# Quantum Algorithms

Lorenzo Laneve

January 11, 2026

# Preface

These notes are my attempt to explain various concept in quantum algorithms. This because I learn and consolidate a lot by actually writing things down.

I take inspiration from a lot of materials, but the biggest ones are the Nielsen and Chuang book [1], as well as the lecture notes by Ronald de Wolf [2] and Andrew Childs [3].

# Contents

# Part I

# Defining quantum computation

# Chapter 1

# Quantum information

In this chapter we show how to represent quantum information and the operations we can carry out on them. We will use complex numbers, so if you need a quick introduction or a refresher, you can check out Appendix **??**.

## 1.1 Qubits

When we construct computers, we need to encode bits in some physical system. As an example, imagine an electrical wire that is either at 5 $V$ or 0 $V$ from the ground. These two physical states can be used to encode one bit (and they are actually used in many architectures, like CMOS). From the physical system, we can simply abstract a simple mathematical object to describe only what is of interest to us, namely a variable $b \in \{0, 1\}$.

We now want to encode this bit in an electron, where for example spin-up represents 0 and spin-down represents 1. While these are two possible and distinct states, quantum effects come into play when it comes to particles at these scales.

Mathematically, what we do is to define two unit vectors $|0\rangle, |1\rangle$[1] that are orthogonal, which represent the two distinct states. Just to fix the ideas, we can simply take

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} .$$

These two states would be already enough to encode a bit, but an electron can also be in a state of the form

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

for $\alpha, \beta \in \mathbb{C}$, which we call *amplitudes.* We can thus define a quantum bit.

**Definition 1.1.** A *qubit* is a vector space of dimension 2.

$$\mathcal{H} = \operatorname{span}\{|0\rangle, |1\rangle\}$$

of all the possible linear combinations of $|0\rangle, |1\rangle$. The possible quantum states of a qubit are all the possible *normalized* vectors of this space, i.e., every $\alpha |0\rangle + \beta |1\rangle$, for some $\alpha, \beta \in \mathbb{C}$ satisfying $|\alpha|^2 + |\beta|^2 = 1$.

[1]These are pronounced as ‘ket 0’ and ‘ket 1’. We use this notation to represent column vectors.

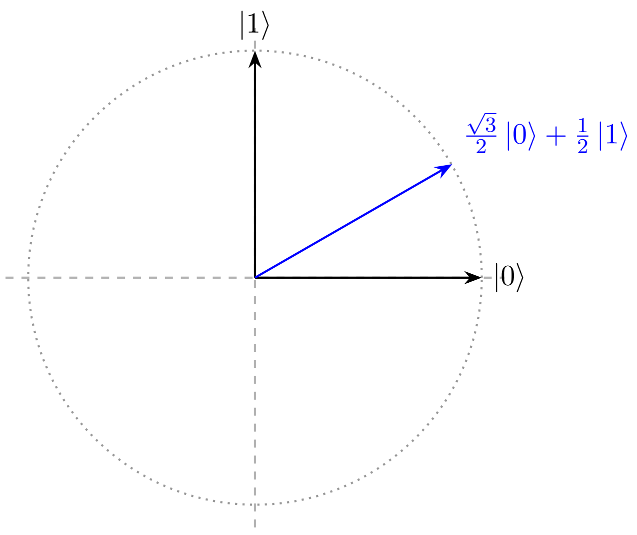


Figure 1.1: The state space of a qubit. The $|0\rangle$ and $|1\rangle$ states are orthogonal, and all the other states can be written as linear combinations of these two basis states. The coefficients must be normalized, i.e., the states must stay on the dotted circle. We remark that the picture is actually not complete, as the amplitudes are complex in general.

**Example 1.2.** The vectors

$$|1\rangle\,, \qquad \frac{1}{\sqrt{3}}|0\rangle + i\sqrt{\frac{2}{3}}|1\rangle\,, \qquad \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$

are valid normalized states, while

$$\frac{1}{3}|0\rangle + \frac{\sqrt{2}}{3}|1\rangle\,, \qquad 2|0\rangle$$

are not.

See Figure 1.1 for a geometric view of the state space of a qubit. The physical meaning of the normalization condition will be clear by the end of this chapter. In linear algebra, the two states $|0\rangle, |1\rangle$ are said to be an *orthonormal basis* for the state space of the qubit. In particular, they form the standard basis (i.e., the columns of the identity matrix) which is also refered to as the *computational basis* in quantum computation[2]. Moreover, for mathematical reasons it is common to use the term *Hilbert space* in quantum theory to refer to our vector spaces. However, since we are working with digital quantum systems, our vector spaces always have finite dimensions, and thus they are all automatically Hilbert spaces.

A first important notion in quantum information is the concept of *superposition*.

**Definition 1.3.** A state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ is said to be a *superposition* of $|0\rangle, |1\rangle$ when both $\alpha, \beta \neq 0$.

[2]Generally, when we use the term basis here, we will always implicitly refer, unless stated otherwise, to an orthonormal basis.

## 1.2 Encoding more than one bit

What we did in the previous section can actually be done with any number of possible values.

Suppose to have a variable `direction`, which can take three possible values: left ($L$), right ($R$), and stay ($S$). Such variable can be simply defined as $d \in \{L, R, S\}$, and in order to 'quantize' this variable, we can define three vectors

$$|L\rangle := \begin{bmatrix}1\\0\\0\end{bmatrix}, \quad |R\rangle := \begin{bmatrix}0\\1\\0\end{bmatrix}, \quad |S\rangle := \begin{bmatrix}0\\0\\1\end{bmatrix}.$$

Therefore, the quantum register containing such a variable would have dimension 3, and all the possible quantum states are of the form

$$\alpha\,|L\rangle + \beta\,|R\rangle + \gamma\,|S\rangle = \begin{bmatrix}\alpha\\\beta\\\gamma\end{bmatrix}$$

with amplitudes $\alpha, \beta, \gamma \in \mathbb{C}$ satisfying $|\alpha|^2 + |\beta|^2 + |\gamma|^2 = 1$. Generally, if we have a variable taking $N$ possible values, we have to define a Hilbert space of dimension $N$, where the elements of the standard basis represent these $N$ values, and their linear combinations give all the possible superpositions.

In classical computer science, however, information is defined in terms of bits because physical systems of two states are always easy to engineer and we anticipate that the same is true also for quantum information. Imagine that we want to encode the variable `direction` above in a classical computer: since the possible values are three, we allocate two bits, associate the configurations $00 = L, 01 = R, 10 = S$ and simply ignore the fourth configuration 11.

Now we see how we can combine qubits in order to mathematically express qubit registers, and we will introduce a new operation for it.

**Definition 1.4** (Tensor product)**.** Consider two vectors $|u\rangle, |v\rangle$ of dimensions $m, n$, respectively. The tensor (Kronecker) product of these two vectors is the $mn$-dimensional vector defined as

$$|u\rangle \otimes |v\rangle = \begin{bmatrix}u_1\\u_2\\\vdots\\u_m\end{bmatrix} \otimes \begin{bmatrix}v_1\\v_2\\\vdots\\v_n\end{bmatrix} = \begin{bmatrix}u_1v_1\\u_1v_2\\\vdots\\u_1v_n\\u_2v_1\\\vdots\end{bmatrix}$$

**Example 1.5.** Let us consider a numerical example:

$$\begin{bmatrix}2\\3\\5\end{bmatrix} \otimes \begin{bmatrix}1\\4\end{bmatrix} = \begin{bmatrix}2\cdot 1\\2\cdot 4\\3\cdot 1\\3\cdot 4\\5\cdot 1\\5\cdot 4\end{bmatrix} = \begin{bmatrix}2\\8\\3\\12\\5\\20\end{bmatrix}$$

The dimension of the resulting vector is $3 \cdot 2 = 6$.

**Remark 1.6.** The tensor product is *bilinear*. Bilinearity means that scalars can be moved freely:

$$(a\,|\psi\rangle) \otimes |\phi\rangle = |\psi\rangle \otimes (a\,|\phi\rangle) = a(|\psi\rangle \otimes |\phi\rangle)\ .$$

In other words, if we stretch one of the two vectors by some factor and take the tensor product, this would be equivalent to stretching the combination. While this property doesn't seem a big deal, it is actually *very important*, and many quantum algorithms exploit this property.

Moreover, bilinearity implies distributivity, a property you might be already familiar with:

$$|\psi\rangle \otimes (|\phi_1\rangle + |\phi_2\rangle) = |\psi\rangle \otimes |\phi_1\rangle + |\psi\rangle \otimes |\phi_2\rangle$$
$$(|\psi_1\rangle + |\psi_2\rangle) \otimes |\phi\rangle = |\psi_1\rangle \otimes |\phi\rangle + |\psi_2\rangle \otimes |\phi\rangle$$

We remark that the tensor product is *not* commutative, however.

The physical meaning of the tensor product is simple: suppose that we have two physical systems $A, B$. If system $A$ is in state $|\psi\rangle$ and $B$ is in state $|\phi\rangle$, then we say that the joint system can be seen as a single system, with state $|\psi\rangle_A \otimes |\phi\rangle_B$ (we will sometimes use the subscripts to remark which systems we are referring to). Sometimes we will just omit the $\otimes$ symbol and write $|\psi\rangle\,|\phi\rangle$.

Back to qubits, if we have two qubits both in state $|0\rangle$, then the two-qubit register will be in the state $|0\rangle \otimes |0\rangle$. As mentioned before, we will also use the notation $|0\rangle\,|0\rangle$, and sometimes even $|00\rangle$. By working out the tensor product, we can see that the two-qubit register will have dimension 4 and

$$|00\rangle = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \quad |01\rangle = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \quad |10\rangle = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \quad |11\rangle = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}.$$

This is in line with what we said at the beginning of this section: as two bits have four possible configurations, the Hilbert space for two qubits will have dimension four. If we want to encode the `direction` variable from above, we can take these two qubits, name $|L\rangle := |00\rangle\,, |R\rangle := |01\rangle\,, |S\rangle := |10\rangle$ and ignore the fourth dimension.

More generally, if we have $n$ qubits, the dimension of the Hilbert space will be $2^n$, and by applying the tensor product $n$ times we can deduce that the state $|x\rangle$ where the qubits make the binary string $x \in \{0,1\}^n$ is represented by the $x$-th element of the standard basis. Usually we will use the integer numbers to refer to the basis states instead of the bit strings. For example, considering a 8-qubit register, we use $|5\rangle$ as a shorthand for $|00000101\rangle$ and $|190\rangle$ instead of $|10111110\rangle$.

The general state of a $n$-qubit register is of the form

$$\sum_{x=0}^{2^n-1} \alpha_x\,|x\rangle$$

for a choice of $2^n$ amplitudes $\alpha_x \in \mathbb{C}$, again satisfying the normalization $\sum_{x=0}^{2^n-1} |\alpha_x|^2 = 1$. It is important to remark that not all of these states can be written in the form

$$|\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle \tag{1.1}$$

i.e., there are states for a $n$-qubit register that *cannot* be expressed using states of the single qubits.

**Definition 1.7.** A state for a $n$-qubit register is *separable* if it can be written in the form of (1.1). Any other state is said to be *entangled*.

**Example 1.8.** The state

$$\frac{|00\rangle + |01\rangle + |10\rangle + |11\rangle}{2} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

is separable, while the state

$$\frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

is entangled.

In other words, when the joint system is in an entangled state, we cannot describe the states of single systems. The concept of entanglement is the physical phenomenon at the heart of many quantum algorithms, as we shall see.

If we have two systems $A, B$ represented by the Hilbert spaces $\mathcal{H}_A, \mathcal{H}_B$, then the joint system is represented by the Hilbert space

$$\mathcal{H}_A \otimes \mathcal{H}_B := \text{span}\{|\phi\rangle \otimes |\psi\rangle : \phi \in \mathcal{H}_A, \psi \in \mathcal{H}_B\}$$

The joint space will thus be formed by all the possible separable states, *plus all their possible linear combinations*, representing the entangled states.

## 1.3 Measurements

We said that the state of a qubit is a generic normalized vector $\alpha|0\rangle + \beta|1\rangle$ over two dimensions, for $\alpha, \beta \in \mathbb{C}$ satisfying $|\alpha|^2 + |\beta|^2 = 1$. To introduce the concept of measurement, we need to define row vectors.

$$|\psi\rangle = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \qquad \Longrightarrow \qquad \langle\psi| = (|\psi\rangle)^\dagger = \begin{bmatrix} a_1^* & a_2^* & \cdots & a_n^* \end{bmatrix}$$

The symbol $\langle\psi|$ (pronounced 'bra $\psi$') is used to denote *row* vectors. Transforming a ket in a bra (and vice versa) means taking the *conjugate transpose*, which we will denote with the $\dagger$ symbol (pronounced 'dagger')[3]. If we have two vectors $|\psi\rangle, |\phi\rangle$ of the same dimension, you might be familiar with the scalar (inner) product.

$$|\psi\rangle = \begin{bmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_n \end{bmatrix}, \quad |\phi\rangle = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_n \end{bmatrix} \qquad \Longrightarrow \qquad \langle\psi|\phi\rangle = \sum_{k=1}^{n} \psi_k^* \phi_k$$

This is Dirac's *braket* notation: you can imagine that taking the inner product is just like snapping two puzzle pieces together, and by doing so you get back a complex number. Remember that there is a complex conjugation involved, since we use conjugate transposition, and not transposition alone.

[3]Recall that in linear algebra the transpose operation $\cdot^T$ flips the columns and rows of matrices (and vectors can be seen as $n \times 1$ or $1 \times n$ matrices). When linear algebra is defined over the complex numbers, however, we never take the transpose operation alone, but we also have to conjugate the entries. If we go back to the real numbers, complex conjugation has no effect, and thus $\cdot^\dagger$ and $\cdot^T$ are equivalent.

**Example 1.9.** Take the two vectors

$$|\psi\rangle = \begin{bmatrix} 1 \\ 2+3i \\ 4i \end{bmatrix}, \qquad |\phi\rangle = \begin{bmatrix} 5+2i \\ 7i \\ 3 \end{bmatrix}$$

We can compute the inner product

$$\langle\psi|\phi\rangle = \begin{bmatrix} 1 & 2-3i & -4i \end{bmatrix} \begin{bmatrix} 5+2i \\ 7i \\ 3 \end{bmatrix} = 1\cdot(5+2i) + (2-3i)\cdot 7i + (-4i)\cdot 3 = 26+4i$$

**Remark 1.10.** Notice that $\langle\phi|\psi\rangle = \langle\psi|\phi\rangle^*$, so the inner product is *not* commutative. Such property is called anti-symmetry, which turns into commutativity only when we work with real numbers.

**Remark 1.11.** Importantly, the inner product of some vector by itself is always real and non-negative.

$$|\psi\rangle = \begin{bmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_n \end{bmatrix} \qquad \Longrightarrow \qquad \langle\psi|\psi\rangle = \sum_{k=1}^{n} \psi_k^* \psi_k = \sum_{k=1}^{n} |\psi_k|^2 \geq 0$$

This is exactly like what you might recall from linear algebra courses. The length (2-norm) of the vector can then be computed as $\|\psi\| = \sqrt{\langle\psi|\psi\rangle}$. This is one of the reasons why we take $\cdot^\dagger$ instead of $\cdot^T$ when we deal with complex numbers, otherwise we would have vectors with negative length! The normalization condition required for a quantum state to be meaningful can then be restated as $\langle\psi|\psi\rangle = 1$.

**Remark 1.12.** Remember that two vectors $|\psi\rangle, |\phi\rangle$ are orthogonal when $\langle\psi|\phi\rangle = 0$.

We start with the simplest case: given a qubit in some state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, how can we extract information from it? From quantum mechanics we have the concept of *projective measurement*: we choose a basis $\{|\psi_1\rangle, |\psi_2\rangle\}$ for the Hilbert space. Because this is an orthonormal basis, the identity matrix can be written as

$$I = |\psi_1\rangle\langle\psi_1| + |\psi_2\rangle\langle\psi_2| \tag{1.2}$$

**Remark 1.13.** The notation $|\psi\rangle\langle\phi|$ is called *ketbra*, and will be a matrix. In particular, the matrix $|\psi_1\rangle\langle\psi_1|$ is the orthogonal projection onto the vector $|\psi_1\rangle$.

$$\begin{aligned} |\psi_1\rangle\langle\psi_1|\,|\phi\rangle &= |\psi_1\rangle\,\langle\psi_1|\phi\rangle && \text{Associativity of matrix product} \\ &= \langle\psi_1|\phi\rangle\,|\psi_1\rangle && \text{Scalars can be moved freely} \end{aligned}$$

As an example, we show (1.2) with the standard basis $\{|0\rangle, |1\rangle\}$.

$$|0\rangle\langle 0| = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$|1\rangle\langle 1| = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

So they sum up to the identity matrix. This will be true for any orthonormal basis.

Quantum mechanics tells us that if we measure a qubit in a state $|\phi\rangle$ with respect to the $\{|\psi_1\rangle, |\psi_2\rangle\}$ basis, the measurement apparatus will detect $|\psi_1\rangle$ with probability $|\langle\psi_1|\phi\rangle|^2$, while $|\psi_2\rangle$ will be detected with probability $|\langle\psi_2|\phi\rangle|^2$. Another fancy way to write down these probabilities is

$$\begin{aligned}
|\langle\psi_1|\phi\rangle|^2 &= \langle\psi_1|\phi\rangle^* \langle\psi_1|\phi\rangle && \text{since } z^*z = |z|^2 \\
&= \langle\phi|\psi_1\rangle \langle\psi_1|\phi\rangle && \text{anti-symmetry of the inner product} \\
&= \langle\phi| \, |\psi_1\rangle\langle\psi_1| \, |\phi\rangle
\end{aligned}$$

and the same for $|\phi_2\rangle$. But if these two numbers are probabilities, shouldn't they sum up to 1? This is actually always the case:

$$\begin{aligned}
|\langle\psi_1|\phi\rangle|^2 + |\langle\psi_2|\phi\rangle|^2 &= \langle\phi| \, |\psi_1\rangle\langle\psi_1| \, |\phi\rangle + \langle\phi| \, |\psi_2\rangle\langle\psi_2| \, |\phi\rangle \\
&= \langle\phi| \Big( |\psi_1\rangle\langle\psi_1| + |\psi_2\rangle\langle\psi_2| \Big) |\phi\rangle \\
&= \langle\phi|\phi\rangle && \text{by (1.2)}
\end{aligned}$$

Here is why quantum states are only the *normalized* states. The squared components of the vector are measurement probabilities, and as such they must sum up to 1 for the state to be physically meaningful. As long as $\langle\phi|\phi\rangle = 1$, linear algebra guarantees us that the sum of the squared components will be 1 regardless of the basis we choose for the measurement.

**Example 1.14.** Suppose that the qubit is in the state $|\psi\rangle = \frac{1}{\sqrt{3}}|0\rangle - \sqrt{\frac{2}{3}}|1\rangle$. If we measure with respect to the standard basis $\{|0\rangle, |1\rangle\}$, the probability to measure a 0 is

$$|\langle 0|\psi\rangle|^2 = \left|\langle 0| \left(\frac{1}{\sqrt{3}}|0\rangle - \sqrt{\frac{2}{3}}|1\rangle\right)\right|^2 = \left|\frac{1}{\sqrt{3}}\langle 0|0\rangle - \sqrt{\frac{2}{3}}\langle 0|1\rangle\right|^2 = \left|\frac{1}{\sqrt{3}}\right|^2 = \frac{1}{3}$$

Another important rule imposed by quantum mechanics is state collapse: if we measure a qubit with respect to a basis $\{|\psi_1\rangle, |\psi_2\rangle\}$ and, for example, the measurement apparatus returns $|\psi_1\rangle$, the qubit will immediately *collapse* to the state $|\psi_1\rangle$, so subsequent measurements with respect to the same basis would detect $|\psi_1\rangle$ with certainty. Thus, while the possible states of a qubit are possibly infinite, keep in mind that only one bit of information (saying whether we detect $\psi_1$ or $\psi_2$) can be extracted from one qubit.

What we said can be extended also for multiple dimensions.

**Definition 1.15.** Given a quantum system over $N$ dimensions, a *projective measurement* is a set of orthogonal projection matrices $\Pi = \{\Pi_1, \dots, \Pi_K\}$ such that the sum

$$\Pi_1 + \cdots + \Pi_K = I_N$$

The measurement generally works similarly as what we have seen with the qubit: upon measure of a $N$-dimensional state $|\psi\rangle$ with respect to $\Pi$, the measurement device will return a number in $\{1, \dots, K\}$. The probability that the device returns $j$ is $\langle\psi|\Pi_j|\psi\rangle$. In this case, the state after the measurement will collapse to

$$\frac{\Pi_j|\psi\rangle}{\|\Pi_j|\psi\rangle\|}$$

where the denominator is needed to renormalize the new state projected by $\Pi_j$. Notice that the number $K$ of projectors needs not be $N$: essentially the measurement partitions the state space into $K$ orthogonal subspaces, and the state being measured will be projected onto one of these subspaces according to the probability distribution.

**Example 1.16.** Consider a three-dimensional space $\mathcal{H} = \text{span}\{|0\rangle, |1\rangle, |2\rangle\}$, and a projective measurement:

$$\Pi_1 = |0\rangle\langle 0|$$
$$\Pi_2 = |1\rangle\langle 1| + |2\rangle\langle 2|$$

The state $|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ has a $\langle\psi|\Pi_1|\psi\rangle = \frac{1}{2}$ probability to be detected by $\Pi_1$, and $\langle\psi|\Pi_2|\psi\rangle = \frac{1}{2}$ probability to be detected by $\Pi_2$. The post-measurement states in the two cases are

$$\frac{\Pi_1|\psi\rangle}{\|\Pi_1|\psi\rangle\|} = |0\rangle, \qquad \frac{\Pi_2|\psi\rangle}{\|\Pi_2|\psi\rangle\|} = |1\rangle$$

If the register is in a state $|\phi\rangle = \frac{1}{\sqrt{2}}|1\rangle + \frac{1}{\sqrt{2}}|2\rangle$, then the probability of being detected by $\Pi_1$ is 0. The post-measurement state in this case is not definable (we would need to divide by zero), but it's a state we would never get anyway. The other case has thus probability 1, and the post-measurement state is

$$\frac{\Pi_2|\phi\rangle}{\|\Pi_2|\phi\rangle\|} = \frac{1}{\sqrt{2}}|1\rangle + \frac{1}{\sqrt{2}}|2\rangle \ .$$

The measurement $\Pi = \{\Pi_1, \Pi_2\}$ is essentially asking the question 'is the number in the register $\neq 0$?'. However, if that number turns out to be non-zero, the measurement outcome will not tell us whether it will be 1 or 2. As a consequence, any superposition between $|1\rangle$ and $|2\rangle$ will not be touched by $\Pi$.

We conclude this section with two properties of extremely importance in quantum algorithms. The first one regards what we call *global phases*: the two states $|\phi\rangle$ and $|\phi'\rangle = e^{i\theta}|\phi\rangle$ are the same physically, because for any matrix $A$ (and in particular projectors)

$$\begin{aligned}
\langle\phi'|A|\phi'\rangle &= (e^{i\theta}|\phi\rangle)^\dagger A e^{i\theta}|\phi\rangle && \text{definition of bra} \\
&= (e^{i\theta})^* e^{i\theta}\langle\phi|A|\phi\rangle && \text{since } (a|\phi\rangle)^\dagger = a^*(|\phi\rangle)^\dagger \\
&= \langle\phi|A|\phi\rangle && \text{since } (e^{i\theta})^* = e^{-i\theta}.
\end{aligned}$$

Thus the global phase $e^{i\theta}$ does not influence the measurement probabilities. In practice this means that if our objective is to create a state $|\psi\rangle$ and we find an algorithm that creates $e^{i\theta}|\psi\rangle$, we are done.

The second remark regards the *distance* between two states. Here $\|\cdot\|$ is the usual 2-norm of vectors:

$$|\phi\rangle = \begin{bmatrix}\phi_1 \\ \phi_2 \\ \vdots \\ \phi_n\end{bmatrix} \qquad \Longrightarrow \qquad \||\phi\rangle\|^2 = \sum_{k=1}^{n}|\phi_k|^2 \ .$$

**Theorem 1.17.** *If two states $|\phi\rangle, |\phi'\rangle$ are $\epsilon$-close in norm, then their measurement probabilities are $\epsilon^2$-close in total variation distance.*

We are not going into detail of the proof of this theorem, which simply involves a bunch of linear algebra. The total variation distance is a measure of how close two probability distributions are. If the total variation distance between two distributions is small, we would need many samples to tell them apart, i.e., to say from which of the two distributions we are actually sampling from. We highlight that this is true *regardless* of the basis we choose for the measurement. In practice, this means that if our goal is to construct a state $|\phi\rangle$ but we manage to only get very close to it, we are happy anyway. We shall see in Chapter 2 that this is important, as not every quantum state can be constructed exactly, but all of them can be approximated.

## 1.4 Unitary operations

While quantum states are vectors as we have seen, the operations we can carry out to manipulate these states are matrices. We will see that a quantum computer can be seen as a machine that applies matrices to its states

$$|\psi_{end}\rangle = A\,|\psi_{start}\rangle \ .$$

where $A$ is a $n \times n$ matrix if the two vectors have dimensions $n$. We cannot apply any matrix, however: remember that a vector is a physically meaningful quantum state only when it has length one. Therefore, if $|\psi_{start}\rangle$ is a quantum state, $A$ must ensure that $|\psi_{end}\rangle$ is a quantum state as well.

**Definition 1.18.** A square matrix $U$ is said to be *unitary* if it preserves the norm, i.e.,

$$\|U\,|\phi\rangle\| = \||\phi\rangle\|$$

for any vector $|\phi\rangle$.

Unitaries have different equivalent definitions, which all give us useful information about their structure that we can exploit.

**Theorem 1.19.** *$U$ is unitary if and only if $U^\dagger U = UU^\dagger = I$. In other words, $U^{-1} = U^\dagger$.*

*Proof.* For the definition to imply $(i)$ we rewrite the norm of $U\,|\phi\rangle$ as

$$\|U\,|\phi\rangle\|^2 = (U\,|\phi\rangle)^\dagger U\,|\phi\rangle = (\langle\phi|\,U^\dagger)U\,|\phi\rangle = \langle\phi|\,U^\dagger U\,|\phi\rangle \ .$$

By Definition 1.18, $\langle\phi|\,U^\dagger U\,|\phi\rangle = \langle\phi|\phi\rangle$ which is equal to one if $|\phi\rangle$ is a quantum state. This means that $U^\dagger U$ maps any unit vector to 1, a property exclusive to the identity matrix, implying $U^\dagger U = I$. Since $U$ is a square matrix and $U^\dagger U = I$, then $U^\dagger$ is also the right inverse of $U$, and thus $UU^\dagger = I$. Conversely, $U^\dagger U = I$ implies that $\langle\phi|\phi\rangle = \langle\phi|\,U^\dagger U\,|\phi\rangle$, for any vector $|\phi\rangle$, which is the initial definition we gave. □

From Theorem 1.19 we obtain a very important perspective: the operations we can carry out on quantum systems *must be invertible*! Invertible (a.k.a. *reversible*) operations are the ones that do not lose information from the input to the output, and the former can be fully reconstructed from the latter. Thermodynamics tells us that reversibility implies full preservation of energy and, roughly speaking, this is why constructing robust quantum hardware is a challenging engineering problem: losing even one tiny bit of energy in the process could be a disaster for our superpositions, and it's very easy to dissipate energy (this process of losing quantum information because of dissipation goes by the name of *decoherence*, and a unitary operation is in contrast called *coherent*). From the point of view of information, we shall see in Chapter 3 that being forced to implement only reversible operations does not prevent us from carrying out any classical computation, even when we have to compute non-invertible functions.

**Example 1.20.** Considering the matrix

$$U = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ -i & i \end{bmatrix} \qquad \Longrightarrow \qquad U^\dagger = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & i \\ 1 & -i \end{bmatrix}$$

We can check that $U^\dagger U = I$, so $U$ is unitary.

Also the tensor product can be carried out on matrices. If $A$ is $m_1 \times n_1$ and $B$ is $m_2 \times n_2$, then $A \otimes B$ will be $m_1 m_2 \times n_1 n_2$. If we have two registers, the meaning of $A \otimes B$ is 'apply $A$ to the first register, and $B$ to the second register'.

We give further properties of the tensor product that will come in handy later.

**Theorem 1.21.** *The following properties hold:*

*(i)* $(A_1 \otimes B_1)(A_2 \otimes B_2) = A_1 A_2 \otimes B_1 B_2$, *provided* $A_1, A_2$ *(resp.* $B_1, B_2$*) have compatible dimensions for matrix multiplication;*

*(ii)* $(A \otimes B)(|\phi_1\rangle \otimes |\phi_2\rangle) = A|\phi_1\rangle \otimes B|\phi_2\rangle$, *provided* $A|\phi_1\rangle$ *(resp.* $B|\phi_2\rangle$*) have compatible dimensions for matrix-vector multiplication;*

*(iii)* $(A \otimes B)^\dagger = A^\dagger \otimes B^\dagger$.

In particular, by using $(i), (iii)$ we can deduce that if $U, V$ are unitaries, then $U \otimes V$ is unitary as well. To conclude, we introduce the *operator norm*.

$$\|A\| := \max_{\|\phi\|=1} \|A|\phi\rangle\|$$

In other words, $\|A\|$ is the maximum stretch that we obtain when we apply $A$ to some vector. By definition of unitaries, $\|U\| = 1$ if $U$ is unitary. If $U, U'$ are unitaries then $\|U - U'\|$ quantifies the maximum distance between $U|\phi\rangle$ and $U'|\phi\rangle$, for any input $|\phi\rangle$. By what we have seen about the measurement probabilities of two close states, two close unitaries are essentially doing the same operation up to statistically indistinguishable differences.

## 1.5 Exercises

**Exercise 1.1.** Suppose we want to encode the variable `color` which can take values in

$$\{\texttt{red}, \texttt{green}, \texttt{blue}, \texttt{yellow}, \texttt{purple}\}\ .$$

Suggest a possible quantum encoding using the minimum possible amount of qubits.

**Exercise 1.2.** Which of the following vectors are valid quantum states for a qubit?

(i) $|\psi_1\rangle = \frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle$

(ii) $|\psi_2\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{3}}|1\rangle$

(iii) $|\psi_3\rangle = \frac{3}{5}|0\rangle + \frac{4i}{5}|1\rangle$

(iv) $|\psi_4\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|2\rangle$

**Exercise 1.3.** Are the following states separable or entangled?

(i) $|\psi_1\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$

(ii) $|\psi_2\rangle = \frac{|00\rangle + |01\rangle + |10\rangle + |11\rangle}{2}$

(iii) $|\psi_3\rangle = \frac{2|00\rangle + 3|01\rangle}{\sqrt{13}}$

**Exercise 1.4.** Let $|\psi\rangle$ be the two-qubit state

$$|\psi\rangle = \frac{3}{5}|00\rangle + \frac{4}{5}|11\rangle\ .$$

If we measure the first qubit with respect to the standard basis $\{|0\rangle, |1\rangle\}$, what is the probability of measuring a 0? What is the post-measurement state? If we instead measure with respect to the projective measurement

$$\begin{aligned}\Pi_1 &= |00\rangle\langle 00| + |11\rangle\langle 11|\\ \Pi_2 &= |01\rangle\langle 01| + |10\rangle\langle 10|\end{aligned}$$

what is the probability of obtaining outcome 1? What is the post-measurement state in this case?

**Exercise 1.5.** The Bell basis is the orthonormal basis for two qubits defined as

$$|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}, \quad |\Phi^-\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}}, \quad |\Psi^+\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}}, \quad |\Psi^-\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}$$

If we measure a two-qubit state $|\psi\rangle$ with respect to the Bell basis, what is the probability of obtaining outcome $|\Phi^+\rangle$? What is the post-measurement state?

**Exercise 1.6.** Show that unitary matrices must have orthonormal columns and rows.

**Exercise 1.7.** Which of the following matrices are unitary?

$$A = \begin{bmatrix}1 & 0\\ 0 & i\end{bmatrix}, \quad B = \frac{1}{\sqrt{2}}\begin{bmatrix}i & 1\\ 1 & i\end{bmatrix}, \quad C = \frac{1}{\sqrt{2}}\begin{bmatrix}i & 1\\ 1 & -i\end{bmatrix}$$

**Exercise 1.8.** Show that the operator norm is *unitarily invariant*, i.e., for any unitary $U$ we have

$$\begin{aligned}\|UA\| &= \|A\|\\ \|AV\| &= \|A\|\end{aligned}$$

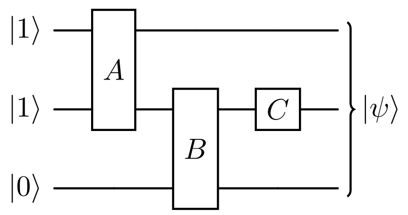
# Chapter 2

# Quantum circuits and the quantum machine

This chapter aims to construct the theoretical model of a quantum computer that we can then use to define our algorithms.

## 2.1 Circuit notation

When we specify our quantum algorithms, we have to specify which operations to apply to our qubits. For this purpose we use a model called *quantum circuits*, for which we give a first example.

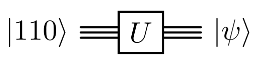


The above circuit means that we have three qubits, starting in the state $|110\rangle$ (remember: vector of dimension $2^3 = 8$). The first step is two apply the two-qubit gate $A$ to the first two qubits (it acts on two qubits, so it is a $4 \times 4$ unitary matrix). Subsequently, we apply $B$ to the second and third qubit (again, a $4 \times 4$ unitary). Finally we apply $C$, which must be a $2 \times 2$ unitary matrix, to the second qubit, and we call the resulting state $|\psi\rangle$. This can be summarized in the following operation:

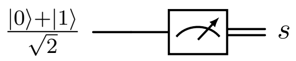
$$|\psi\rangle = (I \otimes C \otimes I)(I \otimes B)(A \otimes I)\,|110\rangle \ . \tag{2.1}$$

Notice how we introduced identity matrices here: $A \otimes I$ means 'apply $A$ to the first two qubits *and* nothing to the third one'. It is needed in the matrix formalism because dimensions must match to carry out the matrix multiplication. Also notice that circuits are read from left to right, but matrices are applied from right to left to the vector.

If we call $U$ the three-qubit unitary in (2.1), we can simply write down the circuit



In other words, we can say that $U$ is a routine calling the gates $A, B, C$. The other component we can find in a quantum circuit represents a measurement:



In this circuit we have one qubit initialized to the state $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$, which is then measured. Unless otherwise specified, measurements are carried out in the standard $\{|0\rangle, |1\rangle\}$ basis. The double line represents the classical bit $s \in \{0, 1\}$ returned by the measurement device. In the above example, $s$ will be 0 with probability $|\langle 0|+\rangle|^2 = \frac{1}{2}$.

## 2.2 Basic gates

We now introduce some basic gates that we'll use in our circuits, starting from the single-qubit gates. The first one is the *Hadamard gate*

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Notice that $H|0\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}} =: |+\rangle$ and $H|1\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}} =: |-\rangle$. The states $|+\rangle, |-\rangle$ form a basis which we call the *Hadamard basis*. Notice that also $H|+\rangle = |0\rangle, H|-\rangle = |1\rangle$ (i.e., $H^2 = I$), so the Hadamard gate essentially transforms the standard basis into the Hadamard basis *and vice versa*.

Then we introduce the three *Pauli matrices*:

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

The Pauli-$X$ gate can be seen as a NOT gate in the standard basis

$$X|0\rangle = |1\rangle, \quad X|1\rangle = |0\rangle$$

and, at the same time, the Hadamard basis $\{|+\rangle, |-\rangle\}$ is an eigenbasis:

$$X|+\rangle = |+\rangle, \quad X|-\rangle = -|-\rangle$$

The Pauli-$Z$ does exactly the same thing, but with the role of the two bases swapped. The Pauli-$Y$ matrix is the least used one, but we mention that it is also a NOT gate on the basis

$$|{+i}\rangle := \frac{|0\rangle + i|1\rangle}{\sqrt{2}}, \quad |{-i}\rangle := \frac{|0\rangle - i|1\rangle}{\sqrt{2}} .$$

It's worth mentioning that the Hadamard gate, as well as the three Pauli matrices are *Hermitian* matrices, i.e., they satisfy $A = A^\dagger$ (the complex counterpart of symmetric matrices).

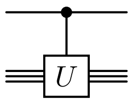
Notice that $HXH = Z$ and $HZH = X$. This follows from the fact that $H^\dagger = H$ and sandwiching with $H$ simply swaps the eigenvectors between the standard basis and the Hadamard basis.

The last single-qubit gate that we are going to introduce for now is the $T$-gate

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$$

which essentially keeps $|0\rangle$ fixed but adds a phase $e^{i\pi/4}$ to $|1\rangle$. The $T$-gate can be seen as one of the fourth roots of the $Z$ gate, since $T^4 = Z$.

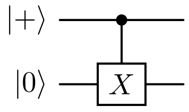
We now turn our attention to multiple-qubit gates, starting with *controlled gates*: if $U$ is some unitary operation on some number of qubits, the circuit notation



means 'if the top qubit is in the $|1\rangle$ state, apply $U$ to the bottom register, otherwise do nothing'. The matrix representation can be written as

$$C(U) = |0\rangle\langle 0| \otimes \mathbb{1} + |1\rangle\langle 1| \otimes U$$

**Example 2.1.** Let's compute the final state in the following circuit



The starting state is $|+\rangle \otimes |0\rangle = \frac{|00\rangle+|10\rangle}{\sqrt{2}}$, while the end state can be computed as

$$\Big(|0\rangle\langle 0| \otimes \mathbb{1} + |1\rangle\langle 1| \otimes X\Big)\frac{|00\rangle+|10\rangle}{\sqrt{2}} = \Big(|0\rangle\langle 0| \otimes \mathbb{1}\Big)\frac{|00\rangle+|10\rangle}{\sqrt{2}} + \Big(|1\rangle\langle 1| \otimes X\Big)\frac{|00\rangle+|10\rangle}{\sqrt{2}}$$

The first term is

$$\begin{aligned}\Big(|0\rangle\langle 0| \otimes \mathbb{1}\Big)\frac{|00\rangle+|10\rangle}{\sqrt{2}} &= \frac{1}{\sqrt{2}}\Big(|0\rangle\langle 0| \otimes \mathbb{1}\Big)|0\rangle \otimes |0\rangle + \frac{1}{\sqrt{2}}\Big(|0\rangle\langle 0| \otimes \mathbb{1}\Big)|1\rangle \otimes |0\rangle \\ &= \frac{1}{\sqrt{2}}|0\rangle\langle 0|0\rangle \otimes |0\rangle + \frac{1}{\sqrt{2}}|0\rangle\langle 0|1\rangle \otimes |0\rangle \\ &= \frac{1}{\sqrt{2}}|0\rangle \otimes |0\rangle\end{aligned}$$

and the second term is analogous

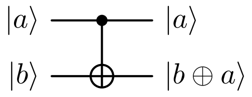
$$\begin{aligned}\Big(|1\rangle\langle 1| \otimes X\Big)\frac{|00\rangle+|10\rangle}{\sqrt{2}} &= \frac{1}{\sqrt{2}}\Big(|1\rangle\langle 1| \otimes X\Big)|0\rangle \otimes |0\rangle + \frac{1}{\sqrt{2}}\Big(|1\rangle\langle 1| \otimes X\Big)|1\rangle \otimes |0\rangle \\ &= \frac{1}{\sqrt{2}}|1\rangle\langle 1|0\rangle \otimes X|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\langle 1|1\rangle \otimes X|0\rangle \\ &= \frac{1}{\sqrt{2}}|1\rangle \otimes X|1\rangle \ .\end{aligned}$$

We conclude that the final state is $\frac{|0\rangle\otimes|0\rangle+|1\rangle\otimes X|0\rangle}{\sqrt{2}} = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$.

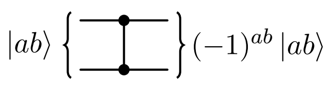
Generally, $C(U)$ will carry out the transformation

$$(\alpha|0\rangle + \beta|1\rangle) \otimes |\psi\rangle \mapsto \alpha|0\rangle \otimes |\psi\rangle + \beta|1\rangle \otimes U|\psi\rangle \ .$$

The controlled-$X$ like the one in Example 2.1 is actually a very special gate, which we call *controlled NOT* or *CNOT*, and has its own symbol



As we said earlier for controlled gates, the CNOT follows the same idea: if the control qubit is in the $|1\rangle$ state, then flip the target, otherwise do nothing. As you might have noticed, the $\oplus$ symbol in the circuit suggests that the target qubit is XOR'ed to the control qubit $b \leftarrow b \oplus a$. Also the controlled-$Z$ or $CZ$ gate has its own symbol



The CZ gate applies a phase $-1$ if and only if $a = b = 1$, and leaves the state unchanged otherwise.

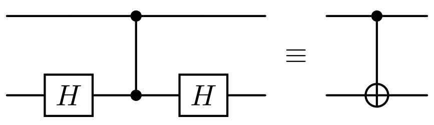
**Remark 2.2.** You might be wondering what the CZ actually does, since it seems to only apply global phases. Remember, however, that $|ab\rangle$ for $a, b \in \{0, 1\}$ are not the only possible states. If we apply a CZ to $|+\rangle \otimes |+\rangle$, then

$$\begin{aligned} |+\rangle \otimes |+\rangle &= \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \\ &= \frac{|00\rangle + |01\rangle + |10\rangle + |11\rangle}{2} \\ &\overset{CZ}{\mapsto} \frac{|00\rangle + |01\rangle + |10\rangle - |11\rangle}{2} \end{aligned}$$

Only the $|11\rangle$ component gets its phase flipped, so there is a change in the relative phase.

**Remark 2.3.** The symbol for the CZ gate is symmetric because its action doesn't change regardless of which qubit we consider 'the control' or 'the target'.

**Example 2.4.** Let us prove the following equivalence:

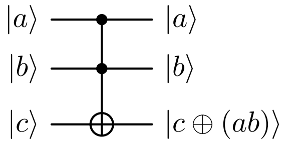


Remember that if two matrices give the same results for a basis, then by linearity the equivalence is extended to all the linear combinations as well. In this case, showing the equivalence for the standard basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ seems the easiest path.

$$\begin{aligned} |00\rangle &\overset{I\otimes H}{\mapsto} |0\rangle \otimes |+\rangle = \frac{|00\rangle + |01\rangle}{\sqrt{2}} \overset{CZ}{\mapsto} \frac{|00\rangle + |01\rangle}{\sqrt{2}} = |0\rangle \otimes |+\rangle \overset{I\otimes H}{\mapsto} |00\rangle \\ |01\rangle &\overset{I\otimes H}{\mapsto} |0\rangle \otimes |-\rangle = \frac{|00\rangle - |01\rangle}{\sqrt{2}} \overset{CZ}{\mapsto} \frac{|00\rangle - |01\rangle}{\sqrt{2}} = |0\rangle \otimes |-\rangle \overset{I\otimes H}{\mapsto} |01\rangle \\ |10\rangle &\overset{I\otimes H}{\mapsto} |1\rangle \otimes |+\rangle = \frac{|10\rangle + |11\rangle}{\sqrt{2}} \overset{CZ}{\mapsto} \frac{|10\rangle - |11\rangle}{\sqrt{2}} = |1\rangle \otimes |-\rangle \overset{I\otimes H}{\mapsto} |11\rangle \\ |11\rangle &\overset{I\otimes H}{\mapsto} |1\rangle \otimes |-\rangle = \frac{|10\rangle - |11\rangle}{\sqrt{2}} \overset{CZ}{\mapsto} \frac{|10\rangle + |11\rangle}{\sqrt{2}} = |1\rangle \otimes |+\rangle \overset{I\otimes H}{\mapsto} |10\rangle \end{aligned}$$

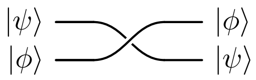
Therefore, the second qubit is flipped only when the first is in the state $|1\rangle$, which is exactly what the CNOT does. Notice that the CZ flips the phase only in the last two cases, when it encounters a $|11\rangle$ component.

The control can be from more than one qubit. The following gate — which we denote with $C^2(X)$ — flips the third qubit only when the two control qubits are in the $|1\rangle$ state.



This is a two-way CNOT, also known as the *Toffoli gate*. Similarly, we can replace the $\oplus$ symbol with any unitary, and define gates with any number of control qubits. The cost of implementing $n$-way controls, however, will not be free, and we will require a number of one- and two-qubit gates that scales with $n$, as we will see.

Last but not least, we introduce the SWAP gate, whose effect is self-explanatory:



A SWAP gate can be implemented easily with CNOTs (see Exercise 2.3), but it actually has the same effect as physically swapping the two qubits, and this property is very important for the implementation of quantum architectures.

## 2.3 Universal gate sets

If you're familiar with classical computers, you might know that usually every CPU architecture has its own set of basic instructions involving a fixed amount of bits. Usually these bits are grouped into 32- or 64-bit registers and operations are carried out natively on these units, and nowadays we tend to use big registers even if we need only a couple of bits. In quantum computation — at least with the current state of the art in terms of hardware —, we don't have the luxury of wasting so many qubits, but the idea of basic instructions is the same: we fix a gate set acting on a fixed set of qubits (usually 1 or 2), and we only have them available for our algorithms.

**Definition 2.5.** A *gate set* is a finite set of unitaries $\mathcal{G} = \{G_1, \dots, G_K\}$ where $G_j$ acts on a fixed number $m_j$ of qubits.

Of course, we have to choose an adequate gate set to get a satisfactory set of instructions. Here with 'satisfactory' we intend that the set of unitary operations that we can obtain by composing gates from the set is big. For example, the set $\{Z\}$ can only generate either $I$ or $Z$ (notice that $Z^2 = I$), while the set $\{X, Z\}$ can generate $I, X, Y, Z$. Let us focus on single-qubit gates for a minute:

**Definition 2.6.** The *special unitary group* of dimension 2, denoted with $SU(2)$, is the group of $2 \times 2$ unitary matrices with determinant 1.

$$SU(2) = \left\{ \begin{bmatrix} \alpha & \beta \\ -\beta^* & \alpha^* \end{bmatrix} : |\alpha|^2 + |\beta|^2 = 1 \right\}$$

We care about the *special* unitary group (which adds the determinant condition) instead of the full unitary group $U(2)$, because any unitary can be simply expressed as a special unitary times a global phase, which has no physical meaning as we said in Chapter 1. The best possible result we could hope for would be to construct *any* $U \in SU(2)$ using strings[1] of gates from a gate set.

There is a problem, however: we can only have countably many strings of gates, but the matrices in $SU(2)$ are uncountably many — we can choose $\alpha$ to be any real in $(0, 1)$ even if we avoid complex numbers —, so it is certainly impossible to construct all the possible $U \in SU(2)$. Remember, however, that if we get sufficiently close to our desired unitary, then the effects are statistically indistinguishable.

[1]I'm going to use the term string to indicate the product of elements from a certain gate set. For example, if we have a gate set $\{A, B\}$, then $ABBABA$ is a string of gates of length 6. By carrying out the matrix multiplication, we obtain a new unitary. Of course, different strings might turn out to generate the same matrix.

Given a gate set $\mathcal{G}$, we use $\langle\mathcal{G}\rangle$ to denote the subgroup of $SU(2)$ generated by $\mathcal{G}$, i.e., the group of all the possible multiplications of elements of $\mathcal{G}$.

**Definition 2.7.** A single-qubit gate set $\mathcal{G}$ is said to be *universal* if $\langle\mathcal{G}\rangle$ is dense in $SU(2)$, i.e., for any $U \in SU(2)$ and $\epsilon > 0$, there exists an element $C \in \langle\mathcal{G}\rangle$ such that

$$\|C - U\| \leq \epsilon \ .$$

Here the concept of density is exactly like the one used for rationals. If you remember from calculus courses, $\mathbb{Q}$ is dense in $\mathbb{R}$ because any real number can be arbitraily well-approximated by some rational number with sufficiently many digits after the decimal point, and here the concept is the same: any $SU(2)$ unitary can be well-approximated by a (sufficiently long) sequence of gates from a universal gate set.

**Theorem 2.8** ([4])**.** *The set $\{H, T\}$ is universal for single-qubit unitaries.*

The proof of this theorem comes from group theory, which is out of the scope of these notes. While we know we can reach (at least approximately) any unitary using $\{H, T\}$, the next question is *how efficiently?* Perhaps there are unitaries that require a very long string of gates to be approximated satisfactorily. Here comes what's probably one of the most fundamental theorems of quantum computation.

**Theorem 2.9** (Solovay-Kitaev)**.** *Fix a universal set of gates $\mathcal{G}$ which contains its own inverses, i.e., if $A \in \mathcal{G}$ then also $A^\dagger \in \mathcal{G}$. There exists a constant $c \leq 4$ such that any $U \in SU(2)$ can be approximated up to $\epsilon$-precision by a string of gates in $\mathcal{G}$ of length $\mathcal{O}(\log^c \frac{1}{\epsilon})$.*

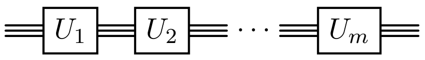
The Solovay-Kitaev theorem tells us that the approximation of $SU(2)$ unitaries is *extremely efficient*, as long as (i) we provide a universal gate set and (2) the set contains its own inverses. For example, the set $\{H, T\}$ is universal, but we should also provide $T^\dagger$, so $\{H, T, T^\dagger\}$ satisfies the conditions of the theorem (remember that $H$ is Hermitian, so it is the inverse of itself). The exponent of the logarithm $c$ depends on the actual gate set we use, but it's certainly $\leq 4$, and there are some very practical universal gate sets for which $c = 1$. The proof of the Solovay-Kitaev theorem (which we delay to Chapter **??**) also provides a recursive algorithm to produce the string for a given desired unitary.

We mention that there exists a SK theorem without inverses (but slightly worse $c$) [5]. Moreover, we also state without proof, that the CNOT is enough to go beyond the single-qubit gates.

**Theorem 2.10.** *If $\mathcal{G}$ is a universal gate set for single-qubit unitaries, then $\mathcal{G} \cup \{CNOT\}$ is universal for all unitaries.*

Where 'universal for all unitaries' means that any unitary $U \in SU(2^n)$ (the set of $n$-qubit unitaries with unit determinants), can be arbitrarily approximated by some quantum circuit involving only gates from $\mathcal{G}$ and CNOTs. However in this case not all unitaries can be approximated *efficiently.*

**Remark 2.11.** Suppose we have a quantum circuit represented by the product of unitaries $U_m U_{m-1} \cdots U_1$



Unfortunately, there is one of these unitaries, say $U_j$ for some $j$, that can only implemented approximately, i.e., we only have a circuit implementing $V_j$, which satisfies

$$\|V_j - U_j\| \leq \epsilon \ .$$

Therefore, the circuit we implemented yields the unitary

$$U_m U_{m-1} \cdots U_{j+1} V_j U_{j-1} \cdots U_1$$

How close is this resulting unitary to the original transformation? For simplicity, let's call $A := U_m \cdots U_{j+1}, B := U_{j+1} \cdots U_1$ the unitaries implemented by the gates after and before $V_j$, respectively. The distance is thus:

$$\|AV_jB - AU_jB\| = \|A(V_j - U_j)B\| = \|V_j - U_j\| \leq \epsilon$$

where the last equality uses unitary invariance of the operator norm (see Exercise 1.8). This implies that the error introduced by approximating some parts of the circuits are not amplified by the rest of the circuit.

## 2.4 The quantum machine

We are now ready to construct the theoretical model of a quantum computer. Once we fix a universal gate set $\mathcal{G}$, any circuit above can be written as a sequence of statements specifying which gate to apply and on which qubit(s) or, in case of measurement, which qubits we want to measure[2].

In the model we present here — which we'll implicitly use when we devise quantum algorithms —, we have a classical controller (e.g. a Turing machine) which sends instructions to a quantum unit, whose register qubits are all initialized to $|0\rangle$. Whenever the controller sends a measurement instruction, the quantum unit will return the corresponding measurement outcome.

Current quantum architectures actually *rarely* allow this dynamic interplay between the controller and the quantum computer, because during the time we wait for the controller to send the next instruction, the quantum unit might lose information because of decoherence (and we are talking about microseconds, if not nanoseconds!). Usually, classical algorithms just send jobs (with a full circuit with measurements at the end), and the quantum computer will just execute the whole circuit and send the measurement outcome to the requester. The only difference with our model is that we cannot continue the execution with the post-measurement states if we require a measurement, but it's not a dramatic difference.

Theory, on the other hand, tells us that the quantum unit could do the whole work, while the classical controller should only care about constructing the quantum circuit (we will see in Chapter 3 that quantum circuits can carry out any classical computation). Also this model is equivalent to the one we initially stated, but it would be an unnecessary pain to write down quantum circuits to do something a simple procedure written in your favorite programming language would easily take care of.

**Definition 2.12.** A *quantum machine* is a random access machine equipped with additional instructions corresponding to a universal unitary gate set, plus a measurement instruction.

See Figure 2.1 for a scheme of the quantum machine. The random access machine (RAM) is the model usually used to specify classical algorithms, which has access to an infinite memory.

[2]This is actually done in practice, see for example the quantum assembly language.

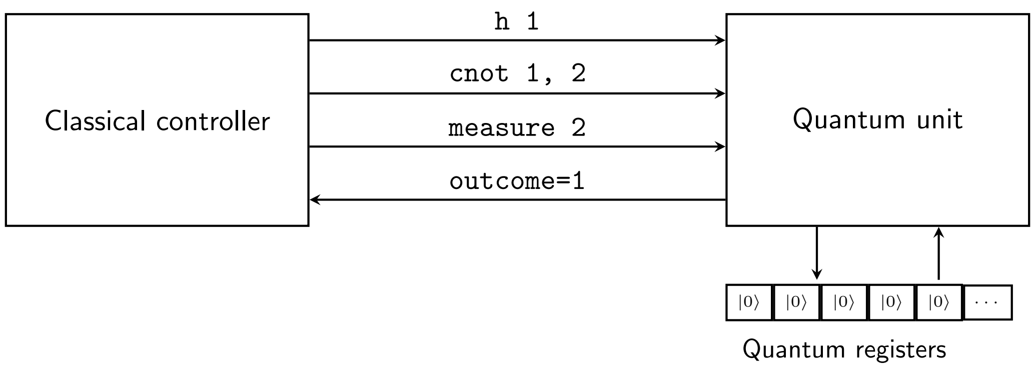


Figure 2.1: Scheme of a quantum machine. The classical controller sends commands to the quantum unit, which manipulates quantum registers containing qubits. In the example, the controller asks to apply $H$ to qubit 1 and a CNOT to qubits 1, 2. Then it asks to measure qubit 2, and the unit responds with the measurement outcome.

The registers of this memory are indexed by the natural numbers which are then specified in the instructions. For example, `add 1, 4, 5` tells the RAM to add the contents of the registers 1 and 4, and write the result to register 5. Similarly, the quantum unit will have its own memory with infinitely many qubits, which are indexed by the natural numbers as well. If we consider a universal gate set containing $H$, CNOT, additional instructions for the RAM will include for example `h 1`, which tells to apply $H$ to qubit 1, or `cnot 5, 7`, to apply a CNOT to qubits 5 and 7. Finally, the instruction `measure 2, 5` would measure qubit 2 and write the (classical) result to register 5[3].

The input is given as a classical string $x \in \{0,1\}^n$ in the classical registers. We now define efficiency.

**Definition 2.13.** The *time complexity* $\tau(x)$ of the quantum machine is the number of steps executed before halting when given input $x$. The quantum machine is said to be *polynomial-time* if $\tau(x) = \mathcal{O}(n^k)$ for some integer $k$, where $n$ is the length of $x$.

**Definition 2.14.** The *space complexity* $\sigma(x)$ is the total number of registers and qubits used by the machine during the execution when given input $x$. The quantum machine is said to be *polynomial-space* if $\sigma(x) = \mathcal{O}(n^k)$ for some integer $k$, where $n$ is the length of $x$.

Of course, we won't specify our algorithms using such low-level instructions: we will use pseudocode as with classical algorithms, and by implicitly invoking the Solovay-Kitaev theorem, we will assume to be able to apply any $\mathcal{O}(1)$-qubit gate in $\mathcal{O}(1)$ time.

## 2.5 Detour on complexity classes

Now we have a full and working model of quantum computation, and we will define the corresponding *complexity classes*. Problems are formalized as decision problems (or languages), i.e., problems for which a yes/no answer must be provided.

**Definition 2.15.** A *language* is a subset $L \subseteq \{0,1\}^*$ of the set of all the bit strings.

[3]With measurement we always mean with respect to the standard basis $\{|0\rangle, |1\rangle\}$. Carrying out measurement in arbitrary bases is actually a hard task in general, see Exercises 2.5, 2.6.

**Example 2.16.** In the FACTORING problem, the input is a natural number $x$ and another number $0 < d < x$ (both given as binary strings, so the input length is $n = \lceil \log_2 x \rceil + \lceil \log_2 d \rceil = \Theta(\log_2 x)$). $(x, d) \in$ FACTORING if and only if $x$ has a non-trivial prime factor $p$ between 1 and $d$. This is the decision version of the problem of returning the prime factorization of a given number $x$.

**Example 2.17.** In the SAT problem, the input is a propositional logic formula $F$ in conjunctive normal form, e.g.,

$$(y_1 \wedge \overline{y_4} \wedge \overline{y_5}) \vee (y_2 \wedge y_3) \vee (\overline{y_2} \wedge y_4 \wedge x_6) \ .$$

If $n$ is the number of variables and $m$ is the number of clauses, the formula can be expressed with a string of length $\mathcal{O}(nm)$. Depending on the assignment $y \in \{0,1\}^n$, the input formula will be either true or false. $F \in$ SAT if the formula is satisfiable, i.e., there exists $y$ that makes the formula true.

We then say that an algorithm either accepts (resp. rejects) an input $x$ if it returns 1 (resp. 0) on that input. Let us define the classical complexity classes first:

**Definition 2.18.** The complexity class $P$ is defined as the set of problems $L$ solved by some polynomial-time classical algorithm $\mathcal{A}$. Specifically, here solving means that $\mathcal{A}$ accepts the input $x$ if and only if $x \in L$.

In practice, $P$ is the set of all problems we can efficiently solve with a deterministic classical algorithm.

**Definition 2.19.** The complexity class $NP$ is defined as the set of problems $L$ verified by some polynomial-time classical algorithm $\mathcal{A}$. Here, the verifier $\mathcal{A}$ must be such that

(i) if $x \in L$ then there exists a polynomial-length string $y$ such that $\mathcal{A}$ accepts the input $(x, y)$.

(ii) if $x \in L$, no string $y$ will make $\mathcal{A}$ accept $(x, y)$.

The string $y$ in the above is a *proof* that certifies that $x$ should be accepted. For example, a proof for $(x, d) \in$ FACTORING would be a prime factor $y < d$ of $x$, and the verifier would simply check that $y$ is indeed a prime factor of $x$. To certify that $F \in$ SAT, a proof would be an assignment $y \in \{0,1\}^n$ to the variables that makes $F$ true, and a possible verifier would check if that $y$ is a satisfying assignment for $F$. By what we just said, we know that FACTORING, SAT $\in NP$.

Understanding whether efficiently verifying a problem is equivalent to efficiently solving it (in other words, $P = NP$) is arguably the most important open problem in computer science. To have a better understanding of complexity, we informally introduce the notion of reduction.

**Definition 2.20.** A language $L$ is said to have a *polynomial-time reduction* to a language $L'$ (written $L \leq L'$) if, given an algorithm $\mathcal{A}_{L'}$ solving $L'$ we can construct a polynomial-time algorithm $\mathcal{A}_L$ solving $L$ which calls $\mathcal{A}_{L'}$ polynomially many times.

By this definition, if $\mathcal{A}_{L'}$ turns out to be polynomial-time (so $L' \in P$), then $\mathcal{A}_L$ will be polynomial-time as well (and thus also $L \in P$).

**Definition 2.21.** For a complexity class $\mathcal{L}$, a problem $S$ is said to be $\mathcal{L}$-hard if $L \leq S$ for every $L \in \mathcal{L}$. Moreover, if $S$ is also in $\mathcal{L}$, then the problem is said to be $\mathcal{L}$-complete.

A celebrated result in complexity theory known as the *Cook-Levin theorem* proves that SAT is $NP$-complete and, therefore, if there turns out to be a polynomial-time algorithm solving it, then this would make $NP$ collapse to $P$.

Before we can turn our attention to the quantum complexity classes, we need to take a detour for *randomized algorithms*, which are essentially classical algorithms which can take random choices (or equivalently, have access to a uniformly random string $r$ of polynomial length). This is important because quantum algorithms are essentially randomized algorithms with extra steps.

**Definition 2.22.** The *bounded-error polynomial time* class $BPP$ is the class of problems $L$ for which there exists a polynomial-time randomized algorithm $\mathcal{A}$ such that

(i) if $x \in L$, then $\Pr[\mathcal{A}(x;r) \text{ accepts}] \geq 2/3$;

(ii) if $x \notin L$, then $\Pr[\mathcal{A}(x;r) \text{ accepts}] \leq 1/3$.

where the probabilities are with respect to the uniformly random string $r$.

Essentially $BPP$ is similar to $P$, in the sense that $\mathcal{A}$ must decide whether $x \in L$, but it is allowed to be wrong sometimes. Notice that there is nothing special in the bounds $(2/3, 1/3)$: as long as they are bounded away from $1/2$, they can be made arbitrarily close to $(1, 0)$ by simply repeating the algorithm a constant number of times. On the other side we have the randomized version of $NP$.

**Definition 2.23.** The *Merlin-Arthur* class $MA$ is the class of problems $L$ for which there exists a polynomial-time randomized algorithm $\mathcal{A}$ such that

(i) if $x \in L$, then there exists a proof $y$ such that $\Pr[\mathcal{A}(x, y; r) \text{ accepts}] \geq 2/3$;

(ii) if $x \notin L$, then for any proof $y$, $\Pr[\mathcal{A}(x;r) \text{ accepts}] \leq 1/3$.

where the probabilities are with respect to the uniformly random string $r$.

Ok, we probably should give a motivation for why this class is called *Merlin-Arthur*, and the reason comes from cryptography, where this class first arised. Essentially in this game we have the prover Merlin which uses his mysterious magic to construct a proof $y$ to convince Arthur (a polynomial-time verifier) that $x \in L$. Arthur doesn't know how Merlin generated the proof, but he can understand by himself that the proof is correct and the input should be accepted. The only difference with the $NP$ class is that the verifier can be wrong with some bounded probability, exactly like in the definition of $BPP$.

The worst part is over, as the classes $BQP$ and $QMA$ are defined analogously.

**Definition 2.24.** The *bounded-error quantum polynomial time* class $BQP$ is the class of problems $L$ for which there exists a polynomial-time quantum algorithm $\mathcal{A}$ such that

(i) if $x \in L$, then $\Pr[\mathcal{A}(x) \text{ accepts}] \geq 2/3$;

(ii) if $x \notin L$, then $\Pr[\mathcal{A}(x) \text{ accepts}] \leq 1/3$.

where the probabilities are with respect to the measurement outcomes.

Nothing really changes from $BPP$, except that the quantum machine doesn't really need to be fed with a uniformly random string $r$, as it can generate random bits by itself: simply take a qubit initialized to $|0\rangle$ and apply $H$. By measuring in the standard basis, the outcome is a uniformly random bit.

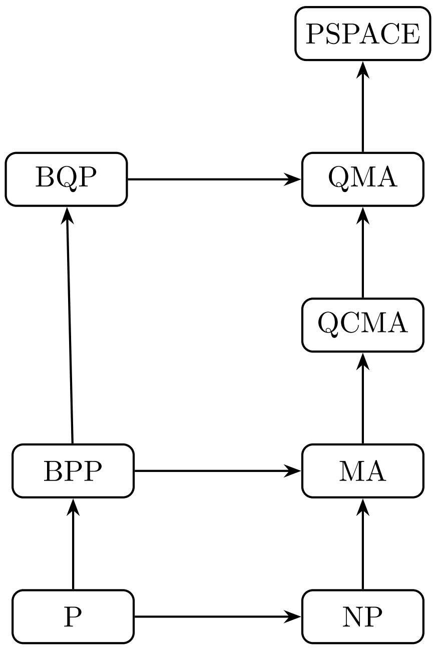

$$\begin{tikzcd}
 & PSPACE \\
BQP \arrow[r] & QMA \arrow[u] \\
 & QCMA \arrow[u] \\
BPP \arrow[uu] \arrow[r] & MA \arrow[u] \\
P \arrow[u] \arrow[r] & NP \arrow[u]
\end{tikzcd}$$

Figure 2.2: A tiny portion of the complexity zoo. The arrows represent inclusion of the classes.

**Definition 2.25.** The *quantum Merlin-Arthur* class $QMA$ is the class of problems $L$ for which there exists a polynomial-time quantum algorithm $\mathcal{A}$ such that

(i) if $x \in L$, then there exists a proof $|\psi\rangle$ such that $\Pr[\mathcal{A}(x, |\psi\rangle) \text{ accepts}] \geq 2/3$;

(ii) if $x \notin L$, then for every proof $|\psi\rangle$, $\Pr[\mathcal{A}(x) \text{ accepts}] \leq 1/3$.

where the probabilities are with respect to the measurement outcomes.

Importantly here, in the $QMA$ game Merlin will generate a *quantum state* as a proof (which we can expect to be placed in the quantum register of the machine defined in Section 2.4)[4].

The reader can convince themself that $P \subseteq BPP \subseteq BQP$ and $NP \subseteq MA \subseteq QMA$. Moreover, we know that $P \subseteq NP$ and analogously $BPP \subseteq MA, BQP \subseteq QMA$. But as we don't know whether $P = NP$, we don't even know whether $BPP = MA, BQP = QMA$.

What we *do* know is that all these classes are contained in $PSPACE$, the class of problems solvable by a polynomial-space classical algorithm (yes, we know a classical algorithm that can simulate a quantum computer in polynomial space, but not polynomial time). Moreover, we know that FACTORING $\in BQP$ (see Chapter 6), but we don't know whether it is also in $BPP$ or $P$. Figure 2.2 summarizes the classes we introduced.

## 2.6 Exercises

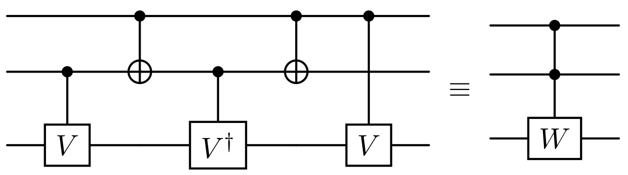
**Exercise 2.1.** Prove by induction on $n$ that

$$|+\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle$$

[4]If we instead forced Merlin to give a proof as a classical string $y$, then we would get a very different class called $QCMA$, which we're not going to discuss here.

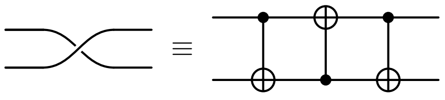
i.e., the $n$-qubit state where all qubits are in the $|+\rangle$ state corresponds to the equal superposition of all the possible bit strings of length $n$.

**Exercise 2.2** (Sleator-Weinfurter construction)**.** Show that the following equivalence holds:



where $V$ is some unitary matrix satisfying $V^2 = W$.

**Exercise 2.3.** Show that the following equivalence holds:



**Exercise 2.4.** Suppose that we have a quantum circuit applying gates $U_1, U_2, \ldots, U_m$ in this order, thus achieving the transformation

$$U_m U_{m-1} \cdots U_1 \ .$$

For every $j$ we replace $U_j$ with $V_j$, where $\|U_j - V_j\| \le \epsilon_j$. Show that

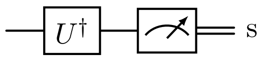
$$\|U_m U_{m-1} \cdots U_1 - V_m V_{m-1} \cdots V_1\| \le \epsilon_1 + \cdots + \epsilon_m \ .$$

To which value can we set each $\epsilon_j$ to finally obtain a total error of $\epsilon$ on the whole circuit?

**Exercise 2.5.** Let $\mathcal{B} = \{|\phi_0\rangle, |\phi_1\rangle\}$ be a basis for the state space of a qubit, and consider a single-qubit unitary $U$ satisfying:

$$U|0\rangle = |\phi_0\rangle$$
$$U|1\rangle = |\phi_1\rangle$$

i.e., $|\phi_0\rangle, |\phi_1\rangle$ are the columns of $U$. Show that the circuit

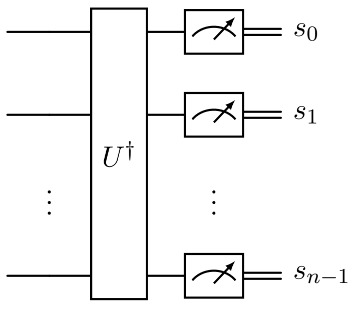


carries out a measurement in the $\mathcal{B}$ basis, i.e., $s$ will be returned if the state is $|\phi_s\rangle$.

**Exercise 2.6.** Let $\mathcal{B} = \{|\phi_0\rangle, |\phi_1\rangle, \ldots, |\phi_{2^n-1}\rangle\}$ be a basis for the state space of a $n$-qubit register, and consider a $n$-qubit unitary $U$ satisfying:

$$U|k\rangle = |\phi_k\rangle$$

for every $0 \le k < 2^n$. Show that the circuit



carries out a measurement in the $\mathcal{B}$ basis, i.e., considering $s = s_{n-1}s_{n-2}\cdots s_0$ be the integer whose $n$-bit representation is given by the $s_k$ bits, $s$ will be returned if the initial state is $|\phi_s\rangle$.

**Exercise 2.7.** Reduce prime factorization to its decision version. That is, use an algorithm for the decision version of Factoring to construct a polynomial-time algorithm that outputs the prime factors of a given number $x$.

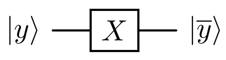
# Chapter 3

# General reversible computation

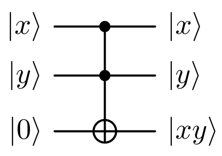
In the previous chapter we defined a model of computation to capture quantum computers. However, we said that the quantum unit could theoretically perform the whole computation by itself, while the classical unit only constructs the quantum circuit to be executed, and this is usually how the quantum circuit model is defined. The fact that Definition 2.12 is equivalent relies on the fact that quantum circuits can simulate arbitrary classical computations. The only caveat is that classical computations are not necessarily reversible, while we saw that unitaries are. In this chapter we will see how to carry out general classical computations on a quantum circuit, circumventing the reversibility problem.

## 3.1 Carrying out classical computation on a quantum computer

Consider a function $f : \{0,1\}^n \to \{0,1\}^m$ that is computed by some classical algorithm. Ultimately, the classical algorithm can be seen as a boolean circuit[1]. Therefore, we can assume that a boolean circuit computing $f$ is given to us. Furthermore, we know that using only AND and NOT gates is enough to compute any function. For example, the OR function can be realized with AND and NOT as $x + y = \overline{\overline{x} \cdot \overline{y}}$ by de Morgan's rule. We thus only need to emulate the NOT and the AND gates. The first one is easy because the NOT gate is reversible, and we know that the Pauli-$X$ gate does exactly that in the standard basis:

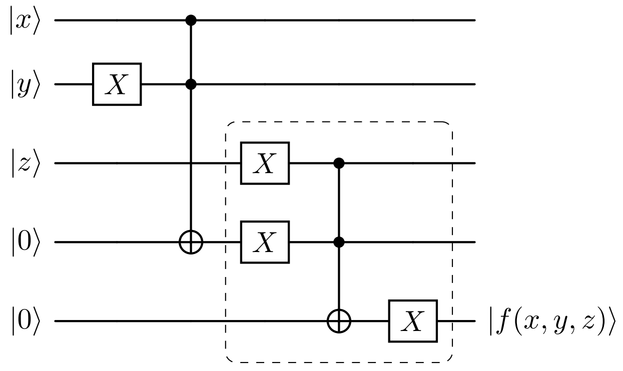


The AND function, on the other hand, is not reversible, but it is still possible if we use an extra qubit and the Toffoli gate:



**Example 3.1.** Consider the function $f(x, y, z) = x\overline{y} + z$. A possible quantum circuit for $f$ is:

[1]To convince yourself, think that CPUs simply 'decide' which operations available in their arithmetic-logic unit (ALU) to apply to the registers, and such operations are implemented with logical circuits. If you unroll this computation (i.e., you write down all the operations applied to the data), you get an equivalent boolean circuit.



The boxed part is the implementation of the OR relying on de Morgan's rule.

We remark that if $f$ can be computed with a logical circuit of $K$ AND and NOT gates, then there exists a quantum circuit using $K$ Toffoli and $X$ gates, therefore the quantum time complexity needed to compute $f$ is at most the classical one.

## 3.2 Bennett's uncomputation trick and the bit oracle

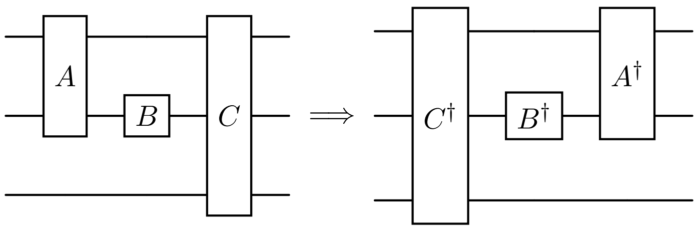
While the circuit constructed in the previous section (let's call it $C_f$) can actually be used to simulate the classical computation, there is a caveat: we need extra qubits to store the intermediate computations, and after we finished the computations, these qubits remain 'dirty'. We might reset them, but the reset operation $|1\rangle, |0\rangle \mapsto |0\rangle$ is irreversible, and remember that we can only apply reversible operations[2].

We want to get back as many qubits as possible, while still computing $f$, and here is where the *uncomputation trick* comes into play: since $C_f$ is a reversible circuit, we can also construct the inverse circuit $C_f^\dagger$, and it is actually very simple: since the conjugate transpose satisfies

$$(AB)^\dagger = B^\dagger A^\dagger$$

it is sufficient to take the gates for $C_f$ and apply their inverses in reverse order.

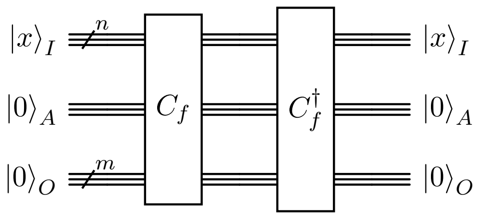
**Example 3.2.** Let us invert the following circuit:



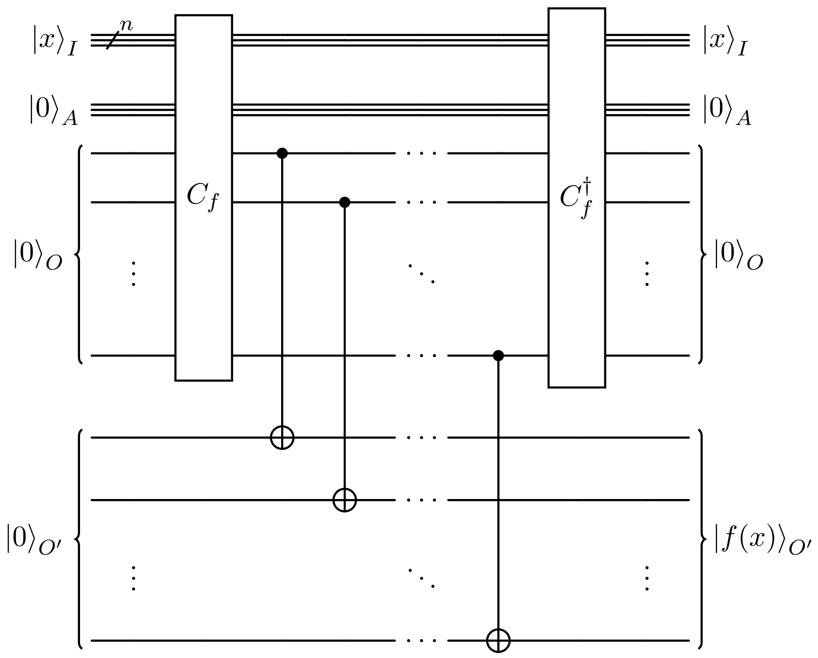
If the circuit on the left implements the unitary $U$, then the one on the right implements $U^\dagger$.

Luckily for us, the Toffoli and $X$ are the inverse of themselves (check it!), and so we only have to apply them in reverse order to construct $C_f^\dagger$. For an input $x \in \{0,1\}^n$, we first construct something appearently stupid:

[2]Of course reset operations are available in quantum architectures, but they are essentially 'swap the information of the qubit with the information of some external system that we know to be 0', and swapping is a reversible operation. But if the register was in an entangled state, then after the swap operation the qubits we did not reset will stay entangled with the external system, and this is an example of decoherence phenomenon we already mentioned.



where $I$ is the input register holding our $x$ at the beginning, $A$ is the set of *ancilla* (helper) qubits we use to store intermediate results, and $O$ is the register of the $m$ qubits holding the output. This circuit computes $f$ and then *uncomputes* it, bringing the state of the registers back to the beginning, but in the middle the $O$ register actually contains $|f(x)\rangle$. So what we do is simply copying it to another output register $O'$ using CNOTs.



The $A$ and $O$ registers are still starting from the $|0\rangle$ state and ending at the same state regardless of the input and thus can be hidden (think of them like the local variables of a function). The time complexity of this circuit is $2T + m = \mathcal{O}(T)$, where $T$ is the time complexity of $C_f$ (and thus of the original classical algorithm).

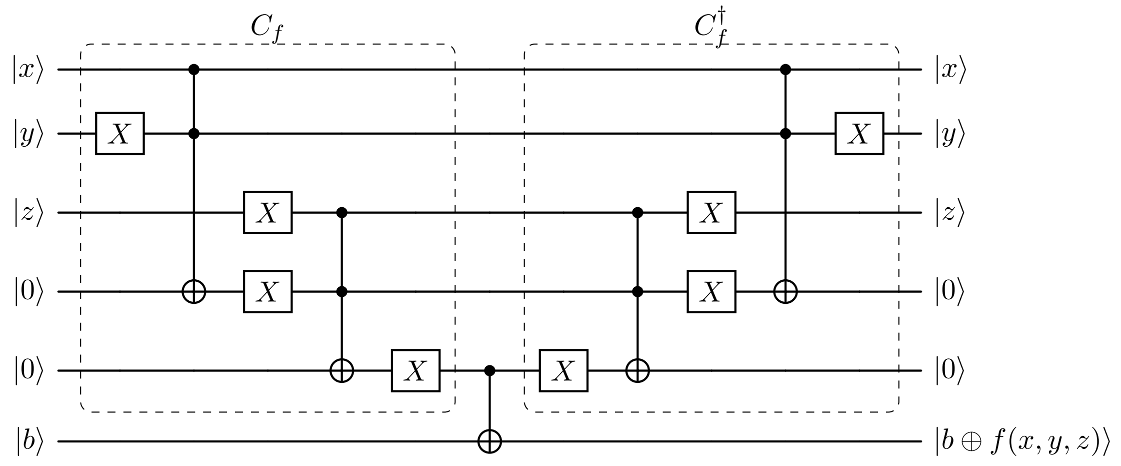
**Definition 3.3.** Given a function $f$, the *bit oracle* $O_f$ is the unitary satisfying:

$$O_f |x\rangle_I |s\rangle_{O'} = |x\rangle_I |s \oplus f(x)\rangle_{O'}$$

for any $x \in \{0,1\}^n, s \in \{0,1\}^m$. Here $\oplus$ denotes the XOR between two bit strings. $O_f$ can be implemented in $\mathcal{O}(T)$ complexity given a classical algorithm running in the same time.

Notice that the XOR (which is a reversible operation) is what the CNOTs really do, so if $O'$ doesn't start from the $|0\rangle$ state, the output is generally XORed to the register.

**Example 3.4.** Taking the function $f(x,y,z) = x\overline{y} + z$ and its circuit $C_f$ from Example 3.1, a bit oracle for $f$ can be implemented with the following circuit:



## 3.3 Quantum parallelism (and why you shouldn't fall for it)

We've seen how to construct the bit oracle $O_f$ for some function $f$, preserving the classical complexity to compute it. But why do we even bother? In the end, we could compute $f$ even with a classical computer. The answer is that we can call the bit oracle for $f(x)$ even when the input register $I$ contains a superposition of different $x$. Suppose, for example, we start from the equal superposition

$$|\phi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle$$

which is very easy to construct with $n$ parallel Hadamard gates (see Exercise 2.1). Let's compute the final state:

$$\begin{aligned} O_f\,|\phi\rangle_I\,|0\rangle_O &= O_f \left( \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle_I \right) |0\rangle_O \\ &= \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} O_f\,|x\rangle_I\,|0\rangle_O && \text{by linearity} \\ &= \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle_I\,|f(x)\rangle_O \end{aligned}$$

Linearity implies that $O_f$ is applied to each component of the superposition 'in parallel', resulting in a final state where the registers $I, O$ are generally entangled. In each of these components, we get all the different possible outputs of $f$, with only one call to $O_f$. This is an extremely important difference from classical computation, where the oracle can only give you one possible output of $f$ per call.

Let's consider an instance SAT: take a logical formula $f(x)$ and construct the bit oracle of a circuit computing $f$. If we call $S \subseteq \{0,1\}^n$ the set of satisfying assignments for $f$ then the final state will be

$$\frac{1}{\sqrt{2^n}} \left( \sum_{x\in S} |x\rangle_I\,|1\rangle_O + \sum_{x\in\{0,1\}^n\setminus S} |x\rangle_I\,|0\rangle_O \right)$$

and the first sum is empty if $f$ is not satisfiable. So I can use this quantum parallelism to solve SAT on a quantum computer in polynomial time? **NO**! Or at least, we don't know. Let's see why.

An algorithm that solves SAT should correctly distinguish the case where $f$ is unsatisfiable (and return 'no') from the case where $f$ is satisfied by only one assignment $x^*$ (in which case it must return 'yes'). The final states in the two cases are

$$
\begin{aligned}
|\psi_{yes}\rangle &:= \frac{1}{\sqrt{2^n}}\left(|x^*\rangle_I |1\rangle_O + \sum_{x\neq x^*} |x\rangle_I |0\rangle_O\right) \\
|\psi_{no}\rangle &:= \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}} |x\rangle_I |0\rangle_O
\end{aligned}
$$

Let's measure qubit $O$, so using a projective measurement $\{\Pi_0 = \mathbb{1}_I \otimes |0\rangle\langle 0|_O, \Pi_1 = \mathbb{1}_I \otimes |1\rangle\langle 1|_O\}$. It's no surprise that $\Pi_1$ will never be detected on $|\psi_{no}\rangle$, let's compute the probability of detecting it on $|\psi_{yes}\rangle$.

$$
\begin{aligned}
\langle\psi_{yes}| \Pi_1 |\psi_{yes}\rangle &= \frac{1}{2^n}\left(\langle x^*|_I \langle 1|_O + \sum_{x\neq x^*} \langle x|_I \langle 0|_O\right) \Pi_1 \left(|x^*\rangle_I |1\rangle_O + \sum_{x'\neq x^*} |x'\rangle_I |0\rangle_O\right) \\
&= \frac{1}{2^n} \langle x^*|_I \langle 1|_O \Pi_1 |x^*\rangle_I |1\rangle_O \\
&= \frac{1}{2^n}(\langle x^*|_I \langle 1|_O)(|x^*\rangle_I |1\rangle_O) \\
&= \frac{1}{2^n} \langle x^*|x^*\rangle \langle 1|1\rangle = \frac{1}{2^n} \ .
\end{aligned}
$$

For the second equality, remember that $\Pi_1 |x\rangle_I |0\rangle_O = 0$ regardless of $x$, so the terms of the product with $|0\rangle_O$ on either side will be annihilated by $\Pi_1$. Therefore, even if $f$ is satisfiable, the probability of detecting 1 on the output register is exponentially small, and thus the number of trials needed in expectation before detecting a 1 with high probability is exponentally large[3]. We can even prove that no measurement can be of any help by showing that $\||\psi_{yes}\rangle - |\psi_{no}\rangle\|^2 = \frac{1}{2^n}$ and invoking Theorem 1.17.

This is why saying that the sentence 'a quantum computer can try all the possible combinations at once' is misleading. While it is true that the algorithm computes $f$ for every possible input at once in some sense, quantum parallelism taken by itself is not conceptually different from a classical randomized algorithm choosing $x \in \{0,1\}^n$ at random and then computing $f(x)$. So certainly we cannot prove $NP \subseteq BQP$ with it alone. This doesn't mean that quantum parallelism is useless, it's the main ingredient of a lot of algorithms we'll explore.

## 3.4 The phase oracle

Now we assume $m = 1$, i.e., the output of $f$ is a single bit. This time, instead of copying the output to an output register, we apply a phase $\{+1, -1\}$ to the input state.

**Definition 3.5.** Given a function $f : \{0,1\}^n \to \{0,1\}$, the *phase oracle* is the unitary $U_f$ satisfying

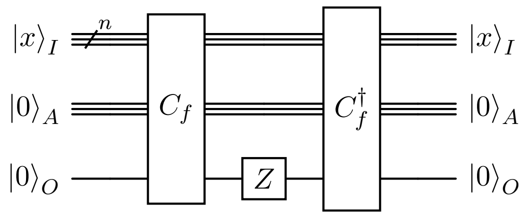
$$U_f |x\rangle = (-1)^{f(x)} |x\rangle$$

[3]If $X$ is the random variable representing the failed attempts before getting a 1, then $X$ is a geometric random variable with parameter $p = 1/2^n$, and thus its expectation is $\mathbb{E}[X] = 1/p = 2^n$.

Notice that the elements of the standard basis $|x\rangle$ are the eigenstates (eigenvectors) of $U_f$ with eigenvalue $-1$ or $+1$ depending on the value of $f(x)$. This means that the matrix representation of $U_f$ is diagonal

$$U_f = \begin{bmatrix} (-1)^{f(0)} & & & \\ & (-1)^{f(1)} & & \\ & & \ddots & \\ & & & (-1)^{f(2^n-1)} \end{bmatrix}$$

where we intend the integers $0, 1, \dots, 2^n - 1$ to be taken as their binary representations. The implementation $U_f$ is very similar to the one of $O_f$ seen in the previous sections.



Instead of the CNOTs used in $O_f$, which copy the output to another register, here we use the Pauli-$Z$ (recall that $Z|0\rangle = |0\rangle, Z|1\rangle = -|1\rangle$, i.e., $Z|b\rangle = (-1)^b|b\rangle$). The evolution of the circuit is as follows:

$$\begin{aligned} |x\rangle_I |0\rangle_A |0\rangle_O &\overset{C_f}{\mapsto} |g_x\rangle_{I,A} |f(x)\rangle_O \\ &\overset{Z}{\mapsto} |g_x\rangle_{I,A} \otimes (-1)^b |f(x)\rangle_O = (-1)^b |g_x\rangle_{I,A} \otimes |f(x)\rangle \\ &\overset{C_f^\dagger}{\mapsto} (-1)^b |x\rangle_I |0\rangle_A |0\rangle_O \end{aligned}$$

where $|g_x\rangle$ is a shorthand for the garbage left by $C_f$ as intermediate computations. The equality relies on the bilinearity of the tensor product, and this trick of moving the phase generated by a local gate to the global state is known in the quantum computing jargon as *phase kickback*.

As for $O_f$, we can hide the registers $I, A$ since their state doesn't change, and the time complexity follows the one of the classical circuit for $f$.

## 3.5 Warmup: the Deutsch-Josza algorithm

We are ready to see a first (toy) application of what we have developed so far. Imagine to have a function $f : \{0,1\} \mapsto \{0,1\}$, (i.e., the function takes and outputs only one bit) given as an oracle[4]. The goal is to say whether $f(0) \neq f(1)$ or, in other words, compute the XOR $f(0) \oplus f(1)$.

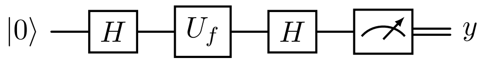
**Claim 3.6.** *Any classical algorithm must query $f$ twice to compute $f(0) \oplus f(1)$ correctly.*

*Proof.* Suppose that a classical algorithm $\mathcal{A}$ does only one query (algorithms doing no queries certainly have no possibility, as their output would be completely independent from $f$), and assume without loss of generality that $\mathcal{A}$ queries $f(0)$ (the other case being symmetric). $\mathcal{A}$ will not learn $f(1)$, and thus its output can only depend on $f(0)$. Consider two different inputs, where $f(1) = 1$ and $f(1) = 0$: in the first case, $\mathcal{A}$ should return $f(0) \oplus 1 = \overline{f(0)}$, while in

[4]From now on, when we say that a function is given as an oracle we mean that the function is given to us as a classical circuit, from which we can construct either a bit or a phase oracle.

the second case, it should return $f(0) \oplus 0 = f(0)$. Since the output cannot depend on $f(1)$, it must be the same for both these cases. We conclude that $\mathcal{A}$ must be wrong in one of these two cases. □

With some more effort, this claim works also for randomized algorithms. On the other hand, we can compute the XOR with one quantum query using *Deutsch's algorithm*:



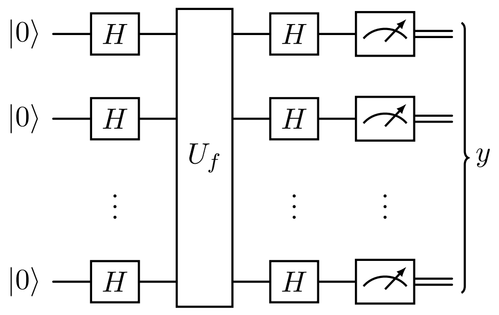
This first algorithm is very simple, as it acts only on one qubit. We claim that the measurement outcome $y$ will always be $f(0) \oplus f(1)$: let's see what happens to the state during the computation

$$\begin{aligned}
|0\rangle &\overset{H}{\mapsto} \frac{|0\rangle + |1\rangle}{\sqrt{2}} \\
&\overset{U_f}{\mapsto} \frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}} \\
&= (-1)^{f(0)} \frac{|0\rangle + (-1)^{f(0)+f(1)}|1\rangle}{\sqrt{2}} && \text{collecting a } (-1)^{f(0)} \text{ term}
\end{aligned}$$

Before applying the second Hadamard gate, notice that depending on whether $f(0) + f(1)$ is even or odd (i.e., depending on the value of the XOR), the state is either $|+\rangle$ or $|-\rangle$, up to the irrelevant global phase we collected. The Hadamard gate will map these two states to $|0\rangle$ or $|1\rangle$, which are then measured and returned as $y$. This is how we prove a so-called *quantum speed-up* or *advantage*: we show that no (randomized or deterministic) classical algorithm can solve a problem efficiently, and then we devise a quantum algorithm that achieves that efficiency. The idea of Deutsch's algorithm can be extended to the following, slightly more interesting problem.

**Problem 3.7** (Constant or balanced)**.** Let $f : \{0,1\}^n \to \{0,1\}$ be a function given as an oracle. Return a bit indicating whether $f$ is constant (all the values are equal) or balanced (half of the values give 0 and half give 1).

This time we start with the quantum algorithm, known as the *Deutsch-Josza* algorithm:



The algorithm is the straight generalization of Deutsch's algorithm to $n$ qubits.

**Claim 3.8.** *If $f$ is constant, the probability of measuring $y = 0$ (i.e., all bits to zero) is 1. If $f$ is balanced, the probability is 0.*

Therefore, by checking if the measurement outcome $y \neq 0$ we can deduce whether $f$ is balanced or not.

*Proof.* If $f$ is constant, then $U_f$ is basically the identity matrix (either $I$ if $f(x) \equiv 0$ or $-I$ if $f(x) \equiv 1$, but the global phase is irrelevant). Therefore, we are applying the Hadamards twice, and they will cancel out. Thus the state right before the measurement is $|00\cdots 0\rangle$, which will yield $y = 0$ with certainty as claimed.

Now suppose that $f$ is balanced: let's compute the measurement probability:

$$\Pr[y=0] = \left|\langle 0| H^{\otimes n} U_f H^{\otimes n} |0\rangle\right|^2$$

Since $H^{\otimes n}|0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle$ is the equal superposition of all the possible strings, we can simplify the expression in the absolute value:

$$\begin{aligned}\langle 0| H^{\otimes n} U_f H^{\otimes n} |0\rangle &= \left(\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} \langle x|\right) U_f \left(\frac{1}{\sqrt{2^n}} \sum_{x'\in\{0,1\}^n} |x'\rangle\right) \\ &= \frac{1}{2^n} \sum_{x\in\{0,1\}^n} \sum_{x'\in\{0,1\}^n} \langle x| U_f |x'\rangle\end{aligned}$$

Since $|x\rangle, |x'\rangle$ are the elements of the standard basis, the expression $\langle x| U_f |x'\rangle$ is the $(x, x')$-entry of the matrix, and remember that $U_f$ is diagonal, so $\langle x| U_f |x'\rangle \neq 0$ only when $x = x'$. Thus the expression above can be simplified to

$$\begin{aligned}\langle 0| H^{\otimes n} U_f H^{\otimes n} |0\rangle &= \frac{1}{2^n} \sum_{x\in\{0,1\}^n} \langle x| U_f |x\rangle \\ &= \frac{1}{2^n} \sum_{x\in\{0,1\}^n} (-1)^{f(x)}\end{aligned}$$

Since $f$ is balanced, half of the elements of this sum are $+1$, and half of them are $-1$. Thus the terms sum up to zero, and so also the probability that $y = 0$. □

**Remark 3.9.** The Deutsch-Josza can thus solve Problem 3.7 with certainty using only one query. Using an argument similar to the one in Claim 3.8, it is possible to show that any deterministic classical algorithm requires to query half of the values plus one (i.e., $2^n - 1 + 1$).

However, if we allow non-zero probability of failure, a randomized algorithm can query one fixed value, say $f(0)$, and then query $k$ independently random values. If $f$ is balanced, each of these values will have a different value than $f(0)$ with probability $1/2$, and so a balanced $f$ is detected with probability $1 - 1/2^k$. So the Deutsch-Josza algorithm does not constitute a quantum advantage in the practical sense.

## 3.6 Exercises

**Exercise 3.1.** Show that $O_f$ can be constructed using $U_f$ and vice versa.

# Part II

# First era: reflections and Fourier transforms

# Chapter 4

# Grover's algorithm and amplitude amplification

In this chapter we see the first practically relevant quantum algorithms used to tackle search problems, known as Grover's algorithm [6, 7]. Its generalization, known as *amplitude amplification* [8], is arguably one of the most important techniques of the first era of quantum algorithmic design, as it turns out to be useful in many other algorithms.

## 4.1 Unstructured search

Consider a function $f : \{0,1\}^n \to \{0,1\}$ that is given to us as an oracle. We say that an element $x$ is *marked* if $f(x) = 1$. The goal is to find and output some marked element (or conclude there is none). This is a very general problem: for example, if $f$ once again represents a SAT formula, then unstructured search is equivalent to SAT.

We first prove that classically we require to query all the $N = 2^n$ elements. We also take advantage of this situation to introduce an important tool to give lower bounds for the query complexity of randomized algorithms.

**Lemma 4.1** (Yao's minimax principle)**.** *Let $\mathcal{A}(x;r)$ be a randomized algorithm. The following holds:*

$$\min_x \Pr_r[\mathcal{A}(x;r) \textit{ succeeds}] \leq \max_r \Pr_{x\sim\mathcal{D}}[\mathcal{A}(x;r) \textit{ succeeds}]$$

*for any probability distribution $\mathcal{D}$ of inputs.*

If we fix the randomness $r$, $\mathcal{A}(\cdot;r)$ can be seen as a deterministic algorithm. Yao's principle then tells us that the worst-case success probability (with respect to the randomness $r$) of a randomized algorithm is lower bounded by the average-case success probability (with respect to the input $x$) of the best possible deterministic algorithm. Notice that this works regardless of the distribution $\mathcal{D}$ of the input, so it is sufficient to choose one that is able to 'fool' any deterministic algorithms.

**Theorem 4.2.** *Any randomized algorithm succeeding with probability $\geq 2/3$ requires $\Omega(N)$ queries to solve unstructured search.*

*Proof.* Take a deterministic algorithm $\mathcal{A}$ which makes $t$ queries, and give it random input $f$ with exactly one marked element $x^*$ chosen uniformly at random. $\mathcal{A}$ can only succeed if it queries the marked element, and this can only happen with probability $\leq t/N$, regardless of the algorithm we choose. By Yao's principle, any randomized algorithm doing $t < \frac{2}{3}N$ queries has success probability $< 2/3$. $\square$

Note that this is not enough to conclude that SAT cannot be solved in polynomial time. We can only say that no polynomial-time algorithm can solve SAT by only looking at the outputs of $f$, but perhaps some valuable information could be retrieved by leveraging its structure.

## 4.2 Grover's search: qubitization and double reflections

We now see how to solve unstructured search using only $\mathcal{O}(\sqrt{N})$ queries to $f$ (along with Theorem 4.2, this proves a *quadratic speed-up*). For this we use the implementation for $f$ to construct its phase oracle $U_f$. We anticipate that Grover's algorithm needs only a $n$-qubit register. Moreover, it doesn't use complex numbers, so all the vectors here will be real.

We denote with $M$ the number of marked elements, and define two states

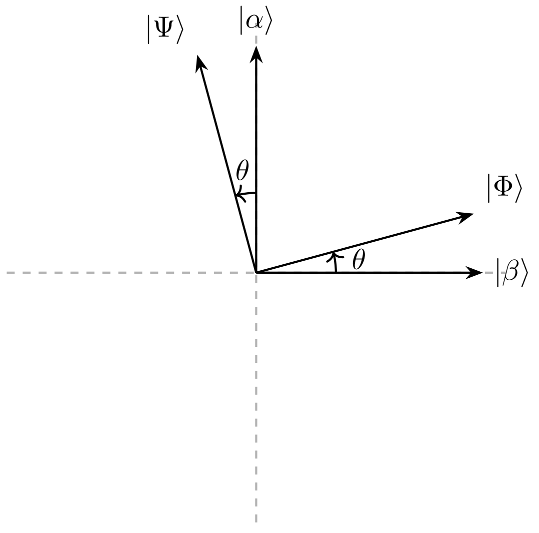
$$|\alpha\rangle := \frac{1}{\sqrt{M}} \sum_{x:f(x)=1} |x\rangle\,, \qquad |\beta\rangle := \frac{1}{\sqrt{N-M}} \sum_{x:f(x)=0} |x\rangle$$

which are the equal superposition of all the marked and unmarked elements, respectively. If we manage to get $|\alpha\rangle$ in our register, upon measurement we will get a random marked element, and we're done. Note that $\langle\alpha|\beta\rangle = 0$, so these two vectors are orthogonal, and identify a two-dimensional subspace $\mathcal{S}$ (i.e., a plane) of the Hilbert space of our register.

We take $|\Phi\rangle = H^{\otimes n}|0\rangle = \frac{1}{\sqrt{N}}\sum_x |x\rangle$ to be the equal superposition of all the elements. We can see that $|\Phi\rangle \in \mathcal{S}$ since

$$\begin{aligned}
|\Phi\rangle &= \frac{1}{\sqrt{N}} \sum_x |x\rangle \\
&= \frac{1}{\sqrt{N}} \sum_{x:f(x)=1} |x\rangle + \frac{1}{\sqrt{N}} \sum_{x:f(x)=0} |x\rangle \\
&= \sqrt{\frac{M}{N}} \left( \frac{1}{\sqrt{M}} \sum_{x:f(x)=1} |x\rangle \right) + \sqrt{\frac{N-M}{N}} \left( \frac{1}{\sqrt{N-M}} \sum_{x:f(x)=0} |x\rangle \right) \\
&= \sqrt{\frac{M}{N}} |\alpha\rangle + \sqrt{\frac{N-M}{N}} |\beta\rangle \ .
\end{aligned}$$

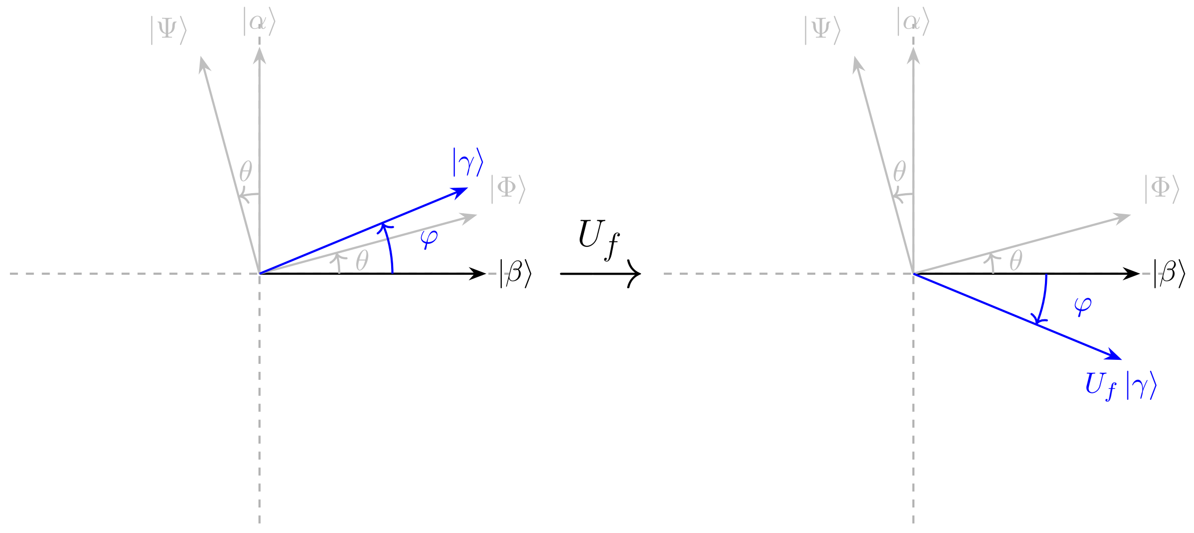
We call $|\Psi\rangle$ the vector in $\mathcal{S}$ that is orthogonal to $|\Phi\rangle$. Finally, we call $\theta$ the angle between $|\Phi\rangle$ and $|\beta\rangle$ (which is equal to the angle between $|\Psi\rangle$ and $|\alpha\rangle$). Since everything is in a plane and real, we can resume all the objects we introduced so far in a geometric picture.



Grover's algorithm starts in the $|\Phi\rangle$ state, which is easy to construct with $n$ Hadamard gates (see Exercise 2.1). The algorithm then proceeds by alternating two different operations: the first one is the phase oracle $U_f$, which acts on $|\alpha\rangle, |\beta\rangle$.

$$U_f |\alpha\rangle = \frac{1}{\sqrt{M}} \sum_{x:f(x)=1} U_f |x\rangle = \frac{1}{\sqrt{M}} \sum_{x:f(x)=1} - |x\rangle = - |\alpha\rangle$$
$$U_f |\beta\rangle = \frac{1}{\sqrt{N-M}} \sum_{x:f(x)=0} U_f |x\rangle = \frac{1}{\sqrt{N-M}} \sum_{x:f(x)=0} |x\rangle = |\beta\rangle$$

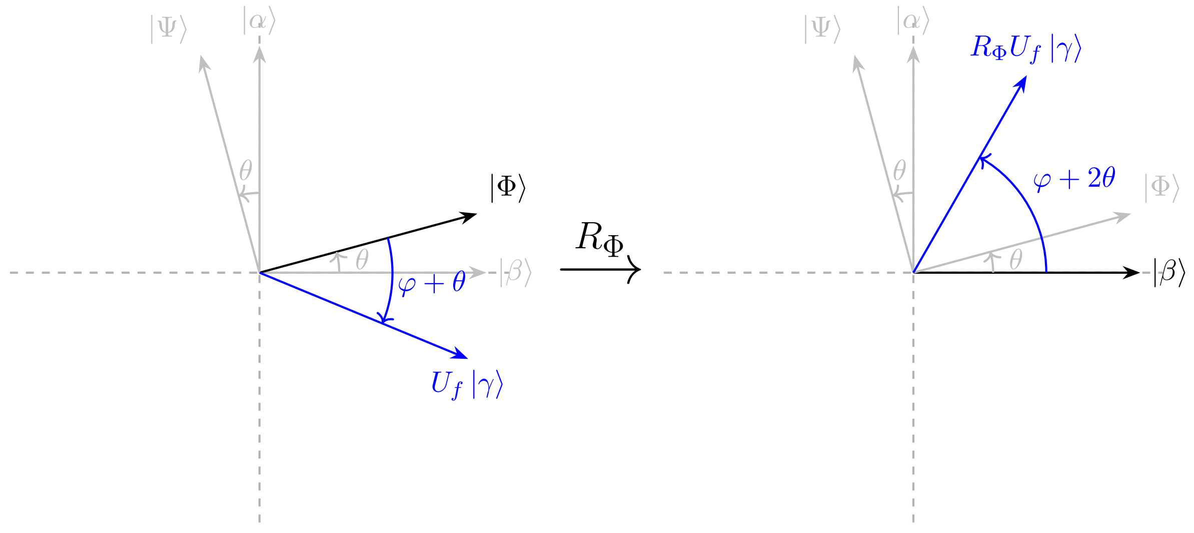
Thus $U_f$ essentially acts like a *reflection* around $|\beta\rangle$ in $\mathcal{S}$.



Mathematically, any linear combination $c_1 |\alpha\rangle + c_2 |\beta\rangle \in \mathcal{S}$ gets mapped to $c_1 |\alpha\rangle - c_2 |\beta\rangle$, which is still in $\mathcal{S}$, so $U_f$ will never make the state leave $\mathcal{S}$. In linear algebra we say that $\mathcal{S}$ is *invariant* with respect to $U_f$.

The second operation is known as the *diffusion operator* $R_\Phi = 2|\Phi\rangle\langle\Phi| - \mathbb{1}$ which geometrically represents a reflection around $|\Phi\rangle$ since

$$R_\Phi |\Phi\rangle = 2|\Phi\rangle\langle\Phi| |\Phi\rangle - |\Phi\rangle = 2 \langle\Phi|\Phi\rangle |\Phi\rangle - |\Phi\rangle = |\Phi\rangle$$
$$R_\Phi |\Psi\rangle = 2|\Phi\rangle\langle\Phi| |\Psi\rangle - |\Psi\rangle = 2 \langle\Phi|\Psi\rangle |\Phi\rangle - |\Psi\rangle = - |\Psi\rangle$$



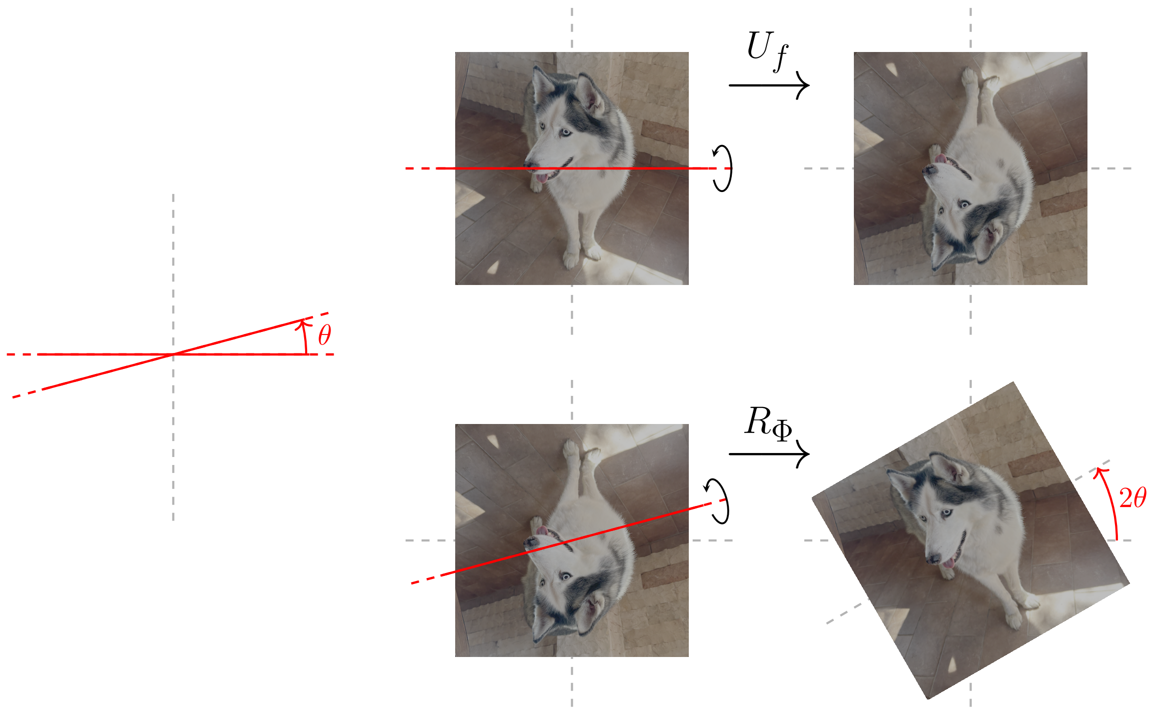


Figure 4.1: The action of $U_f$ and $R_\Phi$ on a picture of my dog Abel. The combined effect $R_\Phi U_f$ of the two reflections is a rotation whose angle is twice the angle between the two reflection axes.

$R_\Phi$ is easy to implement, as it can be decomposed as:

$$R_\Phi = 2H^{\otimes n} |0\rangle\langle 0| H^{\otimes n} - H^{\otimes n} H^{\otimes n} = H^{\otimes n}(2 |0\rangle\langle 0| - \mathbb{1}) H^{\otimes n}$$

and the middle matrix can be implemented with a $n$-way controlled-$Z$ operation $C^n(Z)$ (see Exercise 4.1). Like $U_f$, $R_\Phi$ maps $c_1 |\Phi\rangle + c_2 |\Psi\rangle \in \mathcal{S}$ to $c_1 |\Phi\rangle - c_2 |\Psi\rangle$, which is still in $\mathcal{S}$, so $\mathcal{S}$ is invariant with respect to $\mathcal{S}$ as well. As these two are the only operations we carry out, the state will stay in $\mathcal{S}$ for the whole duration of the algorithm, so it's like the algorithm is working on a single qubit. This idea of reducing the dynamics of the algorithm into a 'virtual qubit' subspace goes by the name of *qubitization* in the modern literature.

Moreover, we've shown that if $|\gamma\rangle$ forms an angle $\varphi$ with $|\beta\rangle$, the state $R_\Phi U_f |\gamma\rangle$ forms an angle $\varphi + 2\theta$ with $|\beta\rangle$. More generally, the combination of the two reflections $U_f, R_\Phi$ yields a counter-clockwise rotation by a $2\theta$ angle, where $\theta$ is the angle between the two reflection axes (Figure 4.1 visually shows this equivalence).

If the Grover iteration $G = R_\Phi U_f$ is a rotation by $2\theta$, then applying $k$ copies of $G$ will make the state rotate by $2k\theta$. Formally, let's define a sequence of states:

$$|\gamma_0\rangle := |\Psi\rangle, \qquad |\gamma_{j+1}\rangle := G |\gamma_j\rangle \text{ for } j > 0$$

The state $|\gamma_k\rangle$ will then form a total $\theta + 2k\theta = (2k+1)\theta$ angle with $|\beta\rangle$, and if we choose $r$ such that $(2r+1)\theta \simeq \frac{\pi}{2}$ then $|\gamma_r\rangle$ will be very close to $|\alpha\rangle$, so a full measurement of $|\gamma_r\rangle$ in the standard basis will yield a marked element with high probability[1]. Let's give a more formal

[1]If $|\gamma_r\rangle = |\alpha\rangle$ then the returned element would be marked with certainty, but we cannot achieve exactly $|\alpha\rangle$ in general, as $k$ must be an integer.

argument: if $r^* = \frac{\pi}{4\theta} - \frac{1}{2} \in \mathbb{R}$ is the real number giving us exactly $\frac{\pi}{2}$, then the two closest integers to $r^*$ are $\lceil r^* \rceil, \lfloor r^* \rfloor$. Let's consider the latter and call it $\hat{r}$:

$$r^* - 1 \leq \lfloor \hat{r} \rfloor \leq r^* \implies \frac{\pi}{2} - \theta \leq (2\hat{r}+1)\theta \leq \frac{\pi}{2} \ .$$

Thus the angle between $|\beta\rangle$ and our state $|\gamma_r\rangle$ is close to $\frac{\pi}{2}$ up to a $\theta$ error or, equivalently, the angle between $|\alpha\rangle$ and $|\gamma_r\rangle$ is at most $\theta$. We can use this to bound the distance between these two states. Note that $|\gamma_r\rangle$ can be represented as a linear combination of $|\alpha\rangle, |\beta\rangle$:

$$|\gamma_r\rangle = \cos\varphi\,|\alpha\rangle + \sin\varphi\,|\beta\rangle$$

where $\varphi = \frac{\pi}{2} - (2\hat{r}+1)\theta \in [0, \theta]$. The distance between the two states can be bounded

$$\begin{aligned}
\||\gamma_r\rangle - |\alpha\rangle\|^2 &= (\cos\varphi - 1)^2 + \sin^2\varphi \\
&= \cos^2\varphi - 2\cos\varphi + 1 + \sin^2\varphi && \text{since } \sin^2\varphi + \cos^2\varphi = 1 \\
&= 2 - 2\cos\varphi \\
&= 4\sin^2\frac{\varphi}{2} && \text{since } 1 - \cos\varphi = \sin^2\frac{\varphi}{2} \\
&\leq 4\sin^2\frac{\theta}{2} \leq 4\sin^2\theta
\end{aligned}$$

By invoking Theorem 1.17, since the probability of measuring a marked element from $|\alpha\rangle$ is 1, then the probability of measuring a marked element from $|\gamma_r\rangle$ is $\geq 1 - 4\sin^2\theta$. We summarize what we proved so far:

**Theorem 4.3.** *By applying Grover's iteration $G = R_\Phi U_f$ for $\hat{r} = \lfloor \frac{\pi}{4\theta} - \frac{1}{2} \rfloor$ times to $|\Phi\rangle$, the probability of measuring a marked element is $\geq 1 - 4\sin^2\theta$.*

The only thing we need to do to conclude the analysis of Grover's algorithm is to determine $\theta$. As we said, $\theta$ is the angle between the initial state $|\Phi\rangle$ and $|\beta\rangle$. Linear algebra gives us the *cosine rule*, stating that the inner product between two normalized states is the cosine of the angle between them.

$$\cos\theta = \langle\beta|\Phi\rangle$$

By the fact that $\langle\alpha|\Phi\rangle^2 + \langle\beta|\Phi\rangle^2 = 1$ (we removed the absolute value since they are real anyway), the above is equivalent to

$$\begin{aligned}
\sin\theta &= \langle\alpha|\Phi\rangle \\
&= \langle\alpha| \left( \sqrt{\frac{M}{N}}\,|\alpha\rangle + \sqrt{\frac{N-M}{N}}\,|\beta\rangle \right) \\
&= \sqrt{\frac{M}{N}}
\end{aligned}$$

from which we conclude $\theta = \arcsin\left(\sqrt{M/N}\right) = \Theta(\sqrt{M/N})$[2]. Therefore, we get the full analysis of Grover's algorithm.

**Corollary 4.4.** *By starting from $|\Phi\rangle$ and applying $G$ for $\hat{r}$ times, where*

$$\hat{r} = \left\lfloor \frac{\pi}{4\theta} - \frac{1}{2} \right\rfloor = \mathcal{O}\left( \sqrt{\frac{M}{N}} \right) ,$$

*measuring the state yields a marked element with probability at least $1 - 4\frac{M}{N}$.*

**Algorithm 4.1** Grover's algorithm

```
Input: A function $f : \{0,1\}^n \to \{0,1\}$ given as an oracle.
Output: An element $x$ such that $f(x) = 1$.
  Let $\hat{r} = \lfloor \frac{\pi}{4\theta} - \frac{1}{2} \rfloor$.        ▷ $\theta = \arcsin\left(\sqrt{M/N}\right)$
  Apply $H^{\otimes n}$.
  for $k = 1$ to $\hat{r}$ do
      Apply $U_f$.
      Apply $H^{\otimes n}$.
      Apply $R_0 = 2|0\rangle\langle 0| - \mathbb{1}$.
      Apply $H^{\otimes n}$.
  end for
  Measure the register and return the outcome.
```

**Remark 4.5.** If $M/N$ is high, then the probability bound given by Corollary 4.4 doesn't really say anything useful. However remember that in this case we don't really need a quantum computer: if you choose an element at random, then it will be marked with probability $\frac{M}{N}$.

**Remark 4.6.** If $M = N$ or $M = 0$ then $|\beta\rangle$ (resp. $|\alpha\rangle$) is not well-defined. In this case the space $\mathcal{S}$ has only one dimension, where $|\Phi\rangle = |\alpha\rangle$ (resp. $|\Phi\rangle = |\beta\rangle$). Moreover, $R_\Phi$ and $U_f$ both act like the identity (up to a $-1$ sign, depending on the case).

## 4.3 Grover's search: the soufflé problem

Notice that $\hat{r}$ chosen by Algorithm 4.1 has to be *exact*: after reaching a $\frac{\pi}{2}$ angle with $|\beta\rangle$ (yielding a maximum success probability), then subsequent applications of $G$ will further increase the angle, making the state rotate away from $|\alpha\rangle$. For example, after roughly $2\hat{r}$ iterations, the total angle would be $\simeq \pi$, and thus the probability of measuring a marked state is essentially zero. This is known as the *soufflé problem*, because too many iterations will 'overcook' the soufflé, which will then collapse. Generally, every odd integer multiple of $r^*$ gives $\pm|\alpha\rangle$, while every even integer multiple yields $\pm|\beta\rangle$.

Because of the this, Algorithm 4.1 has a little caveat: in order to compute $\hat{r}$ we should know the number $M$ of marked elements in advance. This is a problem, especially if we wanted to use the algorithm to understand whether $M \neq 0$ in the first place.

**Claim 4.7** ([9]). *Let $T$ be a number taking uniformly at random in $\{0, \ldots, m-1\}$ for some positive integer $m$. The probability of measuring a marked element from $|\gamma_T\rangle$ is*

$$p_m = \frac{1}{2} - \frac{\sin(4m\theta)}{4m\sin(2\theta)}$$

*In particular, $p_m \geq 1/4$ if $m \geq 1/\sin(2\theta)$.*

*Proof.* The probability of measuring a marked elements from $|\gamma_j\rangle$ for some fixed $j$ is

$$|\langle\alpha|\gamma_j\rangle|^2 = \sin^2((2j+1)\theta) \ .$$

[2] Note that $\arcsin x \sim x$ as $x \to 0$, and $\arcsin x = \Theta(x)$ is a weaker condition.

By the law of total probability, the probability of measuring a marked state from $|\gamma_T\rangle$ is

$$\begin{aligned}
\mathbb{E}_T\left[|\langle\alpha|\gamma_T\rangle|^2\right] &= \frac{1}{m}\sum_{j=0}^{m-1}\sin^2\left((2j+1)\theta\right) \\
&= \frac{1}{2m}\sum_{j=0}^{m-1} 1-\cos\left((2j+1)2\theta\right) \qquad\qquad \text{since } \sin^2\varphi = \frac{1-\cos(2\varphi)}{2} \\
&= \frac{1}{2}-\sum_{j=0}^{m-1}\cos\left((2j+1)2\theta\right) \\
&= \frac{1}{2}-\frac{\sin(4m\theta)}{4m\sin(2\theta)}\ .
\end{aligned}$$

The last equality is a property of the trigonometric functions, which we won't prove here. Note that the claim follows from $\sin(4m\theta)\le 1$ and $m\ge 1/\sin(2\theta)$. ☐

We only need to find a guess $m$ satisfying $1/\sin(2\theta)$. Let $m_0 = 1/\sin(2\theta)$, which we call the *critical stage*. We don't know $m_0$, but we know that once $m \ge m_0$, we will succeed with probability $\ge 1/4$. Moreover, since $\sin^2\theta = \frac{M}{N}$, the critical stage satisfies

$$m_0 = \frac{1}{\sin(2\theta)} = \frac{1}{2\sin\theta\cos\theta} = \frac{N}{2\sqrt{M(N-M)}} \le \sqrt{\frac{N}{M}}$$

where for the last inequality we assumed $M \le \frac{3}{4}N$ (otherwise the problem would be trivial anyway, see Remark 4.5). It is thus sufficient for us to take $m = \lceil\sqrt{N}\rceil \ge m_0$, and we will get a marked state with probability $\ge 1/4$ after $\mathcal{O}(\sqrt{N})$ queries.

**Algorithm 4.2** Grover's algorithm with unknown marked elements

```
Input: A function $f : \{0,1\}^n \to \{0,1\}$ given as an oracle.
Output: An element $x$ such that $f(x) = 1$.
  Choose $T$ uniformly at random in $\{0,\dots,\lceil\sqrt{N}\rceil - 1\}$
  Run Grover's algorithm (Algorithm 4.1) with $T$ iterations.
```

**Remark 4.8.** Note that as $m \to \infty$, the probability lower bound of Theorem 4.7 approaches 1/2. This because in this case $|\gamma_T\rangle$ is essentially a uniformly random real state in $\mathcal{S}$, which has exactly 1/2 overlap with $|\alpha\rangle$ and 1/2 overlap with $|\beta\rangle$, by symmetry.

**Remark 4.9.** It can be proven that Grover's algorithm is optimal, i.e., $\Omega(\sqrt{N})$ quantum queries are needed to solve unstructured search. We delay the proof of this fact to a later chapter.

## 4.4 Amplitude amplification

Grover's search — especially the idea of double reflections — is a precursor of several algorithms from the first era of quantum algorithms research. The very first extension of this algorithm came two years after Grover's result, and is known as *amplitude amplification* [8].

In order to understand its usefulness, we first take a detour to randomized algorithms and talk about its classical counterpart: *probability amplification*. Suppose a randomized algorithm $\mathcal{A}$ outputs a correct solution to some problem with probability $p > 0$ (and when it fails, it declares failure). Can we construct some algorithm $\mathcal{A}'$ that uses $\mathcal{A}$ to output a correct solution with some high probability, e.g., $\frac{99}{100}$? The idea is simple: run $\mathcal{A}$ for up to $K$ independent times, and output a solution as soon as you get one. We can easily bound the probability of failure:

$$\begin{aligned}
\Pr[\mathcal{A}' \text{ fails}] &= \Pr\left[\bigcap_{j=1}^{K} \mathcal{A} \text{ fails at run } k\right] \\
&= \prod_{j=1}^{K} \Pr[\mathcal{A} \text{ fails at run } k] && \text{since runs are independent} \\
&= (1-p)^K \\
&\le e^{-pK} && \text{since } 1-x \le e^{-x}
\end{aligned}$$

In order to make the last term less than $\frac{1}{100}$, we can choose $K \ge \log(100)/p = \Theta(1/p)$. This is also why we are happy enough when we see that some algorithm (classical or quantum) succeeds with constant probability, as the probability can be made arbitrarily close to 1 by repeating the algorithm a constant number of times.

We show that using an idea similar to Grover's algorithm, we can amplify the success probability using only $\mathcal{O}(1/\sqrt{p})$ runs of $\mathcal{A}$. In this setting, our algorithm $\mathcal{A}$ is represented by a unitary. Remember that any classical algorithm can be turned into a quantum circuit, but $\mathcal{A}$ in this case can also be another quantum algorithm.

$$|\Phi\rangle := \mathcal{A}\,|0\rangle = \sqrt{p}\,|\alpha\rangle + \sqrt{1-p}\,|\beta\rangle$$

for some orthogonal states $|\alpha\rangle, |\beta\rangle$[3]. Moreover, we have access to the function $f$, which can be seen as a verifier, returning 1 if and only if the solution is correct. Unlike in Grover's algorithm, here $|\alpha\rangle, |\beta\rangle$ need not be equal superpositions: we're fine as long as $U_f\,|\alpha\rangle = -\,|\alpha\rangle$ and $U_f\,|\beta\rangle = |\beta\rangle$, i.e., $|\alpha\rangle$ contains only correct solutions, and $|\beta\rangle$ contains only wrong solutions.

The diffusion operator this time is

$$R_\Phi = 2\,|\Phi\rangle\langle\Phi| - \mathbb{1} = \mathcal{A}(2\,|0\rangle\langle 0| - \mathbb{1})\mathcal{A}^\dagger$$

The idea of double reflections explained in Section 4.2 can be used also here: the product of the two reflections $G = R_\Phi U_f$ is a rotation of $2\theta$, where $\theta$ this time is:

$$\theta = \arcsin\,\langle\alpha|\Phi\rangle = \arcsin\sqrt{p}$$

**Corollary 4.10.** *By starting from $|\Phi\rangle$ and applying $G$ for $\hat{r}$ times, where*

$$\hat{r} = \left\lfloor \frac{\pi}{4\theta} - \frac{1}{2} \right\rfloor = \mathcal{O}\left(\frac{1}{\sqrt{p}}\right),$$

*measuring the state yields a marked element with probability at least $1-4p$.*

We summarize the amplitude amplification scheme in Algorithm 4.3. Remember that $\mathcal{A}^\dagger$ is easy to impelment if we already have the circuit for $\mathcal{A}$, as we only need to invert all the gates and apply them in reverse order.

**Remark 4.11.** Grover's algorithm is a special case of the amplitude amplification scheme, because we only need to take $\mathcal{A} = H^{\otimes n}$. The $n$-fold Hadamard gate can be seen as the quantization of an algorithm that chooses an element uniformly at random. Amplitude amplification will amplify the success probability $p = M/N$ of this naive algorithm to $1 - o(1)$.

[3]Actually the amplitudes in front of $|\alpha\rangle$ and $|\beta\rangle$ can be complex in general. But we can assume without loss of generality that they are real and positive, by incorporating the phases of these complex numbers into the vectors.

**Algorithm 4.3** Amplitude amplification scheme

```
Input: A function $f : \{0,1\}^n \to \{0,1\}$ given as an oracle.
Output: An element $x$ such that $f(x) = 1$.
  Let $\hat{r} = \lfloor \frac{\pi}{4\theta} - \frac{1}{2} \rfloor$.                    ▷ $\theta = \arcsin(1/\sqrt{p})$
  Apply $H^{\otimes n}$.
  for $k = 1$ to $\hat{r}$ do
      Apply $U_f$.
      Apply $\mathcal{A}^\dagger$.
      Apply $R_0 = 2|0\rangle\langle 0| - \mathbb{1}$.
      Apply $\mathcal{A}$.
  end for
  Measure the register and return the outcome.
```

## 4.5 Oblivious amplitude amplification

We give a further variant of amplitude amplification, which is arguably one of the most useful schemes in many applications. This time we suppose that $\mathcal{A}$ represents a quantum circuit implementing

$$|\Phi\rangle_{AB} := \mathcal{A}|0\rangle_A|0\rangle_B = \sqrt{p}|1\rangle_A|\alpha\rangle_B + \sqrt{1-p}|0\rangle_A|\beta\rangle_B \tag{4.1}$$

i.e., this time $|\alpha\rangle, |\beta\rangle$ *need not be orthogonal*, but $\mathcal{A}$ writes a flag on an ancilla qubit $A$ indicating whether the process is successful. Our final goal is to have the state $|\alpha\rangle$ on the register $B$ (which, for example, will be needed in further computation).

This time we don't have a phase oracle $U_f$ flipping the sign of the desired components because we don't know the structure of $|\alpha\rangle$ (i.e., we are *oblivious* to what we are trying to amplify). However, $\mathcal{A}$ guarantees us that our desired state is always associated to qubit $A$ being in the $|1\rangle$ state, so our final goal would be to measure $A$ and get $|1\rangle$, as in this case the whole state will collapse to $|1\rangle|\alpha\rangle$. Note also that $|1\rangle|\alpha\rangle, |0\rangle|\beta\rangle$ are orthogonal even if $|\alpha\rangle, |\beta\rangle$ are not.

Thus, this time our plane $\mathcal{S}$ is the one spanned by $|1\rangle|\alpha\rangle, |0\rangle|\beta\rangle$. So it only remains to replace $U_f$: essentially we want an operator that flips the sign when it reads a $|1\rangle$ on qubit $A$, and one can check that the following does the trick:

$$R_A = (\mathbb{1}_A - 2|1\rangle\langle 1|_A) \otimes \mathbb{1}_B = Z_A \otimes \mathbb{1}_B$$

i.e., $R_A$ is very easy to implement, we only need to apply a Pauli-$Z$ on $A$! Similarly to what we did in Section 4.2, we can see that $\mathcal{S}$ is invariant with respect to both $R_\Phi, R_A$, and the iteration $G = R_\Phi R_A$ acts as a rotation by $2\theta$, so the rest of the analysis done in the previous section is still valid, giving us a state close to $|1\rangle|\alpha\rangle$ with only $\mathcal{O}(1/\sqrt{p})$ repetitions of $\mathcal{A}$. The crucial difference here is that the full information about the underlying problem is taken care of by $\mathcal{A}$, and we are amplifying its success probability *obliviously*, i.e., without knowing anything about $|\alpha\rangle$ (whereas in the previous section we relied on the fact that it's a superposition of values returning 1 on $f$). Algorithm 4.4 summarizes the procedure for oblivious amplitude amplification.

## 4.6 Exercises

**Exercise 4.1.** Write down a quantum circuit that implements Grover's diffusion operator $R_\Phi$ using $H, X$ and a $n$-way controlled-$Z$.

**Algorithm 4.4** Oblivious amplitude amplification scheme

```
Input: An algorithm $\mathcal{A}$ given as an oracle satisfying (4.1).
Output: A desired state $|\alpha\rangle$ on register $B$.
  Let $\hat{r} = \lfloor \frac{\pi}{4\theta} - \frac{1}{2} \rfloor$.          ▷ $\theta = \arcsin(1/\sqrt{p})$
  Apply $\mathcal{A}$.
  for $k = 1$ to $\hat{r}$ do
      Apply $Z$ on qubit $A$.
      Apply $\mathcal{A}^\dagger$.
      Apply $R_0 = 2|0\rangle\langle 0| - \mathbb{1}$.
      Apply $\mathcal{A}$.
  end for
  Measure qubit $A$.
```

*Note*: remember that you only need to implement it up to a global phase.

**Exercise 4.2.** Modify Algorithm 4.2 to improve the complexity from $\mathcal{O}(\sqrt{N})$ to $\mathcal{O}\left(\sqrt{N/M}\right)$.

*Hint*: instead of setting $m = \lceil \sqrt{N} \rceil$ directly, try to guess $m = 1, 2, 4, \ldots, \lceil \log_2 N \rceil$.

**Exercise 4.3.** Extend the algorithm of Exercise 4.2 to amplitude amplification.

**Exercise 4.4.** Schöning's algorithm is a randomized algorithm for SAT running in $\mathcal{O}(n^2)$ time with success probability $\geq (\frac{3}{4})^n$. Thus by probability amplification, the overall complexity of Schöning's algorithm is $\mathcal{O}(n^2(\frac{4}{3})^n)$.

Describe informally a quantum algorithm that solves SAT in $\mathcal{O}\left(n^2\sqrt{\frac{4}{3}}^n\right)$ time.

**Exercise 4.5.** Suppose that the algorithm $\mathcal{A}$ satisfies (4.1) with $p = \frac{1}{4}$.

(i) Argue that two rounds of amplitude amplification will return $|\alpha\rangle$ with certainty.

(ii) Now suppose that $p > \frac{1}{4}$. With the help of an extra qubit, show how to obtain $|\alpha\rangle$ with certainty, again using only two rounds of amplitude amplification. Assume you are able to apply the rotation gate:

$$R_y(\varphi) = \begin{bmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix}$$

**Exercise 4.6.** We want to construct the equal superposition over the first $N$ elements

$$|\psi_N\rangle := \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} |x\rangle \ .$$

If $N$ is a power of two, then we just need to apply an Hadamard to each of the $\log_2 N$ qubits. Here we are interested in the case of general $N$:

(i) Let $n = \lceil \log_2 N \rceil$, and consider $M = 2^n$ to be the smallest power of two $\geq N$. Given access to as many uniformly random bits as you want, give an algorithm that returns a uniformly random integer in $\{0, \ldots, N-1\}$.

*Hint*: consider $f(x)$ to be the function that returns 1 if $x < N$ and 0 otherwise. How can you use this function?

(ii) Quantize the idea of point (i): let $O_f$ be the bit oracle for $f$. Use copies of $H$, $O_f$, and a single-qubit measurement to construct $|\psi_N\rangle$ with probability $\geq \frac{1}{2}$.

(iii) Use amplitude amplification to achieve $|\psi_N\rangle$ with certainty.

(iv) Now suppose we have a subset $S \subseteq \{0, \ldots, M-1\}$ such that $|S| \geq \frac{M}{2}$, and the implementation of a function $g(x)$ returning 1 is $x \in S$ and 0 otherwise. Show how to construct the equal superposition

$$|\psi_S\rangle := \frac{1}{\sqrt{|S|}} \sum_{x \in S} |x\rangle \ .$$

# Chapter 5

# Phase and amplitude estimation

This chapter concerns with some important tools from the first era of the quantum algorithmic literature, namely the *quantum Fourier transform* and the *phase estimation algorithm*. Besides being important components of Shor's algorithm discussed in the next chapter, they open the way for an important design technique consisting in the estimation and manipulation of the *eigenvalues*.

## 5.1 Eigenvalues and eigenstates

The notion of eigenvalue is a central concept of linear algebra.

**Definition 5.1.** Let $A$ be a square matrix of any dimension. A complex number $\lambda$ is said to be an *eigenvalue* of $A$ if $A\left|\psi\right\rangle = \lambda\left|\psi\right\rangle$, for some vector $\left|\psi\right\rangle \neq 0$ which is said to be an *eigenvector* associated to $\lambda$.

Notice that if $\left|\psi\right\rangle, \left|\phi\right\rangle$ are eigenvectors associated to the same $\lambda$, then for any $\alpha, \beta \in \mathbb{C}$, $\alpha\left|\psi\right\rangle + \beta\left|\phi\right\rangle$ is an eigenvector associated to $\lambda$ as well. In other words, the eigenvectors associated to a particular eigenvalue form a subspace, which we call *eigenspace*. In quantum theory we will also used the term *eigenstate* to indicate an eigenvector that is also a quantum state (i.e., a normalized eigenvector).

The theory of eigenvalues, which also goes by the name of *spectral theory*, is ubiquitous in all parts of science and engineering, and quantum theory is certainly not an exception: in previous chapters we have seen two types of matrices which are central in quantum computation: unitary matrices ($U^{-1} = U^\dagger$) and Hermitian matrices ($H^\dagger = H$).

We show the properties of the eigenvalues and eigenvectors for these two types of matrices.

**Lemma 5.2.** *Let $U$ be unitary. If $U\left|\psi\right\rangle = \lambda\left|\psi\right\rangle$ with $\left|\psi\right\rangle \neq 0$, then $|\lambda| = 1$.*

*Proof.* By Definition 1.18 of unitary matrix, we have $\|U\left|\psi\right\rangle\| = \|\left|\psi\right\rangle\|$. The left-hand side is equal to $\|\lambda\left|\psi\right\rangle\| = |\lambda|\|\left|\psi\right\rangle\|$. As the norm is non-zero (since $\left|\psi\right\rangle \neq 0$), we get $|\lambda| = 1$. □

More intuitively, if we know that unitaries must preserve the length of the vectors, and the eigenvalue is essentially the stretch that the transformation applies to the associated eigenvector, in order for this eigenvector to be left unchanged, the stretch must be 1 in absolute value. Notice that $|\lambda| = 1$ can be written as $\lambda^*\lambda = 1$, which resembles the condition $U^\dagger U = \mathbb{1}$ on the matrix.

**Lemma 5.3.** *Let $H$ be Hermitian. If $H\left|\psi\right\rangle = \lambda\left|\psi\right\rangle$ with $\left|\psi\right\rangle \neq 0$, then $\lambda \in \mathbb{R}$.*

*Proof.* Since $H = H^\dagger$ we can write

$$\lambda \langle\psi|\psi\rangle = \langle\psi| H |\psi\rangle = \langle\psi| H^\dagger |\psi\rangle = (H|\psi\rangle)^\dagger |\psi\rangle = \lambda^* \langle\psi|\psi\rangle$$

The inner product is non-zero (as $|\psi\rangle \neq 0$), so we conclude that $\lambda = \lambda^*$. A complex number being equal to its complex conjugate is equivalent to say that this number is real. □

Also here, notice that the condition $\lambda^* = \lambda$ on the eigenvalues resembles the condition $H^\dagger = H$ on the matrix. Roughly speaking, unitary matrices and Hermitian matrices are the matrix counterparts of unit complex numbers and real numbers, respectively.

When it comes to eigenvectors, we have one of the most important results in linear algebra (and arguably, in mathematics).

**Theorem 5.4** (Spectral theorem)**.** *If a matrix $A$ is a $N \times N$ unitary or Hermitian matrix, then there exists a unitary $V$ that diagonalizes $A$, i.e.,*

$$A = V \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_N \end{bmatrix} V^\dagger \tag{5.1}$$

*where $\lambda_1, \ldots, \lambda_N$ are the $N$ eigenvalues of $A$.*

A fancy way to state the spectral theorem is that any unitary or Hermitian matrix $A$ is *unitarily diagonalizable*[1]. This result gives something very strong: there always exists an orthonormal basis of eigenvectors of $A$ (the columns of $U$, which we call *eigenbasis*), so $A$ divides the space in *orthogonal eigenspaces*. The decomposition in (5.1) is also called *eigendecomposition* or *spectral decomposition* of $A$.

**Example 5.5.** Consider the matrix

$$A = \begin{bmatrix} 2 & 1-i \\ 1+i & 3 \end{bmatrix}$$

We have that $A = A^\dagger$ so $A$ is Hermitian. The eigenvalues of $A$ are $4, 1$. A spectral decomposition is $A = V\Lambda V^\dagger$ where

$$V = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & -(1-i) \\ 1+i & 1 \end{bmatrix}, \qquad \Lambda = \begin{bmatrix} 4 & \\ & 1 \end{bmatrix}.$$

Indeed, one can check that the first column of $V$ is an eigenstate associated to $\lambda = 4$, and the second column is associated to $\lambda = 1$.

Thanks to the spectral theorem, we know that unitaries have a very rigid structure, which we will exploit frequently. In many cases — including Shor's algorithm —, if we construct some operation $U$ using a quantum circuit, it is possible that its eigenvalues carry very useful information. Of course, computing the eigenvalues of a $2^n \times 2^n$ matrix is out of question for a classical computer.

[1]Unitary and Hermitian matrices are part of a bigger class, called *normal matrices*, which are the matrices satisfying $AA^\dagger = A^\dagger A$. The spectral theorem actually says that a matrix is unitarily diagonalizable if and only if it is normal.

As a last remark, we talk about functions of matrices: first, for a unitary or Hermitian matrix $A$, we take its spectral decomposition $A = V\Lambda V^\dagger$. The powers of $A$ are simply

$$A^k = \underbrace{V\Lambda V^\dagger V\Lambda V^\dagger \cdots V\Lambda V^\dagger}_{k} = V\Lambda^k V^\dagger$$

In other words, by the fact that $V^\dagger V = \mathbb{1}$, $A^k$ can be computed by keeping the eigenvectors fixed, and raising the eigenvalues to the $k$-th power. As a consequence, consider the exponential function $e^x$, which can be expressed as the Taylor series:

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

The matrix exponential $e^A$ is easily defined using this series:

$$e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!} = \sum_{k=0}^{\infty} V\frac{\Lambda^k}{k!}V^\dagger = V\left(\sum_{k=0}^{\infty} \frac{\Lambda^k}{k!}\right)V^\dagger =: Ve^\Lambda V^\dagger$$

By the shape of $\Lambda^k$, $e^\Lambda$ is easily seen as exponentiating the elements of the diagonal.

$$\sum_{k=0}^{\infty} \frac{1}{k!} \begin{bmatrix} \lambda_1^k & & & \\ & \lambda_2^k & & \\ & & \ddots & \\ & & & \lambda_N^k \end{bmatrix} = \begin{bmatrix} \sum_{k=0}^{\infty} \frac{\lambda_1^k}{k!} & & & \\ & \sum_{k=0}^{\infty} \frac{\lambda_2^k}{k!} & & \\ & & \ddots & \\ & & & \sum_{k=0}^{\infty} \frac{\lambda_N^k}{k!} \end{bmatrix} = \begin{bmatrix} e^{\lambda_1} & & & \\ & e^{\lambda_2} & & \\ & & \ddots & \\ & & & e^{\lambda_N} \end{bmatrix}$$

**Example 5.6.** Considering the eigendecomposition $A = V\Lambda V^\dagger$ from Example 5.5, we exponentiate the eigenvalues

$$e^\Lambda = \begin{bmatrix} e^4 & \\ & e^1 \end{bmatrix}$$

Therefore $e^A$ can be computed as

$$e^A = Ve^\Lambda V^\dagger = \frac{1}{3}\begin{bmatrix} e^4 + 2e & (e^4 - e)(1-i) \\ (e^4 - e)(1+i) & 2e^4 + e \end{bmatrix}$$

Notice that $e^A$ is also Hermitian, as $\lambda \mapsto e^\lambda$ kept the eigenvalues real.

If instead of only doing the matrix exponential, we also first multiply the Hermitian matrix by $i$, the result will be a unitary matrix.

**Lemma 5.7.** *If $H$ is Hermitian, then $U = e^{iH}$ is unitary.*

In this case, the unitary $U$ is said to be generated by $H$.

*Proof.* Decompose $H = V\Lambda V^\dagger$. Then $e^{iH} = Ve^{i\Lambda}V^\dagger$. If $\lambda_k \in \mathbb{R}$ is in the diagonal of $\Lambda$, then $e^{i\lambda_k}$ is in the diagonal of $e^{i\Lambda}$ as we have seen. Notice that $(e^{i\Lambda})^\dagger$ simply conjugates the entries of the diagonal, which implies that

$$U^\dagger U = (e^{i\Lambda})^\dagger e^{i\Lambda} = e^{-i\Lambda}e^{i\Lambda} = e^{i\cdot 0} = I$$

implying that $e^{i\Lambda}$ is unitary, and so is $e^{iH}$ since it's a product of unitaries. □

**Example 5.8.** Take the following Hermitian matrix:

$$A = \begin{bmatrix} 0 & \frac{\pi}{4} \\ \frac{\pi}{4} & 0 \end{bmatrix}$$

A possible eigendecomposition is:

$$A = \left(\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\right)\begin{bmatrix} \frac{\pi}{4} & \\ & -\frac{\pi}{4} \end{bmatrix}\left(\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\right)$$

and so $e^{iA}$ can be computed as

$$e^{iA} = \left(\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\right)\begin{bmatrix} e^{i\frac{\pi}{4}} & \\ & e^{-i\frac{\pi}{4}} \end{bmatrix}\left(\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\right) = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & i \\ i & 1 \end{bmatrix}.$$

We can easily check that $(e^{iA})^\dagger e^{iA} = I$.

**Remark 5.9.** Notice that in Example 5.8, $A = \frac{\pi}{4}X$. Its exponential $e^{i\frac{\pi}{4}X}$ is an example of *Pauli rotation*. The three families of matrices $e^{i\phi X}, e^{i\phi Y}, e^{i\phi Z}$ are important as they represent physical control pulses we can send to our qubits. The actual methodology to produce and apply these pulses depends on the implementation of the qubits.

## 5.2 The Fourier transform

The Fourier transform is a powerful tool in signal processing. In this case we will need the Fourier transform over $\mathbb{Z}_N$.

**Definition 5.10.** We use $\mathbb{Z}_N$ to denote the ring of integers modulo $N$, i.e., the set $\{0, \dots, N-1\}$ equipped with addition $+$ and multiplication $\cdot$ intended to be modulo $N$.

**Example 5.11.** In $\mathbb{Z}_5$ we have that $3 + 4 = 2, 4 + 1 = 0$ and $2 \cdot 3 = 1$.

In the following, we use $f[t]$ to define a *discrete signal* over $\mathbb{Z}_N$, i.e., a function $f : \mathbb{Z}_N \to \mathbb{C}$.

**Definition 5.12.** Given a signal $f[t]$, its *Fourier transform over* $\mathbb{Z}_N$ is defined as the signal:

$$\hat{f}[s] = \frac{1}{\sqrt{N}}\sum_{t=0}^{N-1} \omega_N^{st} f[t]$$

where $\omega_N := e^{2\pi i/N}$ is the $N$-th root of unity, i.e., a number satisfying $\omega_N^N = 1$.

To simplify the notation, we will write just $\omega$ instead of $\omega_N$ when $N$ is clear from the context. Notice that the exponents given to $\omega$ act exactly like the elements of $\mathbb{Z}_N$ (see Figure 5.1).

**Lemma 5.13** (Shift property of the Fourier transform)**.** *Let $g[t] = f[t - \tau]$ be a version of $f$ shifted by $\tau \in \mathbb{Z}_N$ (the subtraction is in $\mathbb{Z}_N$, see Figure 5.2). The following holds:*

$$\hat{g}[s] = \hat{f}[s] \cdot \omega^{s\tau}$$

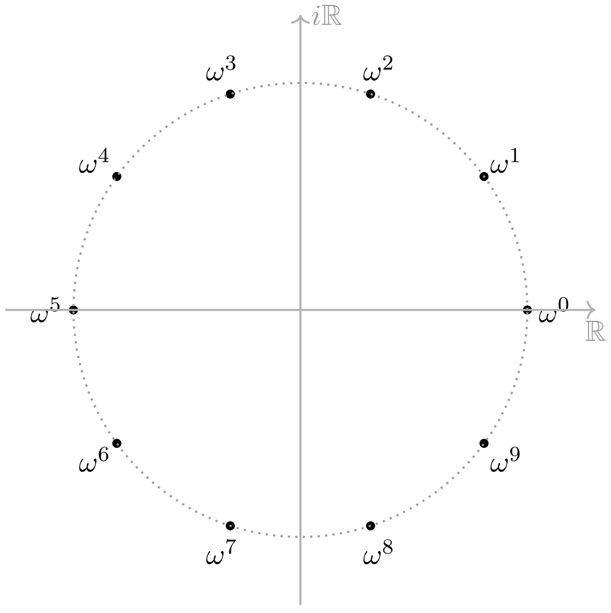


Figure 5.1: Picture of the 10th roots of unity $\omega_{10}^k$ in the complex plane. Multiplying by $\omega$ or, equivalently, adding one to the exponent makes the point rotate counter-clockwise by one position. As effect, the exponents of $\omega$ act like they are in $\mathbb{Z}_{10}$.

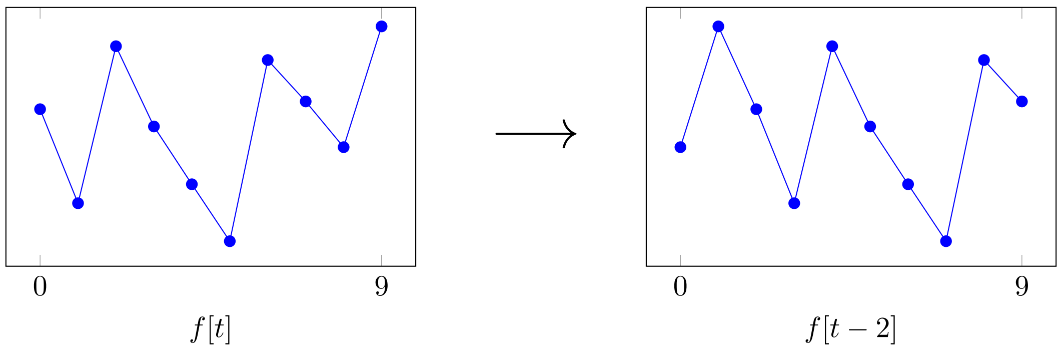


Figure 5.2: Picture of a time-shift $f[t] \to f[t-2]$ on a signal over $\mathbb{Z}_{10}$. The Fourier transform of the function on the right can be obtained by multiplying the Fourier transform $\hat{f}[s]$ of the left function by $\omega^{2s}$. Since the transformation $t \to t-2$ is done in $\mathbb{Z}_N$, the shift is cyclic.

*Proof.* We apply the definition of Fourier transform:

$$\begin{aligned}
\hat{g}[s] &= \frac{1}{\sqrt{N}} \sum_{t=0}^{N-1} \omega^{st} f[t-\tau] \\
&= \frac{1}{\sqrt{N}} \sum_{k=-\tau}^{N-1-\tau} \omega^{s(k+\tau)} f[k] && \text{replacing } k = t-\tau \\
&= \omega^{s\tau} \cdot \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega^{sk} f[k] \\
&= \omega^{s\tau} \cdot \hat{f}[s] \ .
\end{aligned}$$

The second equality simply reshifts the bounds for $k$, since $k$ is an exponent of $\omega$ and as such it can be seen as an element of $\mathbb{Z}_N$. □

Notice that the Fourier transform is a linear transformation: if $h = \alpha f + \beta g$ for $\alpha, \beta \in \mathbb{C}$, then $\hat{h} = \alpha \hat{f} + \beta \hat{g}$. We are now ready to go back to quantum theory: we will consider only normalized signals, i.e., $f[t]$ that satisfies

$$\sum_{t=0}^{N-1} |f[t]|^2 = 1$$

so that we can encode them in a quantum state

$$|f\rangle := \sum_{t=0}^{N-1} f[t] \, |t\rangle \ .$$

Notice that we only need $\lceil \log_2 N \rceil$ qubits to encode a signal over $\mathbb{Z}_N$, while a classical computer would need to store an entire array of $N$ complex numbers. Since the Fourier transform is linear, it can be represented as a matrix $|\hat{f}\rangle = F_N |f\rangle$ where

$$F_N = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(N-1)} \\ \vdots & \vdots & \ddots & \vdots & \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix} \tag{5.2}$$

**Theorem 5.14** (Plancherel's theorem)**.** *$F_N$ is unitary.*

*Proof.* We want to claim that $F_N^\dagger F_N = I$. Notice that, for a matrix $A$, $\langle i| A |j\rangle$ is actually the $(i,j)$-entry of the matrix. Let's compute $\langle i| F_N^\dagger F_N |j\rangle$:

$$\begin{aligned}
\langle i| F_N^\dagger F_N |j\rangle &= (F_N |i\rangle)^\dagger F_N |j\rangle \\
&= \frac{1}{N} \left( \sum_{s'=0}^{N-1} \omega^{s'j} |s'\rangle \right)^\dagger \left( \sum_{s=0}^{N-1} \omega^{si} |s\rangle \right) \\
&= \frac{1}{N} \left( \sum_{s'=0}^{N-1} \omega^{-s'j} \langle s'| \right) \left( \sum_{s=0}^{N-1} \omega^{si} |s\rangle \right) && \text{since } (\omega^k)^* = \omega^{-k} \\
&= \frac{1}{N} \left( \sum_{s'=0}^{N-1} \sum_{s=0}^{N-1} \omega^{si-s'j} \langle s'|s\rangle \right)
\end{aligned}$$

The only terms that survive in this double sum are the ones where $s = s'$ since the vectors $\{|s\rangle\}_{s\in\mathbb{Z}_N}$ are orthonormal. Therefore we can simplify the sum above as

$$\langle i| F_N^\dagger F_N |j\rangle = \frac{1}{N}\sum_{s=0}^{N-1} \omega^{s(i-j)} \tag{5.3}$$

Now we consider two distinct cases: if $i = j$ or $i \neq j$, i.e., we are considering an entry on or off the diagonal of this matrix. The first case is easy, as all the terms in the sum are $\omega^0 = 1$. In the second case we can use the formula for the geometric sum for $q = \omega^{(i-j)}$:

$$\sum_{s=0}^{N-1} q^s = \frac{q^N - 1}{q-1} = 0$$

The numerator is zero since $q^N = \omega^{N(i-j)} = (\omega^N)^{(i-j)} = 1$. On the other hand, the denominator is not, as $q \neq 1$. For a more visual interpretation, the expression in (5.3) can be seen as an average (a.k.a. center of mass) between the points $\omega^{s(i-j)}$ which are equally distributed around the unit circle as in Figure 5.1. Thus the center of all these points must be 0.

We thus proved that diagonal entries are 1 and off-diagonal entries are zero, implying that $F_N^\dagger F_N$ is the identity matrix. □

## 5.3 Quantum Fourier transform in $\mathcal{O}(n^2)$

Plancherel's theorem ensures that the Fourier transform is not only linear but also unitary, and thus we could in principle implement it on a quantum computer. We thus show how to construct a quantum circuit implementing $F_N$, assuming for now that $N = 2^n$ is a power of two. We analyze what happens to the elements of the computational basis $|x\rangle$, as the other vectors will follow by linearity.

$$F_N |x\rangle = \frac{1}{\sqrt{N}}\sum_{y=0}^{N-1} \omega^{xy} |y\rangle \tag{5.4}$$

We will consider the $n$-bit expansions $x = x_{n-1}x_{n-2}\cdots x_0, y = y_{n-1}y_{n-2}\cdots y_0$.

**Theorem 5.15.** *The state $F_N |x\rangle$ in (5.4) can be decomposed in the following way:*

$$F_N |x\rangle = F_{N/2} |x'\rangle_{n-1} \otimes \frac{|0\rangle + \omega^x |1\rangle}{\sqrt{2}} \tag{5.5}$$

*where $|x'\rangle_{n-1} = |x_{n-2}\cdots x_0\rangle$ is obtained by removing the most significant qubit from $|x\rangle$.*

*Proof.* We split the sum in (5.4) into even $y = 2k$ and odd $y = 2k+1$:

$$\frac{1}{\sqrt{N}}\sum_{y=0}^{N-1} \omega^{xy} |y\rangle = \frac{1}{\sqrt{N}}\sum_{k=0}^{\frac{N}{2}-1} \omega^{x(2k)} |2k\rangle + \frac{1}{\sqrt{N}}\sum_{k=0}^{\frac{N}{2}-1} \omega^{x(2k+1)} |2k+1\rangle$$

In the first sum the least significant qubit $y_0 = 0$ always, while $y_n = 0$ in the second sum. Thus we can rewrite $|2k\rangle = |k\rangle_{n-1} |0\rangle, |2k+1\rangle = |k\rangle_{n-1} |1\rangle$, and take out this qubit out of the two sums.

$$\frac{1}{\sqrt{N}}\sum_{y=0}^{N-1} \omega^{xy} |y\rangle = \frac{1}{\sqrt{N}}\left(\sum_{k=0}^{\frac{N}{2}-1} \omega^{x(2k)} |k\rangle_{n-1}\right) \otimes |0\rangle + \frac{1}{\sqrt{N}}\left(\sum_{k=0}^{\frac{N}{2}-1} \omega^{x(2k+1)} |k\rangle_{n-1}\right) \otimes |1\rangle \ .$$

The two sums look almost the same, except for that $+1$, which we can take out of the sum as factor $\omega^x$. Now notice that $\omega_N^2 = \omega_{N/2}$, and that we can split the normalization factor $\sqrt{N} = \sqrt{2} \cdot \sqrt{N/2}$. Therefore we conclude that the state is equal to

$$\frac{1}{\sqrt{2}} \left( \frac{1}{\sqrt{N/2}} \sum_{k=0}^{\frac{N}{2}-1} \omega_{N/2}^{xk} |k\rangle_{n-1} \right) \otimes |0\rangle + \frac{\omega^x}{\sqrt{2}} \left( \frac{1}{\sqrt{N/2}} \sum_{k=0}^{\frac{N}{2}-1} \omega_{N/2}^{xk} |k\rangle_{n-1} \right) \otimes |1\rangle \ .$$

We only need to conclude that the two expressions in the tuples are $F_{N/2} |x'\rangle_{n-1}$, and for that it remains to show that $\omega_{N/2}^{xk} = \omega_{N/2}^{x'k}$: if the most significant digit is $x_{n-1} = 0$, then $x = x'$ and we are done. On the other hand, if $x_{n-1} = 1$, then is means that we have to add $2^{n-1} = N/2$ to $x'$ to make $x$, but in that case we would have

$$\omega_{N/2}^{kx} = \omega_{N/2}^{k(x'+N/2)} = \omega_{N/2}^{kx'} \cdot \omega_{N/2}^{kN/2} = \omega_{N/2}^{kx'} \ .$$

Therefore, the most significant bit $x_n$ does not influence the phases, and the above expression can be written as

$$\frac{1}{\sqrt{2}} F_{N/2} |x'\rangle_{n-1} \otimes |0\rangle + \frac{\omega^x}{\sqrt{2}} F_{N/2} |x'\rangle_{n-1} \otimes |1\rangle = F_{N/2} |x'\rangle_{n-1} \otimes \frac{|0\rangle + \omega^x |1\rangle}{\sqrt{2}} \tag{5.6}$$

as claimed. □

**Remark 5.16.** The fact that the Fourier transform is equal to the left-hand expression of (5.6) is also the heart of the classical divide and conquer algorithm to compute the Fourier transform by Cooley and Tukey [10].

The expression in (5.5) gives us a nice recursive formula to implement our algorithm. The base case is when we have $n = 1$ qubit, and in this case the quantum Fourier transform $F_2$ is simply the Hadamard gate $H$.

**Algorithm 5.1** Radix-2 Quantum Fourier transform

```
Input: $n \geq 1, N = 2^n$, an input state $|x\rangle$ in register $A$.
Output: The Fourier transform $|\hat{x}\rangle = F_N |x\rangle$ in register $A$.
  if $n = 1$ then
      Apply $H$ to the qubit and return.
  end if
  Transform the first (most significant) qubit
      $|x_{n-1}\rangle \to |\hat{x}_{n-1}\rangle = \frac{|0\rangle + \omega^x |1\rangle}{\sqrt{2}}$ .      (5.7)
  Move the first qubit to the last position
      $|\hat{x}_{n-1}\rangle |x'\rangle_{n-1} \to |x'\rangle_{n-1} |\hat{x}_{n-1}\rangle$ .
  Apply the quantum Fourier transform $F_{N/2}$ to the first $n - 1$ qubits.
```

Note that we apply (5.7) on the first (most significant) qubit, unlike what Theorem 5.15 seems to prescribe. This is necessary because this step we won't have easy access to $x_{n-1}$ anymore. This is not a problem since $x_{n-1}$ doesn't appear anywhere else than the $\omega^x$ phase we already prepared, but it would have been a problem for the other bits, which are still needed

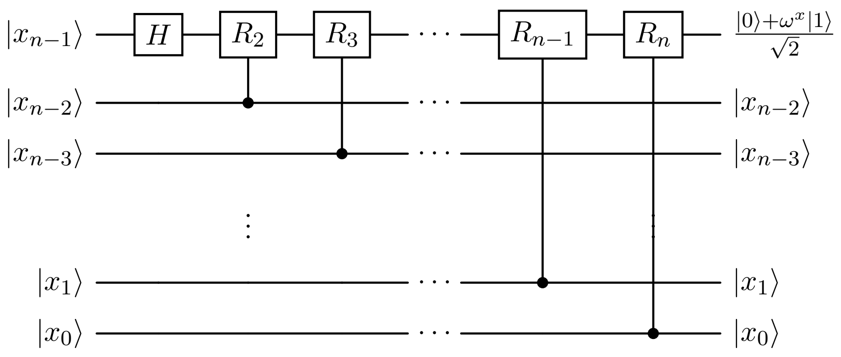


Figure 5.3: Phase transduction circuit for the quantum Fourier transform algorithm

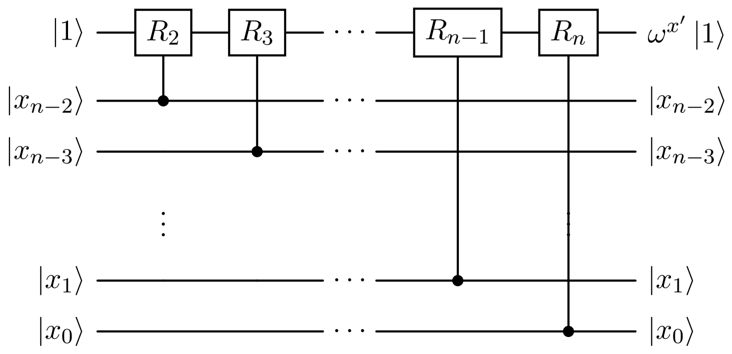
for the recursive call. We conclude by showing how to actually implement the transformation in (5.7). We define a single-qubit phase rotation gate $R_k$

$$R_k = \begin{bmatrix} 1 & \\ & \omega_{2^k} \end{bmatrix}$$

i.e., $R_k$ adds a phase $\omega_{2^k} = e^{2\pi i/2^k}$ to $|1\rangle$, and leaves $|0\rangle$ unchanged. The idea is simple: we need to place a phase $\omega^x$ in front of $|1\rangle$. We can decompose this phase as

$$\omega^x = \omega^{2^{n-1}x_{n-1}} \cdot \omega^{2^{n-2}x_{n-2}} \cdots \omega^{2x_1} \cdot \omega^{x_0} \tag{5.8}$$

where we used the binary expansion $x = 2^{n-1}x_{n-1} + \cdots + 2x_1 + x_0$. Since $\omega_{2^n}^{2^{n-k}} = \omega_{2^k}$, so we can use the rotation gates.



where again $x' = x_{n-2} \cdots x_0$. If $x_k = 1$, then a phase $\omega_{2^{n-k}} = \omega^{2^k}$ is added, otherwise nothing changes. Equivalently, we can say that a phase $\omega^{x_k 2^k}$ is added, so each of these controlled gates adds one of the factors in (5.8). All except the most significant one $\omega^{2^{n-1}x_{n-1}} = (-1)^{x_{n-1}}$, and we do not start from $|1\rangle$ in the above circuit, but on $|x_{n-1}\rangle$. Therefore we need the transformation

$$|x_{n-1}\rangle \mapsto \frac{|0\rangle + (-1)^{x_{n-1}}|1\rangle}{\sqrt{2}}$$

and this is what the Hadamard gate $H$ does, see Figure 5.3 for the final circuit.

The trick of using phase rotations with progressive exponents to make the string saved in the qubits appear as a phase goes by the name of *phase transduction*. Figure 5.4 depicts the recursive circuit for $F_{64}$. The phase transduction requires $n$ gates, and the first qubit can be

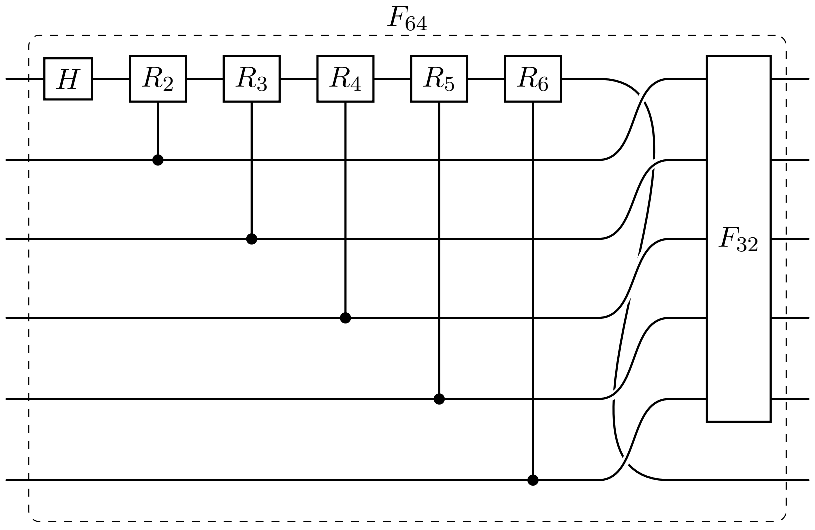


Figure 5.4: Recursive circuit for $F_{64}$, the quantum Fourier transform over $\mathbb{Z}_{64}$.

moved to the last position using $n-1$ SWAP gates. If $T(n)$ is the total number of gates for $F_{2^n}$, then

$$T(n) = T(n-1) + 2n - 1$$

with $T(1) = 1$, which implies $T(n) = n^2$.

**Remark 5.17.** The $n-1$ SWAP gates are actually not required before the recursive call to $F_{N/2}$. We can keep the first qubit there and do all the moves at the end. After all the recursive calls, the qubits will appear in reverse order, so the whole register can be reversed using $\lfloor n/2 \rfloor$ SWAP gates. The asymptotic complexity is still $\mathcal{O}(n^2)$, though.

## 5.4 Approximate quantum Fourier transform in $\mathcal{O}(n \log n)$

We said that $R_k$ applies a phase $e^{2\pi i/2^k}$, but this number is *very* close to 1:

$$\begin{aligned}
\left|e^{2\pi i/2^k} - 1\right| &= \left|e^{\pi i/2^k}\right| \cdot \left|e^{\pi i/2^k} - e^{-\pi i/2^k}\right| \\
&= \left|e^{\pi i/2^k} - e^{-\pi i/2^k}\right| && \text{since } \left|e^{\pi i/2^k}\right| = 1 \\
&= 2 \cdot \left|\sin\left(\frac{\pi}{2^k}\right)\right| && \text{since } e^{ix} - e^{-ix} = 2i \sin x \\
&\leq \frac{\pi}{2^{k-1}} \,. && \text{since } |\sin x| \leq |x|
\end{aligned}$$

Therefore its controlled version $C(R_k)$ is *very* close to the two-qubit identity $I_4$ in operator norm:

$$\|C(R_k) - I_4\| \leq \left\| \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & e^{2\pi i/2^k} \end{bmatrix} - \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \right\| \leq \left|e^{2\pi i/2^k} - 1\right| \leq \frac{\pi}{2^{k-1}}$$

and remember from Section 1.4 that when two unitaries are close in operator norm, then by replacing one with the other, the produced states will be $\epsilon$-close, and the induced probability distributions will be $\epsilon^2$-indistinguishable (Theorem 1.17). So for very high values of $k$, why should we bother implementing $C(R_k)$? Let's take the circuit in Figure 5.3 and we denote with $C$ the resulting unitary: for some constant $c \geq 2$, we remove all the $C(R_k)$ with $k \geq K = \lfloor c \log_2 n \rfloor$. Note that the new circuit (call it $C'$) has $\mathcal{O}(\log n)$ gates instead of $\mathcal{O}(n)$, and the total error is bounded by the sum of the errors introduced by removing each gate (see Exercise 2.4)

$$\begin{aligned}
\|C - C'\| &\leq \sum_{k=K}^{n} \|C(R_k) - I_4\| \\
&\leq \sum_{k=K}^{n} \frac{\pi}{2^{k-1}} \\
&\leq \sum_{k=K}^{\infty} \frac{\pi}{2^{k-1}} \\
&\leq \frac{\pi}{2^{K-1}} \sum_{k=0}^{\infty} \frac{1}{2^k} \\
&\leq \frac{\pi}{2^{K-1}} \sum_{k=0}^{\infty} \frac{1}{2^k} \\
&\leq \frac{4\pi}{2^K} \leq \frac{4\pi}{n^c} && \text{since } \sum_{k=0}^{\infty} \frac{1}{2^k} = 2
\end{aligned}$$

Thus we get an error $\mathcal{O}(1/n^c)$ by discarding the rotations after the $K$-th one. Notice that we have $n-1$ copies of $C$ (of decreasing length) throughout the recursive algorithm, so the total error would become $n$ times the one we computed, i.e., $\mathcal{O}(1/n^{c-1})$. The total number of gates (without counting the $\lfloor n/2 \rfloor$ SWAP gates[2]), on the other hand, would be

$$T(n) \leq T(n-1) + c \log n$$

which gives $T(n) \leq cn \log n = \mathcal{O}(n \log n)$. By increasing this constant $c$ we can increase the exponent of the error bound without penalties in the asymptotic complexity.

## 5.5 Quantum phase estimation: exact case

The QFT algorithm is a subroutine for one of the most important algorithms of the first era.

**Problem 5.18** (Phase estimation)**.** Let $U$ be a unitary given as a quantum circuit, and let $|\psi\rangle$ be an eigenstate of $U$ given in a register and associated with the eigenvalue $e^{2\pi i \varphi}$, i.e., $U|\psi\rangle = e^{2\pi i \varphi}|\psi\rangle$. Return an estimate of $\varphi$.

In other words, we want to estimate the eigenvalue $\lambda$ associated to a given state (remember that $|\lambda| = 1$ by Theorem 5.2 so it can be written as a phase). For now we will assume that $\varphi = \ell/2^n$, i.e., $\varphi = 0.\ell_1\ell_2\cdots\ell_n$ has $n$ digits after the point and $U|\psi\rangle = \omega^\ell|\psi\rangle$. The standard basis state $|k\rangle$ can be seen as the encoding of the peak signal

$$e_k[t] = \begin{cases} 1 & t = k \\ 0 & t \neq k \end{cases}$$

[2] To get $\mathcal{O}(n \log n)$ complexity we are required to use Remark 5.17, otherwise the circuit before the recursion would still cost $\mathcal{O}(n)$.

$$\begin{tikzcd}
f[t] \arrow[r, dashed] \arrow[d, "F_{2^n}"'] & f[t-\ell] \\
\hat{f}[s] \arrow[r] & \hat{f}[s]\cdot\omega^{s\ell} \arrow[u, "F_{2^n}^{\dagger}"']
\end{tikzcd}$$

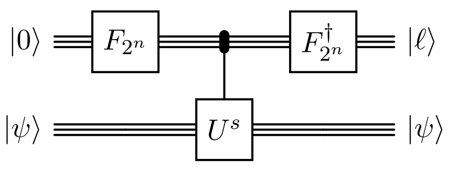
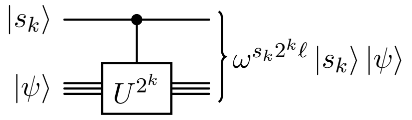
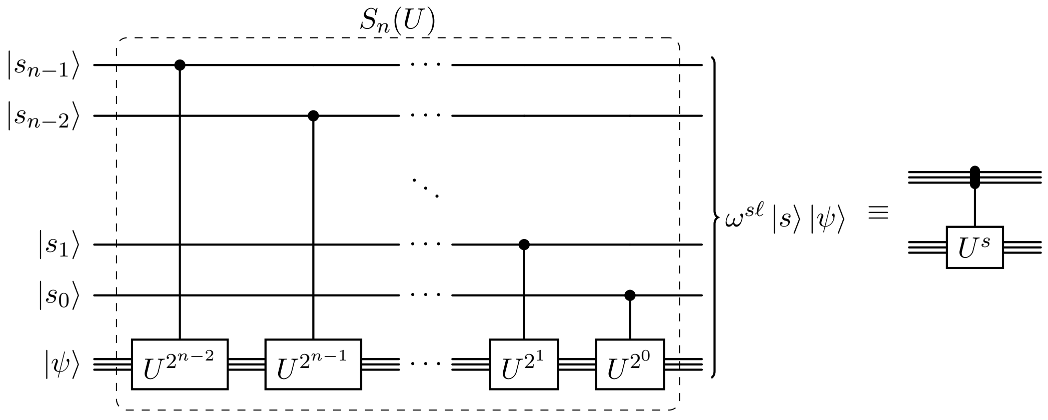


Figure 5.5: Using Lemma 5.13 to solve phase estimation. The quantum phase estimation algorithm will take the bottom path in the diagram with $f = e_0$.

The idea of the algorithm is to leverage Lemma 5.13: we start from the state $|0\rangle$, we apply the Fourier transform, we multiply the signal $\hat{e}_0[s]$ by $\omega^{s\ell}$ and then we apply the inverse Fourier transform. The resulting signal will be the shifted signal $e_0[t-\ell] = e_\ell[t]$, so the final state will be $|\ell\rangle$, which can then be measured in the standard basis. We only need to understand how to multiply the signal by $\omega^{s\ell}$, and we use the idea of phase transduction again: if the $k$-th qubit is in state $|1\rangle$ then apply phase $\omega^{2^k\ell}$, otherwise do nothing. This is equivalent to applying $\omega^{s_k 2^k \ell}$, and can be easily done with $C(U^{2^k})$:



where $|s_k\rangle \otimes \omega^{s_k 2^k \ell}\,|\psi\rangle = \omega^{s_k 2^k \ell}\,|s_k\rangle\,|\psi\rangle$ by phase kickback. We do this for each $k$ so that the overall phase added will be $(s_0 + 2s_1 + \cdots + 2^{n-1}s_{n-1})\ell = s\ell$.



This is enough because by linearity we have

$$|g\rangle\,|\psi\rangle = \sum_{s=0}^{N-1} g[s]\,|s\rangle\,|\psi\rangle \mapsto \sum_{s=0}^{N-1} g[s]\cdot\omega^{s\ell}\,|s\rangle\,|\psi\rangle$$

which is exactly what we were looking for. The full quantum phase estimation circuit is depicted in Figure 5.5.

**Remark 5.19.** Note that $F_N\,|0\rangle = \frac{1}{\sqrt{N}}\sum_{s=0}^{N-1}|s\rangle$ is simply the equal superposition of all the states, so if we start from the $|0\rangle$ state, we might replace the first QFT with a much cheaper $n$-fold Hadamard gate $H^{\otimes n}$ and obtain the same result.

## 5.6 Quantum phase estimation: approximate case

The algorithm of the previous section works perfectly (i.e., it returns the correct estimate with certainty) if $\varphi$ is an integer multiple of $1/2^n$. It happens frequently, however, that $\varphi$ is a generic rational (or even irrational) number, or that maybe we know how many digits $n$ after the point $\varphi$ has, but it's too large and implementing the circuit for such a large $n$ would be costly. On the other hand, we would be happy with just an $\epsilon$-approximation.

**Theorem 5.20.** *Let $\varphi = (\ell+\delta)/2^n$ with $\ell \in \mathbb{Z}_{2^n}$ and $|\delta| \leq \frac{1}{2}$, i.e., $\ell/2^n$ is the best approximation of $\varphi$ with $n$ bits after the point. The quantum phase estimation circuit of Figure 5.5 will return $\ell$ with probability $\geq 0.4$.*

Notice that in this case our estimate $\ell/2^n$ is close to $\varphi$ up to $|\delta|/2^n \leq 1/2^{n+1}$ error.

*Proof.* Let $\ell^* = \ell + \delta = N\varphi \in \mathbb{R}$ be the ideal output, and take once again $\omega = \omega_N = e^{2\pi i/N}$. We compute the final state:

$$\begin{aligned}
|0\rangle &\overset{F_N}{\mapsto} \frac{1}{\sqrt{N}} \sum_{s=0}^{N-1} |s\rangle \\
&\mapsto \frac{1}{\sqrt{N}} \sum_{s=0}^{N-1} \omega^{s\ell^*} |s\rangle && \text{since } \omega^{\ell^*} = e^{2\pi i\varphi} \\
&\overset{F_N^\dagger}{\mapsto} \frac{1}{\sqrt{N}} \sum_{s=0}^{N-1} \omega^{s\ell^*} \left( \frac{1}{\sqrt{N}} \sum_{t=0}^{N-1} \omega^{-st} |t\rangle \right) \\
&= \sum_{t=0}^{N-1} \left( \frac{1}{N} \sum_{s=0}^{N-1} \omega^{s\ell^*} \omega^{-st} \right) |t\rangle
\end{aligned}$$

If this final state is $|\psi\rangle$, the probability of measuring $\ell$ will be

$$\begin{aligned}
|\langle \ell|\psi\rangle|^2 &= \left| \frac{1}{N} \sum_{s=0}^{N-1} \omega^{s(\ell^*-\ell)} \right|^2 \\
&= \left| \frac{1}{N} \cdot \frac{e^{2\pi i\delta} - 1}{e^{2\pi i\delta/N} - 1} \right|^2 && \text{by a geometric sum}
\end{aligned}$$

Note that $\left|e^{2ix} - 1\right| = \left|e^{ix} - e^{-ix}\right| = 2 \cdot |\sin x|$, so the probability ultimately becomes

$$|\langle \ell|\psi\rangle|^2 = \left| \frac{1}{N} \cdot \frac{\sin(\pi\delta)}{\sin(\pi\delta/N)} \right|^2 = \frac{1}{N^2} \cdot \frac{\sin^2(\pi\delta)}{\sin^2(\pi\delta/N)} \ .$$

For $|x| \leq \frac{\pi}{2}$ the inequality $|\sin x| \geq \frac{2|x|}{\pi}$ holds, as well as $|\sin x| \leq |x|$. Since $|\delta| \leq \frac{1}{2}$, we can lower bound the quantity above

$$|\langle \ell|\psi\rangle|^2 \geq \frac{1}{N^2} \cdot \frac{4|\delta|^2}{\pi^2|\delta|^2/N^2} = \frac{4}{\pi^2} \geq 0.4$$

as claimed. □

Therefore the phase estimation circuit will return the best possible approximation to $\varphi$ with probability at least 0.4. We can also prove that, if we instead consider the two closest

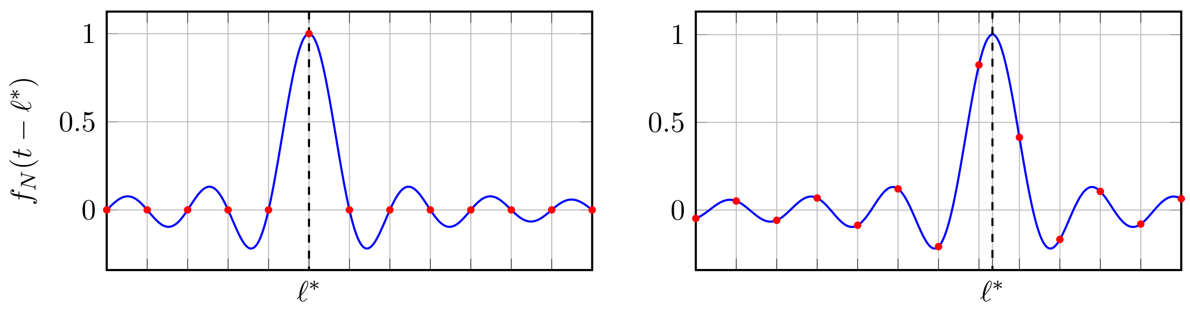


Figure 5.6: Plot of the amplitude wave $f_N(t) = \frac{1}{N} \cdot \frac{\sin(\pi t)}{\sin(\pi t/N)}$ shifted by the quantum phase estimation algorithm. After the shift $\ell^*$ will coincide with the peak of the function. If this number turns out to be an integer, then it will be returned with certainty (exact case, left figure). Otherwise $\ell^*$ will fall somewhere between the two integers $\lfloor \ell^* \rfloor, \lceil \ell^* \rceil$, which are the integers with the highest amplitude since they are the closest to the peak (right figure).

approximations $\frac{\ell}{N} \leq \varphi \leq \frac{\ell'}{N}$, then the circuit will return one of $\ell, \ell'$ with probability at least 0.8 (see Exercise 5.4). We found that the amplitude $\langle t|\psi\rangle$ is equal to[3]:

$$f_N(t-\ell^*) = \frac{1}{N} \cdot \frac{\sin(\pi(t-\ell^*))}{\sin(\pi(t-\ell^*)/N)}$$

where $\delta = t - \ell^*$ in the above proof would generally be the distance from the candidate estimate $t$ to our target $\ell^*$ (see Figure 5.6). Since also our starting state $|0\rangle$ can be rewritten as[4]

$$|0\rangle = \sum_{t=0}^{N-1} f_N(t)\, |t\rangle \ ,$$

the circuit in Figure 5.5 can still be seen as a shift $f_N(t) \to f_N(t-\ell^*)$ in the continuous variable $t$, but this $t$ will only be considered as an element of $\mathbb{Z}_N$ when we measure.

**Remark 5.21.** The function we found for the amplitudes $\langle t|\psi\rangle$

$$g_N(t) = \frac{1}{N} \cdot \frac{e^{2\pi i t} - 1}{e^{2\pi i t/N} - 1}$$

is the so-called *Fourier interpolation* of the discrete signal $e_0[t]$.

$$g_N(t) = \frac{1}{\sqrt{N}} \sum_{s=0}^{N-1} \hat{e}_0[s] \cdot \omega^{-st}$$

i.e., we take the Fourier transform of $e_0$, followed by the inverse, but this time we let $t$ be chosen freely on $\mathbb{R}$. In the points when $t$ is an integer we have $e_0[t] = g_N(t)$.

[3]Here actually we neglected phases in the calculations, but they are not important since we are going to measure in this basis, thus taking squared absolute values of this function anyway.

[4]Note that $f_N(t)$ is not defined in $t = 0$, but $\lim_{t\to 0} f_N(t) = 1$, so we let $f_N(0) = 1$ and remove the singularity.

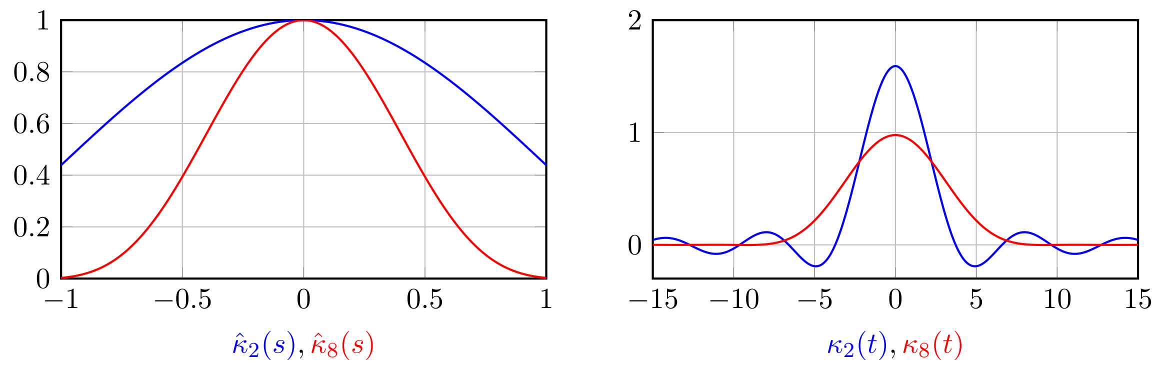


Figure 5.7: Plot of the Kaiser window $\hat{\kappa}_\beta(s)$ and its inverse Fourier transform $\kappa_\beta(t)$. In the left picture the functions are zero outside of the plotted interval. The transformed windows in the right picture are similar in shape to the one in Figure 5.6 obtained when $\beta \to 0$, in which case the Kaiser window becomes a rectangular wave. On the other hand, as $\beta$ increases the side lobes of $\kappa_\beta$ will get exponentially damped, making the function look like a gaussian.

## 5.7 [unfinished] Sharp phase estimation with the Kaiser window

The algorithm of the previous sections will return an estimate with error $\leq 1/2N$ with probability $\geq 0.8$, which is quite enough for a variety of applications: in order to obtain a good estimate with probability $1-\eta$, we can simply run the algorithm $K$ independent times and return the output appeared the most frequently: we succeed with probability $\geq 1-\eta$ if $K = \mathcal{O}(\log \frac{1}{\eta})$[5].

Sometimes, however, we will need to keep the phase *coherently*, i.e., we want to use the output state of the phase estimation algorithm for further quantum computation. In this situation we will need a *sharp* estimate, which is a good approximation with probability $1-\eta$ for some $\eta > 0$ we can choose. The weakness point of the algorithm we presented lies in the shape of $f_N$: the peak is too short, and we would like to produce and shift a wave whose peak is higher compared to the values in the rest of the domain.

**Definition 5.22.** The *Kaiser window* is a function

$$\hat{\kappa}_\beta(s) = \begin{cases} \dfrac{I_0(\beta\sqrt{1-s^2})}{I_0(\beta)} & |s| \leq 1 \\ 0 & |s| > 1 \end{cases}$$

where $I_0(x) = \sum_{k=0}^{\infty} \frac{1}{(2k)!}\left(\frac{x}{2}\right)^{2k}$ is the zeroth-order modified Bessel function of the first kind.

We won't dive into the definition of $I_0$, but essentially the Kaiser window has a shape that is very similar to a truncated gaussian (see Figure 5.7, left picture), except that for $|s| > 1$ the function is *exactly* zero. The parameter $\beta$ determines the sharpness of the curve.

**Theorem 5.23.** *The continuous inverse Fourier transform of the Kaiser window is*

$$\kappa_\beta(t) = \int_{-\infty}^{+\infty} \hat{\kappa}_\beta(s) e^{-ist} \, \mathrm{d}s = \frac{2\sin\left(\sqrt{t^2-\beta^2}\right)}{I_0(\beta)\sqrt{t^2-\beta^2}}$$

[5]Let $G$ be the number of 'good' runs which output $\lfloor \ell^* \rfloor$ or $\lceil \ell^* \rceil$. If $G > \frac{2K}{3}$ then one of $\lfloor \ell^* \rfloor, \lceil \ell^* \rceil$ appears $\geq \frac{K}{3}$ times, while any other value appears $< \frac{K}{3}$ times. Since $\mathbb{E}[G] \geq 0.8K$, the probability that this doesn't happen decays exponentially, by a Chernoff bound.

Notice that the function is well-defined also when $|t| < \beta$, as we can simply take out an imaginary unit, and use the identity $\sin(ix) = i \sinh x$. The function will then be

$$\kappa_\beta(t) = \frac{2\sinh\left(\sqrt{\beta^2 - t^2}\right)}{I_0(\beta)\sqrt{\beta^2 - t^2}}$$

which is still real (see Figure 5.7, right picture). $\kappa_\beta$ has a similar shape to the $f_N$ of Section 5.6 (indeed, as $\beta \to 0$, $\hat{\kappa}_\beta$ becomes a rectangular wave as the one we prepare in the plain phase estimation circuit). As $\beta$ increases, the shapes both $\hat{\kappa}_\beta$ and $\kappa_\beta$ become similar to gaussians[6], so we can imagine the Kaiser window to be halfway between a rectangular wave and a gaussian curve. In particular, the side lobes of $\kappa_\beta$ will become exponentially close to zero, so the probability will be concentrated only on the main lobe, allowing us to obtain better success probability than the 0.8 of the previous algorithm.

Remember that the phase estimation will just leverage the shift property of the Fourier transform to make the peak coincide with the target phase $\varphi$, so if we manage to construct a quantum state containing $\hat{\kappa}_\beta(s)$ we are done. The hard part consists in the fact that we actually have a discrete Fourier transform instead of a continuous one, so we'll have to deal with the approximation errors that come with this difference.

## 5.8 Quantum amplitude estimation

As a very first application of phase estimation, we are going to consider a problem that arises in many settings. Considering the unitary of Section 4.4

$$\mathcal{A}\left|0\right\rangle = \sqrt{p}\left|\alpha\right\rangle + \sqrt{1-p}\left|\beta\right\rangle \ ,$$

our objective is to estimate $p$ given access to $\mathcal{A}$ and the phase oracle $U_f$ flipping the sign of the desired state $|\alpha\rangle$. We've shown that the Grover iteration $G = R_\Phi U_f$ acts like a rotation of angle $2\theta$, where $\theta = \arcsin\sqrt{p}$ in the subspace containing $\mathcal{A}|0\rangle$. A rotation of angle $2\theta$ has eigenvalues $e^{\pm 2i\theta}$ (see Exercise 5.5), so if we apply the phase estimation circuit to $G$ and input one of its eigenstates $|\psi_+\rangle, |\psi_-\rangle$, we will get either $+2\theta$ or $-2\theta$. We do not have an eigenstate of $G$, but we know that $\mathcal{A}|0\rangle$ is in the subspace, and so it can be expressed as a linear combination of them.

$$\mathcal{A}\left|0\right\rangle = \sigma_+\left|\psi_+\right\rangle + \sigma_-\left|\psi_-\right\rangle$$

Assuming a perfect phase estimation, we obtain

$$\begin{aligned} QPE_G\left|0\right\rangle \otimes \mathcal{A}\left|0\right\rangle &= \sigma_+ QPE_G\left|0\right\rangle\left|\psi_+\right\rangle + \sigma_- QPE_G\left|0\right\rangle\left|\psi_-\right\rangle \\ &= \sigma_+\left|+2\theta\right\rangle\left|\psi_+\right\rangle + \sigma_-\left|-2\theta\right\rangle\left|\psi_-\right\rangle \end{aligned}$$

and by measuring the first register we get $\pm 2\theta$ with probability $|\sigma_\pm|^2$ (any of these two values is fine as the sign will go away when we compute $p = \sin^2\theta$). The estimate $\hat{p}$ of $p$ will satisfy

$$|\hat{p} - p| = |\sin^2\hat{\theta} - \sin^2\theta| \le |\hat{\theta} - \theta|$$

and we need $\mathcal{O}(\frac{1}{\epsilon})$ to get an estimate of $\pm\theta$ up to $\epsilon$-error.

[6]Note that the Fourier transform of a gaussian is still a gaussian, so if $\kappa_\beta$ becomes closer and closer to a gaussian, then also its Fourier transform must be, and vice versa.

**Remark 5.24.** We can also estimate $p$ classically in the following sense: construct $K$ copies of $\mathcal{A}\,|0\rangle$, measure them, and take the average of the $K$ outcomes. In order to obtain $\epsilon$ error on the estimate we need $K = \mathcal{O}(\frac{1}{\epsilon^2})$ copies, so amplitude estimation gives a quadratic speed-up.

**Remark 5.25.** Amplitude estimation works with the exact same argument also on the case of oblivious amplitude amplification shown in Section 4.5.

## 5.9 Quantum Fourier transform over general orders

Algorithm 5.1 shows how to construct $F_N$ when $N = 2^n$ is a power of two, but we will also need $F_N$ over general order $N$, and we are going to use phase estimation for it.

Our goal is to construct the transformation

$$|x\rangle \mapsto |\hat{x}\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega_N^{xy} \,|y\rangle$$

for all $x \in \mathbb{Z}_N$ (the action on the rest of the space will follow by linearity). We introduce another register of the same size and carry out the following two steps:

$$|x\rangle_A\,|0\rangle_B \mapsto |x\rangle_A\,|\hat{x}\rangle_B \mapsto |0\rangle_A\,|\hat{x}\rangle_B \ .$$

After this process we can apply a swap operation and discard the additional register (which is reset and thus unentangled). We turn our attention on the first step, namely to prepare $|\hat{x}\rangle$ on register $B$. We first prepare the equal superposition over $N$ elements, obtaining

$$|x\rangle_A \otimes \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} |y\rangle_B$$

and this can be done coherently (see Exercise 4.6), we call $E_N$ the unitary doing this transformation.

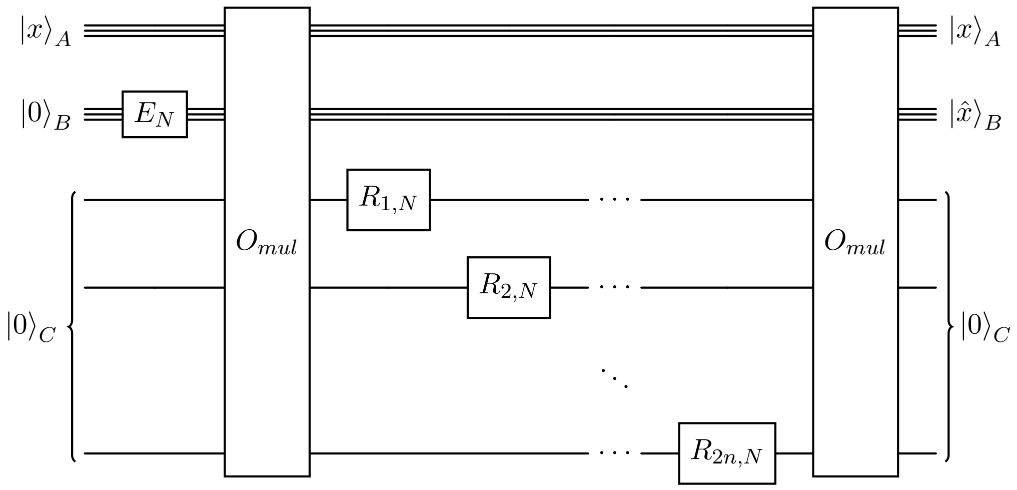
The next step is to perform the transformation $|x\rangle\,|y\rangle \mapsto \omega_N^{xy}\,|x\rangle\,|y\rangle$ in order to make the phases appear in the superposition. This can be done by constructing a bit oracle $O_{mul}$ for the multiplication, so that $x \cdot y$ is stored coherently in a new register $C$ (of double size)

$$|x\rangle_A\,|y\rangle_B\,|0\rangle_C \mapsto |x\rangle_A\,|y\rangle_B\,|x \cdot y\rangle_C \tag{5.9}$$

At this point we use once again the idea of phase transduction to make $\omega_N^{xy}$: if the $k$-th significant bit of $d = xy$ is 1, then we apply a phase $\omega_N^{2^k}$, i.e., we apply the gate

$$R_{k,N} = \begin{bmatrix} 1 & 0 \\ 0 & \omega_N^{2^k} \end{bmatrix} \ .$$

on the $k$-th qubit of $C$ for $0 \leq k < 2n$. Once we're done we can uncompute the multiplication with the inverse bit oracle and discard register $C$. After these steps we have the state $|x\rangle_A\,|\hat{x}\rangle_B$, completing the first transformation in (5.9).



The second step is where phase estimation comes into play: consider the unitary $U_{+1}$ implementing an increment modulo $N$

$$U_{+1}|x\rangle = |(x+1) \bmod N\rangle$$

The Fourier transform $F_N$ we are looking for diagonalizes this unitary, namely $U_{+1}|\hat{x}\rangle = \omega_N^x |\hat{x}\rangle$. A (perfect) phase estimation using $F_N$ would carry out the transformation

$$QPE_N |0\rangle_A |\hat{x}\rangle_B = |x\rangle_A |\hat{x}\rangle_B$$

and so $QPE_N^\dagger$ would do the trick. But of course we'd need $F_N$, which is exactly what we're trying to achieve. So instead of $N$, we take $M = 2^m$ to be the smallest power of two $\geq N$ and carry out phase estimation with $m$ bits of precision on another register $A'$.

$$QPE_M |0\rangle_{A'} |\hat{x}\rangle_B = \sum_{\tilde{x}} \alpha_{\tilde{x}} |\tilde{x}\rangle_{A'} \otimes |\hat{x}\rangle_B$$

where $|\alpha_{\tilde{x}}|^2$ gives the probability distribution of the output (remember that the true phase is $x/N$ which is generally not an integer multiple of $1/M$, so phase estimation will not be exact). If phase estimation succeeds, $\tilde{x}$ is an integer satisfying

$$\left|\frac{\tilde{x}}{M} - \frac{x}{N}\right| \leq \frac{1}{2M} \qquad \Longrightarrow \qquad \left|\tilde{x}\cdot\frac{N}{M} - x\right| < \frac{1}{2}\,. \tag{5.10}$$

We now take the bit oracle $O_g$ of a function $g(\tilde{x})$ that returns the closest integer to $\tilde{x}\frac{N}{M}$, and place the output in register $A$

$$|0\rangle_A |\tilde{x}\rangle_{A'} |\hat{x}\rangle_B \overset{O_g}{\mapsto} |g(\tilde{x})\rangle_A |\tilde{x}\rangle_{A'} |\hat{x}\rangle_B$$

We conclude by undoing the phase estimation $QPE_M^\dagger$, and let $V = QPE_M^\dagger O_g QPE_M$ be the complete unitary of this second step (where each of these matrices is intended to be applied to the correct registers). Notice that in the branches of the superposition where the phase estimation succeeded to give a good estimate we have $g(\tilde{x}) = x$ by (5.10).

$$O_g QPE_M |0\rangle_A |0\rangle_{A'} |\hat{x}\rangle_B = \sum_{\tilde{x}\text{ good}} \alpha_{\tilde{x}} |x\rangle_A |\tilde{x}\rangle_{A'} |\hat{x}\rangle_B + \sum_{\tilde{x}\text{ bad}} \alpha_{\tilde{x}} |g(\tilde{x})\rangle_A |\tilde{x}\rangle_{A'} |\hat{x}\rangle_B \tag{5.11}$$

If we assume that phase estimation is sharp, i.e., the returned $\tilde{x}$ will be good with probability at least $1-\eta$ for some $\eta > 0$, then the squared norm of the vector in the second sum will be at most $\eta$, and so

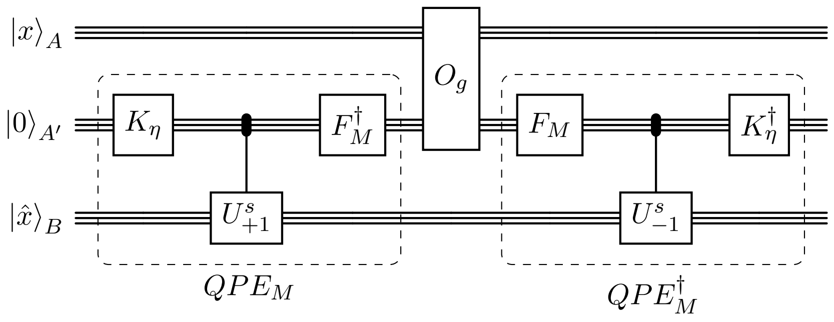
$$\left\| \sum_{\tilde{x}\text{ bad}} \alpha_{\tilde{x}} |g(\tilde{x})\rangle_A |\tilde{x}\rangle_{A'} |\hat{x}\rangle_B - \sum_{\tilde{x}\text{ bad}} \alpha_{\tilde{x}} |x\rangle_A |\tilde{x}\rangle_{A'} |\hat{x}\rangle_B \right\| \le 2\sqrt{\eta}$$

by a triangle inequality. This implies that if we replaced $|g(\tilde{x})\rangle_A$ with $|x\rangle_A$ also for the bad estimates in the vector of (5.11) we wouldn't be changing much. In other words, by replacing and collecting $|x\rangle_A$ we get

$$\left\| O_g QPE_M |0\rangle_A |0\rangle_{A'} |\hat{x}\rangle_B - |x\rangle_A \sum_{\tilde{x}} \alpha_{\tilde{x}} |\tilde{x}\rangle_{A'} |\hat{x}\rangle_B \right\| \le 2\sqrt{\eta} \tag{5.12}$$

By also applying $QPE_M^\dagger$, we obtain the final error bound

$$\| V |0\rangle_A |0\rangle_{A'} |\hat{x}\rangle_B - |x\rangle_A |0\rangle_{A'} |\hat{x}\rangle_B \| \le 2\sqrt{\eta} \; . \tag{5.13}$$



The full procedure we outlined in this section is summarized in Algorithm 5.2.

There is only one final caveat to address: from what we have seen in this chapter, in order to apply $m$-bit phase estimation we need to use $C(U_{+1})$ $M$ times, which is exponentially high. However, notice that

$$U_{+1}^{2^k} |x\rangle = |(x + 2^k) \bmod N\rangle$$

i.e., applying $U_{+1}$ $2^k$ times is equivalent to a much cheaper addition $x \mapsto x + 2^k$, which can be implemented easily. Hence we can directly implement $U_{+2^k}$ for increasing $k$. We say that $U_{+1}$ is *fast-forwardable*, and we only need to implement $\mathcal{O}(\log M)$ of these controlled unitaries in order to carry out a full phase estimation. We will see that also Shor's algorithm heavily relies on fast-forwarding.

**Remark 5.26.** In order to obtain a sharp estimation we do not need to resort to the Kaiser window, but it is sufficient to use the plain phase estimation and increase the number of bits of precision. This requires less gates at the expense of needing more qubits.

## 5.10 Exercises

**Exercise 5.1.** Let $U$ be unitary, and $|\psi_1\rangle, |\psi_2\rangle$ are two eigenstates associated to two distinct eigenvalues $\lambda_1 \neq \lambda_2$. Show that $\langle\psi_1|\psi_2\rangle = 0$.

**Algorithm 5.2** Quantum Fourier transform over general order

```
Input: Integer $N$, an input state $|x\rangle$ in register $A$, and a desired precision $\epsilon$.
Output: The Fourier transform $|\hat{x}\rangle = F_N |x\rangle$ in register $A$, up to $\epsilon$ error.
  Prepare $|\hat{x}\rangle_B$ on a register $B$ as big as $A$ using phase transduction.
      $|x\rangle_A \to |x\rangle_A \otimes |\hat{x}\rangle_B$
  By letting $m = \lceil \log_2 N \rceil$, apply $m$-bit phase estimation with sharpness $\eta \le \epsilon^2/4$ on the unitary
      $U_{+1} |x\rangle = |(x+1) \bmod N\rangle$
  applied to register $B$, saving the estimate to a $m$-qubit register $A'$.
  Apply the bit oracle $O_g$ of the function $g(\tilde{x})$ returning the closest integer to $\tilde{x} \cdot \frac{N}{M}$, with $A'$
  as input register and $A$ as output register.        ▷ This will XOR $x$ to register $A$
  Undo the phase estimation and swap registers $A, B$.
```

**Exercise 5.2.** Consider a unitary or Hermitian matrix $A$ with eigenvalues $\lambda_1, \dots, \lambda_N$ and an associated eigenbasis $|\psi_1\rangle, \dots, |\psi_N\rangle$, i.e., $U|\psi_k\rangle = \lambda_k |\psi_k\rangle$.

Show that $A$ can be written in the following form:

$$A = \sum_{k=1}^{N} \lambda_k |\psi_k\rangle\langle\psi_k|$$

**Exercise 5.3.** Show that the Pauli-$X$ rotation matrix $e^{i\phi X}$, where $\phi \in \mathbb{R}$ is the angle of rotation, is of the form

$$e^{i\phi X} = \begin{bmatrix} \cos\phi & i\sin\phi \\ i\sin\phi & \cos\phi \end{bmatrix}$$

**Exercise 5.4.** Here we extend Theorem 5.20: suppose that $\varphi = (\ell+\delta)/2^n$ with $0 \le \delta < 1$. The two closest estimates to $\varphi$ will then be $\ell/2^n, \ell'/2^n$, where $\ell' = \ell + 1$. Show that the probability of the quantum phase estimation circuit of Figure 5.5 returns $\ell$ or $\ell'$ with probability $\ge 0.8$.

**Exercise 5.5.** Let $R$ be the rotation matrix

$$R = \begin{bmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix}$$

Show that its eigenvalues are $e^{\pm i\varphi}$.

# Chapter 6

# Shor's algorithm

We are finally ready to use what we've discussed so far to tackle FACTORING and DISCRETELOG. The full procedure for Shor's algorithms is outlined [11], including the classical post-processing and the necessary number-theoretic notions.

## 6.1 Number theory

The *fundamental theorem of arithmetic* tells us that any number $x \in \mathbb{N}^+$ can be uniquely represented as a finite multiset[1] $\mathcal{I}(x)$ of prime factors. For example, $\mathcal{I}(10) = \{5, 2\}, \mathcal{I}(12) = \{2, 3, 3\}$. In this view, the prime numbers are precisely represented by the sets containing exactly one element (themselves), and $\mathcal{I}(1)$ is the empty multiset[2].

We say that $b$ divides $a$ (written as $b \,\|\, a$), if $a/b$ is integer or, equivalently, $\mathcal{I}(b) \subseteq \mathcal{I}(a)$. Moreover, $a, b$ are co-prime (written as $a \perp b$) if they don't share any common factor, i.e., $\mathcal{I}(a) \cap \mathcal{I}(b) = \emptyset$. The numbers represented by $\mathcal{I}(a) \cap \mathcal{I}(b), \mathcal{I}(a) \cup \mathcal{I}(b)$ are the *greatest common divisor* $\gcd(a, b)$ and the *least common multiple* $\operatorname{lcm}(a, b)$, respectively. Clearly, $a \perp b$ if and only if $\gcd(a, b) = 1$. Notice also that $\operatorname{lcm}(a, b) = ab/\gcd(a, b)$, so the lcm is easily computed from the gcd and vice versa.

Given an integer $r > 0$, say that an integer $x$ is a *totative* of $r$ if $0 \leq x < r$ and $x \perp r$. We use $\mathbb{Z}_r^*$ to denote the set of totatives of $r$, whose size is known as *Euler's totient function* $\varphi(r) = |\mathbb{Z}_r^*|$. It is known that $\mathbb{Z}_r^*$ is also the subset of those elements of $\mathbb{Z}_r$ that admit a multiplicative inverse, i.e., $x \in \mathbb{Z}_r^*$ if and only if there exists $g \in \mathbb{Z}_r$ such that $gx \equiv 1 \bmod r$. An asymptotic bound on the totient function — important for our complexity analysis — holds[3]

$$\frac{\varphi(r)}{r} = \Omega\left(\frac{1}{\log\log r}\right)$$

and this quantity can be seen as the probability that a uniformly random $s \in \mathbb{Z}_r$ is also in $\mathbb{Z}_r^*$. In other words, such a random $s$ is *very likely* to be co-prime to $r$.

[1] A set, except that elements could appear more than once (the number of occurrences is called *multiplicity*).

[2] This is why it's inconvenient to define 1 to be prime: we could add any number of ones to a multiset without changing the represented number, losing uniqueness.

[3] More precisely, we know that $\varphi(r) > \dfrac{r}{e^\gamma \log\log r + \frac{3}{\log\log r}}$, where $\gamma \simeq 0.577$ is the Euler-Mascheroni constant.

## 6.2 Continued fractions and the Euclidean algorithms

Continued fractions give a way to represent real numbers. Given a sequence of positive natural numbers $a_0, a_1, \ldots \in \mathbb{N}^+$ we define the continued fraction of $a$ to be

$$[a_0; a_1, a_2, \ldots] := a_0 + \cfrac{1}{a_1 + \frac{1}{a_2 + \frac{1}{\ldots}}}$$

Given a positive real number $x \in \mathbb{R}^+$, finding its continued fraction is easy by a recursive process: divide $x$ into its integer and fractional part

$$x = a_0 + \delta = a_0 + \frac{1}{x_0} \ .$$

The process goes on by computing the rest of the sequence as the continued fraction of $x_0$, and the $a_k$ are called *partial quotients.*

**Example 6.1.** Let's compute the first elements of the convergent fraction for $\pi$:

$$\pi = 3 + 0.1415\ldots = 3 + \frac{1}{7.0625\ldots} = 3 + \frac{1}{7 + 0.06\ldots} = 3 + \frac{1}{7 + \frac{1}{15.9965\ldots}}$$

So the first three partial quotients for $\pi$ are $3, 7, 15$.

Notice that $\delta < 1$ always, so $x_0 > 1$ and the $a_k$'s are always non-zero except maybe for $a_0$. This unless $x$ turns out to be rational, in which case we will eventually get $\delta = 0$, which will stop the process.

**Lemma 6.2.** *If $x \in \mathbb{Q}$, then for some finite number $K$*

$$x = [a_0; a_1, \ldots, a_K] = a_0 + \cfrac{1}{a_1 + \frac{1}{a_2 + \frac{1}{\cdots + \frac{1}{a_K}}}}$$

**Example 6.3.** Consider $x = \frac{415}{93}$. Its continued fraction is computed as

$$\frac{415}{93} = 4 + \frac{43}{93} = 4 + \frac{1}{2 + \frac{7}{43}} = 4 + \cfrac{1}{2 + \frac{1}{6 + \frac{1}{7}}}$$

Thus $\frac{415}{93} = [4; 2, 6, 7]$.

Note that for a rational the sequence is not unique: in particular $a_K = (a_K - 1) + \frac{1}{1}$, so if we replace the last $a_K$ with $a_K - 1$ and add a 1 to the sequence, we get the same number. We cannot repeat this procedure though, as we would get a zero in the second to last element which makes the sequence collapse as

$$(a_K - 1) + \frac{1}{0 + \frac{1}{1}} = a_K$$

So for each rational number there is a unique sequence of even length, and a unique sequence of odd length.

For a continued fraction $x = [a_0; a_1, a_2, \ldots]$ (finite or infinite), the truncated sequence $[a_0; a_1, \ldots, a_k]$ yields a rational number $\frac{p_k}{q_k}$ called the $k$-th *convergent* of $x$.

**Example 6.4.** The first two convergents of $\pi$ are $\frac{22}{7} = 3.14$ and $\frac{355}{113} = 3.14159292$. Notice how the second convergent is already close to $\pi$ up to $3 \cdot 10^{-7}$ error.

Given two natural numbers $p, q \in \mathbb{N}^+$, computing the convergents of $x_0 = \frac{p}{q}$ is easily done using the *Euclidean algorithm*: the partial quotients $a_k$ are computed with the recursive formula that follows Example 6.3

$$x_k = a_k + \frac{1}{x_{k+1}} \qquad \Longrightarrow \qquad \begin{cases} a_k = \lfloor x_k \rfloor \\ x_{k+1} = \frac{1}{x_k - a_k} \end{cases}$$

We don't really want to work with floating points, so let's rewrite the equation on the left in fractional form $x_k = \frac{N_k}{D_k}$

$$\frac{N_k}{D_k} = a_k + \frac{N_k - a_k D_k}{D_k} \overset{!}{=} a_k + \frac{D_{k+1}}{N_{k+1}} \qquad \Longrightarrow \qquad \begin{cases} D_{k+1} = N_k - a_k D_k \\ N_{k+1} = D_k \end{cases}$$

The second equation tells us that we only need to work with $N_k$, and the relation will be

$$N_k = a_k N_{k+1} + N_{k+2}$$

with initial conditions $N_0 = p, N_1 = D_0 = q$ (see Algorithm 6.1).

**Remark 6.5.** The last non-zero $N_k$ will be $\gcd(p, q)$, so we can use the Euclidean algorithm to compute greatest common divisor and least common multiple as well.

**Theorem 6.6** (Lamé's theorem)**.** *The Euclidean algorithm converges in at most* $\log_\phi(3q)$ *steps, where* $\phi = \frac{1+\sqrt{5}}{2}$ *is the golden ratio.*

This also implies that any rational number $x = \frac{p}{q}$ is expressible as a continued fraction of at most $\log_\phi(3q)$ terms.

*Proof.* Notice that $a_k \geq 1$ for $k > 0$, therefore we can lower bound

$$N_k \geq N_{k+1} + N_{k+2} \ .$$

If $N_K$ is the last non-zero term of this sequence, then $N_{K-k+1} \geq F_k$, the sequence of Fibonacci numbers. In particular, $q = N_1 \geq F_K$, and we also know that $F_K \geq \phi^{K-2}$, therefore

$$\phi^{K-2} \leq q \qquad \Longrightarrow \qquad K \leq \log_\phi(\phi^2 q) \leq \log_\phi(3q)$$

as claimed. $\square$

We would like to also compute the convergents $p_k/q_k$ along the way, and we can do it using what is known as the *extended Euclidean algorithm* (EEA): the initial convergent $\frac{p_0}{q_0}$ is simply $\frac{a_0}{1}$, while the next one is

$$\frac{p_1}{q_1} = a_0 + \frac{1}{a_1} = \frac{a_0 a_1 + 1}{a_1} \ .$$

Notice that if we allow $a_k$ to also be rationals for a second, then it is easy to check that $[a_0; a_1, \dots, a_n] = [a_0; a_1, \dots, a_{n-1} + \frac{1}{a_n}]$, so in order to obtain the next convergent we just have to replace $a_k$ with $a_k + \frac{1}{a_{k+1}}$. We can then obtain

$$\frac{p_2}{q_2} = \frac{a_0(a_1 + \frac{1}{a_2}) + 1}{a_1 + \frac{1}{a_2}} = \frac{a_2 a_0 a_1 + 1 + a_2}{a_2 a_1 + 1} = \frac{a_2 p_1 + p_0}{a_2 q_1 + q_0} \ .$$

**Algorithm 6.1** Euclidean algorithm for continued fractions

```
Input: $p, q \in \mathbb{N}$.
Output: $[a_0; a_1, \dots, a_K] = \frac{p}{q}$.
  $N_0 \leftarrow p, N_1 \leftarrow q$
  $k \leftarrow 0$
  while $N_{k+1} \neq 0$ do
      $a_k \leftarrow \lfloor N_k / N_{k+1} \rfloor$
      $N_{k+2} \leftarrow N_k \bmod N_{k+1}$
      Increment $k$
  end while
  return $a_0, a_1, \dots, a_{k-1}$
```

For $k \geq 2$, we can prove the same recurrence relation by induction

$$\frac{p_k}{q_k} = \frac{a_k p_{k-1} + p_{k-2}}{a_k q_{k-1} + q_{k-2}} \implies \frac{p_{k+1}}{q_{k+1}} = \frac{(a_k + \frac{1}{a_{k+1}})p_{k-1} + p_{k-2}}{(a_k + \frac{1}{a_{k+1}})q_{k-1} + q_{k-2}} = \frac{a_{k+1}p_k + p_{k-1}}{a_{k+1}q_k + q_{k-1}}$$

The procedure (summarized in Algorithm 6.2) has the same complexity as the base Euclidean algorithm, as it simply iterates through the partial quotients, and the convergents can actually be computed along with the partial quotients.

**Remark 6.7.** What we present here is slightly different from what is usually known as the extended Euclidean algorithm. Traditionally the EEA is used to find two integers $g_1, g_2$ such that

$$g_1 p + g_2 q = \gcd(p, q)$$

which is known as *Bézout's identity* (see Exercise 6.1). In particular when $p < q$ and $\gcd(p, q) = 1$, the identity yields $g_1 p \equiv 1 \bmod q$. This is the standard way to invert elements in $\mathbb{Z}_q$, and we will make use of it as well.

**Algorithm 6.2** Extended Euclidean algorithm for convergents

```
Input: The partial quotients $\frac{p}{q} = [a_0; a_1, \dots, a_K]$.
Output: The factors of the $K + 1$ convergents of $\frac{p}{q}$.
  $p_0 \leftarrow a_0, q_0 \leftarrow 1$
  $p_1 \leftarrow a_0 a_1 + 1, q_1 \leftarrow a_1$
  for $k \in \{2, \dots, K\}$ do
      $p_k \leftarrow a_k p_{k-1} + p_{k-2}$
      $q_k \leftarrow a_k q_{k-1} + q_{k-2}$
  end for
  return $(p_0, q_0), (p_1, q_1), \dots, (p_K, q_K)$.
```

EEA is a very important (yet classical) component to make up for the imprecisions of the phase estimation circuit.

**Theorem 6.8** (Legendre's convergent theorem)**.** *Suppose to have a rational number $\frac{s}{r}$ and another rational number $x$ such that*

$$\left| x - \frac{s}{r} \right| \leq \frac{1}{2r^2}$$

*Then $s/r$ is a convergent of $x$.*

In other words, $(s, r)$ will appear in the list of convergent returned by the EEA. Actually, the returned fractions are always in lowest terms, so if $s \not\perp r$, the EEA will spit out $(s/\gcd(s,r), r/\gcd(s,r))$ instead of $(s, r)$.

*Proof.* By the condition above, the following holds for some $|\delta| \leq \frac{1}{2}$

$$x = \frac{s}{r} + \frac{\delta}{r^2} \ .$$

If $\delta = 0$ the claim is trivial as $s/r$ is certainly a convergent of itself, so let's assume $\delta \neq 0$. Let $\frac{r}{s} = [a_0; a_1, \dots, a_K]$ be a sequence of partial quotients for $\frac{r}{s}$. If we find a rational number $\lambda \geq 1$ such that

$$x = [a_0; a_1, \dots, a_K, \lambda] \quad \Longrightarrow \quad x = [a_0; a_1, \dots, a_K, b_0, \dots, b_{K'}]$$

where $\lambda = [b_0; b_1, \dots, b_{K'}]$. By using the recursive formula we thus need $\lambda$ satisfying:

$$x = \frac{\lambda p_K + p_{K-1}}{\lambda q_K + q_{K-1}} \ .$$

By subtracting $\frac{p_K}{q_K} = \frac{s}{r}$ on both sides and manipulating a bit the expression on the right-hand side we obtain

$$\frac{\delta}{2q_K^2} = x - \frac{p_K}{q_K} = \frac{-(p_K q_{K-1} - p_{K-1} q_K)}{q_K(\lambda q_K + q_{K-1})} = \frac{(-1)^K}{q_K(\lambda q_K + q_{K-1})}$$

where $p_K q_{K-1} - p_{K-1} q_K = (-1)^{K+1}$ by the determinant identity (see Exercise 6.1). By isolating $\lambda$ we obtain

$$\lambda = \frac{(-1)^K}{\delta} - \frac{q_{K-1}}{q_K}$$

We need $\lambda \geq 1$ so the signs of $(-1)^K$ and $\delta$ must match. If this is not the case, we take the other sequence for $\frac{s}{r}$ in order to change the parity of $K$. Since $q_{K-1}/q_K \leq 1$ and $(-1)^K/\delta \geq 2$, we get our desired $\lambda$. □

## 6.3 Period finding

Thanks to the very efficient quantum Fourier transform we outlined in the previous chapter, quantum computers are really good at finding *periods*.

**Problem 6.9** (Period finding)**.** We are given a function $f : \mathbb{Z} \to \mathbb{Z}_N$ as an oracle that is periodic in the following sense: there is a period $r \in \mathbb{Z}_N$ such that

(i) $f(x) = f(x + r)$ for all $x \in \mathbb{Z}$;

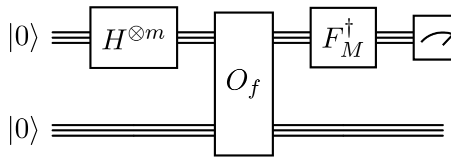
(ii) $f(x) \neq f(x + s)$ for all $x \in \mathbb{Z}, 0 < s < r$.

Output the period $r$.

Condition (ii) tells us that $f$ must be injective over the period, and this guarantees that $r$ is well-defined (notice that if $r$ satisfies (i) then so do $2r, 3r, \dots$, so $r$ would be the smallest such integer). Moreover, notice that $r \leq N$ since $f$ can take at most $N$ possible different values.

**Example 6.10.** The function $f(x) = x \bmod r$ has period $r$.

The quantum part for Shor's period finding algorithm is quite simple: we truncate the domain of $f$ to be $\mathbb{Z}_M$ (preferably we consider $M = 2^m$ to be a power of two), we prepare the equal superposition over these $M$ elements (unexpected, right?), apply the bit oracle for $f$, apply the inverse quantum Fourier transform on the input register and measure.



In order to better understand what is going on, we turn this procedure into what we refer to as the *Kitaev picture*: this circuit is very similar to phase estimation, except for the bit oracle. If we manage to see this as an application of the phase estimation circuit, then it will be much easier to analyze. From the definition of $f$ we define the $f$-shift matrix:

$$S_f |f(x)\rangle = |f(x+1)\rangle$$

This matrix will be a $N \times N$ matrix ($f$ is not necessarily surjective, so if there are elements of $\mathbb{Z}_N$ outside the actual range of $f$, we can simply leave them untouched).

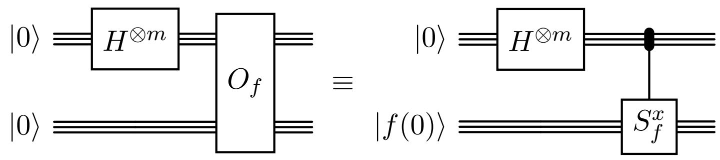
**Theorem 6.11.** *$S_f$ is unitary.*

*Proof.* Consider $S_f^\dagger S_f$: we only care about the space spanned by the elements in the range of $f$ (in the rest of the space $S_f$ already acts as the identity). Therefore, suppose that $0 \le x < y \le r$:

$$\langle f(x)| S_f^\dagger S_f |f(y)\rangle = \langle f(x+1)|f(y+1)\rangle$$

which is equal to zero since $x+1 \neq y+1 \bmod r$ and we have condition (ii). The case $x = y$ gives 1 trivially, so $S_f^\dagger S_f = I$. □

Now notice that if we start from $|f(0)\rangle$ then $S_f^x |f(0)\rangle = |f(x)\rangle$, thus one can check the two circuits below produce the exact same state



implying that the period finding algorithm can be just analyzed as a phase estimation algorithm applied to $S_f$, so the returned measurement outcome yields an estimation of one of the eigenvalues of $S_f$.

**Remark 6.12.** We didn't specify how to implement $S_f$, but we remark that we *don't* need to: we use the Kitaev picture just for the analysis. There are cases (including the two settings of this chapter) where $S_f$ is actually efficient and fast-forwardable, and it might be preferable to implement $S_f$ rather than $O_f$ in practice. Of course this requires to look into the implementation for $O_f$.

Now the question is: what are the eigenvalues of $S_f$?

**Theorem 6.13.** *Consider the following state:*

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi i k s/r} |f(k)\rangle$$

*For every $0 \le s < r$, $|u_s\rangle$ is an eigenstate of $S_f$ associated to the eigenvalue $e^{2\pi i s/r}$.*

*Proof.* Let's apply $S_f$ to the state:

$$\begin{aligned}
S_f |u_f\rangle &= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi iks/r} S_f |f(k)\rangle \\
&= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi iks/r} |f(k+1)\rangle \\
&= \frac{1}{\sqrt{r}} \sum_{k=1}^{r} e^{-2\pi i(k-1)s/r} |f(k)\rangle && \text{shift the index } k \\
&= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi i(k-1)s/r} |f(k)\rangle && \text{since } f(r) = f(0) \\
&= e^{2\pi is/r} \left( \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} e^{-2\pi iks/r} |f(k)\rangle \right)
\end{aligned}$$

so we get that $S_f |u_s\rangle = e^{2\pi is/r} |u_s\rangle$, as claimed. □

Therefore, if we assume for a minute that we input some $|u_s\rangle$ instead of $|f(0)\rangle$, the measurement outcome will be an integer $\ell \in \mathbb{Z}_M$ such that:

$$\left| \frac{\ell}{M} - \frac{s}{r} \right| \leq \frac{1}{2M} \tag{6.1}$$

with probability at least 0.8. This is great, because the eigenphase we are trying to approximate is actually carrying out the $r$ we are looking for. Instead of $|u_s\rangle$ (which we don't even know how to construct) we have $|f(0)\rangle$, but this is not really a problem.

**Lemma 6.14.** $|f(0)\rangle = \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle$.

*Proof.* Simplify the superposition

$$\begin{aligned}
\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle &= \frac{1}{r} \sum_{s=0}^{r-1} \sum_{k=0}^{r-1} e^{-2\pi iks/r} |f(k)\rangle \\
&= \sum_{k=0}^{r-1} \left( \frac{1}{r} \sum_{s=0}^{r-1} e^{-2\pi iks/r} \right) |f(k)\rangle
\end{aligned}$$

The sum in the tuples will be 1 if and only if $k = 0$, and zero otherwise, by the usual geometric sum argument. □

So by inputting $|f(0)\rangle$ we get a uniform superposition of all the eigenstates. If we imagine to have a perfect phase estimation, then we would have the transformation

$$|0\rangle |f(0)\rangle = \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |0\rangle |u_s\rangle \mapsto \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |s/r\rangle |u_s\rangle$$

and a measurement of the first register would return the rational $s/r$ for a uniformly random $s$. Handling the fact that phase estimation has limited precision is what slightly complicates the analysis. Let $\mathcal{L}_s$ be the set of integers $\ell$ satisfying (6.1) for $s/r$. As in the ideal case, a measurement of the first register is better understood as a probabilistic process: take $s$ uniformly at random, and conditioned on $s$, we will get an integer in $\mathcal{L}_s$ with probability $\geq 0.8$. We thus suppose a particular $s$ was chosen and $\ell \in \mathcal{L}_s$, which concludes the quantum part.

### 6.3.1 Classical post-processing

Since $\ell \in \mathcal{L}_s$ and we choose $M$ in the circuit to be the smallest power of two that is $\geq N^2$, we get an accuracy for the estimate

$$\left|\frac{\ell}{M} - \frac{s}{r}\right| \leq \frac{1}{2M} \leq \frac{1}{2N^2} \leq \frac{1}{2r^2}$$

satisfying the hypothesis of Legendre's theorem (Theorem 6.8). Therefore, if we now apply the extended Euclidean algorithm to $(\ell, M)$, one of the returned convergents will turn out to be $s/r$. First of all, how can we recognize this convergent in the list?

**Theorem 6.15.** *Let $\ell/M$ satisfy*

$$\left|\frac{\ell}{M} - \frac{s}{r}\right| \leq \frac{1}{2N^2} .$$

*Then the convergent $\frac{p_k}{q_k} = \frac{s}{r}$ is the unique one with $q_k < N$ satisfying the above condition.*

*Proof.* Suppose for a contradiction there is another convergent $\frac{p_{k'}}{q_{k'}} \neq \frac{p_k}{q_k}$ satisfying both conditions, then by a triangle inequality they must be $\leq 1/N^2$ apart. At the same time:

$$\left|\frac{p_{k'}}{q_{k'}} - \frac{p_k}{q_k}\right| = \frac{|p_k q_{k'} - p_{k'} q_k|}{q_k q_{k'}} > \frac{1}{N^2}$$

because $|p_k q_{k'} - p_{k'} q_k| \geq 1$ as the fractions are not equal. □

Therefore it is sufficient to return the highest $q_k < N$ from the convergent list. This will be $r$ if $s \in \mathbb{Z}_r^*$, otherwise $r/\gcd(s, r)$ will be returned. The full procedure is summarized in Algorithm 6.3.

**Algorithm 6.3** Shor's period finding algorithm

**Input:** A $r$-periodic function $f : \mathbb{Z} \to \mathbb{Z}_N$ given as an oracle.
**Output:** The period $r$.

```
$n = \lceil \log_2 N \rceil, M = 2^{2n}$                ▷ Smallest power of two $\geq N^2$
Allocate register $A$ with $2n$ qubits, register $B$ with $n$ qubits.
Apply $H^{\otimes 2n}$ to $A$
Query $f(0)$ and initialize $B$ to $|f(0)\rangle$
Apply the bit oracle $O_f$ to $A, B$
Apply $F_M^\dagger$ to $A$ and measure, giving $\ell \in \mathbb{Z}_M$.
$\{(p_k, q_k)\}_k \leftarrow$ EXTENDED-EUCLIDEAN$(\ell, M)$
return the highest $q_k$ satisfying $q_k < N$
```

The probability that $s \in \mathbb{Z}_r^*$ *and* phase estimation succeeds is

$$\Pr[\ell \in \mathcal{L}_s, s \in \mathbb{Z}_r^*] = \sum_{x \in \mathbb{Z}_r^*} \Pr[\ell \in \mathcal{L}_x \,|\, s = x] \cdot \Pr[s = x] \geq 0.8 \cdot \Omega\left(\frac{1}{\log\log r}\right)$$

and under this condition, the EEA will correctly retrieve $r$.

**Theorem 6.16.** *The period finding algorithm returns $r$ with probability $\Omega(1/\log\log r)$.*

Thus by repeating this procedure $\mathcal{O}(\log\log N)$ times and taking the minimum outcome $q$ that satisfies $f(0) = f(q)$, we get $r$ with constant probability.

**Theorem 6.17.** *By repeating Algorithm 6.3 $\mathcal{O}(\log n)$ times and taking the minimum valid period, the correct $r$ is returned with constant probability using $\mathcal{O}(n^2 \log n + T \log n)$ quantum gates plus $\mathcal{O}(n^3 \log n + T \log n)$ classical time, where $n = \lceil \log_2 N \rceil$ and $T$ is the time complexity of computing $f$ once.*

In particular, we only need $\mathcal{O}(\log n)$ queries to $f$ to solve period finding. The $n^2$ quantum time is dominated by the QFT (which could be brought down to $n \log n$ if we use the approximate version), while the classical time is dominated by the EEA, which takes $\mathcal{O}(n^3)$ bit operations.

The $\log n$ factor can be removed, however, effectively bringing the amount of queries to $f$ from $\mathcal{O}(\log n)$ to $\mathcal{O}(1)$: instead of repeating those $\mathcal{O}(\log \log N)$ times, we only run Algorithm 6.3 *twice* and take the least common multiple of the two outputs we get. We will get two outputs $(s_1, r_1), (s_2, r_2)$, both of which satisfy $r_j = r/\gcd(s_j, r_j)$. If $s_1, s_2$ turn out to be co-prime, then $\operatorname{lcm}(r_1, r_2) = r$ and the algorithm succeeds. We show that this happens with constant probability: let $P_j$ be the event of phase estimation succeeding in run $j$

$$\Pr[P_1, P_2, s_1 \perp s_2] = \Pr[P_1, P_2 \mid s_1 \perp s_2] \cdot \Pr[s_1 \perp s_2]$$

For any fixed $s_1$, $\Pr[P_1 \mid s_1] \geq 0.8$, and same for $P_2$, so the conditional probability will be $\geq (0.8)^2 = 0.64$ regardless of the values of $s_1, s_2$. We only need to lower bound the probability that two random integers in $\mathbb{Z}_r$ are co-prime:

$$\begin{aligned}
\Pr[s_1 \not\perp s_2] &\leq \sum_{k \text{ prime}} \Pr[k \parallel s_1, k \parallel s_2] && \text{union bound} \\
&= \sum_{k \text{ prime}} \Pr[k \parallel s_1] \cdot \Pr[k \parallel s_2] && s_1, s_2 \text{ are independent} \\
&\leq \sum_{k \text{ prime}} \left( \frac{r}{k} \cdot \frac{1}{r} \right)^2 && k \text{ divides} \leq \frac{r}{k} \text{ elements in } \mathbb{Z}_r \\
&\leq \sum_{k=2}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6} - 1 \leq 0.7
\end{aligned}$$

Therefore the probability that the algorithm succeeds would be $\geq 0.64 \cdot 0.3 \geq 0.15$.

**Theorem 6.18.** *By running Algorithm 6.3 twice and taking the least common multiple, the period finding algorithm returns the correct $r$ with constant probability with $\mathcal{O}(n^2 + T)$ quantum gates plus $\mathcal{O}(n^3 + T)$ classical time, where $n = \lceil \log_2 N \rceil$ and $T$ is the time complexity of computing $f$ once.*

## 6.4 Integer factorization via Miller's reduction

Understanding the period finding algorithm is the most involved part of Shor's algorithm. The last part is to use the period finding algorithm to tackle Factoring: considering an integer $N$, our goal is to output a non-trivial factor of it or, by using the notation introduced at the beginning of this chapter, a number $g \neq 1$ satisfying $\mathcal{I}(g) \subsetneq \mathcal{I}(N)$. In order to do this, we work on $\mathbb{Z}_N$, and define the *multiplicative order*.

**Definition 6.19.** The multiplicative order of a value $a \in \mathbb{Z}_N^*$ is the smallest $r > 0$ satisfying

$$a^r \equiv 1 \bmod N$$

This creates a cyclic pattern: $1 = a^0, a^1, a^2, \ldots, a^{r-1}, a^r = 1$.

**Example 6.20.** The multiplicative order of 3 in $\mathbb{Z}_{10}$ is 4: it is possible to check that

$$\begin{aligned}3^1 &= 3\\ 3^2 &= 9\\ 3^3 &= 27 \equiv 7 \bmod 10\\ 3^4 &= 81 \equiv 1 \bmod 10\end{aligned}$$

The *order finding* problem is about finding the multiplicative order of a given $a \in \mathbb{Z}_N$, and it's not hard to see it as an application of period finding on the function $f(x) = a^x \bmod N$. If we can solve order finding, we can use *Miller's reduction* to solve factoring: we choose a uniformly random $a \in \mathbb{Z}_N$ (see Exercise 4.6 point (i) to see how to sample from $\mathbb{Z}_N$): if $\gcd(a, N) > 1$, then $a$ is already a non-trivial factor of $N$ and we return it. Otherwise, by symmetry $a$ will be uniformly random in $\mathbb{Z}_N^*$. We compute the order $r$ of $a$ in $\mathbb{Z}_N$ using period finding.

The idea now is to decompose $a^r - 1 = (a^{r/2} - 1)(a^{r/2} + 1)$. By the definition of multiplicative order we get that

$$a^r \equiv 1 \bmod N \qquad \Longrightarrow \qquad (a^{r/2} - 1)(a^{r/2} + 1) \equiv 0 \bmod N$$

In other words, $N$ divides this product, which thus must contain all the factors:

$$\mathcal{I}((a^{r/2} - 1)(a^{r/2} + 1)) \supseteq \mathcal{I}(N)$$

On the other hand $(a^{r/2} - 1) \neq 0 \bmod N$, (otherwise $r$ wouldn't be the order of $a$), i.e., this number cannot contain all the factors of $N$ by itself

$$\mathcal{I}(a^{r/2} - 1) \not\supseteq \mathcal{I}(N)$$

If the same holds also for $a^{r/2} + 1$, then none of the two numbers of the product can hold *all* the factors of $N$, but at the same time they must have them all when combined together. Thus they must both contain different non-trivial subsets of the factors, plus other garbage that we don't care about. The garbage is removed by taking a greatest common divisor with $N$, which only takes the interesting stuff, e.g., $g = \gcd(a^{r/2} - 1, N)$ will be a non-trivial factor of $N$. We thus need two things to happen: (1) $r$ must be even (for $r/2$ to even make sense), and (2) $a^{r/2} + 1 \not\equiv 0 \bmod N$ as well.

**Theorem 6.21.** *Let $N$ be odd and have at least two distinct prime factors (i.e., $N \neq p^k$). If $a \in \mathbb{Z}_N^*$ is chosen uniformly at random, then with probability at least $1/2$ we have that*

*(i) $r$ is even, and*

*(ii) $a^{r/2} + 1 \not\equiv 0 \bmod N$.*

*Proof.* Suppose that $N = p_1 \cdots p_m$ where the $p_j$'s are powers of different primes. Taking a random $a \in \mathbb{Z}_N^*$ is equivalent to taking random $a_j \in \mathbb{Z}_{p_j}$ for $1 \leq j \leq m$, and then finding the unique $a$ satisfying

$$a \equiv a_j \bmod p_j$$

for every $j$ (this is a consequence of the *Chinese Remainder Theorem*). If $r_j$ is the order of $a_j$ modulo $p_j$, then $r = \operatorname{lcm}(r_1, \ldots, r_m)$. Moreover, let $c_j$ be the largest number such that $2^{c_j}$ divides $r_j$ (in other words $c_j$ counts how many twos appear in the factorization of $r_j$).

We claim that if (i) or (ii) fail, then $c_1 = \cdots = c_m$ necessarily. Clearly if $r$ is odd then all $r_j$ are odd, i.e., $c_j = 0$ for all $j$. If $a^{r/2} \equiv -1 \bmod N$ then the same is true modulo each $p_j$, implying that each $r_j$ does not divide $r/2$. At the same time all of them divide $r$, i.e., all the $r_j$ have the same number of twos in the factorizations, the same that appear in $r$.

We conclude the proof by showing that $\Pr[c_1 = x] \leq \frac{1}{2}$ for any $x$ (in particular this would be true for $c_1 = c_2$, necessary for the conditions to fail). *[In progress, trying to prove it without invoking half of group theory].* □

Thus if we can guarantee that $N$ is odd and contains at least two distinct primes in its factorization, then we're good to go, and these two cases are extremely easy to exclude algorithmically.

**Algorithm 6.4** Shor's factoring algorithm via Miller's reduction

```
Input: A $n$-bit composite number $N$.
Output: A non-trivial factor of $N$.
  if $N$ is even then return 2.
  for $k \in \{1, 2, \dots, n\}$ do
      if $\sqrt[k]{N}$ is integer then return $\sqrt[k]{N}$.        ▷ This occurs when $N = p^k$.
  end for
  $a \leftarrow$ random element of $\mathbb{Z}_N$.
  if $\gcd(a, N) > 1$ then return $\gcd(a, N)$
  $r \leftarrow$ PeriodFinding($x \mapsto a^x \bmod N$)
  if $r$ is odd then return failure.
  return $g = \gcd(a^{r/2} - 1, N)$
```

We have seen that the period finding subroutine devised in this chapter will succeed with probability $\geq 0.15$ regardless of $a$ and $N$. Conditioned on the event that $\gcd(a, N) \neq 1$ (the other case being negligibly unlikely), we have that $a$ is uniformly random in $\mathbb{Z}_N^*$. If $F$ is the event of period finding succeeding, and $G$ is the event of the conditions of Theorem 6.21 being true, then

$$\Pr[F, G] = \frac{1}{N} \sum_{a \in \mathbb{Z}_N^*} \Pr[F, G \,|\, a]$$

When we fix $a$, $G$ is completely determined, and it is determined by at least $\frac{1}{2}$ of the values of $a$, by Theorem 6.21, therefore,

$$\Pr[F, G] \geq \frac{1}{2N} \sum_{a \in \mathbb{Z}_N^*} \Pr[F \,|\, a] = \frac{1}{2} \cdot \Pr[F] \geq 0.07$$

Note that there is a (small) probability that $\gcd(a, N) > 1$, in which case the algorithm succeeds with certainty, and this only pulls up our overall success probability (by not really much, though).

The final time complexity of Shor's algorithm (in particular, the quantum part) just depends on how we compute the modular exponentiation $f(x) = a^x \bmod N$ (remember that the quantum time complexity of implementing $O_f$ only depends on its classical one). The best complexity known so far for modular exponentiation is $\mathcal{O}(n^2 \log n \log\log n)$, based on the *Schönage-Strassen* multiplication procedure.

**Theorem 6.22.** *Algorithm 6.4 outputs a non-trivial factor of $N$ with constant probability by using $\mathcal{O}(n^2 \log n \log\log n)$ quantum gates and $\mathcal{O}(n^3)$ classical time.*

**Remark 6.23.** The best known classical algorithm for factoring, known as the *number field sieve*, runs in $\exp\Big(\mathcal{O}(n^{1/3}\log^{1/2} n)\Big)$. The super-polynomial speed-up is not confirmed, though, as we don't know whether this is actually the best possible algorithm.

**Remark 6.24.** While the $\mathcal{O}(n^3)$ classical time is not a big deal, being able to coherently realize $O_f$ is the actual challenge of implementing the factoring algorithm.

**Remark 6.25.** The particular $f$ we designed for the factoring algorithm has a very simple shift matrix in the Kitaev picture:

$$S_f\,|x\rangle = |ax \bmod N\rangle$$

i.e., $S_f$ simply multiplies the input by $a$. Moreover, this $S_f$ is fast-forwardable:

$$S_f^{2^k}\,|x\rangle = |a^{2^k}x \bmod N\rangle$$

i.e., the circuit is the same, but if one (classically) precomputes all the $a^{2^k}$ and builds the multiplication circuits, it might be convenient to implement the factoring algorithm directly as a phase estimation on $S_f$ instead of building $O_f$.

## 6.5 Computing discrete logarithms

Let $G = \{1, g, g^2, \dots, g^{q-1}\}$ be a cyclic group generated by an element $g$ of order $q$ equipped with a multiplication operation. We will assume that the elements of $G$ are easily encoded in some way, and that group operations (multiplication, inversion) are efficiently computable. We will talk about groups better in the next chapter, but for now we have all we need to talk about the discrete logarithm problem.

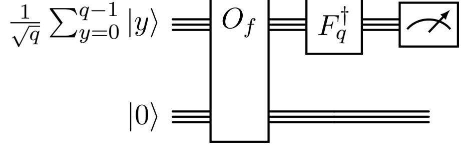
**Problem 6.26** (Discrete logarithm)**.** Given a cyclic group $G$, a generator $g$, and some element $h = g^\ell$, output $\ell$.

We are going to show how to use the same idea of the period finding algorithm to solve discrete logarithm. What's surprising is that this algorithm is much simpler than the factoring algorithm presented in the previous section. The idea is to take the function $f : \mathbb{Z}_q^2 \to G$

$$f(x,y) = g^x h^{-y}$$

and computing its period, i.e., that pair $(r_1, r_2)$ such that $f(0,0) = f(r_1,r_2)$ and $f(0,0) \neq f(s_1,s_2)$ for $0 < s_j < r_j$, and note that $(\ell, 1)$ is actually the period of this function so this is all we need to do. However, we should redefine and redevise period finding for a two-dimensional function, and we don't really want to go through that pain again. A huge advantage over factoring is that here we actually have the order $q$ of the group, which will also be the order of the QFT, so we don't need to complicate our lives to account for an imprecise phase estimation.

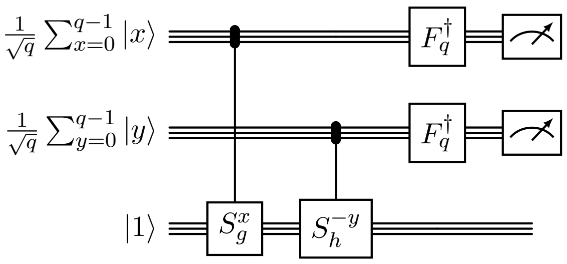
The complete quantum circuit for the discrete logarithm is


$\frac{1}{\sqrt{q}}\sum_{x=0}^{q-1}|x\rangle$
$\frac{1}{\sqrt{q}}\sum_{y=0}^{q-1}|y\rangle$
$|0\rangle$
$O_f$
$F_q^\dagger$
$F_q^\dagger$

We can analyze what happens directly by computing the state obtained by this circuit, but it's probably more elegant to view it in the Kitaev picture: let's define two shift matrices

$$S_g |x\rangle = |g \cdot x\rangle$$
$$S_h |x\rangle = |h \cdot x\rangle \ .$$

If we start from the $|1\rangle$ state, $S_g^x S_h^{-y} |1\rangle = |g^x h^{-y}\rangle = |f(x, y)\rangle$. Therefore, the above circuit is equivalent to


$\frac{1}{\sqrt{q}} \sum_{x=0}^{q-1} |x\rangle$
$\frac{1}{\sqrt{q}} \sum_{y=0}^{q-1} |y\rangle$
$|1\rangle$
$F_q^\dagger$
$F_q^\dagger$
$S_g^x$
$S_h^{-y}$

which is nothing more than two phase estimations executed back to back on the matrices $S_g, S_h^\dagger$. Therefore the above circuit is simply trying to estimate the eigenvalues of these two matrices.

Notice that $S_h = S_g^\ell$, so this already tells us that (1) they have the same eigenstates and (2) the eigenphases of these two matrices are equal up to the $\ell$ factor we would like to learn.

The shift matrix $S_g$ can be effectively seen as the one for the function $f_g(x) = g^x$, and a (non-necessarily minimal) period for this function is the order $q$, so the eigenvalues of $S_g$ are $e^{2\pi i s/q}$ by Theorem 6.13, with the eigenstates $|u_s\rangle$ given by the same claim. Moreover, $|1\rangle$ is the equal superposition of the $|u_s\rangle$ by Lemma 6.14, and since phase estimation is in $\mathbb{Z}_q$, it will be perfect, and the first register will output a uniformly random $s \in \mathbb{Z}_q$ upon measurement.

On the other hand, if $S_g |u_s\rangle = e^{2\pi i s/q} |u_s\rangle$, then $S_h^\dagger |u_s\rangle = e^{-2\pi i s\ell/q} |u_s\rangle$, so the second phase estimation will return $-s\ell$ ($\equiv s(q - \ell)$ since we are in $\mathbb{Z}_q$). In other words,

$$|0\rangle |0\rangle |u_s\rangle \mapsto |s\rangle |-s\ell\rangle |u_s\rangle$$
$$|0\rangle |0\rangle |1\rangle \mapsto \frac{1}{\sqrt{q}} \sum_{s=0}^{q-1} |s\rangle |-s\ell\rangle |u_s\rangle$$

i.e., the measurement outcome will be $(x, y) = (s, -s\ell) \in \mathbb{Z}_q^2$, for a uniformly random $s \in \mathbb{Z}_q$. To retrieve $\ell$ we just hope that the $s$ we get is actually in $\mathbb{Z}_q^*$ (and we know it happens with constant probability in $\mathcal{O}(\log \log q)$ trials), and we're done by computing $\ell = -x^{-1}y$.

As in the factoring algorithm we can do only two trials, get $(s_1, -s_1\ell), (s_2, -s_2\ell)$ and return $\gcd(s_1\ell, s_2\ell)$. As we've seen, with probability $\geq 0.3$, we will have that $s_1 \perp s_2$ and the result will be correct.

**Theorem 6.27.** *Algorithm 6.5 correctly returns the discrete logarithm with probability* $\geq 0.3$ *using* $\mathcal{O}(T \log q + \log^2 q)$ *quantum gates, plus* $\mathcal{O}(\log^3 q)$ *classical time, where* $T$ *is the time needed to carry out one group operation.*

Here $T$ really depends on the group we use to instantiate the discrete logarithm (e.g., $\mathbb{Z}_p^*$ or elliptic curves), but it's important to remark that Shor's algorithm solves them all in polynomial time, as long as the group operations are also constructible in polynomial time (a quite plausible assumption). In other words Shor's algorithm is *group-agnostic*. Moreover, even if we just considered group operations to be oracles, and the encoding of the group elements were 'random' (without a specific structure), then Shor's algorithm would work anyway (it is *encoding-agnostic*).

We do know something more when it comes to discrete logarithms.

**Algorithm 6.5** Shor's algorithm for the discrete logarithm

```
Input: The description of a cyclic group $G = \langle g \rangle$ of order $q$, and an element $h = g^\ell \in G$.
Output: The discrete logarithm $\ell$ of $h$.
  for $j \in \{0,1\}$ do
      Initialize quantum register $C$ to the state $|1\rangle$.
      Apply phase estimation over $\mathbb{Z}_q$ on register $C$ for the unitary $S_g$.
      Measure the estimated phase, giving $x_j$.
      Apply phase estimation over $\mathbb{Z}_q$ on register $C$ for the unitary $S_h^\dagger$.
      Measure the estimated phase, giving $y_j$.
  end for
  return $\gcd(q - y_0, q - y_1)$
```

**Theorem 6.28** (Shoup [12]). *Any encoding-agnostic classical algorithm for the discrete logarithm requires $\Omega(\sqrt{q})$ group operations.*

In other words, if we defined the discrete logarithm adequately as a query problem (e.g., similar to unstructured search), then an exponential speed-up is proven!

**Remark 6.29.** The best known classical algorithms for the discrete logarithm over general groups run in $\mathcal{O}(\sqrt{q})$ classical time. In $\mathbb{Z}_p^*$, the number field sieve (similar to the one for factoring) has the same $\exp\Big(\mathcal{O}(n^{1/3}\log^{1/2} n)\Big)$ complexity.

**Remark 6.30.** We used the QFT $F_q$ here, for $q$ non-necessarily a power of two, which we assumed exact for simplicity. We can either construct $F_q$ approximately with the method described in the previous chapter, or we could take $F_M$ for $M \geq q^2$ and use continued fractions, since we only need to measure after the QFT.

## 6.6 Exercises

**Exercise 6.1.** Here we show how to use the convergents to satisfy Bézout's identity.

(i) Show that the convergents returned by the EEA remain unchanged if we multiply $p, q$ be a common factor $g$.

(ii) Show by induction that the convergents satisfy the determinant identity

$$p_n q_{n-1} - q_n p_{n-1} = (-1)^{n+1}$$

(iii) Suppose $\gcd(p,q) = 1$. Then $(p_K, q_K) = (p,q)$. Show how to use the determinant identity to find two integers satisfying Bézout's identity.

(iv) Extend (iii) to the case where $\gcd(p,q) > 1$.

(v) Now suppose that $p < q$ and $\gcd(p,q) = 1$. Show how to compute the inverse of $p$ in $\mathbb{Z}_q$ using the convergents of $p/q$.

(vi) Point (v) proves that $p$ admits an inverse in $\mathbb{Z}_q$ if $p \in \mathbb{Z}_q^*$. Use Bézout's identity to prove the converse: if $p$ admits an inverse in $\mathbb{Z}_q$, $p \in \mathbb{Z}_q^*$.

# Bibliography

[1] Michael A. Nielsen and Isaac L. Chuang. "Quantum computation and quantum information: 10th anniversary edition". Cambridge University Press. Cambridge (2010). url: `https://doi.org/10.1017/CBO9780511976667`.

[2] Ronald de Wolf. "Quantum Computing: Lecture Notes" (2019). arXiv:1907.09415.

[3] Andrew M Childs. "Lecture Notes on Quantum Algorithms".

[4] Adam Sawicki and Katarzyna Karnas. "Universality of single qudit gates". Annales Henri Poincaré **18**, 3515–3552 (2017). arXiv:1609.05780.

[5] Adam Bouland and Tudor Giurgica-Tiron. "Efficient Universal Quantum Compilation: An Inverse-free Solovay-Kitaev Algorithm" (2021). arXiv:2112.02040.

[6] Lov K. Grover. "A Fast Quantum Mechanical Algorithm for Database Search". In Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing. Pages 212–219. Association for Computing Machinery (1996).

[7] Lov K Grover. "Quantum Mechanics helps in searching for a needle in a haystack". Physical Review Letters **79**, 325–328 (1997).

[8] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. "Quantum Amplitude Amplification and Estimation". Quantum Computation and Information **305**, 53–74 (2002).

[9] Michel Boyer, Gilles Brassard, Peter Hoeyer, and Alain Tapp. "Tight bounds on quantum searching". Fortschritte der Physik **46**, 493–505 (1998). arXiv:quant-ph/9605034.

[10] James W. Cooley and John W. Tukey. "An algorithm for the machine calculation of complex Fourier series". Mathematics of Computation **19**, 297–301 (1965).

[11] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". SIAM Journal on Computing **26**, 1484–1509 (1997).

[12] Victor Shoup. "Lower Bounds for Discrete Logarithms and Related Problems". In Walter Fumy, editor, Advances in Cryptology — EUROCRYPT '97. Pages 256–266. Berlin, Heidelberg (1997). Springer.