Master QLMN Antoine Browaeys
Experimental quantum computing antoine.browaeys@institutoptique.fr

Lecture 1 - Introduction and overview: Qubits, quantum gates and errors

December 10th 2025

Quantum computing is the field that studies how quantum physics could help performing certain computing tasks, such as factoring, sorting, optimizing or simulating, more efficiently that can be done on any classical computer. It relies on the binary encoding of information on quantum bits which can be in two states $|0\rangle$ and $|1\rangle$, as well as in any superpositions $\alpha|0\rangle + \beta|1\rangle$. Using a stream of unitary operations acting on one or two qubits, arranged in a circuit, quantum superposition, entanglement and interferences, when properly harnessed, provide the enhanced power of a quantum computer, at least on the paper.

The reasons why quantum computing is an interesting field of research are several. First, when available, quantum computers hold the promise to solve certain tasks more efficiently than what be done on a classical computer: this is the case of factoring which finds applications in cryptography ensuring secured information transfer on the Internet for example. This is also the case of searching into large lists an item with specific properties, an example of a decision problem. This is also the case of the computation of the electronic structure of large molecules or of materials that could have impact on our societies. But those are for the moment only promises, and it remains to be seen whether one can build such a machine and devise algorithms that make use of them. From the point of view of physicists and engineers, building a quantum computer and thinking of how it can be used present interesting challenges. Physicists have to come up with new ways of controlling quantum systems, eventually by quantum feedback, understand how they can be isolated from the environment and why using quantum physics provides enhanced performances. In a nutshell: they need to learn how to control, stabilize and characterize a quantum system consisting of thousands if not millions of qubits, which amounts to testing and pushing quantum physics into uncharted territories. Thus, even if we never have a large scale quantum computer, we would not have wasted our time as we would have tested quantum physics extremely precisely. For the engineers, it means developing new, better lasers, microwave sources, photon detectors, control softwares and so on.

Although today elementary quantum computers already exist, handling about 100 qubits and running a few hundred gates, they are far from being able to perform "useful" tasks. The main reason why scaling up is hard roots in the coupling of this large quantum system to the environment which results into a collapse of the qubits from a quantum superposition to a diagonal density matrix:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \rightarrow \hat{\rho} = |\alpha|^2 |0\rangle\langle 0| + |\beta|^2 |1\rangle\langle 1| \ . \tag{1}$$

The good news however is that there exists strategies to fight decoherence, which are called Quantum Error Correction and Fault-Tolerant architecture. As we will see, they

are absolutely necessary if one wants to build a functioning quantum computer, and we will devote three lectures in this course to understand how this works.

1 Brief history and motivations

See slides.

2 Quantum bits and quantum gates

The classical information manipulated by any digital device is encoded in binary using bits that can take the value 0 or 1. Any integer number x has a binary decomposition of the form:

$$x = x_{n-1} \times 2^{n-1} + x_{n-2} \times 2^{n-2} + ... + x_1 \times 2 + x_0 \ , \tag{2}$$

where the $x_n \in \{0, 1\}$. For example: $(011) = 3, (111) = 7$, and so on.

Quantum mechanically, the qubit has also two states $|0\rangle$ and $|1\rangle$ that form the basis of a Hilbert space of dimension 2. It can be prepared in any superposition $\alpha\,|0\rangle + \beta\,|1\rangle$. Owing to the fact that the overall phase of the state is irrelevant and that the state is normalized, two numbers are enough to define the general state of a qubit, which is then parametrized as:

$$|\psi\rangle = \cos\frac{\theta}{2}\,|0\rangle + \sin\frac{\theta}{2}\,e^{i\varphi}\,|1\rangle \ . \tag{3}$$

The state is fully determined by two angles θ and φ, in the same way any unit vector $\mathbf{u}$ in $\mathbb{R}^3$ is defined by its spherical coordinates. This mapping between the qubit state and a vector on a unit sphere allows one to represent a superposition state, as shown in Fig. 1: the vector $\mathbf{u}(\theta, \varphi)$ is called the "Bloch vector" and the sphere is named the "Bloch sphere". For example, the states $|0\rangle$ and $|1\rangle$ are represented by the two poles, while the state $(|0\rangle + |1\rangle)/\sqrt{2}$ points along x and $(|0\rangle + i\,|1\rangle)/\sqrt{2}$ along y. As a reminder, be careful that the states $|0\rangle$ and $|1\rangle$ are orthogonal in the Hilbert space (i.e. $\langle 0|1\rangle = 0$), but their associated Bloch vectors are anti-parallel.

We can then define the state $|x\rangle = |x_{n-1}, x_{n-2}, ..., x_1, x_0\rangle$ with $x_n \in \{0, 1\}$ as the quantum encoding of the number x. It requires n qubits. What we have gained in doing so is the ability to encode a *superposition* of two of more numbers, for example on a three-qubit state:

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |1\rangle \otimes |1\rangle = \frac{1}{\sqrt{2}}(|3\rangle + |7\rangle). \tag{4}$$

Classically this is impossible and we would need two sets of three bits to encode the two numbers.

To manipulate classical bits, one relies on gates. For example the NOT gate applies the operation $x \to \bar{x}$, i.e. $0 \to 1$ and $1 \to 0$. It is an example of a single bit operation. More interesting are two-bit gates such as the XOR gate that takes two binary inputs x and y (see Fig. 2a) and outputs a *single* bit $x \oplus y$, where $\oplus$ is the addition modulo 2.

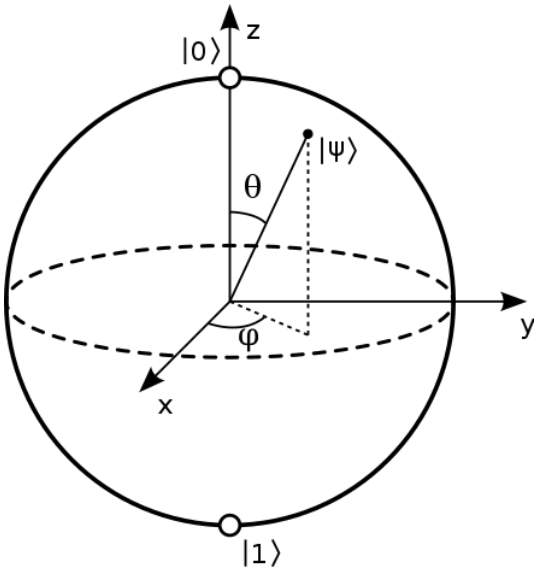

Figure 1: *Bloch vector pointing on the Bloch sphere (from Wikipedia).*

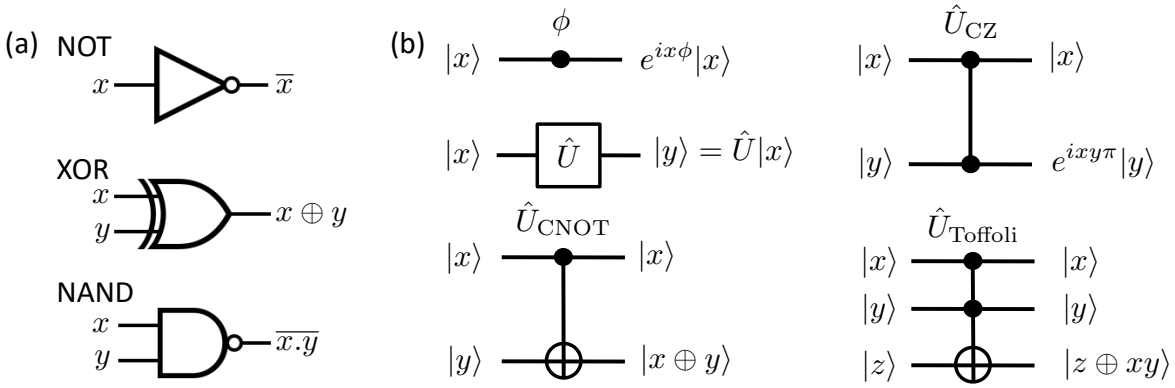

Figure 2: *Examples of classical (a) and quantum (b) gates acting on one or two (qu)bits.*

Exercise 1. Write the table of truth of the XOR and NAND gates. The NAND gate is defined by $(x, y) \to \overline{x.y}$.

The important property of the classical gates as used in our computers is their irreversibility. They erase one bit of information, and one can show that this corresponds to a fundamental energy cost $k_{\rm B}T \ln 2$ with T the temperature of the environment that the computer is placed in (Landauer's principle). One can theoretically construct *reversible* classical gates such as the CNOT or Toffoli (see below), and although not used in practical algorithms, this construction is a useful guide to develop quantum gates.

Quantum mechanically, acting on quantum bits means performing unitary operations on them. Thus, for any state $|\psi_N\rangle$ of an ensemble of N qubits, a gate performs the unitary transformation $\hat{U}$

$$|\psi_N^{\rm f}\rangle = \hat{U}\,|\psi_N^{\rm i}\rangle \ , \tag{5}$$

such that $\hat{U}\hat{U}^\dagger = \hat{\mathbb{1}}$, i.e. $\hat{U}^\dagger = \hat{U}^{-1}$. In analogy with classical computing, single and two-qubit gates are important.

Single-qubit gates. Among the useful ones, one finds:

- the X, Y and Z-gates which are represented by the Pauli matrices in the $\{|0\rangle, |1\rangle\}$ basis:

$$\hat{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \hat{Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad \hat{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} . \tag{6}$$

  Note that in quantum computing, the convention is to use $\hat{X}, \hat{Y}, \hat{Z}$ rather than $\hat{\sigma}_x, \hat{\sigma}_y, \hat{\sigma}_z$.

- the Hadamard gate represented by the matrix in the qubit basis:

$$\hat{H} = \frac{\hat{X} + \hat{Z}}{\sqrt{2}} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} . \tag{7}$$

- the phase gate $\hat{U}_\phi$ and the special cases $\hat{S} = \sqrt{\hat{Z}}$ and $\hat{T}$ gates:

$$\hat{U}_\phi = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix} \qquad \hat{S} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \qquad \hat{T} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix} . \tag{8}$$

All these gates can be represented as a circuit, as represented in Fig. 2(b).

Exercise 2. Remember that a rotation by an angle α around an axis $\mathbf{n} = (n_x, n_y, n_z)$ corresponds to the unitary transformation:

$$\mathcal{R}_{\mathbf{n}}(\alpha) = \exp\left[-i\frac{\alpha}{2}\,\hat{\boldsymbol{\sigma}} \cdot \mathbf{n}\right] = \cos\left(\frac{\alpha}{2}\right)\,\hat{\mathbb{1}} - i\sin\left(\frac{\alpha}{2}\right)\,\hat{\boldsymbol{\sigma}} \cdot \mathbf{n} \ . \tag{9}$$

with $\hat{\boldsymbol{\sigma}} = (\hat{X}, \hat{Y}, \hat{Z})$. Show that $\hat{H} = i\mathcal{R}_{\frac{\hat{\mathbf{x}}+\hat{\mathbf{z}}}{\sqrt{2}}}(\pi)$ and $\hat{H} = i\mathcal{R}_y(\pi/2)\mathcal{R}_z(\pi)$. Represent the action of the Hadamard gate on the Bloch sphere.

Exercise 3. Show that any unitary transformation $\hat{U}$ on a single qubit can be decomposed as

$$\hat{U} = e^{i\alpha}\, \mathcal{R}_z(\beta).\mathcal{R}_y(\gamma).\mathcal{R}_x(\delta)\ . \tag{10}$$

Two-qubit gates and controlled operations. They act on two input qubits and have two outputs. Often one of the input qubit is called the *control* qubit the other one the *target*. These gates are *reversible*, as they correspond to unitary transformations. Among the important ones:

- the CNOT (Controlled-NOT) gate whose matrix representation in the two-qubit basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ is:

$$\hat{U}_{\text{CNOT}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{11}$$

- the Controlled-Phase gate and the related Controlled-Z gate:

$$\hat{U}_{\text{C}\phi} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\phi} \end{pmatrix} \qquad \hat{U}_{\text{CZ}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \tag{12}$$

Their circuit representations are shown in Fig. 2(b). The essential property of these two-qubit gates is that they generate entanglement between the two qubits.

Exercise 4. Show on examples that this is the case.

Multi-qubit gates are also useful, such as the Toffoli gates which involves three qubits. Its circuit representation and working principle is shown in Fig. 2(b).

Finally, the last important manipulation on qubits is the projective measurement. Conventionally, it is performed in the qubit basis (often called computational basis). This operation takes any input qubit state and outputs $|0\rangle$ or $|1\rangle$. It is irreversible. Its circuit representation is a square box with an arrow in it.

Exercise 5. Write a small circuit that allows you to measure in the basis $|0\rangle_x = (|0\rangle + |1\rangle)/\sqrt{2}$ and $|1\rangle_x = (|0\rangle - |1\rangle)/\sqrt{2}$.

3 Quantum circuits and algorithms

Any classical algorithm is realized by a circuit consisting of a sequence of gates. It turns out that if you have only NAND and XOR gate you can run any algorithm. Those two gates form a *universal* set of gates.

The same idea can be applied to the quantum case, although as usual with some subtleties. At a high-level, an algorithm is a unitary evolution under the operator $\hat{U}$ of

the quantum state $|\psi_{\rm i}\rangle$ of a set of qubits forming a quantum register into a final state: $|\psi_{\rm f}\rangle = \hat{U}\,|\psi_{\rm f}\rangle$. This unitary evolution ends by a measurement of the state of the register, which is supposed to yield the answer of the calculation. We will see in the Lecture 5 and 6 how this works in practice.

The question is now how we can construct the evolution operator $\hat{U}$ from the quantum gates we have discussed in the last Section. The answer implies three steps, for which we will only state the results. Their demonstrations are technical. If you are interested, you can look in [1], Chapter 4, Sec. 4.5.

1. Any unitary operator acting on N qubits can be constructed from at most $d(d-1)/2$ ($d = 2^N$) two-level unitary gates. A two-level unitary gate is described by a matrix which acts non-trivially only on two of less vector components. Examples of such matrices for the 3×3 case are:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & a & b \\ 0 & c & d \end{pmatrix} \qquad \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} a & 0 & b \\ 0 & 1 & 0 \\ c & 0 & d \end{pmatrix} \ . \tag{13}$$

   Two-level unitary gates are thus universal. But they may be hard to calculate.

2. Any two-level unitary gate can be decomposed *exactly* into single-qubit gates and CNOT gates which are therefore universal. You need $\mathcal{O}(N^2 4^N)$ of such gates.

3. So far the single-qubit gates can be any, with the general form given by (10). For practical (and theoretical...) purposes, we would like to use only a discrete set of single-qubit gates. But then it becomes impossible to generate any arbitrary unitary operator acting on N qubits. Fortunately, we can *approximate* any arbitrary operator $\hat{U}$ to an arbitrary precision ϵ using a discrete set of universal gates. The standard choice of set consists of Hadamard $\hat{H}$, phase gate $\hat{S}$, T-gate $\hat{T}$ and the CNOT gate. Another, less standard, set is $\hat{H}$, $\hat{S}$, Toffoli and CNOT.

   Approximating a unitary operator $\hat{U}$ by another one $\hat{V} = \Pi_k \hat{U}^{(k)}$ which is the product of one and two-qubit gates means minimizing and error function $\epsilon(V)$ defined by

$$\epsilon(V) = ||(\hat{U} - \hat{V})\,|\psi\rangle\,|| \ , \ \forall\,|\psi\rangle \ . \tag{14}$$

   Reaching a precision ϵ means finding the decomposition $\hat{V}$ such that $\epsilon(V) < \epsilon$. It turns out that in order to approximate a general unitary operator acting on N qubits with a precision ϵ you need $\mathcal{O}(2^N \ln(1/\epsilon)/\ln N)$. In general this is then hard... This is also why it is difficult to find algorithms that present an exponential speedup with respect to the classical ones.

The choice of the universal gates set calls for an important remark. As you can check, $\hat{S} = \hat{T}^2$, so we may naively think that choosing $\hat{S}$ and $\hat{T}$ is redundant. However, the Gottesman-Knill theorem shows that you can efficiently simulate on a classical computer (i.e. in polynomial time) any circuit that is built solely using $\hat{S}, \hat{H}$ and CNOT. Such circuit does generate entanglement but not the kind of entanglement you need to speedup a calculation. One says that $\{\hat{H}, \hat{S}, \text{CNOT}\}$ form the Clifford group: It can generate quantum states with up to N^2 components for N qubits, instead of the full 2^N

that quantum physics allows. Adding the T-gate $\hat{T}$ takes you out of the Clifford group (one calls it a non-Clifford operation), and now allows you to generate any amount of entanglement.

Exercise 6. Try to understand why a circuit that uses only the operators from Clifford group, can generate restricted states. Explain then why the T-gate removes this limitation (discussing with an AI maybe useful to develop an intuition).

In this Lecture we will not develop more the algorithms that one can implement using the universal gate set introduced above. We will study in details in Lecture 5 and 6 the main quantum algorithms: search (Grover), phase estimation, quantum Fourier transform and their application to factoring (Shor). It is enough here to state that many online emulators exist that you can play with to create your own circuits: Qiskit, Quirk...(look on the Internet!). We will use them in the coming Lectures.

As a final application of the quantum circuit model, let us come back to the idea introduced by Richard Feynman in 1982 to use a quantum machine to simulate quantum physical systems such as interacting electrons in a crystal, interacting spin systems, or any many-body Hamiltonian that is considered a good description of a physical system. Finding, for example, the time-evolution of these many-body systems amount to calculating the evolution operator $\hat{U}(t) = \exp(-i\hat{H}t/\hbar)$, with $\hat{H}$ the Hamiltonian of the system (not the Hadamard gate... notations are unfortunate). In general this is a hard problem for many Hamiltonians, for example the Fermi-Hubbard Hamiltonian for interacting fermions

$$\hat{H}_{\rm FH} = J \sum_{\langle i,j \rangle} \hat{c}_i^\dagger \hat{c}_j + U \sum_i \hat{n}_{i\uparrow} \hat{n}_{j\downarrow} \ , \tag{15}$$

or the Transverse field Ising Hamiltonian for spin 1/2:

$$\hat{H}_{\rm Ising} = J \sum_{\langle i,j \rangle} \hat{\sigma}_i^z \hat{\sigma}_j^z + B \sum_i \sigma_i^x \ . \tag{16}$$

However, most of these many-body Hamiltonians are of the form $\hat{H} = \sum_k \hat{H}_k$, with $\hat{H}_k$ involving single or two bodies, for which calculating $\exp(-i\hat{H}_k t/\hbar)$ is easier. Fortunately, the Suzuki-Trotter decomposition helps:

$$\exp[-i(\hat{A} + \hat{B})] = \lim_{n \to \infty} \left(\exp[iAt/n] \exp[iBt/n]\right)^n \ . \tag{17}$$

Thus, the evolution operator $\hat{U}$ can be decomposed into a series a of "easy" to calculate terms when you split the time t into interval of duration t/n (one calls this procedure a "trotterization"):

$$U(t) = \left[e^{-i\frac{\hat{H}}{\hbar}\frac{t}{n}}\right]^n \approx \left[\prod_k e^{-i\frac{\hat{H}_k}{\hbar}\frac{t}{n}}\right]^n \ . \tag{18}$$

Exercise 7. Demonstrate the Suzuki-Trotter formula (17).

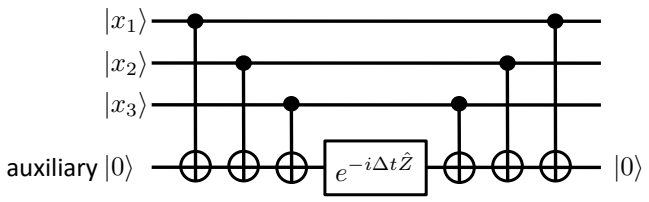

Figure 3: Quantum circuit to synthetize the 3-body spin interaction $\hat{H}_{3\text{body}}$ described by Eq. (19). The circuit calculates $\exp[-i\hat{H}_{3\text{body}}\Delta t]$.

It is therefore possible to synthetise any Hamiltonian with this digital approach, including the ones that do not involve interactions between particles allowed by Nature. As an example, consider the Hamiltonian describing the interaction between three spins

$$\hat{H}_{3\text{body}} = \sigma_1^z \otimes \sigma_2^z \otimes \sigma_3^z \ . \tag{19}$$

This 3-body Hamiltonian does not correspond to any physical interaction, as only two-body interactions are found in Nature. However, it can be digitally synthetized using sets of CNOT gates and the unitary evolution of a fourth auxiliary qubit, as shown in Fig. 3.

Exercise 8. Show that the circuit in Fig. 3 calculates the evolution of $\exp[-i\hat{H}_{3\text{body}}\Delta t]$.

As can be seen on this example, the implementation of an arbitrary Hamiltonian may require auxiliary qubits and therefore may not necessarily scale favorably with the number of particles to consider. Using the digital approach, one can perform a *universal* quantum simulation: the quantum simulator does not need to be rebuilt for each Hamiltonian $\hat{H}$ to be studied, but just reprogrammed for a specific problem.

4 Parallelism and interferences

We now ask the question of why using quantum physics allows performing certain tasks more efficiently than on a classical computer. One reads almost everywhere that this comes from the parallelism: as the calculation can be done in parallel on all the qubits at once, the calculation is more efficient. Even if this statement is not wrong it hides many subtleties. Let us formalize it. Consider a quantum register that is prepared in a superposition of several classical number, for example the three qubit state:

$$|x\rangle \propto (|0\rangle + |1\rangle)^{\otimes n} = |0\rangle + |1\rangle + |2\rangle + ... \, |6\rangle + |7\rangle \ . \tag{20}$$

Then the operation $\hat{U}_f$ acting on $|x\rangle$ should lead to:

$$|f(0)\rangle + |f(1)\rangle + |f(2)\rangle + ... \, |f(6)\rangle + |f(7)\rangle \ , \tag{21}$$

i.e. the superposition of all the results. Assuming that this is correct (and it is not, see below...) we face a problem as the measurement yields one of the results randomly. But even worse, such operation is not unitary: you can not perform the operation

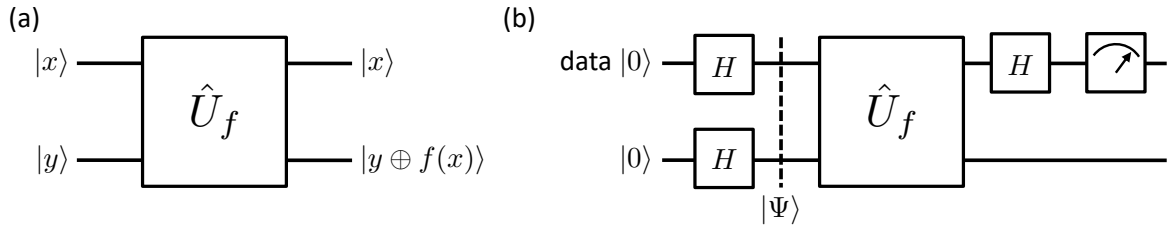

Figure 4: *(a) Operation of a function f. (b) Circuit representation of the Deutsch algorithm.*

$|x\rangle \to |f(x)\rangle$ directly, as it does not correspond to a unitary transform except in the very special case of a bijective function f. Rather, one has to consider the two-qubit transformation, represented in Fig. 4(a):

$$\hat{U}_f : (x, y) \to (x, y \oplus f(x)) \ . \tag{22}$$

Exercise 9. Show that this transformation is unitary with $\hat{U}_f^2 = \hat{\mathbb{1}}$.

Now the correct way to understand the parallelism is the following one, illustrated here on the case of two qubits. Initializing the control qubit in $(|0\rangle + |1\rangle)/\sqrt{2}$ and the target qubit in $|0\rangle$, the output is

$$\frac{1}{\sqrt{2}}(|0, f(0)\rangle + |1, f(1)\rangle) \ . \tag{23}$$

Indeed the operation took place on the two qubits at the same time, but now control and target qubits are entangled!

To understand how one can find the result of an algorithm running on the quantum computer, we consider now the very first one, introduced by David Deutsch, which shows that interferences are key resources. This "Deutsch algorithm" is academic but it does illustrate how quantum can be more powerful than classical. Consider a boolean function $f : \{0, 1\} \to \{0, 1\}$. There can be four possibilities for f: either $f(0) = f(1) = 0, 1$ (constant) or $f(0) \neq f(1)$ (non-constant). Can we decide with a minimal number of queries whether the function is constant or not? Classically, you need two queries to calculate $f(0)$ and $f(1)$. Let us now run the quantum circuit represented in Fig. 4(b). With the notation of the figure:

$$|\Psi\rangle = \frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle - |1\rangle) = \frac{1}{2}\left(\sum_{x=0}^{1} |x\rangle\right)(|0\rangle - |1\rangle) \ . \tag{24}$$

To calculate the action of $\hat{U}_f$, we first look at the two cases

$$f(x) = 0 : |x\rangle\,(|0\rangle - |1\rangle) \to |x\rangle\,(|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle = |x\rangle\,(|0\rangle - |1\rangle \tag{25}$$

$$f(x) = 1 : |x\rangle\,(|0\rangle - |1\rangle) \to |x\rangle\,(|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle = |x\rangle\,(-|0\rangle + |1\rangle \ . \tag{26}$$

Thus

$$\hat{U}_f\,|\Psi\rangle = \frac{1}{2}\left(\sum_{x=0}^{1}(-1)^{f(x)}\,|x\rangle\right)(|0\rangle - |1\rangle) \tag{27}$$

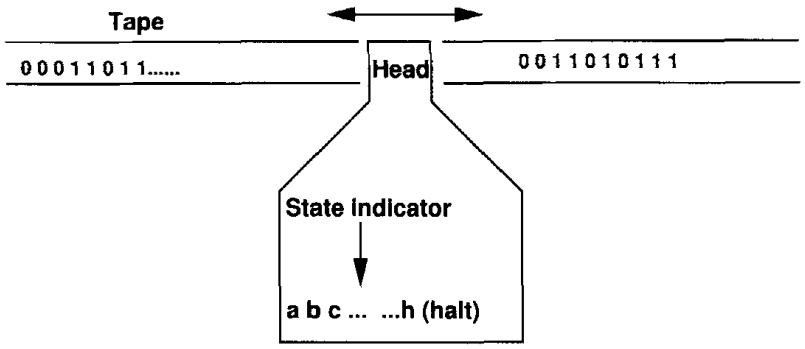

Figure 5: *Principle of a Turing machine (From Ref. [2]).*

After the application of the last Hadamard gate on the data qubit, we find it in the state

$$\frac{1}{\sqrt{2}}\left[(-1)^{f(0)}+(-1)^{f(1)}\right]|0\rangle+\frac{1}{\sqrt{2}}\left[(-1)^{f(0)}-(-1)^{f(1)}\right]|1\rangle \tag{28}$$

What we have done with the last Hadamard gate is interfering the amplitudes $(-1)^{f(x)}$. And now, if we measure on the data qubit $|0\rangle$, we know that the function is constant, otherwise we should get $|1\rangle$. We have reached our goal using only one query, thanks to the interference of amplitudes!

Exercise 10. Find an interferometric version of the algorithm.

When we describe the quantum algorithms in Lecture 5 and 6, we will look at where the interferences and the entanglement play a role.

5 Algorithmic complexity

We want to understand a bit better in which sense a quantum computer can be more efficient than a classical machine. This is the topics of algorithmic complexity, which is a whole field of research that we will only touch upon.

Let us start by what means "calculable". The british mathematician Alan Turing defined in 1936 the concept of calculable by inventing a machine which runs a series of instructions to perform the calculation. This Turing machine is an abstract concept, whose principle is shown in Fig. 5. In a nutshell the machine has two components: a ribbon with numbers or letter $\{q_i\}$ written in boxes, and a moving head carrying an internal state which can take values in a set $\{s_i\}$. When the head reads q_i in a box on the ribbon its internal states determines the updated value q_f of the ribbon, and at the same time its new internal state s_f. The internal states of the head also determines whether the head moves left or right at the next step. The sequence repeats until it stops. The Church-Turing thesis states, roughly, that a function is calculable if it can be calculated by a Turing machine. Don't believe this is obvious: there exists functions that you cannot calculate (actually even a non countable set of such...). As far as we know, it is not conceivable to devise a machine that can calculate something that a

Turing machine can not, not even a quantum computer. Hence a quantum computer will never be able to perform a task that a classical computer can not do...

The difference between classical and quantum computers is therefore a question of efficiency: how fast can you perform a calculation? To discuss the efficiency of an algorithm, let us introduce $L = \log_2 x$ the number of (binary) digits you need to code the integer x in binary, and s the number of steps you need to perform a calculation on x. The efficiency is a statement about the scaling of s with L. For example multiplying two numbers with L digits require $s \sim L^2$ steps, a polynomial number of steps. On the contrary, the best factoring of a number of L digits requires $\mathcal{O}(\exp(1.9L^{1/3}(\ln L)^{2/3}))$, i.e. a nearly exponential scaling. Problems for which s scales polynomially with L ($s \sim L^\alpha$) are considered easy, while problems scaling exponentially (or sub-exponentially as for the factoring) are considered hard.

Complexity theory states that you cannot change the complexity of a problem by changing the classical computer you use. Hence complexity becomes an intrinsic property of the algorithm, and one can define complexity classes. The class **P** contains problems that scale polynomially with L. The class **NP** contains problems for which the solution can be checked in polynomial number of steps. Factoring is such a problem: when you know the decomposition in prime factors of a given number, you can easily check that this is the solution. Intuitively $\mathbf{P} \subset \mathbf{NP}$. Now an open question in complexity theory is whether $\mathbf{P} = \mathbf{NP}$ or not. If this is true, we should be able to find a factoring algorithm that scales polynomially with L...!

Interestingly, a quantum computer can change the class of complexity of a problem, and this is what got people excited about quantum computing. For example factoring a L-digit number by the Shor's algorithm requires $\mathcal{O}(L^3)$ operations, a huge speedup with respect to the known classical algorithm (but remember: we don't know whether a polynomial algorithm exists...). The Grover search algorithm provides a quadratic speedup: searching an element in a list of L entries requires classically $\sim L/2$ queries, while the quantum version requires $\sim \sqrt{L}$. So here, if the quantum algorithm does provide a speed up, it does not change the class of complexity. The class of problems that can be solved efficiently on a quantum computer is called **BQP** (bounded quantum polynomial). The exact relation between **P**, **NP** and **BQP** is still a subject of research, and there are many other classes that have been introduced.

Exercise 11. Calculate the time needed to break RSA 2048 on a Macbook pro operating a 3 GHZ and a quantum computer with the same clock frequency.

6 Quantum errors and their correction

As we saw above, quantum computers hold the promise to provide an exponential speedup for certain tasks (arguably not many so far...). To obtain this, the machine has to run a quantum circuit consisting of many (really many...) gates. However when it comes to the practical implementation of this gates on a physical hardware, they will have errors that can be minimized but probably not to an arbitrarily low rate. Decreasing this rate is to a large extend an engineer problem.

The consequence of these errors is that after each gate, the probability of success will

be $1-\epsilon$, and assuming that errors are independent and of the same order of magnitude, the probability of success of a quantum circuit operating N gates will be $(1-\epsilon)^N \approx e^{-N\epsilon}$, an exponential suppression. Operating $N = 1000$ gates thus requires $\epsilon \ll 10^{-3}$. If you want to run Shor's algorithm on $L = 2048$, you need $\mathcal{O}(10^{10})$ gates, and therefore $\epsilon \ll 10^{-10}$. Not easy... It thus seems that the promises for exponential speedup is plagued by the exponential scaling of the influence of the errors. This problem was identified as early as 1995 by Peter Shor [3], who proposed a method to code in a more robust way, and to correct the errors. We will discuss in details this subject in Lecture 7-9. Here we only sketch the idea.

Classically, errors can only occur as bit flips: $0 \to 1$ or $0 \to 1$. Today's computers have a probability of errors per operation around 10^{-18}, quite amazing actually. Quantum mechanically errors have three forms, making them harder to cope with:

- bit flip: $|0\rangle \to |1\rangle$ or $|0\rangle \to |1\rangle$;
- phase flip: for example $|1\rangle \to -|1\rangle$. This is a problem as the qubit $|0\rangle + |1\rangle$ is changed into the orthogonal state $|0\rangle - |1\rangle$;
- small error: $|0\rangle \to |0\rangle + \epsilon\,|1\rangle$. It looks like a continuous error, but this is not: if you measure the state of the qubit you will get $|1\rangle$ with a probability ϵ^2. So thanks to the measurement process, this error is also digital (more in Lecture 7).

The strategy to fight the errors in *classical* computing relies on redundancy. To illustrate the idea, take three bits and introduce the *logical* bits $0_{\rm L} = (0,0,0)$ and $1_{\rm L} = (1,1,1)$. If a bit flip error occurs with a probability ϵ, the logical bit $0_{\rm L}$ is transformed into $(1,0,0)$, $(0,1,0)$ or $(0,0,1)$ with a probability $P_1 = 3\epsilon(1-\epsilon)^2$, into $(1,1,0)$, $(0,1,1)$ or $(1,0,1)$ with a probability $P_2 = 3\epsilon^2(1-\epsilon)$, and into $1_{\rm L} = (1,1,1)$ with a probability $P_3 = \epsilon^3$. The probability that no error occurs is $(1-\epsilon)^3$. Now if we detect a single bit flip, a majority vote argument allows us to decide that an error occurred and to correct for it. The problem occurs when at least two flip errors occur as now we can not decide whether it was a two flip error from $0_{\rm L}$ or one bit flip from $1_{\rm L}$. Thus the correction of error by flipping back the error following a majority argument works when at most one error occured. The error rate of the *logical* qubit is thus the one resulting from the impossibility to apply the majority vote argument:

$$\epsilon_{\rm L} = P_2 + P_3 = 3\epsilon^2 - 2\epsilon^3 \ . \tag{29}$$

Hence, for $\epsilon < 1/2$, the error rate of the logical bit is smaller than the one of the single bit, and by a lot: if $\epsilon = 10^{-3}$, we get $\epsilon_{\rm L} = 3 \times 10^{-6}$! However, this scheme requires an overhead in the number of bits.

Applying bluntly the redundancy strategy to the quantum case has however two problems. First we can not measure the state of a quantum bit without destroying it (which is possible classically). Second we can not clone unknown quantum states and thus the redundancy strategy could fail.

Exercise 12. Demonstrate that you cannot clone an unknown state $|\psi\rangle$ without destroying it (no-cloning theorem). Assume you could. This would mean that if you

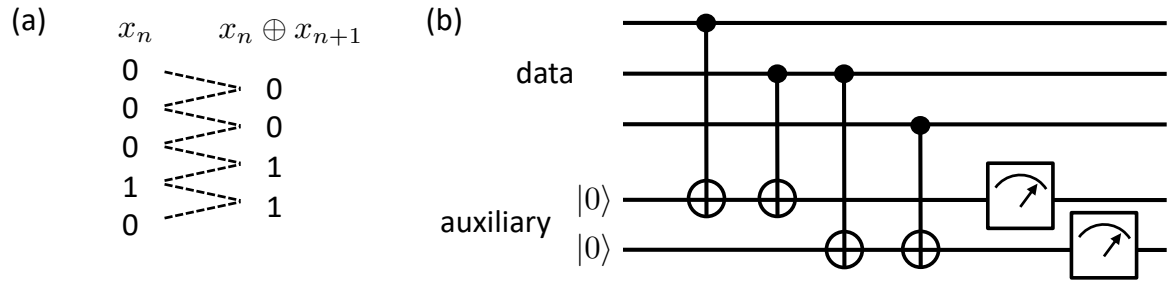

Figure 6: *(a) Detection of classical errors by parity measurement. (b) Circuit used to detect an error by measuring auxiliary qubits in the Z-basis: the circuit measures the parity between the qubits 1 and 2, and 2 and 3.*

have two qubits, one in state $|\psi\rangle$, the other in state $|0\rangle$, you could construct a unitary operator $\hat{U}$ such that $\hat{U}(|\psi\rangle\,|0\rangle) = |\psi\rangle\,|\psi\rangle$. By considering two non-orthogonal states $|\psi\rangle$ and $|\phi\rangle$ and the fact that $\hat{U}$ conserves the hermitian product, show that this is not possible (Wooters and Zurek 1985).

To extend the idea of logical bits to the quantum realm, let us define the two logical quantum bits: $|0_{\rm L}\rangle = |000\rangle$ and $|1_{\rm L}\rangle = |111\rangle$. If we prepare a superposition $|0_{\rm L}\rangle + |1_{\rm L}\rangle$ and if a bit flip occurs leading to, say, $|001\rangle + |110\rangle$, we can detect the error getting inspiration from the classical case: to detect that an error occurred on one bit and localize which one, we can measure the parity of consecutive bits, $x_n \oplus x_{n+1}$, as represented in Fig. 6(a). Quantum mechanically, this can also be done by adding two *auxiliary* qubits (often also called *ancilla* qubits) and performing the circuit shown in Fig. 6(b): it measure the parity between the qubits 1 and 2, and 2 and 3, leaving their state untouched. In this way, you can tell where the error took place and correct for it. Of course, here we measure only qubit flips, but the strategy can be extended to phase flip by measuring in a different basis. We will study all these in details in Lecture 7-9, but the key idea here is that you can detect and correct quantum errors, at the price of an overhead in the number of qubits: first to build the logical qubits and second to add auxiliary qubits to detect the errors. The good news is that if the errors on the physical qubit are low enough, you suppress the logical qubit errors exponentially with the number of physical qubits forming the logical qubits (threshold theorem).

One defines a quantum error correction code by a set of three numbers (that may not be independent): $[\![n, k, d]\!]$, with n the total number of physical qubits, k the number of logical qubits that you construct from the n qubits, and d the code distance related to the number of physical qubit errors that the code can detect (usually $d-1$) and correct $(d-1)/2$.

7 Qubits in practice

We finally get to the point where we need to worry about the implementation of all the abstract concepts discussed above. The detailed discussion of the leading physical systems onto which encoding qubits will fill the next three lectures. Here we will highlight

general properties of the qubits generic to all platforms.

Around 2000, David DiVicenzo from IBM introduced a set of five criteria that a candidate physical system must fulfill to be a viable qubit:

1. The physical system can be well isolated, and can be replicated (scalability);
2. It can be initialized in one of the two states;
3. It can be measured;
4. It can be manipulated by a universal set of gates;
5. It should have a long enough coherence time to keep the superposition state $\alpha|0\rangle + \beta|1\rangle$ coherent during the whole manipulations.

Sometimes, one also adds two criteria relevant for quantum communication:

1. The ability to interconvert stationary and flying qubits, and
2. The ability to transmit flying qubits between distant locations.

Today, the leading platforms fulfilling these criteria are: laser cooled trapped ions and atoms, superconducting quantum circuits, photons, spin of electrons trapped in quantum dots. Historically the platform based on nuclear magnetic moments (NMR) was the first to be demonstrated, but it is no longer considered promising even though it strongly influenced the development of the other platforms. We should however be careful when talking about leading platforms: history has taught us that newcomers are always possible, and sometimes promising platforms show unexpected limitations...

7.1 Coherence properties of qubits

As a reminder, a pure qubit state has the form $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. It corresponds to a perfectly isolated system, and it is thus an idealization. As no physical system is perfectly isolated, the proper description relies on the density matrix of the qubit:

$$\hat{\rho} = \begin{pmatrix} \rho_{00} & \rho_{01} \\ \rho_{10} & \rho_{11} \end{pmatrix} . \tag{30}$$

The diagonal coefficients ρ_{00} and ρ_{11} are called *populations*, and the off-diagonal ones $\rho_{01} = \rho_{10}^*$ are named *coherences*. For a pure state $\rho_{00} = |\alpha|^2$, $\rho_{11} = |\beta|^2$, and $\rho_{01} = \alpha\beta^*$. The density matrix fulfills $\mathrm{Tr}[\hat{\rho}] = \rho_{00} + \rho_{11} = 1$ and $|\rho_{01}|^2 \leq \rho_{00}\rho_{11}$, with the equality holding for a pure state. The purity $\mathrm{Tr}[\hat{\rho}^2] \leq 1$ distinguishes pure and mixed states.

The coupling of the qubit to the environment is usually accounted for phenomenologically by a damping of the populations and coherences. One introduces the lifetime T_1 of the qubit (for example due to spontaneous emission) such that $\dot{\rho}_{11} = -\rho_{11}/T_1$. In the same way, the coherences relaxes with a time scale T_2 such that $\dot{\rho}_{01} = -\rho_{01}/T_2$. As you saw in the Light-Matter Interaction course, a qubit which can decay only by spontaneous emission fulfills $T_2 = 2T_1$. In the more general case where the qubit can also dephase by other means, for example dephasing due to a fluctuating environment

(see HW6 of the Physics of Quantum Information course), one introduces a time T_2^* so that

$$\frac{1}{T_2} = \frac{1}{2T_1} + \frac{1}{T_2^*} \ . \tag{31}$$

The formal description of the effect of the environment on a qubit relies on the Lindblad formalism:

$$\dot{\hat{\rho}}_{\text{relax}} = \sum_{\alpha \neq 0} (\hat{L}_\alpha \rho_{\mathcal{S}} \hat{L}_\alpha^\dagger - \frac{1}{2} \hat{L}_\alpha^\dagger \hat{L}_\alpha \rho_{\mathcal{S}} - \frac{1}{2} \rho_{\mathcal{S}} \hat{L}_\alpha^\dagger \hat{L}_\alpha) \ . \tag{32}$$

Each jump operator $\hat{L}_\alpha$ describes a *decoherence channel*. For example a bit flip around x corresponds to an operator $\hat{L}_x = \sqrt{\gamma}\hat{X}$, and a dephasing (or depolarization) along z to $\hat{L}_z = \sqrt{\gamma}\hat{Z}$.

Importantly, a description by a Lindblad form implies an exponential decay of the coherences and populations. But this is not always the case. Take for example an ensemble of qubits with a spread of frequencies ω_i around a mean ω_0 (inhomogeneous broadening, as for the Doppler effect in a gas of atoms, or a spatially varying crystalline environment in a solid). The coherences of the qubit i will freely evolve according to $\rho_{01}^{(i)}(t) \propto e^{-i\omega_i t}$, so that the average for all the ensemble will be $\rho_{01}(t) \propto \langle e^{-i\omega_i t} \rangle$.

Exercise 13. Assume that the frequencies ω_i have a gaussian distribution. Calculate the evolution of $\rho_{01}(t)$ and show that it is not exponential.

7.2 Control of the qubits

Any gate acting on a qubit results experimentally from its coupling to a *classical* electromagnetic field, such as the one emitted by a microwave generator or the optical field of a laser. This coupling is described by the following Hamiltonian:

$$\hat{H}_{\text{coupl.}} = -\frac{\hbar\delta}{2}\hat{Z} + \frac{\hbar\Omega}{2}(e^{i\varphi}\hat{\sigma}_+ + e^{-i\varphi}\hat{\sigma}_-) \ , \tag{33}$$

with $\delta = \omega - \omega_0$ the detuning of the frequency of the field with respect to the one of the qubit's transition ω_0. This expression is valid in the Rotating Wave Approximation (RWA). Also, $\hat{\sigma}_+ = \hat{\sigma}_-^\dagger = |1\rangle\langle 0|$. When the phase of the driving field is $\varphi = 0$, the second term takes the form $(\hbar\Omega/2)\hat{X}$.

The differential equation ruling the evolution of the density matrix $\hat{\rho}$ is then

$$\dot{\hat{\rho}} = \frac{1}{i\hbar}[H_{\text{coupl.}}, \hat{\rho}] + \dot{\hat{\rho}}_{\text{relax}} \ . \tag{34}$$

It leads to the Bloch equations in the RWA:

$$\dot{\rho}_{11} = -\frac{\rho_{11}}{T_1} - i\left(\frac{\Omega}{2}\rho_{01} - \frac{\Omega^*}{2}\rho_{10}\right) \ , \tag{35}$$

$$\dot{\rho}_{01} = -\left(\frac{1}{T_2} + i\delta\right)\rho_{01} - i\frac{\Omega}{2}(\rho_{11} - \rho_{00}) \ . \tag{36}$$

The solution in the case $T_1, T_2 \to \infty$ (perfect qubit) are the Rabi oscillations.

To measure experimentally the phenomenological decay and dephasing times $T_{1,2}$, one applies the following procedure. For the decay time T_1, excite the system in the state $|1\rangle$ and measure the population remaining in this state after a time t. This is done in practice by repeating the measurement many times on the same qubit, or one time on a collection of qubits and counting the number $N_1(t)$ of qubits in state $|1\rangle$ for N measurements. We should find $P_1(t) \approx N_1(t)/N \propto e^{-t/T_1}$, from which we extract T_1.

The measurement of the dephasing time T_2 relies on Ramsey spectroscopy. Starting from a qubit initialized in $|0\rangle$, we first apply a $\pi/2$-pulse, say around the y axis, thus preparing $(|0\rangle + |1\rangle)\sqrt{2}$. We then let the qubit evolve freely at a rate δ with respect to the microwave frequency, leading to $(|0\rangle + e^{-i\delta t}|1\rangle)\sqrt{2}$. A second $\pi/2$-pulse after a time T mixes the state $|0\rangle$ and $|1\rangle$ and yields a probability $P_0(T) = \sin(\delta T/2)^2$. If the environment fluctuates during between the two pulses the probability $P_0(T) = \langle \sin(\delta T/2)^2 \rangle$ decays as a function of T, with an envelope that depends on the noise model and from which we extract T_2.

Exercise 14. Assume that δ has a gaussian distribution. Calculate the envelope of $P_0(T)$.

Exercise 15. Illustrate the Ramsey sequence on a Bloch sphere.

Exercise 16. Write a circuit implementing the Ramsey spectroscopy and propose an interferometric analogy.

A variant of the Ramsey spectroscopy consists in setting $\delta = 0$ and applying a second $\pi/2$-pulse dephased by φ with respect to the first one. One finds then $P_0(T) = \sin(\varphi/2)^2$ (show it...).

Importantly, there exists methods to fight certain types of dephasing or noise. They are called *dynamical decoupling* methods and are key ingredients in the realization of a quantum computer, as we will see later. The simplest one is the spin-echo (or Hahn) sequence. It consists in applying between the two $\pi/2$-pulses of the Ramsey sequence a π-pulse that flips the qubit states $|0\rangle$ and $|1\rangle$. In this way the evolution in the second half of the sequence, after the π-pulse, leads to a rephasing of the dynamics in the case of inhomogeneous broadening. Other techniques exist that combine various pulses and that allows to partially compensate for pulse errors, i.e. when the area of the pulse differs from the targeted one by an amount ϵ.

The next three lectures will illustrate these methods on specific implementations of qubits.

References

[1] Nielsen and Chuang, Quantum computation and, Cambridge University Press (2008).

[2] Joachim Stolze and Dieter Suter, Quantum Computing, Wiley (2008).

[3] P.W. Shor, "Scheme for reducing decoherence in quantum computer memory", Phys. Rev. A **52**, 2493 (1995).