Institut für theoretische Physik, Leibniz Universität Hannover,
2022

Quantum search by measurement for quantum optimization problems

Bachelor thesis for the degree *BSc. Physik* submitted by

Younes Naceur

First Examiner: Prof. Dr. Tobias J. Osborne

Date of delivery: 23.08.2022

Contents

1 Introduction

Due to the ascending complexity of optimization problems in many fields of natural sciences, like physics, biology, and chemistry, but also in other research fields like computer science, economics, or finance, modern research heavily depends on computational methods for approximating solutions to these problems, that are not analytically solvable anymore. The scale of the problems we can investigate on therefore heavily depends on the computational resources provided.

For some fields of science, like condensed matter, or quantum chemistry, it becomes very hard to study given systems due to their size and behaviour induced by quantum mechanics, using classical computers. For other fields like economics and finance, the evergrowing number of variables influencing the quantitative outcome demands great computational effort for respecting all the crucial factors and correlations. In fact, it has been shown, that some problems won't be solved in a time shorter then the remaining lifetime of the universe [1]. This yields the idea of using another ansatz than classical computers, so called quantum algorithms, that exceed the computational power provided by classical computers [2], for solving optimization problems.

In this work, we're going to apply methods from a paper by Childs et. al, *Quantum search by measurement* [3], that proposed a quantum algorithmic framework based on the measurement scheme by von Neumann [4], which is applied to Grover's search algorithm [5] originally, to solve arbitrary optimization problems encoded into a Spin Glass Ising model. We're going to investigate on its accuracy in finding the ground state of a given Ising model Hamiltonian in dependency on the parameters introduced by Childs et al.

1.1 Introduction to Quantum Information Theory

To understand the quantum algorithms that are necessary for optimizing solutions with the aforementioned quantum search by measurement algorithm, it's important to know the theory underlying quantum information. Therefore, the next sections, that are based on the book [6], will provide an overview about the basic concepts and implementations of quantum information and computing.

1.1.1 Advantages of quantum computation

To start with the introduction to quantum information and computation, we first want to provide some intuition and knowledge on how computers whose workflow bases on the concepts of quantum mechanics differ from classical computers, and how these differences result in advantages at solving certain tasks in comparison to their classical counterpart.

In software engineering, one crucial part about algorithms is the resources used in the processes, namely time and memory. There are always different kinds of implementations for solving the same task, but they may fundamentally differ from each other in terms of resources. The complexity of an algorithm, that represents the amount of resources that are consumed, is always given as a function of the size of the input, denoted as a natural number n which represents e.g. the number of characters in a string, or the length of a list that's fed into the algorithm. To denote a (classical) algorithm's running time complexity, we use the *Landau* notation, which was introduced by Edmund Landau [7]. For a given algorithm, we define the function $f(n)$ that characterizes its running time, giving us the number of operations needed to complete the task. Therefore, $f(n)$ is independent of the actual device

the algorithm is performed on and serves as a tool to compare algorithms regardless of the performing computer.

If an algorithm's number of operations involved doesn't grow faster in the size of input n than a given reference function $g(n)$, we write

$$f(n) \in \mathcal{O}(g(n)) \tag{1}$$

Where we always give the slowest growing $g(n)$ to characterize $f(n)$, to be as precise as possible. This gives an upper bound for the running time and lets us investigate how long the algorithm will take in the worst case scenario. For example, for an algorithm with linear running time, like the search problem in an unstructured list, that will be discussed later on, we write $f(n) \in \mathcal{O}(n)$.

One problem that can be solved substantially faster with quantum computers than with classical ones is the factorization of numbers into their prime factors [8], which is an encryption a lot of today's cyber security relies on because of its complexity using classical algorithms. Using Shor's algorithm instead, the corresponding quantum algorithm that's not going to be discussed further in this work, one can solve these problems fundamentally faster, which implicates a huge impact towards the security on computers. Due to the fact that quantum computer's that are able to perform Shor's algorithm on a relevant scale don't exist yet, the algorithm can not yet yield advantages in comparison to its classical counterpart.

It has been proven mathematically, that algorithms that exploit quantum phenomena, so called quantum algorithms, can generally exhibit the speed of classical computers by magnitudes [9]. For example, Google claimed to have constructed a quantum computer, which is able to solve problems that classical computers need 10000 years for, in around 200 seconds [10]. This lets us see directly, that using quantum phenomena for computation yields the possibility for a big step forward in terms of computational resources. The fact that quantum computation yield these huge advantages leads us to the idea, that optimization problems can be solved with significantly less time complexity, allowing us to increase the number of variables interacting and therefore the given system size of the problems.

1.1.2 From bits to qubits

The fundamental building blocks of a classical computer are bits, which are represented mathematically as a binary number $k \in \{0, 1\}$ and implemented technically through transistor currents being switched on or off. Bits represent the smallest unit of memory, and classical algorithms are formulated upon them: An algorithm in its easiest form takes an n-bit input and performs arithmetical operations on it, returning a result that's again represented by a sequence of bits, e.g. a binary number.

Opposed to the classical analogue, quantum computers work with so called *qubits* instead of bits: Quantum objects with Hilbert dimension $d(H) = 2$, whose basis vectors $|i\rangle$ with $i \in \{0, 1\}$ represent the classical analogue bit states. Actually, DiVincenzo et al. proposed seven criteria, with which quantum system's can be used as qubits, allowing us to perform quantum computation [11].

If no other implementation is mentioned, we usually refer to the Hilbert space of a spin-$\frac{1}{2}$ particle with the two eigenstates of the S_z operator $|\uparrow\rangle$ and $|\downarrow\rangle$, which refer to the magnetic spin quantum number of the particle $m_{\pm 1/2} = \pm\frac{1}{2}$. In this basis, the S_z operator and it's

eigenvectors have the following form:

$$S_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \qquad S_z \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \equiv |\uparrow\rangle \qquad S_z \begin{pmatrix} 0 \\ 1 \end{pmatrix} = - \begin{pmatrix} 0 \\ 1 \end{pmatrix} \equiv - |\downarrow\rangle \tag{2}$$

In later sections, we will see how to make use of other basis of the qubit system, which we will call *register* from now on, but in this part we will focus on the previously mentioned *computational basis*.

The fundamental difference between classical bits and qubits is the number of states they can be in: Classical bits can only be in one of the two states $|0\rangle$ or $|1\rangle$, depending on the current flow. This lets us implement deterministic algorithms that work with binary numbers. In contrast to that, a qubit can't only be in its basis states regarding some operator, but in any superposition between them, where one can write any state $|\psi\rangle$ of a spin-1/2 particle representing a qubit in the following form:

$$|\psi\rangle = \alpha |\uparrow\rangle + \beta |\downarrow\rangle \qquad\qquad \alpha, \beta \in \mathbb{C} \tag{3}$$

where the normalization of the state demands

$$|\alpha|^2 + |\beta|^2 = 1 \tag{4}$$

As we can see easily, the squares of the amplitudes towards the different basis states describe the probability of the wavefunction $|\psi\rangle$ collapsing into one of the basis states and therefore measuring the corresponding eigenvalue.

This characteristic of qubits have a huge impact on the functionality of quantum computers: Since the outcome of the measurement of our quantum system is not deterministic, the workflow of a quantum computer differs fundamentally from a classical one. Due to the indeterministic behaviour, the algorithms performed on a quantum computer will also have a probabilistically distributed outcome (if the quantum algorithm doesn't stop in an eigenstate of the register), which implies that the types of task that can be performed on a quantum computer may look very different from classical algorithms.

1.1.3 Multi-qubit systems

To make use of the quantum nature of the system, we need to extend our system from one qubit to N qubits, and construct the resulting Hilbert space H as a tensor product of the single particle Hilbert spaces H_i:

$$H = H_1 \otimes H_2 \otimes ... \otimes H_N = \bigotimes_{i=1}^{N} H_i \tag{5}$$

We can now think of constructing a basis for the composite Hilbert space. It can be constructed by using the single qubit basis states of the corresponding $S_z \equiv S_z^j$ operator and building a tensor product:

$$|\uparrow\uparrow ... \downarrow\rangle = |\uparrow\rangle \otimes |\uparrow\rangle \otimes ... \otimes |\downarrow\rangle = \bigotimes_{i=1}^{N} |s_i\rangle \qquad s_i \in \{\uparrow, \downarrow\} \tag{6}$$

We can now see, that just like for any other tensor product space, the dimension of the given Hilbert space, and therefore the number of possible basis state of our system is given

as

$$d(H) = \prod_{i=1}^{N} d(H_i) = \prod_{i=1}^{N} 2 = 2^N \tag{7}$$

which results in 2^N state amplitudes for a given wave function $|\psi\rangle$. This number and therefore the space and computational time used grows exponentially in the number of qubits, which led to the development of numerical methods in fields like tensor networks, to make use of symmetries of the system to reduce that number of degrees of freedom and therefore the computational effort needed for manipulating these high dimensional objects.

The random access memory of the quantum computer, which consists of our tensor product Hilbert space, now has 2^N classical counterparts, but opposed to the classical analogue, it can be in superpositions of the 2^N basis states of the composite Hilbert space: Any wavefunction $|\psi\rangle$ can now be expressed in terms of these basis states and a tensor of rank N representing the amplitudes of the constructed basis:

$$|\psi\rangle = \sum_{i=1}^{N} \Psi_{s_1\, s_2\, \ldots s_N} \, |s_1\, s_2 \ldots s_N\rangle \qquad\qquad s_i \in \{\uparrow, \downarrow\} \tag{8}$$

From now on we will identify the different basis states with their classical counterparts, $|\uparrow\rangle \equiv |1\rangle$ and $|\downarrow\rangle \equiv |0\rangle$. With this interpretation in head, our system states can now represent binary numbers $|s_1\, s_2 \ldots s_N\rangle$, e.g. $|110\rangle$, which we can also write in the decimal basis $|110\rangle = |6\rangle$. Therefore, we can now encrypt binary numbers or other forms of data like strings in the eigenstates of our quantum system.

1.1.4 Entanglement

When it comes to describing systems with more than one particle (the qubits in our case), we have to take care of an intrinsic feature of multi-particle quantum systems: *Entanglement*, that will yield wide consequences regarding the data processing in quantum computers.

To understand this feature, we first have to have a look at a tensor product space of two single particle Hilbert spaces H_1 and H_2 with partial wavefunctions $|\psi_1\rangle$ and $|\psi_2\rangle$. Like this, the combined system can be constructed with a tensor product as follows

$$H = H_1 \otimes H_2 \qquad |\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \tag{9}$$

we can see, that we can write any state living in the tensor product space as

$$|\psi\rangle = \sum_{i,j=1}^{N} c_{ij} \, |i\rangle \otimes |j\rangle \tag{10}$$

If we now have a look at the so called *Bell state* [12], we see an interesting property.

$$|\psi_B\rangle = \frac{1}{\sqrt{2}}(|0\rangle \otimes |1\rangle + |1\rangle \otimes |0\rangle) \tag{11}$$

We see, that this state cannot be written in the form mentioned in equation 9, as there are no partial space vectors ψ_1 and ψ_2 with whom the state can be written as a single tensor product. This means, that the system is in a well defined state, without its partial systems being in well defined states. We call this type of state an *entangled* state.

A very easy interpretation of the previously mentioned bell state is seen if we switch to another basis using the addition of angular momenta. We know, that a system of two spin$-1/2$ particles can be interpreted as a spin-1 particle with the following change of basis using Clebsch-Gordan-Coefficients:

$$|s_1\, s_2\, m_1\, m_2\rangle \to |s_1\, s_2\, s\, m\rangle \tag{12}$$

This shows physically, that our system is in a well defined composite state, even though if we lost the information on our partial systems.

If we now consider our whole system being in the state $|\frac{1}{2}\,\frac{1}{2}\,1\,0\rangle$, and measure one of the partial magnetic quantum numbers m_{s_i}, the other magnetic quantum number will be fixed according to our prior measurement. This means, that measuring one partial state $|\psi_1\rangle$ will let the entanglement collapse and therefore reveal the state of the other partial system.

If we start with definite partial system in their eigenstates, they won't be entangled from the beginning. We can introduce entanglement between quantum objects through different types of interaction between the objects, and break it through the interaction with the partial system with the environments, like measurements.

One could ask, why entanglement is considered as such a crucial feature of quantum system when it comes to quantum computation. The reason is, that through entanglement our partial systems can influence each other without being spatially close to each other, which was described by Einstein as a "spooky long distance effect". Actually, entanglement can be seen as another type of computational resource and is thus crucial in the speed up compared to classical computers. Because of this feature, our qubits can be used to influence or control each other, which yields many applications for multi-qubit gates, like controlled gates, that are used widely in quantum computation.

For some states it's easy to see that they cannot be written as a direct tensor product of two vectors $v_i \in \mathcal{H}_1, w_i \in \mathcal{H}_2$, like the previously seen bell state, which already lets us know that the state is entangled. For other states, it's not trivial to recognize the entanglement. An important concept in determining the level of entanglement of a given state $|\psi\rangle$ is the reduced density matrix of the state, which will further be discussed in the appendix.

1.1.5 Quantum simulations

Since the technical implementation of quantum algorithm requires a noiseless quantum computer with a sufficient number of qubits, it makes sense to first rely on classical computation to simulate the quantum algorithms that we want to perform on real quantum computers later on.

We can simulate quantum algorithms with any classical computation environment, while some libraries in python are designed explicitally to provide the tools necessary for simulating them. One widely used example for such a library is *qiskit*, while the one we used for the simulations on the algorithms is *qutip*. Both libraries come with advantages and disadvantages regarding specific tasks, since `qiskit` focuses more on programming on quantum computers, while `qutip` provides a more adequate environment for general quantum objects and manipulations upon them.

Both libraries provide the essential tools for simulating quantum registers, like implemented data types for quantum objects and operators or numerical methods to do manipulations on them. Since there are several implementations of quantum computers, like gate-based or adiabatic quantum computers, the way in we have to simulate them, and notably the problem formulation may fundamentally differ. The simulation of gate-based

quantum computers for example requires the knowledge of the unitary operators representing the gates, while the ansatz for adiabatic quantum computers is a different one and relies more heavily on the mapping towards a problem Hamiltonian.

A question that might raise is the point of simulating quantum algorithms on classical computers: Since our quantum computers are exposed to real worlds influences by their environments and effects like quantum fluctuations that cannot be mapped directly to classical simulations, we always have to be careful on how much we can interpret into the results gathered by classical simulations. To overcome the discrepancy between an ideal environment like its assumed in quantum simulations on classical computers, and real world conditions when implemented technically, some companies like IBM have provided remote open source access to quantum computers [13], that can be used to simulate given quantum algorithms on real quantum hardware and hence investigate the real world effects on the system.

Another crucial point in simulating quantum systems on classical computers is the aforementioned memory needed to store the data of the quantum system: Since an n qubit quantum register's Hilbert space has dimension $d = 2^n$, the memory needed to store and manipulate the amplitudes of the states is exponential in the number of qubits, which makes it almost impossible to simulate large quantum registers on classical computers, if no approximative techniques like matrix product states or general tensor networks are applied to reduce the degrees of freedom of the system.

1.2 Optimization problems

Since the goal of this work is to solve optimization problems using the quantum search by measurement algorithm, we will now give some introduction regarding these kinds of problems. Optimization problems demand to find the *optimal* solution to a given problem, where the optimality is defined in terms of a cost function $f(x)$. Thus, our goal is to find a solution $x = y$ to the problem, for which $f(x = y)$ becomes minimal or maximal, depending on the problem definition.

In theoretical computer science, there's a distinction between so called actual optimization problems, for which we're only interested in the value of $f(x)$ that's extreme, and so called search problems, for which we're interested not only in the value of $f(x)$, but also for the solution x that optimizes the cost function. In real life applications, we're usually interested in search problems, since an optimal solution without the knowledge of the underlying configuration is not helpful in general.

Optimization problems, and especially search problems appear almost in all areas of scientific, and everyday life: A very easy real life example for such a problem could be shopping at a supermarket: We're looking for a specific set of ingredients for a given recipe, while for each ingredients there are different brand and therefore different associated costs. The optimization problem in this case would be finding the needed products while paying the least amount of money. Thus, the cost function $f(x)$ in this case would be the total price for the purchase.

While the previously discussed example is a rather simple one, there's actually no limit to the complexity of optimization problems: The function we're trying to optimize could be the ground state energy of a given molecular system in quantum chemistry, the expectation value of a given stock portfolio in finance, or the solution to a search algorithm in the case of Grover's algorithm.

One of the most fundamental algorithmic problems is actually connected to optimization

problems: It's so far unknown, if every optimization problem can be reduced to polynomial running time in the size of the input, which is known as the *P-NP problem*. Since quantum computers are already known to solve some problems significantly faster than classical computers, mapping these non-polynomially growing p-problems to quantum computers may shed light onto the equivalence of these problem classes.

The problem class that we're going to further investigate on in this work, are so called combinatorial optimization problems, where we try to find a solution, that minimizes a given cost function $C(k)$, from a discrete set of possible configurations. These configurations could for example be a set of n decisions, denoted by 0 or 1, whether if they're taken or not. The space of possible solution is then spanned by the possible number of combinations, in this case 2^n, and the optimization aims to find the combination k' for which $C(k')$ is minimal.

There are several different kinds of combinatorial optimization problems, which all can be solved using similar techniques, including the quantum algorithm that's going to be discussed in this work. One example would be the travelling salesman problem (TSP), where we're given a graph, with nodes and vertices, and want to minimize the total cost associated with the vertices, for visiting all nodes. This can be mapped to many real world problems, especially in fields like logistics, but also others [14]. Another widely known example for a combinatorial optimization problem is the knapsack problem [15]: We're given a domain of items, that all have an associated value, and weight. Also, we're given a bag, which has a maximum possible weight. The optimization problem then lies in finding a set of items, that has the maximum value while not exceeding the weight limit of the bag. This problem again, can be mapped to a wide variety of problems, like the selection of stocks into a portfolio, or general problems were there's a limited capacity that cannot satisfy the demand.

Now that we talked about possible combinatorial optimization problems, we will now have a look at how these kinds of problems can be interpreted in terms of quantum mechanics.

1.3 Quantum Optimization

As we discussed previously, the ascending complexity of real world problems demands the computational effort to exceed the current state provided by classical computers. This becomes even more interesting, when it comes to simulating quantum systems, and finding optimal solutions to a given energy problem defined by a Hamiltonian H:

If we assume a discrete energy spectrum of the Hamiltonian H, where each energy eigenstate corresponds to a configuration $|\psi\rangle$ in the Hilbert space, we can translate the combinatorial optimization problem from last section onto our quantum system: Finding the ground state configuration (the configuration that minimizes the energy) from a discrete space of possible combinations, can be seen as a combinatorial optimization problem (a search problem in fact, which is NP-hard). With this in mind, we can now think of solving real world combinatorial problems, by mapping them to our quantum system, and using techniques from quantum theory and statistical physics to solve the problem. This scheme already has been applied to solve financial optimization problems [16], and many other real world problems can be solved by mapping the problem to a quantum problem. The idea how the mapping between the two problems is performed, and how these can be solved explicitally, will be further discussed in section 1.6.

Even if it's intuitive that quantum problems can be solved more easily using other quantum systems to simulate them, than relying on classical computation techniques, it's still

not clear why exactly quantum computers are superior in solving certain tasks: Solving the ground state problem of a given quantum system needs both, an exponentially large memory, as well as exponentially growing running times, when trying to do so on classical computers, due to the vast number of parameters encrypted in the states' amplitudes that need to be processed. Therefore, simulating macroscopic problems using the laws of quantum mechanics becomes intractable for classical computers [17]. The idea behind quantum optimization is to find a certain ground state of a system, using the laws of quantum mechanics themselves, which is called quantum computation. It becomes very interesting in fields like quantum chemistry [18], quantum biology [19], and other fields in which the macroscopic behaviour of a system is affected by the microscopic behaviour of the quantum objects.

The previously discussed aspects point out, that optimizing problems using quantum systems can yield huge advantages for both, real world optimization problems that are mapped to quantum systems, as well as optimization of quantum systems themselves. In the next section, we're going to discuss a general concept of quantum computation that can be used for finding the ground state of a given Hamiltonian, so called *Adiabatic quantum computation*. We can therefore make use of this scheme to solve quantum optimization problems, and therefore also for encrypting real world optimization problems.

1.4 Adiabatic Quantum Computation

The approach that's most similar to classical computing is so called *gate-based* quantum computing, where quantum logic gates, that are unitary operators acting on the qubit register replace the classical logic gates. This type of quantum computers will be further explained in the appendix A.2, but now we're going to introduce another approach to quantum computing, which is providing a more natural environment for our problem statement.

Adiabatic Quantum Computers (AQC) are implementations of quantum computers, that are specifically good in solving problems formulated as finding the ground state of a given system [20], and therefore work especially well on quantum optimization problems. Some difficulties that are connected to gate-based quantum computers can be overcome using adiabatic quantum computation or its further evolved counterpart, quantum annealing, which both will be discussed in the following sections. Problems defined on adiabatic quantum computers consist of finding the ground state of a given Hamiltonian H_P, where we can encrypt several problem classes into H_P, and therefore the ground state that's representing the solution to the optimization problem. The corresponding class of Hamiltonians that's used widely for encrypting optimization problems, are so called Ising type models, which are going to be discussed in section 1.6.

The first step of AQC is defining the problem Hamiltonian H_P, whose ground state configuration we want to find. We then construct another Hamiltonian H_i, and initialize the system in its ground state $|\psi_0(0)\rangle$. The choice of H_i is important here, since we have to know its ground state beforehand. Furthermore, H_i will serve as a so called *driving* Hamiltonian: The idea is to start in H_i's ground state, and slowly introduce H_P as the system Hamiltonian, to let the state adapt to the new Hamiltonian and its ground state. The transition between both Hamiltonians is denoted in terms of a dimensionless parameter $s \in [0, 1]$ describing the intermediate states between H_i and H_P:

$$H_P \equiv H(s = 1) \qquad H_i \equiv H(s = 0) \tag{13}$$

To understand how the ground state of that Hamiltonian is reached on adiabatic quantum

computers, we will make use of the adiabatic theorem. It was firstly introduced and proven by Born and Fock [21]. It states, that if there's a finite minimal gap $g_{\min}$ between the ground state and the first excited state of all Hamiltonians $H(s)$, and if the perturbation is introduced slowly enough, the system will remain in the corresponding eigenstate of the perturbated Hamiltonian $H(s)$, which will be its ground state, if we start in H_i's ground state. This allows us to investigate on a complex Hamiltonian's ground state by introducing a superposition between this Hamiltonian and an easier one. One possibility to perform the interpolation between the Hamiltonians is a linear schedule function, equivalently to the description in the original paper by Childs et al.:

$$H \equiv H(s) = sH_i + (1-s)H_f \tag{14}$$

With this interpolation, we will start in our initial Hamiltonian $H(0)$ for which we know the ground state, and end in the final Hamiltonian $H(1)$, which lets us investigate on its ground state. It's important to mention here, that we don't have to choose this linear path between H_i and H_P necessarily, since the only constraints we're given for $H(s)$ are $H(0) = H_i$, and $H(1) = H_P$. Therefore, we can use any function that satisfies these conditions. In this work we will focus on polynomial schedule functions:

$$H_k \equiv H_k(s) = s^k H_i + (1-s)^k H_P \tag{15}$$

In terms of the accuracy of finding H_P's ground state, the used interpolation between both Hamiltonians is crucial, since it determines the set of Hamiltonians $H(s)$ we're transitioning through and therefore the spectral gaps $|E_0(s) - E_1(s)|$. We will further investigate on that dependency when focusing on the quantum search by measurement algorithm to find a given Hamiltonian's ground state. The Hamiltonian can also be redefined in terms of a more natural parameter, is the evolution time t with a maximum evolution time T, so that $0 < s = t/T < 1$:

$$H \equiv H(t) = \left(\frac{t}{T}\right)^k H_i + \left(1 - \frac{t}{T}\right)^k H_P \tag{16}$$

The reason why the system will adapt to each Hamiltonian's ground states with high likelyhood, is the quantum tunnel effect: Even if there's a finite gap between the ground state energies of two consecutive Hamiltonians, there's a probability of transiting from one ground state to the other depending on the spectral gap. In terms of the newly introduced variables t and T this means, that we can reach the final Hamiltonian's ground state with high probability by either choosing the evolution time between two Hamiltonians t very small, or by increasing the total evolution time T. This raises the question, how slowly we have to introduce said perturbation Hamiltonian, or in other words, how fast we are allowed to propagate t from $t = 0$ to $t = T$.

For this, we have to think about how the gap size between the two first states throughout the transition between the two Hamiltonians H_i and H_P is influencing the system's behaviour: If the evolution time between two Hamiltonians $H(t = t_0)$ and $H(t = t_0+)$ is too small for a given gap size, the state's energy will change too fast, since there's not enough time to adapt to the new external parameters, which can lead to a transition $|\psi_0\rangle \to |\psi_1\rangle$ into excited states of the new Hamiltonian, which we want to prevent. These transitions are called *Landau-Zener-Transitions*, whose probability scales with the inverse of the total evolution time [22].

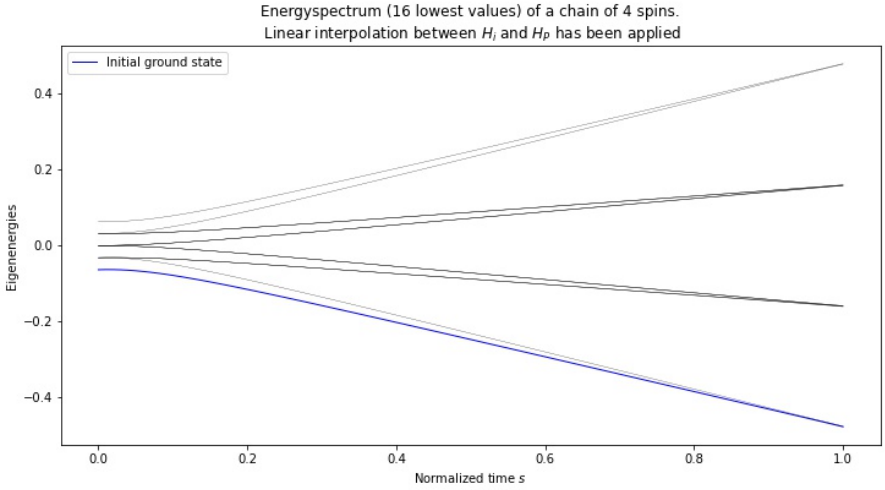

Figure 1: The development of the eigenenergies of an Ising model of four particles. There are no energy level crossings, which is why adiabatic quantum computation can be applied.

It has been found, that the total evolution time has to be in the order of

$$T = \mathcal{O}(\frac{1}{g_{\min}}) \tag{17}$$

for the algorithm to be slow enough to prevent transitions to a Hamiltonian's first excited state. This directly yields the question, if this lower bound for the evolution time is compatible with the idea of quantum computers exceeding their classical counterparts' performance. In terms of the running time, we also have to point out the importance of the lone *existence* of the gap between the ground state and the first excited state of $H(s)$: If there's no finite gap, the time-dependent energy levels will cross for a certain parameter t', effecting a transition of the system to the excited state with certainty, leading the algorithm to fail. Using an adequate schedule for the Hamiltonian $H(s)$ can prevent these level crossings and therefore enhance the accuracy of the algorithm [23].

The presented scheme can be applied to computational problems [24]: We know, that we can force the system to stay in the ground state of a varying Hamiltonian $H(t)$ by introducing the perturbation slow enough. Therefore, we are able to access information about the final Hamiltonian's ground state $|\psi_0(1)\rangle$, if we only know the ground state of the initial Hamiltonian H_i. This can be very useful, since many times we're interested in an eigenvalue problem of the following form:

$$H\,|\psi_0\rangle = E_0\,|\psi_0\rangle \tag{18}$$

which can be mapped to an optimization problem 1.6. The problem is then finding an eigenstate $|\psi\rangle$ that minimizes the energy E, which can be interpreted as the input to a given cost function we want to minimize. This lets us solve optimization problems according to the laws of physics: We first prepare the system in the initial Hamiltonian H_i's eigenstate, then let the system evolve in time according to the adiabatic theorem until we reach the

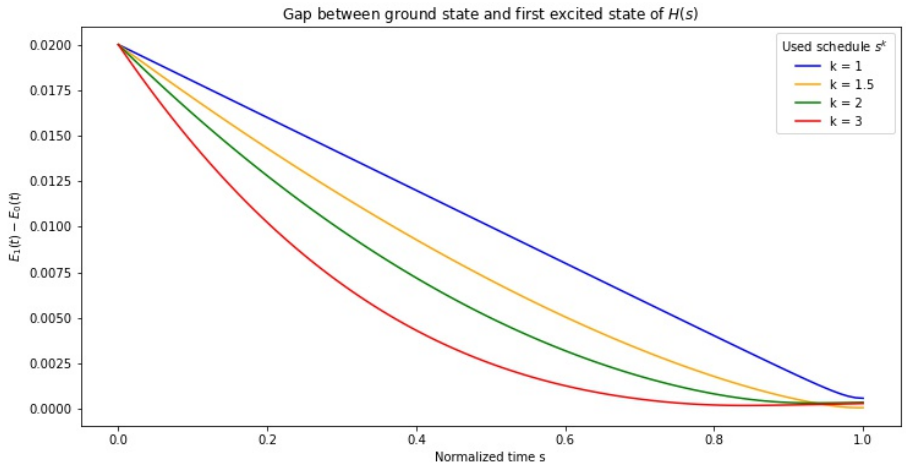

Figure 2: The development of the minimal gap $|E_1(s) - E_0(s)|$ in dependence on the chosen schedule

final Hamiltonian H_P, and finally evaluate the state the system is in to find the solution to the optimization problem.

As it was pointed out, the minimal spectral gap $g_{\min}$ determines how slow we have to perform the adiabatic evolution, which means that we have to investigate on that gap beforehand. This can be quite expensive in terms of computational resources, which is why a technique has been developed to bypass the computation of $g_{\min}$, which will be discussed in the next section.

1.4.1 Quantum Annealing

As discussed in the previous section, the minimum evolution time T in the adiabatic time evolution between the initial and final Hamiltonian depends on the minimal spectral gap $g_{\min}$. Therefore, to determine how long we have to run our algorithm, it's crucial to know the energy gaps $|E_a(s) - E_b(s)|$ between all eigenenergies of the transition Hamiltonians between $H_i = H(0)$ and $H_P = H(1)$. To overcome the computational expense that's connected with diagonalizing all the Hamiltonians for estimating the minimal evolution time, the principle of *Quantum Annealing*, that was introduced by Kadowaki et al. [25] can be used:

The basic idea is, to not first investigate on the minimum spectral gap before performing adiabatic quantum evolution to reach a system's ground state. This is done, because it can be quite challenging to investigate on that gap, since in general we need find the energy eigenvalues of the Hamiltonian, for example by exact diagonalization, which demands exponentially growing computational resources. Instead, the algorithm is run for arbitrary evolution times τ corresponding to arbitrary minimal spectral gaps. If we now perform the algorithm a sufficient number of times, we can investigate on the ground state of the final Hamiltonian in dependency of the evolution time τ chosen. We then pick the evolution time τ, that yields the final state with the lowest energy, which is very likely to be the ground state of the problem Hamiltonian.

This implementation works especially well on optimization or sampling problems with a discrete search space. It further has been previously shown, that using quantum annealing we reach the same running time for quantum algorithms as with gate-based quantum computation. As we can see, this principle is very much connected to adiabatic quantum computation, but can yield improvements regarding the running time, if its hard to estimate the minimum spectral gap, which is crucial for adiabatic quantum computation.

Since problems in quantum annealing are formulated in terms of eigenvalue problems instead of unitary transformations, real quantum computers that exploit this technique can perform computations on a number of qubits that's way higher than the gate-based versions, due to the insensitivity of quantum annealers towards environmental noise [26]. On the other hand, quantum annealers so far don't represent an actual alternative to gate-based quantum computers, since some algorithms, like Shor's algorithm, cannot be executed on them easily [27].

Now that we discussed how its possible to find a given Hamiltonian's ground state using adiabatic quantum computation, we will now discuss a class of Hamiltonians that can be used for encrypting combinatorial optimization problems into quantum systems, so called Ising models.

1.5 Classical Ising Model

A very important model to encrypt optimization problems into and solve them with classical optimization techniques is the so called *Ising model*, where we assume a d-dimensional lattice with N sites, at which classical spin variables $s_i \in \{-1, 1\}$ are positioned. There are several variations of the Ising model, where in the model we're focusing on, the spins are only interacting with their nearest neighbours, where the coupling strength is denoted in the vector $\vec{J} = (J_1, J_2, ...J_{N-1})$. This can also be extended to all to all coupling. Also, we introduce a local field acting on the spins, where its strength is denoted by the matrix h. The (classical) Hamiltonian of the Ising model is then given as

$$H = \sum_{i=0}^{N-1} J_i s_i s_{i+1} + \sum_{i=0}^{N} h_i s_i \tag{19}$$

The Ising model was introduced to understand the behaviour of many-body spin systems interacting through their internal magnetic field, while reducing the possible degrees of freedom and interactions between the particles to a minimum.

The reason why the Ising model yields many possibilities in this case, is that it's possible to map several optimization problems to the classical Ising model, by encrypting the desired solution into the Hamiltonian's ground state, and using techniques from statistical physics to solve the problem. We map a real optimization problem to the Hamiltonian H, and encrypt the solution to the problem into its ground state energy and the corresponding configuration. By finding the ground state of the Ising model, we can therefore gather information about the configuration that minimizes the energy and therefore the solution to the optimization problem.

There are two crucial parts in solving optimization problems with the Ising model: The first is to find a mapping between the optimization problem and a given Ising Hamiltonian, and the second is to actually be able to find the ground state configuration in reasonable time. This lead to the idea, that we can use the quantum version of the Ising model, to make use of the advantages of quantum computation to reach the (classical) Ising model's ground state, in which the solution to the optimization problem is encrypted.

1.6 Transverse Field Ising Model

Now that we discussed the basics of the classical Ising model, we will dive into the quantum formulation of the Ising model, and how it can be used to solve quantum optimization problems with adiabatic quantum computation or related techniques.

In the classical Ising model we assumed the spin variables being fixed to either -1 or 1, while in the quantum Ising model we replace the fixed spins with two dimensional quantum objects, in this case again spin-1/2 particles, which each can be characterized through their amplitudes regarding a fixed direction in the corresponding computational basis. This is done by replacing the fixed spins s_i from the classical model with the pauli matrices σ_i^z acting on the i-th spin site. The Hamiltonian is then defined on the Hilbert space $\mathcal{C}^{2^n}$ if n particles (or qubits) are involved in the chain.

The 1D transverse field Ising Model is defined through two contributions to the Hamiltonian, leading to our Hamiltonian being decomposible into a sum $H = H_P + H_i$:

The first part, H_P, is the problem Hamiltonian whose ground state we want to find, and into which the problem is encrypted. It denotes the interaction terms between the spins, where the crucial parameter, in the case of nearest neighbour interaction, is the matrix J, which determines the strength of interaction between two neighbouring spins. Also, there's a local field acting on each site whose strength is denoted by the vector h:

$$H_P = \sum_i J_i \sigma_i^z \sigma_{i+1}^z + h_i \sigma_i^z \tag{20}$$

If we let alone this Hamiltonian H_P, we see that it's analogue to the classical Ising formulation from the last section, due to the σ_i^z operators at different sites commuting with each other. This means, that finding H_P's ground state, either in the classical, or the quantum Ising model, will provide us with the desired solution to our problem.

In the case of all interactions being the same magnitude, we can set J as a scalar. It's important to note, that the sign of J will determine if our spins rather follow the same, or the opposite direction of its neighbours, or in other words, if we have ferromagnetic or antiferromagnetic coupling.

The second contribution to the Hamiltonian, which we use as the initial Hamiltonian H_i in terms of AQC, is characterized through an external transverse acting on the spins on the correposponding sites, whose strength is denoted by the vector h':

$$H_i = \sum_i h_i' \sigma_i^x \tag{21}$$

It's the combination of both terms H_P and H_i leading to the quantum behaviour of the system, since its components don't commute. This comes from the basic commutator relation of the pauli matrices at the given sites, leading to the Hamiltonian not being diagonal in the σ^z-basis anymore, inducing different behaviour in comparison to the classical analogue:

$$[\sigma_i^z, \sigma_i^x] \neq 0 \tag{22}$$

We have to note here, that if a qubit register representing the Ising model is measured in the computational basis, we cannot estimate the ground state of the given Hamiltonian with certainty, if it has contributions of σ_x terms: A measurement in the computational basis lets the system collapse into an eigenstate of the tensorized σ_z operator representing a

measurement in said basis. Since this "measurement operator's" spectrum is not the same as the Hamiltonian's, a projection into one of the measured eigenstates doesn't effect a projection onto an energy eigenstate.

Combining both Hamiltonians H_i and H_P leads us to the composite Hamiltonian, which inhabits both, a transverse field in x-direction, as well as the nearest neighbour interaction and the local field term:

$$H = \sum_i J_i \sigma_i^z \sigma_{i+1}^z + h_i \sigma_i^z + \sum_i h_i' \sigma_i^x \tag{23}$$

Investigating on the Ising modelHamiltonian and the properties of the lattice will let us have insight in phenomena like critical points or phase transitions of the composite system. [28].

It's important to note that the Ising model and its analytical solvability heavily depends on the dimensionality of the model [29]. Since the 1D Ising model, opposed to higher dimensional models, can be solved explicitally, it's used widely in quantum information. In higher dimensions, approximative numerical methods like the Metropolis algorithm are used to investigate on the system. Due to the complexity of the transverse field Ising model, it's generally intractable for classical computers [30].

Also, we should mention the roles of H_i and H_P here, if we again remember the problem definition given in the section about adiabatic quantum computation 1.4: We need the ground state of the initial Hamiltonian H_i to be both easily findable, as well as easily preparable technically. Setting the H_i term as the transverse field term of the Ising model satisfies these conditions, since we now that the ground state of said Hamiltonian is given by one of the two eigenstates of the composite σ_x operator. Furthermore, the part that's crucial for the optimization is the interaction term 23, since it represents the classical model, to which optimization problems can be mapped. This means, that we can map the solution to the problem to the ground state of H_P, and then start propagating the Hamiltonian $H_i \to H_p$ according to the adiabatic theorem, starting in H_i's ground state, and ending in H_P's ground state, finding the optimal solution to the desired problem.

There are several ways to alter the Ising model and the associated problem definition, for example by implementing a closed chain by also considering the interaction of the first spin of the lattice with the last one, allowing one to encrypt more optimization problems. Another possibility, to construct a more general Ising model is including the interaction between all spins on the lattice. With this its possible to encrypt more different kinds of problems into the problem Hamiltonian H_P. It's still important to note here, that the model here only inhabits the interaction between two sites at a time, and the transverse field only acting on one site, and thus is not believed to be an universal model for encrypting arbitrary optimization problems, as Biamonte et al. stated [31]. Further expansions of the model are possible, if we don't only assume two particle interactions in the underlying Hamiltonian, but also interactions of three or more sites at a time. The interaction Hamiltonian then becomes

$$H_P = \sum_{i,j,k} J_{ijk} \sigma_i^z \sigma_j^z \sigma_k^z + \sum_{i,j,k,l} J_{ijkl} \sigma_i^z \sigma_j^z \sigma_k^z \sigma_l^z + ... \tag{24}$$

where the J_{ijk} denote the components of a third order tensor, J_{ijkl} the fourth order tensor inhabiting the four particle interactions, and so on. In this work, we will restrict on optimization problems, that are possible to represent by Hamiltonians with maximally neigherest neighbour interactions.

As we discussed previously, the Ising model is widely used in quantum computation, which raises the question on how to use it to encrypt computational problems: Solving the Ising model can be interpreted as a combinatorial optimization problem, since we're trying to optimize the energy of the system by changing the spin configuration, which is a discrete search space. We can define the n spins as the variables with whom we try to minimize the energy E of a given lattice model. Encrypting the problem into the Ising model by varying the interaction J and external parameter h will yield the solution to the optimization problem encrypted in the configuration of the ground state of the problem Hamiltonian. In the end, the configuration $|\psi\rangle$ that minimizes the energy of the system, also minimizes the associated cost function.

When the interaction terms J_{ij} and h'_i in 23 are chosen randomly, we call the model an Ising spin glass [32], which can be interpreted as random optimization problems. Due to this fact, Ising spin glasses are often simulated to benchmark algorithms that aim to solve these problems [33], which is why in this work we'll also focus on this model to investigate on the accuracy of the quantum search by measurement algorithm.

Now that we know how to find a ground state of a given Hamiltonian using AQC, and know the class of Hamiltonians that's useful for solving optimization problems, we'll now give some intuition on how to find adequate Ising models for a given optimization problem.

1.6.1 Mapping optimization problems to the Ising model

In the last section, we discussed that Ising models can be used for encrypting optimization problems, which is why now we're focusing on how a real world optimization problem can be translated to the spin space of the Ising model. Various examples of these mappings from the Ising model to NP-hard problems have been introduced by Lucas et al [30]. Actually finding a projection of the problem onto the Hilbert space of the Ising model can be quite hard, but has been performed for various kinds of optimization problems, like the knapsack problem, the TSP, or the slot placement problem [34].

A relatively simple approach for encoding problems into the Ising model, is interpreting the spins in a given configuration as Boolean values, $s_i \equiv b_i \in \{0, 1\}$. With this in mind, each spin corresponds to a decision, which is either taken or not, and each configuration corresponds to a set of decision. In the example of the knapsack problem from section **??**, this could be the decision whether an item is taken into the sack, or not. We then try to find a set of decisions $\{b_i\}_{i=1}^{N}$, which optimizes the cost function, or in other words, a spin configuration that minimizes the energy. This approach can be extended to numerous problems whose outcome depend on a set of decisions.

As discussed previously, in general it's impossible to solve the transverse field Ising model analytically, which would be necessary to confirm the results of quantum simulations regarding the model. For this, it has been shown that we can translate the problem defined in terms of a spin chain to another problem, defined in terms of fermions, that's easier solvable, as it will be discussed in the next section. This will yield the possibility to confirm properties of the Ising model that have been found using quantum computation.

1.6.2 Jordan-Wigner-Transformation

To confirm the properties of a given Transverse field Ising model, like its energy eigenvalues and corresponding eigenstates, it's important to have a reference regarding the outcome of quantum simulations. A very important tool for gathering solutions to the model is the so called Jordan-Wigner-Transformation [35]. Since we're only interested in the idea and the consequences of the theorem, the proof will not be further discussed here.

As we discussed in the previous section, it's possible to encrypt optimization problems into the transverse field Ising model, while the problem defined in terms of a spin chain is not analytically solvable in general. To still be able to investigate on the system's properties classically, we can map the spin chain onto a lattice of fermions: The Jordan-Wigner-theorem states, that an n-particle spin chain is isomorphic to an n-mode fermionic system. This means, that we can map each spin on the chain to a position on the fermionic lattice, that's either occupied, or not, depending on the spin orientation of the j-th spin.

For this, we assume that our n-spin lattice get's translated onto an n-site lattice, which are all occupied according to the order of the spins in the initial system. We then translate the action of the $\sigma^{\pm}$ matrices onto fermionic creation and annihilation operators $a_j, a_j^{\dagger}$, which can be interpreted similarly as the ladder operators for a quantum harmonic oscillator, creating or removing the fermion on the j-th lattice site, or flipping the spin in the terms of the spin chain. This becomes very important in the context of quantum information, since it allows us to investigate on the transverse field Ising model and therefore on our qubit register, by mapping it to a fermionic system and applying the known methods from statistical physics to solve it.

On the other hand, the transformation can be used to investigate on physical problems consisting of interacting fermions, by mapping them to a quantum register. This is possible, since we can represent an n qubit quantum register through a chain of n spins. Afterwards, we apply quantum algorithms on the configuration to determine its behaviour under the simulated effects, and map it back to the initial fermionic system. It has been shown by Ovrum et al., that actually any fermionic quantum system can be mapped to a quantum register using the Jordan-Wigner-Transformation [36], which implies the versatility of quantum computers regarding the simulation of real quantum systems.

1.7 Fidelity of quantum states

Another important prerequisity for verifying the found solution to some given problem encoded in terms of quantum states, is the so called *Fidelity* [37], which gives us a way of quantifying the *similarity* of two quantum states, allowing us to investigate on how close our solution is to the *real* optimal solution, expressed in terms of the corresponding Hamiltonian's ground state.

We know from quantum mechanics, defined in terms of quantum states, that we can quantify the overlap of two quantum states with the square of the natural scalar product on the underlying Hilbert space. In the Dirac notation, for two states $|\psi\rangle$ and $|\phi\rangle$ this is given as

$$|\langle\psi|\phi\rangle|^2 = |\langle\phi|\psi\rangle|^2 \tag{25}$$

If we now want to extend the concept to mixed rather than pure states, we can define the fidelity of two quantum states, described by their density matrices ρ and σ as follows:

$$F(\rho,\sigma) = \mathrm{tr}(\sqrt{\sqrt{\rho}\sigma\sqrt{\rho}})^2 \qquad 0 < F(\rho,\sigma) < 1 \tag{26}$$

For a fidelity of 1, we know that the density matrices are the same, while the case $F = 0$ corresponds to orthogonal states.

It's easy to see, that the fidelity is symmetric, which has to be the case, since equity between two states is a symmetric relation. Furthermore, equation 26 becomes the square of the overlap in the case of pure states:

$$F(\rho, \sigma) = |\langle\psi_\rho|\psi_\sigma\rangle|^2 \tag{27}$$

The reason why we're introducing this concept, is to verify solutions that are encrypted into states in the form of density matrices: If we estimate a density matrix representing a optimal solution with a given algorithm, for example adiabatic quantum computation from section 1.4, we can now easily verify if the reached ground state is actually correspondent to the *real* ground state of the system, that has been found using numerical approximation methods, or using the Jordan-Wigner-Transformation and methods from statistical physics. Hence, in terms of optimization problems, this gives us a measurement for the optimality of the solution.

2 Quantum search by measurement

The paper *Quantum search by measurement* (QSM) by Childs et al. proposes a new algorithm with which the ground state of a complex problem Hamiltonian can be found, using a scheme for measuring introduced by von Neumann. Originally, the scheme was used to perform the search algorithm introduced by Grover et al, while in this work, we will apply the scheme to a transverse field Ising model Hamiltonian, with which optimization problems can be solved by encrypting the solution in the Hamiltonian's ground state. In this section we're going to explain the basic concepts that are necessary to understand the algorithm, to be able to investigate on its accuracy in dependency on the parameters introduced by Childs et al. in the following section.

The algorithm works similarly to the adiabatic quantum computation approach, but it differs in how the system is kept in the ground state through the evolution $H(0) \rightarrow H(1)$, because here we're keeping the system in it's ground state by performing consecutive collapsive measurements according to the *quantum zeno effect*.

2.1 Quantum Zeno Effect

A very important prerequisite for understanding the quantum measurement scheme, is the so called quantum zeno effect, which is used to force the system to stay in its ground state for all Hamiltonians $H(s)$. According to the postulates of quantum mechanics, a wave function that consists of a superposition of eigenstates of a given Observable O collapses into one of O's eigenstates $|v_i\rangle$ when being measured. The outcome of the measurement depends on the probability distribution p_i of the eigenstates in the total wavefunction of the system, which is described in the discrete case with N possible orthogonal eigenvectors $|v_i\rangle$ and their eigenvalues λ_i as followed:

$$|\psi\rangle = \sum_{i=1}^{N} c_i |v_i\rangle \qquad\qquad p_i = p(\lambda_i) = ||c_i||^2 \tag{28}$$

As we see in the equation above, a higher amplitude towards one eigenstate yields a higher probability of measuring the corresponding eigenvalue. This leads directly towards the idea

of the quantum zeno effect: If a system is initially prepared into an eigenstate, it's probability distribution will dissipate from a $\delta(x)$ peak onto a flattened distribution, when evolving in time under a Hamiltonian that's not an eigenfunction of the state. This in mind, if we would measure our state in the limit $t \Rightarrow 0$, the dissipation of the distribution has not yet begun and we would measure our initially prepared eigenstate with a 100% probability. If we now extend this to finite times and assume that the evolution time until the measurement τ is small on the scale of the energy gap, we will still measure our initially prepared state with a high probability, that will scale with the inverse of the evolution time τ. This means, if we choose τ sufficiently small, we can keep our system in the initially prepared state and prevent transitions while evolving in time and performing collapsive measurements of the system after accurately chosen timesteps. If we now think of our Hamiltonian being time-dependent, we're therefore able to force our system to stay in the ground state of $H(s)$ until we reached H_P's ground state, yielding the solution to the optimization problem. The quantum zeno effect also has been proven experimentally in the case of $^9Be^+$ hyperfine transitions by Itano et al [38].

It is important to note that, when it comes to the measurement process, we have to neglect the idea of measuring a certain physical value, but rather interpret the measurements as interruptions of the time evolution induced by the unitary operator U_t, by applying a projection operator corresponding to the measurement. To show, that the probability for staying in the ground state after making use of the quantum zeno effect is actually higher than just evolving the system in time without the measurements, we can have a look at the probability for surviving in the ground state $|\psi\rangle$ of a given Hamiltonian H:

$$p(t) = |\langle\psi| e^{-iHt} |\psi\rangle|^2 \tag{29}$$

Since the times we want to evolve our system for are very small, we can expand the exponential function into a power series:

$$e^{-iHt} = 1 - iHt + \frac{1}{2}H^2t^2 + \mathcal{O}(t^3) \tag{30}$$

We have to take note that, in this case, neglecting the terms of $\mathcal{O}(t^3)$ only makes sense if the time between the measurements is sufficiently small. If we now have a look at the survival probability of the ground state in the power series approximation we find:

$$p(t) = |\langle\psi| e^{-iHt} |\psi\rangle|^2 \approx 1 - (\Delta H)^2t^2; \qquad (\Delta H)^2 = \langle\psi| H^2 |\psi\rangle - \langle\psi| H |\psi\rangle^2 \tag{31}$$

If we now divide our total evolution time into M parts, which represent the number of measurements and multiply the probabilities for staying in the ground states p_n after each of the M measurements, we find

$$p(t) = \prod_{n=1}^{M} p_n(t) = \prod_{n=1}^{M} [1 - (\Delta H)^2(t/M)^2] = [1 - (\Delta H)^2(t/M)^2]^M \tag{32}$$

It can be shown easily that for $M \to \infty$ the probability for staying in the ground state for any finite time t approaches 1. This shows mathematically, that we can force our system to stay in its initially chosen state by applying a sufficient number of measurement interactions throughout the time evolution. It actually has been shown by Paz-Silva et al, that using the technique of projective measurements according to the quantum zeno effect can prevent arbitrarily prepared states from decoherence with arbitrary accuracy [39].

A very important aspect about the scheme is how the measurement is performed technically: Since direct measurements of H will pertubate the system, we will couple the target system to a so called *pointer* system which can be measured, and yields information about the target system, leading it to collapse in one of its eigenstates. This is the measurement scheme that was introduced by John von Neumann.

2.2 Von Neumann's measurement prescription

To perform the quantum search by measurement algorithm to find a given Hamiltonian's ground state, we now have to introduce an important subroutine, the measurement prescription by von Neumann. Our goal with this subroutine is estimating the energy of a given target system, by performing measurements on a second system, the pointer system, to which the target system is coupled, without pertubating the target system itself. Therefore, we start with a Hamiltonian H and a given state $|\psi\rangle$ whose energy we want to estimate. Furthermore, we will need a second system acting as the pointer system.

The pointer can be represented through a free particle in 1D and its position. We know from quantum mechanics, that evolution of a free particle under its momentum operator p will result in a translation in space, due to the relation of the corresponding basis via a Fourier transform:

$$e^{-ip\delta x}\,|x\rangle = |x+\delta x\rangle \tag{33}$$

Now we assume, that we evolve our composite system under the tensor product of the Hamiltonian H, and the momentum operator p on the pointer system, which will lead to the following evolution of the composite system, expressed in its spectral presentation regarding the eigenstates of the problem Hamiltonian H:

$$e^{-iH\otimes p\tau} = \sum_{i=1}^{2^n} |\psi_i\rangle\langle\psi_i| \otimes e^{-iE_i p\tau} \tag{34}$$

As we can see, if we apply this transformation to a tensor product of a given eigenstate $|\psi_i\rangle$ of the underlying Hamiltonian, and a position operator eigenstate initialized in $|x=0\rangle$, we will get

$$e^{-iH\otimes p\tau}\,|\psi_i\rangle \otimes |x=0\rangle = |\psi_i\rangle \otimes |x=\tau E_i\rangle \tag{35}$$

The evolution under this coupled Hamiltonian won't change our target state $|\psi_0\rangle$, because of the counting register being in the momentum basis, leading to a so called phase kickback (2.4.1), but will rather result in a translation of the pointer which is proportional to the Energy E_i of the target system and the pointer-system interaction time τ. This means, that by measuring the pointer variable, we will get information about the energy of our target system, which effectively is a measurement and will lead to the target system's wavefunction to collapse. Combining the quantum zeno effect with the measurement scheme by von Neumann now allows us to prevent a system initially prepared in an eigenstate from decoherence, by performing measurements on a second qubit register.

We have to note, that in order to measure the energy accurately, we have to be able to measure the position of the pointer with sufficient precision. Furthermore, von Neumann introduced the pointer as the position of a free particle, which is a continuous variable. To make use of the scheme on a quantum computer, we will have to introduce a discretized version of the pointer and a discrete mapping of the energy of the target system towards

the pointer system, which is actually the same procedure as the *quantum phase estimation* algorithm (2.4.4).

It's now important to translate the continuous pointer variable into a discrete variable that can be realized using a second quantum register, the so called pointer register, consisting of r qubits. This will lead to 2^r possible pointer states, opposed to infinitely many in the continuous case. Therefore with this discretized version, we're able to measure 2^r different eigenvalues, if the mapping between the registers is accurate. Furthermore, we have to introduce two basis for the pointer register, that are related via a Fourier transform, just like the positional and momentum basis in the continuous case.

2.3 Quantum Fourier Transform

One algorithm that's very important for switching between the aforementioned basis of qubit states, and which is therefore used as a subroutine in the quantum search by measurement algorithm, is the so called *Quantum Fourier Transform* (QFT), which works analogue to the Fourier transformation of continuous signals, like it is used for example in signal processing. The next sections are based on the documentation provided by `qiskit` [40].

As with other vectors, the state vectors describing the state of our quantum system can be expressed in terms of different basis states. If no other basis is mentioned, we usually refer to the computational basis $|k\rangle$, where k denotes the decimal representation of the binary number representing the states $|a_1\rangle \otimes ... \otimes |a_n\rangle$, $a_n \in \{0, 1\}$. In this section, we will have a look at how a basic change from that basis to the so called *Fourier basis* allows us to perform certain quantum operations.

First, we're going to remind ourselves that a quantum system is described by a complete system of commuting operators and their common eigenvectors, for which one possiblity in the case of spin-1/2 particles is

$$O = \{S_z, \vec{S}^2, H\} \tag{36}$$

We have to note, that this set is maximal and therefore all other operators that commute with them, can be constructed from linear combinations. Furthermore, the set is not unique; one can for example choose the eigenvectors of S_x, $\vec{S}^2$, and H as their basis vectors. In the following section, we will see how we can extend this concept to multi-qubit systems, and how to make use of the transitions between the two basis, that are defined via the set of operators.

2.3.1 Discrete Fourier Transform

As an introduction to the Fourier basis of quantum states, we first have to have a look at the underlying classical mathematics, the so called Discrete Fourier Transform (DFT).

As a short recap, we remind ourselves what a Fourier transform looks like in the continuous case: As we know, the wave functions we're looking at have to be square integrable, $\psi(x) \in \mathcal{L}^2$. If we consider finite space, we can reduce this space to the space of 2π periodical functions. We know, that for this vector space several basis are possible, where the functions

$$\frac{1}{\sqrt{2\pi}} \qquad \frac{1}{\sqrt{\pi}}\sin(nx) \qquad \frac{1}{\sqrt{\pi}}\cos(nx) \qquad n \in \mathbb{N} \tag{37}$$

build an orthonormal basis for the space, depending on their frequency. Therefore, we can decompose any periodical function (the 2π periodicity can be extended to other periodicities) into these basis functions, which are characterized by their frequency. This directly leads us

to the idea of the Fourier transform, where we consider an infinite number of basis states, since the space is infinitely dimensional, and decompose a given signal into its frequency components. This gives us the Fourier transform of a given one dimensional function:

$$\mathcal{F}[f(x)](k) = \int_{-\infty}^{\infty} e^{-ikx} f(x) \tag{38}$$

where the e^{-ikx} represent the plane waves the signal is constructed from, and $f(x)$ is the distribution function of the plane waves with respect to their positional argument. Since in digital signal processing and on computers our signals are discrete, it makes sense to extend the idea of Fourier transforms to the discrete case. Let $x = x_1, x_2, ...x_n$ be a sequence of complex numbers, which represent our discrete signal that we want to transform. We can derive the discrete frequency components X_k with frequencies $f = 2\pi k/N$ of this sequence with the following form, analoguely to the continuous case:

$$X_k = \sum_{n=1}^{N} x_n \, e^{-2\pi ikn/N} \tag{39}$$

which gives us the amplitudes of the corresponding frequencies for a given sequence. On the other hand, from the amplitudes of the frequency components X_k of a given signal, we can conclude the time-dependent signal components following the inverse discrete Fourier Transformation:

$$x_n = \sum_{k=1}^{N} X_k \, e^{2\pi ink/N} \tag{40}$$

As we know, the amplitudes of given basis vectors of a given multi qubit state $|\psi\rangle$ are also described by a sequence of complex numbers, which will lead us to the idea of the quantum Fourier transform. These transformations between the two different basis can be understand best using *Bloch spheres.*

2.3.2 Bloch sphere

For later applications in quantum algorithms, it's important to know in which basis we can represent the qubit register, which can be easily understood by the so called *Bloch spheres.* These spheres, that represent the Hilbert space of a given two dimensional quantum state, have been introduced by Bloch et al. in a work about nuclear induction originally, but later have been used to represent spin quantum states, and especially qubits [41].

A Bloch sphere is a sphere with radius $r = 1$, on which all possible pure quantum are mapped to a corresponding point. The sphere has dimension $d = 2$, which means that any state vector can be represented by the linear combination of two arbitrary basis vectors of the sphere. If we also want to consider mixed states, we have to consider points inside of the sphere and not only on the ball's surface.

If we consider the computational basis, whose basis vectors usually refer to the poles of the sphere, we see that every state vector can be represented like

$$|\psi\rangle = \alpha \, |0\rangle + \beta \, |1\rangle \tag{41}$$

where α and β are complex coefficients, which can be writen in the cartesian complex plane as

$$\alpha = a_0 + ib_0 \qquad \beta = a_1 + ib_1 \tag{42}$$

thus, our pure state vector $|\psi\rangle$ can be fully described using four real numbers a_0, a_1, b_0, b_1. Combining this with the fact that only the relative phase between the two basis vectors is relevant, we can fix the phase of one basis vector and thus reduce the dimension of parameters needed to three. Furthermore, the state vectors have to be normalized:

$$\langle\psi|\,|\psi\rangle \stackrel{!}{=} 1 \Longleftrightarrow ||\psi||^2 \stackrel{!}{=} 1 \tag{43}$$

This leads to another reduction of our degrees of freedom and leaves us with the two dimensional Bloch sphere.

To see, how states are represented on the Bloch sphere, we have a look at the parameters. A point on the sphere is described by a polar angle θ and an azimuthal angle ϕ. With these parameters and our preliminary considerations, any pure state vector has the form

$$|\psi\rangle = \cos(\theta/2)\,|0\rangle + e^{i\phi}\sin(\theta/2)\,|1\rangle \tag{44}$$

As we can see, the two angles can be interpreted physically: The polar angle θ describes the proportion between the two basis states' amplitudes' absolute values, while the azimuthal angle ϕ represents the relative phase between the two basis vectors.

Now, we can have a look at how e.g. the basis vectors of the S_z and the S_x vectors look respectively. As we can see easily, for both eigenvectors of S_z, $|0\rangle$, and $|1\rangle$, the relative phase between the two vectors equals zero just like the azimuthal angle. This means, that our computational basis vectors are positioned on the poles of the sphere, as we discussed before.

To make the transition to the so called Fourier Basis of the quantum state, it makes sense to have a look at the eigenstates of the S_x operator in the case of one qubit. We call the eigenstates $|+\rangle$ and $|-\rangle$ respectively.

$$|+\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \qquad |-\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right) \tag{45}$$

As we can see, those eigenvectors are located on the $\theta = \pi/2$ plane, and still we can use these as basis vectors. This brings us to the idea of constructing a full basis for a multi-qubit quantum system that's fully identified by the relative phases between the computational basis states, the so called *Fourier Basis*, which will be discussed in the next section.

2.3.3 Fourier Basis of Quantum States

As we saw before, it's possible to define our single qubit state vectors in a basis that's only defined over the phase angle ϕ between the equally weighted $|0\rangle$ and $|1\rangle$ states on the Bloch sphere, and not over the ratio of the amplitudes towards one of the computational basis state vectors θ. This concept of describing single qubit systems can be extended to the description of entangled multi-qubit states, using the Fourier basis instead of the computational basis.

To understand the change of basis of a system of n qubits, we consider n Bloch sphere's representing our composite state. First, we consider a state $|\psi\rangle$, that's composed from its computational basis vectors $|j\rangle$, weighted with their amplitudes a_j:

$$|\psi\rangle = \sum_{j=0}^{N-1} a_j\,|j\rangle \tag{46}$$

in this case, since we consider n qubits, we'll have $N = 2^n$ possible basis vectors. Now, if we remember the section about discrete Fourier transformation, we can interpret the

amplitudes a_j as entries from a N-dimensional vector $\vec{a} = (a_1, a_2, ...a_N)$, or in the context of DFT, a sequence of numbers $a_j \in [0, 1]$. With this in mind, we can perform the discrete Fourier transformation on the vector representing the amplitudes of the computational basis vectors, which will leave us with the Fourier transform of the sequence $\vec{a}$, which we will call $\vec{b}$. The components b_j of the Fourier transform can be constructed in analogue to the discrete Fourier transformation:

$$b_k = \frac{1}{\sqrt{N-1}} \sum_{j=0}^{N} a_j e^{2\pi i k j/N} \tag{47}$$

We can now interpret this Fourier transform of the computational basis vectors' amplitudes as the amplitudes of the composite state vector $|\psi\rangle$ in the so called *Fourier basis*. The vector $|\psi\rangle$ can now be expressed in the computational basis, as we saw before, as well as in the new Fourier basis:

$$|\psi\rangle = \sum_{k=0}^{N-1} b_k |k\rangle \tag{48}$$

where the basis vectors $|k\rangle$ correspond to the Fourier basis. Hence, to change between the computational and the Fourier basis, we have to apply transformations to our quantum register to map the vector $\vec{a}$ to $\vec{b}$ by changing the basis of the quantum objects. Furthermore, we can perform the QFT in the opposite direction, which is called *inverse Quantum Fourier Transform* (iQFT) and will transform a state in the Fourier basis to its correspondent in the computational basis.

It's important to note here, that the quantum Fourier transform algorithm itself cannot be used to obtain the Fourier components $\vec{b}$ directly, since they are inscribed into the amplitudes of the Fourier basis vectors, that cannot be obtained with single measurements. Anyway, it has been shown by Zhou et al., that using three quantum registers, it's possible to actually obtain the Fourier components by transforming the target register, and writing the Fourier components in the computational basis [42].

To see, how the Fourier basis can be interpreted, we can have a look at the Bloch sphere. In the computational basis, the angles θ_i of each Bloch sphere determine the composite state, while for the Fourier Basis states the angles ϕ_i determine the state. Effectively, we now have a new way to encrypt numbers, apart from the binary encryption in the computational basis: We can count in the Fourier basis, starting in the basis state $|0'\rangle$:

$$|0'\rangle = \bigotimes_{i=0}^{n} \frac{1}{\sqrt{2}} \left(|0\rangle_i + |1\rangle_i\right) \tag{49}$$

which is represented on the Bloch sphere by all angles being $\phi_i = 0$. If we now want to increment the number, we rotate all qubits according to their position in the quantum register: The rightmost qubit performs half a turn around the z-axis every increment, the second rightmost qubit quarter a rotation, and so on, until the leftmost qubit performs $1/2^n$ rotations for each increment. Therefore, the largest number possible, 2^n, is stored in the Fourier basis by a full rotation of the leftmost qubit.

Intuitively, counting in the Fourier basis can be interpreted as reading the time from a clock in the Bloch representation: Each qubit represents a different time unit through the rotation around the z-Axis, while the rightmost qubit refers to the smallest time we can m (e.g. a second), and the leftmost qubit refers to the largest time that can be represented (e.g. a day). With this in head, it's easy to see that after the largest time unit has been passed,

the qubit angles ϕ_i come back to the initial state, just with real clocks when a full day has passed, leading to a periodicity of the states when further rotating, which will become important when mapping values to the register. Therefore, numbers larger than $N = 2^n$ cannot be represented using an n-qubit state in the Fourier basis, due to the periodicity of the angle ϕ_i. This should be also clear if we remember that the number of basis states of the Fourier basis has to be equal to the number of computational basis states. We have to keep this in mind for later, because some algorithms' accuracies depend on the memory capacity, and therefore on the size, of an external qubit register.

As with classical Fourier transformations, QFT will give us a powerful tool to have insight in, and to manipulate the frequency space of quantum registers, and therefore will serve as an important subroutine for implementing the discretized version of von Neumann's measurement prescription.

2.3.4 Implementation of QFT

Now that we saw how QFT allows us to change between equivalent descriptions of quantum states, we will shortly explain how to implement this algorithm technically, since it will be used in the backend of the quantum search by measurement algorithm.

We start with a quantum register in the computational basis $|\psi\rangle$ which we want to manipulate through quantum circuits, to end in a state that corresponds to the initial state in the Fourier basis. For this, the new amplitudes of the qubits in the register have to be transformed according to equation 47.

A crucial component in changing to the Fourier basis in the case of one qubit is the so called Hadamard gate A.2.1. Using a combination of Hadamard gate with gates, that induce controlled rotation around the ϕ-axis on the Bloch sphere, will allow us to change to the Fourier basis.

Hence, we will first introduce the two-qubit gate that's needed for the rotation, the so called CROT_k gate, which is a controlled gate similar to the ones discussed in section A.2.3, but induces a rotation around the ϕ-axis, like the phase shifts gates U_ϕ from section A.2.2 applied on the $|1\rangle$ state rather than a negation of the control qubit. In this case, the index k denotes the position of the control qubit and a rotation around a 2^k-th fraction of a whole rotation. The matrix form of the unitary transformation in the two qubit basis induced by the gate looks as follows:

$$\mathrm{CROT}_k = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & e^{2\pi i/2^k} \end{pmatrix} \tag{50}$$

The algorithm can be implemented as follows: First, perform a Hadamard gate on the first qubit, and after perform $n-1$ controlled rotations $\{\mathrm{CROT}_k\}_{k=2}^{n}$ on the first qubit, using the left qubits as control qubits. We do this iteratively for the other $n-1$ qubits, while always first letting the Hadamard gate act on the k-th qubit, and afterwards applying $(n-k)$ controlled rotations on it, while using the last $(n-k)$ qubits successively as control qubits. This will leave us exactly with the state transformed to its Fourier transform, which we can further use for quantum algorithms like the quantum phase estimation algorithm, which will be discussed in the next section.

To conclude the section about Quantum Fourier transformations, we want to give a bit of physical intuition that's needed for later algorithms. We know from quantum mechanics,

that the momentum basis $|p\rangle$ and the location basis $|x\rangle$ in the continuous case are connected via a Fourier transform. Therefore, we can interpret the Quantum Fourier Transform as a change of basis from the (discrete) location basis to the momentum basis and vice versa.

Now that we introduced the discretized version of the pointer using a counting register, and a method to represent it using two different basis that are related via Fourier transform, the next step is to introduce a method to perform von Neumann's measurement prescription on that discrete register. This is done using the so called *Quantum Phase Estimation* algorithm.

2.4 Quantum Phase Estimation

The subroutine for the QSM algorithm, that acts as a discretized version of the measurement scheme by von Neumann, is the so called *Quantum Phase Estimation* algorithm (QPE). It will therefore serve us as a tool to estimate the energy of the target system by mapping it to the discrete pointer register. Furthermore, the quantum phase estimation algorithm works as an important subroutine for many other quantum algorithms.

The algorithm solves a problem stated as followed: Consider a unitary operator U. As we know, unitary operators satisfy

$$|\lambda|^2 = 1 \tag{51}$$

for all its eigenvalues λ. Since complex numbers with absolute value 1 can be written in polar coordinates using Euler's representation, we can write any eigenvalue problem for a given unitary operator U and its eigenvectors $|v_i\rangle$:

$$U\,|v_i\rangle = \lambda_i\,|v_i\rangle = e^{2\pi i\theta}\,|v_i\rangle \tag{52}$$

where θ denotes the polar angle in the complex plane in polar coordinates. Therefore, to know the eigenvalue of U for a given eigenvector v_i, it's enough to know the angle θ since one degree of freedom is fixed by the absolute value.

The goal of the quantum phase estimation algorithm is to give an estimation of the phase θ induced by U on a given eigenvector. For this, we have to use two seperate quantum registers: One register that's initialized in the state $|\psi\rangle$, which will be manipulated by U and whose phase shift θ we want to estimate, and the other register as a binary representation of that phase after the algorithm has run. We call the second register the *counting register*. The number of qubits in the counting register gives us the number of digits we're given for θ and is therefore crucial for the accuracy we want to estimate θ for. If we consider r counting qubits, the final state of the counting register that we're going to measure will have the eigenvalue $2^r\theta$.

To understand how the phase induced to $|\psi\rangle$ is written into the counting register, we have to introduce yet another subroutine that allows us to transfer phase shifts induced by unitary transformations U to another set of qubits, the so called *Phase Kickback*.

2.4.1 Phase Kickback

The phase of the underlying unitary operator is induced from the target system, which is setting the applied phase, to the pointer register using phase kickback. This can be understood easiest in the gate-based implementation of quantum computers. For this, we need so called controlled-not gates (CNOT) (A.2.3), which are two qubit gates, where one qubit acts as a control qubit (similar to a transistor current), while the other qubit, the target qubit, gets negated if the control qubit is in the state $|1\rangle$.

As it will be discussed in the appendix A.2.3, we can apply the CNOT gate on a two-qubit system in the Fourier basis, which will negate the control bit instead of the target bit. It makes sense to extend this concept to general controlled gates, where we again consider a system consisting of one control and one target qubit. Let ϕ be the phase induced on the target qubits initial state $|\psi\rangle$ and the control qubit be in the state $|0\rangle$. If we now apply a Hadamard gate on the control qubit, and the controlled U gate on both qubits afterwards, we will get

$$(C(U))\,(H \otimes \mathbb{1})\,|\psi\rangle \otimes |0\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{i\phi}\,|1\rangle\right) \otimes |\psi\rangle \tag{53}$$

since the Hadamard gate changes the counting qubit to the S_x basis, this means that in this case, the target unit's state $|\psi\rangle$ has not been changed, and the phase that was meant to be introduced to $|\psi\rangle$ now has been applied to the control qubit. This means, that we can apply the phase that's dependent on the initial state $|\psi\rangle$ onto another dummy state that lets us store the phase. This is called *phase kickback*, and can be used in the quantum phase estimation algorithm as follows: We try to estimate the phase induced by an operator U onto its eigenstate $|\psi\rangle$. With the phase kickback subroutine, we can now induce that phase θ, which represents the eigenvalue of U, to the counting register without changing the state we actually want to examine, and measure the state of the counting register in the computational basis after performing an inverse Quantum Fourier Transformation to get an estimate for the eigenvalue of U.

2.4.2 Heuristic approach for QPE

To provide some intuition regarding how phases are measured using QPE, we will start with a heuristic example. We consider again one target qubit in the eigenstate $|\psi\rangle$ of a given operator U with eigenvalue $e^{i\phi}$, and one counting qubit. After applying a Hadamard gate to the counting qubit, we will apply the phase ϕ to it via phase kickback. It will then transform to

$$U\,|+\rangle = U\frac{1}{\sqrt{2}}\left(|1\rangle + |0\rangle\right) = \frac{1}{\sqrt{2}}\left(|1\rangle + e^{i\phi}\,|0\rangle\right) \tag{54}$$

As we see, the phase is now encrypted in the probability of measuring the ancillary state. If we perform a sufficient amount of measurements in the X-basis, we can investigate on that probability and therefore estimate the phase ϕ. The problem with this procedure is, that the accuracy scales with the number of measurements r performed as $1/\sqrt{r}$. To reach higher accuracy in estimating the phase, we will make use of the quantum nature of the system:

For sake of simplicity, we will now assume that $\phi \in \{0, \pi\}$. If the phase satisfies this, we will measure the correct phase with 100% accuracy , since in both cases we will transform the state into an eigenstate of the S_x-operator. We can extend this range of possible angles being display with 100% accuracy by introducing the concept of binary fractions.

2.4.3 Binary Fractions

To be able to display an angle ϕ, and therefore a relative phase using several binary measurements, we use the concept of *Binary Fractions*. The idea behind this is to extend the concept of integers displayed in the binary system to rational numbers. For this, we remember how integers are displayed in the binary system. Analoguely to the decimal system, the

j-th digit represents the coefficient of 2^j, being either 0 or 1. With this in head, we can also display fractions as follows:

$$1 \equiv 1.0; \quad \frac{1}{2} \equiv 0.1 \quad \frac{1}{4} \equiv 0.01 \tag{55}$$

With this in mind, we can display any fraction if we use a sufficient number of bits. In terms of the phase estimation algorithm, we will now denote our phase as

$$\phi = 2\pi\omega = 2\pi \sum_{j=1}^{\infty} b_j 2^{-j} \tag{56}$$

As we see, the parameter ω now gives us the fraction of a 2π turn associated with the phase ϕ as a binary fraction.

Let's now assume we want to measure the angle $\phi = \pi/2$. As a binary fraction, this will be denoted as 0.01. For this, it makes sense to see what happens if we apply the phase two times, by applying the corresponding unitary operator twice: It will result in the $|-\rangle$ state, which is an eigenstate of S_x and can therefore be measured with certainty. This means, that by applying the phase two times, and measuring after, we can gather information about the last digit of the binary fraction being one or zero.

Now, to gather information about the next and last bit, we will shift the phase back according to the last bit ($-\pi/2$), and apply the phase one time. This will result in the state $|+\rangle$, and therefore yield a zero, leaving us with $\phi = 2\pi \cdot 0.01 = \pi/2$.

We can extend that concept for an arbitrary number of significant digits: First, we apply the unitary transformation U 2^M times to the counting qubit and measure it, giving us the value of the least significant bit. Afterwards, we keep in mind whether we reached the $|+\rangle$ or $|-\rangle$ state after the rotation and apply the inverse rotation depending on the outcome of the first measurement.Then we apply $U^{2^{(M-1)}}$ to gather information about the $(M-1)$-th bit and so on, until we reached a full M-bit binary fraction representation of our angle ϕ.

2.4.4 Implementation of QPE

In comparison to the previously discussed naive scheme, it's possible to perform all the measurements for the bits representing ϕ simultaneously on a quantum computer, using r-qubits that inhabit the digits of the binary fraction. In terms of accuracy, with each ancillary qubit we add to the pointer register, we can measure twice the amount of angles we could measure certainly without the added qubit, which means that our accuracy in estimating the angle now goes as $\frac{1}{2^r}$, which is a significant speed up in comparison to the initially discussed naive scheme providing an accuracy of $\frac{1}{\sqrt{r}}$. This comes from the fact, that we actually perform discrete measurements on a continuous variable (the phase angle), where the area where ϕ can possibly found gets split in half for each ancillary qubit we add, leading to the exponentially growing accuracy.

We can implement the algorithm, performing all measurements simultaneously as follows: First, we have to apply Hadamard gates to all the qubits in the counting register, to change to the Fourier basis and make use of the quantum phase kickback algorithm. The other register will start in the state $|\psi\rangle$ whose eigenvalue we want to estimate. The Hadamard gates' action on the composite system can be described as

$$H^{\otimes n} |0\rangle^{\otimes n} \otimes |\psi\rangle = \frac{1}{2^{n/2}} (|0\rangle + |1\rangle)^{\otimes n} \otimes |\psi\rangle \tag{57}$$

This means that we changed all the control qubits' states to the S_x basis. Now, we have to apply controlled U^{2^j} gates $C_j(U^{2^j})$, that correspond to the j-th qubit in the counting register. The application of U^{2^j} yields:

$$U^{2^j} |\psi\rangle = e^{2\pi i \theta 2^j} \tag{58}$$

Thus, successively controlling the U^{2^j} transformation by the j-th qubit of the counting register shifts the relative phase ϕ between the computational basis states $|0\rangle_j$ and $|1\rangle_j$ from zero, like it has been initialized through the Hadamard gate, to encrypt the j-th significant binary digit of $2^j\theta$ in the Fourier basis. Therefore, to represent a non-rational phase θ, in general many qubits are necessary for sufficient accuracy. It's important to note here, that the order of the qubits is not arbitrary, since the position of the qubit in the register will later correspond to the significancy of the digit it's state is representing. The composite state $|\psi'\rangle$ after applying the controlled rotations looks as follows:

$$|\psi'\rangle = \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi i \theta k} |k\rangle \otimes |\psi\rangle \tag{59}$$

After all the n counting qubits' phases have been shifted accordingly we can perform an inverse Quantum Fourier Transformation to translate the phase shifts that were induced into a binary number we can measure in the computational basis and hence get a value for $2^n\theta$, which we can divide by 2^n to get an estimation for θ. The iQFT leaves us with the state

$$\frac{1}{2^n} \sum_{x=0}^{2^n-1} \sum_{k=0}^{2^n-1} e^{\frac{-2\pi i k}{2^n}(x-2^n\theta)} |x\rangle \otimes |\psi\rangle \tag{60}$$

Now we can measure it to get an estimation for θ. We have to note here, that the state also has contributions $|x\rangle \otimes |\psi\rangle$ with $x \neq 2^n\theta$, but the amplitudes in dependence of the phase shift and the counting register in the computational basis peak at $x = 2^n\theta$, which implies that we'll measure the right value of the counting register with high probability.

2.5 Implementation of QSM

Having discussed the basics behind the theory of von Neumann's measurement scheme and the quantum zeno effect, we can now combine the results and dive into the quantum search by measurement algorithm for finding ground states of Ising type Hamiltonians. The basic idea behind the algorithm is similar to the adiabatic quantum computation concept:

We start in the known ground state of a given initial Hamiltonian H_i, while the ground state should be both easy to find, and easily to prepare technically on a real quantum computer. We then let the Hamiltonian propagate towards a final Hamiltonian (see section 1.4) H_P, whose ground state encrypts the desired solution to the problem. In contrast to adiabatic quantum computation, we will perform measurements after each evolution $H(s) \to H(s+\delta)$, which will let the system collapse into the eigenstate of $H(s+\delta)$, that's closest to the previously measured state of $H(s)$. As the quantum zeno effect 2.1 states, if the time δ after each measurement is chosen small enough, the system is very likely to be kept in the corresponding ground state of each Hamiltonian $H(s)$. After we reached the last Hamiltonian $H(s=1)$, the last measurement will therefore yield the final Hamiltonian's ground state with high probability.

Therefore, a crucial point in terms of accuracy of the algorithm is the number of timesteps $1/\delta = M$ we want to evolve our system for to perform projective measurements after, because this determines the number of Hamiltonians we're transitioning through until we reached the final Hamiltonian, and thus also the change in the ground state energies $E_0(s)$ while evolving in s. We divide the time evolution of the Hamiltonian in M steps, where $1/M = \delta$ is the dimensionless evolution time between two Hamiltonians $H(s)$ and $H(s+\delta)$. It's very important to choose M accordingly to the problem to reach sufficient accuracy in the algorithm, which will be discussed later.

Let's assume we will have r qubits in our pointer register, which will lead to 2^r possible pointer basis states, that represent the analogue to 2^r different positions in the case of the previously discussed continuous pointer. We call the state representing our pointer $|z\rangle$, while z denotes the integer representation of the binary number representing the eigenstate in the S_z-basis. We can then introduce an operator p, that works analoguely to the momentum in the continuous case and is defined as follows:

$$p = \sum_{j=0}^{2^r-1} 2^{-j} \frac{1-\sigma_z^j}{2} \tag{61}$$

where σ_z^j denotes the Pauli z-matrix acting on the j-th qubit in the pointer register. According to the normalization chosen, this will result in the following operator-pointer interaction of p on its eigenstates:

$$p\,|z\rangle = \frac{z}{2^r}\,|z\rangle \tag{62}$$

Now, we can construct a discrete state $|z_0\rangle$, which corresponds to the initial state $|x=0\rangle$ in the continuous case:

$$|z_0\rangle \equiv |x=0\rangle = \frac{1}{2^{r/2}} \sum_{z=0}^{2^r-1} |z\rangle \tag{63}$$

which is equivalent to the normalized uniform superposition of all basis states of p, and can be constructed initializing the system in $|0\rangle = |0\rangle \otimes |0\rangle \otimes ... \otimes |0\rangle$ and performing an inverse Fourier transform afterwards. This works, since, if we have a look at the formulation in terms of frequencies, a perfectly distinct state in momentum space $|0\rangle$ has to result in a uniform superposition in the position space, which is analogue to the continuous Fourier relation. We have to note here, that we express our pointer state in terms of its momentum eigenstates to perform the time evolution induced by p upon it. To make use of the position eigenstate by measuring it, we first have to perform an inverse Fourier transformation on the state to bring it back to its position representation. Evolving the composite state $|\psi_0\rangle \otimes |x=0\rangle$ according to the measurement scheme will allow us to perform measurements on the discrete pointer to keep the target system in the ground state and gather information about the corresponding energy, if the mapping is accurate.

After we changed the Hamiltonian $H(s) \to H(s+\delta)$ we let our combined Hamiltonian $H(s+\delta) \otimes p$ interact with the system for a time τ, whose magnitude is yet to be discussed. The time evolution induced by the Hamiltonian on the composite state initialized in an energy eigenstate with the pointer $|x=0\rangle$, expressed in the Fourier basis $|z\rangle$ can be written a follows

$$e^{-iH\otimes p\tau}\,|E_a\rangle \otimes |x=0\rangle = \sum_{z=0}^{2^r-1} e^{-iE_a z\tau/2^r}\,|E_a\rangle \otimes |z\rangle \tag{64}$$

Where the exponential $e^{-iH\otimes pt}$ acts on a large Hilbert space, which is why approximative techniques like the Suzuki-Trotter-Decomposition can be used to simulate the time evolution using few-qubit gates [43].

We can see the previously discussed phase estimation algorithm here: The phase $E_a\tau 2^r$ that's dependent on the target system $|E_a\rangle$ has been kicked back to the qubit register by the time evolution operator, by rotating each qubit dependent on their position in the register appropriately, to encrypt the phase to the pointer register. The factor $1/2^r$ therefore comes from the normalization of the phase regarding the size of the pointer register. This is possible, since unitary transformations' eigenvalues always have magnitude 1 and thus can be interpreted as a shift in phase. Hence, gathering information about the system by performing measurements on the pointer variable, is the same as estimating the phase induced by the unitary operator acting on the target system, and kicking that phase back to the pointer register. Thus, the whole evolution process $H(s=0) \Rightarrow H(s=1)$ can be interpreted as applying the quantum phase estimation algorithm M times according to the number M of measurements during the evolution process chosen.

To actually measure the pointer after the last evolution, we have to transform it back from the Fourier basis to the computational basis, which leaves the pointer in the state $|\phi\rangle$, expressed in the discrete 2^r-dimensional positional basis $|x\rangle$, $x = 0, ..., 2^r - 1$ [44]:

$$|\phi\rangle = \sum_{x=0}^{2^r-1} \left(\frac{1}{2^r} \sum_{z=0}^{2^r-1} e^{2\pi i(x-E_a\tau)z/2^r} \right) |x\rangle \tag{65}$$

which is peaked at $x = E_a\tau$, analoguely to the result of QPE, which means that the contribution of one pointer state towards the correspondent target state is the highest. Nonetheless, as we discussed in section 2.4.4, there's a mapping between the target system towards a discrete pointer system, which is not bijective. To avoid measuring a pointer state $|x\rangle$ that doesn't actually correspond to the target state's energy, we have to minimize the contributions of given pointer states towards "wrong" energies, which will be discussed in section 2.6.

While it's possible to actually conclude a system's energy with von Neumann's measurement scheme, we're actually not interested in the energy, but rather in the fact, that the system will collapse in one of its energy eigenstates after the measurement. Therefore it's not actually necessary to perform an iQFT on the pointer system after the measurement, since the measurement procedure itself already let the target system collapse into one of its eigenstates.

For simulating the algorithm, we can trace over the pointer's degrees of freedom after performing the time evolution under the composite Hamiltonian $H(s) \otimes p$ and an iQFT on the pointer register, since we're only interested in the behaviour of the target system. This will leave us with the reduced density matrix of the target system, which then can be evolved under the next Hamiltonian $H(s+\delta)$ using a new pointer register. We repeat this pattern from $H(0)$ until we reached the Hamiltonian $H(1)$, for which we again perform the measurement. This last measurement should provide us with the ground state of $H(1)$, in which the solution to an optimization problem can be encrypted. The action of one Hamiltonian evolution and system-pointer interaction can also be denoted recursively in terms of the entries a, b of the reduced density matrix $\rho_{ab}^{(j)}$ after the Hamiltonian has been evolved j times:

$$\rho_{ab}^{(j+1)} = \kappa_{ab}^{(j+1)} \sum_{c,d}^{2^n} U_{ca}^{(j)} \rho_{ab}^{(j)} U_{bd}^{(j)*} \tag{66}$$

where the $U_{ab}^{(j)}$ denote the overlaps between the eigenenergies of the jth Hamiltonian, and the $(j+1)$th Hamiltonian:

$$U_{ab}^{(j)} = \langle E_a((j+1)\delta) | E_b(j\delta) \rangle \tag{67}$$

and the $\kappa_{ab}^{(j)}$ inhabit the trace over the pointer system, and thus the coefficients of the position basis states $|x\rangle$ of our pointer regarding a transition between two energies E_a, E_b (see equation 65).

$$\kappa_{ab} = \frac{1}{2^r} \sum_{z=0}^{2^r-1} e^{2\pi i (E_b - E_a) t z / 2^r} \tag{68}$$

It's very important to note the role of the κ_{ab} in this case, which represent the coefficients of a given pointer state $|x\rangle$ in dependence of the energy of the system, given in equation 65. Therefore, if normalized, $|\kappa|^2$ will serve as a probability distribution for the system being measured in the right or wrong state depending on the target system. In the next section, we will show why the mapping between the target and the pointer system is crucial for the accuracy of the algorithm with respect to its running time.

2.6 Running time

As we discussed previously, the crucial variables in terms of resources needed for our algorithm, are on the one hand the number of intervals M we want to divide our Hamiltonian's evolution in, and the time τ for which the system is left to interact with the pointer. If we remember the quantum zeno effect, the system is only guaranteed to be kept in the ground state for short evolution times after each measurement and thus only in the limit of infinitely many measurements $M \Rightarrow \infty$ during the evolution between the initial and the final Hamiltonian.

The other crucial part is, that if we don't use as many ancillary qubits as qubits in the target system, there is no perfect mapping between the counting register and the target register, which means that it's possible to measure a state in the counting register, that doesn't correspond to only one eigenenergy of the target system.

Because of these two facts, we have to choose the number of measurements M, the evolution time to map the target state's phase to the counting register τ, and the number of qubits in the counting register r adequately.

It has been shown in the paper by Childs et al. that the number of measurements M needed for the success probability being close to 1 is dependent on two variables of the system: On the one hand, as we discussed previously in the section about adiabatic quantum computation, the probability for a transition between two states $|E_0\rangle \Rightarrow |E_1\rangle$ with $E_0 \neq E_1$ depends on its gap $|E_0 - E_1|$. Since we're interested in M different Hamiltonians during the evolution, we have to take care of the minimum gap between the ground state's energy and the first excited state's energy of *all* Hamiltonians we're evolving through. Thus, the first variable that's interesting in terms of running time is the minimum spectral gap g over all

Hamiltonians:

$$g = \min_{s\in[0,1]} g(s) = \min_{s\in[0,1]} |E_1(s) - E_0(s)| \tag{69}$$

The other important quantity is the maximum amount of fluctuations regarding the ground state energy $E_0(s)$ while evolving in s:

$$\Gamma = \max_{s\in[0,1]} \Gamma(s) = \max_{s\in[0,1]} \langle E_0| \left(\frac{\mathrm{d}H}{\mathrm{d}s}\right)^2 |E_0\rangle - \langle E_0| \left(\frac{\mathrm{d}H}{\mathrm{d}s}\right) |E_0\rangle^2 \tag{70}$$

This quantity is crucial in terms of success probability, because it determines the energy fluctuations while the Hamiltonian is evolving, and therefore determines how *close* the energy will fluctuate to the first excited state and thus the probability of transition. Both of these quantities are defined upon the Hamiltonians $H(s)$ we're transitioning through, which is why choosing a different schedule (1.4) can change these parameters and therefore yield a different accuracy of the algorithm.

Combining the results, it can be shown that the probability of the target system being in the ground state is close to 1 if the number of measurements M satisfies

$$M \gg \frac{\Gamma^2}{g^2} \tag{71}$$

as we discussed before, the total computation time is given by $T = M\tau$. Now that we investigated on the minimum number of measurements, we have to see for what time τ we have to let our system interact with the pointer to be able to determine the system's energy with sufficient accuracy by measuring the pointer.

Since the parameter τ determines how *far* the pointer propagates for each evolution, it also determines the spacings between the mappings of the different energies E_i. Therefore the evolution time has to be large enough to resolve the different energies with respect to the pointer variable. Hence, the evolution time τ determines to which pointer state $|x\rangle = |\tau E_a\rangle$ a distinct eigenenergy E_a will be mapped.

As we discussed in the previous section, we want the contributions of pointers pointing towards wrong energies, κ_{ab} with $a \neq b$ low to avoid non-projective measurements. Due to the fact that we want our system to stay in the ground state of all Hamiltonian's we're transitioning through, we're actually only interested in the contributions κ_{0b} with $b \neq 0$ of coefficients of pointer states $|b\rangle$ where the target system is in the ground state. Furthermore, we need to restrict these contributions for all Hamiltonians, and are therefore interested in the maximum contribution $\kappa_{\max} = \max_{b,j} \kappa_0 b^{(j)}$. Constraints on this coefficient of the pointer state let's one derive a lower bound for the evolution time to resolve the ground state from the first excited state:

$$\tau \gg \frac{\mathcal{O}(1)}{g_{\min}} \tag{72}$$

This comes from the fact, that $E_0\tau$ and $E_1\tau$ have to be mapped to different pointer states, which are represented by integers in the Fourier basis. Hence, the product of τ and $_{\min}$ should be close to an integer for the mapping to be accurate. This is important, because the mapping will determine to which eigenstate the target system actually collapses when a distinct pointer state is measured; in the worst case of all eigenstates being mapped to the same pointer, a measurement of that pointer will not induce a collapse of the target system at all, leading to the algorithm to fail. It's also important to mention here, that we also

should place a lower bound to τ, since values $E_i\tau > 2\pi$ get subject to the periodicity of the pointer state. Therefore τ shouldn't exceed $\frac{2\pi}{E_{\max}}$, for the maximum energy value $E_{\max}$ occurring in the target spectrum.

Another crucial variable for the mapping between the target system and the pointer system is the number of pointer qubits r used, since this will determine the maximum position $|x_{\max} = 2^r\pi\rangle$ after which the periodicity of the pointer will affect the outcome of the mapping. We therefore have to choose r adequately large to provide enough space towards higher excited states, since if r is chosen too small, we can not provide sufficient pointer states for all the relevant energy eigenstates of the target system. This places as a lower bound for the sufficient number of ancillary qubits r:

$$r = \mathcal{O}(\log n) \tag{73}$$

Combining the results for the number of measurements M and the evolution time per measurement τ leaves us with a lower bound for the total running time T to have a success rate close to one:

$$T \gg \frac{\Gamma^2}{g_{\min}^3} \tag{74}$$

It's again important to note here, that both $g_{\min}$ and Γ depend on the Hamiltonians $H(s)$ that are applied, and therefore also on the way of interpolating between $H(0)$ and $H(1)$. Henceforth, as we discussed in section 1.4, the applied schedule also states an important variable in terms of the accuracy of the algorithm, on which we're further going to investigate in the next sections.

3 Simulation of the algorithm

Now that we gathered all the important prerequisites, we can simulate the quantum search by measurement algorithm, for optimization problems, to investigate on its accuracy in finding H_P's ground state, and therefore the optimal solution to the encrypted problem.

3.1 Methodology

As a tool to find whether the system is in the ground state of the underlying Hamiltonian $H(s)$ in terms of its density matrix, we will use the fidelity F (1.7) between the *real* ground state $\psi_0(s)$, that has been found using `qutip`'s master equation solver, and the current state ψ_{com}, that has been found by simulating the quantum search by measurement algorithm: Since two density matrices are considered describing equivalent systems, if their fidelity is close to one, our goal is to keep the fidelity close to one for all Hamiltonian's we're transitioning through.

$$F(s) \equiv F(\psi_{\text{com}}(s), \psi_0(s)) = \text{tr}\left(\sqrt{\sqrt{\rho_{\text{com}}(s)}\sqrt{\rho_0(s)}}\right)^2 \tag{75}$$

Where ρ are the density matrices corresponding to the pure states $|\psi\rangle$. The most crucial variable in this case, is the fidelity of the problem Hamiltonian H_P's ground state compared to the computed one, $F(1)$, since it's a way to quantify the distance of the computed solution to the real optimal solution.

As discussed in section 1.6, it's possible to encode several kinds of optimization problems into the problem Hamiltonian H_P, which is why we're choosing it to inhabit nearest

neighbour interactions, with random coupling strengths J_i, that can be interpreted as the correlation effects of two given sites on the cost function, and a single site local field term, that can be interpreted as the associated cost function of a given site:

$$H_P = \sum_i J_i \sigma_i^z \sigma_{i+1}^z + \sigma_i^z \qquad J_i \in [-1, 1] \tag{76}$$

To display a wider range of possible problems that can be encrypted in such a model, we compute F for a set of 30 random interaction strength matrices J, and display the mean values and standard deviations of F in the corresponding plots. It's again important to note here, that the algorithm will fail if the Hamiltonian $H(s)$ is gapless, which is why we chose the interaction strengths accordingly to avoid energy levels crossing.

Furthermore, we define the initial Hamiltonian H_i inhabiting only the transverse field interaction terms, with all coupling strengths h_i set to one:

$$H_i = \sum_i h_i \sigma_i^x \qquad h_i = 1 \tag{77}$$

This initial Hamiltonian has, as discussed in section 1.4, an easily preparable ground state, from which we start and then develop our system according to section 2. We're then able to investigate on the dependency on the set parameters (2.6) by computing the fidelities $F(s)$.

For this, we first construct all the Hamiltonian's between H_i and H_P, by using python's list functions, and compute their real ground states and corresponding energies using `qutip`'s master equation solver. With this reference, we can start with the simulation of the quantum search by measurement algorithm:

We chose the initial Hamiltonian to be a sum of σ_x terms, which is why we already know its ground state, as we should:

$$|\psi_0(0)\rangle = \bigotimes_{i=1}^{N} |+\rangle \tag{78}$$

Afterwards, we define our coupled Hamiltonian as the tensor product between the system Hamiltonian $H(s)$, and the pointer Hamiltonian p from equation 61, to be able to perform time evolution, that kicks the phase induced by $e^{-iH(s)t}$ back to the pointer register. We then perform an inverse quantum Fourier transformation on the pointer system, and trace over it to get the reduced density matrix of the system after the simulated measurement. Repeating this pattern until we reached the final Hamiltonian H_P should provide us the ground state, if the parameters were chosen adequately beforehand.

3.2 Accuracy of the algorithm

The evolution time τ, the total number of measurements M, the number of ancillary pointer qubits r, and the chosen schedule are crucial for the algorithm's accuracy (2.6), which is why in this section we will investigate on the dependency of the algorithm's accuracy, expressed by the fidelity $F(s)$ between the ground state and the computed state.

3.2.1 Time development of the fidelity

Since the fidelity $F(s)$ for each timestep s shows us how close the algorithm brings us to the ground state of the corresponding Hamiltonian $H(s)$, we will now investigate on the evolution of the fidelity over time. Also, since the chosen schedule is crucial for the minimum energy gaps $g_{\min}$ and the energy fluctuations Γ, we're performing all simulations for four differently chosen polynomial schedules (15), characterized by their exponent k:

$$k \in [1, 1.5, 2, 3] \tag{79}$$

The other parameters for the simulation were set as follows: We simulate a chain of eight spins, set the number of measurements to $M = 100$, the evolution time to $\tau = 10$, and the number of pointer qubits to $r = 2$. Then we started simulating the algorithm and investigating on its accuracy:

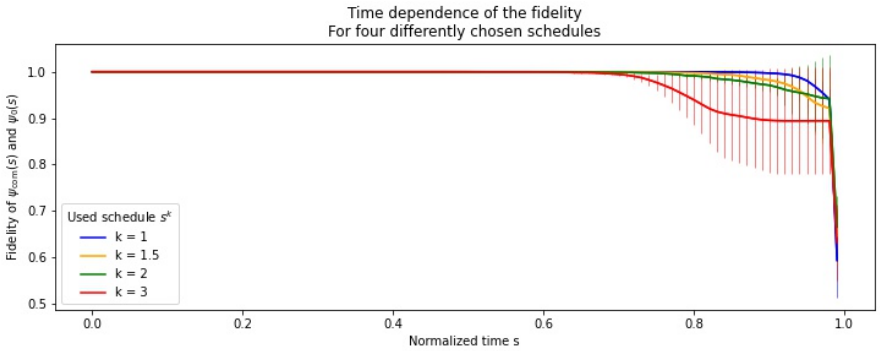

Figure 3: The evolution of the fidelity between the computed state $\psi_{\text{com}}(s)$ and the real ground state $\psi_0(s)$ for four differently chosen schedules

As we can see, the fidelity starts at one, which should be clear since we're starting in the exact ground state of H_i. It then drops, but stays above 0.5 for all times s. Furthermore, the fidelity stays near one for all four schedules, while the time-dependent behaviour with respect to the chosen schedule only starts changing in the second half of the evolution. This shows, that the crucial parts in the evolution of $H(s)$ are when the Hamiltonian is close to the final Hamiltonian H_P.

As it can be seen, the time-dependent fidelity $F(s)$ and especially the final fidelity $F(1)$ heavily depends on the chosen schedule, where $F(1)$ is around ≈ 0.1 higher for the schedule $k = 2$ than it is for the worst performing one, the linear schedule. What's noticeable here is that F doesn't necessarily scale with the size of the exponent k: From the outcome of the graphs, the linear interpolation performs worst while the $k = 1.5$ and $k = 2$ schedules perform better, but the $k = 3$ schedule doesn't provide more accuracy just of the polynomial interpolation being of higher order. To further investigate on the dependency on the chosen schedule, we will perform all other simulations for all four schedules from this section.

Since in terms of optimization problems the crucial variable is the outcome of the measurement of the last Hamiltonian H_P, the final fidelity $F(1)$ is the important quantity to look out for. Since figure 3 doesn't provide enough information about this value and its schedule dependency, we're going to further investigate on $F(1)$ and its dependence on the parameters mentioned in 2.6.

3.2.2 Number of measurements

We will now investigate on the accuracy of the system's final fidelity $F(1)$ with respect to the number of measurements performed during the algorithm. As discussed in section 2.6, the ground states of two consecutive Hamiltonians only have a sufficient overlap, if the two Hamiltonians don't differ too much. Therefore, the system is only guaranteed to be kept in the ground state in the limit $M \to \infty$. For the other parameters, this time we chose $\tau = 5$, and $r = 4$, to prevent a bias regarding fixed secondary variables.

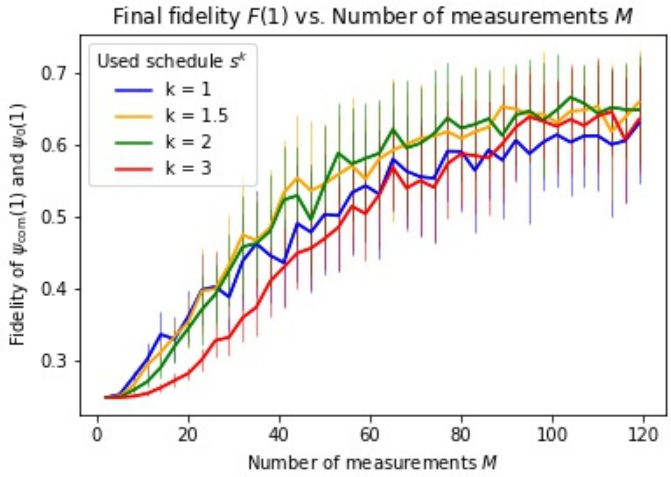

Figure 4: The dependency of the final fidelity between the computed state $\psi_{\text{com}}(1)$ and the real ground state $\psi_0(1)$ on the number of measurements performed throughout the simulation, for four differently chosen schedules

As we can see, the behaviour of $F(1)$ with respect to M is almost the same for all chosen schedules, where again the schedules $k = 1.5$ and $k = 2$ perform best. As we were expecting, the accuracy of the algorithm in terms of the final fidelity rises with an ascending amount of measurements M, which can be explained with the quantum zeno effect 2.1, since shorter times $\delta = 1/M$ between the measurements increase the chance of the system collapsing into the ground state of $H(s+\delta)$ when being measured, because of the energy levels being closer to each other.

Now that we investigated on the dependency on the total number of measurements M, and could confirm the results from Childs et al. and the quantum zeno effect, we will now simulate on how the evolution time τ per measurement affects the outcome of the algorithm.

3.2.3 Evolution time

The evolution time τ per measurement is crucial in terms of the mapping between the target system and the pointer system, since it determines the spacings between the energy eigenstates in the pointer space, and therefore has to be chosen adequately large to resolve the states with different eigenenergies from each other. On the other hand, choosing τ too large can cause overflow errors because of the periodicity of the pointer states. Therefore, in this part we're going to investigate on the accuracy of the algorithm with respect to the evolution time τ per iteration, while choosing the other parameters as $M = 30$ and $r = 4$.

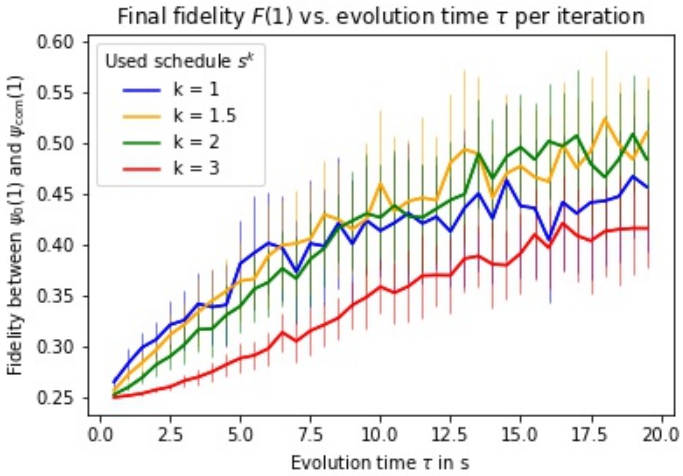

Figure 5: The dependency of the final fidelity between the computed state $\psi_{\text{com}}(1)$ and the real ground state $\psi_0(1)$ on the evolution time τ per iteration, for four differently chosen schedules

For this, we calculated the final fidelities $F(1)$ for 30 different random optimization problems with respect to τ, and in dependence of the used schedule. The mean values of the final fidelities and the standard deviations can be seen in figure 5.

As we were expecting, we see that the accuracy of the algorithm defined in terms of the final fidelity $F(1)$ and the energy error $\Delta E(1)$ will increase with increasing evolution time τ, due to the mapping between both registers being more precise and therefore less contributions in the density matrix towards "wrong" eigenstates. Furthermore, the final fidelity seems to start stagnating with increasing τ, which has an easy explanation: Since we're mainly interested in the energy gap $|E_1(1) - E_0(1)|$, we need our evolution time sufficiently big to map both eigenenergies to two different pointer states. If higher energy contributions affect our target state, we also have to map these to distinct pointer states. Therefore, if τ is chosen adequately large to resolve the spacings $|E_i(s) - E_j(s)|$ in the pointer register, increasing it further will not necessarily increase the accuracy of the algorithm expressed in terms of $F(1)$. Instead, the accuracy can even worsen if the evolution time is choosen to big, leading to higher energy eigenstates being mapped towards low energy pointer states because of the periodicity of the pointer states in the Fourier basis.

3.2.4 Number of pointer qubits

Regarding the mapping between the measured pointer register, and the target register, the other crucial variable besides the evolution time τ is the number of ancillary pointer qubits used r, since this will set the *size* of our pointer register, or in other words, how many different pointer states we can measure. Therefore we have to choose r adequately large, to provide enough space for all the relevant eigenenergies occuring in the target system's spectrum, and on the other hand as small as possible to keep the number of qubits used for computation low. For this, we again calculated the fidelity $F(1)$, this time in dependency on the number of pointer qubits used, with secondary parameters $\tau = 10$ and $M = 30$.

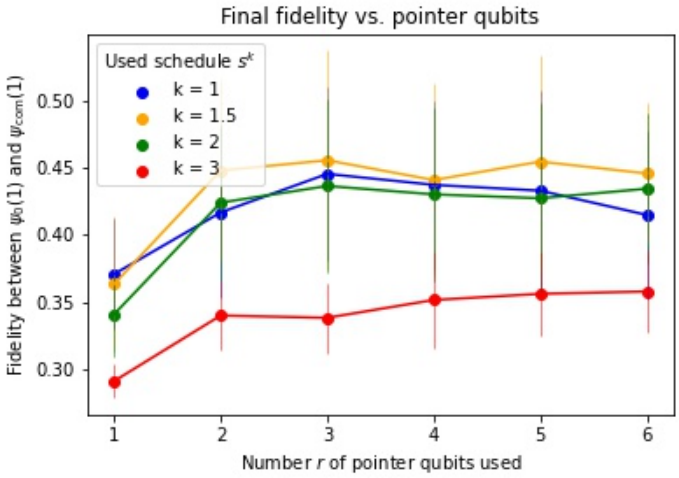

Figure 6: The dependency of the final fidelity between the computed state $\psi_{\text{com}}(1)$ and the real ground state $\psi_0(1)$ on the number of pointer qubits used r, for four differently chosen schedules

What's interesting here is that there's no indicator for direct dependency of r and $F(1)$, in contrast to the previously examined parameters M and τ. The fact that there's no significant increment in $F(1)$, even if more ancillary qubits can have several reasons: One possibility is, that since the most crucial energy gap $|E_a(s) - E_b(s)|$, that needs to be resolved for the algorithm to succeed, is the one between the ground state and the first excited state, $|E_1(s) - E_0(s)|$, fewer pointers than the number of computational qubits N are necessary to provide the highest accuracy with respect to r: If the mapping regarding τ is chosen adequately, even two pointer states can suffice, in the case of $E_0\tau$ being mapped to the first pointer state, and $E_1\tau$ being mapped to the second one. The fidelity still rising after $r = 1$ therefore indicates that we also have higher energy contributions in the target state, for which we have to provide pointer states they can be mapped to.

Now that we investigated on the dependency of $F(1)$ on the chosen schedule, the number of measurements M, the evolution time τ, and the number of pointer qubits r, we will combine the found relations to reach a fidelity as close to one as possible. We chose the number of measurements adequately large, with $M = 300$ to prevent Landau-Zener-Transitions, we chose $\tau = 20$ to resolve the ground state energy E_0 from the first excited energy E_1 with certainty, and chose $r = 3$ pointer qubits, as well as a schedule of s^2, because its

performance was among the best. Combining these results we could achieve a fidelity of $F(1) = 0.696 \pm 0.004$, which indicates a success probability of $\approx 70\%$ for the chosen set of parameters. The fidelity could be further improved by varying the schedule, for example with a non-polynomial schedule, or by further increasing the number of measurements M and performing a more far-reaching analysis on the mapping between the target and the pointer register.

4 Conclusion

The quantum search by measurement algorithm is an algorithm by Childs et al., that uses a discretized version of von Neumann's measurement scheme or the quantum phase estimation algorithm respectively, to find the ground state of a given Hamiltonian H_P. We found that, since solving Ising models can be mapped to optimization problems, this scheme can be used for solving these kinds of problems with high accuracy, if the parameters are set adequately.

We investigated on the accuracy on finding such ground states of Ising Spin Glass models, that can be interpreted as random optimization problems, that are used for benchmarking optimization algorithms. By applying the algorithmic framework for different sets of parameters, we found that the accuracy of the algorithm, defined in terms of the fidelity $F(1)$, heavily depends on these chosen parameters: The number of measurements M between the Hamiltonians H_i and H_P needs to be sufficiently large to prevent transitions from $|E_0(s)\rangle$ towards $|E_1(s)\rangle$, the evolution time τ per iteration needs to be sufficiently large to resolve $E_0\tau$ from $E_1\tau$ in the pointer register, and the number of pointer qubits r has to be large enough to provide space for the energy eigenstates that are relevant in the time evolution. Furthermore, the chosen schedule to reach the desired Hamiltonian also effects the Hamiltonians that are applied and therefore the accuracy of the algorithm.

Combining the results, we found that by varying the given parameters we could achieve a final fidelity of $F(1) = 0.696 \pm 0.004$, which implicates a success probability of $\approx 70\%$. This means, that the algorithm will provide us with the desired ground state and therefore the optimal solution of Ising type spin glass optimization problems with high probability, if the parameters are chosen adequately. Further improvements on the success probability could be achieved by increasing the number of measurements, choosing a non-polynomial schedule function, or performing a greater analysis on the mapping of the target system towards the pointer system.

A Appendix

A.1 The search problem

To provide a bit of insight to other applications of the quantum search by measurement routine, we will now briefly discuss the search problem and Grover's solution to it [5], which has been altered to be performed with the measurement scheme by Childs et al.

Besides the task of sorting lists, the task of searching in lists is one of the most fundamental algorithmic problems, for which numerous classical algorithms have been implemented. For this, it's crucial if the list is sorted regarding some key, because in this case we can speed up the process significantly through techniques like binary or exponential search.

On the other hand, if we assume an unsorted list of N items $\{v_i\}_{i=1}^{N}$ in which we want to search for a desired item w, we'd classically have to check for each of the items if they fulfill the criteria $v_i = w$ which, in the worst case of the desired item being at the last position, takes N comparisons. In the average case, when we assume that the probability for finding the item is $1/N$ at all positions, we find that $N/2$ comparisons are needed to find the item. Since the average case's runtime scales linearly with the length of the entry list, one could ask if different algorithmic implementations could speed up the process, but is has been shown that classical algorithms are restricted to that. Anyways, to actually overcome the linearity of the running time growth, we can make use of the probabilistic behaviour of qubits in quantum computers.

A.2 Gate-based Quantum Computation

There are different implementations on how to formulate problems and how to process data on quantum computers, where some implementations can solve problems more easily, since they're formulated in their native language. The approach to quantum computation that's most similar to the classical computer's approach is gate-based quantum computing, opposed to the models described in sections 1.4 and 1.4.1. This concept of the gate-based quantum analogue of classical computers has been picked up already in the twentieth century, long before its first experimental implementations, by Richard Feynman [45].

As an analogue to classical logic gates like the `AND`, the `OR`, or the `NOT` gate, the fundamental processing units in gate-based quantum computers are quantum logic gates, that are implemented via unitary transformations on the state vector. It actually has been shown, that any quantum logic gate can be decomposed into a set of fundamental gates, just like with classical logic gates [46].

Since the quantum gates represent unitary operators whose eigenvalues u_i all have magnitude 1, the gates act on its eigenstates $|\psi_i\rangle$ as rotations on the Bloch sphere:

$$U |\psi_i\rangle = e^{2\pi i \theta_i} |\psi_i\rangle$$

The operators can be represented by unitary matrices that act on the state vector. If the matrix acts on N qubits, it's size will be $2^N \times 2^N$, due to the 2^N entries in the amplitude tensor. Usually used quantum gates are single or two qubit gates, whose acting on the tensor space can be represented mathematically, as shown for a two qubit gate acting on the qubits at positions i and j:

$$U_{ij} = \mathbb{1} \otimes \mathbb{1} ... \otimes U \otimes ... \otimes U \otimes ... \otimes \mathbb{1} \tag{80}$$

These transformations act only on the two qubits and let the others unchanged. This concept of coupling 2-qubit gates to the identity operator can also be extended to n-qubit gates acting on only n qubits of a larger quantum register.

The quantum logic gate corresponding to the identity operator is the identity gate, which leaves the circuit unchanged and can be put between other unitary transformations as a trivial transformation. It's still important to note, since the action of several unitary transformations performed after each other might lead to their cancellation and thus to the application of the identity gate.

In the next sections we're going to discuss widely used quantum gates, that are necessary to understand the gate-based implementation of the algorithms discussed.

A.2.1 The Hadamard Gate

One frequently used gate, that occurs in the QFT and the QPE algorithm, is the so called Hadamard gate H. It's is a single qubit gate that acts on the basis states as followed:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \qquad H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{81}$$

The outcome of applying the Hadamard transformation equals the eigenstates of the S_x operator in the S_z basis. Therefore, the Hadamard gate and its matrix implementation map the amplitudes of the states in the S_z basis onto the amplitudes of the basis states of S_x, which can be seen for any vector by applying the Hadamard matrix on the vector expressed in its basis states. Geometrically, this can be interpreted as a rotation on the Bloch sphere. This behaviour of creating a superposition of the basis states can be exploited in several ways, of which one is the random number generator, which is one of the simplest quantum algorithms one can think of: Initialize the system in the ground state, apply a Hadamard gate, and measure the outcome. There will be a 50% chance of measuring each spin, since our wavefunction before the collaps is in a equally weighted superposition of the base states.

A.2.2 Phase shift gates

We just saw, that applying the Hadamard gate to a state initialized in the $|0\rangle$ state transforms it to the state $|+\rangle$, which can be visualized on the previously introduced Bloch sphere as a rotation around the polar angle $|theta\rangle$. It's also possible to perform rotations around the azimuthal angle ϕ on the Bloch sphere, and therefore introducing a phase shift between the two computational basis states $|0\rangle$ and $|1\rangle$. This is done by unitary transformations U_ϕ whose matrix form in the computational basis looks as follows:

$$U_\phi = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix} \tag{82}$$

As this is a transformation only acting on one qubit, it can be realized with a one-qubit gate, the so called *Phase shift gate*. It is a crucial part in manipulating quantum registers, since it will enable us to change from the aforementioned computational basis to the Fourier basis from section 2.3.3, and is therefore very important for estimating a given Hamiltonian's ground state by applying the QPE algorithm.

A.2.3 The CNOT Gate

We're now going to introduce the gate that is crucial for the phase kickback in the QPE algorithm, which is the controlled-not gate (CNOT), which uses one qubit as a control qubit, and one as a target qubit. The amplitudes of the basis states of the target state are switched, e.g. $|0\rangle \rightarrow |1\rangle$, but only if the control qubit is in the state $|1\rangle$. In matrix form and in the computational basis, the CNOT gate has the following represenation:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{83}$$

We now know how the CNOT gate acts on the eigenstates of the system, when the two qubits are in either one of the classical analogons $|0\rangle$ or $|1\rangle$. Since both of our qubits can be in superpositions of these states, we can have a look at what happens to the target qubit if the control qubit is in the so called Bell State, which can be constructed with the application of a Hadamard gate. Let the system, composed of target and control qubit, be in the product state of the S_x operator's eigenvectors $|\pm\rangle$

$$|-+\rangle = \frac{1}{2}(|00\rangle + |01\rangle - |11\rangle - |10\rangle) \tag{84}$$

We know, that the CNOT gate changes the amplitudes of the target qubit when the control qubit is in the state $|1\rangle$, which means that the overall action of CNOT can be seen as switching the amplitudes of $|01\rangle$ and $|11\rangle$. If we now apply the CNOT gate to our previously mentioned state, it will yield

$$\text{CNOT}\,|-+\rangle = \frac{1}{2}(|00\rangle - |01\rangle + |11\rangle - |10\rangle) = |--\rangle \tag{85}$$

As we can see, if we let our control and target qubits be in the S_x basis rather than the computational S_z basis, the target and the control qubit switch places: The target qubit's state has not changed, while the control qubits state was negated in the S_x basis.

This raises the question, how this is implemented technically. We can apply Hadamard gates on both qubits before applying the CNOT gate, that perform a change of basis as we saw in the previous section. After CNOT is applied, we again apply two Hadamard gates to change back to the computational basis. This can be very useful, since some setups may initially only allow CNOT operations in one direction, while wrapping it in Hadamard gates allows us to change the target and control qubit. Here, the effect of one qubit controlling the other, while they are spatially seperated is exemplary for the use of quantum entanglement in a quantum computer.

It's possible to extend the concept of controlling unitary transformation with a qubit acting as the control unit onto general one-qubit gates, which works as follows: The NOT part in the CNOT gate is getting replaced by another transformation of the target qubit. Therefore, the transformation on the target qubit is only performed, if the control qubit is in the state $|1\rangle$.

A.2.4 Limits to gate-based quantum computers

In comparison to other quantum computer models, like the quantum annealer, which was discussed in section 1.4.1, the gate-based quantum computer is heavily affected by all different kinds of noise. Since the data inscribed into the qubits has to be restored after the

quantum logic gates operated on them, we need our gates and the whole environment to not influence the state of the quantum register itself. Furthermore, with each qubit we're adding to the register, there's a new deviation introduced which influences the system and lowers the ideality of the setup. Therefore, gate-based quantum computers need to work at really low temperatures and several noise-cancelling techniques have to be implemented.

Another important point is the number of qubits the gates maximally act on: So far, only one and two qubit gates are widely used, while the fact that we limit our unitary gates to act on a maximum number of qubits will result in a restriction to our problem statement itself: Since two-qubit gates only allow the entanglement of two qubits at the same time, we would need our number of possible qubits involved per gate transformation to be the whole number of qubits in our quantum register to be able to construct all kinds of wavefunctions that inhabit entanglement of more than two qubits with each other.

A.3 Probabilistic behaviour of quantum computers

We already discussed the advantages of quantum computers in comparison to their classical counterparts, which is why now we will further investigate the reason for this computational advantage. If we have a look at a very basic classical algorithm, in comparison to a basic quantum algorithm, we can understand how fundamentally differently the two systems work:

Assume a function that returns the square of the input:

$$f(n) = n^2 \tag{86}$$

It's clear that a classical computer will have a definite output $f(n)$ for a given input n. Even if we consider e.g. the inverse of polynomial functions with several solutions per input, the possible outputs are yet determined, which is why we speak of *deterministic* behaviour. In comparison to that, we can have a look at a very easy quantum algorithm that let's us see the difference:

Let the system consist of one qubit, whose wavefunction is described as the equally weighted superposition of its computational basis vectors:

$$|\psi\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle + |1\rangle\right) \tag{87}$$

In this case the squares of the amplitudes constitute the probabilities of measuring a given eigenvalue:

$$p_\uparrow = |c_\uparrow|^2 = \frac{1}{2} \qquad p_\downarrow = |c_\downarrow|^2 = \frac{1}{2} \tag{88}$$

As we can see, the measurement of both classical bit analogues $|0\rangle$ and $|1\rangle$ have the same probability. This means that, by initializing our system in the formerly described state $|\psi\rangle$ and performing a measurement of the state afterwards, we can construct a real random one bit number generator [47], which is made possible by the *indeterministic* nature of quantum mechanics.

If we want to extend the concept of the random number generator to general binary numbers, we can just stock up the system to N qubits whose eigenstates can represent an N-bit binary number. This can later be interpreted as any data type, for example real numbers. Let the composed system's state be $|\Psi\rangle$ whose wavefunction's eigenstates can be described as

$$|n_1\, n_2 \ldots n_N\rangle = |n_1\rangle \otimes |n_2\rangle \otimes \ldots |n_N\rangle \tag{89}$$

The measurement of these tensor states will yield the random N-Bit binary number.

Another application that exhibits the indeterministic nature of quantum computers, besides the algorithms that are relevant for the Search algorithm, is the previously mentioned Shor's algorithm [8], which is used to find non-trivial factors of given whole numbers, and can reach an essential speed up compared to classical algorithms. The indeterministic nature of the algorithm manifests itself in the fact that we are given a *random* non-trivial factor, and not a determined one.

As we saw, the workflow on a quantum computer differs fundamentally to classical computers, which raises the question if there are classical computational tasks, that cannot be solved using a quantum computer. The so called *Toffoli* gate is a classical logic gate, that is reversible and can be used to implement all Boolean functions on a classical computer [48], and thus to simulate any reversible classical circuit. Since there's a distinct correspondent three qubit quantum gate to the Tofolli gate, it's possible to perform all classical computational tasks on a quantum computer by using the quantum analogue to the classical gates.

A.3.1 Reduced density matrix and entanglement entropy

Since entanglement is considered a very important computational resource in terms of quantum mechanics, we will now further investigate on the concept of entanglement and how it can be quantized.

In general, it's not trivial to see whether two subsystems are entangled or not. In the easy case of a low number of qubits, we can see if the systems are entangled and thus share joint information, we can have a look at the probabilities for measuring one subsystem A, after measuring the other subsystem B. If the probability distribution of the outcome of the measurement A changes in the case of measuring, or not measuring B beforehand, we know that the system's are entangled. This can be seen easily with the Bell state: If we measure both qubits independently, we always measure each computational basis state with 50% probability, while measuring the two qubits subsequently will force the second qubits state to be in the opposite state of the first qubit, changing the probability distribution to a certain measurement of the counterpart.

For larger systems, a very important tool to mathematically recognize entanglement is the reduced density matrix of a system. With this in hand, we can define a quantitative way of measuring the degree of entanglement of a given state. As an introduction, we will first start with the concept of density matrices in general.

To understand that concept, we first have to talk about how quantum states of many particles can be composed. We know, that a so called pure state $|\psi\rangle \in \mathcal{H}$ can always be written as a linear superposition of the basis vectors of the underlying Hilbert space, weighted with the amplitudes:

$$|\psi\rangle = \sum_{i=1}^{N} c_i \, |i\rangle \tag{90}$$

If we now go to the thermodynamic limit of a large number $N >> 1$ of quantum objects involved, we see that we can reach the same macroscopical variables, like the inner energy E of a system, with several different states $|\psi_i\rangle$ that all follow

$$H \, |\psi_i\rangle = E \, |\psi_i\rangle \tag{91}$$

The set of all the possible pure states that differ in the composition regarding the basis vectors, that all are the eigenstates of a given observable of the composite space, is called

an *Ensemble*. For systems for which we know the values of macroscopical observables, but don't have microscopical information, we can denote the state as a so called density matrix for the ensemble as

$$\rho = \sum_{i=1}^{N} p_i \left|\psi_i\right\rangle \left\langle\psi_i\right| \tag{92}$$

where the $\left|\psi_i\right\rangle \left\langle\psi_i\right|$ are the projectors onto the eigenspaces of the possible eigenvectors, and p_i denote the *classical* probability of measuring said eigenvector. It's important to note in this case, that the probabilities in the density matrices are classical probabilities and are not to be confused with the probabilities given as the squares of the amplitudes of the basis vectors $p_i = |c_i|^2$, which inhabit the quantum probability of the system.

From a density matrix's point of view, it can be quite difficult to tell whether the described state can be written as a pure state, or an ensemble of states. For this, we can decide which kind of state the density matrix is describing using the trace of its square. For pure states we have the condition

$$\mathrm{tr}(\rho^2) = 1 \tag{93}$$

To see whether a state is entangled or not, we can make use of the so called *entanglement entropy*, which was introduced by von Neumann [4]. To introduce this, we first have to introduce the concept of the reduced density matrix.

Let a system be composed of two partial systems, $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$ and $|\psi\rangle \in \mathcal{H}$ with $\rho = |\psi\rangle \langle\psi|$ the corresponding density operator. We can then define the reduced density matrix of one of the partial systems $i \in \{1, 2\}$ as follows:

$$\rho_i := \mathrm{Tr}_j(\rho) = \sum_{k=1}^{D} \langle k| \rho |k\rangle \tag{94}$$

which is called obtaining the reduced density matrix of the system i by tracing over the system j. In this case, D denotes the dimension of the dimension of the partial Hilbert space $\mathcal{H}_j$ and $|k\rangle$ denote its basis vectors. This concept is important in terms of the QSM algorithm, because performing the partial trace over the pointer system will affect the outcome of the measurement of the pointer register with respect to the target register, and therefore serves as a way to simulate the measurement procedure.

The reduced density matrix ρ_i inhabits all the information about the partial system i, including its behaviour because of the entanglement with the system j, but no information about the system j itself anymore. Therefore, to obtain information like measuring probabilities or expectation values about partial systems, we can make use of that concept.

Now we can introduce the previously announced way of a quantitative measurement of the entanglement using the reduced density matrix. The so called entanglement entropy is defined as follows:

$$\mathcal{S}(\rho_i) = -\mathrm{Tr}(\rho_i \log \rho_i) = -\mathrm{Tr}(\rho_j \log \rho_j) = \mathcal{S}(\rho_i) \equiv \mathcal{S}(\rho) \tag{95}$$

which is basically the definition of the entropy of an ensemble regarding statistical quantum physics, and gives us information about the possible number of pure states the system can be in to reach the previously defined fixed macroscopical observables.

If the entanglement entropy of a subsystem is non-zero, the subsystem is in a mixed state, while if it is zero, the subsystems are pure states. If the subsystems themselves, described by their reduced density matrices, are in mixed states, the two partial systems have to be entangled.

As we see, with this definition it doesn't make a difference whether we have a look at the reduced density matrix of one subsystem or the other, or over which subsystem we perform the trace. That's important, because otherwise there wouldn't be a distinct way for defining the composite system's degree of entanglement.

A.4 Grover's Algorithm

To provide some insight on the original problem that the QSM algorithm has been applied to, we're now going to briefly explain Grover's search problem and the corresponding algorithm. In comparison to a linear running time $f(n) \in \mathcal{O}(n)$ in the case of classical algorithms, it has been shown that by implementing a quantum algorithm, the so called Grover's algorithm, on a quantum computer, we can reach quadratic speed up, leading to a running time in $\mathcal{O}(\sqrt{N})$, which becomes significant when it comes to big data having to be processed.

First, we have to know how a search problem can be understood in terms of quantum information. Our search room S will be composed of the set of possible eigenvalues of the underlying qubit register, which means that its size is equal to the dimension of the Hilbert space $H = H_1 \otimes H_2 \otimes ... \otimes H_N$ underlying the system, which is 2^N in the case of N qubits involved:

$$|S| = d(H) = \prod_{i=1}^{N} d(H_i) = 2^N \tag{96}$$

This means, that 2^N eigenvectors of $S_z^{\otimes n}$ exist, which all represent an object in the search room. Let $v_i{}_{i=1}^{2^N}$ be the possible eigenvectors, and $w_i \in \{v_i\}_{i=1}^{2^N}$ the desired solution to the search problem. The quantum implementation of the algorithm works as follows: We start with an equally weighted superposition of all basis vectors $|v_i\rangle$, and then manipulate the many-qubit state and the probabilities of measuring the given basis states with unitary transformations, until it becomes very likely to measure the eigenvalue of the desired vector $|w\rangle$, which corresponds to the index of the element in the search space in the binary system.

We have to note, that the problem itself consists of two subproblems: The first one is identifying the right solution $|w\rangle$, which is done with a so called *Oracle* U_w, and the second subproblem is getting the quantum computer to actually measure the corresponding eigenvalue. The construction of oracles can be very hard, which is why we assume that we already know which vector $|w\rangle$ we want to measure, and focus on how to bring our quantum computer to yield the correct index as we measure. The oracle in this case can be understood as some kind of black box, that already knows which index is the right one and multiplies the correct state's amplitude with (-1) and leaves the others' unchanged:

$$U_w |w\rangle = -|w\rangle \qquad U_w |v_i\rangle = v_i \quad \forall v_i \neq w \tag{97}$$

We will further use the oracle U_w to tell us which index is the right one, and no longer focus on how to construct it.

To begin with the algorithm, we first of all have to construct the superposition of our possible eigenvectors that represent the search space. Let this superposition be $|s\rangle$:

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} |v_i\rangle \tag{98}$$

where $|v_i\rangle$ represent the eigenvectors of the full system. We have to note, that N in this case is the number of possible eigenvectors and not the number of qubits involved.

To construct the superposition, we can simply apply the tensor product of n Hadamard gates $H^{\otimes n}$ onto a system, that's initialized in the ground state $|0\rangle = |0\rangle^{\otimes n}$. With this superposition, the probability of measuring the right vector w_i is just the same as for any other vector in the basis:

$$p(w_i) = p(v_i) = \frac{1}{N} \qquad \forall i \in \{0, ..., N\} \tag{99}$$

Our goal now is to improve the probability of measuring the desired eigenvalue, and therefore making the algorithm give us the index of the right element with high probability.

This improvement of probability is done by successively performing certain unitary gates, which brings us to the idea of amplitude amplification.

A.4.1 Amplitude amplification

The key process in the implementation of Grover's search algorithm lies in improving the amplitude of the *winner's* state vector in the superposition, making it more likely to measure the corresponding eigenvalue. This is done via a subroutine called *Amplitude amplification*.

We start with the aforementioned superposition of all possible basis states:

$$|s\rangle = H^{\otimes n} |0\rangle \tag{100}$$

We then deconstruct the superposition $|s\rangle$ into the desired solution $|w\rangle$, and a vector $|s_\perp\rangle$ that's perpendicular to the solution vector and can be obtained by removing $|w\rangle$ from the superposition vector $|s\rangle$. These two vectors span a two dimensional plane, in which the initially constructed superposition $|s\rangle$ lies. This means, that our vector $|s\rangle$ can be written as follows:

$$|s\rangle = \alpha |w\rangle + \beta |s_\perp\rangle \tag{101}$$

where the coefficients α and β inhabit the probability of measuring the correct or the incorrect solution respectively. We therefore need to manipulate our vector, to improve the value of α. This is done in two steps, that are repeated successively. Let θ be the angle between the current vector $|s\rangle$ and the axis representing the wrong solutions $|s_\perp\rangle$. We can then write our current vector $|s\rangle$ as

$$|s\rangle = \cos\theta |s_\perp\rangle + \sin\theta |w\rangle \tag{102}$$

Our goal can then be reinterpreted as our vector being rotated onto the axis characterized by $|w\rangle$, which leads to the following conditions for measuring the correct solution with 100% probability:

$$||\alpha||^2 = 1 \Leftrightarrow \sin\theta = 1 \Leftrightarrow \theta = \frac{\pi}{2} \tag{103}$$

As the system is initialized in an equally weighted superposition, the initial angle θ is characterized only by the number of basis vectors spanning the Hilbert space:

$$\theta = \arcsin\frac{1}{\sqrt{N}} \tag{104}$$

Both substeps of improving the solution's amplitude can be interpreted as reflections of $|s\rangle$ in the aforementioned plane around certain vectors, that result in an overall improvement of θ.

The first step is applying a reflection of $|s\rangle$ around $|s_\perp\rangle$, which leads to the amplitude of $|w\rangle$ switching its sign. This is exactly what's done by the aforementioned oracle U_w, which can be written as the following unitary transformation

$$U_w = |s\rangle \langle s| - 1 \tag{105}$$

One could ask how that improves the amplitude of our winner state, and the answer is that it actually does only when combined with a second unitary transformation U_s. Since we now have an angle of 2θ between our state $U_w |s\rangle$ and the initial state $|s\rangle$, we can reach an angle $\theta' > \theta$ by performing another reflection around $|s\rangle$. For this, we have to add a negative phase to all vectors orthogonal to $|s\rangle$, which can be done using quantum logic gates we discussed previously: First, the superposition $|s\rangle$ has to be transformed into the ground state $|0\rangle$, using Hadamard gates again:

$$|0\rangle = H^{\otimes n} |s\rangle \tag{106}$$

Now we use the following unitary transformation U_0, that adds a negative phase to all vectors orthogonal to $|0\rangle$:

$$U_0 |s\rangle = \frac{1}{\sqrt{N}} \left(|0\rangle - \sum_{i=1}^{N-1} |i\rangle \right) \tag{107}$$

Since we previously transformed our vector $|s\rangle$ into the ground state $|0\rangle$, now every vector orthogonal to $|s\rangle$ has a negative phase. We can transform the state back with applying the Hadamard gates again, which leaves us with all amplitudes of vectors orthogonal to $|s\rangle$ having a sign switch. The overall rotation U_s can then be denoted as

$$U_s = H^{\otimes n} U_0 H^{\otimes n} \tag{108}$$

This now lead to an overall rotation towards the desired axis characterized by $|w\rangle$. Applying these two transformations t times successively, will lead to a resulting state

$$(U_w U_s)^t |s\rangle = \sin \theta_t; \qquad \theta_t = (2t+1)\theta \tag{109}$$

Since the algorithm yields a 100% success probability for $\theta_t = \pi/2$, the number of iterative applications t for the algorithm to succeed only depends on the initial angle θ, and therefore on the number of items in the search space. If we take the relation from equation 104 and insert it into the success condition we get an estimate for the number of iterations needed:

$$\theta_t = (2t+1)\theta = (2t+1) \arcsin \frac{1}{\sqrt{N}} \overset{!}{=} \frac{\pi}{2} \tag{110}$$

$$\Leftrightarrow t \overset{!}{=} \frac{\pi}{4 \arcsin \frac{1}{\sqrt{N}}} - \frac{1}{2} \tag{111}$$

which, for the example case of $n = 5 \Rightarrow N = 36$ yields $t = 3.92$. This lets us see, that not every initial number of states N results in t being an integer. If there is no integer that fulfills the condition 110, we will gain the biggest success probability if our angle θ_t is closest to $\pi/2$, which is we have to round t to the closest integer.

Hence, it's important to note, that in the case of t being not an integer, we cannot exactly reach the final angle $\theta_t = \pi/2$ and thus there will be a probability $p > 0$ of measuring the wrong item.

A.5 Code for the simulation

The code for the simulation can be found on github here.

References

[1] Han-Sen Zhong et al. "Quantum computational advantage using photons". In: *Science* 370.6523 (2020), pp. 1460–1463. DOI: 10.1126/science.abe8770. eprint: https://www.science.org/doi/pdf/10.1126/science.abe8770. URL: https://www.science.org/doi/abs/10.1126/science.abe8770.

[2] Ashley Montanaro. "Quantum algorithms: an overview". In: *npj Quantum Information* 2.1 (Jan. 2016). DOI: 10.1038/npjqi.2015.23. URL: https://doi.org/10.1038%2Fnpjqi.2015.23.

[3] Andrew M. Childs et al. "Quantum search by measurement". In: *Physical Review A* 66.3 (Sept. 2002). DOI: 10.1103/physreva.66.032314. URL: https://doi.org/10.1103%2Fphysreva.66.032314.

[4] John von Neumann. *Mathematische Grundlagen der Quantenmechanik*. 1996. DOI: 10.1126/science.abe8770. URL: https://link.springer.com/book/10.1007/978-3-642-61409-5.

[5] Lov K. Grover. *A fast quantum mechanical algorithm for database search*. 1996. DOI: 10.48550/ARXIV.QUANT-PH/9605043. URL: https://arxiv.org/abs/quant-ph/9605043.

[6] Masahito Hayashi. *Quantum Information*. Springer Berlin, 2018. DOI: 10.1007/3-540-30266-2. URL: https://link.springer.com/book/10.1007/3-540-30266-2.

[7] Edmund Landau. "Handbuch der Lehre von der Verteilung der Primzahlen". In: (1909). URL: https://archive.org/details/handbuchderlehre01landuoft/page/59/mode/1up.

[8] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1484–1509. DOI: 10.1137/s0097539795293172. URL: https://doi.org/10.1137%2Fs0097539795293172.

[9] Yulin Wu et al. "Strong Quantum Computational Advantage Using a Superconducting Quantum Processor". In: *Physical Review Letters* 127.18 (Oct. 2021). DOI: 10.1103/physrevlett.127.180501. URL: https://doi.org/10.1103%2Fphysrevlett.127.180501.

[10] F. et al. Arute. "Quantum supremacy using a programmable superconducting processor". In: *Nature* 574 (Oct. 2019), pp. 505–510. DOI: 10.1038/s41586-019-1666-5.

[11] David P. DiVincenzo. "The Physical Implementation of Quantum Computation". In: *Fortschritte der Physik* 48.9-11 (Sept. 2000), pp. 771–783. DOI: 10.1002/1521-3978(200009)48:9/11<771::aid-prop771>3.0.co;2-e. URL: https://doi.org/10.1002%2F1521-3978%28200009%2948%3A9%2F11%3C771%3A%3Aaid-prop771%3E3.0.co%3B2-e.

[12] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511976667.

[13] *IBM Quantum*. 2021.

[14] Richard W. Eglese Abdelkader Sbihi. *Combinatorial optimization and Green Logistics*. 2007. DOI: 10.1007/s10288-007-0047-3. URL: https://arxiv.org/abs/quant-ph/9605043.

[15] Valentina Cacchiani. "Knapsack problems — An overview of recent advances. Part I: Single knapsack problems". In: *Computers Operations Research* 143 (2022). DOI: 10.1016/j.cor.2021.105692. URL: https://www.sciencedirect.com/science/article/pii/S0305054821003877.

[16] Samuel Mugel, Enrique Lizaso, and Roman Orus. *Use Cases of Quantum Optimization for Finance*. 2020. DOI: 10.48550/ARXIV.2010.01312. URL: https://arxiv.org/abs/2010.01312.

[17] Engineering National Academies of Sciences and Medicine. "Quantum computing: Progress and Prospects". In: (2019). DOI: 10.17226/25196.

[18] Panagiotis Kl. Barkoutsos. *Quantum algorithm for alchemical optimization in material design*. 2021. DOI: 10.1039/D0SC05718E. URL: https://arxiv.org/abs/2010.01312.

[19] Marais Adriana. et al. *The future of quantum biology*. 2018. DOI: 10.1098/rsif.2018.0640. URL: https://arxiv.org/abs/2010.01312.

[20] Edward Farhi et al. "A Quantum Adiabatic Evolution Algorithm Applied to Random Instances of an NP-Complete Problem". In: *Science* 292.5516 (2001), pp. 472–475. DOI: 10.1126/science.1057726. eprint: https://www.science.org/doi/pdf/10.1126/science.1057726. URL: https://www.science.org/doi/abs/10.1126/science.1057726.

[21] M. Born and V. Fock. "Beweis des Adiabatensatzes". In: *Zeitschrift fur Physik* 51.3-4 (Mar. 1928), pp. 165–180. DOI: 10.1007/BF01343193.

[22] Clarence Zener. "Non-Adiabatic Crossing of Energy Levels". In: *Proceedings of the Royal Society of London Series A* 137.833 (Sept. 1932), pp. 696–702. DOI: 10.1098/rspa.1932.0165.

[23] Shunji Matsuura et al. "Variationally scheduled quantum simulation". In: *Phys. Rev. A* 103 (5 May 2021), p. 052435. DOI: 10.1103/PhysRevA.103.052435. URL: https://link.aps.org/doi/10.1103/PhysRevA.103.052435.

[24] Edward Farhi et al. *Quantum Computation by Adiabatic Evolution*. 2000. DOI: 10.48550/ARXIV.QUANT-PH/0001106. URL: https://arxiv.org/abs/quant-ph/0001106.

[25] Tadashi Kadowaki and Hidetoshi Nishimori. "Quantum annealing in the transverse Ising model". In: *Physical Review E* 58.5 (Nov. 1998), pp. 5355–5363. DOI: 10.1103/physreve.58.5355. URL: https://doi.org/10.1103%2Fphysreve.58.5355.

[26] Asim et al. Ghosh. "Quantum Annealing and Computation: A Brief Documentary Note". In: (2013). DOI: 10.48550/ARXIV.1310.1339. URL: https://arxiv.org/abs/1310.1339.

[27] S. et al. Jiang. *Quantum Annealing for Prime Factorization*. 2018. DOI: 10.1038/s41598-018-36058-z.

[28] K. Binder and K. Schröder. "Phase transitions of a nearest-neighbor Ising-model spin glass". In: *Phys. Rev. B* 14 (5 Sept. 1976), pp. 2142–2152. DOI: 10.1103/PhysRevB.14.2142. URL: https://link.aps.org/doi/10.1103/PhysRevB.14.2142.

[29] Herbert S. Green and C. A. Hurst. *Order-Disorder Phenomena*. 1964.

[30] Andrew Lucas. "Ising formulations of many NP problems". In: *Frontiers in Physics* 2 (2014). DOI: 10.3389/fphy.2014.00005. URL: https://doi.org/10.3389%2Ffphy.2014.00005.

[31] Jacob D. Biamonte and Peter J. Love. "Realizable Hamiltonians for universal adiabatic quantum computers". In: *Physical Review A* 78.1 (July 2008). DOI: 10.1103/physreva.78.012352. URL: https://doi.org/10.1103%2Fphysreva.78.012352.

[32] K. H. Fischer and J. A. Hertz. *Spin Glasses*. Cambridge Studies in Magnetism. Cambridge University Press, 1991. DOI: 10.1017/CBO9780511628771.

[33] V. Bapst et al. "The quantum adiabatic algorithm applied to random optimization problems: The quantum spin glass perspective". In: *Physics Reports* 523.3 (Feb. 2013), pp. 127–205. DOI: 10.1016/j.physrep.2012.10.002. URL: https://doi.org/10.1016%2Fj.physrep.2012.10.002.

[34] Sho Kanamaru et al. "Efficient Ising Model Mapping to Solving Slot Placement Problem". In: *2019 IEEE International Conference on Consumer Electronics (ICCE)*. 2019, pp. 1–6. DOI: 10.1109/ICCE.2019.8661947.

[35] P. et al. Jordan. "Über das Paulische Äquivalenzverbot". In: *Zeitschrift für Physik* 47 (1928). DOI: 10.1007/BF01331938. URL: https://link.springer.com/article/10.1007/BF01331938.

[36] E. Ovrum and M. Hjorth-Jensen. *Quantum computation algorithm for many-body studies*. 2007. DOI: 10.48550/ARXIV.0705.1928. URL: https://arxiv.org/abs/0705.1928.

[37] Richard Josza. *Fidelity for Mixed Quantum States*. 1994. DOI: 10.1080/09500349414552171.

[38] W. M. Itano. *Perspectives on the quantum Zeno paradox*. 2006. DOI: 10.48550/ARXIV.QUANT-PH/0612187. URL: https://arxiv.org/abs/quant-ph/0612187.

[39] Gerardo A. Paz-Silva et al. "Zeno Effect for Quantum Computation and Control". In: *Physical Review Letters* 108.8 (Feb. 2012). DOI: 10.1103/physrevlett.108.080501. URL: https://doi.org/10.1103%2Fphysrevlett.108.080501.

[40] Anis Sajid et al. *Qiskit: An Open-source Framework for Quantum Computing*. 2021. DOI: 10.5281/zenodo.2573505.

[41] F. Bloch. "Nuclear Induction". In: *Phys. Rev.* 70 (7-8 Oct. 1946), pp. 460–474. DOI: 10.1103/PhysRev.70.460. URL: https://link.aps.org/doi/10.1103/PhysRev.70.460.

[42] S. S. Zhou et al. "Quantum Fourier transform in computational basis". In: *Quantum Information Processing* 16.3 (Feb. 2017). DOI: 10.1007/s11128-017-1515-0. URL: https://doi.org/10.1007%2Fs11128-017-1515-0.

[43] M. et al Suzuki. *Finding Exponential Product Formulas of Higher Orders*. 2005. DOI: 10.1007/11526216_2.

[44] K. Temme et al. "Quantum Metropolis sampling". In: *Nature* 471.7336 (Mar. 2011), pp. 87–90. DOI: 10.1038/nature09770. URL: https://doi.org/10.1038%2Fnature09770.

[45] R.P. Feynman. "Quantum mechanical computers". In: *Found Phys* 16 (1986), pp. 507–531. DOI: 10.1007/BF01886518. URL: https://link.springer.com/article/10.1007/BF01886518.

[46] Adriano Barenco et al. "Elementary gates for quantum computation". In: *Phys. Rev. A* 52 (5 Nov. 1995), pp. 3457–3467. DOI: 10.1103/PhysRevA.52.3457. URL: https://link.aps.org/doi/10.1103/PhysRevA.52.3457.

[47] Z. et al. Zhang. *Quantum random number generation*. 2016. DOI: 10.1038/npjqi.2016.2.

[48] Tommaso Toffoli. *Reversible Computing*. 1980. DOI: 10.1007/3-540-10003-2_104.

[49] Chase Cook et al. *GPU Based Parallel Ising Computing for Combinatorial Optimization Problems in VLSI Physical Design*. 2018. DOI: 10.48550/ARXIV.1807.10750. URL: https://arxiv.org/abs/1807.10750.

[50] Asher Peres. "Reversible logic and quantum computers". In: *Phys. Rev. A* 32 (6 Dec. 1985), pp. 3266–3276. DOI: 10.1103/PhysRevA.32.3266. URL: https://link.aps.org/doi/10.1103/PhysRevA.32.3266.