide area of mankind's knowledge connecting a variety of, at the first sight unrelated, subjects. In order to uncover at least some important relations, it is inevitable to change our attitude to its study (comparing to study of e.g. classical mechanics).

References

1. Bourbaki, N. *Théorie des ensembles* (Springer, 2006).
2. Pudlák, P. *Logical foundations of mathematics and computational complexity: a gentle introduction* (Springer Science & Business Media, 2013).
3. Fraenkel, A. S. & Lichtenstein, D. *Computing a perfect strategy for $n \times n$ chess requires time exponential in n* in *International Colloquium on Automata, Languages, and Programming* (1981), 278–293.
4. *London Taxi Services* `http://www.the-london-taxi.com/london_taxi_knowledge` (2019).
5. Dijkstra, E. W. A note on two problems in connexion with graphs. *Numerische mathematik* **1,** 269–271 (1959).
6. De Bruijn, N. G. *Asymptotic methods in analysis* (Courier Corporation, 1981).
7. Agrawal, M., Kayal, N. & Saxena, N. PRIMES is in P (2002).
8. Agrawal, M., Kayal, N. & Saxena, N. PRIMES is in P. *Annals of mathematics,* 781–793 (2004).
9. Papadimitriou, C. H. *Computational complexity* (John Wiley and Sons Ltd., 2003).
10. Aaronson, S. *Quantum computing since Democritus* (Cambridge University Press, 2013).
11. Aaronson, S. Why philosophers should care about computational complexity. *Computability: Turing, Gödel, Church, and Beyond,* 261–328 (2013).
12. Feynman, R. P. Simulating physics with computers. *International journal of theoretical physics* **21,** 467–488 (1982).
13. Nielsen, M. A. & Chuang, I. L. *Quantum Computation and Quantum Information* (2010).
14. Gandy, R. in (1980).
15. Holevo, A. S. Information-theoretical aspects of quantum measurement. *Problemy Peredachi Informatsii* **9,** 31–42 (1973).
16. Eduard, P. Information-theoretical aspects of quantum measurement. *International Journal of Theoretical Physics* **16** (5 May 1977).
17. Arrighi, P. & Dowek, G. The physical Church-Turing thesis and the principles of quantum theory. *International Journal of Foundations of Computer Science* **23,** 1131–1145 (2012).
18. Gutkin, E. On a multi-dimensional generalization of the notions of orthostochastic and unistochastic matrices. *Journal of Geometry and Physics* **74** (Dec. 2013).
19. Messiah, A. Quantum mechanics, volume II. *Appedix C (Section IV)(North-Holland Publishing Company, Amsterdam, 1969)* (1962).
20. Jirí Blank Pavel Exner, M. H. *Hilbert Space Operators in Quantum Physics Theoretical and Mathematical Physics* 2nd ed (Springer ; Melville, N.Y., 2008).
21. Hearn, R. A. & Demaine, E. D. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science* **343,** 72–96 (2005).

22. Benioff, P. The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. *Journal of Statistical Physics* **22** (5 May 1980).

23. Deutsch, D. Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer. *Proceedings Mathematical Physical & Engineering Sciences* **400** (1818 1985).

24. Deutsch, D. Quantum Computational Networks. *Proceedings Mathematical Physical & Engineering Sciences* **425** (1868 Sept. 1989).

25. Burden, R. L. & Faires, J. D. Numerical Analysis, Brooks. *Cole Pub* **7** (1997).

26. Dawson, C. M. & Nielsen, M. A. The solovay-kitaev algorithm. *arXiv preprint quant-ph/0505030* (2005).

27. Albash, T. & Lidar, D. A. Adiabatic quantum computation. *Reviews of Modern Physics* **90,** 015002 (2018).

28. Apolloni, B., Carvalho, C. & De Falco, D. Quantum stochastic optimization. *Stochastic Processes and their Applications* **33,** 233–244 (1989).

29. Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. Optimization by simulated annealing. *science* **220,** 671–680 (1983).

30. Farhi, E., Goldstone, J., Gutmann, S. & Sipser, M. Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106* (2000).

31. Van Dam, W., Mosca, M. & Vazirani, U. *How powerful is adiabatic quantum computation?* in *Proceedings 42nd IEEE Symposium on Foundations of Computer Science* (2001), 279–287.

32. Aharonov, D. *et al.* Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM review* **50,** 755–787 (2008).

33. Bravyi, S., DiVincenzo, D. P., Loss, D. & Terhal, B. M. Quantum simulation of many-body Hamiltonians using perturbation theory with bounded-strength interactions. *Physical review letters* **101,** 070503 (2008).

34. Kempe, D. A. D. G. S. I. J. The Power of Quantum Systems on a Line. *Communications in Mathematical Physics* **287** (1 Apr. 2009).

35. Rønnow, T. F. *et al.* Defining and detecting quantum speedup. *Science* **345,** 420–424 (2014).

36. Grover, L. K. Quantum mechanics helps in searching for a needle in a haystack. *Physical review letters* **79,** 325 (1997).

37. Bennett, C. H., Bernstein, E., Brassard, G. & Vazirani, U. Strengths and weaknesses of quantum computing. *SIAM journal on Computing* **26,** 1510–1523 (1997).

38. Papageorgiou Anargyros; Traub, J. F. Measures of quantum computing speedup. *Physical Review A* **88** (2 Aug. 2013).

39. Shor, P. *[IEEE Comput. Soc. Press 35th Annual Symposium on Foundations of Computer Science - Santa Fe, NM, USA (20-22 Nov. 1994)] Proceedings 35th Annual Symposium on Foundations of Computer Science - Algorithms for quantum computation: discrete logarithms and factoring* in (1994).

40. Bravyi Sergey; Terhal, B. Complexity of Stoquastic Frustration-Free Hamiltonians. *SIAM Journal on Computing* **39** (4 Jan. 2010).

41. Ehrenfest, P. Adiabatische invarianten und quantentheorie. *Annalen der Physik* **356,** 327–352 (1916).

42. Born, M. & Fock, V. Beweis des adiabatensatzes. *Zeitschrift für Physik* **51,** 165–180 (1928).

43. Kato, T. On the adiabatic theorem of quantum mechanics. *Journal of the Physical Society of Japan* **5,** 435–439 (1950).

44. Marzlin, K.-P. & Sanders, B. C. Inconsistency in the application of the adiabatic theorem. *Physical review letters* **93,** 160408 (2004).

45. Tong, D., Singh, K., Kwek, L. C. & Oh, C. H. Quantitative conditions do not guarantee the validity of the adiabatic approximation. *Physical review letters* **95,** 110407 (2005).

46. Amin, M. H. S. Consistency of the Adiabatic Theorem. *Physical Review Letters* **102** (22 June 2009).

47. Ambainis, A. & Regev, O. An elementary proof of the quantum adiabatic theorem. *arXiv preprint quant-ph/0411152* (2004).

48. Jansen, S., Ruskai, M.-B. & Seiler, R. Bounds for the adiabatic approximation with applications to quantum computation. *Journal of Mathematical Physics* **48,** 102111 (2007).

49. Nenciu, G. Linear adiabatic theory. Exponential estimates. *Communications in mathematical physics* **152,** 479–496 (1993).

50. Elgart, A. & Hagedorn, G. A. A note on the switching adiabatic theorem. *Journal of Mathematical Physics* **53,** 102202 (2012).

51. Ge, Y., Molnár, A. & Cirac, J. I. Rapid adiabatic preparation of injective peps and gibbs states, 2015. *arXiv preprint arXiv:1508.00570.*

52. Farhi Edward; Gutmann, S. Analog analogue of a digital quantum computation. *Physical Review A* **57** (4 Apr. 1998).

53. Roland Jérémie; Cerf, N. J. Quantum search by local adiabatic evolution. *Physical Review A* **65** (4 Mar. 2002).

54. Freedman, M., Kitaev, A., Larsen, M. & Wang, Z. Topological quantum computation. *Bulletin of the American Mathematical Society* **40,** 31–38 (2003).

55. Jozsa, R. An introduction to measurement based quantum computation. *NATO Science Series, III: Computer and Systems Sciences. Quantum Information Processing-From Theory to Experiment* **199,** 137–158 (2006).

56. Gottesman, D. & Chuang, I. L. Quantum teleportation is a universal computational primitive. *arXiv preprint quant-ph/9908010* (1999).

57. Nielsen, M. Universal quantum computation using only projective measurement, quantum memory, and preparation of the— 0¿ state, 2001. *arXiv preprint quant-ph/0108020.*

58. Raussendorf, R. & Briegel, H. J. Quantum computing via measurements only. *arXiv preprint quant-ph/0010033* (2000).

59. Artin, M. *Algebra* United States ed (Prentice Hall, 1991).