Advanced Quantum Algorithms

Lecture notes

Guglielmo Mazzola

Fall semester 2023

University of Zurich[UZH]

Institute for Computational Science
University of Zurich
Switzerland

Abstract

"Advanced quantum algorithms" is a course that aims to provide a resonably complete overview of the state of the art in the algorithmic side of quantum computing. These lecture notes will be updated regularly to keep students informed about the rapidly evolving field of quantum computing. The focus of the course is on practical applications of quantum computing, including the benefits and limitations of this technology. The notes will cover the most promising quantum algorithms that are expected to achieve quantum advantage in the near future, as well as the challenges and limitations of quantum computing. Overall, the goal of the course is to provide a balanced review of quantum computing, highlighting both its potential and its limitations.

A short bibliography is placed at the end of each chapter. While a short and self-contained description of basic quantum gates and circuit will be provided, this should *not* be understood as standard quantum information course. We are not going to cover topics such as foundations of quantum mechanics, cryptography, teleportation, or paradoxes. *Gedanken experiments* conducted by Alice and Bob are also outside the scope of the course.

The course is designed to be accessible to students with limited knowledge of quantum information science and diverse backgrounds, such as theoretical physics, condensed matter, theoretical chemistry, or quantum engineering.

Contents

Chapter 1

Introduction

As quantum technology advances, it is becoming increasingly possible to develop quantum devices, such as quantum communication systems, quantum random number generators, quantum simulators and computers, which may have capabilities that surpass those of classical capabilities. The field of quantum computing has garnered significant attention due to its potential to solve certain computational problems much more quickly than classical computers can. These problems include factoring integers and simulating quantum systems. Shor's algorithm, for example, can find the prime factors of an integer in a time that grows in a polynomial fashion with the number of digits in the integer, while classical algorithms for this task require exponential time. Similarly, simulating the time evolution of a quantum system on a classical computer requires exponential resources, due to the exponentially large size of the system's Hilbert space, while quantum hardware can perform the same simulation with polynomial complexity(1). Moreover, complexity theory arguments often applies for *exact* solutions. In real-world, approximate -good- solutions are very valuable (let't think about optimization, quantum chemistry, machine learning..). This means that it is not obvious to identify a set of problems and quantum algorithms able to showcase quantum advantage in practice.[1]

The quest for quantum speed-up is on. As of today, *it is still unclear* what will be the first concrete problem, or practical relevance, that will enjoy quantum speed-up. The answer to this question will depend on several factors: the **specific use-case** application, the **algorithm** used, and the **hardware**.

The purpose of this chapter is to outline the goals of a general quantum algorithm, the current capabilities of available ad future devices. It is also important to understand the limitations that these architectures will impose on a quantum algorithm. Finally, we will discuss the abstractions and layers of a generic quantum software.

Disclaimer: Some concepts will appear undefined or loosely defined for an un-initiated reader. While these definitions will be provided later, the logical implications outlined in the Chapter can still be accessible. I prefer to give first an high-level overview before zooming in to definitions.

[1]For instance, in the context of optimization is not clear if, and how, practical quantum advantage can be reached. Also in quantum chemistry, a field which has been considered for a long time to be the most promising one, people are critically reassessing overly optimistic claims of exponential quantum advantage.

1.1 Scalings and prefactors

In certain cases, like the ones listed above, quantum algorithms can outperform their classical counterparts by an exponential margin (as the problem size increases). This type of exponential speedup makes it easier to compare the efficiency of different algorithms. According to the extended Church-Turing thesis, all classical computers are equivalent in terms of their computational capabilities, differing only by polynomial factors. Similarly, all proposed models of quantum computation are believed to be equivalent, meaning that an exponential quantum speedup would be independent of the specific model being used. There are quantum algorithms that can provide instead "only" a polynomial speedup. Most of the time, this polynomial speed-up is a quadratic one (like in the Grover algorithm), thus introducing complications in assessing the potential advantage of quantum algorithms as we will see.

Many of the most well-known quantum algorithms, such as Shor's and Grover's algorithms, were developed (around the 90's) before the actual quantum devices have been built. For a long time, the field of quantum computation had a strong connection to pure quantum information science and was driven more by mathematicians than by physicists. The primary focus of research was on achieving scaling advantage, or the ability for quantum algorithms to perform better than classical algorithms as the size L of the problem increases. However, it is now understood that **scaling advantage alone is not sufficient for achieving practical advantage**. For instance, an algorithm that scales exponentially but has a weak exponent (i.e. $e^{0.01\ L}$) may still be faster than a polynomial algorithm with a large power and prefactor (i.e. L^{10}) for problem sizes below a certain threshold (in this case $L \approx 10^4$). In this case, the exponentially scaling solution would be the better choice for practical purposes.

When considering the performance of quantum computers, with a practical mindset, we cannot neglect the hardware. Currently, there are two main types of quantum computers: analog and digital. While we will discuss the distinction between these types later, it is important for the time being just to note that all quantum computers must be controlled and their states must be changed in order to perform a computation. It turns out that the speed at which we can **control a quantum system** depends on the specific quantum architecture being used, and is generally much slower than the clock rate of classical CPUs (this will be discussed below).

This means that quantum computers will eventually be more powerful not because they are intrinsically faster (at the gate level), but because they can process vast amounts of information in a different way than classical computers can. This is why people are interested in scaling assessment, or the ability of quantum algorithms to perform better as the size of the problem increases. However, the fact that quantum computers are "slow", in the sense defined above, means that there is a **large prefactor** that can overshadow the scaling advantage until a certain crossover problem size L_C is reached. In general, quantum computers are likely to be slower than classical computers for small problem sizes, and the quantum speedup will only occur for large sizes. The key challenge is to identify a class of problems for which the threshold size L_C is still meaningful and the total runtime t_C is still practical.

While quite obvious, this concept has been internalized by the community only recently, especially thanks to end-to-end resource assessment like this (2) for quantum chemistry, or the debate around the insufficiency of a quadratic speed-up to achieve practical advantage in

the fault tolerant regime(3)

1.2 Hardwares overview

There are numerous ways to physically implement a qubit. For example, it can be represented by a single atom, electron, or photon. Alternatively, a qubit can also be represented by a more complex system, such as a superconducting electrical circuit cooled to a very low temperature, in which many electrons are involved. Matematically a qubit is just a two-level system, but not all two-level systems in Nature can be good qubits.

A working qubit needs to have the following:

- It must behave as a two-level systems, isolated from external influences as much as possible.
- It must interact strongly with other qubits in perform computational tasks.
- It can be controlled and readable from outside

It is clear that an electron in a molecule does not have all these properties, despite being a spin 1/2 quantum particle.

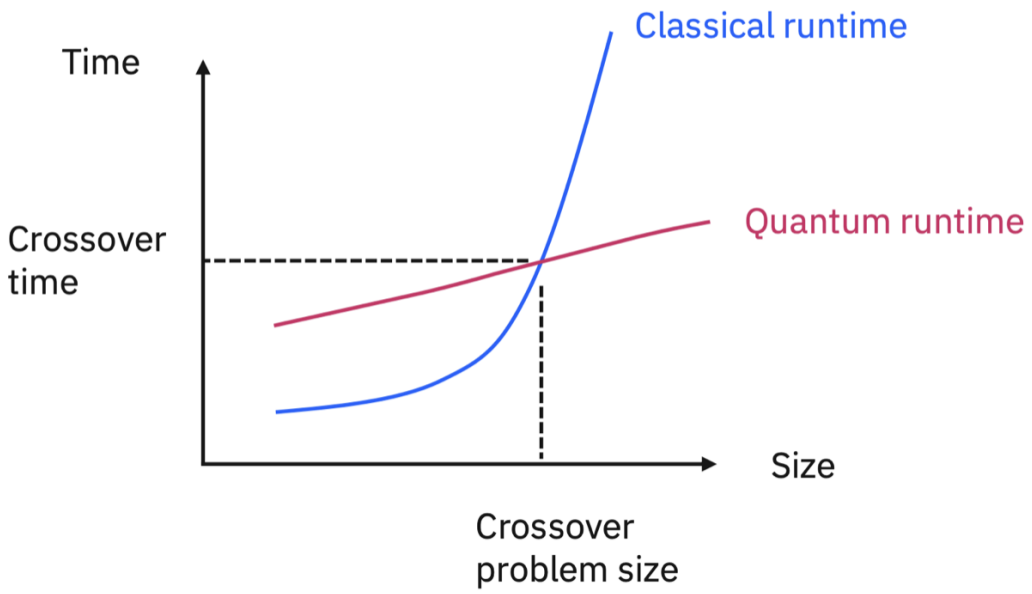


Figure 1.1: Cartoon of quantum and classical runtimes for a generic situation where a quantum algorithm has scaling advantage over its classical counterpart (e.g. it may be a the task of time-evolving a quantum state, under the action of a lattice hamiltonian, like Hubbard one.) For small system size, the scaling advantage is obfuscated by the large prefactor. Only at sufficiently large size, we can achieve practical advantage.

.

It is not the purpose of this lecture note to provide an exhaustive account of quantum hardware. For our purpose, it can be enough to separate quantum hardwares in classes: analog and digital ones. For the purpose of this lecture note, we can think about superconducting qubits, which are made of solid state electrical circuits fabricated using techniques similar to conventional integrated circuits. When cooled down to mK temperatures they develop superconductivity, which is a *macroscopic* quantum phenomena. A superconducting qubit, satisfies those constraints as it is a macroscopic quantum system that can be well approximated by a two level quantum system, it can be controllable by classical electronics, and can be manifactured to realize bigger systems. Being a macroscopic system, it features more interactions with the environment, i.e. noise, this is the price to pay to have a controllable and scalable qubits. For an introductory review of supercondicting qubits, which is the main technology in use today, I refer the reader to (4).

1.2.1 Analog devices

A quantum simulator is a **special purpose quantum device** that allows researchers to study a quantum system in a controlled, programmable manner. These simulators are specifically designed to provide insight into specific physics problems, like finding phase diagrams of lattice models, or study their out-of-equilibrium dynamics. Concretely this means to find a time evolved state, under the action of an Hamiltonian operator H,

$$|\psi(t)\rangle = e^{-iHt}|\psi_0\rangle \tag{1.1}$$

where $|\psi_0\rangle$ is not an eigenstate of H, or the hamiltonian itself is time dependent, $H \to H(t)$.

Quantum simulators can be defined in constrast to general purpose, “digital" quantum computers, which have the ability to solve a broader range of task. Quantum simulators can be build using trapped ions, ultracold atoms, and superconducting qubits. These quantum simulators are made of many-body quantum systems whose hamiltonians can be engineered to approximate the one we are interested in, e.g. quantum simulators for the Hubbard or transverse field Ising model.

We will set aside the philosophical question of what constitutes a *simulation versus a computation*. A practical, though not entirely satisfactory, answer may be that a quantum computer has the ability to perform a wide range of tasks, such as simulating a quantum system or calculating the fair value of a financial derivative, while a quantum simulator is designed with a specific physical model in mind to simulate. For instance, in the case of ultracold atoms simulators, if one aims to “solve” a fermionic model hamiltonian, fermionic atoms will be loaded in the optical potential, otherwise bosonic atoms will be required. This is in contrast with standard “classical” simulations, e.g with quantum Monte Carlo algorithms, where we do not have to change our laptop or HPC cluster when we change the model under study.

However, there are two reasons why the previous answer may not be entirely correct. For instance, consider a quantum simulator designed to implement the transverse field Ising Hamiltonian using superconducting qubits,

$$H = -\sum_{i,j=1}^{L} J_{i,j}\sigma_i^z\sigma_j^z - \Gamma\sum_{i=1}^{L}\sigma_i^x \tag{1.2}$$

If used in its "simulator" mode, it could simulate the real-time dynamics of a quenched Hamiltonian, which would be intractable for classical methods to solve for large enough times. However, if the programmable, time-dependent Hamiltonian has the shape of a linear ramp (i.e. $\Gamma \to \Gamma(t)$), it can be used to perform a **quantum annealing** simulation (see Chapter 3). This process can be used in optimization, and if programmable couplings between the qubits can also be engineered, a combinatorial optimization problem can be embedded into an Ising spin Hamiltonian. Finding the ground state of this Hamiltonian through quantum annealing would solve the optimization problem, performing a computation. [2] This is exactly what D-Wave quantum machines have been attempting to do since 2011(5).

The second reason is that the progress in controlling quantum particles in quantum simulators is enabling the **realization of gates**. After all, hamiltonian dynamics is a unitary operation, such as the one implied by quantum gates. For instance, trapped ions are systems that can be used for quantum simulations of spin models but can be used to construct, more generally, quantum computers. A qubit in this case is the space represented by the ground state electronic level and an excited level of a single ion. At present (2023), it is possible to use around 50 ions in linear chains. By applying laser excitations and utilizing the collective motion of the array, it is possible to generate effective interactions, which can be used to create either spin models interaction with custom long-range interactions, or quantum gates. For instance, rotation gates (see Chapter 2) can be realized by manipulating the frequency and total time of an external electromagnetic field that is locally applied to an ion. This drives a transition from the ground-state to the excited state. The same holds for superconducting qubits.

One major drawback of analog simulators is that **errors cannot be reliably detected and corrected**. For example, if the pulse used to generate a rotation gate has a slightly different frequency than intended, the rotation gate will be imperfect and it is not possible to correct infinitesimal errors that can accumulate and lead to significant effects, known as calibration errors. Another type of error is decoherence, which occurs when the quantum system interacts with the environment and limits its coherence time. These errors can significantly impact the final results of the calculation in an unpredictable and uncorrectable manner. This is why there is also interest in a digital model of quantum computation. From an historical point of view, the first quantum analog simulation of a strongly correlated quantum model dates back to 2002(6). After more than twenty years, we can safely observe that they didn't replace classical simulations.

To summarize, specifying a given architecture, e.g superconducting qubits or ions, is not per se enough to distinguish between a quantum simulator or a computer. What matters is the use of the machine, and the quality of the hardware that can enable the realization of quantum gates using precise control.

1.2.2 Digital devices

Digital quantum computers are the opposite end of the spectrum compared to analog machines. They aim to implement the quantum circuit model for quantum computation, which

[2] Moreover, adiabatic quantum computing has been shown to be equivalent to gate based computing. However, this would require machines which are not present today.

is similar to classical circuits in that it involves a sequence of quantum gates, measurements, and initialization of qubits to known values. **Quantum logic gates** are reversible, unitary transformations that act on one or more qubits. In the circuit model, quantum algorithms are constructed using these elementary blocks, regardless of the specific hardware type.

Coming to our quantum simulation task example, digital quantum simulators have the advantage of being able to realize any desired Hamiltonian of the system, which allows for the study of a wide range of models without the need for direct engineering in the laboratory. In addition, quantum logic gates can be used to create arbitrary quantum circuits with a wide variety of applications.

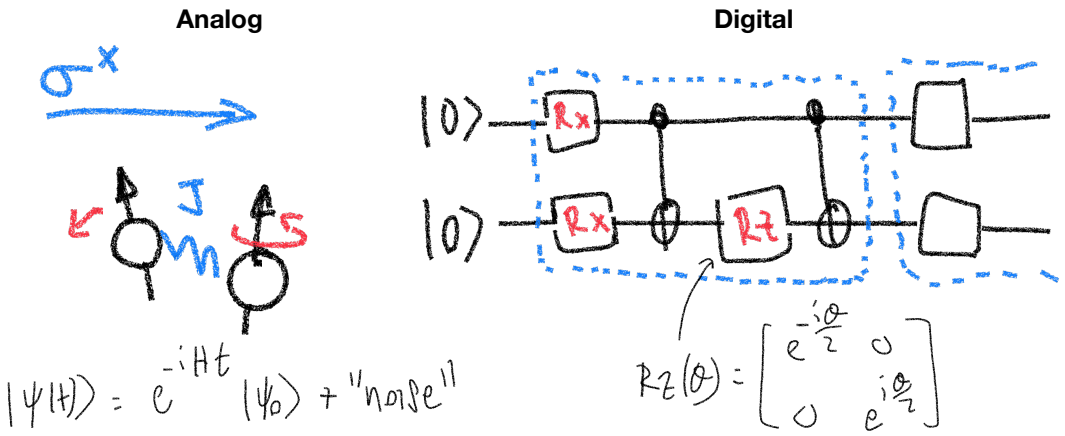


Figure 1.2: The two ways to perform quantum simulations of a two-site transverse field Ising model. In the analog regime, one engineer couplings between the qubits to represent the hamiltonian under study H. In the digital regime, one can only play with building blocks: digital quantum gates. For the Ising model case, one needs only parametrized R_x and R_z rotation gates, and CNOT gates (details will be provided in Chapter **??**). Quantum simulations need to be performed using an algorithm: the Trotter algorithm. Analog simulations are philosophically closer to an experiment.
.

On the other hand, performing a quantum simulation with analog devices is much like performing an experiments, asking a programmable quantum device to simulate itself and measure the state of the device after a certain amount of physical time. The digital setting works at a higher level of abstraction: the real-time quantum dynamics must be performed using algorithms, similar to classical computing. The Hamiltonian evolution operator is subjected to Trotterization, and each of the non-commuting unitary operators is compiled into logical gates (more in Chapter **??**). In the fault-tolerant regime, there are no errors other than the finite-time Trotterization scheme, which can be systematically controlled (cfn Figure 1.2).

Most of the algorithms presented in these notes are devised within the circuit model of quantum computation. A self-contained description of quantum register, circuits, and quantum gates will be provided in Chapter 2. For the time being, we continue with our

high level description of the capabilities of digital devices, however, if these concepts are completely unfamiliar, readers who prefer having definitions first can jump to Chapter 2 or read the introductory Chapters of books such as (1), and come back.

They key advantage of digital quantum machines is that they can be error corrected. The quantum threshold theorem (or quantum fault-tolerance theorem) states that a quantum computer with a physical error rate below a certain threshold can, through the use of quantum error correction schemes, reduce the logical error rate to arbitrarily low levels (cfn. Ref (1)).

Fault-tolerant digital computation then introduces the concept of a **logical qubit**, as opposed to the **physical qubits** that are the units presents in the hardware. A logical qubit is an ideal qubit that is not affected by noise and decoherence, and gates that are perfectly implementing the unitary matrices they are intended to. It is currently believed that many physical qubits are necessary for error-correction in order to create an entity that behaves like a single qubit. To achieve this, multiple physical qubits are used along with algorithms that are able to detect and correct errors, creating a logical qubit. Alternatively, it is possible to directly construct a logical qubit using a topologically protected physical qubit, a technology currently pursued by Microsoft.

Here we briefly consider the standard approach to fault-tolerance using error correction. Error correction algorithm include topological stabilizer codes like the surface code(7) and subsystem codes like the Bacon-Shor code. The surface code, with its high error tolerance and simple, two-dimensional physical layout with only nearest-neighbor coupling, is one of the most realistic approaches to building a solid-state quantum computer. However, it comes at a cost, as implementing the surface code requires a **large number of physical qubits**. In fact, it takes a minimum of 13 physical qubits to create a single logical qubit. Interestingly, a first instance of error correction using 17 qubits has recently been realized at ETH(8), meaning that the possibility of building fault tolerant machines in the next future is realistic.

At this point, we are not yet able to explain the surface code without formally introducing the concept of quantum gates. For now, it is sufficient to know that there is a clear strategy for building fault-tolerant quantum computers, giving hope that the circuits we will see in the next chapter could be realized without worrying about hardware noise.

However, there are two issues that the reader should be aware of at this stage. These are fundamental questions that will apply to any hardware implementation of a quantum computer and will be relevant in assessing the feasibility of the quantum algorithms we will propose.

Computer's size. The first issue is that, because a logical qubit is made up of several physical qubits, and these physical qubits have a non-negligible size of around hundreds of micrometers (for superconducting qubits), a large fault-tolerant quantum computer will also be very big, also taking into account the classical electronics needed to control the system. As the physical qubit error rate increases, the number of physical qubits needed to create a single logical qubit error rate increases from the minimum value of 13 to (tens of) thousands. For example, IBM is building a dilution refrigerator larger than any commercially available today anticipating the possibility to host a million-qubit chip. While this number may seem large, it is an optimistic estimate for the size of the chip needed to fault-tolerantly encode algorithms for quantum chemistry(2) or factoring 4096-bit numbers using the Shor algorithm(7). The main takeaway here is that, given that the typical spacing between two qubit centers in a

superconducting qubit chip is about 1 mm, if your quantum algorithm requires millions of logical qubits, then be prepared to build a very large fridge.

Computer's clock frequency. Beside the "space" issue, the second, perhaps even more relevant is the "time" one. According to the surface code, operating a logical qubit involve several cycles of physical qubit operations, including measurements. It is certainly beyond the scope of this note to provide a justification, but current estimates for the **logical gate frequency** in the fault-tolerant regime are in the range of **10-100 kHz** (9). This motivates the concerns outlined in Sec 1.1 where we emphasize the importance of going beyond complexity scaling in order to understand thresholds for quantum advantage.

1.2.3 NISQ devices

To solve problems like integer factorization a universal fault-tolerant quantum computer would require millions of qubits with low error rates and long coherence times. While it may take decades of research to achieve such devices experimentally, noisy intermediate-scale quantum (NISQ) computers already exist. These computers consist of hundreds of noisy qubits, which are **qubits that are not error-corrected** and thus perform imperfect operations with a limited coherence time. Strictly speaking, NISQ machines include also analog simulators. However, currently this term indicates quantum devices build with the circuit model in mind but without error correction, e.g. the ones by IBM or Google. We can think about them as intermediate between analog devices and fault-tolerant digital machines.

NISQ machines are capable of supporting quantum logic gates, but in practice, these gates will not be perfect. For example, a CNOT gate may not implement the unitary 4×4 matrix it is supposed to (cfn. Chapter 2). Qubit readouts will also be noisy, leading to misclassification of 0s as 1s, and finite coherence times for circuits. It is unrealistic to expect a NISQ machine to execute a quantum circuit with a depth of 10000 as the result would likely be pure noise. However, if the algorithm is relatively short and the number of two-qubit gates is moderate, it is possible to prepare a quantum state that is close to the theoretical one within the coherence time of the qubits. It is worth noting that the terms "relatively" and "moderate" are used intentionally to be vague, as the required depth and number of gates for successful execution can vary. In 2023, the fidelity of a CNOT operation for IBM's superconducting machines is approximately 0.001, meaning that roughly 1 out of every 1000 CNOT operations will be faulty. Since some companies, like IBM are allowing free cloud access to some of their machines, you can check yourself the status of read-out, 1-qubit and 2-qubit errors in real-time.

NISQ machines can surely fit into fridges, since they work directly at the level of physical qubits. The gate time is also much shorter compared to fault-tolerant logical clock speed. Quoting the popular 2019 *quantum supremacy* experiment by Google(10), we obtain about a **10 ns** gate time, or **100 MHz**. This value is much better than the previous case, yet still worse than classical CPUs. Our claim that the power of a quantum device relies on the possibility to run powerful quantum algorithms and not on a constant gate speed improvement remains intact.

1.3 Quantum software engineering

One of the main advantages of the digital model is that it allows for the use of **multiple layers and abstractions**, enabling the development of complex programs. This is the approach we will take in these notes. It is similar to the way standard computational science or physics courses operate. When writing a program, such as in *C* or *Fortran*, we use abstractions. Few people actually look at the code contained in linear algebra libraries, instead they simply use them as needed. Even fewer people understand how a compiler works or the physical laws underlying the operation of a transistor.

Currently, we are in a hybrid analog era of quantum devices, where quantum application researchers have knowledge of algorithms and a basic understanding of qubits. However, as time goes on, quantum software engineers will likely focus more on the higher-level aspects of their job. Algorithms will be largely hardware agnostic, with the exception of being aware of their fundamental limitations as discussed earlier. This is similar to how classical software engineers must consider memory and wall time limitations. However, unlike in typical computer science courses, in the next chapter we will begin at a lower level, starting with the definition of basic logic gates.

There are generally several levels of abstractions, as exemplified in Fig. 1.3.

High-level program. At the top of the design stack is a high-level language in which quantum programs are written. These programs are **sequences of classical and quantum instructions** to be executed on hybrid systems consisting of both classical and quantum hardware. We shall use therefore high-level classical languages, such as *Python*, to write these instructions, knowing that at the end, everything will be translated into a series of sequential or parallel executions of quantum circuits and qubit measurements, with classical processing in between. In our example, we will write pseudo-code to find the ground state of a physical Hamiltonian. This involves initializing one or more quantum registers, defining the Hamiltonian, and executing a *quantum phase estimation* (QPE) subroutine. The output of the QPE primitive will be further processed, e.g. to translate a binary "phase" into a float point value of an "energy". The QPE primitive will be explained in Chapter **??**.

Primitives. The QPE algorithm can be considered itself an algorithmic primitive that can be used by many types of quantum programs. At the level above, it is simply "called", but in reality it involves the execution of circuits and measurements. The QPE itself assumes the existence of other building blocks, such as controlled unitaries and the quantum Fourier transform (QFT). Other important primitives for quantum programming are: QFT, quantum arithmetic, finding expectation vales of operators, etc.

Circuit level. One step below, we find the actual circuits expressed at the logical gate level, such as one-qubit Pauli gates, one-qubit rotations, controlled rotations, SWAP, and CNOT gates. For instance, each primitive, needs to be developed into logic gate.

These three layers are usually the ones really visible to quantum algorithms developers. In our note, we will consider also one layer below, because it is important to provide a self-contained explanation about what are really the "atoms" of quantum computation, in both

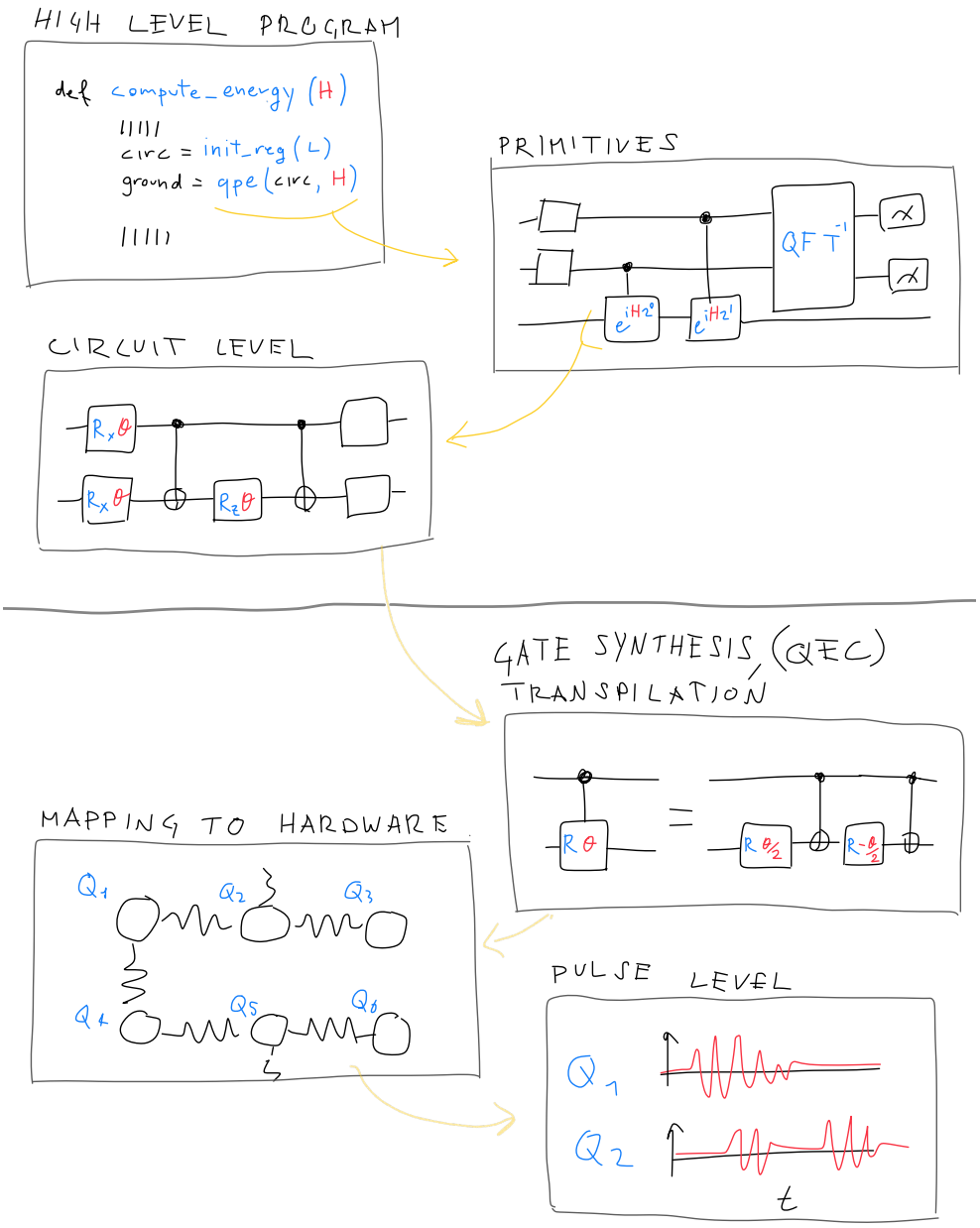


Figure 1.3: The several layers of a quantum program, from high-level human-readable instruction, to transpilation into the available gate-set, to pulse-level control of the physical qubits.

.

near-term and fault-tolerant regimes.

Gate synthesis. Not all logic quantum logic gates used at the circuit level are actually native to a given architecture. For instance, a controlled rotation $C - R_x$ gate, for supercon-

ducting qubits, needs to be expanded into CNOTs and single-qubit rotations as we will see in Chapter 2. More broadly, also in the fault-tolerant regime, gates need to be "compiled" using a smaller set of logical operations. For instance, a parametrized gate admits a (fairly) long expansion of discrete gates. This will also be discussed in Chapter 2 because it directly impacts runtimes. More details can be found in Ref. (11).

Finally, this layer also includes the so-called "transpilation". Similarly to classical computing compilation, a circuit can be optimized and simplified to reach shorter runtimes. In the NISQ regime, decreasing the number of gates is beneficial for decreasing the noise. Current quantum software platforms have built-in transpilers, so one does not need to worry about this.

Hardware mapping. At this level the quantum instruction finally becomes aware of the hardware. Superconducting chips have generally planar and local connectivity, e.g square, hexagon, heavy hexagon lattices. This means that if our program feature two-qubit gates that entangle two qubits which are not directly connected in the hardware, a sequence of SWAP operations needs to be inserted. The positive side is that in principle even an algorithm that requires all-to-all type of connectivity can be realized in a planar chip. The price to pay is that the effective depth of the algorithm, when mapped to a specific hardware, can be longer than naively estimated.

Pulse level. This is the lowest level and generally not visible to software developers. Gates require qubit control. This is achieved by sending a well defined sequence electromagnetic pulses of given waveforms. This is taken care by the low-level part of the quantum software, as much as it happens in conventional software engineering.

Bibliography

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[2] M. Reiher, N. Wiebe, K. M. Svore, D. Wecker, and M. Troyer, "Elucidating reaction mechanisms on quantum computers," *Proceedings of the National Academy of Sciences*, vol. 114, p. 7555–7560, 6 2017.

[3] R. Babbush, J. R. McClean, M. Newman, C. Gidney, S. Boixo, and H. Neven, "Focus beyond quadratic speedups for error-corrected quantum advantage," *PRX Quantum*, vol. 2, p. 010103, Mar 2021.

[4] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver, "A quantum engineer's guide to superconducting qubits," *Applied Physics Reviews*, vol. 6, no. 2, p. 021318, 2019.

[5] M. W. Johnson, M. H. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, *et al.*, "Quantum annealing with manufactured spins," *Nature*, vol. 473, no. 7346, pp. 194–198, 2011.

[6] M. Greiner, O. Mandel, T. Esslinger, T. W. Hänsch, and I. Bloch, "Quantum phase transition from a superfluid to a mott insulator in a gas of ultracold atoms," *nature*, vol. 415, no. 6867, pp. 39–44, 2002.

[7] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Physical Review A*, vol. 86, no. 3, p. 032324, 2012.

[8] S. Krinner, N. Lacroix, A. Remm, A. Di Paolo, E. Genois, C. Leroux, C. Hellings, S. Lazar, F. Swiadek, J. Herrmann, *et al.*, "Realizing repeated quantum error correction in a distance-three surface code," *Nature*, vol. 605, no. 7911, pp. 669–674, 2022.

[9] C. Gidney and A. G. Fowler, "Efficient magic state factories with a catalyzed $|ccz>$ to $2|t>$ transformation," *Quantum*, vol. 3, p. 135, 2019.

[10] F. Arute *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[11] T. Häner, D. S. Steiger, K. Svore, and M. Troyer, "A software methodology for compiling quantum programs," *Quantum Science and Technology*, vol. 3, no. 2, p. 020501, 2018.

Chapter 2

Qubits, gates, and circuits

This Chapter provides a reasonably self-contained introduction to gates and circuits, which constitute the minimal building blocks of quantum primitives and algorithms that we will see in the course.

2.1 Qubits and Hilbert spaces

Formally speaking, **one qubit** is a quantum two-level system, and can be represented by a two-dimensional complex vector

$$|q\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} , \tag{2.1}$$

where we define $|0\rangle$ and $|1\rangle$ as the basis states of the two-dimensional complex vector space $\mathbb{C}^2$, and $\alpha, \beta \in \mathbb{C}$ are complex amplitudes satisfying the normalization condition, i.e. $|\alpha|^2 + |\beta|^2 = 1$. This condition ensures that the total probability of measurement outcomes sums up to 1. The two vectors $|0\rangle$ and $|1\rangle$ are normalized and mutually orthogonal:

$$\langle 0|0\rangle = 1, \quad \langle 1|1\rangle = 1, \quad \langle 0|1\rangle = 0, \quad \langle 1|0\rangle = 0. \tag{2.2}$$

Notice that the notation $|0\rangle$ and $|1\rangle$ has been introduced to carry the analogy with classical *bit* values. A different notation could have worked, e.g. $|\uparrow\rangle$ and $|\downarrow\rangle$. In fact, I suggest unfamiliar readers to pause here to be fully comfortable with the notation, as there are several 0s and 1s in Eq. 2.1.

Alternatively, the state of one qubit can be described as a point on the so-called Block sphere (cfn. Fig. 2.1).

$$|q\rangle = \cos[\theta/2]\,|0\rangle + \sin[\theta/2]e^{i\phi}\,|1\rangle\,, \tag{2.3}$$

where θ, ϕ are now two real-valued parameters. While formally a vector in $\mathbb{C}^2$ is defined by four real parameters, these are not independent: the normalization constraint and the global phase invariance cut down this number to two real independent real parameters.

Crucially, and in contrast to a classical bit, a qubit may be in an arbitrary superposition of its two basis states. This is what is meant by the commonly used sentence "a qubit can be 0 and 1 at the same time". However, even if α and β assume values in the continuum, the

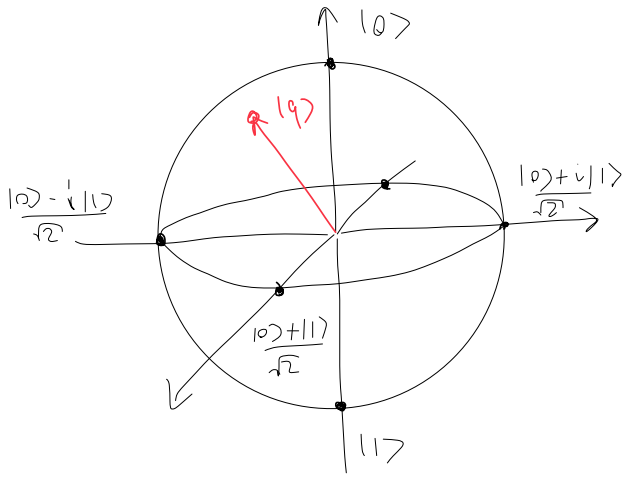


Figure 2.1: A qubit state as a point on the surface of the Bloch sphere.

.

information stored in a qubit can only be **retrieved by measurements**, hence converted into classical information. When measured, a qubit can "collapse" in its '0' value with probability $|\alpha|^2$, or in '1' with probability $|\beta|^2$. This is type of measurement is said to be *in the computational, or z basis.*

Notice also that it is not possible to reconstruct the starting state $|q\rangle$ only from the measurements in the computational basis. For instance, one could not resolve the state $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ from the state $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Both cases are compatible with the measurement outcome "50% 0s, 50% 1s ". To distinguish between the two, we must perform an additional set of measurements in a different "direction". An additional complication is introduced by the measurement's statistical noise. The observed probability of measuring i.e. "0" converges to the expected value $|\alpha|^2$ only in a large number of measurements, or "shots", M limit. The statistical error decreases only as $M^{-1/2}$.

These concepts will be further explored when we will discuss how to measure the expectation values of operators, an essential building block of many algorithms.

The notation for two qubits generalizes as follows. Now the Hilbert space becomes four dimensional, spanned by the basis vector $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ (cfn Sect 2.1.1). A general two-qubit state can be written in vector notation using this basis as

$$\alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + \delta\,|11\rangle \tag{2.4}$$

$$= \alpha \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} + \gamma \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} + \delta \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix}, \tag{2.5}$$

where $\alpha, \beta, \gamma, \delta \in \mathbb{C}$ are the component along these four orthonomal basis vectors, and also satisfy the usual normalization condition. It is important to notice that here we implicitely choose a convention about the ordering of the qubits! Some popular software development

tools use another convention, see Sect.2.3. So please keep this in mind when benchmarking the calculations of this Chapter using numerical tools.

More generally, the quantum state of n qubits can be represented by a complex-type vector of length 2^n. The state of a general n-qubit system can be an arbitrary superposition over all 2^n computational basis states, i.e.,

$$\sum_{q_1 q_2 ... q_n \in \{0,1\}^n} c_{q_1 ... q_n} |q_1 ... q_n\rangle = \sum_{i=0}^{2^n-1} c_i |i\rangle , \tag{2.6}$$

where we can define a short-hand notation for the basis state $q_1...q_n$ in its binary number format, i.e an integer i. For instance the basis state $|11\rangle$ can be labeled with $|3\rangle$. The complex amplitudes c_i satisfy the normalization condition $\sum_i |c_i|^2 = 1$.

From Eq. 2.6 we can see that a quantum state can encode any probability distribution that can be discretized on a grid of 2^n. However, specifying all coefficients c_i is impossible as soon as the number n increases. With just $n = 50$ qubits, one can encode a wavefunction that is formally defined by 2^{50} complex parameters, i.e 2×2^{50} independent real-valued parameters, therefore order of *petabyte* of equivalent memory in a classical computer.

2.1.1 Product and entangled states

The Hilbert space of a n-qubit system is the tensor product of n single qubit Hilbert spaces. If we consider two qubits (or generally two sub-systems) a and b, the Hilbert space of the composite system is product of the subsystems, $\mathcal{H} = \mathcal{H}_a \otimes \mathcal{H}_b$. The dimension of the final Hilbert space is the product of the dimensions of the subsystems one, in the two qubits case, four. Let us define $\{|a_i\rangle\}$ and $\{|b_j\rangle\}$ the orthonormal bases of $\mathcal{H}_a$ and $\mathcal{H}_b$, then every vector of the total system, belonging to $\mathcal{H}$ can be written in the form

$$|\psi\rangle = \sum_{ij} M_{ij}(|a_i\rangle \otimes |b_j\rangle), \tag{2.7}$$

where M_{ij} are complex coefficients. The shorthand notation for $|a_i\rangle \otimes |b_j\rangle$ is $|a_i\ b_j\rangle$, so we recover the "two-qubits" basis states introduced above, e.g. $|0\rangle \otimes |1\rangle := |01\rangle$. Interestingly, not all vectors in $\mathcal{H}$ belong to the same class.

Product states. Suppose two qubits are in single-qubit states each $|\alpha\rangle := \alpha_0 |0\rangle + \alpha_1 |1\rangle$ and $|\beta\rangle := \beta_0 |0\rangle + \beta_1 |1\rangle$, then the entire system is defined by the tensor product of the two individual states, which corresponds to the Kronecker product in vector notation, i.e.

$$|\psi\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \otimes \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} = \begin{pmatrix} \alpha_0\beta_0 \\ \alpha_0\beta_1 \\ \alpha_1\beta_0 \\ \alpha_1\beta_1 \end{pmatrix} . \tag{2.8}$$

or,

$$|\psi\rangle = \alpha_0\beta_0 |00\rangle + \alpha_0\beta_1 |01\rangle + \alpha_1\beta_0 |10\rangle + \alpha_1\beta_1 |11\rangle . \tag{2.9}$$

Crucially, not all two-qubit states can be written in this way, i.e. as a product of single-qubit states. These are called entangled states.

Entangles states. These vectors are not product states. While Eq. 2.7 gives the impression that every vector in $\mathcal{H}$ can be written as a product state, this is true only if the rank, i.e. the number of linearly independent rows or columns, of M_{ij} is 1. One can show that the 2×2 matrix M_{ij} constructed from the amplitudes in Eq. 2.8 has indeed rank 1.

For instance, Eq. 2.8 cannot represent the following vector (the reader is invited to check it)

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle). \tag{2.10}$$

Indeed, the matrix M_{ij} has rank 2. Entangled states should necessarily be present in any meaningful quantum computation. Product states can be already realized by disconnected, non-interacting qubits.

2.1.2 Operators

An operator A is a linear map of the Hilbert space onto itself. For practical purposes, it is given by its matrix elements defined over a basis. For our qubits, this basis is the computational one

$$A = \begin{pmatrix} \langle 0| A |0\rangle & \langle 0| A |1\rangle \\ \langle 1| A |0\rangle & \langle 1| A |1\rangle \end{pmatrix} \tag{2.11}$$

In these notes, we will care about *hermitian* operators, i.e. $A = A^\dagger$, which are linked to observable quantities, such as the Hamiltonian, and *unitary* operators, i.e $A\, A^\dagger = A^\dagger\, A = I$, that represent single or multi-qubit gates. Traditionally, a generic unitary operator is denoted with the symbol U.

It is also important to define how *local* operators are written as operators acting on a larger Hilbert space. This is the basis of writing down the matrix representation of a quantum circuit.

Suppose that the operators A and B act respectively on the unentangled subsystems $|a\rangle$ and $|b\rangle$ respectively. The action of the the operator in $\mathcal{H}$ is:

$$(A \otimes B)(|a\rangle \otimes |b\rangle) = (A\,|a\rangle) \otimes (B\,|b\rangle) \tag{2.12}$$

Since any operator on $\mathcal{H}$ can be written as a sum of product operators, the above equation is sufficient to define the action of any operator on any state of $\mathcal{H}_a \otimes \mathcal{H}_b$. For instance, the unitary matrix of the CNOT gate cannot be written as $(A \otimes B)$ but as a sum of these terms.

If we define the usual basis for the two subsystems $\{|a_i\rangle\}$ and $\{|b_j\rangle\}$, and order them in lexicographically, i.e in ascending binary order, then the matrix form of the operator is given by the *tensor product*

$$\begin{bmatrix} a_{00}B & a_{01}B & \cdots \\ a_{10}B & a_{11}B & \\ \vdots & & \ddots \end{bmatrix} \tag{2.13}$$

Namely, the matrix B is nested inside a bigger matrix, multiplied by each element of A. If the two subsystems are two qubits, one concrete example can be following

$$(X \otimes I) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & 1 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ 1 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & 0 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{2.14}$$

where the X operator will be also better defined as "X gate" in the next Section. The total $(X \otimes I)$ operator, "flips" the first qubit and does nothing on the second one. As an exercise, we can check that the matrix formulation of the operation is consistent with the "physical" way to carry out the same process. Let's suppose that this operation acts on two qubits in their $|0\rangle$ state.

$$(X \otimes I)(|0\rangle \otimes |0\rangle) = X|0\rangle \otimes I|0\rangle = |1\rangle \otimes |0\rangle = |10\rangle \tag{2.15}$$

Following Eq. 2.8, the basis state denoted by $|10\rangle$ is the vector $(0010)^T$ in $\mathcal{H}$. This is the same outcome that we receive if we perform the matrix-vector multiplication, using the right-hand side matrix in Eq 2.14, and the basis vector $(1000)^T := |00\rangle$.

Turns out that a quantum circuit is just a very big linear algebra problem. And this is indeed what classical emulators of quantum computation are doing. Under this perspective, it is also easy to understand that simulating classically a general quantum circuit, made of L qubits, becomes exponentially costly in memory and runtimes, as it requires handling matrices of size $2^L \times 2^L$.

2.1.3 Projectors

In the following chapters we will make use of projectors, which we formally define here as operators that select a given component of a quantum state. For instance, the projector P_i,

$$P_i = |i\rangle \langle i|, \tag{2.16}$$

applied to Eq. 2.6 gives $P_i|\psi\rangle = c_i|i\rangle$. They satisfy the properties

$$P^2 = P, \quad P^\dagger = P, \tag{2.17}$$

and their eigenvalues are either 0 or 1. This follow from the above property.

2.2 Quantum gates

2.2.1 One qubit gates

After this mathematical digression, we can safely introduce quantum gates. A one-qubit quantum gate is a unitary operator on the Hilbert space of one qubit, that, following the above consideration, can be represented by a 2×2 matrix. Here we list the most common gates, which also include the "Pauli gates", I, X, Y, Z.

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \tag{2.18}$$

The qubit basis $|0\rangle$, $|1\rangle$ are eigenvectors of the Z operator. This is why the computational basis, is also called z-basis. Other important gates are:

$$\mathsf{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \mathsf{T} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}, \quad \mathsf{S} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}. \tag{2.19}$$

These first seven gates share a common pattern: they are *discrete* gates. The H gate is called *Hadamard* gate. Among these, theTgate is infamously known to be the most expensive one

to realize in the fault-tolerant regime. Indeed, using a surface code for error correction, T gates consume special states, called *magic states*,(1) which require many qubits and many surface code cycles. As a surface code cycle requires frequent measurements for all gates, the gate time of a T operation is the longest one(2). As we will see, the T-gate may look exotic and far from concrete applications, but in reality it is central to realize many useful gates.

As a matter of fact, quantum resource estimates for fault-tolerant algorithms, use the *T-count* or the *T-depth* as a proxy to estimate runtimes (see e.g this paper that provides end-to-end resource estimate for a quantum algorithm useful for finance(3)).

Parametrized gates. These gates are also crucial for quantum algorithms, as they allow to "rotate" arbitrarily the state of a qubit. The rotation operators are generated by exponentiation of the Pauli matrix $\sigma \in \{X, Y, Z\}$ according to

$$e^{-i\sigma\theta} = \cos[\theta]I - i\sin[\theta]\sigma \tag{2.20}$$

The three rotations gates, of a angle θ are given by the following matrices

$$R_X[\theta] = \begin{bmatrix} \cos[\frac{\theta}{2}] & -i\sin[\frac{\theta}{2}] \\ -i\sin[\frac{\theta}{2}] & \cos[\frac{\theta}{2}] \end{bmatrix}, R_Y[\theta] = \begin{bmatrix} \cos[\frac{\theta}{2}] & -\sin[\frac{\theta}{2}] \\ \sin[\frac{\theta}{2}] & \cos[\frac{\theta}{2}] \end{bmatrix}, R_Z[\theta] = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}, P[\theta] = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}. \tag{2.21}$$

We also include the *phase* gate, that is equivalent to a RZ gate up to a phase factor $P[\theta] = e^{i\frac{\theta}{2}} R_Z[\theta]$.
Notice the factor $\theta/2$ in the definition of the gate!

Useful gate identities. Many algebraic identities can be proved and may be useful when constructing, understanding and low-level compiling circuits. For instance, $\mathsf{S} = \mathsf{T}^2$, $\mathsf{H} = (X + Z)/\sqrt{2}$, $XYX = -Y$, $\mathsf{H}\ X\ \mathsf{H} = Z$, and so on.. For instance, the latter identity will be useful to perform a *change of basis*, as much as $\mathsf{H}\ Z\ \mathsf{H} = X$.

Here we also provide identities involving rotations. In particular, a unitary operation U on a single qubit, can be expressed as a sequence of rotations and a phase

$$U = e^{i\alpha} R_z(\beta) R_y(\gamma) R_z(\delta), \tag{2.22}$$

where $\alpha, \beta, \gamma, \delta$ are real parameters. Finally, we also include an identity which is very useful in practice, and follows from Eq. 2.22. Given, again, a single qubit unitary U, there exist three single qubit operators, A, B, C and a real parameter α, such that

$$ABC = I, \tag{2.23}$$

$$e^{i\alpha} AXBXC = U \tag{2.24}$$

2.2.2 Gate synthesis, logical vs parametrized gates

As we have seen, some of the elements of the one-qubit set of gates presented above are "redundant". For instance, the R_X gate can be substituted by the following operation $R_X = \mathsf{H} R_Z H$, which follows from the above $\mathsf{H}\ Z\ \mathsf{H} = X$ identity. Indeed, real hardware are usually limited to a smaller set of *physical gates*, that can be engineered as a sequence of pulses, and

that are used to "synthesize" other gates (cfn. Sec. 1.3) Interestingly, in the case of IBM Quantum hardware, all one qubit gates are compiled using the minimal set of I, X, R_Z and $SX = \frac{1}{2}\begin{bmatrix} 1+i & 1-i \\ 1-i & 1+i \end{bmatrix}$ gates.

The need of compiling one-qubit gates using a smaller set will hold also in fault-tolerant-hardware as fault-tolerant operations are typically available only for a limited number of gates. Indeed, you may have noticed the apparent contradiction of having gates, which are parametrized by a countinuos parameter, listed as possible building block of a digital quantum computer. Indeed, in the previous section we motivated the need to go beyond quantum simulators, exactly because continuos operation cannot be error corrected.

In NISQ setting, a qubit rotation can be easily realized by driving the qubit with a well definite pulse shape, realizing the θ-rotation. However, calibration errors will likely realize an imperfect gate: 1) the rotation angle will be affected by an unpredictable error $\theta + \delta\theta$, 2) the rotation axis can also suffer from error, in such a way that e.g. the intened R_Z rotation operation may have spurious components $R_Z + \epsilon R_X + \epsilon' R_Y$.

Fault-tolerant computers needs to realize a **digital rotation** as much as classical computers do. It is possible to convince ourselves that it is possible to generate rotations of arbitrary angles by using e.g. only H and T discrete gates. For instance, already the TH operation generate a rotation of an angle which is not an exact divisor of 2π. It follows by definition that T is a rotation of $\pi/4$ on the z-axis, and it can be seen that the H gate can also be understood as a π rotation, about the vector $\vec{h} = (\vec{x} + \vec{z})/\sqrt{2}$. The rotation $R_{\vec{n}}(\theta)$ produced by this operation is

$$R_{\vec{n}}(\theta) = R_{\vec{h}}(\pi) \times R_{\vec{z}}(\pi/4). \tag{2.25}$$

The resulting angle θ can be found from the following general formula:

$$\cos(\theta/2) = \cos(\theta_1/2)\cos(\theta_2/2) - \vec{n}_1\vec{n}_2 \sin(\theta_1/2)\sin(\theta_2/2) \tag{2.26}$$

. Inserting the specific values we get

$$\cos(\theta/2) = \cos(\pi/2)\cos(\pi/8) - \frac{1}{\sqrt{2}}\sin(\pi/2)\sin(\pi/8) = -\frac{1}{\sqrt{2}}\sin(\pi/8) \tag{2.27}$$

. The solution $\theta \approx 1.096$ is an irrational divisor of 2π. Therefore, repeated application of the sequence TH will never produce an angle which is multiple of 2π. Starting with the appropriate state, that can be prepared with just H or T gates, it is possible to reach any point in the Bloch sphere with arbitrary precision, without continuous rotations. As an example, it can be shown that a rotation by an angle $\pi/8$ admits the following expansion in terms of H, T, S, X gates: HTHTSHTSHTSHTSHTHTSH THTHTSHTHTSHTHT HTSHTSHTSHTSHTSHT SHTSHTSHTSHTHTSHTH TSHTHTHTSHTSHT SHTHTHTHTSHTSHTHTSHX to attain an error of 10^{-4}(see (4)).

One can already see that, while continuous rotations are very simple and fast operation in NISQ regime, they become very long and complex in fault-tolerant-regime, with a T-depth that depends on the required accuracy.

This example is a specific instance of the more general **Solovay-Kiteav theorem**, perhaps one of the most fundamental results enabling quantum computation. The theorem

provides an upper bound for the number of gates required to achieve a desired accuracy in the realization of an arbitrary gate. Indeed, a quantum circuit of m gates can be approximated with an error ε by a quantum circuit of depth $\mathcal{O}(m \log^c[m/\varepsilon]$ made of gates from a desired finite universal gate set(5).

2.2.3 Two qubit gates

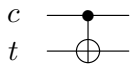
Since one-qubit gates cannot produce entangled states, as by definition a product state remains a product state under the action of a tensor product of one-qubit gates, we need to introduce at least one genuine two-qubit gate to perform quantum computation.

Let's consider a two qubit system $|c\rangle \otimes |t\rangle = |c\, t\rangle$, i.e where the first qubit is in the *control* state c, and the second is in a *target* state t. Then, following this *ordering* the **CNOT gate** is defined via:

$$\mathsf{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.28}$$

The CNOT operator leaves unchanged the state of the control qubit but it "flips" the target qubit. Component-wise this operation swaps the $|10\rangle$ component with the $|11\rangle$ one. This operation can be also called *controlled X* or C-X operation. Notice also that the matrix form of the CNOT crucially depends on the ordering of the qubits. If the control qubit is the right-most one, then the matrix is different. This is one of the most trivial errors that one can do when approaching quantum circuits and reading different sources, or using different software packages that assume different qubit's ordering.

The CNOT gate is usually denoted in quantum circuits with the following symbol



where the upper(lower) qubit is the control(target). The CNOT gate is the minimal operation wherw we can appreciate superposition in action. Indeed, the control qubit may be in superposition, such that, loosely speaking, you control and do not control at the same time. As we will see, this minimal example is able to prepare the Bell state.

Another important two qubit gate is the **SWAP gate**. The operation it performs is the following

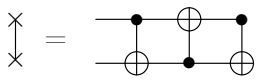
$$\mathrm{SWAP}\,|ab\rangle = |ba\rangle \tag{2.29}$$

and is mathematically represented by the following matrix,

$$\mathrm{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.30}$$

This gate is essential when an ideal circuit, which is designed with arbitrary connectivity in mind, needs to be embedded in a chip with local connectivity. This gate can be "compiled"

into a sequence of three CNOTs.



This is the first *circuit identity* we see in these notes. Other two-qubit gates can be introduced as needed in the following Chapters, for the time being these two gates are enough for our purposes.

2.3 Circuits and circuit emulators

A quantum circuit is a graphical model to describe the sequence of operations performed on a qubit register. A circuit features *gates* and *wires*. The *evolution* of a qubit state is depicted by a horizontal line. Time flows from left to right. Therefore the horizontal lines are not physical wires but represent the qubit, or a qubit register, during the computation.

Vertical lines connecting qubits imply an interaction between them (cfn. the CNOT gate). One-qubit gates are usually depicted by a "box" containing the letter specifying the gate. Bigger boxes encompassing multiple qubits denote an unspecified multi-qubit operation.

Classical bit registers are denoted by doubled line. These are typically used, to formally store results of a measurement, and are often neglected unless there is a specific reason to include them, i.e. a non-trivial quantum-classical feedback operation, like applying a gate to qubit i conditionally to a given measurement outcome of qubit j.

Finally, measurements operation, which is not described by a unitary gate, are depicted with a special symbol. Notice that not all qubit needs to be measured at the end of the circuit.

An illustrative example of a quantum circuit is plotted below (Fig 2.2). It is perhaps useful to refresh the notation concerning the order of the operations and the qubit ordering.

The time flows from left to right, so the initial state is $(|0\rangle \times |\psi\rangle)$, if we assume the *convention of assigning to the upper most qubit, the leftmost place in the ordering of the tensor product*. This is called the big-endian convention. It is important to notice this, because some software packages, like Qiskit, use the opposite convention. This impact the definition of the CNOT matrix for instance, as well as the matrix format of any tensor product of operators. In case you want to validate your understanding using quantum circuit emulators, this is certainly something to keep in mind when reading the output wavefunction.

To recap: big-endian convention for the tensor product $|abc\rangle = |a\rangle \otimes |b\rangle \otimes |c\rangle$ assumes that the state $|a\rangle$ is qubit 0, i.e. q_0. Little-endian convention instead assigns the state $|c\rangle$ to the qubit register 0. Clearly, this choice only impacts the classical emulation of the circuit. Quantum mechanics is invariant under the re-labeling of the qubits.

Notice that here, to draw the most general setting, we assumed that the second qubit has already been initialized in arbitrary one-qubit state $|\psi\rangle = (\psi_0, \psi_1)^T$. After the initialization, the following operations are applied, in this order: $(\mathsf{H} \otimes T)$, CNOT, (C-U_1), and U_2. This means that the final state is

$$|\phi_{final}\rangle = U_2 \; CU_1 \; \mathsf{CNOT} \; (\mathsf{H} \otimes T) \, |0 \; \psi\rangle \, . \tag{2.31}$$

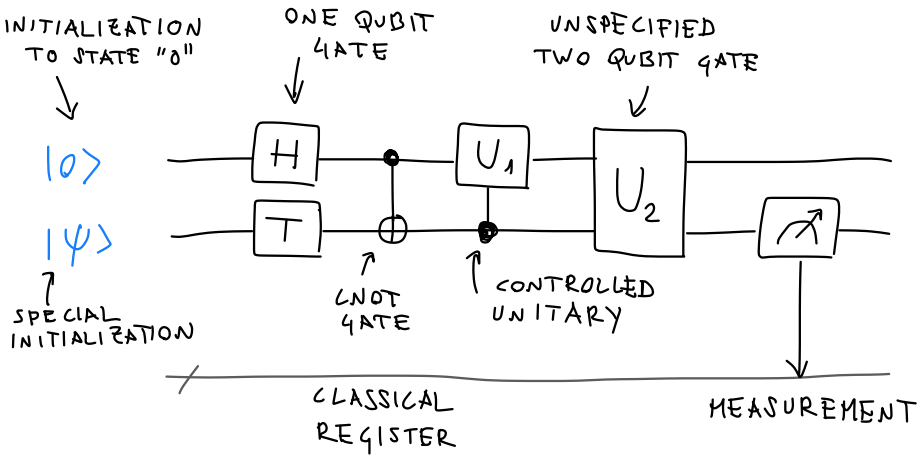


Figure 2.2: A generic quantum circuit featuring a two-qubit register and a classical register. The circuit is made of two single-qubit gates, a CNOT gate, a controlled unitary gate (C-U_1), where the second qubit controls the first, and a generic two-qubit operation U_2. Finally the second qubit is measured on the computational basis, and the measurement outcome is stored in a classical bit register.

.

Classical circuit emulators implement this chain of linear algebra operations. For instance, the first two steps are represented by the following

$$\mathsf{CNOT}\ (\mathsf{H}\otimes T)\,|0\ \psi\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}\begin{bmatrix}1&0&1&0\\0&e^{i\pi/4}&0&e^{i\pi/4}\\1&0&-1&0\\0&e^{i\pi/4}&0&-e^{i\pi/4}\end{bmatrix}\begin{bmatrix}\psi_0\\\psi_1\\0\\0\end{bmatrix}. \tag{2.32}$$

A final note about **measurements**. Suppose that the final output state is $|\phi_{final}\rangle = (\phi_0,\phi_1,\phi_2,\phi_3)^T = (\phi_{00},\phi_{01},\phi_{10},\phi_{11})^T$. Let's recall for the last time that, while in a classical emulator (e.g. linear algebra processor) we have access to the full statevector, i.e. all four components, in real quantum hardware we can only infer the quantum state through measurements. If we measure all qubits, then we could measure the normalized frequency count of all four-bit strings '00','01','10', and '11'. In this example we only measure the second qubit, therefore we expect our output to be either a '0' or a '1'. From the definition, and given the big-endian convention, the probabilities to measure '0' or '1' on the *second* qubit are

$$P(0) = |\phi_{00}|^2 + |\phi_{10}|^2, \quad P(1) = |\phi_{01}|^2 + |\phi_{11}|^2. \tag{2.33}$$

Measurements can be classically emulated by drawing samples from the vector of the squared amplitudes. It is often crucial to assess how many measurements M are needed to resolve with the required accuracy the desired output. This is particularly compelling when one needs to evaluate the expectation value of an operator,

$$\langle A\rangle = \langle\phi|\,A\,|\phi\rangle\,. \tag{2.34}$$

A classical emulator can evaluate this "exactly" from a vector-matrix-vector operation. Traditionally, in literature, this is called *statevector* emulation mode. A quantum computer, as we will see, needs to use measurements in potentially several bases to evaluate this quantity.

Finally, we notice in Fig. 2.2 the presence of a big box, labeled as U_2. Sometimes, these black boxes are called **oracles**. These are operations where one knows the intended input-output mapping, but, in the most general case, a given circuit implementation is not available or not specified.

2.3.1 Oracles

Besides the above definition as "unspecified" unitary, oracles are often used as building blocks to many textbook quantum algorithms, such as Grover's algorithm. The usage of oracles allows quantum information theorists to develop new codes, without worrying about details. However, it is implied that the execution of oracles is the most time-consuming part of the algorithm, such that the complexity of these quantum algorithms is often given as the number of *oracle calls*. When a concrete circuit implementation of said oracle is available, one can simply multiply the number of T-gates per oracle by the number of oracle calls per circuit, to get a reasonable runtime estimate.

Roughly speaking, oracles are the quantum way to perform $f(x)$'s without sacrificing unitarity. Let's assume we want to implement a *known* function $f : \{0,1\}^n \to \{0,1\}^m$, that acts on a n-bit input $x = x_0 x_1 \cdots x_n$ and product a m-bit output $f(x) = f_0 f_1 \cdots f_m$. For instance, $x = 5$ and with a register size $n = 4$ could be represented by the binary encoding 0101.

The first problem is that n and m can be different so the operation has a different number of qubits incoming and outgoing. However, this could be fixed by assuming the two registers of the same size of $\max[n, m]$. The real issue is that defining the action of the oracle as $U|x\rangle = |f(x)\rangle$ would be still incorrect. If the function $f(x)$ is not invertible, one cannot safely **reverse the operation**, i.e cannot construct the $O^\dagger$.

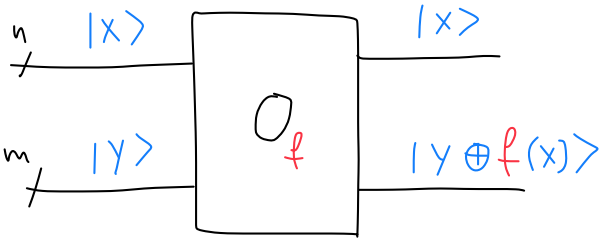


Figure 2.3: A generic circuit drawing of a Boolean oracle, O_f implementing the classical function f.

.

To solve this conundrum, one needs to construct and allow for two registers, the input one, of size n, and an output one of size m, which is initialized on a m-bit zero state$|0\rangle$. After the operation, the input register still stores the input value x, while the output stores $f(x)$. In this way, the operation is reversible, since the input also "survives" the operation. This

choice is still not fully satisfactorily because it requires a fixed input in the second register. To allows for a real unitary operation, the output register should be in an arbitrary state $|y\rangle$, and is changed into $|y \oplus f(x)\rangle$, where $\oplus$ denotes the addition modulo 2 (or XOR). These oracles, that require this two-register structure are called *Boolean oracles* (shown in Fig. 2.3).

From a complexity theory perspective, if $f(x)$ can be implemented efficiently using a classical computer, then the corresponding oracle O_f is also efficient. This is often enough for quantum information scientists.

However, being efficient, i.e. non-exponentially scaling, is not a guarantee for practicality as discussed in Chapter 1. The branch of quantum information devoted to the investigation of practical ways to implement *reversible* mathematical operations, at the circuit level, is called **quantum arithmetic**.

2.3.2 Controlled operations

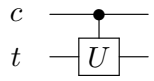
After these basic definitions we are in the position to define circuit identities and general controlled gates. Moreover, we can revisit the CNOT under this "oracular" perspective, which may be useful.

We can see the control qubit as the input register and the target one as the output as in Fig. 2.3. Then the CNOT can be understood as two-qubit Boolean oracle implementing the function $f(x) = x$. The action of the CNOT gate is

$$\mathsf{CNOT}\,|x\rangle\,|y\rangle = |x\rangle\,|y \oplus x\rangle \tag{2.35}$$

If we apply the CNOT gate to $|x\rangle\,|0\rangle$ we obtain $|x\rangle\,|x\rangle$. This procedure is called *entangled copy*.

The CNOT gate is a controlled X gate, and it can be generalized to other unitaries U. A controlled U, C-U, gate, where U is a single qubit gate, and assuming our standard qubit ordering is charaterized by the following operator, circuit symbol, and matrix:



$$C-U = |0\rangle\langle 0|\, I + |1\rangle\langle 1|\; U = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & u_{00} & u_{01} \\ 0 & 0 & u_{10} & u_{11} \end{bmatrix} \tag{2.36}$$

Controlled rotations are not native gates, but can be prepared using CNOT and one-qubit rotations.

In these note we will consider, as illustrative example, controlled single-qubit unitaries. The identity of Eq. 2.23 is useful to derive the circuit decomposition for controlled operations. The first step is to understand that a controlled phase shift on the target qubit is equvalent to placing a phase gate $P(\alpha)$ (cfn. Eq. 2.2.1). Now, it is easy to understand that the circuit in Fig. 2.4 realized the condition that the unitary is not applied when the control qubit is in '0', because $ABC = I$. Instead, when the control qubit is in '1', the presence of CNOT gates (which are C-X gates) implies that the effective sequence of operations applied to the

target qubit are $AXBXC$. Moreover, when the control qubit is in '1', then the phase $e^{i\alpha}$ is applied, thus realizing the unitary U.

Now, in special and relevant cases, i.e $U = R_z$ and U $= R_y$ rotations, the circuit further simplifies. And it is possible to guess the operators A, B, C without much mathematical steps. The idea here is to perform two half-rotations in different direction. In one case, they will cancel out, in the other case the will add up.

Notice however, that this simplified circuit is not valid for controlled R_x rotations. In this case the general construction is needed[1]

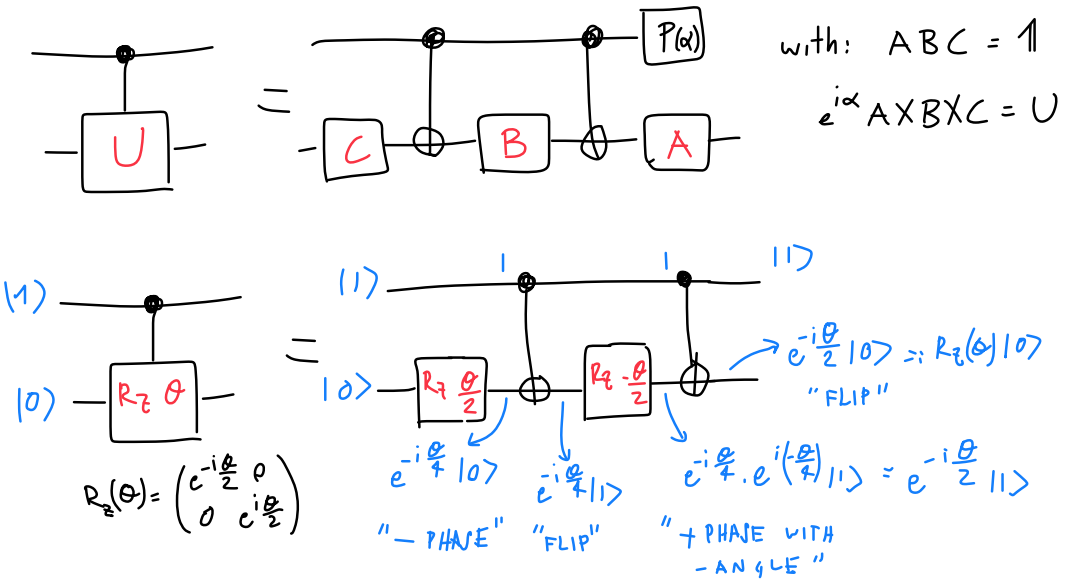


Figure 2.4: Upper circuit: circuit equivalence for a generic single qubit controlled-U gate. Lower circuit: particular example of a $C - R_z$ rotation, with all intermediate states, for the input state $|1\rangle \otimes |0\rangle$, that is mapped to the $|1\rangle \otimes R_z(\theta)\,|0\rangle$. The reader is encouraged to try it out with the other three basis states.

.

The circuit is shown in lower panel of Fig. 2.4 for the special case of R_z gate. It is instructive to keep track of the state transformation at each step of the circuit evolution, to convince ourselves that the circuit is indeed producing the intended output. As we will see, this "sandwich of CNOTs" structure is a recurring pattern in many algorithms.

Finally, we also introduce the **Toffoli** gate, which is the first example of a genuine three-qubit gate. This gate that flips the last qubit only when the first two qubits are '1'. The action of this operator is

$$|x_1\rangle\,|x_2\rangle\,|y\rangle = |x_1\rangle\,|x_2\rangle\,|y \oplus x_1\ x_2\rangle\,, \tag{2.37}$$

[1]Exercise: find the operators A, B, C in this case. Hint: it is not qualitatively different from the previous cases.

and represents the reversible version of the conventional AND operation. Moreover, the Toffoli gate can also simulate the NAND operation, so we can transform in principle any classical circuit into a quantum one, using Toffoli gates.

2.3.3 Universal gate sets

A *universal* gates set can **approximate any unitary transformation** on any number of qubits up to any desired precision. Differently from the classical case, the number of elements in this set is greater than one.

Adapting the exposition by S. Aaronsoon, one can intuitively understand why this is the case. For instance, the gate set must create interference and superposition, so it must go beyond a simple CNOT, which cannot create superposition starting from a basis state.

Single qubit rotations, or the Hadamard, may create superpositions but cannot create entanglement. So, a mixture of one and two qubit gates is needed.

Moreover there must be at least one gate to span the imaginary space, like the phase gate of Eq. 2.19. However, even the set $\{\mathsf{CNOT}, \mathsf{H}, \mathsf{S}\}$ that fulfill these requirements is still not universal. This set is called *Clifford* set, as they can generate any gate in the Cliffor group, e.g. the Pauli gates. The Gottesman-Knill Theorem proves that the Clifford set is not universal, as it cannot approximate arbitrarily good and with a finite set of operations all other possible gates. The theorem also shows that quantum circuit using only the Clifford set can be simulated efficiently on a classical computer. This may appear conterintuitive, as one can generate highly entangled states using Clifford gate.

It turns out that there exist many universal gate sets. For instance, $\{\mathsf{CNOT}, R_x(\pi/4), \mathsf{S}\}$, $\{\text{Toffoli}, \mathsf{H}, \mathsf{S}\}$, or $\{\text{Clifford}, T\}$ are universal. Currently the two most popular universal gate sets are the ones made of CNOT and single qubit rotations, which is widely used in NISQ era

$$\{R_x(\theta), R_y(\theta), R_z(\theta), \mathsf{S}, \mathsf{CNOT}\}, \tag{2.38}$$

and the Clifford+ T set, which is more popular in fault-tolerant algorithms

$$\{\mathsf{CNOT}, \mathsf{H}, \mathsf{S}, \mathsf{T}\}. \tag{2.39}$$

The Solovay-Kitaev theorem then assure us that we can approximate all the other gates using an efficient number of operation taken from these sets. In Sect.2.2.2, for instance, we gave an example about constructing continuos rotations using the Clifford+ T set.

2.4 Measuring operators

Many quantum algorithms require the evaluation of

$$\hat{O} = \langle\psi| O |\psi\rangle \tag{2.40}$$

.

We stress once again that in a real setting we cannot access all the components of the state $|\psi\rangle = (\psi_0, \psi_1, \cdots, \psi_{2^N-1})^T$ and therefore the above cannot be computed using linear algebra operations[2]

[2]it could be in the emulation of quantum computation, which is however, exponentially costly.

Similarly to quantum Monte Carlo, one needs to sample from $|\psi|^2$ introducing a statistical error. Let's consider first the case of a diagonal operator, Z and one qubit. From the definition we have

$$\hat{Z} = \langle\psi| O |\psi\rangle = \begin{bmatrix}\psi_0^* & \psi_1^*\end{bmatrix} \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix} \begin{bmatrix}\psi_0\\ \psi_1\end{bmatrix} = |\psi_0|^2 - |\psi_1|^2 = \text{Prob}(\sigma_z = 0) - \text{Prob}(\sigma_z = 1), \tag{2.41}$$

namely, the differences of the normalized frequency counts of the readout '0' and '1'. Notice also that, a number M of repetitions, i.e. state preparations + measurement, is needed to compute $\hat{Z}$.

It is already clear that the expectation value of Z on the state $|+\rangle = 1/\sqrt{2}\ (1\ 1)^T$, if estimated empirically, will be 0 only within statistical error, which decreases with $1/\sqrt{M}$.

2.4.1 Measuring non-diagonal operators

To measure arbitrary single-qubit operators one needs to perform additional operations. Suppose that we want to measure the expectation value of $\langle\psi|X|\psi\rangle$: we make use of the following identity:

$$X = H\ Z\ H, \tag{2.42}$$

which can be verified using the definition of Sect 2.2.1. This means that the expectation value $\langle\psi|X|\psi\rangle$ is equivalent of the expectation value of the operator Z taken on the *rotated* state $|\psi'\rangle = |\mathsf{H}\psi\rangle$, because $\langle\psi|X|\psi\rangle = \langle\psi|\mathsf{H}Z\mathsf{H}|\psi\rangle = \langle\psi'|Z|\psi'\rangle$. All in all, to measure the X operator, one simply need to place an Hadamard gate before the measurement in the computational basis.

Similarly, measurements in the Y basis can be performed using the following gate

$$K = \frac{1}{\sqrt{2}} \begin{bmatrix}1 & -i\\ 1 & i\end{bmatrix} = \mathsf{H}S^\dagger \tag{2.43}$$

which has not a well defined name, but in our notes will we label it with K. Similarly the $R_x(\pi/2)$ gate can be used. While $R_x(\pi/2) \neq K$, it is possible to show that $R_x(\pi/2)^\dagger Z R_x(\pi/2) = R_x(-\pi/2) Z R_x(\pi/2) = Y$, as much as $S\mathsf{H}Z\mathsf{H}S^\dagger = Y$

2.4.2 Measuring multi-qubits and non-diagonal operators

The next step is the ability to calculate expectation values of correlators such as ZZ, or XX. It follows the definition of the operator ZZ that the practical way to calculate the expectation value from samples is the generalization of Eq. 2.44, where we check the *parity* of the readout string.

$$\langle\psi|\, ZZ\, |\psi\rangle = \text{Prob}(\sigma = 00) - \text{Prob}(\sigma = 01) - \text{Prob}(\sigma = 10) + \text{Prob}(\sigma = 11), \tag{2.44}$$

i.e when two spins which are parallel (anti-parallel) they contributes with a + (−) sign to the overall sum over the sampled strings σ. This rule generalizes i.e. to the calculation of operator such as $ZZIIZ$: in this case the string $'11111'$ count contributes with a − sign because qubits

0 and 1 are parallel, but qubit 4 is read out in its $'1'$ state (which has eigenvalue -1). The readout of the qubits 2 and 3 is irrelevant because there the identity operator is applied.

The more general case of a generic Pauli string operator, tensor product of arbitrary single qubit Pauli operator is treated similarly to the single-qubit case. One needs to find a unitary operator U, such that

$$P = U^\dagger P^{\text{diag}} U, \tag{2.45}$$

where P^{diag} is a tensor product of diagonal Pauli matrices Z, I, where the occurence of identity operators in P are left unchanged in P^{diag}. For instance, the operator $XX = X \otimes X$ can be measured using $U = \mathsf{H} \otimes \mathsf{H}$ and $P^{\text{diag}} = ZZ$. Similarly, the operator $XIYY$, can be measured using $U = \mathsf{H} \otimes I \otimes K \otimes K$ and $P^{\text{diag}} = ZIZZ$.

Measurement of (linear combination of) Pauli strings is an essential step of one of the most commonly used quantum algorithms, the variational quantum eigensolver (VQE). Luckily, most quantum software packages enable an automatic generation of these "measurements" circuits, i.e. appending the required gates at the end of the state preparation circuit. These are sometimes called *post-rotations*.

Bibliography

[1] S. Bravyi and A. Kitaev, "Universal quantum computation with ideal clifford gates and noisy ancillas," *Physical Review A*, vol. 71, no. 2, p. 022316, 2005.

[2] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Physical Review A*, vol. 86, no. 3, p. 032324, 2012.

[3] S. Chakrabarti, R. Krishnakumar, G. Mazzola, N. Stamatopoulos, S. Woerner, and W. J. Zeng, "A Threshold for Quantum Advantage in Derivative Pricing," *Quantum*, vol. 5, p. 463, June 2021.

[4] T. Häner, D. S. Steiger, K. Svore, and M. Troyer, "A software methodology for compiling quantum programs," *Quantum Science and Technology*, vol. 3, no. 2, p. 020501, 2018.

[5] C. M. Dawson and M. A. Nielsen, "The solovay-kitaev algorithm," *arXiv preprint quant-ph/0505030*, 2005.

Chapter 3

Quantum annealing

3.1 A quantum device for classical optimization

These notes will primarily focus on the digital version of quantum computation due to its power and potential for error correction. However, it's important to review quantum annealing and the capabilities of analog quantum devices for historical context and to establish useful concepts for the Chapter on real-time dynamics**??** and quantum optimization.

As per our definition in Chapter 1, a quantum annealer falls between an analog simulator and a NISQ computer. Although it isn't based on the gate model yet, it aims to perform computations beyond what could be considered a quantum many-body experiment(1).

Quantum annealing is a method for **solving combinatorial optimization problems**. These problems can be transformed into minimizing the energy of classical Ising-type Hamiltonians. By adding quantum mechanics, it has been suggested that the ground state of these frustrated Ising systems can be found faster than with any classical technique.(2)

The adiabatic principle is used, whereby a ground state of a quantum Hamiltonian is prepared and slowly changed. The system evolves following the instantaneous ground state of the perturbed Hamiltonian: if a suitable time-dependent quantum Hamiltonian is defined, connecting an initial Hamiltonian to the final classical Ising Hamiltonian, a quantum system undergoing adiabatic relaxation can reach its ground state and solve the optimization problem, at low temperatures.

The main experimental challenge of quantum annealing is the long runtime needed to adiabatically change the driving hamiltonian, that is currently much longer than coherence time. If thought as a coherent machine, the total run-time of the algorithm is well beyond the coherence limit imposed by hardware noise. However, one could also think about a scenario where incoherent tunneling events may provide a computational advantage.

Early numerical studies predicted quantum annealing would be a competitive computational resource for solving spin glass problems (3). Simulated annealing, a classical counterpart, is the most powerful heuristic solver for generic combinatorial problems. The slowly varying parameter in this case is the effective temperature of a Markov-chain simulation aimed to sample the Boltzmann distribution generated by the classical Ising model. However, experiments have not yet observed quantum speedup. The difference between the two methods lies in the type of fluctuations that drive the system away from multiple local min-

ima occurring in rugged energy landscapes. Quantum fluctuations are expected to give an advantage to quantum annealing, particularly when the free energy landscape displays tall but narrow barriers, which are easier to tunnel through quantum-mechanically. After years of research, it has been shown that such tall but thin barriers occur only in carefully crafted toy models, not in realistic problem instances such as Ising glasses in two or three dimensions. Tough this remains an open challenge.

Challenging optimization problems include (i) determining the optimal (ground-state) energy arrangement of a group of L classical Ising spins with frustrated interactions[1]: the **spin glass**, (ii) determining the shortest possible route for a travelling salesman visiting L cities (known as the travelling salesman problem, or TSP), or (iii) identifying a bit assignment that satisfies an L-bit Boolean formula with numerous logical clauses, each containing k bits (known as k-SAT), known as satisfiability problem.

Many other types of optimization problems exist, however, the common theme is that the configuration space of the L input variables increases exponentially with L, reaching 2^L in the case of Ising and k-SAT, and $L!$ in TSP. The total computational space is too vast for a brute force approach beyond a few tens of sites.

3.2 A bit of complexity theory

In this subsection we provide a simple overview of complexity classes that may be useful also for the rest of the notes. The exposition here is colloquial, as more rigorous definitions exists. The runtime (the worst-case scenario) determine the complexity class of a problem. If we consider classical computation, very common classes are P and NP:

- a problem belongs to the P class if a polynomial (in L) algorithm can solve the worst-case instances. These problems are considered *easy* from the complexity perspective, although the order of the polynomial function can make a great difference in practice.
- The NP class include computational **decision** problems for which any given "yes"-solution can be verified in polynomial time by a deterministic machine. Notice that optimization problems are not stricly speaking decision problems, as their output is a function value. They can be recasted as decision problems if we include also a threeshold value, i.e. E_c. The corresponding verifiable statement, in poly time, is wether the found solution E_{found} is lower than E_c. In this sense, also optimization problem can be included as member of complexity classes.
- NP-hard problems include classes of problems which are at least as hard as the hardest problems in NP. It may sound strange but the NP-hard class is not fully included in NP. NP-hard problems which are in NP, are called NP-complete. All NP-complete problems are also NP-hard, but the opposite is not true. The examples above, in their most general formulation, belong to the category of NP-complete problems, which include the most difficult classes of all NP problems)[2]

[1] i.e. both ferromagnetic and antiferromagnetic with no regular structure

[2] exceptions are the Ising model on certain two-dimensional lattices or the 2-SAT problem

Notice again that this labeling is valid only if we consider the worst-case scenario. Think about the classical ferromagnetic model: here it is easy to guess the ground state, yet it is a special instance of an Ising spin-glass Hamiltonian.

There are complexity classes appropriate also for quantum computation:

- BQP, bounded-error quantum polynomial time, is the class of decision problems solvable by a quantum computer in polynomial time, with an error probability of at most 1/3 for all instances. Roughly speaking is the quantum generalization of P.[3].

  However, the BQP class contains P. We have at least one example of problems, the factorization of large integers which is *not* in P but it belongs to BQP thanks to Shor's algorithm.

  Unfortunately, it is believed that BQP does *not* contains NP. This means that not even with a quantum computer we can exactly solve -with exponential speed-up- optimization problems in this class.

- QMA, quantum Merlin-Arthur, class of problems is the generalization of NP. It includes class of problems which "yes"-solution can be verified in polynomial time by a quantum computer. In physics, the most famous example of such a problems is deciding whether the ground state energy of a given k-local Hamiltonian is smaller than a given value E_c.[4] It has been proven that $k = 2$ is already enough to obtain a QMA-complete problem(4).

The Venn diagram illustrating the mentioned complexity classes (many others exist!) is in Fig. 3.1.

3.2.1 Approximate solutions

From what has been discussed above, the prospects for a quantum advance in optimization problems seem limited, at least in terms of scaling. An argument (not rigorous) that could negate a potential exponential advantage is the fact that search algorithms (see Chapter **??**), such as Grover's, offer at most a quadratic speedup over brute force search. Brute force search appears to be the only method capable of guaranteeing the exact solution's discovery.

However, it should be noted that for problems with a certain structure, there may exist better methods than brute force search and correspondingly better quantum methods. It is important to remember that everything mentioned above applies when seeking an exact solution. In practice, though, sometimes one can settle for a good solution, even if it's not the true global minimum. Something good is still better than nothing.

In such cases, one can turn to approximation algorithms, which aim to find these near-optimal solutions. These solutions must provide provable guarantees regarding their closeness to the optimal solution. But how can we quantitatively define if a solution is near-optimal?

In this case, the approximation ratio can be used, which is the ratio between the returned solution and the optimal one. An approximation algorithm can thus provide the following

[3] or, better, its probabilistic version BPP, bounded-error probabilistic polynomial time, which is the class of decision problems solvable by a probabilistic machine in polynomial time with an error probability bounded by 1/3 for all instances. However, this makes a little difference as it is widely believed that BPP=P.

[4] the actual decision problem is slighly more complicated.

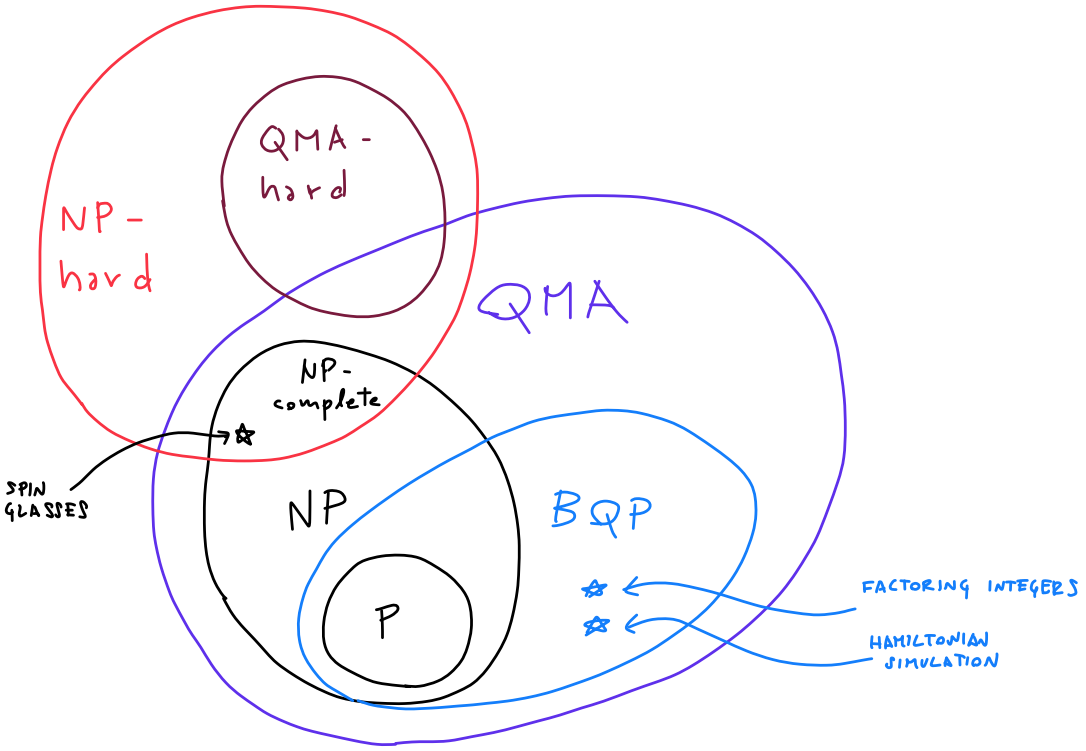


Figure 3.1: Conjectured relationships between the classical and quantum computational complexity classes.

guarantee: it provides a solution that cannot be worse than a given approximation ratio. This defines the APX class: the set of NP optimization problems that allow polynomial-time approximation algorithms with an approximation ratio bounded by a constant.

As you can see, the world of optimization is much more nuanced than one might expect, particularly when delving into the realm of approximate algorithms. It is possible that an exponential quantum advantage could reemerge in this case.

3.3 Spin glasses

Ising spin glasses on general graphs (5) are NP-hard optimization problems, where no efficient (i.e. polynomial time) algorithm exist. Therefore the resources needed to exactly solve spin glasses must scale exponentially with problem size as $\sim 2^{kL}$. Many optimization problems can be cast into a Ising-like form, using the Quadratic unconstrained binary optimization (QUBO) formalism.

Since random ensembles of hard problems are closely connected to spin glass models, we can choose the problem Hamiltonian we want to solve to be an Ising spin glass

$$H_P = \sum_{i,j} J_{ij} \sigma_i^z \sigma_j^z \ , \tag{3.1}$$

where σ_i^z are Pauli matrices, acting on spins i, and J_{ij} is the coupling between spins i and j, which value is set by the specific instance. For instance, a well definite airline routing problem or traveling salesman instance will realize a given instance of this spin-glass hamiltonian, with a well defined connectivity and coupling values. Notice that, in principle, higher-order operators like $\sigma_i\sigma_j\sigma_k$ could be allowed, but in practice only two-spin interaction can be implemented in real hardwares, such as *D-WAVE* machine. For this reason, the QUBO formalism is in general the first step needed to map a user-defined problem into a spin-hamiltonian that can be realized in hardware.

Quantum advantage. The adiabatic annealing method presented below is an exact algorithm for solving sping glasses[5] Despite early speculative claim of possible exponential advantage, quantum algorithms are not expected to turn the exponential scaling into a polynomial one, the exponent k might be smaller, thus potentially realizing a substantial polynomial speed-up over classical algorithms (see above). All in all, the *holy-grail* of quantum annealing is discovering **scaling advantage** for classes of optimization problems which have real applications, i.e. beyond carefully crafted artificial models(6). Morevover, it is entirely possible that the average spin glass problem can be solved in polynomial time, even though the worst case may be exponential.

3.4 The adiabatic principle perspective

As mentioned the initial theoretical bases behind quantum annealing is the adiabatic theorem: if a quantum system undergoing evolution by a slowly time-dependent Hamiltonian $H(t)$ closely follows the instantaneous eigenstate in its time evolution, then by starting the dynamics in one of the ground state of the initial Hamiltonian, we will reach the ground state of the final one.[6]

In a QA machine $H(t=0)$ is dominated by a pure quantum fluctuation part H_Q whereas the final Hamiltonian $H(t_{\text{final}})$ encodes only the cost function H_P of the combinatorial optimization problem. The Hamiltonian as a function of the time t may read, in the case of simple linear annealing schedules (see Fig. 3.2)

$$H(t) = sH_P + (1-s)\ H_Q, \tag{3.2}$$

where $s = t/\tau$, and τ is the total runtime of the process. The driver H_Q operator is a transverse-field (TF) Hamiltonian

$$H_Q = -\Gamma \sum_i \sigma_i^x\ , \tag{3.3}$$

where the σ_i^x operator acts locally on the spin index i inducing quantum fluctuations.

In a close-system quantum dynamics picture, the quantum state obeys a time-dependent Schoedinger equation

$$i\frac{d}{dt}\left|\psi(t)\right\rangle = H(t)\left|\psi(t)\right\rangle, \tag{3.4}$$

[5] being exact does not implies that is going also to be efficient.

[6] Readers familiar with condensed matter may recall the use of the adiabatic theorem to explain the similarity between non-interacting and interacting Fermi liquids. In that case, the adiabatic theorem is only used to derive the theory. In this case, there is a real harwdare, implementing physical couplings J_{ij} and a real trandverse field Γ which vary with time.

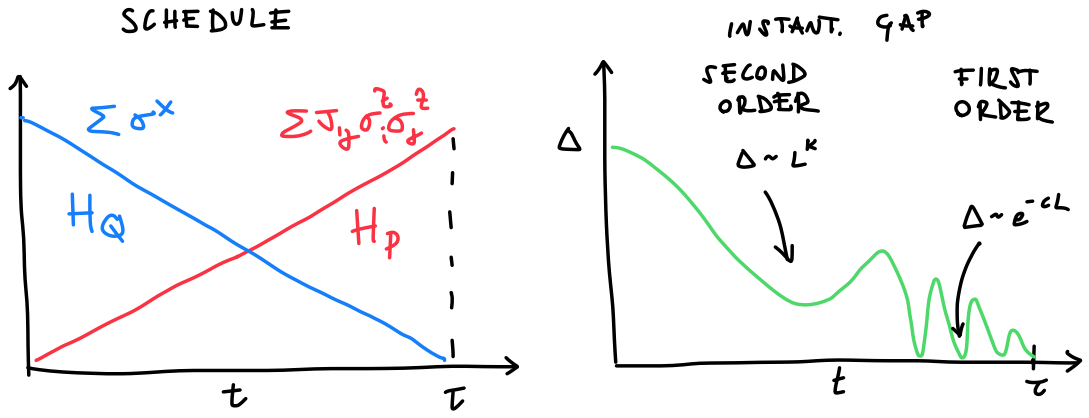


Figure 3.2: Left: Annealing schedule. The quantum fluctuation term is lowered with time while the problem Hamiltonian is increased linearly, as in Eq. (3.2). Right: a typical evolution of the minimum gap (between the ground state and the first excited state of $H(t/\tau)$) during the annealing. In spin glasses, we first encounter a second order quantum phase transition, where the gap closes polynomially with L, followed by the glassy region, where the gap closes exponentially. These are the bottleneck of QA, as here the annealing must proceed very slowly. Obviously, we don't know at which parameter s the hamiltonian will have the minimum gap, so we cannot adjust beforehand the schedule accordingly.

.

where $t \in (0, \tau)$. The total time of annealing is a free parameter of the simulation, though the adiabatic principle suggests to set a large τ. The adiabatic theorem is a fundamental principle that provides a powerful foundation for various applications. However, it also imposes restrictions on the speed of the process. Specifically, the duration of the process must be proportional to the inverse of the minimum energy gap squared, namely by the smallest energy gap Δ encountered during annealing and is solely dependent on the instantaneous Hamiltonian, $H(s)$.

$$\tau \sim 1/\Delta^2, \quad \text{with } \Delta = \Delta(L) \tag{3.5}$$

This is where QA approaches the realm of quantum statistical mechanics: the evolution must traverse a quantum phase transition, because the final (eigenstate of H_P) and initial state (eigenstate of H_Q) are qualitatively different. Therefore the gap closes with the system's size L (in the thermodynamic limit).

Effective QA can be achieved by utilizing problem Hamiltonians that possess a gap in their energy spectrum which does not close exponentially fast with L, i.e $\Delta \sim e^{-L}$ (like in first-order transitions), because in this case the total runtime must increase exponentially with L. If instead the gap closes polynomially (like in continuos phase transtitions), the total runtime is also polynomial with L. Therefore the size dependence of the gap and the type of transition directly determines the computational complexity of QA.

While it is not possible to determine a priori the minimum gap, or the type of transition for every $H(s)$, theory and numerical simulations indicates that it closes exponentially with L in quantum spin glasses at low Γ,(7) as well as for the ferromagnetic Ising model ((see

Fig. 3.2)).

Given that QUBO Hamiltonians do not match exactly the connectivity of the hardware[7], **embedding** is often needed. For instance, real problem Hamiltonians may require *all-to-all* connectivity while the hardware only feature sparse connectivity. Embedding results in the usage of extra qubits, which interact with each others ferromagnetically(8). The resulting embedded, QUBO hamiltonian therefore may feature additional strings of ferromagnetically coupled qubits.

3.4.1 Practical metrics for measuring QA success

The adiabatic annealing method is exact in principle, but it require exponentially long runtimes τ. Practical implementations can only allow for a finite τ, so QA becomes an heuristic method. After some years of usage, it is becoming clear that it is better to allocate the available QPU using multiple runs, instead of performing one single run, but with the longest possible τ (this is due to the limited coherence time of the available hardware, see below).

There are two key metrics, which are generally related, to assess the performance of QA. The first is the residual energy $E_{\text{res}}(\tau) = E(\tau) - E_0$, which decreases for increasing annealing time τ , and where E_0 is the exact minimum of H_P, and $E(\tau)$ is the average energy produced by quantum system after the evolution, $\langle\psi(\tau)| H_P |\psi(\tau)\rangle$.

In a practical setting, the average value can be taken by considering R repetitions of the QA run. Additionally, if one wants to benchmark QA as a solver for a class of problems, i.e. the 2D disordered Ising model, one could perform an overall average over several representative instances.

The residual energy may decrease polynomially with τ, i.e $E_{\text{res}}(\tau) \sim \tau^{-c}$ or logaritmically, i.e. $E_{\text{res}}(\tau) \sim (\log \tau)^{-c}$. In the second case, which is typical of spin-glasses, the problem class is *hard*.

Another commonly used benchmark is to calculate the *success probability*. In this case, we don't aim for the energy but for the configuration, a more stringent requirement. The empirical success probability p can be defined as the number of runs, at fixed τ, that achieve the global minima over R repetitions , $p = \text{success count}/R$. From this, one could measure the number of repetitions to observe at least once the solution with a desired probability p_{target}, for instance 0.99 as

$$R_{\text{target}} = R_{\text{target}}(\tau) = \frac{\log(1 - p_{\text{target}})}{\log(1 - p(\tau))} \tag{3.6}$$

. Therefore the time-to-solution (TTS) of QA, assuming a desired p_{target} is simply

$$TTS(\tau) = \tau \; R_{\text{target}}(\tau), \tag{3.7}$$

since each repetition lasts τ. Empirically we see that there exists a trade-off between running a single, long run, relying fully on the adiabatic theorem and performing several R_{target} trials with a shorter τ. There exist an optimal combination that make uses of the quantum processing time (QPU) of the machine in the best way. Performing several set-up with different τ, allows one to identify the optimal TTS for a given problem size.

[7] for instance D-Wave machine implement a chimera graph connectivity

This optimal TTS should then be plotted against L to rigorously find the runtime scaling. At a first glance, this could seem a quite a pedantic post-processing step, however, if done incorrectly, it may lead to qualitatively wrong claim of quantum speedup (see e.g. the discussions in Refs. (9; 10)) For this reason, we explicitely lay down these formula in this Section.

3.5 The incoherent tunneling perspective

Most recent results (2022) indicate that new generation of quantum annealers have coherence time of order of 100 ns. However, the tipical runtime ar of order of tens of microsecond(11).

It is clear therefore that the system interacts with the enviroment, and the ideal assumption of a close-system dynamics obeying the adiabatic principle is not a faithful representation of the process.

Quantum annealing goes beyond the adiabatic computation framework. Indeed, an alternative perspective can be taken where the machine is understood as to be in instantaneous equilibrium at the Hamiltonian $H(s)$ and with the environment at any given time s. The coherence of the system can easily be disrupted by external noise, resulting in incoherent quantum tunnelling becoming the primary computational resource.

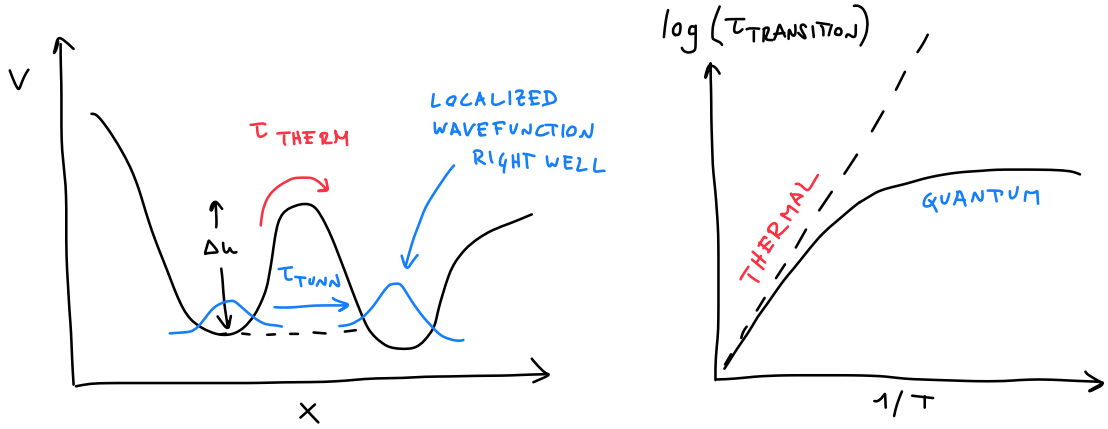


Figure 3.3: Left: Energy barrier with height ΔU. In the quantum tunneling picture, a localized quantum state undergoes quantum tunneling into a new localized state. Right: Transition time (logscale) as a function of the inverse temperature. At high temperature, classical regime, the transition time follows the inverse Kramers rate featuring an explicitic temperature dependence. At low temperature, transition are driven by quantum fluctuations and are independent of T. In this regime, quantum tunneling is more effective in overcoming energy barriers than thermal excitations.

.

When considering a double-well model, the thermal transition rate scales exponentially with the barrier height $k_{\text{thermal}} \sim e^{\Delta U/T}$, whereas the tunnelling rate, $1/\tau_{\text{tunn}}$, scales exponentially with the area beneath the barrier, and is independent from the temperature T. This leads to the popular notion that quantum annealing is advantageous for energy landscapes

with tall yet narrow barriers (cfn. Fig. 3.3).

Interestingly, it is possible to derive a theory for the incoherent tunneling rate in double-wells models, showing that

$$\tau_{\text{tunn}} \sim 1/\Delta^2, \tag{3.8}$$

where, again, Δ is the energy gap of the quantum hamiltonian of the double-well model(12). Strikingly, if we assume that the the annealing process occurs as a sequence of uncorrelated, incoherent tunneling events, the runtime of the annealing is dominanted by the longest tunneling timescale, i.e. the one which smallest gap Δ. This runtime scaling is therefore consistent with Eq. 3.5.

3.5.1 Quantum inspired classical algorithms

In general, simulating real-time quantum dynamics requires the direct integration of the time dependent Schr$\acute{}$odinger equation. This is a formidable task as the Hilbert space of the systems grows exponentially with the number of constituents, and is the one of the reason why quantum computers are built in first place. Simulating the microscopic non-equilibrium dynamics inside a quantum annealer, or any other noisy quantum device, is even mode difficult as we don't know exacly how to model the hardware noise.

However, the characterization of quantum dynamics simplifies when it is dominated by tunneling events. While, quantum dynamics can not be accessed efficiently by classical algorithms, equilibrium properties of certain quantum hamiltonian can be simulated efficiently.

Quantum Monte Carlo (QMC) techniques are based on the Feynman path integral formulation of quantum mechanics and are used to simulate the thermodynamic equilibrium of Hamiltonians that do not have the sign problem.[8] An example of such Hamiltonians is the transverse field Ising Hamiltonian.

Practically speaking, QMC is a class of classical Monte Carlo algorithms that uses Metropolis sampling on a $(d+1)$-dimensional lattice model, where the additional dimension is known as the imaginary time axis. Although QMC and quantum tunneling events are fundamentally different, it has been shown that algorithmic tunneling-like Metropolis updates are needed to overcome energy barriers, just like in real incoherent tunneling events. Moreover, the time required by QMC to simulate quantum mechanical tunneling scales identically, in leading exponential order, with the problem size to the tunneling rate of a physical system ($\sim 1/\Delta^2$)(13).

From an algorithmic standpoint, QMC tunneling events are needed to sample the configuration space, so the probability of finding the local minima is proportional to the QMC tunneling rate, which is related to the exploration of the computational space. In this sense, if QMC is considered as a solver, the classical QMC runtime should theoretically scale as Δ^2 when performing a sequence of equilibrium simulations of Hamiltonians at decreasing Gamma values. This context is often referred to as Simulate Quantum Annealing (SQA).(3)

The existence of competing QMC simulations, which show the same scaling as QA, has raised the bar for quantum advantage. Quantum Annealing not only needs to outperform Simulate Annealing but also QMC, a quantum-inspired but still very classical solver. Experimental results have confirmed this scenario. Although it has been possible to devise toy

[8] The sign problem-free Hamiltonians are those for which efficient QMC simulations are possible.

models in which QA outperforms SA, it has not been possible to outperform SQA in terms of scaling.

Furthermore, classical simulations of SQA can be performed with arbitrary setups. It has also been demonstrated that running SQA with an unconverged setup, i.e., using a small number of imaginary-time Trotter steps or adopting unphysical open-boundary conditions in imaginary time, further improves the scaling.

The positive side of this comparison is that it is possible to see that, despite the noise, quantum annealers are not thermal machines. The instances for which QA is successful correlate better with the ones where also SQA is, rather than SA, see e.g. Ref. (14).

In conclusion, QMC is part of a family of classical algorithms that **mimic quantum mechanics but run on conventional hardware**. This family is today referred to as quantum-inspired algorithms.

Bibliography

[1] M. W. Johnson, M. H. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, *et al.*, "Quantum annealing with manufactured spins," *Nature*, vol. 473, no. 7346, pp. 194–198, 2011.

[2] T. Albash and D. A. Lidar, "Adiabatic quantum computation," *Reviews of Modern Physics*, vol. 90, no. 1, p. 015002, 2018.

[3] G. E. Santoro, R. Martoňák, E. Tosatti, and R. Car, "Theory of quantum annealing of an ising spin glass," *Science*, vol. 295, no. 5564, pp. 2427–2430, 2002.

[4] J. Kempe, A. Kitaev, and O. Regev, "The complexity of the local hamiltonian problem," *Siam journal on computing*, vol. 35, no. 5, pp. 1070–1097, 2006.

[5] F. Barahona, "On the computational complexity of Ising spin glass models," vol. 15, no. 10, p. 3241, 1982.

[6] V. S. Denchev, S. Boixo, S. V. Isakov, N. Ding, R. Babbush, V. Smelyanskiy, J. Martinis, and H. Neven, "What is the computational value of finite-range tunneling?," *Physical Review X*, vol. 6, no. 3, p. 031015, 2016.

[7] S. Knysh, "Zero-temperature quantum annealing bottlenecks in the spin-glass phase," *Nature communications*, vol. 7, no. 1, p. 12370, 2016.

[8] M. S. Könz, W. Lechner, H. G. Katzgraber, and M. Troyer, "Embedding overhead scaling of optimization problems in quantum annealing," *PRX Quantum*, vol. 2, p. 040322, Nov 2021.

[9] T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar, and M. Troyer, "Defining and detecting quantum speedup," *science*, vol. 345, no. 6195, pp. 420–424, 2014.

[10] T. Albash and D. A. Lidar, "Demonstration of a scaling advantage for a quantum annealer over simulated annealing," *Phys. Rev. X*, vol. 8, p. 031016, Jul 2018.

[11] A. D. King, S. Suzuki, J. Raymond, A. Zucca, T. Lanting, F. Altomare, A. J. Berkley, S. Ejtemaee, E. Hoskinson, S. Huang, *et al.*, "Coherent quantum annealing in a programmable 2000-qubit ising chain," *arXiv preprint arXiv:2202.05847*, 2022.

[12] U. Weiss, H. Grabert, P. Hänggi, and P. Riseborough, "Incoherent tunneling in a double well," *Physical Review B*, vol. 35, no. 18, p. 9535, 1987.

[13] S. V. Isakov, G. Mazzola, V. N. Smelyanskiy, Z. Jiang, S. Boixo, H. Neven, and M. Troyer, "Understanding quantum tunneling through quantum monte carlo simulations," *Physical Review Letters*, vol. 117, no. 18, p. 180402, 2016.

[14] S. Boixo, T. F. Rønnow, S. V. Isakov, Z. Wang, D. Wecker, D. A. Lidar, J. M. Martinis, and M. Troyer, "Evidence for quantum annealing with more than one hundred qubits," *Nature physics*, vol. 10, no. 3, pp. 218–224, 2014.