Advanced Quantum Algorithms

Lecture notes

Guglielmo Mazzola

Fall semester 2023

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.

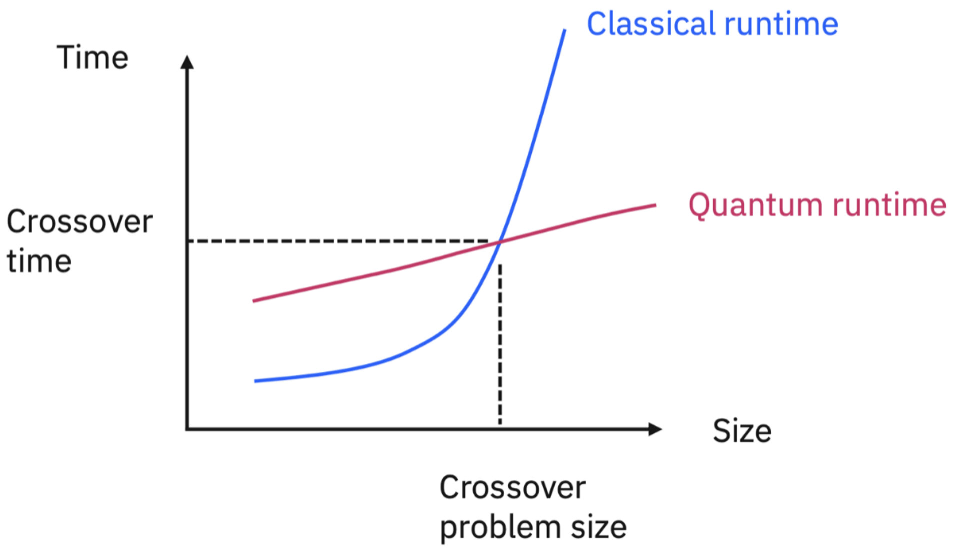


Figure 1.1: Cartoon of quantum and classical runtimes for a generic situation where a quantum algorithm has scaling advantage over its classical counterpart (e.g. it may be a the task of time-evolving a quantum state, under the action of a lattice hamiltonian, like Hubbard one.) For small system size, 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.

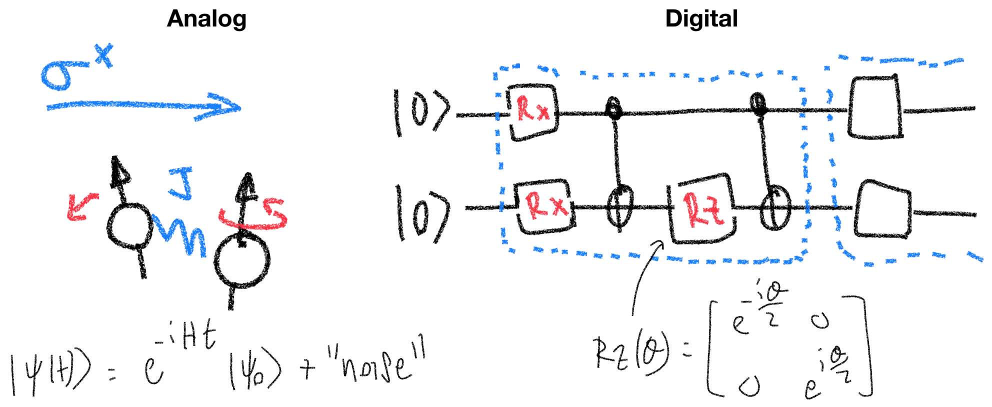

Analog
Digital
σ^x
J
$|\psi(t)\rangle = e^{-iHt}|\psi_0\rangle + \text{"noise"}$
$|0\rangle$
$|0\rangle$
Rx
Rx
Rz
$Rz(\theta) = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}$

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 4). 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 4). 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 4.

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

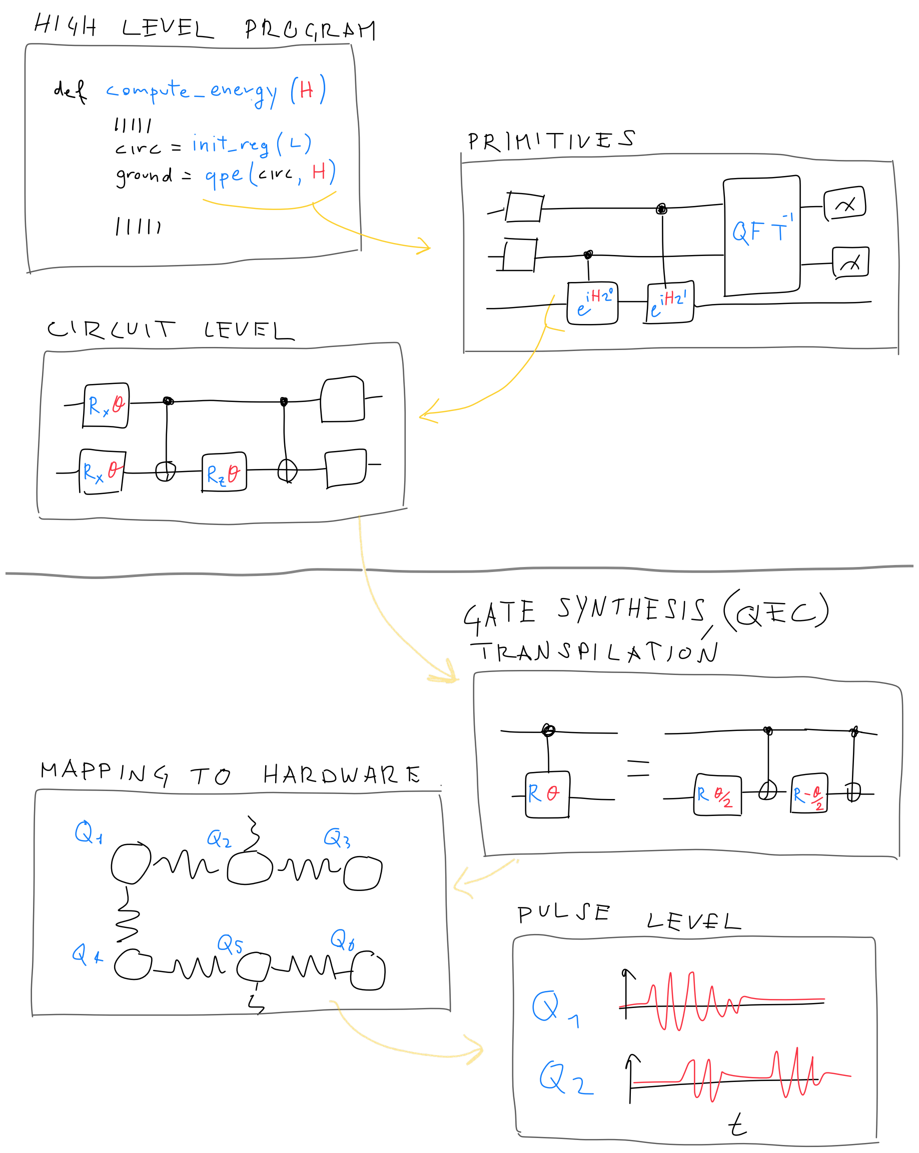


Figure 1.3: The several layers of a quantum program, from high-level human-readable instruction, to transpilation into the available gate-set, to pulse-level control of the physical qubits.
.

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 superconducting 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

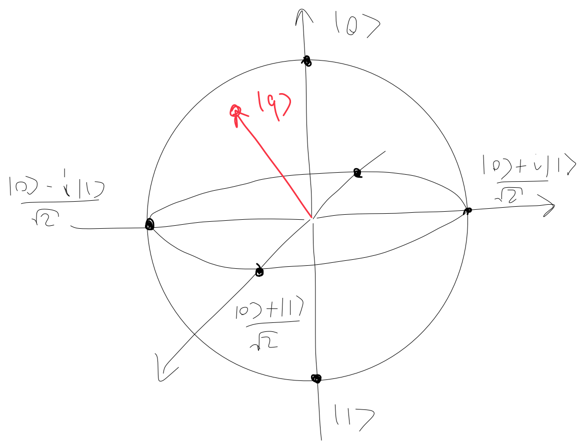

$|0\rangle$
$|q\rangle$
$\frac{|0\rangle - i|1\rangle}{\sqrt{2}}$
$\frac{|0\rangle + i|1\rangle}{\sqrt{2}}$
$\frac{|0\rangle + |1\rangle}{\sqrt{2}}$
$|1\rangle$

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.,

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

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

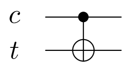
Since one-qubit gates cannot produce entangled states, as by definition a product state remains a product state under the action of a tensor product of one-qubit gates, we need to introduce at least one genuine two-qubit gate to perform quantum computation.

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.

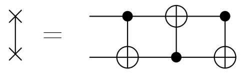
Another important two qubit gate is the **SWAP gate**. The operation it performs is the following

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

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

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

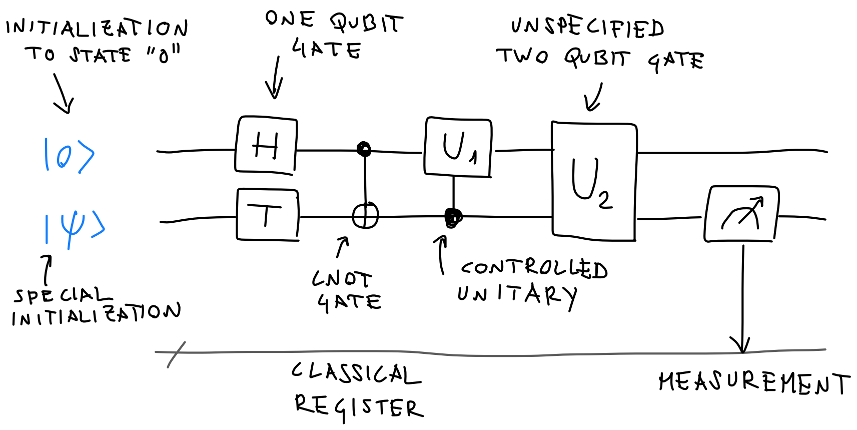


Figure 2.2: A generic quantum circuit featuring a two-qubit register and a classical register. The circuit is made of two single-qubit gates, a CNOT gate, a controlled unitary gate (C-U_1), where the second qubit controls the first, and a generic two-qubit operation U_2. Finally the second qubit is measured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

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

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 $O^\dagger$.

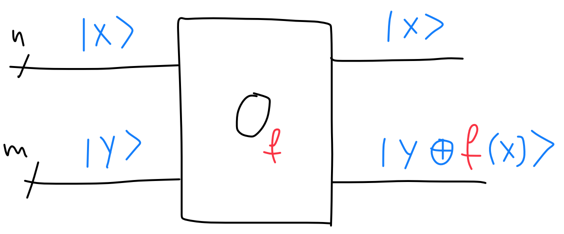


Figure 2.3: A generic circuit drawing of a Boolean oracle, O_f implementing the classical function f.

.

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

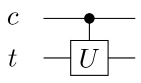
After these basic definitions we are in the position to define circuit identities and general controlled gates. Moreover, we can revisit the CNOT under this "oracular" perspective, which may be useful.

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]

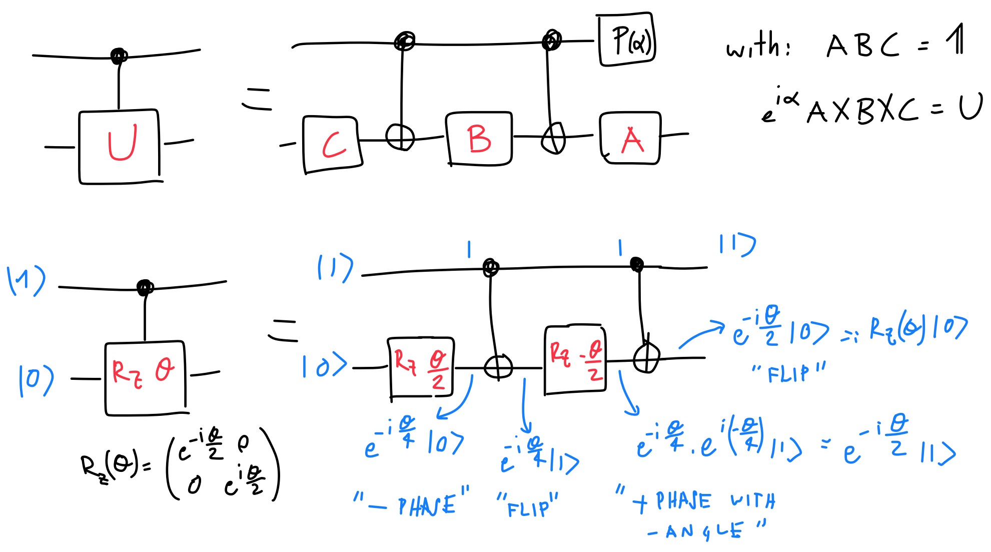


Figure 2.4: Upper circuit: circuit equivalence for a generic single qubit controlled-U gate. Lower circuit: particular example of a $C - R_z$ rotation, with all intermediate states, for the input state $|1\rangle \otimes |0\rangle$, that is mapped to the $|1\rangle \otimes R_z(\theta)\,|0\rangle$. The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

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

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. If we suppose that the systems is also made of two-qubits

$$\langle\psi|\, ZZ\, |\psi\rangle = \begin{bmatrix}\psi_{00}^* & \psi_{01}^* & \psi_{10}^* & \psi_{11}^*\end{bmatrix} \begin{bmatrix}1 & 0 & 0 & \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1\end{bmatrix} \begin{bmatrix}\psi_{00} \\ \psi_{01} \\ \psi_{10} \\ \psi_{11}\end{bmatrix} = \tag{2.44}$$

$$= |\psi_{00}|^2 - |\psi_{01}|^2 - |\psi_{10}|^2 + |\psi_{11}|^2 = \tag{2.45}$$

$$= \mathrm{Prob}(\sigma = 00) - \mathrm{Prob}(\sigma = 01) - \mathrm{Prob}(\sigma = 10) + \mathrm{Prob}(\sigma = 11), \tag{2.46}$$

(2.46)

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

$$P = U^{\dagger} P^{\mathrm{diag}} U, \tag{2.47}$$

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 $P^{\mathrm{diag}} = ZZ$. Similarly, the operator $XIYY$, can be measured using $U = \mathsf{H} \otimes I \otimes K \otimes K$ and $P^{\mathrm{diag}} = ZIZZ$.

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 dynamics4 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 minima 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 7), 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.

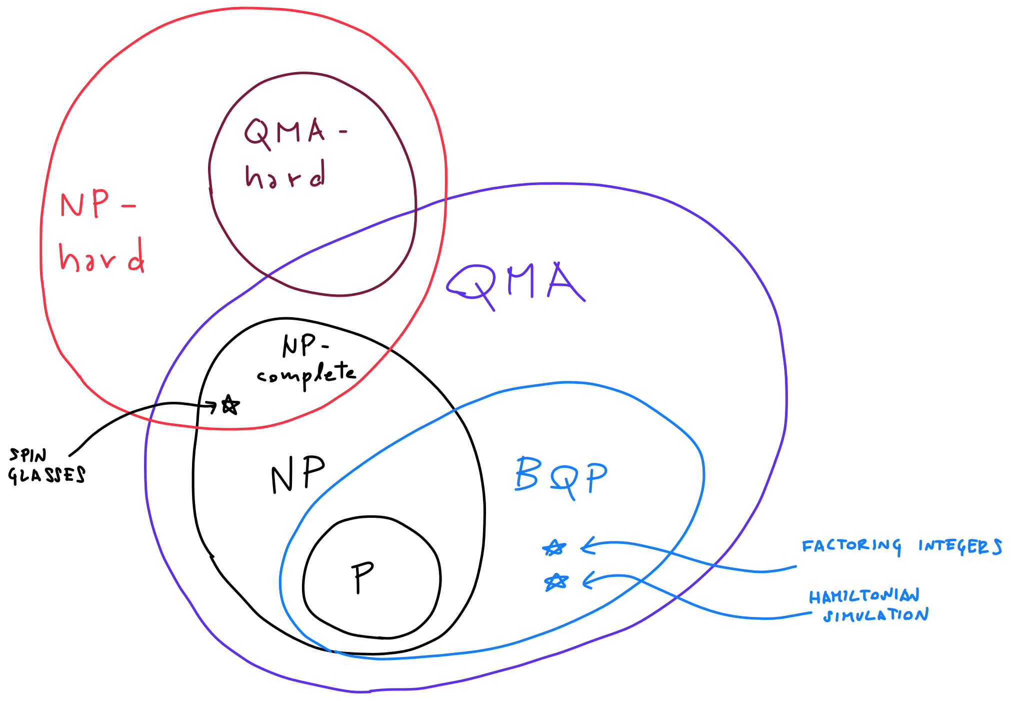


Figure 3.1: Conjectured relationships between the classical and quantum computational complexity classes.

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.

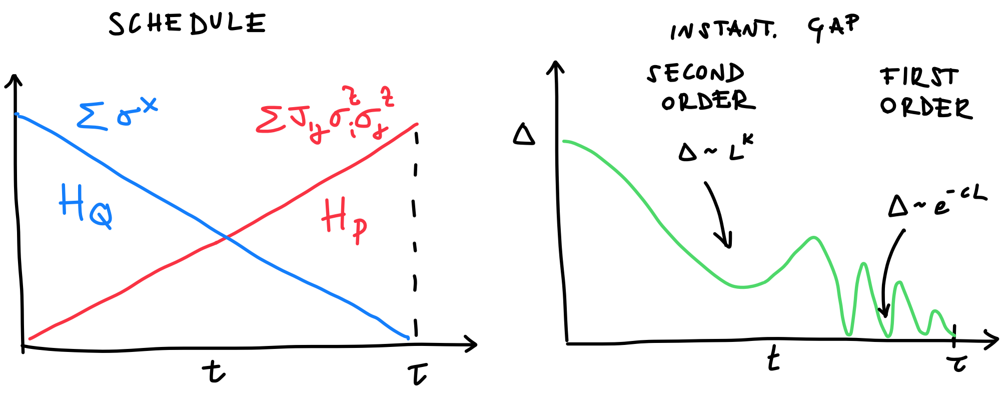


Figure 3.2: Left: Annealing schedule. The quantum fluctuation term is lowered with time while the problem Hamiltonian is increased linearly, as in Eq. (3.2). Right: a typical evolution of the minimum gap (between the ground state and the first excited state of $H(t/\tau)$) during the annealing. In spin 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[8] to observe at least once the solution with a desired probability p_{target}, for instance 0.99 as

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

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

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

[8]The formula can be obtained as follows. Given $p(\tau)$ the empirical probability of finding the solution during a run of lenght τ, the probability of still *not* observing the solution after R repetitions is $1-p_{\text{target}} = (1-p(\tau))^R$

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.

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.

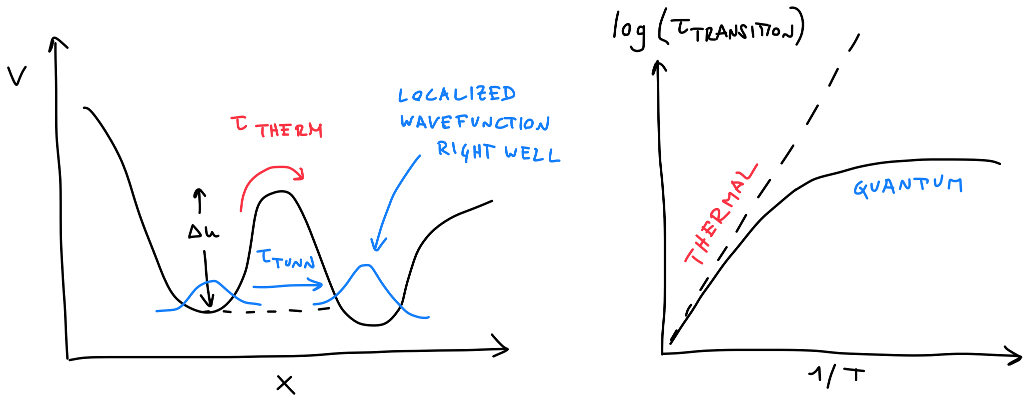


Figure 3.3: Left: Energy barrier with height ΔU. In the quantum tunneling picture, a localized quantum state undergoes quantum tunneling into a new localized state. Right: Transition time (logscale) as a function of the inverse temperature. At high temperature, classical regime, the transition time follows 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´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.[9] 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

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

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 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.

Chapter 4

Hamiltonian simulation

4.1 Motivation and quantum advantage

Hamiltonian simulation, also known as quantum dynamics, is a fundamental application of digital quantum computers. Formally, the task is to solve the Schroedinger equation

$$i\frac{d}{dt}\left|\psi(t)\right\rangle = H(t)\left|\psi(t)\right\rangle, \tag{4.1}$$

starting from a given initial state $|\psi(t=0)\rangle$, and where the Hamiltonian may be or not be explicitely time-dependent, $H(t)$. Clearly even if the Hamiltonian is time-independent, we will observe a non-trivial evolution if the starting state $|\psi(t=0)\rangle$ is not an eigenstate of H. In "real life" this is typically the case. Experiments may probe a quantum system by perturbing it, so even if the system was originally in a ground-state of some H_{init}, it is not anymore. If the perturbation is fast-enough, i.e. the opposite case of an adiabatic evolution presented before, then the system develops a non-trivial evolution. This case if often called, a quantum quench[1]. The study of non-equilibrium dynamics is a the center of many important disciplines, such as nano electronics, because of the possibility of photo-induced metal-to-insulator and magnetic transitions, but also to understand "pump-probe" experiments and the rich physics of non-equilibrium strongly correlated systems. On a more theoretical side, such algorithms are essential to address closed-system thermalization, i.e. how and when thermalization occurs as a result of the unitary time evolution of a *closed* quantum system. Going beyond condensed matter, the unitary dynamics of quantum systems made of electrons and nuclear wavepackets is needed to model certain types of chemical reactions, such as photochemistry. In this case, the Born-Oppenheimer approximation breaks down, and one needs to consider the dynamical interplay between electrons moving on (superpositions of) several potential energy surfaces. Finally, there is a large demand for efficient solvers in high-energy physics, to simulate the evolution of lattice gauge theory models.

Coming back to quantum computers, here the goal is to perform unitary operations on a quantum computer that match the evolution of a quantum system. The reason why hamiltonian dynamics is computationally challenging is also what makes quantum computers so powerful. While Feynman (1982) initially proposed using a quantum computer as the natural way to simulate quantum mechanics[2], it wasn't until more than a decade later that

Lloyd (1996) introduced the idea of a universal quantum simulator, which could practically perform Hamiltonian simulation[3]

The good news is that the simulation of physical Hamiltonians, which have k-local interactions, can be achieved with polynomial complexity in both space (number of qubits) and time. This kind of problems is in the BPQ complexity class, as defined in the previous Chapter. These Hamiltonians include electronic structure and condensed matter problems that involve pairwise electron-electron interactions, which, for instance in a second-quantized formalism, only require four-index operators. In contrast, all known classical methods[1] for these problems have exponential time costs that grow with system size. During the dynamics, even if we start with a localized quantum state, we will visit an exponential number of states due to the spreading of correlations out of equilibrium, such as the *light-cone* effect after quantum quenches.

The scope of hamiltionian simulations goes beyond the realm of direct quantum dynamics of physical systems. *(*i) Performing blocks of e^{iHt} evolutions will prove also useful to get *ground state* properties of H. *(ii)* Going beyond quantum-many body physics, hamiltonian simulation is a building block of other algorithms, i.e. for optimization (i.e. QAOA) and sampling of classical cost functions, up to solving linear systems.

4.1.1 Classical competitors for physics applications

Classical simulation of quantum dynamics is primarily limited by memory. As soon as we reach a number of quantum constituents larger than what can be simulated classically (around 50 qubits), we achieve the quantum advantage regime. Storing a quantum state of 50 qubits with double-precision coefficients for each of the 2^{50} possible components requires 16 PB of memory. To perform arbitrary discrete-time evolution, we would need to manipulate such an array thousands of times. For instance, direct matrix exponentiation e^{iHt} for a many-body quantum Hamiltonian for 50 spin-1/2 particles would require, an array made of 10^{15} entries undegoing a matrix-vector multiplication of size $2^{50} \times 2^{50} = 10^{30}$.

Approximate classical methods rely on compressing information into polynomially scaling control variables, which are efficient for ground state properties but less so for dynamics. These methods are difficult to benchmark because of the absence of a variational energy principle.

Several methods have been proposed to extend the power of approximate ground state approaches, with one of the most notable being tensor-networks. These methods are particularly effective for dynamics that occur near the ground state. In its simplest form, tensor-networks begin with a quantum state in a matrix product state (MPS) format, where the evolution operator can be expressed as a matrix product operator. To compute the action of one step of evolution on the MPS, a sequence of operators is applied to produce an MPS with a larger bond dimension, which is then truncated to maintain a fixed dimension. This process is repeated for the required number of steps to calculate expectation values over the evolved state.

Real-time variational Monte Carlo and real-time neural-network state methods function similarly, updating a number of variational parameters θ specifying the ansatz $|\psi(\theta(t))\rangle$ dur-

[1]methods with controlled approximations.

ing discrete-time iteration to satisfy the Dirac-Frenkel or McLachlan variational principle. These methods ensure that the complexity of the calculations remains unchanged over time, by maintaining the flexibility of the variational form constant. However, accuracy inevitably degrades as time increases. Notable examples of these methods include simulations of an 8x8 transverse field Ising model plaquette for thousands of time slices using neural-networks(4), and a 4x4x4 lattice quantum electrodynamics simulation with tensor-networks(5). Other statistical methods include diagrammatic real-time quantum Monte Carlo, where one samples, at each time step, configurations to be evolved.

For condensed matter, strongly-correlated systems, another well-established classical method is represented by the nonequilibrium extension of the dynamical mean field theory (DMFT), which treats quantum fluctuations in the time domain and works directly in the thermodynamic limit(6). Going back to chemistry, the multi-configuration time-dependent Hartree (MCTDH) algorithm is a versatile method for solving the time-dependent Schrödinger equation in multidimensional dynamical systems that involve distinguishable particles. In this case it is possible to calculate the quantum motion of a molecular system's nuclei as it evolves on one or more interconnected electronic potential energy surfaces. However, it becomes computational demanding with increased dimensionality, such that state-of-the art simulations consider up to tens of (internal) coordinates.

At the time of writing this notes, IBM showcased a record-sized real-time dynamics of a 2D heavy-hexagon lattice Ising model using Trotterization, 127 qubits, and error mitigation techniques(7) (see later for definition of these words) Just ten days later, classical tensor network simulations achieved the same result(8; 9), raising the bar once again to declare a quantum advantage.

4.1.2 Recent claims of quantum advantage / quantum utility

As we mentioned, today's hardware is not error-corrected as we are in the so-called NISQ era. However, one of the (if not *the*) most pressing challenges of today's quantum computing is demonstrating whether present imperfect quantum technologies could offer a real benefit compared to classical computers without error correction. Hamiltonian simulations algorithms, even when performed without error correction could produce such as advantage.

At the time of writing these lecture notes, (June 2023) researchers at IBM showcased a record-sized real-time dynamics of a 2D heavy-hexagon lattice Ising model using product formulas, 127 qubits, and some error mitigation techniques.(7)

Simply looking at the size of the register, it would be easy to conclude that a simulation with 127 qubits is beyond the reach of any classical competitor, solely because storing such a quantum state is not feasible (2^{127} components). However, there are important considerations to be made: the first is that this statement is true if we want to reproduce the *exact* dynamics. The fact that we are dealing with noisy machines means that hardware noise still limits the total simulation time, so classical benchmarks must be able to replicate the short-time dynamics of a quantum many-body model, not an evolution for an arbitrarily long time. This means that not all of the Hilbert space is visited during the quantum simulation. In practice, this limits the effective memory required in the classical case.

The second thing to keep in mind is the type of Hamiltonians (Ising in this case) that are being time-evolved. An approximate classical algorithm can leverage the locality and lattice structure to eliminate further components of the quantum states that have not become relevant during the dynamics.

Putting all of this together, several groups have successfully simulated IBM's experiment classically using classical tensor networks, achieving relatively reduced wall time and memory usage.([8]; [9]) This suggests that the threshold for declaring quantum advantage in Hamiltonian simulations (using noisy qubits) has shifted significantly toward larger sizes and longer evolution times.

In the rest of the chapter, however, we will consideri a fully digital, error-corrected simulation, so let us return to our world of noiseless unitary operators.

4.2 Product formulas

Quantum algorithms for real-time dynamics have the benefit of unifying all various application oriented methods. Currently, different communities use distinct classical algorithms that are best suited for specific applications. Quantum computing can afford to employ the brute-force approach, which is exponentially scaling in conventional hardware but not in quantum computing. The different applications will then reveal themselves in the particle-to-qubit mapping which is problem-specific and produces different qubit hamiltonians.

This note is specific to digital devices, which can run "algorithms" in a familiar sense compared to quantum simulators, where the Hamiltonian must be realized at the hardware level and the time-evolution is obtained "for free", like in an experiment. However, controlling all degrees of freedom of the system is typically not possible, resulting in hardware calibration errors[2] and decoherence that can compromise the accuracy of the evolution. These errors are difficult to detect, and therefore validating the results is essential. This difference mirrors the discussion in the Chapter 1 about error-correcting and analog machines. An informative, short review about analog and digital quantum simulators and threshold for quantum advantage is Ref ([10]).

After all this introduction and preambles, we can finally outline the quantum algorithm.

The first thing to consider, which is almost never explicitly stated in textbooks, is that, despite the fact e^{iHt} is a unitary operator, this does not mean that there exists an exact way to implement it on a quantum computer. For this reason, we need to resort to approximation that allows us to *compile* the problem using an efficient number of gates. Not surprisingly the most used algorithm for that is based on product formulas (or Trotter decomposition), where the exponent of a sum of operators is approximated by a product, introducing a discretisation error.

Let us start by assuming the existence of a qubit hamiltonian, acting on L qubits i.e

$$H = \sum_p P_p, \tag{4.2}$$

where P_p are Pauli strings, i.e. tensor products of L Pauli matrices, such as $ZZZ..$ or $XYIZ$. We will see later how to match a specific use-case with a qubit hamiltonian. Here let us be a

[2]to give an example, the Hubbard model can be effectively realized with cold atoms in optical lattices. Here fermionic atoms model the electrons and are trapped with laser beams as to model the lattice. The Hubbard U and t interactions are tuned by changing the optical trap parameters, however due to imperfections the effective Hubbard parameters simulated could be a different (U', t') set.

bit flexible with the notation and wording, as sometimes we can identify the spin $1/2$ operators with the X, Y, Z operators. A typical example of such hamiltonian is the Transverse field one,

$$H = \sum_i^L Z_i Z_{i+1} + \sum_i^L X_i. \tag{4.3}$$

This form coincides with the former one if we consider e.g. the term $Z_i Z_{i+1}$ as a Pauli string made of two consecutive Z operators embedded into a string on $L-2$ identities, i.e $II \cdots IZ_i Z_{i+1} I \cdots I$. In general a qubit Hamiltonian can be expressed as a sum of several terms

$$H = \sum_{k=1}^K H_k, \tag{4.4}$$

In the worst case scenario, $H_k = P_k$, i.e. where each Pauli string can be identified with one term, but here we used different indexes because the terms H_k are commuting groups of sets of Pauli strings. For instance, for the Ising model, $K = 2$, and $H_1 = \sum_i^L Z_i Z_{i+1}$, whereas $H_2 = \sum_i^L X_i$.

In the simplest algorithm, one just discretizes the total time t, into n timesteps, of length $\delta t = t/n$, and approximates

$$e^{-iH\delta t} \approx e^{-iH_K \delta t} \cdots e^{-iH_1 \delta t}, \tag{4.5}$$

introducing an $O(\delta t^2)$ error per timestep. The total evolution of duration t is implemented by repeating n times this decomposition

$$e^{-iHt} \approx U_{(1\text{st})} = \left(e^{-iH_K t/n} \cdots e^{-iH_1 t/n}\right)^n, \tag{4.6}$$

where we make evident the trick of dividing and multiplying by n. The overall error from repeating this sequence n times accumulates at most linearly with n going as $O(t^2/n)$, which can be made **arbitrarily small with large** n. More precisely, the Trotter error of this first-order decomposition is given by

$$\text{error}_{(1\text{st})} = \frac{t^2}{2n} \sum_{k,\ell}^K ||[H_k, H_\ell]|| + \text{higher orders} \tag{4.7}$$

From this equation we see that the error is controlled by the size of the commutators[3]. The first-order formula can be systematically generalized to higher-order formulas. For instance the second-order expansion reads For instance, the second-order PF (PF2) approximates e^{-iHt} by

$$e^{-iH\delta t} \approx U_{(2\text{nd})} = \left[\left(e^{-iH_1 \frac{t}{2n}} \cdots e^{-iH_K \frac{t}{2n}}\right)\left(e^{-iH_K \frac{t}{2n}} \cdots e^{-iH_1 \frac{t}{2n}}\right)\right]^n, \tag{4.8}$$

which introduces an error of order $\mathcal{O}(\delta t^3)$ per cycle and a cumulative error of order $\mathcal{O}(\delta t^3/n^2)$. Interestingly, in the case of $K = 2$, i.e. of the Ising model, the second order formula, per time slice, simplifies to

$$e^{-iH\delta t} \approx e^{-iH_1 \frac{t}{2n}} e^{-iH_2 \frac{t}{n}} e^{-iH_1 \frac{t}{2n}}, \tag{4.9}$$

[3]by $|| \cdots ||$ is implied the spectral norm of an operator, meaning the maximum singular value of the matrix

such that, when joined together it is equal to the first-order one *in the bulk* and differs only at the two extremities, where the operator H_1 needs to be evolved for half time at the beginning and at the end. In this case, we can have the accuracy of the second-order formula at (almost) the cost of a first order one, since it only introduces one additional time slice.[4]

Overall, the product formulas are a powerful method to simulate the dynamics of quantum systems with arbitrary and systematically improvable accuracy, at the cost of increasing the computational cost of the algorithm.

4.2.1 Trotter errors and randomized compilation

The $t^2/2n$ error bound, to leading order, provided by Lloyd has been further refined considering the contribution from higher-order terms. All in all the error scaling has and is still subject of theoretical research, also because theoretical bounds can be several orders of magnitude larger than what is observed, even for quite small systems, see e.g. the fairly recent (2021) Ref. ([11]) Sometimes, the observed scaling is also asymptotically better than the deterministically predicted one, just looking at the magnitude of the commutators, and the upper bound spectral norm of the terms, $\max_k ||H_k||$. This means that the errors also depends on other factors, such as the type and the sparsity of the Hamiltonians, and different bounds exists for different problems.

By studying better the error, researchers devised clever implementation of the product formula. For instance, the randomized product methods improve the scaling([12]) by sampling order of the H_k terms, rather than performing always the Trotterization sequentially. There are two ways to see why randomization may help, the first is high-level one: random compilation has been shown to reduce errors below what is feasible with a deterministic compiler, and since as we said, producing Hamiltonian simulation circuits is a special case of compilation, one can guess that randomization can also be helpful in this setting.
The second one, can be understood by simply inspecting the origin of the first-order trotter error arises: Let's consider again a simple Hamiltonian made of two operators, $H = H_1 + H_2$. The Taylor expansion of the first-order formula as a function of δt is

$$U_{(1\text{st})}(\delta t) = \exp(\delta t H_1)\exp(\delta t H_2) = I + \delta t(H_1+H_2) + \frac{\delta t^2}{2}(H_1^2 + 2H_1H_2 + H_2^2) + O(\delta t^3), \quad (4.10)$$

whereas the Taylor series of the ideal evolution is

$$e^{iH\delta t} = \exp((H_1+H_2)\delta t) = I + \delta t(H_1+H_2) + \frac{\delta t^2}{2}(H_1^2 + H_1H_2 + H_2H_1 + H_2^2) + O(\delta t^3). \quad (4.11)$$

We can bind the spectral-norm error as

$$\left\| e^{iH\delta t} - U_{(1\text{st})}(\delta t) \right\| \leq \|[H_1, H_2]\| \frac{\delta t^2}{2} + O\big((\max\{\|H_1\|, \|H_2\|\})\delta t)^3\big), \quad (4.12)$$

recovering Llyod result, as expected. However, we can also see the origin of the commutator: it is impossible to approximate the exact operator to second order using a (ordered)

[4]the cost only depends on the number of layers, not on the size of scalar δt, which will just be absorbed into rotation angles.

product of only two exponentials of H_1 and H_2, as we can have only one of the products H_1H_2 or H_2H_1 in the expansion. If only we could have a symmetric operator of type $1/2[\exp(\delta t H_1)\exp(\delta t H_2)+\exp(\delta t H_2)\exp(\delta t H_1)]$ we could match the expansion up to second-order. This operator may however be not unitary so a clever solution is to reverse sometimes the order of the operators in the time-slices, as to make the H_2H_1 product also appear.

A further extension is the "Qdrift" algorithm, where the terms are sampled with probability proportional to their coefficient in the Hamiltonian expansion[5]

A rather technical paper with improved bounds for product formula errors, which are "problem aware" is Ref. (11).

4.3 Gate compilation for Pauli strings

The usage of product formulas makes sense only if, while we don't know an exact compilation of e^{itH}, we know how to write a circuit of each of the K terms e^{itH_k}. We also remind that we consider compilation using standard gate sets introduced in Sect. 2.3.3, namely Clifford and single qubit rotations.

Gladly, evolution of terms like e^{itP_k}, where P_k, as above, is a Pauli string, is simple. They can be easily simulated because they can be diagonalized using Clifford operation, i.e. the same concept outlined in Sect. 2.4.2, when discussing the **change of basis** to measure operators.

Let's start from the simplest Hamiltonian, the one-body operator $H = \alpha Z$, where Z is the Pauli matrix. The corresponding evolution operator

$$e^{-i\alpha Z t} = R_z(2\alpha t) \tag{4.13}$$

is exactly the rotation-z gate we introduced in Eq. 2.2.1, where we take into account that the definition of the gate in its matrix format includes a factor 1/2.

The second easiest case is for instance a non-diagonal Hamiltonian $H = \alpha X$. This case admits two circuits: (i) the evolution operator can be simulated using directly the $R_X(2\alpha t)$ rotation, or we can use the $R_Z(2\alpha t)$ and a change of basis. The reason why this can be done is connected with Sect. 2.4.2 but it is actually less straightforward as it requires the following Theorem[6]: if U is a unitary matrix, then

$$e^{UAU^\dagger} = Ue^AU^\dagger. \tag{4.14}$$

This can be proven using Taylor expansion[7] and it allows us to "transfer" the unitary U, enabling the change of basis, from the exponent to outside the rotation gate. In this case, we would make use of the known identity $X = HZH$[8] and

$$e^{-i\alpha X} = e^{-i\alpha HZH} = H\ e^{-i\alpha Z}\ H. \tag{4.15}$$

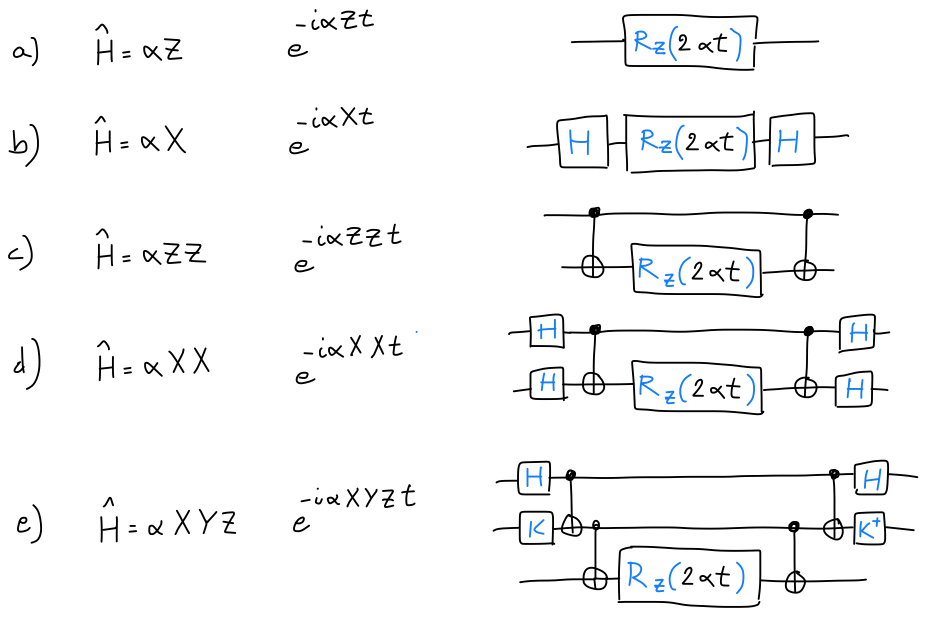
[5]these are not appearing explicitly in Eq. 4.4, though we can assume that each H_k in there can be written as c_kH_k where c_k is a scalar coefficient and H_k is the operator.

[6]with additional assumptions on A which are satisfied here, i.e A is a matrix of the same size of U.

[7]the key step is realizing that $(UAU^\dagger)^k = UA^kU^\dagger$, and insert this in the Taylor expansion $e^{tUAU^\dagger} = \sum_{k=0}^{\infty}\frac{(tUAU^\dagger)^k}{k!}$

[8]where this H is the hadamard gate, not the hamiltonian.

This can be realized using the same rotation R_Z gate, sandwiched between two Hadamard gates.



a) $\hat{H} = \alpha Z$ $e^{-i\alpha Z t}$
$R_z(2\alpha t)$
b) $\hat{H} = \alpha X$ $e^{-i\alpha X t}$
H $R_z(2\alpha t)$ H
c) $\hat{H} = \alpha Z Z$ $e^{-i\alpha ZZ t}$
$R_z(2\alpha t)$
d) $\hat{H} = \alpha X X$ $e^{-i\alpha XX t}$
H H $R_z(2\alpha t)$ H H
e) $\hat{H} = \alpha XYZ$ $e^{-i\alpha XYZ t}$
H K $R_z(2\alpha t)$ H $K^\dagger$

Figure 4.1: Circuits for time-evolving exactly some Pauli strings operators.

.

The next case is a multi-qubit hamiltonian $H = \alpha Z \otimes Z$. The general trick in these cases, where the operator is not directly a defined gate is to diagonalize it and look at the effect of the operator to the eigenstates. For the $Z \otimes Z$ operator the eigenbasis is already the computational basis (by definition). Let's take for instance the first ket $|00\rangle$: then $Z \otimes Z |00\rangle = (1 \times 1) |00\rangle = +1 |00\rangle$, while we would have $Z \otimes Z |01\rangle = -1 |01\rangle$. Since

$$e^{-iHt} |\psi\rangle = e^{-i\lambda t} |\psi\rangle \,, \quad \text{if} H |\psi\rangle = \lambda |\psi\rangle \,, \tag{4.16}$$

then $e^{-i\alpha Z \otimes Z} |00\rangle = e^{-i\alpha} |00\rangle$, while $e^{-i\alpha Z \otimes Z} |01\rangle = e^{+i\alpha} |01\rangle$ So the trick is to find a sequence of operations that assigns the correct *phase* to each of the four basis states. The systematic way to determine the correct sign is by performing a so-called *parity-check* of the vector. The circuit in panel (c) of Fig. 4.1 performs exactly this task[9] We remember that, if the circuit works for each basis state, then it will work when a generic quantum state is the input.

The circuit for the non-diagonal two-qubit case, i.e $H = \alpha X \otimes X$ can be produced by combining the last two ideas, the basis rotation and the parity check, yielding the circuit (d)

[9]The reader can check the evolution of every computational basis state $|00\rangle, |01\rangle, \cdots$ after each check-point, similarly to the calculation featured in Fig. 2.4

of Fig. 4.1.

Further, we can generalize these prescriptions to a multi-qubit Pauli such as $H = \alpha X \otimes Y \otimes Z$. The basis rotations will transform the original operator into $\alpha Z \otimes Z \otimes Z$, which can be dealt with an extended CNOT parity check. Now, if the support[10] of these Pauli strings is finite, i.e. $\mathcal{O}(1)$, the number of CNOT needed to implement these terms is also finite. While this kind of operators seems exotic, we will see that they naturally arise in fermionic simulations.

Finally, we remind that, in case of sums of terms like $H = \alpha Z \otimes Z + \beta X \otimes I$, one needs to find the basis that diagonalize H to exploit the trick outline above to time-evolve exactly e^{-iHt}. But in general, this is as hard as to solve the problem beforehand so that's why we use Trotterization. For small systems, one can resort to matrix identities that allow for shortening the circuit. On a different perspective, such optimized (in terms of CNOT counts)[11] circuits could also be found in the *transpilation* phase (cfn. Sect 1.3). In Fig. 4.2 we give an example for a two-qubit anysotropic Heisemberg Hamiltonian[13] The circuit further simplifies in case where two parameters become equal.

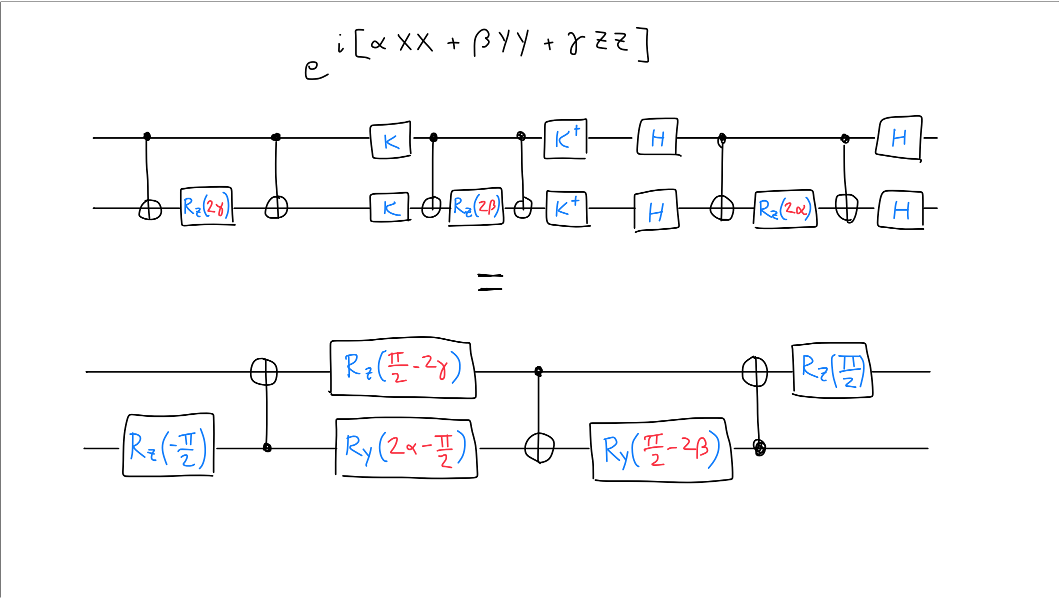

$e^{i[\alpha XX + \beta YY + \gamma ZZ]}$
$R_z(2\gamma)$
K
$K^\dagger$
$R_z(2\beta)$
H
$R_z(2\alpha)$
=
$R_z(\frac{\pi}{2}-2\gamma)$
$R_z(\frac{\pi}{2})$
$R_z(-\frac{\pi}{2})$
$R_y(2\alpha-\frac{\pi}{2})$
$R_y(\frac{\pi}{2}-2\beta)$

Figure 4.2: Initial version of trotterized circuit and the optimized version which yields the equivalent unitary operations where the three blocks are not visible anymore.
.

[10]the number of non-identity Pauli matrices
[11]therefore optimized for NISQ applications where the CNOT is the most noisy operation

4.4 Linear Combination of Unitaries (LCU)

We saw how it is possible to perform a product of unitaries. What about the sum?

First of all, let's notice that the sum of a number of unitaries is not guaranteed to be also an unitary operator. Most likely the opposite occurs[12] In general, summing up unitaries has a lot of applications, but in the context of hamiltonian dynamics it is easy to understand its value: one can directly add up the terms of the Taylor expansion in a very controllable way, without Trotter commutator-related error,

$$e^{-iHt} = 1 - iHt - H^2t^2/2 + \cdots = \sum_{r=0}^{R} \frac{(-itH)^r}{r!}, \tag{4.17}$$

for an expansion up to the R-th order. Notice that, assuming that the Hamiltonian itself is a sum of terms

$$H = \sum_{k=1}^{K} c_k H_k, \tag{4.18}$$

where we explicitely write the coefficients, then the sum of unitaries of the truncated Taylor, at order R, expansion of e^{-iHt}, assume the form

$$\sum_{j=0}^{\kappa} a_j V_j, \tag{4.19}$$

where $\kappa = \sum_{r=0}^{R} K_r$, i.e. the sum of the number of terms constituting each of the possible powers of H, and V_j is in the form of products of $H_1 H_2 \cdots H_k(r)$, and $a_j > 0$.

However, without entering too much into cumbersome details and notations, let's present here the general idea of linear combinations of unitaries. In a more general format, let's assume we have the following sum of L unitaries U_ℓ,

$$W = \sum_{\ell=0}^{L-1} \alpha_\ell U_\ell. \tag{4.20}$$

where $\alpha_\ell \geq 0$[13] In the general case. W is non unitary, so we need to use a larger system such that the action on a subregister $|\psi\rangle$ is our non-unitary operation $W|\psi\rangle$, but the overall operation is unitary.

$$U = \begin{bmatrix} W & \cdot \\ \cdot & \cdot \end{bmatrix}, \tag{4.21}$$

promoting the state into a larger state

$$|\psi\rangle \rightarrow \begin{bmatrix} |\psi\rangle \\ \mathbf{0} \end{bmatrix}, \tag{4.22}$$

where we use the notation $\mathbf{0}$ to remind that the additional register can be made of more than a single ancilla qubit, so this is a multi-qubit zero state. The idea is that, if we apply the

[12]Simplest counter-example is the operator $I + Z$. For instance $(I + Z)/2 = |0\rangle\langle 0|$ is a projector.

[13]we can always incorporate the minus sign into the definition of the unitary U_ℓ.

unitary U to the total state $|\mathbf{0}\rangle |\psi\rangle$, **when** we measure $\mathbf{0}$, we applied the sum. We stress the word "when", because this is a probabilistic approach, with a failure probability (which can be actually large). This general idea is called **block-encoding**. Notice that, since the operation is non unitary, what we (could) produce (non-deterministically) is actually the *normalized* state

$$|\psi\rangle \to \frac{W|\psi\rangle}{|W|\psi\rangle|}. \tag{4.23}$$

Let us start with a concrete simple case, to show how the corresponding circuit for U can be build. So far we know that we need at least an ancilla qubit. Suppose that we want to realize the sum $W = A + B$, where A, B are unitaries, and suppose we can obtain the "controlled" version of these unitaries as in Sect. 2.3.2. Then the circuit in Fig. 4.3 will work.

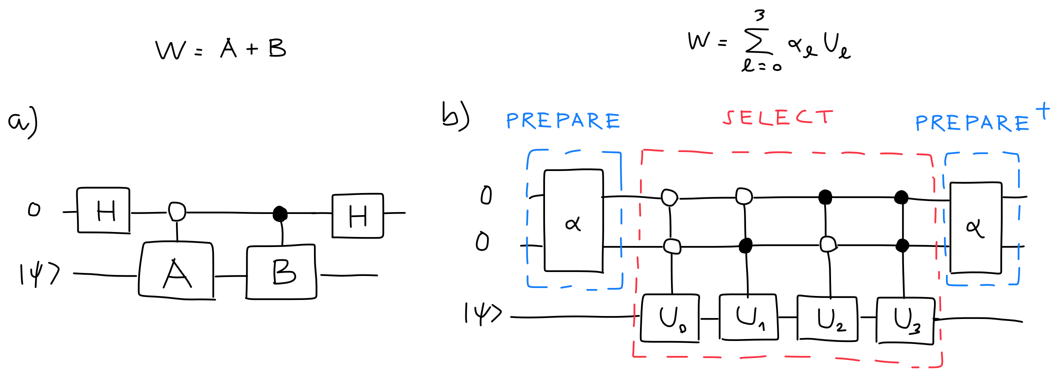


Figure 4.3: Left: Circuit to perform $A + B$. Right: More general case, where the PREPARE and SELECT blocks are highlighted.

.

To see this we can ideally break the evolution at each layer. The initial state reads $|0\rangle |\psi\rangle$. After the Hadamard, the ancilla is in superposition, so (neglecting for brevity the factor $1/\sqrt{2}$ each time) $(|0\rangle + |1\rangle) |\psi\rangle$. Then the first controlled operation (where the white dot means controlling with '0', rather than '1') activates the application of A: $|0\rangle A |\psi\rangle + |1\rangle |\psi\rangle$, the second controlled operation activates also B: $|0\rangle A |\psi\rangle + |1\rangle B |\psi\rangle$. Now the final Hadamard on the ancilla creates $(|0\rangle + |1\rangle) A |\psi\rangle + (|0\rangle - |1\rangle) B |\psi\rangle$, which after redistributing the components (and restoring the prefactors) reads

$$|\phi\rangle = \frac{1}{2} \left(|0\rangle (A + B) |\psi\rangle + |1\rangle (A - B) |\psi\rangle \right). \tag{4.24}$$

This means that, when the ancilla qubit is measured in "0", then we have applied $(A + B)/2$ to the state $|\psi\rangle$.

What is the success probability of measuring '0'? Formally we need to compute

$$P_0 = \langle\phi| \big(|0\rangle\langle 0| \otimes I\big) |\phi\rangle \tag{4.25}$$
$$= \frac{1}{4}\langle\psi| (A^\dagger + B^\dagger)(A+B)|\psi\rangle \tag{4.26}$$
$$= \frac{1}{4}\langle\psi| (2 + A^\dagger B + B^\dagger A)|\psi\rangle \tag{4.27}$$
$$= \frac{1}{2}\big(1 + \frac{1}{2}\langle\psi| (A^\dagger B + B^\dagger A)|\psi\rangle\big), \tag{4.28}$$

which is ≤ 1. For instance, if A, B are two Pauli matrices, the anticommutator is 0, such that the success probability is 1/2.

The procedure can be generalized to the case where the unitaries are more than two (in this case we need more ancillas), and the coefficients of the sum are not equal. In the traditional jargon, the first step of initializing the ancilla register in order to encode the coefficients of the sum is called PREPARE, while the block featuring the controlled operations is called SELECT. The final rotation on the ancilla is PREPARE†.

In the general case of Eq. (4.20), the first operation initializes the ancilla register in such a way that all the binary components $|\ell\rangle$ have the correct amplitudes. Let's define the scale factor

$$\lambda = \sum \alpha_\ell. \tag{4.29}$$

Then the PREPARE operation (acting on the ancilla register) is

$$\mathsf{PREPARE}(\alpha)|\mathbf{0}\rangle = \sum_\ell \sqrt{\frac{\alpha_\ell}{\lambda}}|\ell\rangle \tag{4.30}$$

which generalizes the previous case where $\alpha_0 = \alpha_1 = 1$, and $\lambda = 2$, that, could be simply performed by an Hadamard gate. Then, the SELECT block, formally implements the following

$$\mathsf{SELECT}_U = \sum_\ell |\ell\rangle\langle\ell| \otimes U_\ell, \tag{4.31}$$

which extends the definition of a "controlled-U_ℓ", to a multi-bit string "ℓ". Defining the "prepared" state $|A\rangle = \mathsf{PREPARE}(\alpha)|\mathbf{0}\rangle$ (eq. 4.30), the application of SELECT_U on the register $|A\rangle|\psi\rangle$ reads

$$\mathsf{SELECT}_U |A\rangle|\psi\rangle = \frac{1}{\sqrt{\lambda}}\sum_\ell \sqrt{\alpha_\ell}|\ell\rangle U_\ell|\psi\rangle \tag{4.32}$$

We then use the PREPARE† circuit to revert to the computational basis for the ancilla, such that when we measure the multi-qubit **0** state, the operation has been performed.

More formally the demonstration proceeds as follows. First of all, let's point out that we cannot explicitly write the full action of PREPARE† on a generic state, because we only know its definition when acting on $|\mathbf{0}\rangle$ (cfn. Eq. 4.30). So to proceed further, we can only apply the bra $\langle\mathbf{0}|$ to the resulting state $\mathsf{PREPARE}^\dagger\big(1/\sqrt{\lambda}\sum_\ell\sqrt{\alpha_\ell}|\ell\rangle U_\ell|\psi\rangle\big)$. Now we can calculate the inner product of $\langle \mathbf{0}| \, \mathsf{PREPARE}^\dagger|$ with the state in Eq. 4.32

$$\langle \mathbf{0}\otimes I|\mathsf{PREPARE}^\dagger\big(1/\sqrt{\lambda}\sum_\ell \sqrt{\alpha_\ell}\,|\ell\rangle\, U_\ell\,|\psi\rangle\,\big) \tag{4.33}$$

$$\langle \mathbf{0}|\mathsf{PREPARE}^\dagger\big(1/\sqrt{\lambda}\sum_\ell \sqrt{\alpha_\ell}\,|\ell\rangle\, U_\ell\,|\psi\rangle\,\big) \tag{4.34}$$

$$\frac{1}{\lambda}\sum_{j,\ell}\sqrt{\alpha_j\alpha_\ell}\,\langle j|\ell\rangle\, U_\ell\,|\psi\rangle \tag{4.35}$$

$$\frac{1}{\lambda}\sum_{\ell}\alpha_\ell U_\ell\,|\psi\rangle\,, \tag{4.36}$$

where we used the fact that the identity I acts on the $|\psi\rangle$ register, then the definition of Eq. 4.30, and $\langle j|\ell\rangle = \delta_{j,\ell}$ because these are basis states of the ancilla register. Notice that we were able to perform the calculation without a recipe for evaluating $\mathsf{PREPARE}^\dagger\,|\ell\rangle$, if we are interested in evaluating only the matrix element corresponding to the ancilla zero state.

We have therefore proven that

$$W/\lambda = \langle \mathbf{0}|\,\mathsf{PREPARE}^\dagger\ \mathsf{SELECT}_U\ \mathsf{PREPARE}\,|\mathbf{0}\rangle\,, \tag{4.37}$$

i.e. $\mathsf{PREPARE}^\dagger$ SELECT $\mathsf{PREPARE}$ block-encodes W/λ in the $\langle \mathbf{0}|\cdots|\mathbf{0}\rangle$ block. The normalization is essential because the row and column of the *larger* unitary matrix needs to sum to one, but can be arbitrarily large.

One important aspect to comment on is the success probability. From this, we see that the probability of success is

$$||W\,|\psi\rangle\,||^2/\lambda^2, \tag{4.38}$$

which can be in principle very small. There are however methods, called *amplitude amplification* which can increase this probability (cfn. 7.1). Finally, it is assumed that each $\mathsf{PREPARE}$, and SELECT steps are efficient.

Going back to our hamiltonian dynamics example, it can be shown[14] that the complexity of LCU for Hamiltonian simulation, to achieve a target error ϵ is(14)

$$\mathcal{O}\big(\lambda t \text{ polylog}(\lambda t/\epsilon)\big) \tag{4.39}$$

So it offers an improved scaling with t compared to product formulas[15], at the cost of **introducting ancillas**, which generally scales logarithmically with the number of terms to be added. Notice that, with k ancilla qubits we can manage a number of terms which is $L = 2^k$, as in Fig. 4.3.

It is the subject of research improved methods to express the sum of unitaries as to minimize λ. Just to give an idea about what can be done, for electronic structure hamiltonians one could resort to using different basis sets, or by grouping the Pauli strings in different ways(15).

[14]proof is actually very cumbersome and needs amplitude amplification

[15]Notice that this is t, not δt. So it is generally $t >> 1$.

However, we observe that it is not possible to do much more than this asymptotically because the $\mathcal{O}(t)$ scaling is optimal because having a sub-linear scaling in t would violate the *no fast forwarding theorem* according to which it is **impossible** to simulate a generic quantum system faster than t, its real evolution.

Further, we remark that LCU is a technique which **scope goes much beyond hamiltonian simulation**. It extends the reach of quantum computing from processing unitary matrices to arbitrary matrices, but now with a non-zero failure probability.

4.5 Finding ground states through dynamics

Interestingly, hamiltonian simulation can be used also to access ground state properties. The primitive which is used is the Quantum Phase Estimation algorithm.

We saw that the eigenvalues of operators H in the exponent of unitary operator $U = e^{-iHt}$ result in a *phase*, such that measuring the phase results in measuring the eigenvalue. Let us first assume that $|\psi_n\rangle$ is an eigenstate of H with eigenvalue E_n. Under time evolution $e^{iHt}|\psi_n\rangle = e^{iE_nt}|\psi_n\rangle$ the state picks up a phase $\phi = E_nt$.

The general idea is to introduce (again) an ancilla qubit. The circuit in Fig. 4.4 (left) allows us, in principle, to evaluate ϕ assuming the existence of an oracle U that applies the desired unitary operator. To be practical, we can assume that this whole controlled-U operation is itself a Trotter expansion of e^{iHt}, in the case of physics problems, though QPE is much more general.

To see how the circuits allows us to compute the phase through measurements we follow step by step the evolution, similarly to what we did when discussing the LCU case. The sequence of state transformation reads

$$|0\rangle\ \ |\phi_n\rangle \rightarrow \tag{4.40}$$

$$\frac{1}{\sqrt{2}}\big(\,|0\rangle + |1\rangle\,\big)\ \ |\phi_n\rangle = \frac{1}{\sqrt{2}}\big(\,|0\rangle\,|\phi_n\rangle + |1\rangle\,|\phi_n\rangle\,\big) \rightarrow \tag{4.41}$$

$$\frac{1}{\sqrt{2}}\big(\,|0\rangle\,|\phi_n\rangle + |1\rangle\,U\,|\phi_n\rangle\,\big) = \frac{1}{\sqrt{2}}\big(\,|0\rangle\,|\phi_n\rangle + |1\rangle\,e^{i\phi_n}\,|\phi_n\rangle\,\big). \tag{4.42}$$

So while, the bottom register remains unchanged in its $|\phi_n\rangle$ state, the ancilla qubit picks up a phase in its $|1\rangle$ component[16] Now, a measurement at this point would not be able to extract such information, as we would collapse the ancilla state in '0' with 0.5 probability and in '1' with $|e^{i\phi_n}/\sqrt{2}|^2 = 0.5$ probability.

The final Hadamard rotation transfers this information into the *amplitudes* of the two states. We have indeed

$$\frac{1}{2}\big[\,|0\rangle\,(1 + e^{i\phi_n})\,|\phi_n\rangle + |1\rangle\,(1 - e^{i\phi_n})\,|\phi_n\rangle\,\big], \tag{4.43}$$

[16]this case, where the ancilla qubit picks up a phase after controlling the execution of an operator acting on another register, is called *phase kick-back*.

such that the probability of measuring '0' is now $(1 + \cos \phi_n)/2$ making the value of ϕ_n accessible to measurements in the computational basis.

Now, if we relax the condition that we prepared the exact eigenstate $|\psi_n\rangle$ of U, but we have a generic quantum state

$$|\psi\rangle = \sum_n c_n |\psi_n\rangle \tag{4.44}$$

when we measure the ancilla register we will measure the phase ϕ_n with probability $|c_n|^2$. In this case, we will also collapse the state register in the eigenvector $|\psi_n\rangle$. It's clear that, if we want to find the ground state E_0, this procedure will have a success probability of $|c_0|^2$, so the state we need to prepare must have a good overlap with the ground state.

This method, although simple to rationalize, is inefficient because it relies on repeated measurements to accumulate sufficient statistics to resolve the measurement's imbalance between the two read-outs.

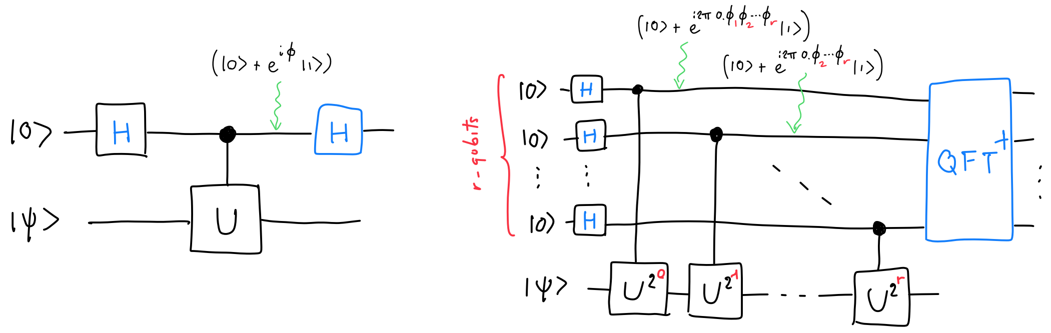

$(|0\rangle + e^{i\phi}|1\rangle)$
$|0\rangle$
$|\psi\rangle$
H
U
H
$(|0\rangle + e^{i2\pi 0.\phi_1\phi_2\cdots\phi_r}|1\rangle)$
$(|0\rangle + e^{i2\pi 0.\phi_2\cdots\phi_r}|1\rangle)$
r-qubits
$|0\rangle$
$|0\rangle$
$|0\rangle$
H
H
H
$QFT^\dagger$
$|\psi\rangle$
U^{2^0}
U^{2^1}
U^{2^r}

Figure 4.4: Left: One qubit QPE. Right: Kiteav QPE. The rightmost ancilla transformation is depicted in blue, as QFT on one qubit reduces to the H gate. Notice a typo in this panel: the last controlled unitary should be $U^{2^{r-1}}$, as the index of the ancillas ranges from 0 to $r-1$.
.

4.5.1 Quantum Phase estimation with Quantum Fourier Transform

The following algorithm is the canonical one (by Kitaev) and here we expand the presentation of Ref. (16) more pedagogically.[17] Perhaps the first thing to notice to understand the notation is that, since the eigenvalues λ's of a unitary operator U has a unitary norm, they can be written as $e^{i2\pi\phi}$, where $0 \leq \phi < 1$. The phase ϕ can be written as a binary expansion

$$\phi = 0.\phi_1\phi_2\cdots\phi_r \quad \rightarrow \quad \sum_{i=1}^{r} \phi_i 2^{-i} \tag{4.45}$$

where the ϕ_i's can be either 0 or 1.[18] Notice that now we drop the index n from $\phi_n = \phi$, as to not confuse the index of the eigenvalue, with the one of the binary expansion of the phase.

[17] if you are already confident about the demonstration of QPE, feel free to skip this subsection.
[18] For instance $0.5_{\text{dec}} = 0.1_{\text{bin}}$, or $0.75_{\text{dec}} = 1/2 + 1/4 = 0.11_{\text{bin}}$

Kitaev method introduces r ancilla qubits. Each ancilla qubit controls the application of a given power of the unitary U, i.e $U, U^2, U^4, \cdots, U^{2^{r-1}}$, and then measures all bits to receive a binary representation of the phase at once (cfn. Fig. 4.4).

The algorithm can be best understood step-by-step in this way:

Let's start with $r = 1$, and *assume* that the incoming register state $|\psi\rangle$ is an eigenstate of U^{2^0}, with eigenvalues

$$\lambda = e^{i2\pi 0.\phi_1}, \tag{4.46}$$

where we repeat that $\phi_1 = \{0, 1\}$. This is of course a strong assumption[19], but I find it easier to rationalize the effect of the circuit starting from this case, and then add more ancilla to resolve an arbitrary phase with r bits. Then, the application of the first two operations (H and c-U$^{2^0}$) produces the same state as in Eq. 4.42, but written using the new notation:

$$\big(\,|0\rangle + e^{i2\pi 0.\phi_1}\,|1\rangle\,\big) \otimes |\psi\rangle \tag{4.47}$$

omitting the prefactor $\frac{1}{2}$ for clarity, and explicitely writing the $\otimes$ symbol just for easier visualization of the separate subregisters.

Now, let's use $r = 2$, and assume that now the eigenvalues can be written exactly as

$$\lambda = e^{i2\pi 0.\phi_1\phi_2}, \tag{4.48}$$

and the incoming state $|\psi\rangle$ is again eigenstate of $U = U^{2^0}$. The state after the first controlled c-U$^{2^0}$ operation, but before the second c-U$^{2^1}$ reads as before

$$\underbrace{\big(\,|0\rangle + e^{i2\pi 0.\phi_1\phi_2}\,|1\rangle\,\big)}_{\text{ancilla 1}} \otimes \underbrace{\big(\,|0\rangle + |1\rangle\,\big)}_{\text{ancilla 2}} \otimes |\psi\rangle\,, \tag{4.49}$$

because the $0.\phi_1\phi_2 = \phi$ is by definition the 'new' phase of U, and we introduce the second ancilla qubit, which at this point has just been initialized in superposition.
Now we proceed with the second c-U$^{2^1}$ operation, controlled now by the second ancilla, unused so far. Since $U^{2^1} = UU$, $|\psi\rangle$ is still an eigenvector. Moreover the phase of U^2 is twice the phase of U, i.e. $U^2\,|\psi\rangle = e^{i2\pi(2\phi)}\,|\psi\rangle$.
Now, it comes the trick. First, it is easy to prove that twice the phase $0.\phi_1\phi_2$ is $\phi_1.\phi_2$, exactly as a multiplication by 10 of a decimal number shifts all the digits to the left by one position[20]. Then, the exponent simplifies to

$$e^{i2\pi(2\cdot 0.\phi_1\phi_2)} = e^{i2\pi\ \phi_1.\phi_2} \equiv e^{i2\pi(\phi_1 + 0.\phi_2)} = e^{i2\pi\phi_1} e^{i2\pi 0.\phi_2} = e^{i2\pi 0.\phi_2}, \tag{4.50}$$

where we use the fact that $e^{i2\pi\phi_1} = 1$ because ϕ_1 is integer. Therefore, the state after the second controlled rotation reads

$$\underbrace{\big(\,|0\rangle + e^{i2\pi 0.\phi_1\phi_2}\,|1\rangle\,\big)}_{\text{ancilla 1}} \otimes \underbrace{\big(\,|0\rangle + e^{i2\pi 0.\phi_2}\,|1\rangle\,\big)}_{\text{ancilla 2}} \otimes |\psi\rangle\,. \tag{4.51}$$

[19] i.e. the spectrum of U is highly degenerate with only two distinct values, $e^{-i2\pi 0.0} = 1$, $e^{-i2\pi 0.1}$.
[20] $2 \cdot 0.\phi_1\phi_2 = 2 \cdot (\phi_1/2 + \phi_2/4) = \phi_1 + \phi_2/2 = \phi_1.\phi_2$.

The general k-bit case, is a straighforward generalization of the previous steps, after proving that

$$e^{i2\pi 2^k\phi} = e^{i2\pi 0.\phi_k\phi_{k+1}\cdots}, \tag{4.52}$$

such that the quantum state produced after all the r sequence of controlled-U reads

$$\big(\,|0\rangle + e^{i2\pi 0.\phi_1\phi_2\cdots\phi_r}\,|1\rangle\,\big) \otimes \big(\,|0\rangle + e^{i2\pi 0.\phi_2\cdots\phi_r}\,|1\rangle\,\big) \otimes \cdots \otimes \big(\,|0\rangle + e^{i2\pi 0.\phi_r}\,|1\rangle\,\big) \otimes |\psi\rangle\,. \tag{4.53}$$

Now, the final step is to find a way to read-out the r bits $\{\phi_1, \phi_2, \cdots, \phi_r\}$. While a simple layer of Hadamard gates is not enough anymore, it is very convenient that there exists an operator that does exactly what we want, the (inverse) quantum Fourier transform (QFT)[21](16):

$$\begin{aligned}\mathsf{QFT}^\dagger\big[\big(\,|0\rangle + e^{i2\pi 0.\phi_1\cdots\phi_r}\,|1\rangle\,\big) \otimes \big(\,|0\rangle + e^{i2\pi 0.\phi_2\cdots\phi_r}\,|1\rangle\,\big) \otimes \cdots \otimes \big(\,|0\rangle + e^{i2\pi 0.\phi_r}\,|1\rangle\,\big)\big] &= \quad (4.54)\\ = |\phi_r\rangle \otimes |\phi_{r-1}\rangle \otimes \cdots \otimes |\phi_2\rangle \otimes |\phi_1\rangle\,. & \quad (4.55)\end{aligned}$$

The circuit of Fig. 4.4 (right) therefore allows us to extract the binary format of the phase ϕ, by simply measuring the ancilla register. Notice also that the QFT and the inverse can be performed using $\mathcal{O}(r^2)$ gates, so the whole procedure is efficient.

In the general case, where the phase is not exactly written as a finite $r-$bit expansion, the resulting measurements will not be exactly peaked on just one bitstring, but we will obtain an histogram of read-outs. The most frequent outcomes will be the strings that, in binary format, represent the values just below and just above the exact one. In principle, by increasing r, we will increase the resolution of the estimate. More rigorously, and following Ref. (16), to obtain an estimate of the phase which is precise up to n bits, we need to use a slightly larger number of ancilla qubits. In particular, to obtain a binary estimate of the phase, precise to n bits, with success probability $p = 1 - p_{\text{fail}}$, requires

$$r = n + \left\lceil \log_2\left(2 + \frac{1}{2p_{\text{fail}}}\right)\right\rceil. \tag{4.56}$$

As discussed, this is just the error introduced by the $r-$bit truncation of the precision for the phase read-out, assuming we prepared the exact eigenstate $|\psi\rangle$.

4.5.2 Errors and runtime for energy estimation

Now we can combine all sources of errors to understand how feasible is this strategy for computing ground-state energy with hamiltonian simulations. Notice that this final analysis contains details that are never properly discussed even in highly cited literature(17), so here I will mainly follow the arguments put forward in Ref.(18).

First of all, let's remember that in our case

$$U = e^{iHt}, \tag{4.57}$$

[21]see e.g. Nielsen and Chuang book for the equivalence of the two definitions of the QFT.

such that $U^{2^k} = e^{iH2^kt}$, and that we will need a Trotterized version of that. In principle, we would be tempted to use a very small evolution time t, requiring the smallest amount of Trotter steps. However, the resulting phase $\phi = Et$, would also be very small, requiring a lot of n-bits of precision to be resolved. So there is no gain in shrinking artificially the evolution time to 'save' Trotter steps. However, there is a rough 'a priori' recipe to tune t, which takes into account an upper bound of range of the spectrum of H, i.e $W_H > \mathrm{range}[H] = E_{\max} - E_{\min}$:

$$t\, W_H = 2\pi \tag{4.58}$$

such that there is a one-to-one correspondence between an energy eigenvalue and a phase eigenvalue. Clearly, the better is the bound of tW_H, the smaller is r, because we will "populate" efficiently all the 2π range with eigenvalues. As a matter of fact, one case use the evolution time t to rescale the range, or one can set $t = 1$ and rescale the Hamiltonian. What is important is that the phase spectrum fits a tightly the range 2π.

For read-out simplicity, we can further preprocess the hamiltonian in such a way that all energy eigenvalue are positive, such that the energy will be directly mapped to a positive phase between 0 and 2π. If we don't do this, the negative eigenvalues would simply appear in the $[\pi, 2\pi)$ sector. Notice that range of a physical hamiltonian W_H can be usually estimated without knowing the ground state E_0 a priori. For instance, in the case of a Pauli sum Hamiltonian

$$H = \sum_k h_k P_k, \tag{4.59}$$

one can bound the range summing up all the absolute value of the coefficients, $W_H = \sum_k |h_k|$.

What remain to be study is the runtime of the QPE to measure the energy with a target precision.

Let's call N_U, the number of applications of the "minimal unit" unitary $U = e^{iHt}$. Then, it's clear that the operator U^{2^k}, controlled by the ancilla at index $k{+}1$, requires 2^k applications of U to be realized. Since adding a new ancilla means doubling the total number of unitaries, the final number of unitaries for a $r-$bit QPE is $N_U \propto 2^r$. The total evolution time, T, is proportional to N_U.

To be more precise, and setting $W_H = 1$ (since we can always imagine rescaling the hamiltonian) we can set $t = 2\pi$, such that the evolution induced by U^{2^0} is 2π, for U^{2^1} is 4π, and for the last $U^{2^{r-1}}$ is $2^r\pi$. Summing up all the time-steps, we find that the total evolution time is $T = 2^{r+1}\pi$. Let us also define the 'discretization' precision error as $\epsilon = 1/2^n$. From Eq. 4.56 we calculate that we need $r = n + 2$ ancillas for a success probability of 50%. Therefore the total time is

$$T = 2^{r+1}\pi = 2^{n+3}\pi = 8\pi 2^n := \frac{8\pi}{\epsilon}, \tag{4.60}$$

for a success probability of 1/2. This means that on average we would need to repeat the algorithm twice to observe success, so the full cost, expressed in T, is $T = 16\pi/\epsilon$.

Then, there is a second source of increased cost, which is connected with the initial state preparation. As we saw, the nice thing about QPE is that, the state register collapses into the eigenstate ψ_n, when the phase ϕ_n is read-out. This happens with probability $|c_n|^2$ (cfn.

Eq. 4.44), such that, if we aim to read-out the ground state energy, we need to repeat the QPE algorithm $1/|c_0|^2$ times, i.e. inversely proportional to the overlap.

Bibliography

[1] A. Polkovnikov, K. Sengupta, A. Silva, and M. Vengalattore, "Colloquium: Nonequilibrium dynamics of closed interacting quantum systems," *Rev. Mod. Phys.*, vol. 83, pp. 863–883, Aug 2011.

[2] R. P. Feynman, "Simulating physics with computers," *International Journal of Theoretical Physics*, vol. 21, pp. 467–488, June 1982.

[3] S. Lloyd, "Universal quantum simulators," *Science*, vol. 273, no. 5278, pp. 1073–1078, 1996.

[4] M. Schmitt and M. Heyl, "Quantum many-body dynamics in two dimensions with artificial neural networks," *Phys. Rev. Lett.*, vol. 125, p. 100503, Sep 2020.

[5] G. Magnifico, T. Felser, P. Silvi, and S. Montangero, "Lattice quantum electrodynamics in (3+ 1)-dimensions at finite density with tensor networks," *Nature communications*, vol. 12, no. 1, p. 3600, 2021.

[6] H. Aoki, N. Tsuji, M. Eckstein, M. Kollar, T. Oka, and P. Werner, "Nonequilibrium dynamical mean-field theory and its applications," *Rev. Mod. Phys.*, vol. 86, pp. 779–837, Jun 2014.

[7] Y. Kim, A. Eddins, S. Anand, K. X. Wei, E. Van Den Berg, S. Rosenblatt, H. Nayfeh, Y. Wu, M. Zaletel, K. Temme, *et al.*, "Evidence for the utility of quantum computing before fault tolerance," *Nature*, vol. 618, no. 7965, pp. 500–505, 2023.

[8] J. Tindall, M. Fishman, M. Stoudenmire, and D. Sels, "Efficient tensor network simulation of ibm's kicked ising experiment," 2023.

[9] T. Begusic and G. K.-L. Chan, "Fast classical simulation of evidence for the utility of quantum computing before fault tolerance," 2023.

[10] A. J. Daley, I. Bloch, C. Kokail, S. Flannigan, N. Pearson, M. Troyer, and P. Zoller, "Practical quantum advantage in quantum simulation," *Nature*, vol. 607, pp. 667–676, July 2022.

[11] A. M. Childs, Y. Su, M. C. Tran, N. Wiebe, and S. Zhu, "Theory of trotter error with commutator scaling," *Physical Review X*, vol. 11, no. 1, p. 011020, 2021.

[12] A. M. Childs, A. Ostrander, and Y. Su, "Faster quantum simulation by randomization," *Quantum*, vol. 3, p. 182, sep 2019.

[13] A. Smith, M. Kim, F. Pollmann, and J. Knolle, "Simulating quantum many-body dynamics on a current digital quantum computer," *npj Quantum Information*, vol. 5, no. 1, p. 106, 2019.

[14] G. H. Low and N. Wiebe, “Hamiltonian simulation in the interaction picture,” *arXiv preprint arXiv:1805.00675*, 2018.

[15] I. Loaiza, A. M. Khah, N. Wiebe, and A. F. Izmaylov, “Reducing molecular electronic hamiltonian simulation cost for linear combination of unitaries approaches,” *arXiv preprint arXiv:2208.08272*, 2022.

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

[17] S. McArdle, S. Endo, A. Aspuru-Guzik, S. Benjamin, and X. Yuan, “Quantum computational chemistry,” *arXiv preprint arXiv:1808.10402*, 2018.

[18] 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.

Chapter 5

Physics-to-qubits mappings

In the previous chapters, we applied algorithms assuming to have Hamiltonians as sums of Pauli matrices, postulating that this is a generic form, although not so evident at first glance. In this chapter, we will show why very common Hamiltonians in physics and chemistry can actually be written in this way. This was already evident for lattice models where qubit and spin 1/2 operators are already equivalent. In general, this chapter addresses the topic of how to perform an encoding of a specific problem, for example in chemistry, into a qubit Hilbert space, and how to translate the operators describing the system in the physical space into that of the qubits. At the end of the chapter, we will also discuss a counterexample in which the Hamiltonian of the physical system does not admit an efficient translation as a sum of Pauli strings, yet it is efficiently computable in another framework.

5.1 Fermionic hamiltonians

We will start with this kind of physical system because is the most important for chemistry and physics. The most general fermionic-hamiltonian reads

$$H = \sum_{p,k} h_{pk} a_p^\dagger a_k + \sum_{p,q,r,s} h_{pqrs} a_p^\dagger a_q^\dagger a_r a_s, \tag{5.1}$$

where $a_p^\dagger$ and a_p are the fermionic creation and destruction operators, which create (annihilate) a particle in the spin-orbital p. While the name "orbital" is more chemistry-oriented, we could have also used the name "mode" or "site" which are more appropriate for condensed matter. What matters is that these operators obey **fermionic anti-commutation** relations

$$\begin{aligned} \{a_p, a_q^\dagger\} &= a_p a_q^\dagger + a_q^\dagger a_p = \delta_{pq}, \\ \{a_p, a_q\} &= \{a_p^\dagger, a_q^\dagger\} = 0. \end{aligned} \tag{5.2}$$

The computational basis in this case is a Fock state, which stores the *occupation* number of the orbitals. Let's also fix some definition so as to not generate confusion between condensed matter and chemistry. While *electrons* are the most popular *fermions*, the latter is a more general class. Electrons possess also a spin *up* or *down*, but there exist also spinless-fermions. This is precisely our notation.

5.1.1 Fock state encoding

Each single particle "orbital" p, is a spin-orbital. Therefore it can only host one fermion. A molecular orbital of quantum chemistry instead may host the *up* and the *down* particle. Our notation is however general, as we can think of the spin-orbital p to be the designated place for the spin-*up* fermion, in the molecular orbital ϕ_i (or site, in case of the Hubbard model), and the spin-orbital $p+1$ to be the one hosting the spin-*down* component of the same molecular orbital, or site ϕ_i.

For instance, according to this convention, Eq. 5.1 reduces to the Hubbard model when p and k, in the first sum, label spin-orbitals belonging to the same spin-sector (because an *up*-electron cannot change spin while hopping), and $p = r$, $q = s$, where p and q label two spin orbitals localized on the same site, but with opposite spin components. Then $h_{pk} = h_{i,i+1} = t$ is the hopping term, and $h_{pqpq} = h_{i\uparrow,i\downarrow} = U$ is the Couloumb repulsion term.

We can therefore consider the system to be defined in terms of spin-less fermions, doubling the total number of orbitals in case of electrons (or spinful fermions). In the case of one lattice site, or molecular orbital, this site can have four configurations: $\{\emptyset, \uparrow, \downarrow, \uparrow\downarrow\}$, i.e. the site can be empty, occupied by one of the two kind of fermions, or doubly occupied. This physical picture can be represented in terms of **occupancy** (un-occupancy) of two spin-orbitals, where the left (right)-most label the occupancy of the *up* (*down*) fermionic flavour. We can label with '0' the empty spin-orbital, and with '1' the occupied case: the above four physical combinations can be expressed using a binary notation, which is well suited for quantum computing:

$$\{\ \emptyset, \uparrow, \downarrow, \uparrow\downarrow\} \xrightarrow{\text{mapping}} \{|00\rangle, |10\rangle, |01\rangle, |11\rangle\}. \tag{5.3}$$

This completely defines our encoding. Now suppose to have two sites, where for instance, we have an *up*-electrons at site 1, and a *down*-electron at site 2. This particular electronic configuration reads

$$|\uparrow_1, \downarrow_2\rangle \xrightarrow{\text{mapping}} |1001\rangle, \tag{5.4}$$

where we need to be consistent with the *ordering* of the labels of the spin-orbitals. The symmetric case where now the electrons occupies the opposite sites would be $|\uparrow_2, \downarrow_1\rangle = |0110\rangle$.

Further, if we want to describe a physical quantum state where the two electrons delocalize, as to lower the kinetic energy, it could be done by

$$|\psi\rangle = \alpha\,|1001\rangle + \beta\,|0110\rangle. \tag{5.5}$$

More generally, the quantum state[1] describing two-electrons with opposite spins on two sites can be given by the following state

$$|\psi\rangle = \alpha\,|1001\rangle + \beta\,|0110\rangle + \gamma\,|1100\rangle + \delta\,|0011\rangle, \tag{5.6}$$

where the two additional terms describe the two electrons being on the same site (left or right). This wavefunction, which has a physical meaning to us, can be encoded in a four-qubit register. In this encoding, the spin-z and particle number conservations laws, which come from the physical theory, shows up in constraints in the number of '1's in the bit-strings

[1]the usual normalization condition on the coefficients is implied.

and their placement[2]

Turns out that this register can also store the electronic wavefunction of a H_2 molecule, in the minimal basis made of two spacial molecular orbitals: ϕ_g (this in turn constructed with the symmetric combination of the two localized atomic orbitals) and ϕ_u, in chemistry jargon. The state $|1100\rangle$ state describes the so called Hartree-Fock state, where both electrons occupy the lowest energy (spacial) molecular orbital ϕ_g.

The important fact is that, despite introducing some symmetries which limit the size of the Hilbert space, the computational cost of a fermionic simulations still scales exponentially. Indeed, if we consider M spin-orbitals and N electrons, there are combinatorially many, $\binom{M}{N}$, ways to place N electrons on M (spin-)sites. In principle, a quantum state that may specify the ground state (or a time-evolved state) of a fermionic hamiltonian, requires a linear combination of exponentially many configurations, each being a M-bit strings, with N '1's and $M-N$ '0's. This state **can compactly be stored in superposition** by a M qubit quantum register instead.

This is the main argument in favor of performing quantum chemistry using a quantum computer, that here we put forward without having to introduce much quantum chemistry jargon.

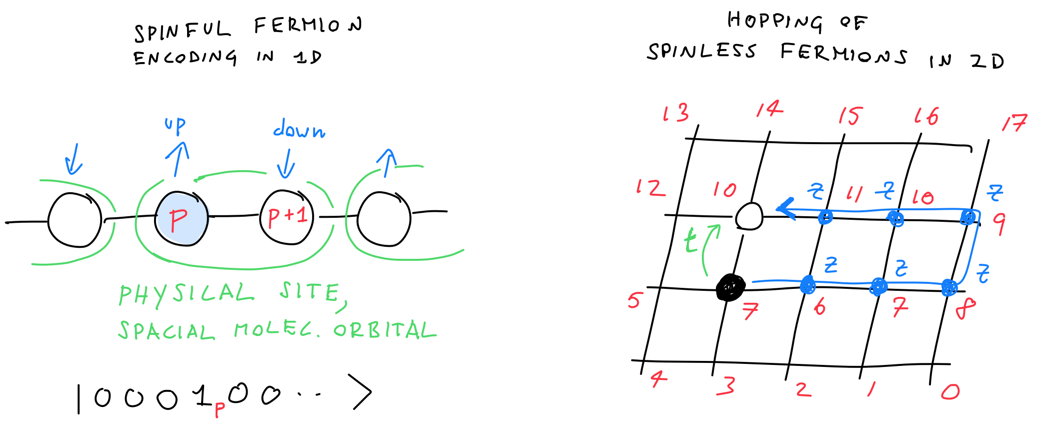

SPINFUL FERMION ENCODING IN 1D
up
down
p
p+1
PHYSICAL SITE, SPACIAL MOLEC. ORBITAL
$|0001_p00\cdots\rangle$
HOPPING OF SPINLESS FERMIONS IN 2D
13
14
15
16
17
12
10
11
10
9
t
z
5
7
6
7
8
4
3
2
1
0

Figure 5.1: Left: A possible convention for encoding a Hubbard model in 1D. Each physical lattice site can host two fermionic species, whose spin-orbital are labeled with adjacent integer. The occupied spin-orbital p is translated into an '1' at bit p of the corresponding encoded state. Right: Hopping of a spinless (for simplicity) fermion from site 7 to site 10, through the action of the operator $t\, a^\dagger_{10} a_7$. The two lattice sites are close in space, but distant in the chosen ordering. It is not possible to define an ordering that preserves the locality of all four hopping terms to the mapped qubit operators.

.

[2]for instance the basis state $|1010\rangle$ would not be allowed since it encodes two *up* spins.

5.1.2 Jordan-Wigner transformation

After the definition of the encoding, i.e the mapping between a 'physical' quantum state into a quantum state which live in a $M-$qubit Hilbert space, we need to translate the physical fermionic Hamiltonian, Eq. 5.1 into a qubit Hamiltonians, like the ones we defined generically in Chapter 4. Recall that a qubit is not a fermionic particle.

Given our binary encoding of a Fock state, it would be tempting to define the creation operator as the spin-1/2 raising operator,

$$a_p^\dagger \overset{?}{=} \sigma_p^- = |1\rangle\langle 0| = \frac{1}{2}(X_p - iY_p) = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}. \tag{5.7}$$

Notice here that we need to use opposite convention compared to the spin-1/2 *raising* operator, as $\sigma^- |0\rangle = |1\rangle$, while $\sigma^+ |1\rangle = |0\rangle$.[3]

This definition is however not correct exactly because the Pauli matrices do not satisfy the fermionic anticommutation relation. In short, the order in which we *create* a particle in the Fock state matters:

$$a_1^\dagger a_2^\dagger |00\rangle = |11\rangle\,, \quad a_2^\dagger a_1^\dagger |00\rangle = -|11\rangle\,, \tag{5.8}$$

because in the second case, we create the fermion in the second orbital, and after that, we create the first in orbital one. This purely fermionic constraint is not captured by the operator $\sigma_i^- \sigma_j^-$. Notice that, the encoding defines an *ordering* of the spin-orbital in the Fock state, and this is essential to satisfy the commutation relation, so in practical calculations, the ordering must be consistent along the steps.

In a more general fashion, if we denote a Fock state with M fermionic modes as

$$|\psi\rangle = |f_{M-1}, \dots, f_1, f_0\rangle \tag{5.9}$$

where $f_p = \{0, 1\}$ can only take binary values, then (single-particle) fermionic creation and annihilation operators are formally defined as

$$\begin{aligned} a_p^\dagger |f_{M-1}, \dots, f_0\rangle &= \delta_{f_p,0}(-1)^{\sum_{i=0}^{p-1} f_i} |f_{M-1}, \dots, f_p \oplus 1, \dots, f_0\rangle \\ a_p |f_{M-1}, \dots, f_0\rangle &= \delta_{f_p,1}(-1)^{\sum_{i=0}^{p-1} f_i} |f_{M-1}, \dots, f_p \oplus 1, \dots, f_0\rangle\,, \end{aligned} \tag{5.10}$$

where the first delta, i.e. $\delta_{f_p,0}$ ensures that we can create a particle, only if the spin-orbital p is empty.[4] and the phase term $(-1)^{\sum_{i=0}^{p-1} f_i}$ restores the correct sign by counting the number of occupied modes *before* the orbital p. The spin-orbital occupation operator is given by

$$\hat{n}_i = a_i^\dagger a_i, \tag{5.11}$$

and counts the number of fermions in a given fermionic mode.

[3]this is due to the convention $|\uparrow\rangle = |0\rangle, |\downarrow\rangle = |1\rangle$. To increase the occupation from 0 to 1, we need to actually use the lowering spin-1/2 operator.

[4]$\oplus$ denotes addition modulo 2 and it is just a general notation to ensure that $1 \oplus 1 = 0$.

To match these definitions, the fermionic operators, written in terms of qubit operators needs, read

$$\begin{aligned} a_p &= \frac{1}{2}(X+iY)_p \otimes Z_{p-1} \otimes \cdots \otimes Z_0, \\ a_p^\dagger &= \frac{1}{2}(X-iY)_p \otimes Z_{p-1} \otimes \cdots \otimes Z_0, \end{aligned} \tag{5.12}$$

where the $\frac{1}{2}(X+iY)_p$ and $\frac{1}{2}(X-iY)_p$ spin *minus* and *plus* operators change the occupation number of the target mode p, while the string of Z operators introduce the phase factor $(-1)^{\sum_{i=0}^{p-1} f_i}$. This convention is called **Jordan-Wigner mapping**, and maps any second quantised fermionic Hamiltonian into a linear combination of products of single-qubit Pauli operators

$$H = \sum_j h_j P_j \tag{5.13}$$

where h_j is a real scalar coefficient. Each term P_j in the Hamiltonian is typically referred to as a *Pauli string*. In this way we recover the general assumptions on the format of physical Hamiltonians.

The presence of the strings of operators may be unnoticed for local operators, such as the density at site i, since

$$n_i = a_i^\dagger a_i = \sigma_i^- \big(Z_{i-1}\cdots Z_0\big)\sigma_i^+ \big(Z_{i-1}\cdots Z_0\big) = \sigma_i^- \sigma_i^+ = \frac{1}{2}(I - Z_i), \tag{5.14}$$

where the strings $\big(Z_{i-1}\cdots Z_0\big)$ coming from the two operators cancel out since $Z_j Z_j = I$, for each $j < i$. The situation would be different for operators such as $a_i^\dagger a_j$, where i, j are not consecutive integer labels in the chosen ordering. This will lead to strings of Z's:

$$a_i^\dagger a_j = \sigma_i^- \big(Z_{i-1}\cdots Z_{j+1}\big)\sigma_j^+ = \frac{1}{4}(X - iY)_i \big(Z_{i-1}\cdots Z_{j+1}\big)(X+iY)_j \tag{5.15}$$

therefore to *four* Pauli strings, after expressing σ_i^+, σ_i^- as combination of X and iY.[5] Overall, the Jordan-Winger mapping introduces non-locality even if the parent fermionic hamiltonian is local, this can be seen already by considering a 2D lattice hamiltonian with only hopping terms (see Fig. 5.1).

5.1.3 Circuits for hamiltonian simulations

In this Section, we explicitly write the circuits corresponding to some fermionic operators, for real-time dynamics. The simplest case is the one-body operator

$$\exp\big(-i\ t\ h_{pp} a_p^\dagger a_p\big). \tag{5.16}$$

This operator will simply multiply by a phase $e^{-ith_{pp}}$ the state if the orbital p is occupied and will do nothing if p is empty. This operator remains "local" also in the qubit operator form a can be implemented by a single-qubit gate, the P gate[6] defined in Eq. 2.2.1.

[5] The following equalities holds $Z\sigma^+ = \sigma^+$, $\sigma^+ Z = -\sigma^+$, $Z\sigma^- = -\sigma^-$ and $\sigma^- Z = \sigma^-$. The first one explains why the Z_j operator is not present in Eq. 5.15

[6] we defined this gate with a positive phase, so here we put a minus sign to the angle.

The second operator we consider is the *hopping* operator. We consider directly a generic operator, plus is hermitean conjugate

$$\exp\big(-i\ t\ h_{pq}(a_p^\dagger a_q + a_q^\dagger a_p)\big). \tag{5.17}$$

Let's first consider each type of hamiltonian term, and then its exponentiation. Following Eq. 5.15 each term in the exponent produces four Pauli strings, for a total of eight. Four of them cancel out due to the alternating sign of the XY combination, and the other four terms sum up to

$$\frac{h_{pq}}{2}\left[X_p(Z_{p-1}\cdots Z_{q+1})X_q + Y_p(Z_{p-1}\cdots Z_{q+1})Y_q\right]. \tag{5.18}$$

At first glance, the exponentiation of the sum of these two operators would imply a Trotter error. However, and this is very rarely discussed in literature, the two terms commute[7], such that the circuit in Fig. 5.2(b) perform the exact time evolution of Eq. 6.1.1.

The more frequent kind of operator is

$$\exp\big(-i\ t\ h_{pqrs}(a_p^\dagger a_q^\dagger a_r a_s + a_r^\dagger a_s^\dagger a_p a_q)\big), \tag{5.19}$$

which in the language of chemistry, is responsible for *double-excitations.* In general, there are M^4 of such terms, so they take most of the computational time. Following the same ideas, each term produces a sum of eight Pauli strings operators

$$\frac{h_{pqrs}}{8}\left[\mathcal{M}_p(Z_{p-1}\cdots Z_{q+1})\mathcal{M}_q\ \ \mathcal{M}_r(Z_{r-1}\cdots Z_{s+1})\mathcal{M}_s\right] \tag{5.20}$$

where the set of $(\mathcal{M}_p, \mathcal{M}_q, \mathcal{M}_r, \mathcal{M}_s)$ Paulis can take eight different combination with an even number of Y elements, e.g (X, X, X, X), (X, X, Y, Y), $(X, Y, X, Y), \cdots, (Y, Y, Y, Y)$ (see also Eq. 5.24 for a concrete example). The circuit for one of such term is shown in Fig. 5.2(c). More circuits, and concrete examples can be found in the seminal Refs. ([2]).

Notice that, in the worst-case scenario two *ladders* of CNOTS gate may extend along all the M-qubit register. The total number of gates, for hamiltonian simulations of quantum chemistry in second quantization is $\mathcal{O}(M^5)$, since there are $\mathcal{O}(M^4)$ terms in the qubit Hamiltonian and these terms require $\mathcal{O}(M)$ gates, exactly because the Jordan-Wigner transformation introduce non-locality.

This scaling is worst case, and can be improved by optimal compilation and using hamiltonian symmetries. However, when applied to hamiltonian dynamics, it still enables an exponential advantage compared to exact classical simulators.

5.1.4 A worked example: an hydrogen-like molecule with four orbitals

Staying true to the spirit of these notes, we derive a qubit Hamiltonian for a simple yet not entirely trivial example of a fermionic Hamiltonian.

[7]You can check that, for instance, while X and Y do not commute, XX and YY do. It is not obvious to see but two Pauli strings commute if they fail to commute at an even number of indices. This condition is called General commutativity([1]), and generalize the qubit-wise commutativity rule, where two strings commute only if all Paulis commute at each index, e.g XI and XZ.

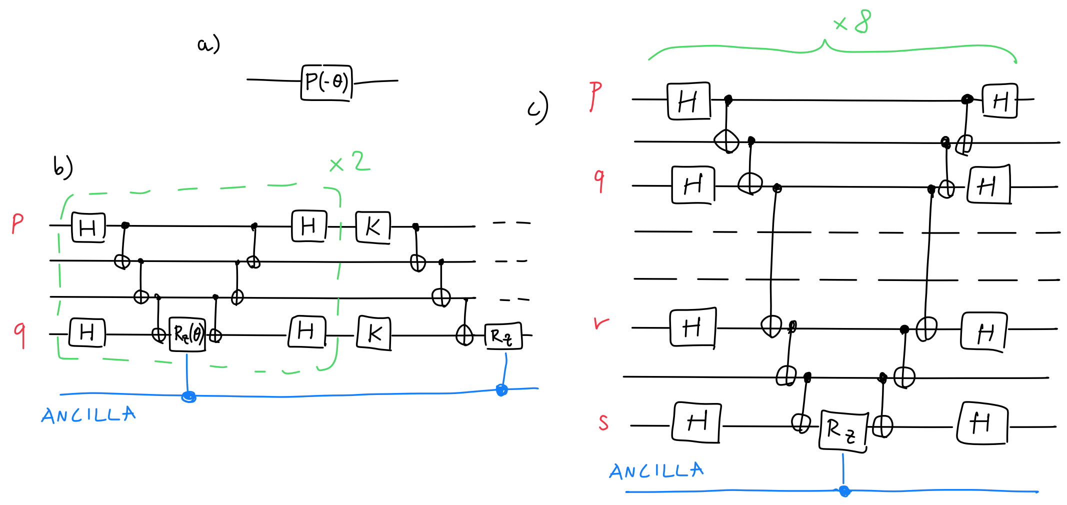


Figure 5.2: a) Single qubit circuit for the density operator. b) Multi qubit circuit for the hopping term h_{pq}. For clarity we only draw the first of the two terms (XX), and only half of the (YY) one. **If used in QPE**, these blocks needs to be 'controlled' so we also add here an ancilla qubit (blue) that controls the application of the unitary. To control the whole unitary it is sufficient to control only the R_Z gate. (see Chapter 2) c) Multi qubit circuit for the h_{pqrs} double excitation term. For clarity we only draw one of the eight similar terms, i.e. the $(XXXX)$ term.
.

Let's imagine that we have four spin orbitals, and we assume a staggered ordering as follows

$$|\phi_0^\uparrow, \phi_0^\downarrow, \phi_1^\uparrow, \phi_1^\downarrow\rangle = |f_0, f_1, f_2, f_3\rangle. \tag{5.21}$$

To be more concrete, this could describe a hydrogen molecule using two spatial molecular orbitals, and therefore, 4 spin orbitals. Where the first two positions are reserved for the lower-energy molecular orbital and the other two for the higher-energy one. We assume that an independent electron calculation has been done previously to determine the form of the orbitals. Assuming this convention $|1,1,0,0\rangle$ would be the Hartree-Fock state, where both electrons occupy the lowest energy spatial orbital ϕ_0. Even more explicitly, the occupation number f_3 of the third fermionic mode means physically that one *down* electron occupy the spin orbital $\phi_1^\downarrow$.

However, these details are not important for now. At the moment, it suffices for us to assume that the occupation of the spin orbitals follows this precise ordering, and from this, the labeling of the hopping operators of the Hamiltonian follows. Since an electron can only hop between spin orbitals with the same spin, the Hamiltonian can only contain terms like $a_2^\dagger a_0$ or $a_3^\dagger a_1$, but not $a_1^\dagger a_0$. Other selection rules (conservation of magnetization) determine that there can be only one type of four-body operator: $a_3^\dagger a_2^\dagger a_1 a_0$, plus its hermitean conjugate $a_0^\dagger a_1^\dagger a_3 a_2$.

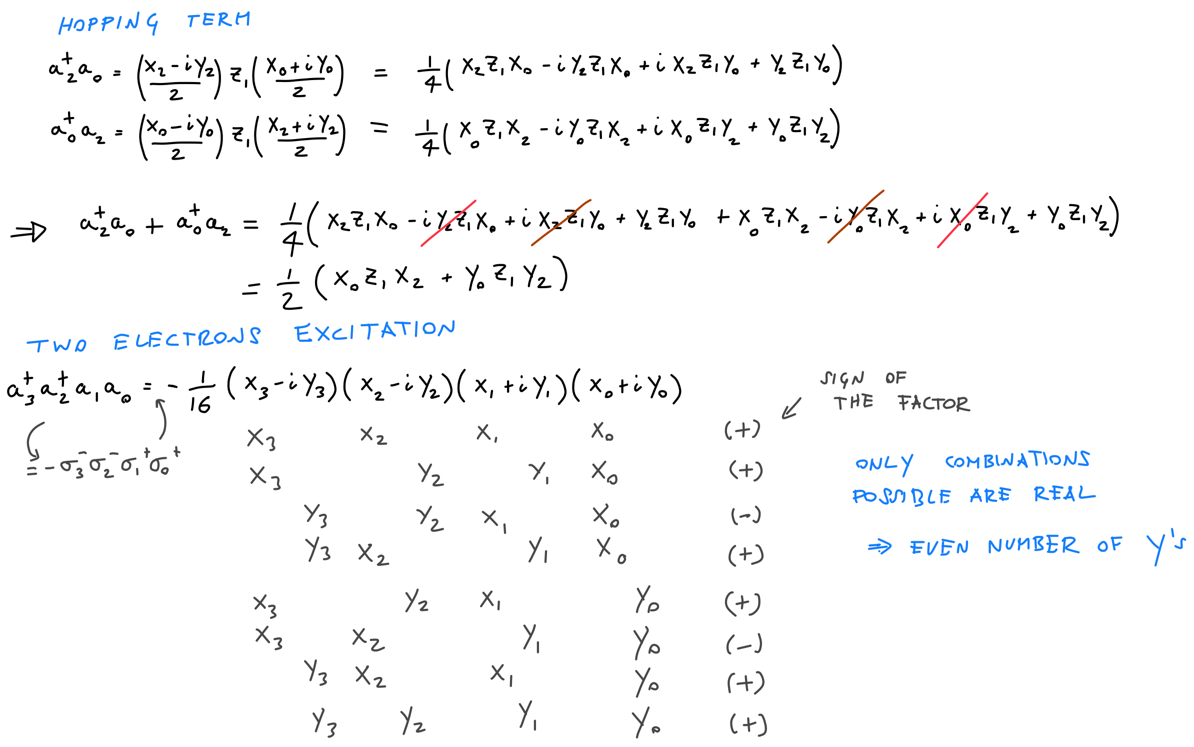


Figure 5.3: Upper panel. Jordan Wigner transformation of one of the hopping terms. Each $a_i^\dagger a_{i+1}$, with $i = 0, 1$, produces four terms. Two of them cancel out when we consider the hermitean conjugate $a_{i+1}^\dagger a_i$. Therefore each term $a_{i+2}^\dagger a_i + a_i^\dagger a_{i+2}$ results into the sum of two Pauli strings, $X_i Z_{i+1} X_{i+2} + Y_i Z_{i+1} Y_{i+2}$. Notice again that the e.g. the string $\sigma_3^- Z_2 Z_1 \sigma_1^+ = \sigma_3^- Z_2 \sigma_1^+$ because $Z\sigma^+ = \sigma^+$. Bottom panel. Sketch of the calculation for the four-indices operator. In this case, 16 Pauli strings are generated by the operator $a_3^\dagger a_2^\dagger a_1 a_0$. It is possible to see that the summation with its hermitean conjugate, $a_0^\dagger a_1^\dagger a_3 a_2$, eliminates all the complex valued terms (similarly to the above case), so we will select only the combination with an even number of Y elements. There are 8 possibilities.

An artificial hamiltonian which conserves charge and magnetization can read

$$H = a_2^\dagger a_0 + a_0^\dagger a_2 + a_3^\dagger a_1 + a_1^\dagger a_3 + a_3^\dagger a_2^\dagger a_1 a_0 + a_0^\dagger a_1^\dagger a_3 a_2 \tag{5.22}$$

where we set all coefficient h's to one for simplicity. This is a specific instance of the general Eq. 5.1.

Let's first analyze how the hopping terms transform under the action of the Jordan-Wigner transformation. The steps are shown in Fig. 5.3. Each of the single electron excitation operator transforms into

$$a_{i+2}^\dagger a_i + a_i^\dagger a_{i+2} = \frac{1}{2}\big(X_i Z_{i+1} X_{i+2} + Y_i Z_{i+1} Y_{i+2}\big), \tag{5.23}$$

with $i = 0, 1$. The double electron operator instead generates 8 Pauli strings,

$$a_3^\dagger a_2^\dagger a_1 a_0 + a_0^\dagger a_1^\dagger a_3 a_2 = -\frac{1}{8}\big(X_3X_2X_1X_0 + X_3Y_2Y_1X_0 - Y_3Y_2X_1X_0 + Y_3X_2Y_1X_0 \\ X_3Y_2X_1Y_0 - X_3X_2Y_1Y_0 + Y_3X_2X_1Y_0 + Y_3Y_2Y_1Y_0\big). \tag{5.24}$$

The systematic way to decompose the $a_3^\dagger a_2^\dagger a_1 a_0$ is as follows,

$$a_3^\dagger a_2^\dagger a_1 a_0 = \sigma_3^- Z_2 Z_1 Z_0 \ \sigma_2^- Z_1 Z_0 \ \sigma_1^+ Z_0 \ \sigma_0^+ \tag{5.25}$$
$$= \sigma_3^- \otimes Z_2\sigma_2^- \otimes Z_1 Z_1 \sigma_1^+ \otimes Z_0 Z_0 Z_0 \sigma_0^+ \tag{5.26}$$
$$= \sigma_3^- \otimes Z_2\sigma_2^- \otimes \sigma_1^+ \otimes Z_0\sigma_0^+ = \sigma_3^- \otimes (-\sigma_2^-) \otimes \sigma_1^+ \otimes \sigma_0^+ . \tag{5.27}$$

At this point one case safely apply the definition of $\sigma_p^\pm$ and obtain the combination of Pauli strings, with the correct overall sign, see Fig. 5.3.

5.2 Bosonic hamiltonians

Interestingly, simulating bosons on a quantum computer is not much easier than fermions. Notice that a spin-1/2 object is not a boson, since it can exist in only two states, up or down in the compuational basis, while a bosonic degrees of freedom can be infinitely excited. Namely, the occupation number of bosonic mode can be arbitrary, or equivalently, a lattice site can host infinite bosons[8].

Therefore, we need to *truncate* the Hilbert space of a single boson, to a finite d-level system to perform the calculation. In this section we are not going to discuss in details all possible multi-particle bosonic operators, and we focus only on the building blocks, the creation and destruction operator.

The computational cost to perform these basic operations strongly depends on the encoding used, so we will first encounter an instance of the *memory-time* tradeoff dilemma, which is ubiquitous in quantum software development.

5.2.1 Unary, one-hot encoding

Let's consider the occupation number basis, and **just one bosonic mode**. The first way to encode such a state, $|s\rangle$ with $s = 0, 1, \dots, d-1$, is by using a seemingly inefficient mapping, the so-called *unary* or one-hot encoding. This mapping requires d qubit per bosonic mode, and is defined as

$$|s\rangle = \otimes_{j=0}^{s-1} |0\rangle_j \, |1\rangle_s \otimes_{j=s+1}^{d-1} |0\rangle_j \,, \tag{5.28}$$

where basically to label the state $|0\rangle$ we use a string of qubits with only the first one set to '1', $|0\rangle \to |100\dots00\rangle$, $|1\rangle \to |010\dots00\rangle$, and so on, up to $|d\rangle \to |000\dots01\rangle$.

The creation and destructions operators, in the occupation number basis[9] is defined as

$$b^\dagger \,|s\rangle = \sqrt{s+1}\,|s+1\rangle \,, \tag{5.29}$$
$$b \ \ |s\rangle = \sqrt{s}\,|s-1\rangle \tag{5.30}$$

[8] a spin-1/2 can be seen as an hardcore-boson, which occupation number can be 0 or 1.

[9] here we consider only one bosonic mode, so we drop the "site" index, i.e. $b_j^\dagger$.

so in this encoding it translates to

$$b^\dagger = b^\dagger_{\text{unary}} = \sum_{s=0}^{d-2} \sqrt{s+1}\big(\left|0\right\rangle\left\langle 1\right|\big)_s \otimes \big(\left|1\right\rangle\left\langle 0\right|\big)_{s+1}. \tag{5.31}$$

This operator, in qubit space, simply swaps the '0' and '1' values between the qubit at position s and $s+1$. This can be efficienly performed using $\mathcal{O}(d)$ Pauli operators, taking into account the following equalities, that maps projectors into Paulis

$$\begin{aligned}
\left|0\right\rangle\left\langle 0\right| &= \frac{1}{2}(I+Z),\\
\left|1\right\rangle\left\langle 1\right| &= \frac{1}{2}(I-Z),\\
\left|1\right\rangle\left\langle 0\right| &= \frac{1}{2}(X-iY),\\
\left|0\right\rangle\left\langle 1\right| &= \frac{1}{2}(X+iY).
\end{aligned} \tag{5.32}$$

For instance, the transition from a single configuration basis state $|s\rangle$ to $|s+1\rangle$ requires only to the action of σ^- on qubit s and σ^+ on qubit $s+1$, irrespective of the total register size d. The annihilation operator can be obtained by taking the Hermitian conjugate of $b^\dagger_{\text{unary}}$.

A notable operator in bosonic hamiltonians is the *position* operator, which in this encoding simply reads

$$(b^\dagger + b) = \frac{1}{2}\sum_{s=0}^{d-2} \sqrt{s+1}\ X_s X_{s+1} + Y_s Y_{s+1}. \tag{5.33}$$

The sum can be divided into two terms, s-even and $s-$odd. All terms within each block commute for the same reason explained in the above section. Each unit $XX+YY$ can be compiled as shown in Fig. 5.2(b), with $p=s$ and $q=s+1$, i.e without the ladder of CNOTS.

5.2.2 Binary, compressed encoding

As an alternative to the direct mapping, we can use a *binary mapping*, using $k=\lfloor \log d \rfloor$ qubits,

$$|s\rangle = |i_{k-1}\rangle\,|i_{k-2}\rangle \ldots |i_0\rangle\,, \tag{5.34}$$

with binary representation $s = i_{k-1}2^{k-1} + i_{k-2}2^{k-2} + \ldots i_0 2^0$. Again, this holds for just one bosonic mode. In this case the compilation of the creation/destruction operators is more cumbersome. Each multi-qubit projector in Eq. 5.29 needs to be expressed as a sum of tensor product of Paulis using Eq. 5.32. The support of a generic operation $\left|s+1\right\rangle\left\langle s\right|$ now scales with k, as in the worst case scenario the addition involve a global rearrangements of the bits.

We explicitely derive the mapping of a position operator to show how this is done in practice. Let's consider the case $d=4$, which means $k=2$ qubits:

$$b^\dagger + b = \left|01\right\rangle\left\langle 00\right| + \left|00\right\rangle\left\langle 01\right| + \sqrt{2}\big(\left|10\right\rangle\left\langle 01\right| + \left|01\right\rangle\left\langle 10\right|\big) + \sqrt{3}\big(\left|11\right\rangle\left\langle 10\right| + \left|10\right\rangle\left\langle 11\right|\big). \tag{5.35}$$

Now, let's only focus on the first two terms

$$
\begin{align}
&|01\rangle\langle 00| + |00\rangle\langle 01| = \tag{5.36}\\
&\big(|0\rangle\langle 0|\big) \otimes \big(|1\rangle\langle 0|\big) + \big(|0\rangle\langle 0|\big) \otimes \big(|0\rangle\langle 1|\big) = \tag{5.37}\\
&\frac{1}{4}\big(I+Z\big) \otimes \big(X - iY\big) + \frac{1}{4}\big(I+Z\big) \otimes \big(X + iY\big) = \tag{5.38}\\
&\frac{1}{4}\big(I\otimes X + Z\otimes X - iZ\otimes Y - iI\otimes Y + I\otimes X + Z\otimes X + iZ\otimes Y + iI\otimes Y\big) = \tag{5.39}\\
&\frac{1}{2}\big(I\otimes X + Z\otimes X\big) := \frac{1}{2}\big(IX + ZX\big). \tag{5.40}
\end{align}
$$

Notice that this Pauli decomposition is specific to the $|01\rangle\langle 00| + |00\rangle\langle 01|$ operator, and other terms need to be compiled separately.

5.3 First-quantized hamiltonians

In Section 5.1 we introduced fermionic hamiltonians in their (standard) second-quantization formalism. We see that, second quantized hamiltonian imply an efficient encoding in terms of occupation number states. However, physics and chemistry problems can be simulated beyond this formalism, i.e. directly in their real-space representation.

Concerning molecular systems there are several advantages of the real-space approach: *1.* while second-quantized simulation assumes the Born-Oppenheimer approximation with electronic orbitals chosen for fixed nuclear positions, first quantized hamiltonians treat electrons and ions at the same footing allowing for direct non-adiabatic simulations of chemical reaction. *2.* In addition, real-space simulations are basis-set free thus allowing a more systematic extrapolation toward the exact result. *3.* Real-space simulations incorporate the principle of physical locality. One way to see this is that, while the real-space Coulomb interaction is a *two* index electron-electron interaction, the corresponding second-quantized term is a *four* body operator. As we saw, simulating the real-time dynamics of a fermionic chemistry hamiltonians using M orbitals, with $M \propto N$, comes with a cost $\mathcal{O}(M^5)$, because there are M^4 terms to evolve, and each of them requires $\mathcal{O}(M)$ gates. On the other hand, first-quantized hamiltonians features N^2 terms, where N is the number of electrons. Morevover, first-quantized operators do not require anti-symmetrization, so one does not need a Jordan-wigner trasformation. Therefore, hamiltonian simulations of fermionic systems in first quantization can be asymptotically faster than the standard second-quantized form.

The drawbacks, in case of fermion are that, (1) the wavefunction must be antisymmetrized in the real-space basis, (2) the number of qubits to represent the position of the electrons with a reasonable space-discretization error is challenging even for H_2, and (3) compiling the Coulomb interactions $1/r$ into gates requires costly quantum arithmetic operations.

However, it is instructive to see at least one simple example of enconding of a first-quantized hamiltonian in a qubit register. In this Section we will put forward the simplest case of a single-particle evolution in a simple potential $V(x)$.

5.3.1 Wave-packet propagation in one dimension

Let us consider a quantum particle in one dimension. The evolution is given by a Schrödinger equation

$$i\frac{d}{dt}\psi(x,t) = H\psi(x,t), \tag{5.41}$$

where the Hamiltonian H is given by

$$H = K + V(x) = -\frac{1}{2m}\frac{d^2}{dx^2} + V(x)\,. \tag{5.42}$$

The kinetic term $K = -(1/2m)\, d^2/dx^2$ governs the free motion of the particle, while $V(x)$ is a one-dimensional potential.

The first step enabling the encoding is to discretize the continuous variables x. If the motion essentially takes place inside a finite region, say $-L \leq x \leq L$, we decompose this region into 2^n intervals of length $\Delta = 2L/2^n$, using a n-qubit quantum register. Each discrete position x_i:

$$x_i \equiv -L + \left(i + \tfrac{1}{2}\right)\Delta, \tag{5.43}$$

is mapped to an integer number i, that can be represented using the usual binary encoding: $|i\rangle = |i_{n-1}\rangle|i_{n-2}\rangle \ldots |i_0\rangle$ as a state of the computational basis of the n-qubit quantum register.

Hence, the wave function $\psi(x,t)$ is defined on the register as by

$$|\psi\rangle = \sum_{i=0}^{2^n-1} c_i(t)\,|i\rangle = \frac{1}{\mathcal{N}}\sum_{i=0}^{2^n-1}\psi(x_i,t)\,|i\rangle\,, \tag{5.44}$$

where is a factor that ensures correct normalization of the wave function[10] We can forget about this factor, and just take into account its existence in postprocessing.

The Hamiltonian operator would feature an exponentially scaling number of Pauli strings. So we need an alternative way to perform the blocks e^{iKt} and e^{iVt}, in a Trotter scheme, since V and K do not commute. Let's start from the potential operator, which is diagonal in the computational basis. Given a state $|i\rangle$,

$$e^{iVt}\,|i\rangle = e^{iV(x_i)t}\,|i\rangle\,. \tag{5.45}$$

In a general case, where $V(x_i)$ is simply a mapping from $i \to V_i$, one would need 2^n multi-controlled phase rotations $|i\rangle\langle i|\, e^{iV_i t}$ to assign the correct phase to each component. However, if $V(x_i)$ is a polynomial function, the operation $e^{iV(x_i)t}$ requires only a polynomial number (in n) of gates. For instance, $V(x) = \alpha x^2$ requires $\mathcal{O}(n^2)$ controlled rotations, as explained in Ref. [3].

The unitary evolution induced by the kinetic operator can be performed using the same circuit of the $e^{i\alpha x^2}$ operator, and a change of basis from the position to the momentum

[10] basically there is the element of measure $\sqrt{\Delta}$ that connects the two normalizations.

space. We call p the momentum variable conjugate to x, that is, $-i(d/dx) = \mathsf{QFT}^{-1} p \mathsf{QFT}$, where QFT is the Fourier transform. Therefore,

$$e^{-iKt} = \mathsf{QFT}^{-1}\, e^{-i\left(\frac{p^2}{2m}\right)t}\, \mathsf{QFT}\,. \tag{5.46}$$

In this expression, we pass, by means of the Fourier transform QFT, from the x-representation to the p-representation, in which this operator is diagonal. The quantum Fourier transform maps a generic n-qubit state $\sum_{k=0}^{N-1} f(k)|k\rangle$ into $\sum_{l=0}^{N-1} \tilde{f}(l)|l\rangle$, where $N = 2^n$ and the vector $\{\tilde{f}(0), ..., \tilde{f}(N-1)\}$ is the discrete Fourier transform of the vector $\{f(0), ..., f(N-1)\}$, that is,

$$\tilde{f}(l) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i k l/N} f(k). \tag{5.47}$$

It can be shown that this transformation can be efficiently implemented in $O((\log_2 N)^2)$ quantum gates, while the best known classical algorithm, the fast Fourier transform, requires $O(N \log_2 N)$ elementary operations.[4]

Bibliography

[1] P. Gokhale, O. Angiuli, Y. Ding, K. Gui, T. Tomesh, M. Suchara, M. Martonosi, and F. T. Chong, "Minimizing state preparations in variational quantum eigensolver by partitioning into commuting families," *arXiv preprint arXiv:1907.13623*, 2019.

[2] J. D. Whitfield, J. Biamonte, and A. Aspuru-Guzik, "Simulation of electronic structure hamiltonians using quantum computers," *arXiv preprint arXiv:1001.3855*, 2010.

[3] G. Benenti and G. Strini, "Quantum simulation of the single-particle schrödinger equation," *American Journal of Physics*, vol. 76, no. 7, pp. 657–662, 2008.

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

Chapter 6

Variational algorithms

Variational algorithms form the backbone of computational physics. When considering chemistry or many-body quantum problems, a variational description almost always provides a satisfactory understanding of the problem, which can then be refined using projection methods.

In this chapter, we will attempt to treat variational algorithms in the most general manner possible. Although deeply rooted in chemical and physical applications, this method also finds its place in quantum machine learning. In practice, quantum neural networks are nothing but glorified variational circuits, where, instead of optimizing an energy, we optimize a loss function.

Every quantum variational algorithm conceptually consists of two parts: the first part involves the existence of a variational ansatz, determined by the choice of the circuit and where the variational parameters enter. The second part is the optimization procedure for these variational parameters, aiming to minimize a target cost function. The optimization is managed through a classical feedback loop. For this reason, such algorithms are often referred to as hybrid quantum-classical.

In conventional many-body physics and chemistry, there are numerous variational states. For example, the BCS wavefunction used in superconductivity, the Jastrow wavefunction for liquid helium, or the configuration interaction method in quantum chemistry. In some cases, the variational parameters within the ansatz can be optimized analytically, but in the most general cases, they need to be optimized numerically (as is the case for quantum variational algorithms).

6.1 Variational quantum eigensolver

Variational quantum computation features parametrized circuits that produce a trial state $|\psi(\boldsymbol{\theta})\rangle$. Its N_p parameters, $\boldsymbol{\theta} = (\theta_1, \cdots, \theta_{N_p})$, are adjusted at every step following an iterative classical procedure. Usually the parameters $\boldsymbol{\theta}$ enters in the circuit as rotation angles, this is evident in the case of hardware efficient circuits, but it is also true for hamiltonian inspired circuits (cfn. Sect. 5.1.3). Different hardware architectures come with different sets of native gates, so in this chapter we try to maintain an high-level description of the algorithm. This algorithm usually takes the name of variational quantum eigensolver (VQE)([1; 2]). The goal is to minimize a cost function $C_{\boldsymbol{\theta}} = \langle\psi_{\boldsymbol{\theta}}| \hat{O} |\psi_{\boldsymbol{\theta}}\rangle$, where $\hat{O}$ is an operator that describes the problem we are interested. For chemical systems, $\hat{O}$ is the full Hamiltonian of the problem (qubit operator), after that a suitable physics-to-qubit mapping has been performed (see Ch. 5. In this case, at the end of a successful optimization, we have the optimized parameters $\boldsymbol{\theta}^*$, such that $|\psi(\boldsymbol{\theta}^*)\rangle$ should be a close approximation of the ground stae of the problem, provided that the variational ansatz is able to reach the ground state. Formally

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} C_{\boldsymbol{\theta}}. \tag{6.1}$$

It is important to notice that we already showed a method to calculate energy using real-time dynamics and QPE in Sec. 4.5. However, that approach requires the possibility to implement arbitrarily long Hamiltonian simulation circuits, thus it can only be implemented in the fault-tolerant regime. Variational methods are, in principle, amenable to NISQ implementation because the circuit should be much shorter. Moreover, the QPE-based method requires, as a starting point, a good quantum state, i.e., one with a large overlap with the ground state. Variational methods could provide such a starting point.

Considerations for Quantum Advantage: It's important to state from the beginning that these algorithms do not have a proven quantum advantage, unlike Hamiltonian simulation. When considering physics or chemistry, the idea that a wave function can represent the ground state of a Hamiltonian better than any possible classical ansatz is an educated guess. But there is no a priori proof that an efficient circuit always exists to create this state or that optimization can lead us to it. This is also true for classical variational algorithms, such as variational quantum Monte Carlo. Only in hindsight can we verify whether the obtained energy (with a certain zero point fixed) is better or not.

Despite this necessary disclaimer, variational algorithms find such a wide range of applications that it is necessary to discuss them in these notes.

6.1.1 Circuit trial states

Strictly speaking, what is parameterized is not the state but the circuit, specifically the unitary operator $U_{\boldsymbol{\theta}}$. This requires establishing a convention regarding the initial state to which $U_{\boldsymbol{\theta}}$ is applied. Very often, this state is simply the zero state, but there are exceptions such as in QAOA. In chemistry, some trial ansatz start from the Hartree-Fock state, which is easy to prepare from the zero state. It simply involves applying a suitable number of X gates to create the bit string with the correct positions of 1's (remember that the 1s denote the occupation of the molecular orbital assigned to the given qubit, see Sect 5.1)

It is also important to note that any variational computation worthy of its name has a number of variational parameters that is smaller than the Hilbert space dimension. Unfortunately, in the literature, where small-scale examples are very popular, this condition is not always satisfied.

To be more concrete, we can define a couple of circuit ansatz that find application in chemistry, physics, and machine learning. Let's start with a heuristic ansatz, which has the advantage of being very compact but also has several drawbacks, as it lacks any structure that we can exploit to reduce the number of parameters or attribute them a physical meaning.

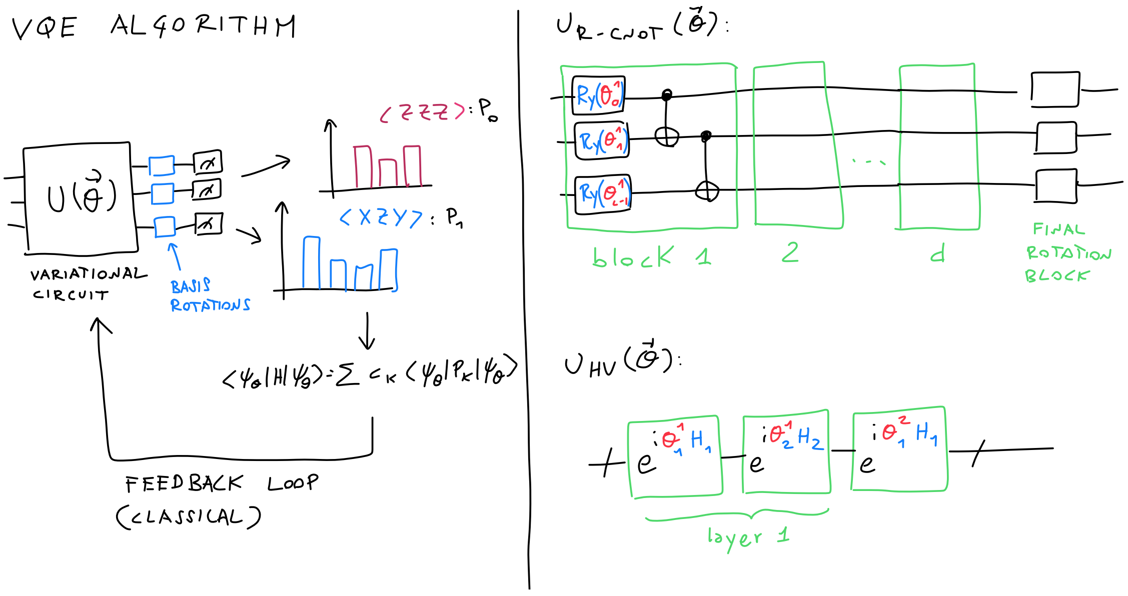


Figure 6.1: Left panel. The VQE algorithm features a parametrized circuit, which needs to be repeated many times (with appropriate post-rotations) to compute the expectation value of the energy (as a sum of expectation values of the Pauli strings). The parameters are optimized classically to lower the energy. Right panel. Two possible choices for the parametrized circuit. The heuristic R-CNOT and the hamiltonian variational.

As the capabilities of hardware improve, I predict and hope that this circuit will gradually be abandoned for chemistry. However, it may still find applications outside of VQE. For example, many quantum machine learning circuits are of this type.

Heuristic circuits. This type of circuit generally features entangling blocks made of a cascade of two-qubit gates, like the CNOT gates, and rotations blocks where the angles plays the role of parameters. The most commonly studied heuristic circuit is made of a series of d blocks built from single-qubit rotations $U_\mathrm{R}(\boldsymbol{\theta}^k)$, interlayered with an entangler block U_ent, that covers the whole qubit register (see Fig. 6.1) In our case, this block is made of a ladder of CNOTgates with linear connectivity, such that qubit q_i is target of qubit q_{i-1} and controls qubit q_{i+1}, with $i = 0, \cdots, L-1$, with open or closed boundary conditions, depending on the hardware topology[1] This choice is commonly used as it mirrors the current sparse qubit connectivity of today's hardware. The single-qubit rotations are all local operations, $U_\mathrm{R}(\boldsymbol{\theta}^k)$, so they can be written as a tensor product of single qubit rotations. A common choice for the rotation axis is Y, sicne this yields a real-amplitude ansatz, but diffent choices are possible. In this case, the ansatz is commonly labeled as R_Y-CNOTcircuits, and find applications in

[1] i.e. the qubit connectivity.

chemistry, optimization and machine learning. The rotation block is

$$U_{\mathrm{R}}(\boldsymbol{\theta}^k) = \bigotimes_{i=0}^{L-1} R_y(\vartheta^k_{q_i}), \tag{6.2}$$

where $R_y(\vartheta^k_{q_i})$ is a rotation on the y–axis on the Bloch sphere of qubit q_i, and $k = 1, \cdots, d+1$. Here, $\boldsymbol{\theta}^k$ denotes an array of L angles. The full unitary circuit operation is described by

$$U_{\text{R-CNOT}}(\boldsymbol{\theta}) = U_{\mathrm{R}}(\boldsymbol{\theta}^{d+1}) \overbrace{U_{\mathrm{ent}} U_{\mathrm{R}}(\boldsymbol{\theta}^d) \dots U_{\mathrm{ent}} U_{\mathrm{R}}(\boldsymbol{\theta}^1)}^{d\text{-times}}, \tag{6.3}$$

and the final parametrized state is

$$|\psi(\boldsymbol{\theta})\rangle = U_{\text{R-CNOT}}(\boldsymbol{\theta}) \left(|0\rangle^{\otimes L}\right). \tag{6.4}$$

The total number of variational parameter is $L(d+1)$.

Hamiltonian variational An alternative is to use physically motivated ansatzes, such as the Hamiltonian variational (HV) ansatz [3]. This ansatz is one of the most powerful for simulations of physical hamiltonians, which can be written generally as sum of non-commuting operators

$$H = \sum_{\alpha=1}^{\ell} H_\alpha \tag{6.5}$$

The unitary operator defining the HV ansatz is again made of d blocks, but this time each block is a product more structured operators, i.e. of ℓ operators $U_\alpha = \exp(\mathrm{i}\theta^k_\alpha H_\alpha)$, with $\alpha = 1, \dots, \ell$ indexing the non-commuting terms of the Hamiltonian. Suppose we want to find the ground state of a transverse field Ising model

$$H = \sum_i^L Z_i Z_{i+1} + \sum_i^L X_i, \tag{6.6}$$

then we only need $\ell = 2$, and the full unitary operator[2] is

$$U_{\mathrm{HV}}(\theta) := \prod_{i=d}^{1} U_2(\theta^i_2) U_1(\theta^i_1), \tag{6.7}$$

which can be efficiently decomposed using one- and two-qubit quantum gates, as shown in Sec. 4.3, and the final parameterized state is

$$|\psi(\theta)\rangle = U_{\mathrm{HV}}(\theta) \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)^{\otimes L}, \tag{6.8}$$

[2]the unconventional order of index i is just a trick to respect the from-right-to-left order of application of operators to a quantum state. The rightmost U is applied first.

where the initial non-entangled state can be obtained from $|0\rangle^{\otimes L}$ by placing one Hadamard gate on each qubit. The total number of parameters is ℓd.

Some considerations that may be obvious but are worth stressing: The heuristic circuit does not feature the hamiltonian operators, so it is totally not guaranteed to succeed in finding a good approximation of the ground state. Its only merit is that it can be run with short depth and therefore is feasible for early and today's hardware experiment. It also feature *a lot* of parameters, and no easy way to initialize them. This means that optimization, which is a classical loop, becomes impractical for large sizes. As the hardware becomes better, I expect that heuristic circuits will be abandoned in favour of more structured ones, such the HV.

From the point of view of circuit's compilation, the HV ansatz features by construction the same building blocks of a hamiltonian simulation circuit, just using variable, optimizable time-steps. In case of fermionic hamiltonians, each block takes the circuit forms explained in Sect. 5.1.3. It is clear that the HV circuits features less parameters, thus allowing us for an easier optimization, but they can be quite long, preventing us from a NISQ implementation.

One possible justification of a HV circuit is that, in the $d \to \infty$ limit, it can approach a *trotterized* version of an adiabatic state preparation procedure (cfn. Ch 3). However, since the ansatz depth is finite, and there may be an exponentially closing gap during this "digitized annealing", also in this case we are not guaranteed to find the exact ground state.

Other physics inspired circuits. Another possible strategy is, instead of trotterizng the full Hamiltonian, is to use pieces of that, or other physics operators, to generate the circuit. One example is the popular *unitary coupled cluster* anzatz([4]), which operator takes the form

$$U = e^{T - T^\dagger}, \tag{6.9}$$

where T is an excitation operator, that may contain several order of molecular orbitals excitation, for instance $T = T_1 + T_2$, where

$$T_1 = \sum_{i,n} t_{in} a_i^\dagger a_n, \qquad T_2 = \sum_{ij,nm} t_{ijnm} a_i^\dagger a_j^\dagger a_n a_m. \tag{6.10}$$

Here i, j label un-occupied orbitals, while n, m occupied ones, and the creation and destruction operators have been introduced in Ch. 5. Finally, t_{in} and t_{ijnm} are variational parameters. Notice that we already provided circuit decomposition of this kind of operators, which closely resemble Eqs 5.17 and 5.19, in Sect. 5.1.3. This time, there is a *minus* sign between the T_1 (or T_2) and its hermitean conjugate. This only means that, after the Jordan-Wigner transformation is applied, a different set of Pauli strings emerge from the cancellation. For instance, a generic operator of the form $e^{T_1 - T_1^\dagger}$, yields a Pauli operator of the form

$$\frac{it_{in}}{2} \left[X_i (Z_{i-1} \cdots Z_{n+1}) Y_n - Y_i (Z_{i-1} \cdots Z_{n+1}) X_n \right], \tag{6.11}$$

which features the Pauli strings that are absent in Ref. of Sect. 5.1.3. The variational parameters are, again, encoded in the rotation angles of the blocks depicted in Fig. 5.2. Overall, the unitary coupled cluster circuit looks very similar to a fermionic hamiltonian evolution circuit, and therefore is considered also not feasible in the NISQ era, for large molecules.

6.1.2 Energy and gradient based optimization

VQE finds applications in a multitude of fields. In these notes, we will specifically address the very popular case in which the cost function to be optimized is an energy derived from a many-body or quantum chemistry Hamiltonian. In this case, the operator $\hat{O}$ in the introduction is $\hat{H}$. And the final wave function must be the ground state of the Hamiltonian.

At this point, the search for optimal parameters is delegated to a classical optimization loop. In the general case, this is an iterative process, so at each step of the classical optimizer, new parameters theta $\boldsymbol{\theta}' = \boldsymbol{\theta} + \delta\boldsymbol{\theta}$ are proposed, which are used to create a new circuit and thus a new trial state, and so on.

The first thing to consider is that the cost function surface can be arbitrarily rugged (this depends on the circuit) and therefore, even if the exact ground state is contained within the variational ansatz, we may not find it due to optimization being trapped in local minima.

The simplest thing to use as optimizer are energy-based methods. For instance, those routinely available in SciPy like COBYLA or BFGS. These mimimizers needs to perform repeated calls to the function $C_{\boldsymbol{\theta}}$ and compute one or a set of energy differences

$$\Delta = C_{\boldsymbol{\theta}} - C_{\boldsymbol{\theta}'}, \tag{6.12}$$

where $\boldsymbol{\theta}'$ is a proposed change of the variational parameters' array.

Another possibility, which should be preferred is the usage of gradients, in this case the parameters' *psudo-dynamics* reads

$$\boldsymbol{\theta}' = \boldsymbol{\theta}' + \eta \boldsymbol{f}_{\boldsymbol{\theta}}, \tag{6.13}$$

where the generalized force is $f_{\theta_i} = -\frac{\partial C_{\boldsymbol{\theta}}}{\partial \theta_i}$, for each θ_i component $i = 1, \cdots, N_p$, and η is a tunable time-step or, adopting fancier ML jargon, the learning rate. In many numerical methods, the gradient can be calculated using finite differences, i.e. from the energy difference calculated at two short distance points

$$f_{\theta_i}^{FD} = -\frac{C(\theta_i + \delta) - C(\theta_i)}{\delta}, \tag{6.14}$$

where $\delta \to 0$. This is not the case for quantum computing, as each energy evaluation comes with a measurement noise (see Sect. 2.4).

For the rest of the Chapter, we assume that the cost function is actually the *energy* of the system

$$E_\theta = \langle \psi_\theta | H | \psi_\theta \rangle \tag{6.15}$$

6.1.3 Parameter shift rule

In some cases, the finite difference formula becomes exact, for specific and large values of δ. In this notes, we will derive the formula for the simplest case in which the variational parameters are appearing in a single qubit rotation gate.

For simplicity, let us consider that we have only one parametrized gate,

$$U(\theta) = e^{-i\theta P}, \tag{6.16}$$

where P is a Pauli matrix. Here we omit the factor 1/2 in the definition of the angle, as to match the standard rotation gate definition. In this case we have that $\partial_\theta U(\theta) = -iPe^{-i\theta P} = -iPU(\theta)$. Suppose that the parametrized states is $|\psi_\theta\rangle = U(\theta)|\psi\rangle$, namely, we want to differentiate with respect to the variational parameter appearing in the *last* unitary operator U, of a possibly long chain, i.e $|\psi\rangle = U'(\theta')U''(\theta'')\cdots|0\rangle$. This is assumed here to simplify the notation, but it is not a limiting factor.

Then, the derivative of the energy with respect to θ is

$$\partial_\theta E = \partial_\theta \langle\psi|U^\dagger(\theta)|\, H\, |U|\psi\rangle = \langle\psi|\, \partial_\theta U^\dagger H U\, |\psi\rangle + \langle\psi|\, U^\dagger H \partial_\theta U\, |\psi\rangle\,, \tag{6.17}$$

where we drop the dependency on θ of U. Using Eq. 6.16, we have that

$$\partial_\theta E = i\,\langle\psi|\, U^\dagger(PH - HP)U\, |\psi\rangle\,, \tag{6.18}$$

where we used the fact that P commutes with U, and the minus plus sign comes from the derivative of $U^\dagger$.

Now comes the first trick. Eq. 6.18 can be rewritten as

$$\partial_\theta E = \frac{1}{2}\big(\,\langle\psi|\, U^\dagger(I - iP)^\dagger H(I - iP)U\,|\psi\rangle - \langle\psi|\, U^\dagger(I + iP)^\dagger H(I + iP)U\,|\psi\rangle\,\big). \tag{6.19}$$

This form is useful because this other equality holds

$$U\left(\frac{\theta}{4}\right) = \frac{1}{\sqrt{2}}(I - iP), \tag{6.20}$$

because $P^2 = I$, i.e. P has two eigenvalues ± 1. [3]. This also means that $\frac{1}{\sqrt{2}}(I - iP)^\dagger = \frac{1}{\sqrt{2}}(I + iP^\dagger) = \frac{1}{\sqrt{2}}(1 + iP) = U^\dagger(\pi/4) = U(-\pi/4)$.

Eq. 6.19 then can be written as

$$\partial_\theta E = \langle\psi|\, U^\dagger U^\dagger(\pi/4) H U(\pi/4) U\,|\psi\rangle - \langle\psi|\, U^\dagger U^\dagger(-\pi/4) H U(-\pi/4) U\,|\psi\rangle\,. \tag{6.21}$$

We basically applied an additional rotation of $\pm\pi/4$ to the parametrized operator, since $U(\pi/4)U(\theta) = U(\theta + \pi/4)$.

Therefore

$$\begin{aligned} \partial_\theta E &= \langle\psi|\, U^\dagger(\theta + \pi/4) H U(\theta + \pi/4)\,|\psi\rangle - \langle\psi|\, U^\dagger(\theta - \pi/4) H U(\theta - \pi/4)\,|\psi\rangle && (6.22)\\ &= E(\theta + \pi/4) - E(\theta - \pi/4). && (6.23) \end{aligned}$$

One can re-do the calculation for the case in which the parametrized gate is a properly defined rotation gate, i.e. with the factor $\theta/2$. In this case, the parameter shift rule formula is:

$$\partial_\theta E = \frac{1}{2}\left(E(\theta + \pi/2) - E(\theta - \pi/2)\right). \tag{6.24}$$

So, one needs to compute the energy at two angle values, shifted by $\pm\pi/2$ to obtain the exact derivative at point θ.

[3] The general case where the eigenvalues of P are $\pm\lambda$ is $U(\frac{\theta}{4\lambda}) = \frac{1}{\sqrt{2}}(I - i/\lambda P)$

6.1.4 The measurements problem

Up to this point, everything seems simple and straightforward. Unfortunately, a crucial point we haven't touched upon is how to measure the energy itself. This point allows us to close the loop on all the information provided in previous chapters 2 and 5. The first step is to express H as a sum of Pauli terms, and the second step is to measure each term individually. Finally, we understand again the usefulness of Sect. 2.4.2.

Let's consider a fermionic many-body Hamiltonian (but it could also be a spin or bosonic), common in quantum chemistry. As we saw in Ch. 5 the hamiltonian takes the general form of linear combination of products of single-qubit Pauli operators

$$H = \sum_{j=1}^{N_P} h_j P_j \tag{6.25}$$

where h_j is a real scalar coefficient, and N_P is of order $\mathcal{O}(n^4)$ since there are n^4 terms to transform in Eq. 5.1. Each term P_j in the Hamiltonian is typically referred to as a *Pauli string*, and is a tensor product $P_j = \bigotimes_{i=1}^{n} \sigma_i^\alpha$ of n Pauli matrices $\sigma^\alpha \in \{I, Z, X, Y\}$. What is important for our discussion is that the coefficient h_p can take very large values (in modulus). They are determined by the Jordan-Wigner (or similar) transformation, starting from the "physical" parameters h_{pk}, h_{pqrs} in Eq. 5.1. Just to give an idea, $\sum_j |h_j|$ is of the order of 40 Ha, already for a moderate example of a H_2O molecule described in STO-6G basis[3].

Let us suppose to have prepared a variational quantum state $|\psi\rangle$: following Eq. 5.13, we see that the mean value of the Hamiltonian is the linear combination of the expectation values of the N_p Pauli strings,

$$\langle\psi| H |\psi\rangle := \langle H \rangle = \sum_{j=1}^{N_P} h_j \langle P_j \rangle. \tag{6.26}$$

For simplicity we assume that the expectation values of the P_j's are obtained from N_P independent sets of measurements: the error on the estimate is then given by

$$\epsilon = \sqrt{\sum_j |h_j|^2 \mathrm{Var}[P_j]/M_j}, \tag{6.27}$$

where $\mathrm{Var}[P_j] = \langle P_j^2 \rangle - \langle P_j \rangle^2 \leq 1$ is evaluated using M_j repeated measurements, or *shots*. For the example of Sect. 5.1.4 this procedure entails the preparation of 12 different circuits, each of them containing the necessary postrotations to measure a given Pauli string expectation value. For instance, to measure $\langle\psi| X_3 X_2 X_1 X_0 |\psi\rangle$, we would need to append four Hadamard gates to the circuits that prepares $|\psi\rangle$. Each of these circuits needs to be repeated M_j times to gather sufficent statistics to compute the expectation value.

Since we need to evaluate each $\langle P_j \rangle$ independently, their statistical fluctuations are not correlated, so one needs to reach chemical accuracy by resolving each expectation value with very high accuracy as it gets multiplied by a possibly very large prefactor h_j. In Wecker et. al.[3], it is estimated that $M = \sum_j M_j = 10^9$ measurements would be needed to reach $\epsilon \sim 10^{-3}$ Ha, for H_2O, and up to 10^{13} for Fe_2S_2 decribed with 112 spin-orbitals and STO-3G basis.

There is the possibility that some of the Pauli strings commute with some others. These can be evaluated using the same post-rotation setting. For intance, for the Ising models, all the terms such as $XIII$, $IXII$, and so on, commute and can be measured in just one basis. However, this reduces only marginally the number of different circuits to be run. All in all, this *measurement problem* is one of the major drawback of VQE, keeping also in mind that it is not related to hardware noise, which conceptually could be eliminated instead.

Further details and comparison with how classical variational methods work can be found in Ref. (5).

6.2 QAOA

Variational algorithms also find application in cases where the cost function is not the expected value of a many-body Hamiltonian. Here, for completeness, we present a fairly well-known variational algorithm for optimizing classical cost functions.

In principle, one could use any variational circuit. However, why should there be an advantage if the solution to a classical problem is a single bit-string, and not a superposition of many (as is the case for a genuine ground-state many-body system)? Quantum computing cannot offer, unlike quantum chemistry, an advantage in storing the solution, but perhaps an advantage in finding it.

These arguments seem even more heuristic than the previous case, and indeed they are. However, there is a type of circuit that might work, based on the physical intuition behind it. This algorithm is called the Quantum Approximate Optimization Algorithm (QAOA)[4] (6)

The optimization problems we can solve are Ising models such as the ones presented in Chapter 3, defined over L variables $(\sigma_1, \ldots, \sigma_L) = \boldsymbol{\sigma}$ with $\sigma_j = \pm 1$. The energy of a spin configuration $\boldsymbol{\sigma}$ reads:

$$E(\boldsymbol{\sigma}) = -\sum_{i,j=1}^{L} J_{i,j}\sigma_i\sigma_j - \sum_{j=1}^{L} h_j\sigma_j. \tag{6.28}$$

Representing a generic spin configuration $\boldsymbol{\sigma} \in \{1, -1\}^L$ as a binary string $\boldsymbol{x} \in \{0, 1\}^L$, and writing the energy as $E(\boldsymbol{\sigma}) \to f(\boldsymbol{x})$, we write the problem Hamiltonian $\hat{H}_\text{P}$ as a diagonal operator

$$\hat{H}_\text{P} = \sum_x f(x)|x\rangle\langle x|, \tag{6.29}$$

defined by its diagonal matrix elements $f : \{0, 1\}^L \to \mathbb{R}$. The classical spin variables σ_j are promoted to single-qubit Pauli operators $\hat{\sigma}_j^z$.

QAOA can be understood as a digitized version of quantum annealing (cfn. Ch. 3) that requires variational optimization of circuit parameters. These parameters can be seen as the optimizable time steps that control the evolution of the state under the action of the *problem* and the *mixing* operators, interleaved like in the Trotter algorithm (cfn. Ch. 4). Notice that, here we don't need to realize any continuous time real-time dynamics: the variational parameters can be arbitrary, and one can also utilize shallow circuits.

[4]note that the word 'algorithm' is already contained in the acronym, so it's incorrect to write QAOA algorithm, although this might slip through in the text or in speech

To be precise, the unitary operator defining the ansatz is made of d blocks, each of them being the product of two unitary operators $\hat{U}_{\mathrm{P}} = \exp\left(i\theta^l_{\mathrm{P}}\hat{H}_{\mathrm{P}}\right)$, and $\hat{U}_{\mathrm{M}} = \exp\left(i\theta^l_{\mathrm{M}}\hat{H}_{\mathrm{M}}\right)$, with $l = 1, \dots, d$ and where $\hat{H}_{\mathrm{P}}$ is the problem Hamiltonian, and

$$\hat{H}_{\mathrm{M}} = \sum_{j=1}^{L} \hat{\sigma}^x_j \tag{6.30}$$

is the non-diagonal *mixing* operator.

The implementation of these unitary operators involves efficient single-qubit rotations along the x-axis, denoted as $R_x(\theta) = \exp\left(i\theta\hat{\sigma}^x/2\right)$, and two-qubit parametrized gates, $R_{zz}(\theta) = \exp\left(i\theta\hat{\sigma}^z \otimes \hat{\sigma}^z/2\right)$. The structure of the QAOA ansatz implies that all the local $\hat{\sigma}^z \otimes \hat{\sigma}^z$ interactions within the same block are 'evolved' with the same time step θ^l_P, while all the x-rotations within the block are parametrized by the same angle θ^l_M (see Fig. **??**). The total number of parameters is $n_{\mathrm{par}} = 2d$, i. e., is independent of the problem size L, and the full unitary operator reads

$$U_{\mathrm{QAOA}}(\boldsymbol{\theta}) = \overbrace{\hat{U}_{\mathrm{M}}(\theta^d_M)\hat{U}_{\mathrm{P}}(\theta^d_P) \dots \hat{U}_{\mathrm{M}}(\theta^1_M)\hat{U}_{\mathrm{P}}(\theta^1_P)}^{d\text{-times}}. \tag{6.31}$$

The final parametrized state is

$$|\psi(\boldsymbol{\theta})\rangle = U_{\mathrm{QAOA}}(\boldsymbol{\theta}) \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)^{\otimes L}, \tag{6.32}$$

where the initial non-entangled state can be obtained from the state $|0\rangle^{\otimes L}$ by acting with one Hadamard gate on each qubit.

The expectation value of $\hat{H}_{\mathrm{P}}$ over the prepared state is given by the sum of all spin configurations

$$\tilde{C} = \langle\psi_{\boldsymbol{\theta}}|\hat{H}_{\mathrm{P}}|\psi_{\boldsymbol{\theta}}\rangle = \sum_{x=0}^{2^L-1} |\psi_{\boldsymbol{\theta}}(x)|^2 f(x). \tag{6.33}$$

Howver, as explained above, the full sum can be approximated using a finite sample of configurations

$$\tilde{C} \approx \frac{1}{M}\sum_{i=1}^{M} f(x_i), \tag{6.34}$$

where x_i are sampled from $|\psi_{\boldsymbol{\theta}}(x)|^2$. The precision of this estimate is affected by statistical noise induced by the finite number of quantum measurements M. This part is exactly the same as the VQE case, but with the nice property that only samples in the computational basis are needed. Indeed H_P is a purely diagonal Hamiltonian.

We see that the QAOA methods borrow concept from the variuos Chapters presented so far: adiabatic optimization, real-time dynamics, and variational methods.

Bibliography

[1] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, "A variational eigenvalue solver on a photonic quantum processor," *Nature Communications*, vol. 5, Jul 2014.

[2] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, *et al.*, "Variational quantum algorithms," *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, 2021.

[3] D. Wecker, M. B. Hastings, and M. Troyer, "Progress towards practical quantum variational algorithms," *Phys. Rev. A*, vol. 92, p. 042303, Oct 2015.

[4] S. McArdle, S. Endo, A. Aspuru-Guzik, S. Benjamin, and X. Yuan, "Quantum computational chemistry," *arXiv preprint arXiv:1808.10402*, 2018.

[5] G. Mazzola, "Quantum computing for chemistry and physics applications from a monte carlo perspective," *arXiv preprint arXiv:2308.07964*, 2023.

[6] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," 2014.