Notes on Quantum Computing

Ryan Joo

Last updated: August 31, 2026

Contents

1 Introduction

1.1 Postulates of quantum mechanics

Postulate 1.1. Associated to any isolated physical system is a complex vector space with inner product (i.e., a Hilbert space) known as the **state space** of the system. The system is completely described by its **state vector**, which is a unit vector in the system's state space.

A **classical bit** is an object that can be either 0 or 1. A **quantum bit** (or **qubit** for short) is the quantum version of a bit, which has a two-dimensional state space; let $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ be orthonormal basis for the space. Then a qubit is a linear combination of $|0\rangle$ and $|1\rangle$:

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle = \alpha_0 \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \alpha_1 \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \in \mathbb{C}^2$$

for some $\alpha_0, \alpha_1 \in \mathbb{C}$, called **amplitudes** for the states $|0\rangle$ and $|1\rangle$ respectively. Since $|\psi\rangle$ is a unit vector in $\mathbb{C}^2$, we have $\langle\psi|\psi\rangle = 1$, so $|\alpha_0|^2 + |\alpha_1|^2 = 1$.
Geometrically, a single qubit in the state $a|0\rangle + b|1\rangle$ can be visualised as a point (θ, ϕ) on the unit sphere, where

$$a = \cos\frac{\theta}{2}, \quad b = e^{i\phi}\sin\frac{\theta}{2}.$$

This is called the **Bloch sphere** representation, and the vector $(\cos\phi\sin\theta, \sin\phi\sin\theta, \cos\theta)$ is called the Bloch vector. The angle ϕ is called the **phase** of $|\psi\rangle$.

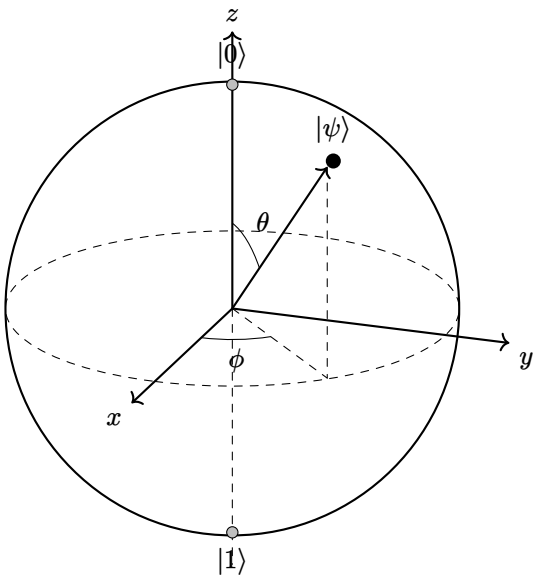


Figure 1.1: Bloch sphere representation of a qubit.

More generally, any state $|\psi\rangle \in \mathbb{C}^n$ can be expressed in the standard basis as follows:

$$|\psi\rangle = \begin{pmatrix} \alpha_0 \\ \vdots \\ \alpha_{n-1} \end{pmatrix} = \sum_{i=0}^{n-1} \alpha_i |i\rangle = \sum_{i=0}^{n-1} \langle i|\psi\rangle |i\rangle ,$$

where $\alpha_i = \langle i|\psi\rangle$ is the i-th coordinate of $|\psi\rangle$ (in the standard basis) satisfying $\sum_{i=0}^{n-1} |\alpha_i|^2 = 1$. A linear combination $\sum_i \alpha_i |i\rangle$ is called a **superposition** of states with **amplitude** α_i for the state $|i\rangle$.

More generally, if $\{|u_1\rangle, \ldots, |u_n\rangle\}$ is an orthonormal basis of $\mathbb{C}^n$, we can express $|\psi\rangle$ in this basis as

$$|\psi\rangle = \sum_{i=1}^{n} \langle u_i|\psi\rangle \, |u_i\rangle ,$$

where $|u_i|\psi\rangle$ is the i-th coordinate of $|\psi\rangle$ in the basis $\{|u_1\rangle, \ldots, |u_n\rangle\}$.
We will also often use the states

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

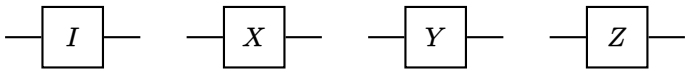
Consider performing operations on a quantum state $|\psi\rangle$. Since $|\psi\rangle$ is a vector of norm 1, we want operations to be norm-preserving, so that the resulting vector $|\psi'\rangle$ still has norm 1, so it is a valid quantum state. We also want operations to be linear. Hence, operations on quantum states are *unitary*.

Postulate 1.2. The evolution of a closed quantum system is described by a unitary transformation. That is, the state $|\psi\rangle$ of the system at time t_0 is related to the state $|\psi'\rangle$ of the system at time t_1 by a unitary operator U which depends only on the times t_1 and t_2:

$$|\psi'\rangle = U \, |\psi\rangle .$$

Remark. A *closed* system is one that does not interact in any way with other systems.

A particularly useful set of one-qubit unitaries are the **Pauli gates**:



- The I gate: $I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $I\,|0\rangle = |0\rangle$, $I\,|1\rangle = |1\rangle$
- The X gate: $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, $X\,|0\rangle = |1\rangle$, $X\,|1\rangle = |0\rangle$

  Geometrically, the X gate flips the cubic π radians around the x-axis on the Bloch sphere.
- The Y gate: $Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$, $Y\,|0\rangle = i\,|1\rangle$, $Y\,|1\rangle = -i\,|0\rangle$

  Geometrically, the Y gate flips the cubic π radians around the y-axis on the Bloch sphere.
- The Z gate: $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$, $Z\,|0\rangle = |0\rangle$, $Z\,|1\rangle = |-1\rangle$

  Geometrically, the Z gate flips the cubic π radians around the z-axis on the Bloch sphere.

Example. Let $|\psi\rangle = \frac{\sqrt{3}}{2}\,|0\rangle + \frac{1}{2}\,|1\rangle$. Then $Y\,|\psi\rangle = \frac{\sqrt{3}}{2}Y\,|0\rangle + \frac{1}{2}Y\,|0\rangle = \frac{\sqrt{3}}{2}i\,|1\rangle - \frac{1}{2}i\,|0\rangle$.

Since the X, Y and Z gates rotate around the specified axis π radians, if we apply the same gate twice, we rotate around 2π radians, so the qubit ends up in the original state. This means the X, Y and Z gates are their own inverses, so they are unitary matrices.
The Pauli matrices give rise to three useful classes of unitary matrices when they are exponentiated, the

rotation operators about the x, y, and z-axes:

$$R_x(\theta) := e^{-i\theta X/2} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}X = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$$

$$R_y(\theta) := e^{-i\theta Y/2} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}Y = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$$

$$R_z(\theta) := e^{-i\theta Z/2} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}Z = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}$$

There are also some other quantum gates:

- The Hadamard gate $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$, $H|0\rangle = |+\rangle$, $H|1\rangle = |-\rangle$

  The Hadamard gate is frequently used to create superpositions at the beginning of quantum algorithms.

- The phase gate $S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$ adds a relative phase of $\frac{\pi}{2}$ radians

- The $\pi/8$ gate $T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$ adds a relative phase of $\frac{\pi}{4}$ radians

Postulate 1.3. Quantum measurements are described by a collection $\{M_m\}$ of **measurement operators**. These are operators acting on the state space of the system being measured. The index m refers to the measurement outcomes that may occur in the experiment.
If the state of the quantum system is $|\psi\rangle$ immediately before the measurement, the probability that outcome m occurs is

$$p(m) = \langle\psi|{M_m}^\dagger M_m|\psi\rangle$$

and the state of the system after measurement is

$$\frac{M_m|\psi\rangle}{\sqrt{\langle\psi|{M_m}^\dagger M_n|\psi\rangle}}.$$

The measurement operators satisfy the **completeness equation**

$$\sum_m {M_m}^\dagger M_m = I.$$

The completeness equation expresses the fact that probabilities sum to one:

$$\sum_m p(m) = \sum_m \langle\psi|{M_m}^\dagger M_m|\psi\rangle = \langle\psi|\left(\sum_m {M_m}^\dagger M_m\right)|\psi\rangle = \langle\psi|I_n|\psi\rangle = \langle\psi|\psi\rangle = 1.$$

Consider the measurement of a qubit in the computational basis. This is a measurement on a single qubit with two outcomes defined by the measurement operators $M_0 = |0\rangle\langle 0|$ and $M_1 = |1\rangle\langle 1|$:

$$M_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}\begin{pmatrix} 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

$$M_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}\begin{pmatrix} 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

Observe that each measurement operator is Hermitian, i.e., ${M_0}^2 = M_0$, ${M_1}^2 = M_1$. Thus,

$$M_0{}^\dagger M_0 + M_1{}^\dagger M_1 = M_0 + M_1 = I,$$

so the completeness equation is satisfied.

Let $|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$ be the state measured. The probability of obtaining measurement outcome 0 is

$$p(0) = \langle\psi|M_0{}^\dagger M_0|\psi\rangle = \langle\psi|M_0|\psi\rangle = \begin{pmatrix}\alpha_0^* & \alpha_1^*\end{pmatrix}\begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix}\begin{pmatrix}\alpha_0\\ \alpha_1\end{pmatrix} = |\alpha_0|^2.$$

Similarly, the probability of obtaining measurement outcome 1 is $p(1) = |\alpha_1|^2$.

Hence, the state after measurement in the two cases is

$$\frac{M_0|\psi\rangle}{|\alpha_0|} = \frac{\alpha_0}{|\alpha_0|}|0\rangle, \quad \frac{M_1|\psi\rangle}{|\alpha_1|} = \frac{\alpha_1}{|\alpha_1|}|1\rangle.$$

More generally, given $|\psi\rangle = \sum_i \alpha_i |i\rangle$, the probability of seeing $|i\rangle$ is $|\alpha_i|^2$, and the post-measurement state is $|i\rangle$.

Remark. After the measurement, the system is in the measured state, so repeating the measurement will always yield the same value. We say that the measurement causes a "collapse" to the observed outcome.

Example. Let $|\psi\rangle = \frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle$. Then $p(0) = \frac{3}{4}$, $p(1) = \frac{1}{4}$.

We can write any one-qubit state as

$$|\psi\rangle = e^{i\theta}(\alpha|0\rangle + \beta e^{i\phi}|1\rangle) = e^{i\theta}|\psi'\rangle,$$

where α and β are positive real numbers. θ is known as the **global phase**, and has no observable consequences because

$$U|\psi\rangle = Ue^{i\theta}(\alpha|0\rangle + \beta e^{i\phi}|1\rangle) = e^{i\theta}U(\alpha|0\rangle + \beta e^{i\phi}|1\rangle) = e^{i\theta}U|\psi'\rangle$$

and for any measurement operator P_m,

$$\langle\psi|P_m^\dagger P_m|\psi\rangle = \langle\psi'|e^{-i\theta}P_m^\dagger P_m e^{i\theta}|\psi'\rangle = \langle\psi'|P_m^\dagger P_m|\psi'\rangle$$

where we used the fact that $(e^{i\theta}|\psi'\rangle)^\dagger = \langle\psi'|e^{-i\theta}$. Thus, we typically neglect global phase, since it has no effect on our computations.

We focus on the following special class of measurements.

Definition 1.4. A projective measurement is described by an observable M, which is a Hermitian operator. The observable has a spectral decomposition

$$M = \sum_m mP_m,$$

where P_m is the projector onto the eigenspace of M with eigenvalue m. The possible outcomes of the measurement correspond to the eigenvalues, m, of the observable.

Upon measuring the state $|\psi\rangle$, the probability of getting outcome m is

$$p(m) = \langle\psi|P_m|\psi\rangle,$$

and the state of the quantum system after the measurement is

$$\frac{P_m|\psi\rangle}{\sqrt{p(m)}}$$

where we normalised the projection to make it a unit vector.

Example. Consider the observable $M = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$ for a qubit. This matrix has two eigenvectors:

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \text{ with eigenvalue 3, and } \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix} \text{ with eigenvalue 1.}$$

This means we can express M as a linear combination of projectors:

$$\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} = 3 \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} + \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix} = 3 \left|+\right\rangle \left\langle+\right| + \left|-\right\rangle \left\langle-\right| .$$

Theorem 1.5. *Given a state $|\psi\rangle$ and an observable M, the expectation value of the measurement is*

$$\mathbb{E}[M] = \langle\psi|M|\psi\rangle .$$

Proof.

$$\mathbb{E}[M] = \sum_m mp(m) = \sum_m \langle\psi|P_m|\psi\rangle = \langle\psi| \left(\sum_m mP_m \right) |\psi\rangle = \langle\psi|M|\psi\rangle .$$

□

Example. Let $|\psi\rangle = \frac{3}{5}|0\rangle + \frac{4}{5}|1\rangle$. The observable $M = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$ measures it in the basis $|+\rangle, |-\rangle$, and returns the value 3 for the outcome $|+\rangle$ and the value 1 for the outcome $|-\rangle$.
By the theorem, the expectation value of the observable is

$$\langle\psi|M|\psi\rangle = \begin{pmatrix} \frac{3}{5} & \frac{4}{5} \end{pmatrix} \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} \frac{3}{5} \\ \frac{4}{5} \end{pmatrix} = \frac{74}{25}.$$

Postulate 1.6. The state space of a composite physical system is the tensor product of the state spaces of the component physical systems. If we have systems numbered $1, \ldots, n$, and system number i is prepared in the state $|\psi_i\rangle$, then the joint state of the total system is

$$|\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle$$

which lives in an n-qubit system, i.e., the n-fold tensor product space

$$C^{2^n} \cong (\mathbb{C}^2)^{\otimes n} := \mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2.$$

Standard basis notation for the joint system: $|i\rangle \otimes |j\rangle = |ij\rangle$. For example, for 2 qubits,

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

and similarly

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

Thus, a quantum state of 2 qubits has the form

$$\alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle = \begin{pmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{pmatrix}.$$

Note that the ordering of $\alpha_{00}, \alpha_{01}, \alpha_{10}, \alpha_{11}$ in the vector follows the lexicographical order of the bit strings. Generalising, a n-qubit state has the form

$$\sum_{i \in \{0,1\}^n} \alpha_i |i\rangle .$$

Remark. We usually identify the set of n-bit strings $\{0,1\}^n$ with the integers $\{0, \dots, 2^n - 1\}$.

Definition 1.7. A state $|\Psi\rangle \in \mathbb{C}^n \otimes \mathbb{C}^m$ of a combined system is **product** if it can be factored into

$$|\Psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$$

for some $|\psi_1\rangle \in \mathbb{C}^n$, $|\psi_2\rangle \in \mathbb{C}^m$. Otherwise, it is called **entangled**.

The significance of entangled states is: By measuring one of the qubits, we know the state of the other qubit.

Example. This two-qubit state is a product state:

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

Example. Neither of the following two-qubit states can be written as a product of single-qubit states, so they are both entangled:

$$\frac{1}{\sqrt{2}}(|10\rangle + |01\rangle), \quad \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

Theorem 1.8 (No-cloning theorem)**.** *There does not exist a 2-qubit unitary U that maps*

$$|\psi\rangle |0\rangle \mapsto |\psi\rangle |\psi\rangle$$

for every qubit $|\psi\rangle$.

Proof. For a contradiction, suppose there exists such a unitary operator U. Then for any $|\psi\rangle, |\phi\rangle \in \mathbb{C}^2$,

$$U(|\psi\rangle |0\rangle) = |\psi\rangle |\psi\rangle , \quad U(|\phi\rangle |0\rangle) = |\phi\rangle |\phi\rangle .$$

Taking the inner product of these two equations,

$$\begin{aligned} \text{LHS} &= \langle\psi|\langle 0|U^\dagger U|\phi\rangle|0\rangle = \langle\psi|\phi\rangle\langle 0|0\rangle = \langle\psi|\phi\rangle \\ \text{RHS} &= (\langle\psi|\langle\psi|)(|\phi\rangle|\phi\rangle) = \langle\psi|\phi\rangle\langle\psi|\phi\rangle = (\langle\psi|\phi\rangle)^2 \end{aligned}$$

Thus, $\langle\psi|\phi\rangle = (\langle\psi|\phi\rangle)^2$. Hence, $|\psi\rangle = |\phi\rangle$, or $|\psi\rangle$ and $|\phi\rangle$ are orthogonal.
Hence, a cloning device can only clone states which are orthogonal to one another, and therefore a general quantum cloning device is impossible. □

Corollary 1.9 (No-deleting principle)**.** *There does not exist a 2-qubit unitary $\tilde{U}$ that maps*

$$|\psi\rangle |\psi\rangle \mapsto |\psi\rangle |0\rangle$$

for every qubit $|\psi\rangle$.

1.2 Quantum teleportation

Superdense coding is a quantum protocol that allows us to send multiple bits of classical information using 1 qubit. Suppose Alice has a qubit $\alpha_0 |0\rangle + \alpha_1 |1\rangle$ that she wants to send to Bob via a *classical* channel. Without further resources, this would be impossible, because the amplitudes α_0 and α_1 may require an infinite number of bits of precision to write them down exactly.

(1) Alice shares an EPR-pair with Bob:

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

Alice holds the first qubit, and Bob holds the second qubit. Initially, their joint state is

$$(\alpha_0 |0\rangle + \alpha_1 |1\rangle) \otimes \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

The first two qubits belong to Alice, the third to Bob.

(2) Alice performs a CNOT on her two qubits and then a Hadamard transform on her first qubit. Their joint 3-qubit state is

$$\begin{aligned}
&\frac{1}{2} |00\rangle (\alpha_0 |0\rangle + \alpha_1 |1\rangle) + \\
&\frac{1}{2} |01\rangle (\alpha_0 |1\rangle + \alpha_1 |0\rangle) + \\
&\frac{1}{2} |10\rangle (\alpha_0 |0\rangle - \alpha_1 |1\rangle) + \\
&\frac{1}{2} \underbrace{|11\rangle}_{\text{Alice}} \underbrace{(\alpha_0 |1\rangle - \alpha_1 |0\rangle)}_{\text{Bob}}.
\end{aligned}$$

(3) Alice measures her two qubits in the computational basis, and sends the result (2 random classical bits ab) to Bob over a classical channel.

(4) Bob receives a and b from Alice, and depending on the values of these bits he performs these operations:

- If $b = 1$, Bob performs a bit flip (X-gate) on his qubit.
- If $a = 1$, Bob performs a phase flip (Z-gate) on his qubit.

For instance, if Alice sent $ab = 11$, then Bob knows that his qubit is $\alpha_0 |1\rangle - \alpha_1 |0\rangle$. A bit flip followed by a phase flip will give him Alice's original qubit $\alpha_0 |0\rangle + \alpha_1 |1\rangle$.

1.3 Circuit model

A classical circuit model consists of **gates**. Logical gates model elementary computational steps in digital electronic circuits, such as AND, NOT, OR, XOR.

- NOT gate: flips 0 to 1, 1 to 0
- AND gate: takes in 2 bits, returns 1 if both are 1, returns 0 otherwise
- OR gate: takes in 2 bits, returns 1 if either is 1, returns 0 otherwise
- XOR gate: takes in 2 bits, returns 1 if either one is 1, returns 0 otherwise

More generally, such gates are functions. A quantum gate is a linear map (matrix) on a vector space. Single qubit gates are 2×2 matrices, 2-qubit gates are 4×4 matrices, and 3-qubit gates are 8×8 matrices.

We have discussed 1-qubit gates. Here is a 2-qubit gate, called the **controlled NOT**:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X$$

so that $|00\rangle \mapsto |00\rangle$, $|01\rangle \mapsto |01\rangle$, $|10\rangle \mapsto |11\rangle$, $|11\rangle \mapsto |10\rangle$. Observe that:

- if the first qubit is $|0\rangle$, then we apply an identity to the second qubit;
- if the first qubit is $|1\rangle$, we apply a NOT gate to the second qubit.

The first qubit is called **control qubit**, the second qubit is called **target qubit**. If the control qubit is $|1\rangle$, then the target qubit is flipped; otherwise, the target qubit is unchanged.

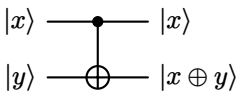


Figure 1.2: CNOT gate.

More generally, let U be an arbitrary single qubit unitary operation. A **controlled-U operation** is a two qubit operation, again with a control and a target qubit. If the control qubit is set, then U is applied to the target qubit; otherwise, the target qubit is unchanged.

$$CU = \begin{pmatrix} I & 0 \\ 0 & U \end{pmatrix} = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U$$

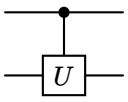


Figure 1.3: Controlled-U operation.

Now that we know how to condition on a single qubit being set, what about conditioning on multiple qubits? One example of multiple qubit conditioning is the **Toffoli gate**. If the first two control qubits are $|1\rangle$, then it flips the third qubit.

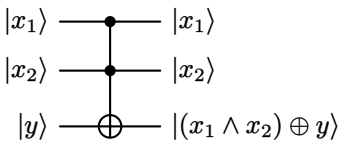


Figure 1.4: The Toffoli gate.

Applying the Toffoli gate twice has the effect $|x_1, x_2, y\rangle \mapsto |x_1, x_2, y \oplus (x_1 \wedge x_2)\rangle \mapsto |x_1, x_2, y\rangle$, so the Toffoli gate is a reversible gate, since it has an inverse – itself.

Quantum circuits cannot be used to directly simulate classical circuits, because unitary quantum logic gates are inherently reversible, but many classical logic gates are irreversible. Hence, we replace any classical circuit by an equivalent circuit containing only reversible elements, using the Toffoli gate.

The generalised CNOT gate of $n+1$-bit input and $n+1$-bit output takes n bits as the controller. If they all are 1, the $(n+1)$-th input bit is flipped; otherwise, the $(n+1)$-th bit does not change.

$$\text{GCNOT}\,|x_1, \ldots, x_n, y\rangle = |x_1, \ldots, x_n, y \oplus (x_1 \wedge \cdots \wedge x_n)\rangle\,.$$

If $n = 1$, then it is the CNOT gate; if $n = 2$, then it is the Toffoli gate.

Consider the following circuit, which has a Hadamard gate followed by a CNOT gate. Let β denote this operation.

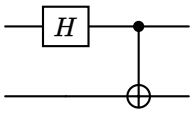


Figure 1.5: Quantum circuit to create Bell states.

The output states of the four computational basis states are called **Bell states** or **EPR states**, all of which are entangled.

$$
\begin{aligned}
|\Phi^+\rangle &= \beta\,|00\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\
|\Phi^-\rangle &= \beta\,|01\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \\
|\Psi^+\rangle &= \beta\,|10\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \\
|\Psi^-\rangle &= \beta\,|11\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)
\end{aligned}
$$

The four Bell states $\{|\Phi^+\rangle, |\Phi^-\rangle, |\Psi^+\rangle, |\Psi^-\rangle\}$ form an orthonormal basis for $\mathbb{C}^4$, known as the **Bell basis**, after physicist John Bell.

1.4 Oracles

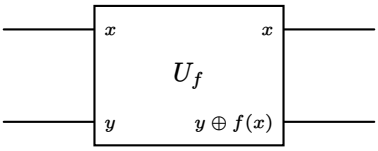
A **quantum oracle** O is a "black box" operation that is used as input to another algorithm. Often, such operations are defined using a classical function $f\colon \{0,1\}^n \to \{0,1\}^m$, which takes n-bit binary input and produces an m-bit binary output.

We may first attempt to define O so that $O\,|x\rangle = |f(x)\rangle$. The problem is that f may not be invertible, but quantum gates must be unitary.

To deal with this problem, we introduce a second register of m qubits to hold our answer. Then we will define the effect of the oracle on all computational basis states: for all $x \in \{0,1\}^n$, $y \in \{0,1\}^m$,

$$U_f(|x\rangle \otimes |y\rangle) = |x\rangle \otimes |y \oplus f(x)\rangle$$

where the notation $y \oplus f(x)$ refers to the bitwise exclusive-OR (addition modulo 2) of two strings; for example, $001 \oplus 101 = 100$.



What the gate U_f does is to echo the top input string x, and XOR the function value $f(x)$ onto the bottom input string y.

The first register is called the **data** register, the second register is called the **target** register.

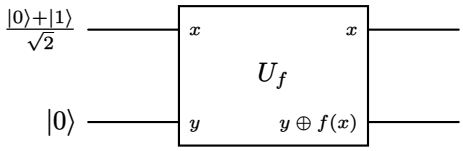
Since XOR is its own inverse, namely $y \oplus f(x) \oplus f(x) = y$, we see that $U_f = U_f^\dagger$. Hence, U_f is unitary.

For a Boolean function $f(x)$ where $f(x) = 1$ for desired solutions and 0 otherwise, a **phase oracle** O_f applies the following transformation to the computational basis states:

$$O_f\,|x\rangle = (-1)^{f(x)}\,|x\rangle\,.$$

For non-solutions ($f(x) = 0$), the state remains unchanged; for solutions ($f(x) = 1$), the state has a phase shift of -1.

Using oracles, we can achieve **quantum parallelism**, which intuitively allows quantum computers to evaluate a function $f(x)$ for many different values of x simultaneously.



Applying U_f yields the state

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

Note that the final superposition contains $f(0)$ and $f(1)$.

More generally, suppose we have a classical algorithm that computes some function $f\colon \{0,1\}^n \to \{0,1\}^m$. Then we can build an oracle U (consisting only of Toffoli gates) that maps $|x\rangle\,|0\rangle \to |x\rangle\,|f(x)\rangle$ for every $x \in \{0,1\}^n$. Apply U to a uniform superposition of *all* inputs x (which can be prepared using n Hadamard gates $H^{\otimes n}$):

$$U\left(\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n} |x\rangle\,|0\rangle\right) = \frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n} |x\rangle\,|f(x)\rangle\,.$$

We applied U just once, but the final superposition contains $f(x)$ for *all* 2^n input values x.

2 Quantum Algorithms

2.1 Deutsch-Jozsa algorithm

Let $f\colon \{0,1\}^n \to \{0,1\}$. We say that f is a **constant function** if $f(x) = 0$ for all $x \in \{0,1\}^n$, or $f(x) = 1$ for all $x \in \{0,1\}^n$. We say that f is a **balanced function** if $f(x) = 0$ for a half of $x \in \{0,1\}^n$ and $f(x) = 1$ for the other half of x, namely, $|\{x \in \{0,1\}^n : f(x) = 0\}| = 2^n/2$.

Deutsch-Jozsa problem:

> Input: a function $f\colon \{0,1\}^n \to \{0,1\}$
>
> Promise: f is either constant or balanced
>
> Output: 0 if f is constant, 1 if f is balanced.

In the classical case, $2^{n-1}+1$ evaluations of f are required in the worst case. This is because we may input half ($2^n/2 = 2^{n-1}$) inputs and get the same output, so we need to query the function one more time to determine if f is constant or balanced.

First consider the case where $n = 1$. Classically, we check whether $f(0) = f(1)$, which is equivalent to checking $f(0) \oplus f(1)$. If $f(0) \oplus f(1) = 0$, then f is constant; if $f(0) \oplus f(1) = 1$, then f is balanced. Thus, we need to evaluate the function twice, $f(0)$ and $f(1)$.

Using **Deutsch's algorithm**, we only need to evaluate the function once.

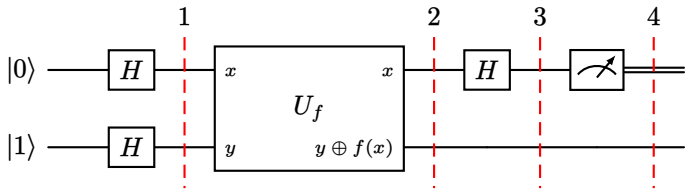


Figure 2.1: Deutsch's algorithm

(1) The input state is $|0\rangle|1\rangle$. Applying a Hadamard gate to each qubit yields

$$|\psi_1\rangle = H|0\rangle \otimes H|1\rangle = |+\rangle|-\rangle.$$

(2) Apply the gate U_f, which maps $|x\rangle|y\rangle \mapsto |x\rangle|y \oplus f(x)\rangle$. This yields the state

$$|\psi_2\rangle = \begin{cases} \pm|+\rangle|-\rangle & \text{if } f(0) = f(1), \\ \pm|-\rangle|-\rangle & \text{if } f(0) \neq f(1). \end{cases}$$

(3) Apply a Hadamard gate to the first qubit. This gives

$$|\psi_3\rangle = \begin{cases} \pm|0\rangle|-\rangle & \text{if } f(0) = f(1), \\ \pm|1\rangle|-\rangle & \text{if } f(0) = f(1). \end{cases}$$

Note that $f(0) \oplus f(1) = 0$ if $f(0) = f(1)$, and 1 if otherwise, so we can rewrite the above as

$$|\psi_3\rangle = \pm|f(0) \oplus f(1)\rangle|-\rangle.$$

(4) Measure the first qubit to determine $f(0) \oplus f(1)$.

Lemma 2.1. *For any n-bit string $|x\rangle$, the general formula for a Hadamard transform on $|x\rangle$ is*

$$H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{z \in \{0,1\}^n} (-1)^{x \cdot z} |z\rangle,$$

where $x \cdot z$ is the bitwise inner product $x_1 z_1 \oplus \cdots \oplus x_n z_n$.

Proof. For the case of 2-bit strings, recall that the action of H is

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

These two equations can be combined into a single formula:

$$H|a\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}(-1)^a|1\rangle = \frac{1}{\sqrt{2}}\sum_{b\in\{0,1\}}(-1)^{ab}|b\rangle,$$

which is true for both choices of $a \in \{0,1\}$.
For the general case, let $x = x_1 \ldots x_n$, where $x_i \in \{0,1\}$. Then

$$\begin{aligned}
H^{\otimes n}|x\rangle &= H|x_1\rangle \otimes \cdots \otimes H|x_n\rangle \\
&= \left(\frac{1}{\sqrt{2}}\sum_{z_1\in\{0,1\}}(-1)^{x_1 z_1}|z_1\rangle\right) \otimes \cdots \otimes \left(\frac{1}{\sqrt{2}}\sum_{z_n\in\{0,1\}}(-1)^{x_n z_n}|y_n\rangle\right) \\
&= \frac{1}{\sqrt{2^n}}\sum_{z_1\ldots z_n\in\{0,1\}^n}(-1)^{x_1 z_1+\cdots+x_n z_n}|z_1\ldots z_n\rangle \\
&= \frac{1}{\sqrt{2^n}}\sum_{z\in\{0,1\}^n}(-1)^{x\cdot z}|z\rangle.
\end{aligned}$$

□

The algorithm for the general case is known as the **Deutsch-Jozsa algorithm**.

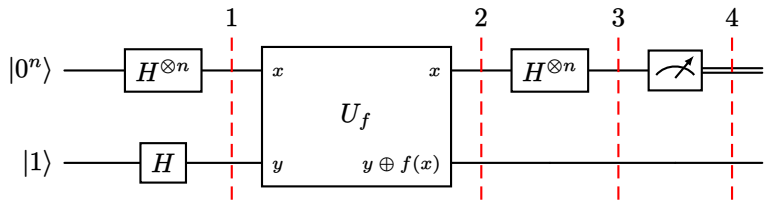


Figure 2.2: Deutsch-Jozsa algorithm.

(1) The input state is $|0^n\rangle|1\rangle$. Applying $H^{\otimes n}$ to $|0^n\rangle$ yields

$$\begin{aligned}
H^{\otimes n}|0^n\rangle &= H|0\rangle \otimes \cdots \otimes H|0\rangle \\
&= |+\rangle \otimes \cdots \otimes |+\rangle \\
&= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \cdots \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\
&= \frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle.
\end{aligned}$$

which is a uniform superposition over all n-bit strings. Hence,

$$|\psi_1\rangle = \left(\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle\right) \otimes |-\rangle.$$

(2) Apply $U_f\colon |x,y\rangle \mapsto |x, y \oplus f(x)\rangle$ to $|\psi_1\rangle$. We first see what happens to a single term $|x\rangle|-\rangle$:

$$U_f|x\rangle|-\rangle = \begin{cases} |x\rangle|-\rangle & \text{if } f(x)=0 \\ -|x\rangle|-\rangle & \text{if } f(x)=1 \end{cases}$$

which can be written concisely as

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

Applying this to the whole superposition gives

$$|\psi_2\rangle = \left(\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}|x\rangle\right)|-\rangle.$$

(3) Apply a second $H^{\otimes n}$ to the top register. Using the general formula for Hadamard transform,

$$\begin{aligned} H^{\otimes n}\left(\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}|x\rangle\right) &= \frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}H|x\rangle \\ &= \frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}\left[\frac{1}{\sqrt{2^n}}\sum_{z\in\{0,1\}^n}(-1)^{x\cdot z}|z\rangle\right] \\ &= \sum_{x\in\{0,1\}^n}\frac{1}{2^n}\sum_{z\in\{0,1\}^n}(-1)^{f(x)}(-1)^{x\cdot z}|z\rangle \\ &= \sum_{z\in\{0,1\}^n}\left[\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)+x\cdot z}\right]|z\rangle \end{aligned}$$

which is a superposition of n-bit strings, where each basis state $|z\rangle$ has amplitude

$$\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)+x\cdot z}.$$

(4) Observe the query register. The amplitude for the state $|0^n\rangle$ is

$$\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)+x\cdot 00\ldots 0} = \frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)+0} = \frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}.$$

If f is constant, the amplitude of $|0^n\rangle$ is 1 or -1, depending on the constant value f takes. Thus, the probability of measuring $00\ldots0$ is 1.

If f is balanced, since for half of the inputs $f(x)=0$ and the other half $f(x)=1$, the sum adds together an equal number of 1's and -1's, which cancel out, resulting in amplitude of 0. Thus, the probability of measuring $00\ldots0$ is 0.

Conclusion: If we measure $00\ldots0$, then f is constant. If we measure any other state, then f is balanced.

2.2 Bernstein-Vazirani algorithm

Bernstein-Vazirani problem:

Input: a function $f\colon \{0,1\}^n \to \{0,1\}$

Promise: there exists a binary string s for which $f(x) = x \cdot s$ for all $x \in \{0,1\}^n$

Output: the string s

Classical solution: Let $s = s_1 s_2 \dots s_n$. To find s_i, evaluate f at $0 \dots 1 \dots 0$ where the i-th position is 1:

$$\begin{aligned}
f(0\dots001) &= 0\dots001 \cdot s = s_1(0) + s_2(0) + \dots + s_{n-2}(0) + s_{n-1}(0) + s_n(1) = s_n \\
f(0\dots010) &= 0\dots010 \cdot s = s_1(0) + s_2(0) + \dots + s_{n-2}(0) + s_{n-1}(1) + s_n(0) = s_{n-1} \\
f(0\dots100) &= 0\dots100 \cdot s = s_1(0) + s_2(0) + \dots + s_{n-2}(1) + s_{n-1}(0) + s_n(0) = s_{n-2} \\
&\vdots \\
f(1\dots000) &= 1\dots000 \cdot s = s_1(1) + s_2(0) + \dots + s_{n-2}(0) + s_{n-1}(0) + s_n(0) = s_1
\end{aligned}$$

More generally, any classical algorithm needs to ask n queries, because the final answer consists of n bits, and one classical query only gives 1 bit of information.

The Bernstein-Vazirani algorithm is *exactly* the same as the Deutsch-Jozsa algorithm, using the function $f(x) = x \cdot s$. The amplitude for the state $|s\rangle$ is

$$\frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{(s \oplus s) \cdot x} = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{(00\dots0) \cdot x} = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^0 = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} 1 = \frac{1}{2^n} 2^n = 1$$

where $\oplus$ represents bitwise XOR; for example, $00110 \oplus 10101 = 10011$.

2.3 Simon's algorithm

Simon's problem:

> Input: a function $f\colon \{0,1\}^n \to \{0,1\}^m$
>
> Promise: there exists $s \in \{0,1\}^n$ such that $f(x) = f(y)$ iff ($x = y$ or $x \oplus s = y$) for all $x, y \in \{0,1\}^n$
>
> Output: the string s

Simon's algorithm:

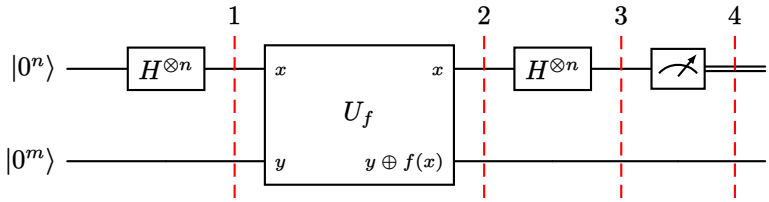


Figure 2.3: Simon's algorithm.

(1) Apply $H^{\otimes n}$ to the first n qubits to put them in a uniform superposition, giving

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

where the second m-qubit register still holds only zeroes.

(2) Apply U_f, which maps $|x\rangle |0^m\rangle \mapsto |x\rangle |f(x)\rangle$, so the state becomes

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle |f(x)\rangle .$$

(3) Applying $H^{\otimes n}$ to the first n qubits transforms each state $|x\rangle$ into $\frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle$. The

joint state of the system becomes

$$\frac{1}{2^n} \sum_{x \in \{0,1\}^n} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle |f(x)\rangle .$$

The probability for the measurements to result in each string $y \in \{0,1\}^n$ is

$$p(y) = \left| \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot y} |f(x)\rangle \right|^2$$

which we rewrite as

$$\begin{aligned} p(y) &= \left| \frac{1}{2^n} \sum_{z \in \text{range}(f)} \left(\sum_{x \in f^{-1}(z)} (-1)^{x \cdot y} |z\rangle \right) \right|^2 \\ &= \frac{1}{2^{2n}} \sum_{z \in \text{range}(f)} \left| \sum_{x \in f^{-1}(z)} (-1)^{x \cdot y} \right|^2 . \end{aligned}$$

To simplify this probability further, let's take a look at the value

$$\left| \sum_{x \in f^{-1}(z)} (-1)^{x \cdot y} \right|^2$$

for an arbitrary selection of $z \in \text{range}(f)$.

Case 1: $s = 0^n$**.** Then f is a one-to-one function, so there are 2^n strings $z \in \text{range}(f)$. For every $z \in \text{range}(f)$, there is a single element $x \in f^{-1}(z)$. Hence,

$$p(y) = \frac{1}{2^{2n}} \cdot 2^n \cdot 1 = \frac{1}{2^n}.$$

In words, the measurements result in a string $y \in \{0,1\}^n$ chosen uniformly at random.

Case 2: $s \neq 0^n$**.** Then f is two-to-one, so there are exactly two strings in the set $f^{-1}(z)$. That is, if $w \in f^{-1}(z)$ is one of the strings, then the other string must be $w \oplus s$, by the promise in Simon's problem. Thus, we simplify

$$\begin{aligned} \left| \sum_{x \in f^{-1}(z)} (-1)^{x \cdot y} \right|^2 &= \left| (-1)^{w \cdot y} + (-1)^{(w \oplus s) \cdot y} \right|^2 \\ &= \left| (-1)^{w \cdot y} \big(1 + (-1)^{s \cdot y}\big) \right|^2 \\ &= \left| 1 + (-1)^{y \cdot s} \right|^2 \\ &= \begin{cases} 4 & \text{if } y \cdot s = 0 \\ 0 & \text{if } y \cdot s = 1 \end{cases} \end{aligned}$$

which is independent of the choice of $z \in \text{range}(f)$. Since there are 2^{n-1} elements in $\text{range}(f)$, we obtain

$$p(y) = \begin{cases} \frac{1}{2^{n-1}} & \text{if } y \cdot s = 0 \\ 0 & \text{if } y \cdot s = 1 \end{cases}$$

In words, we obtain a string chosen uniformly at random from the set $\{y \in \{0,1\}^n : y \cdot s = 0\}$, which contains 2^{n-1} strings.

We now know what the probabilities are for the possible measurement outcomes when we run the quantum circuit for Simon's algorithm. Is this enough information to determine s?

The answer is yes, provided that we are willing to repeat the process several times and accept that it could fail with some probability, which we can make very small by running the circuit enough times.

Suppose we run the circuit independently k times, where $k = n + r$ for some arbitrary positive integer r. Then we obtain strings $y^{(1)}, \ldots, y^{(k)} \in \{0,1\}^n$. Form the matrix M, where the i-th row is the bits of $y^{(i)}$:

$$M = \begin{pmatrix} y_1^{(1)} & \cdots & y_n^{(1)} \\ y_1^{(2)} & \cdots & y_n^{(2)} \\ \vdots & \ddots & \vdots \\ y_1^{(k)} & \cdots & y_n^{(k)} \end{pmatrix}.$$

Let $v = (s_1, \ldots, s_n)^T$ be the vector containing the bits of s. Then

$$Mv = \begin{pmatrix} y^{(1)} \cdot s \\ y^{(2)} \cdot s \\ \vdots \\ y^{(k)} \cdot s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Hence, $s \in \ker M$, where $\ker M$ can be found using Gaussian elimination modulo 2, which runs in $O(n^3)$. Note that the zero vector is always in $\ker M$, as well as s in case $s \neq 0^n$. The probability that the vectors corresponding to 0^n and s are alone in $\ker M$ is at least $1 - 2^{-r}$.

Hence, so long as the vectors corresponding to 0^n and s are alone in $\ker M$, which happens with high probability, we can deduce s from the results of this computation.

2.4 Quantum Fourier transform

Definition 2.2. The **discrete Fourier transform** takes as input a vector of complex numbers $x_0, x_1, \ldots, x_{N-1}$, and outputs the transformed data, a vector of complex numbers $y_0, y_1, \ldots, y_{N-1}$, defined by

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \omega_N^{jk},$$

where $\omega_N = e^{2\pi i/N}$ is the N-th root of unity.

The quantum Fourier transform is exactly the same transformation:

Definition 2.3. The **quantum Fourier transform** on an orthonormal basis $|0\rangle, |1\rangle, \ldots, |N-1\rangle$ is defined as a linear operator $F_N : \mathbb{C}^N \to \mathbb{C}^N$ with the following action on the basis states:

$$|j\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega_N^{jk} |k\rangle.$$

That is,

$$|0\rangle \mapsto \frac{1}{\sqrt{N}} \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}, \qquad |1\rangle \mapsto \frac{1}{\sqrt{N}} \begin{pmatrix} 1 \\ \omega_N \\ \vdots \\ \omega_N^{N-1} \end{pmatrix}, \qquad \cdots, \qquad |N-1\rangle \mapsto \frac{1}{\sqrt{N}} \begin{pmatrix} 1 \\ \omega_N^{N-1} \\ \vdots \\ \omega_N^{(N-1)^2} \end{pmatrix}.$$

By linearity, the action on any arbitrary state is

$$\sum_{j=0}^{N-1} x_j |j\rangle \mapsto \sum_{k=0}^{N-1} y_k |k\rangle$$

where the amplitudes y_k are the discrete Fourier transform of the amplitudes x_j.
Written as a matrix, the QFT is

$$F_N = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega_N & \cdots & \omega_N^{N-1} \\ 1 & \omega_N^2 & \cdots & \omega_N^{2(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_N^{N-1} & \cdots & \omega_N^{(N-1)^2} \end{pmatrix}$$

which maps $|j\rangle \to |\psi_j\rangle$. The inverse of QFT is

$$F_N^{-1} = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega_N^{-1} & \cdots & \omega_N^{-(N-1)} \\ 1 & \omega_N^{-2} & \cdots & \omega_N^{-2(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_N^{-(N-1)} & \cdots & \omega_N^{-(N-1)^2} \end{pmatrix}$$

which maps $|\psi_j\rangle \to |j\rangle$.

Lemma 2.4. *The quantum Fourier transform is a unitary transformation.*

Proof. We will show that the columns of the matrix for F_N form an orthonormal basis. Define a vector corresponding to column number y, where $y = 0, \ldots, N-1$:

$$|\psi_y\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} \omega_N^{xy} |x\rangle .$$

Taking the inner product between any two of these vectors gives

$$\langle \psi_z | \psi_y \rangle = \frac{1}{N} \sum_{x=0}^{N-1} \omega_N^{x(y-z)} = \frac{1}{N} \sum_{x=0}^{N-1} \left(\omega_N^{y-z}\right)^x$$

which is a geometric series. If $y = z$, then

$$\langle \psi_z | \psi_y \rangle = \frac{1}{N} \cdot N = 1.$$

If $y \neq z$, then

$$\langle \psi_z | \psi_y \rangle = \frac{1}{N} \frac{\omega_N^{N(y-z)} - 1}{\omega_n^{y-z} - 1} = \frac{1}{N} \frac{1-1}{\omega_N^{y-z} - 1} = 0.$$

Geometrically, we are summing a bunch of points that are equally distributed around the unit circle, and they cancel out and leave 0 when summed. Hence, we have

$$\langle \psi_z | \psi_y \rangle = \begin{cases} 1 & \text{if } y = z \\ 0 & \text{if } y \neq z \end{cases}$$

which shows that the columns $\{|\psi_0\rangle, \ldots, |\psi_{N-1}\rangle\}$ form an orthonormal set. □

In the following, we take $N = 2^n$, where n is some integer, and the basis $|0\rangle, \ldots, |2^n - 1\rangle$ is the

computational basis for an n qubit quantum computer. It is helpful to write the state $|j\rangle$ using the binary representation

$$j = j_1 j_2 \dots j_n = j_1 2^{n-1} + j_2 2^{n-2} + \dots + j_n 2^0$$

where $j_1, \dots, j_n \in \{0, 1\}$. We also use the notation for binary fractions:

$$0.j_l j_{l+1} \dots j_m = j_l/2 + j_{l+1}/4 + \dots + j_m/2^{m-l+1}.$$

Theorem 2.5. *The quantum Fourier transform has the following useful product representation:*

$$|j_1 \dots j_n\rangle \mapsto \frac{\left(|0\rangle + e^{2\pi i 0.j_n} |1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_{n-1} j_n} |1\rangle\right) \cdots \left(|0\rangle + e^{2\pi i 0.j_1 \dots j_n} |1\rangle\right)}{2^{n/2}}.$$

Proof. First substitute $N = 2^n$:

$$|j\rangle \mapsto \frac{1}{2^{n/2}} \sum_{k=0}^{2^n - 1} e^{2\pi i j k / 2^n} |k\rangle .$$

Write each $|k\rangle$ in binary representation: $k = k_1 2^{n-1} + k_2 2^{n-2} + \dots + k_n 2^0$, so that

$$\begin{aligned}
\frac{1}{2^{n/2}} \sum_{k=0}^{2^n - 1} e^{2\pi i j k / 2^n} |k\rangle &= \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} e^{2\pi i j \left(\sum_{l=1}^{n} k_l 2^{-l}\right)} |k_1 \cdots k_n\rangle \\
&= \frac{1}{2^{n/2}} \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} \bigotimes_{l=1}^{n} e^{2\pi i j k_l 2^{-l}} |k_l\rangle \\
&= \frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left[\sum_{k_l=0}^{1} e^{2\pi i j k_l 2^{-l}} |k_l\rangle \right] \\
&= \frac{1}{2^{n/2}} \bigotimes_{l=1}^{n} \left[|0\rangle + e^{2\pi i j 2^{-l}} |1\rangle \right] \\
&= \frac{\left(|0\rangle + e^{2\pi i 0.j_n} |1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_{n-1} j_n} |1\rangle\right) \cdots \left(|0\rangle + e^{2\pi i 0.j_1 \dots j_n} |1\rangle\right)}{2^{n/2}}.
\end{aligned}$$

□

Hence, the QFT can be factored into a product of n qubit states:

$$|j_1 \dots j_n\rangle \mapsto \frac{\left(|0\rangle + e^{2\pi i 0.j_n} |1\rangle\right)}{\sqrt{2}} \otimes \frac{\left(|0\rangle + e^{2\pi i 0.j_{n-1} j_n} |1\rangle\right)}{\sqrt{2}} \otimes \cdots \otimes \frac{\left(|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n} |1\rangle\right)}{\sqrt{2}}.$$

Let R_k denote a single qubit unitary rotation gate:

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i / 2^k} \end{pmatrix}.$$

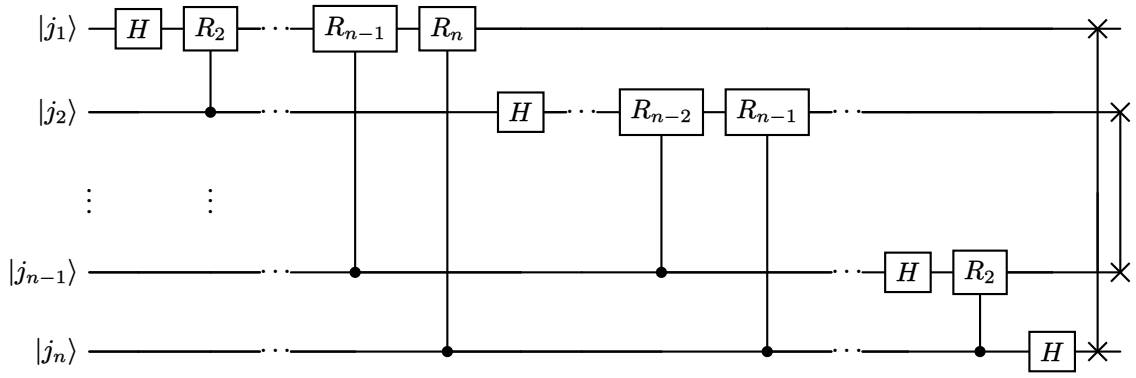


Figure 2.4: Quantum Fourier transform.

To see that this $O(n^2)$-gate circuit indeed computes the quantum Fourier transform, consider what happens to the input state $|j_1 \dots j_n\rangle$. Apply the Hadamard gate to the first bit. Note that

$$H|j_1\rangle = \frac{|0\rangle + (-1)^{j_1}|1\rangle}{\sqrt{2}} = \frac{|0\rangle + e^{i\pi j_1}}{\sqrt{2}} = \frac{|0\rangle + e^{2\pi i(j_1/2)}}{\sqrt{2}} = \frac{|0\rangle + e^{2\pi i 0.j_1}}{\sqrt{2}}.$$

Thus, we obtain the state

$$\frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i 0.j_1}|1\rangle\right)|j_2 \dots j_n\rangle.$$

Applying the controlled-R_2 gate produces the state

$$\frac{1}{2^{1/2}}\left(|0\rangle + e^{2\pi i 0.j_1 j_2}|1\rangle\right)|j_2 \dots j_n\rangle.$$

We continue applying the controlled-R_3, R_4, $\dots$, R_n gates, each of which adds an extra bit to the phase of the coefficient of the first $|1\rangle$. At the end of this procedure, we have the state

$$\frac{1}{2^{1/2}}\left(|0\rangle + e^{2\pi i 0.j_1 j_2 \dots j_n}|1\rangle\right)|j_2 \dots j_n\rangle.$$

Next, we perform a similar procedure on the second qubit. The Hadamard gate puts us in the state

$$\frac{1}{2^{2/2}}\left(|0\rangle + e^{2\pi i 0.j_1 \dots j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_2}|1\rangle\right)|j_3 \dots j_n\rangle$$

and the controlled-R_2, $\dots$, R_{n-1} gates yield the state

$$\frac{1}{2^{2/2}}\left(|0\rangle + e^{2\pi i 0.j_1 \dots j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_2 \dots j_n}|1\rangle\right)|j_3 \dots j_n\rangle.$$

We continue in this fashion for each qubit, giving the final state

$$\frac{1}{2^{n/2}}\left(|0\rangle + e^{2\pi i 0.j_1 \dots j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_2 \dots j_n}|1\rangle\right)\cdots\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right).$$

To invert the QFT, we run the circuit in reverse, with the inverse of each gate.

2.5 Phase estimation

Let U be a unitary operator, and $|u\rangle$ be an eigenvector of U with eigenvalue $e^{i\phi}$ (since all eigenvalues of a unitary operator have norm 1), so $U|u\rangle = e^{i\phi}|u\rangle$.
Consider a controlled-U gate where the control qubit is $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$, target qubit is $|u\rangle$. Applying the

controlled-U gate to the combined state

$$|\psi_0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |u\rangle = \frac{|0\rangle |u\rangle + |1\rangle |u\rangle}{\sqrt{2}}$$

yields

$$\begin{aligned} |\psi_1\rangle = U|\psi_0\rangle &= \frac{|0\rangle |u\rangle + |1\rangle U|u\rangle}{\sqrt{2}} \\ &= \frac{|0\rangle |u\rangle + e^{i\phi}|1\rangle U|u\rangle}{\sqrt{2}} \\ &= \frac{|0\rangle + e^{i\phi}|1\rangle}{\sqrt{2}} \otimes |u\rangle . \end{aligned}$$

Observe that the target qubit $|u\rangle$ remains unchanged, but a phase $e^{i\phi}$ has been added to the control qubit. This effect is called **phase kickback**, since the phase was *kicked back* into the control qubit rather than being applied to the target qubit.

Suppose a unitary operator U has an eigenvector $|\psi\rangle$ with eigenvalue λ. Since $|\lambda| = 1$, write $\lambda = e^{2\pi i\theta}$, for some $\theta \in [0, 1)$ whose value is unknown.

Phase estimation problem:

> Input: a unitary quantum circuit for an n-qubit operation U along with an n-qubit quantum state $|\psi\rangle$
>
> Promise: $|\psi\rangle$ is an eigenvector of U
>
> Output: an approximation to the number $\theta \in [0, 1)$ satisfying $U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle$.

Phase estimation procedure:

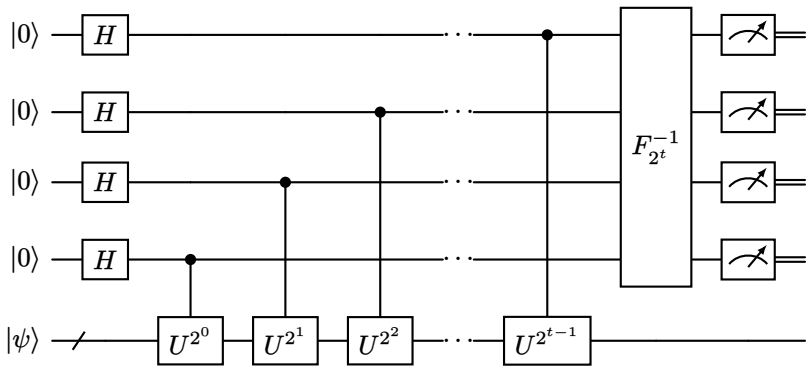


Figure 2.5: Phase estimation procedure.

(1) The first register contains t qubits in the state $|0\rangle$. The second register is $|\psi\rangle$, and contains as many qubits as is necessary to store $|\psi\rangle$.

First apply Hadamard gates to all t qubits in the first register:

$$|\psi_1\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \cdots \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) |\psi\rangle .$$

(2) Apply controlled-U operations on the second register, with U raised to successive powers of two.

We first apply the controlled-U gate to $|\psi\rangle$ with the rightmost qubit in first register as control. Since $|\psi\rangle$ is an eigenvector of U, phase kickback occurs, which adds a relative phase of $e^{2\pi i\theta}$ to the

qubit:

$$|\psi_2\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \cdots \otimes \frac{|0\rangle + e^{2\pi i\theta} |1\rangle}{\sqrt{2}} \right) |\psi\rangle .$$

Then, apply U^2. Once again, phase kickback occurs, which adds a phase of $e^{2\pi i\theta} \cdot e^{2\pi i\theta} = e^{2\pi i\theta \cdot 2}$:

$$|\psi_3\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \cdots \otimes \frac{|0\rangle + e^{2\pi i\theta \cdot 2} |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + e^{2\pi i\theta} |1\rangle}{\sqrt{2}} \right) |\psi\rangle .$$

Continue this process for each of the qubits in the first register, which yields the state

$$|\psi_4\rangle = \left(\frac{|0\rangle + e^{2\pi i\theta \cdot 2^{t-1}} |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + e^{2\pi i\theta \cdot 2^{t-2}} |1\rangle}{\sqrt{2}} \otimes \cdots \otimes \frac{|0\rangle + e^{2\pi i\theta \cdot 2} |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + e^{2\pi i\theta} |1\rangle}{\sqrt{2}} \right) |\psi\rangle .$$

We shall omit the second register, since it remains as $|\psi\rangle$ throughout the computation.

(3) Since $\theta \in [0, 1)$, suppose θ can be written exactly as a t-bit binary fraction $\theta = 0.\theta_1 \dots \theta_t$, so that $\theta = \sum_{k=1}^{t} \theta_k 2^{-k}$. Then we can rewrite the first register as

$$\frac{|0\rangle + e^{2\pi i 0.\theta_t} |1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + e^{2\pi i 0.\theta_{t-1}\theta_t} |1\rangle}{\sqrt{2}} \otimes \cdots \otimes \frac{|0\rangle + e^{2\pi i 0.\theta_1 \dots \theta_t} |1\rangle}{\sqrt{2}}.$$

This state is exactly the result of applying the quantum Fourier transform applied on the state $|\theta_1 \dots \theta_t\rangle$. Hence, applying the inverse QFT recovers $|\theta_1 \dots \theta_t\rangle$.

(4) Finally, measure the t qubits in the first register. The result is a bit string $\theta_1 \dots \theta_t$. When we convert this binary fraction back to a decimal, we get our estimate $\theta \approx \sum_{k=1}^{t} \theta_k 2^{-k}$.

2.6 Shor's algorithm

Factoring problem:

Input: an integer $N \geq 2$

Output: the prime factorisation of N

Factoring small numbers is easy, but factoring very large values of N becomes impossibly difficult. The best classical algorithm, known as the **number field sieve**, takes time $2^{n^{1/3}}$, which is way too slow if numbers have 1000 bits, i.e., $n \approx 1000$. The computational difficulty of the factoring problem is the basis of RSA public-key cryptography, in the sense that an efficient algorithm for the factoring problem would break it.

Group theory

For $N \geq 1$, define $\mathbb{Z}_N = \{0, 1, \dots, N-1\}$ and $\mathbb{Z}_N^* = \{a \in \mathbb{Z}_N : \gcd(a, N) = 1\}$. Then $\mathbb{Z}_N^*$ forms a group under multiplication. The Euler totient function $\phi(N) = |\mathbb{Z}_N^*|$ counts the number of positive integers less than N that are coprime to N. The order of an element $a \in \mathbb{Z}_N^*$ is the smallest r such that $a^r \equiv 1 \pmod{N}$; this is finite by Euler's theorem, which states that $a^{\phi(N)} \equiv 1 \pmod{N}$. This also implies that the order of a is a divisor of $\phi(N)$.

Reduction from factoring to period-finding

Assume N is odd and not a prime power. Randomly choose an integer x, where $1 < x < N$. Compute $\gcd(x, N)$ (e.g., using Euclid's algorithm). If $\gcd(x, N) > 1$, then it is a non-trivial factor of N, so we are done. Thus, assume that x is coprime to N.

This means that $x \in \mathbb{Z}_N^*$, where $\mathbb{Z}_N^*$ is the multiplicative group of integers modulo N. Thus, x has multiplicative order r modulo N, meaning

$$x^r \equiv 1 \pmod{N},$$

and r is the smallest positive integer satisfying this congruence.
Shor's algorithm finds period r. It can be shown that with probability $\geq \frac{1}{2}$, the period r is even and $x^{r/2}+1$ and $x^{r/2}-1$ are not multiples of N. In that case, we have

$$x^r - 1 = (x^{r/2}+1)(x^{r/2}-1) \equiv 0 \pmod{N},$$

so there exists some k such that

$$(x^{r/2}+1)(x^{r/2}-1) = kN.$$

Note that $k > 0$, because both $x^{r/2}+1 > 0$ and $x^{r/2}-1 > 0$. Thus, $x^{r/2}+1$ or $x^{r/2}-1$ shares a factor with N. Since $x^{r/2}+1$ and $x^{r/2}-1$ are not multiples of N, this factor must be $< N$, and in fact both $x^{r/2}+1$ and $x^{r/2}-1$ share a non-trivial factor with N. Hence, $\gcd(x^{r/2}+1, N)$ and $\gcd(x^{r/2}-1, N)$ are non-trivial factors of N, so we have factorised N.

Shor's period-finding algorithm

Shor's algorithm solves the **order-finding problem**:

> Input: coprime integers N and $1 < x < N$
>
> Output: the period r of the function $f(a) = x^a \pmod{N}$

The algorithm sets up two registers: the first register contains l qubits, where we pick $q = 2^l$ such that $N^2 < q \leq 2N^2$. The second register consists of n qubits, where we suppose N can be written in n bits, i.e., $n = \lceil \log N \rceil$.
Let U_f denote the oracle that maps $|a\rangle\,|0^n\rangle \mapsto |a\rangle\,|f(a)\rangle$.

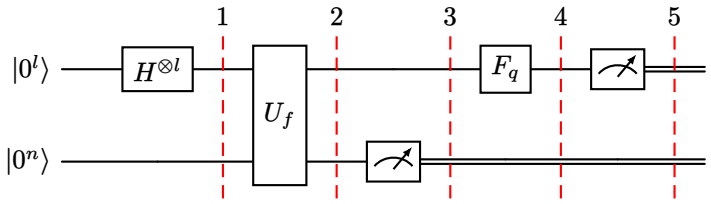


Figure 2.6: Shor's algorithm.

(1) The initial state is $|0^l\rangle\,|0^n\rangle$. Apply Hadamard gates to the first register to build the uniform superposition

$$\frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle\,|0^n\rangle\,.$$

(2) Evaluate the function $f(a)$ by applying the oracle U_f, which gives

$$\frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle\,|f(a)\rangle\,.$$

(3) Measure the second register. The second register collapses to a state $|f(s)\rangle$ for some $s < r$. Then, the first register collapses to only the values of a that map to $f(s)$, namely, $f(a) = f(s)$. Since f is periodic with period r,

$$f(a) = f(s) \iff a \equiv s \pmod{r} \iff a = jr + s,$$

so the values of a are $s, r+s, 2r+s, \ldots$. Thus, the first register collapses to a superposition of $|s\rangle, |r+s\rangle, |2r+s\rangle, \ldots$:

$$\frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |jr+s\rangle$$

where m is the number of elements in the superposition, i.e., the number of integers j such that $jr+s \in \{0, \ldots, q-1\}$.

(4) Apply QFT to the first register. Recall that QFT acts on a basis state $|x\rangle$ as

$$F_q |x\rangle = \frac{1}{\sqrt{q}} \sum_{b=0}^{q-1} e^{\frac{2\pi i x b}{q}} |b\rangle .$$

Applying F_q linearly to our state gives

$$\begin{aligned} F_q \left(\frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} |jr+s\rangle \right) &= \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} F_q |jr+s\rangle \\ &= \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} \frac{1}{\sqrt{q}} \sum_{b=0}^{q-1} e^{2\pi i \frac{(jr+s)b}{q}} |b\rangle \\ &= \frac{1}{\sqrt{mq}} \sum_{b=0}^{q-1} e^{2\pi i \frac{sb}{q}} \left(\sum_{j=0}^{m-1} e^{2\pi i \frac{jrb}{q}} \right) |b\rangle \end{aligned}$$

where we rearranged the order of summations to see the amplitude for each basis state $|b\rangle$:

$$\alpha_b = \frac{1}{\sqrt{mq}} e^{2\pi i \frac{sb}{q}} \left(\sum_{j=0}^{m-1} e^{2\pi i \frac{jrb}{q}} \right)$$

so that

$$p(b) = |\alpha_b|^2 = \left| \frac{1}{\sqrt{mq}} e^{\frac{2\pi i s b}{q}} \sum_{j=0}^{m-1} \left(e^{\frac{2\pi i r b}{q}} \right)^j \right|^2 = \frac{1}{mq} \left| \sum_{j=0}^{m-1} \left(e^{\frac{2\pi i r b}{q}} \right)^j \right|^2 .$$

We want to see which states $|b\rangle$ have large $p(b)$, since those are the states we are likely to see if we now measure. Thus, we look at the internal summation $\sum_{j=0}^{m-1} e^{2\pi i \frac{jrb}{q}}$, which is a geometric series:

$$\sum_{j=0}^{m-1} e^{2\pi i \frac{jrb}{q}} = \sum_{j=0}^{m-1} \left(e^{2\pi i \frac{rb}{q}} \right)^j = \begin{cases} m & \text{if } e^{2\pi i \frac{rb}{q}} = 1 \\ \dfrac{1 - e^{2\pi i \frac{mrb}{q}}}{1 - e^{2\pi i \frac{rb}{q}}} & \text{if } e^{2\pi i \frac{rb}{q}} \neq 1 \end{cases} \qquad (*)$$

Easy case: r divides q. The whole period "fits" an integer number of times in the domain $\{0, \ldots, q-1\}$ of f, and $m = q/r$.

Now we look at the internal summation for different values of b. Since b ranges from 0 to $q-1$, we consider values of b that are multiplies of m, and those that are not. If b is a multiple of $\frac{q}{r}$, then $b = c \cdot \left(\frac{q}{r}\right)$ for some $c \in \{0, 1, \ldots, r-1\}$. Then $e^{\frac{2\pi i r b}{q}} = e^{\frac{2\pi i r \cdot \left(c \frac{q}{r}\right)}{q}} = e^{2\pi i c} = 1$ since $c \in \mathbb{Z}$. Thus, by $(*)$, $\sum_{j=0}^{m-1} \left(e^{\frac{2\pi i r b}{q}} \right)^j = m$, so

$$p(b) = \frac{1}{mq} \cdot |m|^2 = \frac{m^2}{mq} = \frac{m}{q} = \frac{1}{r}.$$

Hence, any state b that is an exact multiple of q/r has a uniform, non-zero probability of exactly $\frac{1}{r}$.

Since there are r such basis states $|b\rangle$, we have $\sum p(b) = 1$. This implies that the amplitudes of $|b\rangle$ that are *not* integer multiples of q/r must all be 0. Hence, we are left with a superposition where only the b

that are integer multiples of q/r have non-zero amplitude.

Finally, observe this final superposition. The measurement outcome is some random multiple $b = cq/r$, where $c \in \{0, 1, \dots, r-1\}$ is uniformly random, so

$$\frac{b}{q} = \frac{c}{r}$$

where b and q are known to the algorithm, c and r are not. Since b and q are known values, we can simplify $\frac{b}{q}$ to its lowest terms. If c and r are coprime, then the denominator is the period r.

Fact: $\phi(r) \in \Omega(r/\log\log r)$. Thus, the probability that c is coprime to r is

$$\frac{\phi(r)}{r} = \Omega(1/\log\log r) \geq \Omega(1/\log\log N).$$

Hence, we need expected $O(\log\log N)$ repetitions of the algorithm to obtain $b = cq/r$ such that c is coprime to r.

Hard case: r does not divide q. Then q/r is not an integer, so $e^{\frac{2\pi i r b}{q}} \neq 1$. Using $|1 - e^{i\theta}| = 2|\sin\theta/2|$, we rewrite the absolute value of the second case of $(*)$:

$$\left|\frac{1 - e^{2\pi i \frac{mrb}{q}}}{1 - e^{2\pi i \frac{rb}{q}}}\right| = \frac{\left|1 - e^{2\pi i \frac{mrb}{q}}\right|}{\left|1 - e^{2\pi i \frac{rb}{q}}\right|} = \frac{\left|\sin m\pi \frac{rb}{q}\right|}{\left|\sin \pi \frac{rb}{q}\right|}.$$

We want to see which values of b maximise this ratio. The ratio becomes large when the denominator is close to 0, meaning $\frac{rb}{q}$ is close to an integer $c \in \mathbb{Z}$, so $\frac{b}{q} \approx \frac{c}{r}$.

Fact: With high probability, the measurement yields some b such that

$$\left|\frac{b}{q} - \frac{c}{r}\right| \leq \frac{1}{2q},$$

for a random $c \in \{0, \dots, r-1\}$. Equivalently, $|b - cq/r| \leq 1/2$, so the measurement outcome b is an integer multiple of q/r rounded up/down. As in the easy case, b and q are known to us, while c and r are unknown.

Since $\frac{b}{q} \neq \frac{c}{r}$, we cannot just try to find r by writing b/q in lowest terms like we did in the easy case. Since $q > N^2$, the distance between any two distinct fractions with denominators $\leq N$ must be at least $\frac{1}{N^2} > \frac{1}{q}$.[1] Hence, $\frac{c}{r}$ is the unique fraction with denominator $\leq N$ within a distance of $\frac{1}{2q}$ from $\frac{b}{q}$.

To find $\frac{c}{r}$, we pass the known value $\frac{b}{q}$ into the classical Continued Fractions Algorithm, which gives us the fraction with denominator $\leq N$ that is closest to b/q (see next section); this fraction is c/r. Again, c and r are coprime with probability $\Omega(1/\log\log r)$, so the denominator is r.

Continued fractions

Any real number x can be expressed as a **continued fraction**

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \frac{1}{\dots}}}$$

We write this compactly as $[a_0, a_1, a_2, \dots]$, where the integers a_i are called **partial quotients**; we assume these to be positive natural numbers.

Given a real number x, the following "algorithm" gives a continued fraction expansion of x, where we

[1] Consider two distinct fractions $\frac{c}{r}$ and $\frac{c'}{r'}$ where $r, r' \leq N$. Then

$$\left|\frac{c}{r} - \frac{c'}{r'}\right| = \left|\frac{cr' - c'r}{rr'}\right| \geq \frac{1}{|rr'|} \geq \frac{1}{N^2}.$$

iteratively peel off the integer part and invert the remaining fractional part:

$$\begin{aligned} a_0 &= \lfloor x \rfloor, & x_1 &= 1/(x - a_0) \\ a_1 &= \lfloor x_1 \rfloor, & x_2 &= 1/(x_1 - a_1) \\ a_2 &= \lfloor x_2 \rfloor, & x_3 &= 1/(x_2 - a_2) \\ &\vdots \end{aligned}$$

If we truncate this infinite process at step n, we obtain a rational fraction called the **n-th convergent**, denoted $\frac{p_n}{q_n} = [a_0, a_1, \ldots, a_n]$.
These convergents can be computed inductively using the following recurrence relations:

$$\begin{aligned} p_0 &= a_0, & p_1 &= a_1 a_0 + 1, & p_n &= a_n p_{n-1} + p_{n-2} \\ q_0 &= 1, & q_1 &= a_1, & q_n &= a_n q_{n-1} + q_{n-2} \end{aligned}$$

RSA cryptography

RSA, named after its founders Rivest-Shamir-Adleman, is a public key cryptosystem still widely used today. Roughly, in a **public key** cryptosystem, Alice wishes to allow the public to send her private messages. For this, she creates a pair of keys: a public key P, a private key S. The public key P is distributed openly to the world; anyone can use P to encrypt a given message M, obtaining ciphertext $E(M)$ for transmission to Alice. Upon receipt of any such $E(M)$, Alice uses her private key S, which is only known to her, to decrypt $E(M)$ and recover M.
For RSA, we shall specify the public and private keys P and S, respectively, as well as the encryption and decryption algorithms:

- **How Alice creates the keys:**
  (1) Alice chooses two large prime integers p and q, and computes $N = pq$.
  (2) She randomly chooses small odd $e \in \mathbb{Z}$ coprime to $\phi(N) = (p-1)(q-1)$.
  (3) She computes d satisfying $e^d \equiv 1 \pmod{\phi(N)}$.
  (4) She publicly releases $P = (e, N)$, and privately stores $S = (d, N)$.
- **How the public encrypts messages given $P = (e, N)$:** Suppose Bob wishes to encrypt a string $M \in \{0,1\}^{\lfloor \log N \rfloor}$. Viewing M as the binary encoding of an integer, he computes ciphertext
$$E(M) = M^e \pmod{N}.$$
- **How Alice decrypts ciphertext $E(M)$:** Given ciphertext $E(M)$, Alice recovers M by computing
$$M = (E(M))^d \pmod{N}.$$
- **How factoring allows one to break RSA:** Suppose an adversary Eve can factor N efficiently into primes p and q. Then, since e is public knowledge, she can recompute d such that $e^d \equiv 1 \pmod{\phi(N)}$ (Step 3 of Alice's creation protocol), giving her Alice's private key $S = (d, N)$.
  Thus, not only can Eve decrypt any incoming ciphertext to Alice, but she in fact has complete knowledge of Alice's secret key.

In sum, what makes RSA classically "hard" to break is that given just $N = pq$, it is not clear how to compute $\phi(N) = (p-1)(q-1)$. But given the factors p and q, recovering $\phi(N)$, and hence the private key d, is easy.

2.7 Grover's algorithm

Search problem:

Input: an arbitrary $x \in \{0,1\}^N$, where $N = 2^n$

Output: an i such that $x_i = 1$ (and to output "no solutions" if there are no such i).

We denote the number of solutions in x by t (i.e., t is the Hamming weight of x).
Grover's algorithm:

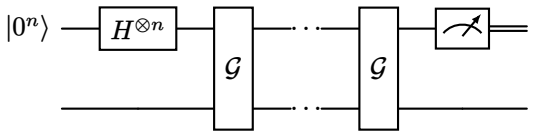


Figure 2.7: Grover's algorithm.

(1) Apply $H^{\otimes n}$ to put $|0^n\rangle$ in uniform superposition:

$$|U\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle .$$

(2) For k times (where k is chosen later), apply the Grover iterate

$$\mathcal{G} = H^{\otimes n} R_0 H^{\otimes n} O_{x,\pm}$$

where $O_{x,\pm} |i\rangle = (-1)^{x_i} |i\rangle$ denotes the $\pm$-type oracle for the input x (i.e., a phase-query), and R_0 denotes the unitary transformation that puts a -1 in front of all basis states $|i\rangle$ where $i \neq 0^n$, and does nothing to the basis state $|0^n\rangle$.

(3) Measure the final state.

Intuitively, what happens is that in each iteration some amplitude is moved from the indices of the 0-bits to the indices of the 1-bits. The algorithm stops when almost all of the amplitude is on the 1-bits, in which case a measurement of the final state will probably give the index of a 1-bit.
In order to analyse this, define the following "good" and "bad" states, corresponding to the solutions and non-solutions, respectively:

$$|G\rangle = \frac{1}{\sqrt{t}} \sum_{i:x_i=1} |i\rangle , \quad |B\rangle = \frac{1}{\sqrt{N-t}} \sum_{i:x_i=0} |i\rangle .$$

Then the uniform state over all indices can be written as

$$\begin{aligned} |U\rangle &= \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle \\ &= \frac{1}{\sqrt{N}} \left(\sum_{i:x_i=1} |i\rangle + \sum_{i:x_i=0} |i\rangle \right) \\ &= \frac{\sqrt{t}}{\sqrt{N}} |G\rangle + \frac{\sqrt{N-t}}{\sqrt{N}} |B\rangle \\ &= \sin\theta \, |G\rangle + \cos\theta \, |B\rangle \end{aligned}$$

where $\theta = \arcsin(\sqrt{t/N})$.

Let V be a subspace of the state space. Then any $v \in V$ can be uniquely written as $v = u + w$, where $u \in V$ and w is orthogonal to V. A **reflection** through the subspace V is a unitary A such that $Au = u$ for all $u \in V$, and $Aw = -w$ for all w orthogonal to V; that is,

$$Av = u - w.$$

Note that we can write $A = 2P_V - I$, where P_V is the projector onto subspace V.
The Grover iterate $\mathcal{G}$ can be written as the product of two reflections.

- Firstly, $O_{x,\pm}$ is a reflection through the subspace V spanned by the basis states that are not solutions; restricted to the 2-dimensional space spanned by $|G\rangle$ and $|B\rangle$ this is in fact just a reflection through the state $|B\rangle$.
- Secondly,

$$\begin{aligned} H^{\otimes n} R_0 H^{\otimes n} &= H^{\otimes n}\left(2\left|0^n\right\rangle\left\langle 0^n\right| - I\right) H^{\otimes n} \\ &= 2H^{\otimes n}\left|0^n\right\rangle\left\langle 0^n\right| H^{\otimes n} - H^{\otimes n} I H^{\otimes n} \\ &= 2\left|U\right\rangle\left\langle U\right| - I \end{aligned}$$

  is a reflection through $|U\rangle$.

Geometric argument

Consider the 2-dimensional real plane spanned by $|B\rangle$ and $|G\rangle$. Initially,

$$|U\rangle = \sin\theta\,|G\rangle + \cos\theta\,|B\rangle\,.$$

The two reflections increase the angle from θ to 3θ, moving us towards the good state.

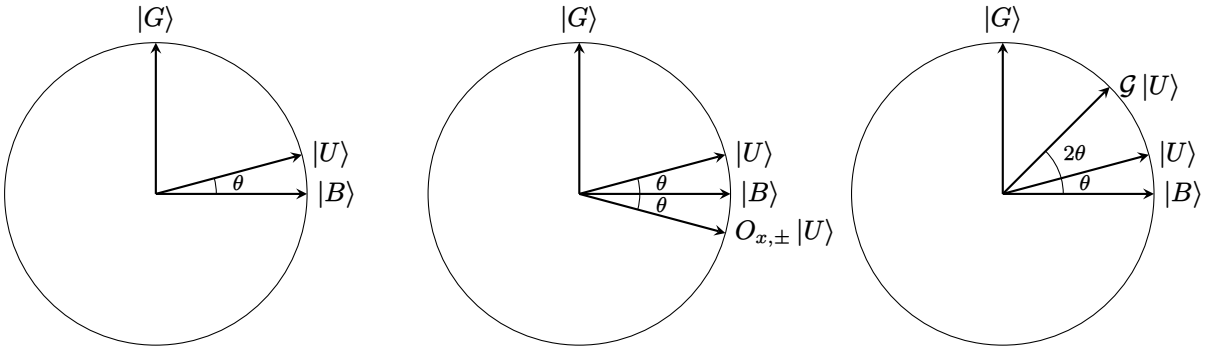


Figure 2.8: The first iteration of Grover: (left) start with $|U\rangle$; (middle) reflect through $|B\rangle$ to get $O_{x,\pm}\,|U\rangle$; (right) reflect through $|U\rangle$ to get $\mathcal{G}\,|U\rangle$.

The next two reflections increase the angle with another 2θ, etc. More generally, after k applications of (a) and (b), our state has become

$$\sin((2k+1)\theta)\,|G\rangle + \cos((2k+1)\theta)\,|B\rangle\,.$$

If we now measure, the probability of seeing a solution is $P_k = \sin((2k+1)\theta)^2$. We want P_k to be as close to 1 as possible. Note that if we can choose $\tilde{k} = \frac{\pi}{4\theta} - \frac{1}{2}$, then $(2\tilde{k}+1)\theta = \frac{\pi}{2}$ and so $P_{\tilde{k}} = \sin^2(\pi/2) = 1$. Unfortunately, $\tilde{k} = \frac{\pi}{4\theta} - \frac{1}{2}$ will usually not be an integer, and we can only do an integer number of Grover iterations. However, if we choose k to be the integer closest to $\tilde{k}$, then our final state will still be close to

$\tilde{G}$, so the failure probability will still be small (assuming $t \ll N$):

$$\begin{aligned} 1 - P_k &= \cos((2k+1)\theta)^2 \\ &= \cos((2\tilde{k}+1)\theta + 2(k-\tilde{k})\theta)^2 \\ &= \cos(\pi/2 + 2(k-\tilde{k})\theta)^2 \\ &= \sin(2(k-\tilde{k})\theta)^2 \\ &\leq \sin^2\theta = \frac{t}{N}, \end{aligned}$$

where we used $|k - \tilde{k}| \leq \frac{1}{2}$. Since $\arcsin\theta \geq \theta$, the number of queries is $k \leq \frac{\pi}{4\theta} \leq \frac{\pi}{4}\sqrt{\frac{N}{t}}$, which is $O(\sqrt{N})$.

3 Hidden Subgroup Problem

Hidden subgroup problem:

> Input: a known group G and a function $f\colon G \to S$ where S is some finite set
>
> Promise: f has the property that there exists a subgroup $H \leq G$ such that f is constant within each coset, and distinct on different cosets: $f(g) = f(g')$ iff $gH = g'H$
>
> Output: H

Since H may be large, "finding H" typically means finding a generating set for H.

3.1 Some instances of HSP

Many important problems can be written as an instance of the HSP.

- **Simon's problem:** Take G to be the additive group ${\mathbb{Z}_n}^2 = \{0,1\}^n$, $H = \{0, s\}$ for a "hidden" $s \in \{0,1\}^n$, and f satisfies $f(x) = f(y)$ iff $(x - y = 0$ or $x - y = s)$ iff $x - y \in H$.

  Hence, finding the generator of H (i.e., finding s) solves Simon's problem.

- **Period-finding:** Given x that is coprime to N and associated function $f\colon \mathbb{Z} \to \mathbb{Z}_N^*$ by $f(a) = x^a \pmod N$, we want to find the period r of f. Consider $\langle x \rangle$, which is a subgroup of $\mathbb{Z}_N^*$ of order r. Thus, r divides $|\mathbb{Z}_N^*| = \phi(N)$. Hence, we can restrict the domain of f to $\mathbb{Z}_{\phi(N)} = \{0, 1, \ldots, \phi(N) - 1\}$.

  Let $G = \mathbb{Z}_{\phi(N)}$. Consider its subgroup $H = \langle r \rangle$ of all multiples of r up to $\phi(N)$, (i.e., $H = r\mathbb{Z}_{\phi(N)} = \{0, r, 2r, \ldots, \phi(N) - r\}$). Because of its periodicity, f is constant on each coset $s + H$ of H, and distinct on different cosets. Since the hidden subgroup H is generated by r, finding the generator of H solves the period-finding problem.

- **Discrete logarithm:** The discrete logarithm problem is as follows: Given a cyclic multiplicative group $C = \langle \gamma \rangle$ of size N, namely,

  $$C = \{\gamma^a : a = 0, 1, \ldots, N - 1\}$$

  and $A \in C$, find the unique a such that $\gamma^a = A$. This value of a is called the *discrete logarithm* of A (w.r.t. generator γ).

  Take $G = \mathbb{Z}_N \times \mathbb{Z}_N$. Define $f\colon G \to C$ by $f(x, y) = \gamma^x A^{-y}$. For group elements $g_1 = (x_1, y_1), g_2 = (x_2, y_2) \in G$,

  $$\begin{aligned} f(g_1) = f(g_2) &\iff \gamma^{x - ay_1} = \gamma^{x_2 - ay_2} \\ &\iff x_1 - ay_1 \equiv x_2 - ay_2 \pmod N \\ &\iff (x_1 - x_2) \equiv a(y_1 - y_2) \pmod N \\ &\iff g_1 - g_2 \in \langle (a, 1) \rangle. \end{aligned}$$

  Let H be the subgroup of G generated by the element $(a, 1)$, then we have an instance of the HSP. Finding the generator of the hidden subgroup H gives us a, solving the discrete log problem.

3.2 Abelian HSP

A **representation** of a group G on a vector space V is a group homomorphism $\phi\colon G \to \mathrm{GL}(V, \mathbf{F})$, where $\mathrm{GL}(V, \mathbf{F})$ is the set of invertible linear operators on V; if V is finite-dimensional, then $\mathrm{GL}(V, \mathbf{F}) \cong \mathrm{GL}(n, \mathbf{F})$, the group of $n \times n$ invertible matrices with entries in $\mathbf{F}$.

A d-dimensional representation of a multiplicative group G is a homomorphism $\rho\colon g \mapsto \rho(g)$ from G to the set of $d \times d$ invertible complex matrices.

A representation of G is **irreducible** if it cannot be decomposed further into the direct sum of lower-dimensional representations of G.

A 1-dimensional representation of G is called a **character** of G. Note that a character χ is irreducible, and the complex values $\chi(g)$ must have absolute value 1, because $|\chi(g^k)| = |\chi(g)|^k$ for all integers k.

For example, the group $\mathbb{Z}_2 = \{0, 1\}$ has two characters: the χ that maps both elements to 1, and the χ that maps 0 to 1 and 1 to -1.

Recall the discrete Fourier transform, which is an $N \times N$ matrix:

$$F_N = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \cdots \\ 1 & \omega_N & \omega_N^2 & \cdots \\ 1 & \omega_N^2 & \omega_N^4 & \cdots \\ \vdots & \vdots & \vdots & \end{pmatrix}$$

where $\omega_N = e^{2\pi i/N}$. Notice every entry is

$$(F_N)_{j,k} = \frac{1}{\sqrt{N}} \omega_N^{jk}.$$

Ignoring the normalizing factor of $\frac{1}{\sqrt{N}}$, its k-th column may be viewed as a map $\chi_k \colon \mathbb{Z}_N \to \mathbb{C}$ defined by

$$\chi_k(j) = \omega_N^{jk}.$$

Note that, by exponent rules,

$$\chi_k(j + j') = \omega_N^{(j+j')k} = \omega_N^{jk} \omega_N^{j'k} = \chi_k(j)\chi_k(j'),$$

so χ_k is a homomorphism. Hence, χ_k is a 1-dimensional representation (i.e., a character) of $\mathbb{Z}_N$. In fact, the N characters corresponding to the N columns of the Fourier matrix are all the characters of $\mathbb{Z}_N$.

Let G be a finite Abelian group. Classification of finite Abelian groups tells us that

$$G \cong \mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_\ell}$$

for some $\ell, N_1, \ldots, N_\ell \in \mathbb{N}$. That is, every finite Abelian group can be decomposed into cyclic pieces.

The $|G| = N_1 \cdots N_\ell$ characters of the product group are just the products of the characters of the individual cyclic groups $\mathbb{Z}_{N_j}$. For example, if $G \cong \mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2}$, then a character is obtained by multiplying characters from each coordinate, e.g., $\chi(a, b) = \chi^{(1)}(a)\chi^{(2)}(b)$, so the characters naturally factor coordinatewise.

Note that the characters are pairwise orthogonal.

The set of all characters of G forms a group $\hat{G}$ with the operation of pointwise multiplication. This is called the **dual group** of G.

If $H \leq G$, then the following is a subgroup of G of size $|G|/|H|$:

$$H^\perp = \{\chi_k : \chi_k(h) = 1 \text{ for all } h \in H\}.$$

We now interpret the quantum Fourier transform in terms of characters. For $k = 0, 1, \ldots, N-1$, define the state whose amplitudes are the (normalised) values of χ_k:

$$|\chi_k\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \chi_k(j) \, |j\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \omega_N^{jk} \, |j\rangle \, .$$

This is simply the k-th column of the discrete Fourier transform matrix interpreted as a quantum state.

With this notation, the QFT just maps the standard (computational) basis of $\mathbb{C}^N$ to the orthonormal basis corresponding to the characters:

$$F_N \colon \, |k\rangle \mapsto |\chi_k\rangle \, .$$

The QFT corresponding to a group G that is isomorphic to $\mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_\ell}$ is just the tensor product of the QFTs for the individual cyclic groups.

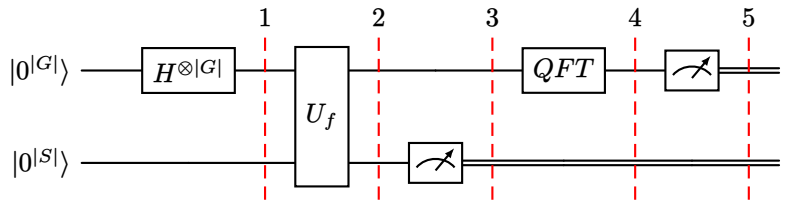


Figure 3.1: Abelian HSP.

(1) The input state is $|0^{|G|}\rangle\,|0^{|S|}\rangle$. Create a uniform superposition in the first register:

$$\frac{1}{\sqrt{|G|}}\sum_{g\in G}|g\rangle\,|0\rangle\,.$$

(2) Apply U_f to compute f in superposition:

$$\frac{1}{\sqrt{|G|}}\sum_{g\in G}|g\rangle\,|f(g)\rangle\,.$$

(3) Measure the second register. This yields some value $f(s)$ for unknown $s \in G$. The first register collapses to a superposition over the g with the same f-value as s (i.e., the coset $s + H$):

$$\frac{1}{\sqrt{|H|}}\sum_{h\in H}|s+h\rangle\,.$$

(4) Apply the QFT corresponding to G to this state, giving

$$\frac{1}{\sqrt{|H|}}\sum_{h\in H}|\chi_{s+h}\rangle\,.$$

Expanding the definition,

$$\begin{aligned}\frac{1}{\sqrt{|H|}}\sum_{h\in H}|\chi_{s+h}\rangle &= \frac{1}{\sqrt{H}}\sum_{h\in H}\left(\frac{1}{|G|}\sum_{g\in G}\chi_{s+h}(g)\,|g\rangle\right)\\ &= \frac{1}{\sqrt{|H||G|}}\sum_{h\in H}\sum_{g\in G}\chi_{s+h}(g)\,|g\rangle\\ &= \frac{1}{\sqrt{|H||G|}}\sum_{h\in H}\sum_{g\in G}\chi_{s}(g)\chi_{h}(g)\,|g\rangle\\ &= \frac{1}{\sqrt{|H||G|}}\sum_{g\in G}\chi_{s}(g)\sum_{h\in H}\chi_{h}(g)\,|g\rangle\end{aligned}$$

which is a superposition of states $|g\rangle$. To evaluate the amplitudes of each $|g\rangle$, we need to evaluate the inner sum $\sum_{h\in H}\chi_h(g)$:

$$\sum_{h\in H}\chi_h(g) = \sum_{h\in H}\chi_g(h) = \begin{cases}|H| & \text{if } \chi_g \in H^\perp\\ 0 & \text{if } \chi_g \notin H^\perp\end{cases}$$

To see this, if $\chi_g \in H^\perp$, then $\chi_g(h) = 1$ for all $h \in H$, so $\sum_{h\in H}\chi_g(h) = 1 + \cdots + 1 = |H|$. If $\chi_g \notin H^\perp$, since χ_g is a homomorphism, its values on H form a nontrivial subgroup of the complex

roots of unity. Summing over all elements of that subgroup cancels out each other, so $\sum_{h \in H} \chi_g(h)$. Hence, the sum becomes

$$\sqrt{\frac{|H|}{|G|}} \sum_{g : \chi_g \in H^\perp} \chi_s(g) \, |g\rangle \, .$$

(5) Measure and output the resulting g. For every surviving g, since $\chi_g \in H^\perp$, using $|\chi_s(g)|^2 = 1$,

$$p(g) = \left| \sqrt{\frac{|H|}{|G|}} \chi_s(g) \right|^2 = \frac{|H|}{|G|}.$$

The above algorithm thus samples uniformly from the (labels of) elements of $H^\perp$. Each such element $\chi_g \in H^\perp$ gives us a constraint on H because $\chi_g(h) = 1$ for all $h \in H$. Generating a small number of such elements will give sufficient information to find the generators of H itself.

For example, consider **Simon's problem**. Let $G = {\mathbb{Z}_2}^n = \{0,1\}^n$ and $H = \{0, s\}$ for some unknown s. Then $\chi_g(h) = (-1)^{g \cdot h}$, where $g \cdot h$ denotes the bitwise dot product, so

$$\begin{aligned} \chi_g \in H^\perp &\iff \chi_g(h) = 1 \quad \text{for all } h \in H \\ &\iff g \cdot s \equiv 0 \pmod 2. \end{aligned}$$

Every time we run the quantum circuit, we sample a random $g \in \{0,1\}^n$ such that $g \cdot s \equiv 0 \pmod 2$. This gives linear equations about s:

$$\begin{aligned} g^{(1)} \cdot s &\equiv 0 \pmod 2 \\ g^{(2)} \cdot s &\equiv 0 \pmod 2 \\ &\vdots \end{aligned}$$

After running the algorithm $O(n)$ times, we obtain a system of $n - 1$ independent linear equations about s. Then solve the linear system using Gaussian elimination to find s.

3.3 General non-Abelian HSP

Unfortunately we do not have an efficient algorithm for most non-Abelian HSPs.

Graph isomorphism problem

An instance of the non-Abelian HSP is the **graph isomorphism problem**:

> Given two undirected n-vertex graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, decide whether there exists an isomorphism between $\mathcal{G}_1$ and $\mathcal{G}_2$.

To solve this problem via HSP, let $\mathcal{G}$ be the $2n$-vertex graph that is the disjoint union of the two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$.

Let $G = S_{2n}$ be the symmetric group, which is the set of all bijections from $\{1, \dots, n\}$ to $\{1, \dots, n\}$; this is the set of all possible ways to permute the $2n$ vertices of $\mathcal{G}$.

Let S be the set of all possible $2n$-vertex graphs. Define $f \colon G \to S$ by $\pi \mapsto \pi(\mathcal{G})$, which takes a permutation $\pi \in S_{2n}$ and applies it to the graph $\mathcal{G}$; the output is the newly rearranged graph.

An **automorphism** of a group G is a group isomorphism $G \to G$. The set of automorphisms on a group G forms a group under composition, known as the automorphism group $\mathrm{Aut}(G)$. Take $H = \mathrm{Aut}(\mathcal{G})$, the **automorphism group** of $\mathcal{G}$; it consists of all permutations of the graph $\mathcal{G}$.

We check that f is constant within each coset of H, and distinct on different cosets. If two permutations π_1 and π_2 belong to the same coset, then $\pi_2 = \pi_1 \circ h$ for some $h \in \mathrm{Aut}(\mathcal{G})$. Then $f(\pi_2) = \pi_2(\mathcal{G}) = \pi_1(h(\mathcal{G})) = \pi_1(\mathcal{G}) = f(\pi_1)$. Conversely, suppose $f(\pi_1) = f(\pi_2)$. Then $\pi_1(\mathcal{G}) = \pi_2(\mathcal{G})$, so $\mathcal{G} = (\pi_1^{-1}\pi_2)(\mathcal{G})$, which implies $\pi_1^{-1}\pi_2 \in \mathrm{Aut}(G) = H$. Thus, $\pi_1 H = \pi_2 H$.
For simplicity, assume that $\mathcal{G}_1$ and $\mathcal{G}_2$ are connected.

Case 1: If $\mathcal{G}_1$ and $\mathcal{G}_2$ are not isomorphic, then the only automorphisms of $\mathcal{G}$ are the ones that permute vertices inside $\mathcal{G}_1$ and inside $\mathcal{G}_2$, so we can factor

$$\mathrm{Aut}(\mathcal{G}) = \mathrm{Aut}(\mathcal{G}_1) \times \mathrm{Aut}(\mathcal{G}_2).$$

Case 2: If $\mathcal{G}_1$ and $\mathcal{G}_2$ are isomorphic, then $\mathrm{Aut}(\mathcal{G})$ contains a permutation that swaps the first n vertices with the second n vertices.

Hence, if we are able to find a generating set of $H = \mathrm{Aut}(\mathcal{G})$, then we can just check whether all generators are in $\mathrm{Aut}(\mathcal{G}_1) \times \mathrm{Aut}(\mathcal{G}_2)$ and decide graph isomorphism.

4 Quantum Walk Algorithms

4.1 Classical random walks

Problem:

Input: an undirected graph G with N vertices

Promise: an ε-fraction of the vertices are marked

Output: a marked vertex

One way to do this is with a **random walk**:

(1) Start at a specific vertex y.

(2) Check if y is marked. If it is, we are done.

(3) If it isn't, randomly choose one of its neighbours, move there, update y to be that neighbour, and repeat the process.

Transition matrix

We restrict our attention to d-regular graphs without self-loops, so each vertex has exactly d neighbours. A random walk on such a graph G corresponds to an $N \times N$ matrix P, called a **transition matrix**, defined by

$$P_{x,y} = \begin{cases} 1/d & \text{if } (x,y) \text{ is an edge in } G \\ 0 & \text{otherwise} \end{cases}$$

where $P_{x,y}$ is the probability of moving from vertex y to vertex x. Since a system must go somewhere, the sum of probabilities in any given column equals 1.

Since the graph is undirected, if (x,y) is an edge, (y,x) is also an edge. Thus, $P_{xy} = P_{yx}$, meaning P is symmetric. This implies that the sum of probabilities in any given row also equals 1.

Suppose we are initially at vertex y. Our position can be represented as a vector $v \in \mathbb{R}^N$ where the y-th entry is 1 and all other entries are 0; we call this a **state vector**. Then Pv is a vector whose x-th entry is

$$(Pv)_x = \sum_{j=1}^{N} P_{x,j} v_j.$$

Since $v_j = 1$ only when $j = y$, and 0 everywhere else, the summation reduces to

$$(Pv)_x = P_{x,y} \cdot 1 = P_{x,y} = \begin{cases} 1/d & \text{if } (x,y) \text{ is an edge in } G \\ 0 & \text{otherwise} \end{cases}$$

This shows that Pv is the uniform probability distribution over the neighbours of y, since we have equal probability of $1/d$ of landing on any adjacent vertex, and 0 probability of landing anywhere else.

More generally, let $v = (v_1, \ldots, v_N)^T$ be a probability distribution on the vertices, where $v_i \geq 0$ represents the probability of starting at vertex i, satisfying $\sum_{i=1}^{N} v_i = 1$. Then for each x,

$$(Pv)_x = \sum_{y=1}^{N} P_{x,y} v_y$$

which means that the total probability of being at vertex x after one step is the sum over all vertices y of (probability of starting at y) $\times$ (probability of moving from y to x).

Hence, Pv is the new probability distribution on vertices after taking one step of the random walk. By induction, $P^k v$ is the probability distribution after taking k steps.

Spectral graph theory

Suppose we start with some probability-distribution vector v. Assume G is connected and not bipartite. In spectral graph theory, we analyse the eigenvalues and eigenvectors of the adjacency matrix P.

The characteristic polynomial of P is $p(\lambda) = \det(P - \lambda I)$, whose roots are the eigenvalues of P. To determine the degree of $p(\lambda)$, note that

$$P - \lambda I = \begin{pmatrix} P_{11} - \lambda & P_{12} & \cdots & P_{1N} \\ P_{21} & P_{22} - \lambda & \cdots & P_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ P_{N1} & P_{N2} & \cdots & P_{NN} - \lambda \end{pmatrix}.$$

When computing the determinant, the highest power of λ comes directly from multiplying the N diagonal elements together:

$$(P_{11} - \lambda)(P_{22} - \lambda) \cdots (P_{NN} - \lambda)$$

so the degree of $p(\lambda)$ is N. By the fundamental theorem of algebra, $p(\lambda)$ has N (possibly complex) roots, counting algebraic multiplicity. Since P is a real symmetric matrix, its eigenvalues are real.[2] Since P is normal (and thus self-adjoint), by the spectral theorem, the corresponding eigenvectors are mutually orthogonal.

Let $\lambda_1 \geq \cdots \geq \lambda_N$ be the eigenvalues of P, and $v_1, \ldots, v_N$ be corresponding orthogonal eigenvectors. An eigenvalue of P is $\lambda_1 = 1$, and corresponds to the $v_1 = u = (1/N, \ldots, 1/N)^T$ which is the uniform distribution over all vertices:

$$(Pu)_x = \sum_{y=1}^{N} P_{xy} \cdot \frac{1}{N} = \frac{1}{N} \sum_{y=1}^{N} P_{xy} = \frac{1}{N}$$

so $Pu = u$.

Next, we show that all eigenvalues of P have absolute value ≤ 1. Let λ be any eigenvalue of P with corresponding eigenvector w, so $Pw = \lambda w$. Write $w = (w_1, \ldots, w_N)^T$, and let x be the specific index of the vector w that has the maximum absolute value, meaning $|w_x| = \max_{1 \leq i \leq N} |w_i|$.

Consider the x-th component of both sides of $Pw = \lambda w$:

$$\lambda w_x = (Pw)_x = \sum_{y=1}^{N} P_{xy} w_y.$$

Taking the absolute value of both sides, by triangle inequality,

$$|\lambda||w_x| = \left| \sum_{y=1}^{N} P_{x,y} w_y \right| \leq \sum_{y=1}^{N} P_{x,y} |w_y| \leq \sum_{y=1}^{N} P_{x,y} |w_x| = |w_x| \sum_{y=1}^{N} P_{x,y} = |w_x|.$$

Dividing both sides by the non-zero $|w_x|$ yields $|\lambda| \leq 1$, as desired.

One can show that our assumption that G is connected implies $\lambda_2 < 1$, and our assumption that G is not bipartite implies $\lambda_N > -1$. Hence, all eigenvalues λ_i for $i = 2, \ldots, N$ will be in $(-1, 1)$. Since each corresponding eigenvector v_i is orthogonal to the uniform vector u, the sum of its entries is 0.

Let $\delta > 0$ be the difference between $\lambda_1 = 1$ and $\max_{i \geq 2} |\lambda_i|$ (hence $|\lambda_i| \leq 1 - \delta$ for all $i \geq 2$). This δ is called the **spectral gap** of the graph; it is the "gap" between the first eigenvalue of P and all other eigenvalues.

[2] To see this, let λ be an eigenvalue of P with eigenvector v. Since $P = P^T = P^*$,

$$\langle Pv, Pv \rangle = v^* P^* P v = v^* P^2 v = v^* (P^2 v) = \lambda^2 \|v\|^2.$$

Hence, $\lambda^2 = \frac{\langle Pv, Pv \rangle}{\|v\|^2}$ is a real non-negative number, so λ must be real.

Theorem 4.1. *Suppose G is connected and not bipartite with N vertices. Let $v \in \mathbb{R}^N$ be an arbitrary initial probability distribution over the vertices of G. Then $P^k v$ converges to the uniform distribution over all vertices as $k \to \infty$, and the speed of convergence is determined by the spectral gap of P.*

Proof. Let $\eta > 0$ be given. We will show that there exists $K \in \mathbb{N}$ such that

$$\left\| P^k v - u \right\| \leq \eta \quad \text{for all } k \geq K.$$

Since $\{\lambda_1, \ldots, \lambda_N\}$ is an orthonormal basis of $\mathbb{R}^N$, we can uniquely write v as a linear combination of them: $v = \sum_{i=1}^N \alpha_i v_i$. Since the sum of v's entries is 1, and the sum of v_1's entries is 1, while each other eigenvector v_i $(i \geq 2)$ has entries summing to 0, it follows that $\alpha_1 = 1$.
Apply the random walk for k steps, starting from v:

$$P^k v = P^k \left(\sum_{i=1}^N \alpha_i v_i \right) = \sum_{i=1}^N \alpha_i \lambda_i^k v_i = u + \sum_{i \geq 2} \alpha_i \lambda_i^k v_i.$$

Consider the squared norm of the difference between $P^k v$ and u:

$$\left\| P^k v - u \right\|^2 = \left\| \sum_{i \geq 2} \alpha_i \lambda_i^k v_i \right\|^2 = \sum_{i \geq 2} |\alpha_i|^2 |\lambda_i|^{2k} \leq \|v\|^2 (1-\delta)^{2k}.$$

We want to bound $\left\| P^k v - u \right\| \leq \eta$, or $\left\| P^k v - u \right\|^2 \leq \eta^2$. Since v is a probability distribution, we have $\|v\|^2 \leq 1$. Then

$$(1-\delta)^{2k} \leq \eta^2 \implies (1-\delta)^k \leq \eta.$$

Using the inequality $1 - \delta \leq e^{-\delta}$, we get $e^{-\delta k} \leq \eta$, so $k \geq \ln(1/\eta)/\delta$. □

In particular, if δ is not too small, then we get quick convergence of the random walk to the uniform distribution u, no matter which distribution v we started with. Once we are close to the uniform distribution, we have probability roughly ε of hitting a marked vertex.
Now we compute the total cost to find a marked vertex. Define:

- $\boldsymbol{S}$: Setup cost for the initial distribution v.
- $\boldsymbol{U}$: Update cost for performing 1 step of the random walk.
- $\boldsymbol{C}$: Checking cost to check whether a given vertex is marked.

Consider a classical search algorithm:

(1) Start at v.

(2) Check if the current vertex is marked.

(3) If yes, we are done. If not, run a random walk for roughly $1/\delta$ steps to get close to the uniform distribution.

In each search trial, it costs $\boldsymbol{C}$ to check the current vertex, and $\frac{1}{\delta}\boldsymbol{U}$ to run a random walk. From a uniform distribution, the probability of hitting a marked vertex is ε, so the expected number of independent trials required is $\frac{1}{\varepsilon}$ (by geometric distribution).
Hence, the expected cost before this procedure finds a marked vertex is

$$\boldsymbol{S} + \frac{1}{\varepsilon}\left(\boldsymbol{C} + \frac{1}{\delta}\boldsymbol{U} \right).$$

4.2 Quantum walks

In a classical random walk, we only need to store the current vertex we are at. In a quantum walk, a basis state requires two registers: the first register tracks the current vertex $|x\rangle$, the second register tracks the previous vertex $|y\rangle$. Thus, a basis state $|x\rangle\,|y\rangle$ of a quantum walk corresponds to an edge (y, x) of the graph.[3]

Similar to Grover's algorithm, we call a basis state $|x\rangle\,|y\rangle$ **good** if x is a marked vertex, and **bad** otherwise. Let M denote the set of marked vertices; then $|M|$ is the number of marked vertices.

For any vertex x, define a uniform superposition over all of its neighbors:

$$|p_x\rangle = \sum_y \sqrt{P_{xy}}\,|y\rangle = \frac{1}{\sqrt{d}} \sum_{y\sim x} |y\rangle\,,$$

where we took square roots $\sqrt{P_{xy}}$ to make $|p_x\rangle$ a unit vector.

Define **good** and **bad** states as the superpositions over good and bad basis states:

$$|G\rangle = \frac{1}{\sqrt{|M|}} \sum_{x\in M} |x\rangle\,|p_x\rangle\,, \qquad |B\rangle = \frac{1}{\sqrt{N-|M|}} \sum_{x\notin M} |x\rangle\,|p_x\rangle\,.$$

Note that $|G\rangle$ is the uniform superposition over all edges (x, y) where the first coordinate is marked, and $|B\rangle$ is the uniform superposition over all edges (x, y) where the first coordinate is not marked.

The fraction of marked vertices is $\varepsilon = \frac{|M|}{N}$. Our starting state $|U\rangle$ is the uniform superposition over all vertices:

$$|U\rangle = \frac{1}{\sqrt{N}} \sum_x |x\rangle\,|p_x\rangle\,.$$

Split this summation into two: one over the marked vertices and one over the unmarked vertices:

$$\begin{aligned}
|U\rangle &= \frac{1}{\sqrt{N}} \left(\sum_{x\in M} |x\rangle\,|p_x\rangle + \sum_{x\notin M} |x\rangle\,|p_x\rangle \right) \\
&= \frac{\sqrt{|M|}}{\sqrt{N}}\,|G\rangle + \frac{\sqrt{N-|M|}}{\sqrt{N}}\,|B\rangle \\
&= \sqrt{\frac{|M|}{N}}\,|G\rangle + \sqrt{1-\frac{|M|}{N}}\,|B\rangle \\
&= \sqrt{\varepsilon}\,|G\rangle + \sqrt{1-\varepsilon}\,|B\rangle \\
&= \sin\theta\,|G\rangle + \cos\theta\,|B\rangle
\end{aligned}$$

where $\theta = \arcsin\sqrt{\varepsilon}$. Hence, $|U\rangle$ is a vector in the 2D plane spanned by $|B\rangle$ and $|G\rangle$, and it makes an angle θ relative to $|B\rangle$.

Search algorithm:

(1) Prepare the starting state $|U\rangle$.

(2) Repeat the following operations $O(1/\sqrt{\varepsilon})$ times:

  (a) Reflect through $|B\rangle$.

  (b) Reflect through $|U\rangle$.

(3) Observe the first register and check that the resulting vertex x is marked.

Similar to Grover, the two reflections (a) and (b) increase the angle from θ to 3θ, moving us towards the good state. After k applications of (a) and (b), the state becomes

$$\sin((2k+1)\theta)\,|G\rangle + \cos((2k+1)\theta)\,|B\rangle\,.$$

[3] We store current and previous vertices, because quantum evolution must be unitary (reversible).

To maximise the probability of measuring a good state, we want $\sin((2k+1)\theta)$ to be as close to 1 as possible:

$$(2k+1)\theta \approx \frac{\pi}{2} \implies 2k\theta \approx \frac{\pi}{2} \implies k \approx \frac{\pi}{4\theta} \approx \frac{\pi}{4\sin\theta} = \frac{\pi}{4\sqrt{\varepsilon}},$$

where we used the small angle approximation $\sin\theta \approx \theta$. Hence, $k = O(1/\sqrt{\varepsilon})$. At this point, measuring the first register yields a marked vertex with a high probability.

We now see how to implement the two reflections:

(a) Reflect through $|B\rangle$. This is a conditional phase flip on the marked items: if the first register contains a marked x, then put a -1.

(b) Reflect through $|U\rangle$. Define the subspaces

$$\mathcal{A} = \text{span}\{|x\rangle\,|p_x\rangle\}, \qquad \mathcal{B} = \text{span}\{|p_y\rangle\,|y\rangle\}$$

where $\mathcal{A}$ contains all states where the first register stores a vertex and the second register stores the uniform superposition of its neighbours. Similarly, $\mathcal{B}$ is the analogous space with the two registers swapped.

Let $\text{ref}(\mathcal{A})$ and $\text{ref}(\mathcal{B})$ denote reflections through $\mathcal{A}$ and $\mathcal{B}$, respectively. Suppose we are able to implement the following two operations:

$$R_1\colon\ |x\rangle\,|0\rangle \mapsto |x\rangle\,|p_x\rangle\,, \qquad R_2\colon\ |0\rangle\,|y\rangle \mapsto |p_y\rangle\,|y\rangle\,.$$

We implement $\text{ref}(\mathcal{A})$ as follows:

(1) Apply the inverse of R_1 to map $|x\rangle\,|p_x\rangle \to |x\rangle\,|0\rangle$.

(2) Perform a conditional phase shift that targets the second register: if it is $|0\rangle$, leave it unchanged; else, it gets a negative sign.

(3) Apply R_1 to restore $|x\rangle\,|0\rangle \to |x\rangle\,|p_x\rangle$.

Similarly, we implement $\text{ref}(\mathcal{B})$ as follows:

(1) Apply the inverse of R_2 to map $|p_y\rangle\,|y\rangle \to |0\rangle\,|y\rangle$.

(2) Perform a conditional phase shift that targets the first register: if it is $|0\rangle$, leave it unchanged; else, it gets a negative sign.

(3) Apply R_2 to restore $|0\rangle\,|y\rangle \to |p_y\rangle\,|y\rangle$.

The **quantum walk operator** is built by reflecting first about $\mathcal{A}$ and then about $\mathcal{B}$. This pair of reflections plays the role of "taking one step" in the quantum walk.

$$W(P) = \text{ref}(\mathcal{B})\text{ref}(\mathcal{A}).$$

To implement the reflection through $|U\rangle$, consider the eigenvalues and eigenvectors of operator $W(P)$. Let $\lambda_1, \lambda_2, \ldots$ be the eigenvalues of the classical transition matrix P. Since each $\lambda_j \in [-1, 1]$, we can write $|\lambda_j| = \cos\theta_j$ for some $\theta_j \in [0, \pi/2]$.

Fact: The eigenvalues of $W(P)$ are of the form $e^{\pm 2i\theta_j}$.

For the uniform distribution $|U\rangle$ where $\lambda_1 = 1$, since $\theta_1 = 0$, the corresponding eigenvalue of $|U\rangle$ is $e^0 = 1$, which has phase $= 0$.

Let δ denote the spectral gap of P. Then for all other eigenvalues, $\lambda_j \le 1 - \delta$, so $\cos\theta_j \le 1 - \delta$. Using the Taylor series for cosine $\cos\theta \ge 1 - \frac{\theta^2}{2}$, we obtain

$$1 - \frac{{\delta_j}^2}{2} \le 1 - \delta \implies \theta_j \ge \sqrt{2\delta}.$$

That is, the phase of every non-principal eigenvalue of $W(P)$ is separated from 0 by some distance.

To reflect across $|U\rangle$, we want to implement a reflection operator $R(P)$ such that

- if the state is $|U\rangle$, then leave it unchanged;
- if the state is orthogonal to $|U\rangle$, then flip its sign.

Idea: Since $|U\rangle$ has a phase of 0, and every other state has a phase that is bounded from 0 by a gap of at least $\sqrt{2\delta}$, we can use phase estimation to tell them apart.

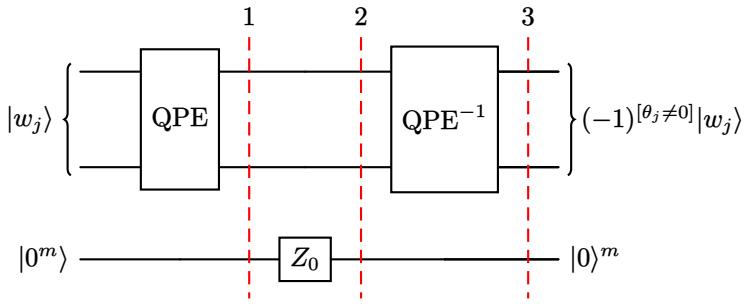


Figure 4.1: Reflection across $|U\rangle$.

(1) Apply phase estimation, which reads the phase angle of the incoming state and writes that value directly into the auxiliary register:

$$|w_j\rangle\,|0\rangle \xrightarrow{\text{PE}} |w_j\rangle\,|\tilde{\theta}_j\rangle$$

where w_j is an eigenvector of $W(P)$.

(2) Conditional phase flip: If the estimated phase $\tilde{\theta}_j$ is sufficiently far from 0, multiply the state by -1. If it is 0, do nothing.

$$|w\rangle\,|\tilde{\theta}_j\rangle \longrightarrow (-1)^{[\theta_j \neq 0]}\,|w\rangle\,|\tilde{\theta}_j\rangle\,.$$

(3) Apply the inverse of phase estimation to put the auxiliary qubits back to 0:

$$(-1)^{[\theta_j \neq 0]}\,|w\rangle\,|\tilde{\theta}_j\rangle \xrightarrow{\text{PE}^{-1}} (-1)^{[\theta_j \neq 0]}\,|w\rangle\,|0\rangle\,.$$

Hence, applied to some eigenvector $|w\rangle$ of $W(P)$ with corresponding eigenvalue $e^{\pm 2i\theta_j}$, $R(P)$ maps

$$|w\rangle\,|0\rangle \longrightarrow (-1)^{[\theta_j \neq 0]}\,|w\rangle\,|0\rangle\,.$$

This has the desired effect: ignoring the auxiliary qubits (which start and end in 0), $R(P)$ maps eigenvalue-1 eigenvectors of $W(P)$ to themselves, and puts a -1 in front of the other eigenvectors.

Finally, we analyse the complexity of the quantum walk algorithm.

- $\boldsymbol{S}$: The cost to construct the initial state $|U\rangle$.
- $\boldsymbol{C}$: The checking cost to determine if a state is in the marked space $|B\rangle$.
- $\boldsymbol{U}$: The cost of executing a single step of the quantum walk operator $W(P)$.

The cost of part (a) of the algorithm is $\boldsymbol{C}$. Since $R(P)$ uses $O(1/\sqrt{\delta})$ applications of $W(P)$, the cost of part (b) of the algorithm is $O(\boldsymbol{U}/\sqrt{\delta})$. Hence, the total cost is

$$\boldsymbol{S} + \frac{1}{\sqrt{\varepsilon}}\left(\boldsymbol{C} + \frac{1}{\sqrt{\delta}}\boldsymbol{U}\right).$$

Note that quantum search square-roots both ε and δ.

5 Hamiltonian Simulation

5.1 Hamiltonians

Postulate 2 states that the time evolution of the state of a closed quantum system is described by the Schrödinger equation

$$i\hbar \frac{\mathrm{d}\,|\psi(t)\rangle}{\mathrm{d}t} = H\,|\psi(t)\rangle$$

where $\hbar$ is Planck's constant divided 2π. In practice, it is common to absorb the factor $\hbar$ into H, effectively setting $\hbar = 1$. H is a fixed Hermitian operator known as the **Hamiltonian** of the closed system.

Thus, if we start in some state $|\psi(0)\rangle$, the solution to this differential equation is the following unitary evolution of the state:

$$|\psi(t)\rangle = U\,|\psi(0)\rangle\,, \quad \text{where } U = e^{-iHt}.$$

Hence, t time-steps of evolution induced by Hamiltonian H corresponds to applying the unitary matrix e^{-iHt} times.

Remark. For a function $f\colon \mathbb{R} \to \mathbb{R}$ and Hermitian matrix H, we write $f(H)$ for the matrix we get by applying f to the eigenvalues of H while keeping the eigenvectors the same, i.e.,

$$f(H) := U \begin{pmatrix} f(\lambda_1) & & \\ & \ddots & \\ & & f(\lambda_n) \end{pmatrix} U^{-1} \quad \text{where } H = U \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix} U^{-1}.$$

Remark. Here, t need not be an integer: this evolution is continuous in time, in contrast to the discrete picture one gets from the circuit model with elementary quantum gates.

We need methods to efficiently implement the unitary evolution that is induced by a given Hamiltonian. In other words, we need methods to implement $U = e^{-iHt}$ as a quantum circuit of gates, and apply this to a given initial state $|\psi\rangle$. This is known as the problem of **Hamiltonian simulation**.

5.2 Method 1: Lie-Suzuki-Trotter methods

An n-qubit Hamiltonian H is a $2^n \times 2^n$ matrix. As n grows, this matrix becomes physically impossible to write out classically because its size scales exponentially. However, physical Hamiltonians are usually **d-local**, meaning they can be decomposed into a sum of m smaller terms:

$$H = \sum_{j=1}^{m} H_j$$

where each H_j acts only on a few of the n qubits. For concreteness, assume they are 2-local, meaning each H_j acts on only 2 qubits. Since H_j only acts on 2 qubits, the operator $e^{iH_j t}$ is just a simple 2-qubit gate acting as the identity matrix on the remaining $n-2$ qubits.

Our goal is to implement $U = e^{iHt} = e^{i\sum_{j=1}^{m} H_j t}$. If all H_j commute, we can split the exponential, so $U = \prod_{j=1}^{m} e^{iH_j t}$. However, in general, two matrices A and B do not commute, so $e^{A+B} \neq e^A e^B$.

The **Lie-Suzuki-Trotter decomposition** gives us a way to handle this. It uses the fact that if A and B have small operator norm, then e^{A+B} and $e^A e^B$ are *approximately* equal: $e^{A+B} = e^A e^B + E$, where the error-term E is a matrix whose operator norm is $O(\|A\| \cdot \|B\|)$. To visualise this error, using Taylor expansion,

$$e^{A+B} \approx I + (A+B) + \frac{1}{2}(A+B)^2 = I + A + B + \frac{1}{2}A^2 + \frac{1}{2}B^2 + \frac{1}{2}(AB+BA)$$
$$e^A e^B \approx (I + A + \frac{1}{2}A^2)(I + B + \frac{1}{2}B^2) \approx I + A + B + \frac{1}{2}A^2 + \frac{1}{2}B^2 + AB$$

How can we use this to approximate U by a circuit $\tilde{U}$ of 2-qubit gates? Assume each H_j has operator

norm ≤ 1. First consider the simple case $m = 2$, so $H = H_1 + H_2$. To minimise the error, we want to minimise the matrix exponentials, so we implement $U = e^{iHt}$ by doing a little bit of H_1, a little bit of H_2, a little bit of H_1, etc. More precisely, for every integer $r \geq 1$ of our choice,

$$\begin{aligned} U = e^{iHt} &= (e^{iHt/r})^r \\ &= (e^{iH_1t/r + iH_2t/r})^r \\ &= (e^{iH_1t/r}e^{iH_2t/r} + E)^r \\ &\approx (e^{iH_1t/r}e^{iH_2t/r})^r = \tilde{U} \end{aligned}$$

so the error-term E has norm

$$\|E\| = O\left(\left\|iH_1\frac{t}{r}\right\| \cdot \left\|iH_2\frac{t}{r}\right\|\right) = O\left(\|H_1\| \cdot \|H_2\|\frac{t^2}{r^2}\right).$$

The approximating circuit $\tilde{U} = (e^{iH_1t/r}e^{iH_2t/r})^r$ uses $2^r = O(t^2/\varepsilon)$ 2-qubit gates. Since errors in a product of unitaries add at most linearly (see Exercise 4.4), we have approximation error

$$\left\|U - \tilde{U}\right\| \leq r\|E\| = r \cdot O\left(\frac{t^2}{r^2}\right) = O\left(\frac{t^2}{r}\right) \leq \varepsilon$$

by choosing $r = O(t^2/\varepsilon)$. In general, if our Hamiltonian H is a sum of m terms,

$$U = e^{iHt} = (e^{iHt/r})^r = (e^{iH_1t/r + \cdots + iH_mt/r})^r = \left(e^{iH_1t/r}\cdots e^{iH_mt/r} + E\right)^r$$

where $\|E\| = O(m^2t^2/r^2)$, so

$$\left\|U - \tilde{U}\right\| \leq r\|E\| = O\left(\frac{m^2t^2}{r}\right).$$

Thus, choose $r = O(m^2t^2/\varepsilon)$ to bound the error by ε.
Since each slice requires executing m individual 2-qubit gates, the total gate count of our approximating circuit $\tilde{U} = \left(e^{iH_1t/r}\cdots e^{iH_mt/r}\right)^r$ is $m \cdot r = m \cdot O\left(\frac{m^2t^2}{\varepsilon}\right) = O\left(\frac{m^3t^2}{\varepsilon}\right)$.

5.3 Method 2: Linear combination of unitaries (LCU)

5.4 Method 3: Transforming block-encoded matrices

6 Quantum Noise and Quantum Operations

6.1 Classical noise

Suppose we wish to send a bit from one location to another through a noisy classical communications channel (e.g., magnetic fields may flip bits). The effect of the noise in the channel is to flip the bit being transmitted with probability $p > 0$, while with probability $1 - p$ the bit is transmitted without error. Such a channel is known as a **binary symmetric channel**.

Let p_0 and p_1 be the initial probabilities that the bit is in the states 0 and 1 respectively. Let q_0 and q_1 be the corresponding probabilities after the noise has occurred.

Let X be the initial state of the bit, and Y the final state of the bit. By the law of total probability,

$$\mathrm{P}[Y = y] = \sum_x \mathrm{P}[Y = y \mid X = x]\,\mathrm{P}[X = x].$$

That is,

$$\begin{pmatrix} q_0 \\ q_1 \end{pmatrix} = \begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix} \begin{pmatrix} p_0 \\ p_1 \end{pmatrix}$$

which we can write concisely as

$$\vec{q} = E\vec{p}$$

where $\vec{p}$ is the input probabilities, $\vec{q}$ is the output probabiltiies, E is a matrix of transition probabilities, known as the **evolution matrix**.

What properties must the evolution matrix E possess? We require that if $\vec{p}$ is a probability distribution, then $E\vec{p}$ must also be a probability distribution. Satisfying this condition turns out to be equivalent to two conditions on E:

(1) **Positivity**: All the entries of E must be non-negative.

If they weren't, then it would be possible to obtain negative probabilities in $E\vec{p}$.

(2) **Completeness**: All the columns of E must sum to one.

Suppose this weren't true. Imagine, for example, that the first column didn't sum to one. Letting $\vec{p}$ contain a one in the first entry and zeroes everywhere else, we see that $E\vec{p}$ would not be a probability distribution in this case.

6.2 Density operator

We have formulated quantum mechanics using the language of state vectors. An alternate formulation is possible using the **density operator** or **density matrix**, which provides a convenient means for describing quantum systems whose state is not completely known.

Definition 6.1. Suppose a quantum system is in one of a number of states $|\psi_i\rangle$ with respective probabilities p_i. We call $\{p_i, |\psi_i\rangle\}$ an **ensemble of pure states**. The **density operator** for the system is defined as

$$\rho = \sum_i p_i \,|\psi_i\rangle \,\langle\psi_i|\,.$$

The density operator is often known as the **density matrix**; we use the two terms interchangeably.

Example. The density matrix $\rho = \frac{1}{2}\,|0\rangle\,\langle 0| + \frac{1}{2}\,|1\rangle\,\langle 1|$ is

$$\rho = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix} = \frac{I}{2}.$$

In contrast, the state vector $|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ is given by

$$|\psi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\end{pmatrix} + \frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}.$$

Note that the former is a matrix, while the latter is a vector.

It turns out that all the postulates of quantum mechanics can be reformulated in terms of the density operator language.

For example, suppose that the evolution of a closed quantum system is described by a unitary operator U. If the system was initially in the state $|\psi_i\rangle$ with probability p_i, then after the evolution the system will be in the state $U|\psi_i\rangle$ with probability p_i. Thus, the evolution of the density operator is given by

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i| \xrightarrow{U} \sum_i U|\psi_i\rangle\langle\psi_i|U^\dagger = U\rho U^\dagger.$$

Measurements can also be described in the density operator language. Suppose we perform a measurement described by measurement operators M_m. If the initial state was $|\psi_i\rangle$, then the probability of getting result m is

$$p(m \mid i) = \langle\psi_i|M_m^\dagger M_m|\psi_i\rangle = \operatorname{tr}(M_m^\dagger M_m |\psi_i\rangle\langle\psi_i|).$$

To see why the last equality holds, let $|\psi\rangle$ be a unit vector and A be an arbitrary operator. By cyclic property of trace, $\operatorname{tr}(A|\psi\rangle\langle\psi|) = \operatorname{tr}(\langle\psi|A|\psi\rangle) = \langle\psi|A|\psi\rangle$, where the last equality follows since $\langle\psi|A|\psi\rangle$ is a 1×1 matrix.

By the law of total probability, the probability of obtaining outcome m is

$$\begin{aligned} p(m) &= \sum_i p(m \mid i)p_i \\ &= \operatorname{tr}(M_m^\dagger M_m |\psi_i\rangle\langle\psi_i|) \\ &= \operatorname{tr}(M_m^\dagger M_m \rho). \end{aligned}$$

What is the density operator of the system after obtaining the measurement result m? If the initial state was $|\psi_i\rangle$, then the state after obtaining the result m is

$$|\psi_i^m\rangle = \frac{M_m|\psi_i\rangle}{\sqrt{\langle\psi_i|M_m^\dagger M_m|\psi_i\rangle}}.$$

Thus, after a measurement which yields the outcome m, we have an ensemble of states $|\psi_i^m\rangle$ with

respective probabilities $p(i \mid m)$. The corresponding density operator ρ_m is therefore

$$\begin{aligned}
\rho_m &= \sum_i p(i \mid m) \left|\psi_i^m\right\rangle \left\langle\psi_i^m\right| \\
&= \sum_i p(i \mid m) \frac{M_m \left|\psi_i\right\rangle \left\langle\psi_i\right| M_m^\dagger}{\left\langle\psi_i|M_m^\dagger M_m|\psi_i\right\rangle} \\
&= \sum_i \frac{p(m \mid i) p_i}{p_m} \cdot \frac{M_m \left|\psi_i\right\rangle \left\langle\psi_i\right| M_m^\dagger}{\left\langle\psi_i|M_m^\dagger M_m|\psi_i\right\rangle} && \text{[Bayes' theorem]} \\
&= \sum_i \frac{\operatorname{tr}(M_m^\dagger M_m \left|\psi_i\right\rangle \left\langle\psi_i\right|) p_i}{\operatorname{tr}(M_m^\dagger M_m \rho)} \cdot \frac{M_m \left|\psi_i\right\rangle \left\langle\psi_i\right| M_m^\dagger}{\left\langle\psi_i|M_m^\dagger M_m|\psi_i\right\rangle} \\
&= \sum_i p_i \frac{M_m \left|\psi_i\right\rangle \left\langle\psi_i\right| M_m^\dagger}{\operatorname{tr}(M_m^\dagger M_m \rho)} \\
&= \frac{M_m \rho M_m^\dagger}{\operatorname{tr}(M_m^\dagger M_m \rho)}.
\end{aligned}$$

A quantum system whose state $|\psi\rangle$ is known exactly is said to be in a **pure state**. In this case, the density operator is simply $\rho = |\psi\rangle\langle\psi|$.
Otherwise, ρ is in a **mixed state**; it is said to be a mixture of the different pure states in the ensemble for ρ.

Properties of the density operator

The class of density operators are characterised by the following useful theorem:

Theorem 6.2. *An operator ρ is the density operator associated to some ensemble $\{p_i, |\psi_i\rangle\}$ if and only if it satisfies the conditions:*

(1) Trace condition: $\operatorname{tr}(\rho) = 1$.

(2) Positivity condition: ρ *is a positive operator.*

Proof.
$\boxed{\Rightarrow}$ Suppose $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$ is a density operator. Then, by linearity of trace,

$$\operatorname{tr}(\rho) = \sum_i p_i \operatorname{tr}(|\psi_i\rangle\langle\psi_i|) = \sum_i p_i = 1,$$

so the trace condition $\operatorname{tr}(\rho) = 1$ is satisfied. Let $|\phi\rangle$ be an arbitrary vector in state space. Then

$$\begin{aligned}
\langle\phi|\rho|\phi\rangle &= \langle\phi| \left(\sum_i p_i \left|\psi_i\right\rangle \left\langle\psi_i\right| \right) |\phi\rangle \\
&= \sum_i p_i \langle\phi|\psi_i\rangle \langle\psi_i|\phi\rangle \\
&= \sum_i p_i |\langle\phi|\psi_i\rangle|^2 \geq 0,
\end{aligned}$$

so the positivity condition is satisfied.
$\boxed{\Leftarrow}$ Suppose ρ is any operator satisfying the trace and positivity conditions. Since ρ is positive, it has a spectral decomposition

$$\rho = \sum_j \lambda_j \left|j\right\rangle \left\langle j\right|,$$

where the vectors $|j\rangle$ are orthogonal, and λ_j are real, non-negative eigenvalues of ρ.

From the trace condition, we see that $\sum_j \lambda_j = 1$. Therefore, a system in state $|j\rangle$ with probability λ_j will have density operator ρ. That is, the ensemble $\{\lambda_j, |j\rangle\}$ is an ensemble of states giving rise to the density operator ρ. □

Partial trace

Definition 6.3. Suppose we have physical systems A and B, whose state is described by a density operator ρ^{AB}. The **reduced density operator** for system A is defined by

$$\rho^A = \mathrm{tr}_B(\rho^{AB})$$

where tr_B is the **partial trace** over system B, defined by

$$\mathrm{tr}_B(|a_1\rangle\langle a_2| \otimes |b_1\rangle\langle b_2|) = |a_1\rangle\langle a_2| \,\mathrm{tr}(|b_1\rangle\langle b_2|)$$

where $|a_1\rangle, |a_2\rangle$ are any two vectors in the state space of A, $|b_1\rangle, |b_2\rangle$ are any two vectors in the state space of B.

The trace operation appearing on the RHS is the usual trace operation for system B, so $\mathrm{tr}(|b_1\rangle\langle b_2|) = \langle b_2|b_1\rangle$.

Example. Suppose a quantum system is in the product state $\rho^{AB} = \rho \otimes \sigma$, where ρ is a density operator for system A, and σ is a density operator for system B. Then

$$\rho^A = \mathrm{tr}_B(\rho \otimes \sigma) = \rho\,\mathrm{tr}(\sigma) = \rho,$$

which is the result we intuitively expect. Similarly, we have $\rho^B = \sigma$.

6.3 Quantum operations

Just as a classical state is described by a vector of probabilities, we describe quantum states in terms of the density operator (density matrix) ρ. Quantum states transform as

$$\rho' = \mathcal{E}(\rho)$$

where the map $\mathcal{E}$ is a **quantum operation**. Two simple examples of quantum operations on density matrices which we have encountered previously (see previous subsection) are unitary transformations $\mathcal{E}(\rho) = U\rho U^\dagger$ and measurements $\mathcal{E}_m(\rho) = M_m \rho M_m^\dagger$.

7 Error Correction

7.1 Classical error-correction

Suppose we wish to send a bit from one location to another through a noisy binary symmetric channel. To protect a bit b against the effects of noise, we replace the bit we wish to protect with three copies of itself, known as **repetition code**:

$$b \mapsto bbb.$$

These two binary strings 000 and 111 are called **codewords**. Beforehand, Alice and Bob agree that, from now on, each time you want to send a logical 0, you will encode it as 000, and each time you want to send a logical 1, you will encode it as 111.

To decode the encoded bit b from the (possibly corrupted) 3-bit codeword, we take the majority value of the 3 bits; this type of decoding is called **majority voting**.

$$1011 \xrightarrow{\text{encoding}} 111\,000\,111\,111 \xrightarrow{\text{noise}} 110\,010\,101\,100 \xrightarrow{\text{decoding}} 1010$$

In order for an error to occur, it is necessary that either two of the three bits have been flipped (which can occur in three different ways), or all three have been, that is:

$$3p^2(1-p) + p^3.$$

Hence, the error-rate has been reduced if $p < \frac{1}{2}$.

We might try a quantum repetition code:

$$|\psi\rangle \to |\psi\rangle \otimes |\psi\rangle \otimes |\psi\rangle\,.$$

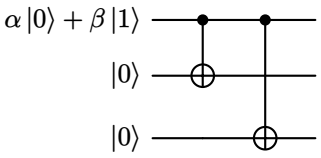
However, no such code exists because of the no-cloning theorem, which states that there is no quantum operation that takes $|\psi\rangle \to |\psi\rangle \otimes |\psi\rangle$ for all states $|\psi\rangle$.

7.2 Three-qubit bit-flip code

In the three-qubit bit-flip code, we encode the computational basis states

$$|0\rangle \to |000\rangle$$
$$|1\rangle \to |111\rangle$$

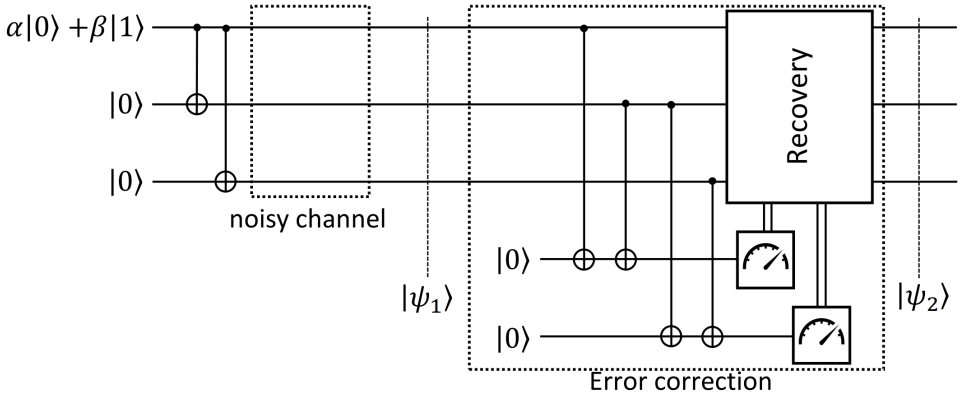
which is achieved using the following circuit that introduces two auxiliary qubits $|0\rangle\,|0\rangle$ and applies two CNOT gates:



This has the following action on an arbitrary qubit state:

$$(\alpha\,|0\rangle + \beta\,|1\rangle)\,|0^2\rangle \to \alpha\,|000\rangle + \beta\,|111\rangle\,.$$

To detect and recover errors, we supplement the circuit with two ancillas that we use for error detection:



We can thus detect and recover single-qubit bit-flips:

Bit-flip error location	$\|\psi_1\rangle$	M_1	M_2	Recovery	$\|\psi_2\rangle$
No error	$\alpha\|000\rangle + \beta\|111\rangle$	0	0	$I \otimes I \otimes I$	$\alpha\|000\rangle + \beta\|111\rangle$
Qubit 1	$\alpha\|100\rangle + \beta\|011\rangle$	1	0	$X \otimes I \otimes I$	$\alpha\|000\rangle + \beta\|111\rangle$
Qubit 2	$\alpha\|010\rangle + \beta\|101\rangle$	1	1	$I \otimes X \otimes I$	$\alpha\|000\rangle + \beta\|111\rangle$
Qubit 3	$\alpha\|001\rangle + \beta\|110\rangle$	0	1	$I \otimes I \otimes X$	$\alpha\|000\rangle + \beta\|111\rangle$

where M_1 and M_2 represent the measurement outcome of the first and second ancilla respectively.

7.3 Three-qubit phase-flip code

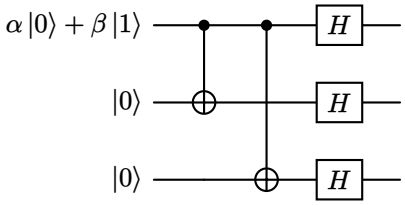
However, we still have not addressed the fact that quantum states are continuous. To begin to do this, we look at an error correction code for phase-flip errors. We encode the computational basis states

$$|0\rangle \to |+++\rangle$$
$$|1\rangle \to |---\rangle$$

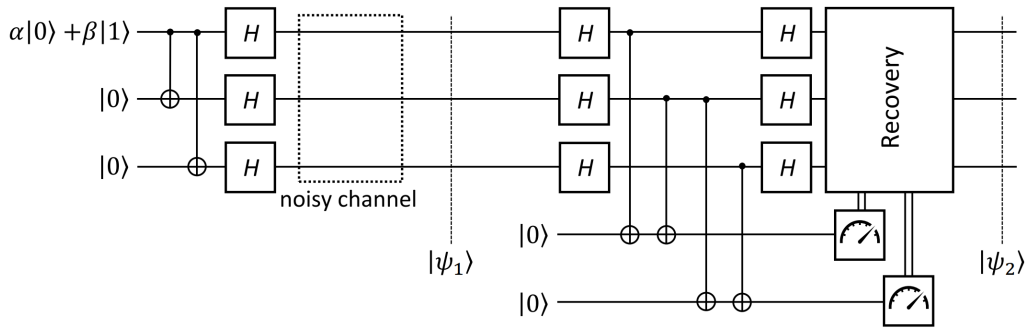
The three-qubit phase-flip code has the following action on an arbitrary single-qubit state:

$$(\alpha|0\rangle + \beta|1\rangle)|0^2\rangle \to \alpha|+++\rangle + \beta|---\rangle$$

which is achieved by the following circuit:



Once again, to detect and recover errors, we supplement the circuit with two ancillas that we use for error detection:



Recall that a phase-flip sends $|+\rangle \to |-\rangle$, $|-\rangle \to |+\rangle$. Thus, we have:

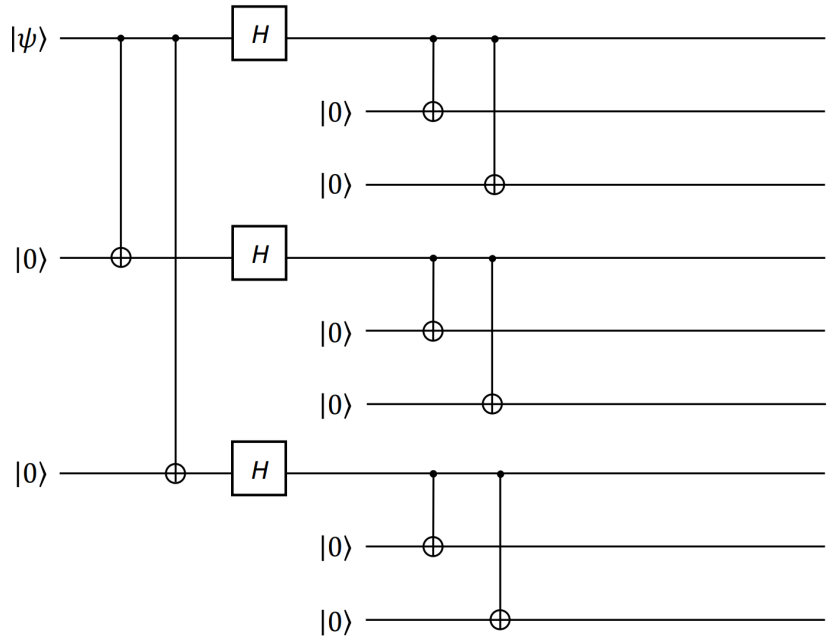
Bit-flip error location	$\|\psi_1\rangle$	M_1	M_2	Recovery	$\|\psi_2\rangle$
No error	$\alpha\|+++\rangle+\beta\|---\rangle$	0	0	$I\otimes I\otimes I$	$\alpha\|+++\rangle+\beta\|---\rangle$
Qubit 1	$\alpha\|-++\rangle+\beta\|+--\rangle$	1	0	$Z\otimes I\otimes I$	$\alpha\|+++\rangle+\beta\|---\rangle$
Qubit 2	$\alpha\|+-+\rangle+\beta\|-+-\rangle$	1	1	$I\otimes Z\otimes I$	$\alpha\|+++\rangle+\beta\|---\rangle$
Qubit 3	$\alpha\|++-\rangle+\beta\|--+\rangle$	0	1	$I\otimes I\otimes Z$	$\alpha\|+++\rangle+\beta\|---\rangle$

7.4 Shor code

Shor code is a 9-qubit code that encodes 1 logical qubit into 9 physical qubits, and can protect against the effects of an *arbitrary* error on a single qubit.

We first encode the qubit using the phase-flip code: $|0\rangle \to |+++\rangle$, $|1\rangle \to |---\rangle$. Next, we encode each of these qubits using the bit-flip code: $|+\rangle$ is encoded as $(|000\rangle+|111\rangle)/\sqrt{2}$, and $|-\rangle$ is encoded as $(|000\rangle-|111\rangle)/\sqrt{2}$. Thus, the **codewords** are

$$|0\rangle \to |\bar{0}\rangle = \frac{1}{2\sqrt{2}}(|000\rangle+|111\rangle)(|000\rangle+|111\rangle)(|000\rangle+|111\rangle),$$
$$|1\rangle \to |\bar{1}\rangle = \frac{1}{2\sqrt{2}}(|000\rangle-|111\rangle)(|000\rangle-|111\rangle)(|000\rangle-|111\rangle)$$

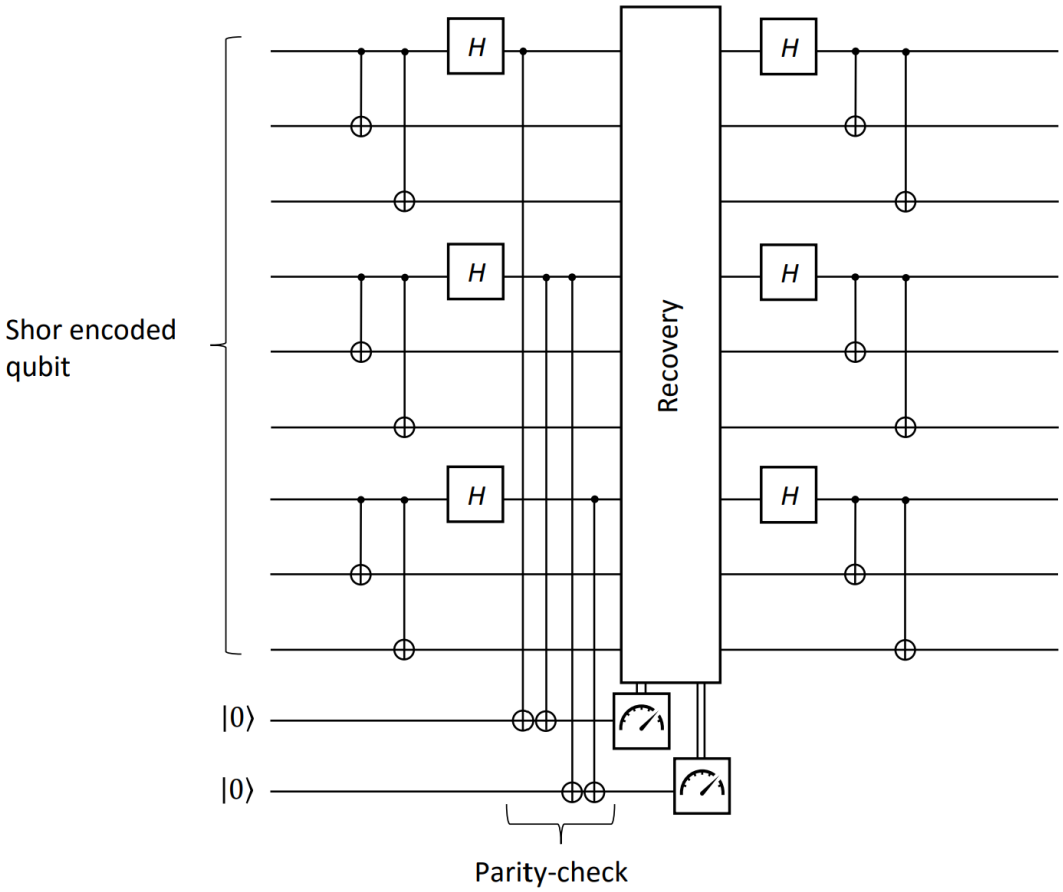


These two quantum codewords $|\bar{0}\rangle$ and $|\bar{1}\rangle$ span a 2-dimensional space $\{\alpha|\bar{0}\rangle+\beta|\bar{1}\rangle\}$. This 2-dimensional subspace of the overall 2^9-dimensional space is called the **codespace**.

Suppose an error happens on one of these 9 qubits. We would like to have a procedure that maps the resulting state back to the codespace. By linearity, it suffices if we can do this for the basis states $|0\rangle$ and $|1\rangle$. Our **error correction procedure** is as follows:

- The inner layer (addition/subtraction) of this code corrects *bit flip* errors: We take the majority within each set of three, so
$$|010\rangle \pm |101\rangle \to |000\rangle \pm |111\rangle\,.$$
- The outer layer (tensor products) corrects *phase flip* errors: We take the majority of the three signs, so
$$(|\cdot\rangle + |\cdot\rangle)(|\cdot\rangle - |\cdot\rangle)(|\cdot\rangle + |\cdot\rangle) \to (|\cdot\rangle + |\cdot\rangle)(|\cdot\rangle + |\cdot\rangle)(|\cdot\rangle + |\cdot\rangle).$$

Since these two error correction steps are independent, the code also works if there is both a bit flip error *and* a phase flip error.



8 Convex optimization using quantum oracles [JW18]

The generic problem in convex optimization is minimizing a convex function $f\colon K \to \mathbb{R}$, where $K \subseteq \mathbb{R}^n$ is a convex set.

We consider the setting where an interior point $x_0 \in \operatorname{int}(K)$ is given, and radii $r, R > 0$ are known such that $B(x_0, r) \subseteq K \subseteq B(x_0, R)$.

Time complexity: $\tilde{O}$ is a variant of big O notation that ignores logarithmic factors, so $f(n) \in \tilde{O}(g(n))$ means $f(n) \in O(g(n) \log^k n)$ for some k.

Summary

- Section 3: Quantum approach to compute an approximate subgradient of a convex Lipschitz function at origin using $O(\log(n))$ queries; it picks a random point near origin, runs Jordan's algorithm to get an approximate gradient at that point, and finally shows that this gradient is an approximate subgradient at origin. Classical approach requires $O(n)$ queries.
- Section 4: Implement a separation oracle using $\tilde{O}(1)$ quantum queries to a membership oracle. Classical approach requires $\Omega(n)$ membership queries.
- Section 5: $\tilde{O}(n)$ quantum queries to a membership oracle suffice to implement an optimization oracle; the best known classical upper bound on the number of membership queries is quadratic.

  Also proves several lower bounds: $\Omega(\sqrt{n})$ quantum separation queries needed for optimization if the algorithm knows an interior point of the convex set; $\Omega(n)$ quantum separation queries are needed if it does not.

8.1 Preliminaries

Convex optimization

For a vector $x \in \mathbb{R}^n$, the **ℓ_p-norm** of x is

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}.$$

If $p = 2$, then the subscript is sometimes omitted. For $p \geq 1$, $\varepsilon \geq 0$, and a set $C \subseteq \mathbb{R}^n$, let

$$B_p(C, \varepsilon) = \{\, x \in \mathbb{R}^n : \exists y \in C \text{ such that } \|x - y\|_p \leq \varepsilon \,\}$$

be the set of points of distance at most ε from C in the ℓ_p-norm. When $C = \{x\}$ is a singleton, we abuse notation and write $B_p(x, \varepsilon)$. We overload notation by setting

$$B_p(C, -\varepsilon) = \{\, x \in \mathbb{R}^n : B_p(x, \varepsilon) \subseteq C \,\}$$

which is the set of points whose ε-ball is contained in C.

Let $C \subseteq \mathbb{R}^n$. A function $f\colon C \to \mathbb{R}$ is **K-Lipschitz** if there exists a constant $K > 0$ such that

$$|f(x) - f(y)| \leq K\|x - y\| \quad \text{for all } x, y \in C.$$

Definition 8.1. A set $C \subseteq \mathbb{R}^n$ is **affine** if the *line* through any two points in C lies in C, i.e., for any $x, y \in C$ and $\lambda \in \mathbb{R}$,

$$\lambda x + (1 - \lambda) y \in C.$$

More generally, a point of the form $\lambda_1 x_1 + \cdots + \lambda_k x_k$, where $\lambda_1 + \cdots + \lambda_k = 1$ is called an **affine combination** of the points $x_1, \ldots, x_k$. By induction from the definition of affine set, it follows that

an affine set contains every affine combination of its points: If C is an affine set, $x_1, \dots, x_k \in C$, and $\lambda_1 + \cdots + \lambda_k = 1$, then $\lambda_1 x_1 + \cdots + \lambda_k x_k \in C$.

Definition 8.2. A set $C \subseteq \mathbb{R}^n$ is **convex** if the *line segment* between any two points in C lies in C, i.e., for any $x, y \in C$ and $\lambda \in [0,1]$,

$$\lambda x + (1-\lambda) y \in C.$$

Obviously every affine set is also convex.

Similarly, a point of the form $\lambda_1 x_1 + \cdots + \lambda_k x_k$, where $\lambda_1 + \cdots + \lambda_k = 1$, $\lambda_i \geq 0$ is called a **convex combination** of the points $x_1, \dots, x_k$.

A **hyperplane** is a set of the form

$$\{x : a^T x = b\}$$

where $a \in \mathbb{R}^n$, $a \neq 0$, $b \in \mathbb{R}$. Analytically, it is the solution set of a non-trivial linear equation among the components of x. Rewritten the hyperplane as

$$\{x : a^T (x - x_0) = 0\}$$

where x_0 is any point in the hyperplane, geometrically it is a plane with normal vector a; the constant b determines the offset of the hyperplane from the origin.

A hyperplane divides $\mathbb{R}^n$ into two halfspaces. A (closed) **halfspace** is a set of the form

$$\{x : a^T x \leq b\}$$

where $a \neq 0$, i.e., the solution set of one (non-trivial) linear inequality. Halfspaces are convex, but not affine. Similarly, the halfspace can also be expressed as

$$\{x : a^T (x - x_0) \leq 0\},$$

where x_0 is any point on the associated hyperplane.

Definition 8.3. A function $f : C \to \mathbb{R}$ is **convex** if for all $x, y \in C$ and $\lambda \in [0,1]$,

$$f(\lambda x + (1-\lambda) y) \leq \lambda f(x) + (1-\lambda) f(y).$$

Geometrically, the line segment between $(x, f(x))$ and $(y, f(y))$ lies above the graph of f.

We say that f is **concave** if $-f$ is convex.

Note that a function is affine if and only if it is convex and concave.

By the second derivative test, convexity or concavity of a function can also be shown by checking that the second derivative is non-negative or non-positive.

Definition 8.4. The **convex hull** of a set C is the set of all convex combinations of points in C:

$$\operatorname{conv}(C) := \{\, \theta_1 x_1 + \cdots + \theta_k x_k : x_i \in C, \theta_i \geq 0, i = 1, \dots, k, \theta_1 + \cdots + \theta_k = 1 \,\}.$$

As the name suggests, the convex hull $\operatorname{conv}(C)$ is always convex. It is the smallest convex set that contains C: If B is any convex set that contains C, then $\operatorname{conv}(C) \subseteq B$.

In general, a convex optimisation problem asks to minimise a convex function $f(x)$ over a convex set C:

$$\inf_{x \in C} f(x).$$

Here f is called the **objective function**. We call the value of the infimum the **optimal value** of the problem, and an x that attains this value is called an **optimiser**. We often write x^* for such an optimiser if the infimum is attained.

Instead of only considering f we may also consider its derivatives. Of particular interest is the first derivative, the **gradient**; $-\nabla f$ tells us in which direction f decreases most.

Theorem 8.5 (First-order convexity condition)**.** *Suppose f is differentiable (i.e., its gradient ∇f exists at each point in* $\operatorname{dom} f$*). Then f is convex if and only if* $\operatorname{dom} f$ *is convex, and*

$$f(y) \geq f(x) + \nabla f(x)^T (y - x)$$

holds for all $x, y \in \operatorname{dom} f$.

Fix $x \in \operatorname{dom} f$. Geometrically, $f(x) + \nabla f(x)^T (y - x)$ defines a hyperplane[4] that is tangent to f at x (where y is an arbitrary point on the hyperplane). Since f is convex, the hyperplane lies below the graph of $f(y)$ for all y.

Proof. See [BV09], p. 70. □

Since f may not have a derivative at every point of the domain, we have the following more general notion of a subgradient.

Definition 8.6. Let $C \subseteq \mathbb{R}^n$ be convex and let x be an element of the interior of C. For a convex function $f \colon C \to \mathbb{R}$, a **subgradient** of f at $x \in C$ is a vector $g \in \mathbb{R}^n$ satisfying

$$f(y) \geq f(x) + g^T (y - x) \quad \text{for all } y \in C.$$

We denote by $\partial f(x)$ the set of subgradients of f at x, called the **subdifferential** of f at x.

Note that in the above definition $\partial f(x) \neq \emptyset$ due to convexity; if $\nabla f(x)$ exists, $\partial f(x) = \{\nabla f(x)\}$.

Example. Consider $f(x) = |x|$, which is not differentiable at $x = 0$.

$$\partial f(x) = \begin{cases} \{1\} & x > 0, \\ [-1, 1] & x = 0, \\ \{-1\} & x < 0. \end{cases}$$

If $f \colon C \to \mathbb{R}$ is K-Lipschitz, then for any x in the interior of C and any $g \in \partial f(x)$, we have $\|g\| \leq K$, as follows. Consider a $y \in C$ such that $y - x = \alpha g$ for some $\alpha > 0$. Then since g is a subgradient of f at x,

$$\alpha \|g\|^2 = g^T (y - x) \leq f(y) - f(x) \leq K \|y - x\| = \alpha K \|g\|$$

and therefore $\|g\| \leq K$.

Often the convex set C is described via a set of m convex functions which should be smaller than some constant, these inequalities are called **constraints** (which define the "boundaries" of C):

$$C = \{\, x \in \mathbb{R}^n : h_i(x) \leq b_i \text{ for } i = 1, \ldots, m \,\}.$$

By absorbing the b_i into the h_i we can assume without loss of generality that the RHS are all zero. We can now write the optimisation problem as

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & h_1(x) \leq 0 \\ & \vdots \\ & h_m(x) \leq 0. \end{aligned}$$

A point that satisfies all constraints is called a **feasible point**. An optimisation problem that has a feasible point is called a **feasible problem**.

[4] For a function $z = f(x, y)$, the equation of tangent plane at a point (x_0, y_0, z_0) is $z = z_0 + f_x(x_0, y_0)(x - x_0) + f_y(x_0, y_0)(y - y_0)$, which we can rearrange to get $z = f(x_0) + \begin{pmatrix} f_x(x_0, y_0) \\ f_y(x_0, y_0) \end{pmatrix} \cdot \begin{pmatrix} x - x_0 \\ y - y_0 \end{pmatrix}$.

A class of convex optimisation problems is the class of **linear programming** problems (LPs). These are optimisation problems where both f and all the h_i are linear functions.

Remark. Since convexity is more general than linearity, any linear program is therefore a convex optimisation problem, so we can consider convex optimisation to be a generalisation of linear programming.

Since each h_i is a linear function, we can write $h_i(x) = a_i^T x$ for some $a_i \in \mathbb{R}^n$. Without loss of generality, we assume LPs are of the form

$$\begin{aligned} \max_{x \in \mathbb{R}^n} \quad & c^T x \\ \text{s.t.} \quad & a_i^T x \leq b_i, \quad i = 1, \ldots, m \\ & x \geq 0. \end{aligned}$$

For normalisation purposes, assume that all entries of the a_i and c are in $[-1, 1]$.

Remark. Since linear functions are both convex and concave, we may assume without loss of generality that we want to maximise the objective.

If we let $A \in \mathbb{R}^{m \times n}$ be the matrix whose rows are $a_1, \ldots, a_m$, we can rewrite the LP problem concisely as

$$\begin{aligned} \max_{x \in \mathbb{R}^n} \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b \\ & x \geq 0. \end{aligned}$$

Oracles for convex sets

Recall that a function evaluation oracle for a function $f \colon X \to Y$ maps $|x\rangle |0\rangle$ to $|x\rangle |f(x)\rangle$, where $|x\rangle$ and $|f(x)\rangle$ are basis states corresponding to binary representations of x and $f(x)$ respectively.

The standard quantum oracle described above models problems where there is a single correct answer to a query. When there are multiple good answers (for instance, different good approximations to the correct value) and the oracle is only required to give a correct answer with high probability, then we work with the more general notion of a *relational quantum oracle*.

Definition 2: Let $\mathcal{F} \colon X \to \mathcal{P}(Y)$ be a function that maps each $x \in X$ to the subset $\mathcal{F}(x) \subseteq Y$, is the set of valid answers to an x query. A **relational quantum oracle** for $\mathcal{F}$ which answers queries with success probability $\geq 1 - \rho$, is a unitary that for all $x \in X$ maps

$$U \colon |x\rangle |0\rangle |0\rangle \mapsto \sum_{y \in Y} \alpha_{x,y} |x\rangle |y\rangle |\psi_{x,y}\rangle$$

where $|\psi_{x,y}\rangle$ denotes some normalised quantum state and $\sum_{y \in \mathcal{F}(x)} |\alpha_{x,y}|^2 \geq 1 - \rho$. Thus measuring the second register of $U |x\rangle |0\rangle |0\rangle$ gives a valid answer to the x query with probability at least $1 - \rho$.

Remark. The name "relational" stems from a *relation*, where an element from the domain can map to multiple elements in the range.

The setup above is as follows:

- the first register holds the input query;
- the second register holds a superposition of many possible answers y with amplitudes $\alpha_{x,y}$;
- the third register is an auxiliary register.

Since $\mathcal{F}(x)$ is the set of valid answers, and $|\alpha_{x,y}|^2$ is the probability of obtaining outcome y upon measuring the second register, summing up $|\alpha_{x,y}|^2$ for all $y \in \mathcal{F}(x)$ gives the total probability of the state collapsing into a correct answer, which is why we require $\sum_{y \in \mathcal{F}(x)} |\alpha_{x,y}|^2 \geq 1 - \rho$.

A convex set can be given explicitly (by a set of constraints) or implicitly (by an oracle). The five basic oracles for a convex set K that we consider are as follows, where we allow some geometric error ε and error probability ρ.

Definition 3: Queried with a vector $y \in \mathbb{R}^n$, the **membership oracle** $\mathrm{MEM}_{\varepsilon,\rho}(K)$, with success probability $\geq 1-\rho$, correctly asserts one of the following

- $y \in B(K, \varepsilon)$, or
- $y \notin B(y, -\varepsilon)$.

Given a point, the membership oracle checks whether it is either inside (or ε-near) or outside the convex set.

Definition 4: Queried with a vector $y \in \mathbb{R}^n$, the **separation oracle** $\mathrm{SEP}_{\varepsilon,\rho}(K)$, with success probability $\geq 1-\rho$, correctly asserts one of the following

- $y \in B(K, \varepsilon)$, or
- $y \notin B(K, -\varepsilon)$;

in the second case, it returns a unit vector $g \in \mathbb{R}^n$ such that $\langle g, x\rangle \leq \langle g, y\rangle + \varepsilon$ for all $x \in B(K, -\varepsilon)$.
Similar to the membership oracle, the separation oracle checks if a given point is inside or outside the convex set. Since g is a unit vector, $\langle g, y\rangle$ is the projection of y along g. Considering $\langle g, y\rangle + \varepsilon$ as some fixed value, we see that $\langle g, x\rangle \leq \langle g, y\rangle + \varepsilon$ for all $x \in B(K, -\varepsilon)$ forms a half-space.

Definition 5: Queried with a unit vector $c \in \mathbb{R}^n$, the **optimization oracle** $\mathrm{OPT}_{\varepsilon,\rho}(K)$, with probability $\geq 1-\rho$, does one of the following:

- it returns a vector $y \in \mathbb{R}^n$ such that $y \in B(K, \varepsilon)$ and $\langle c, x\rangle \leq \langle c, y\rangle + \varepsilon$ for all $x \in B(K, -\varepsilon)$,
- or it correctly asserts that $B(K, -\varepsilon)$ is empty.

This oracle is designed to find the best possible point in the convex set along a specific direction (maximizing a linear function). Given a direction vector c, the oracle returns a vector y such that no other point x inside the set can give a larger dot product with c than this y (with margin of error ε).

Definition 6: Queried with a unit vector $c \in \mathbb{R}^n$ and a real number γ, the **violation oracle** $\mathrm{VIOL}_{\varepsilon,\rho}(K)$, with probability $\geq 1-\rho$, does one of the following:

- it asserts that $\langle c, x\rangle \leq \gamma + \varepsilon$ for all $x \in B(K, -\varepsilon)$,
- or it finds a vector $y \in B(K, \varepsilon)$ such that $\langle c, y\rangle \geq \gamma - \varepsilon$.

Given a direction vector c and a threshold γ, the oracle checks whether the projection of every x in the direction of c exceeds the threshold.

Definition 7: Queried with a unit vector $c \in \mathbb{R}^n$ and a real number γ, the **validity oracle** $\mathrm{VAL}_{\varepsilon,\rho}(K)$, with probability $\geq 1-\rho$, does one of the following:

- it asserts that $\langle c, x\rangle \leq \gamma + \varepsilon$ for all $x \in B(K, -\varepsilon)$,
- or it asserts that $\langle c, y\rangle \geq \gamma - \varepsilon$ for some $y \in B(K, \varepsilon)$.

This is similar to the violation oracle, but does not return a specific vector if a violation occurs.
If $\varepsilon = \rho = 0$, then the oracle is called **strong**; otherwise, it is called **weak**.

8.2 Subgradient approximation for convex Lipschitz functions

Here we show how to compute an approximate subgradient (at 0) of a convex Lipschitz function. That is, given a convex set C such that $0 \in \text{int}(C)$ and a convex function $f\colon C \to \mathbb{R}$, we show how to compute a vector $\tilde{g} \in \mathbb{R}^n$ such that

$$f(y) \geq f(0) + \langle \tilde{g}, y \rangle - a\|y\| - b$$

for some real numbers $a, b > 0$, which are coefficients of the error term.

Classical approach

Definition 8 (Approximation of gradient): For a function $f\colon C \to \mathbb{R}$, a real $r > 0$, and a point $x \in \mathbb{R}^n$ such that $B_1(x, r) \subseteq C$, the partial derivative approximation[5] along coordinate i is

$$\nabla_i^{(r)} f(x) := \frac{f(x + re_i) - f(x - re_i)}{2r}.$$

Collecting all n coordinate approximations in a single vector, we define the finite-difference **gradient approximation**

$$\nabla^{(r)} f(x) := \begin{pmatrix} \nabla_1^{(r)} f(x) \\ \vdots \\ \nabla_n^{(r)} f(x) \end{pmatrix}.$$

Comparing $\nabla_i^{(r)} f(x)$ with the partial derivative $\frac{\partial f}{\partial x_i}(x) = \lim_{r \to 0} \frac{f(x+re_i) - f(x)}{r}$, we see that $\nabla_i^{(r)} f(x)$ is indeed an approximation for the partial derivative.

Definition 9 (Approximation of the Laplacian): For a function $f\colon C \to \mathbb{R}$, a real $r > 0$, and a point $x \in \mathbb{R}^n$ such that $B_1(x, r) \subseteq C$, and $i \in [n]$, define

$$\Delta_i^{(r)} f(x) := \frac{f(x + re_i) - 2f(x) + f(x - re_i)}{r^2}.$$

Then define the finite-difference **Laplace approximation** as

$$\Delta^{(r)} f(x) := \sum_{i=1}^{n} \Delta_i^{(r)} f(x).$$

Again, compare this with the definition of the Laplacian, $\Delta f = \sum_{i=1}^n \frac{\partial^2 f}{\partial x_i^2}$.

Note: for a convex function we have $\Delta_i^{(r)} f(x) \geq 0$ for all x such that $B_1(x, r) \subseteq C$, because

$$f(x) = f\left(\frac{(x + re_i) + (x - re_i)}{2}\right) \leq \frac{f(x + re_i) + f(x - re_i)}{2}.$$

Lemma 10: Bound the deviation $\left\| g - \nabla^{(r_2)} f(z) \right\|_1$ of a finite-difference gradient approximation $\nabla^{(r_2)} f(z)$ from an actual subgradient g at a point z, in terms of finite-difference Laplace approximation $\nabla^{(r_2)} f(z)$.

Lemma 11: In expectation (over the points of a small ball around x), the finite difference Laplace approximation is small.

Lemma 12 (main result): Let f be convex Lipschitz function. Then $\nabla^{(r_2)} \tilde{f}(z)$ for a random z close enough to 0, where $\tilde{f}$ is an approximate version of f, is an approximate subgradient of f at 0.

[5] See https://adamdjellouli.com/articles/numerical_methods/3_differentiation/central_difference. for the smaller error using central difference method compared to forward or backward difference methods

Since computing each $\nabla_i^{(r)} \tilde{f}$ requires 2 queries to $\tilde{f}$, in total we need $2n$ classical queries to $\tilde{f}$.

Quantum improvements

Jordan's quantum gradient computation algorithm [Jor05] approximately calculates the gradient of $f\colon \mathbb{R}^n \to \mathbb{R}$ with a single evaluation of f. In contrast, using standard classical techniques, since

$$\nabla_i f(x) := \lim_{\delta \to 0} \frac{f(x+\delta e_i) - f(x)}{\delta} \approx \frac{f(x+\delta e_i) - f(x)}{\delta}$$

for some small $\delta > 0$, one would need $n+1$ function evaluations to calculate the gradient at a point $x \in \mathbb{R}^n$. By Taylor's theorem, if f is twice differentiable at x, then for a tiny displacement δ,

$$f(x+\delta) = f(x) + \nabla f \cdot \delta + O(\|\delta\|^2).$$

Thus, for small $\|\delta\|$, the function f is locally affine. Using the value of $f(x+\delta)$, one can implement a phase oracle

$$O_{2\pi S f}\colon |\delta\rangle \to e^{2\pi i S f(x+\delta)} |\delta\rangle \approx e^{2\pi i S f(x)} e^{2\pi i S \nabla f \cdot \delta} |\delta\rangle$$

for some scaling factor $S > 0$. Note that $e^{2\pi i S f(x)}$ is a global phase which can be ignored, and $e^{2\pi i S(\nabla f \cdot \delta)}$ is a relative phase proportional to $\nabla f \cdot \delta$.

Definition 13: We discretize the interval $(-1/2, 1/2)$ into 2^m points, so that we perform computations on m qubits. For a binary string $b = (b_1, \ldots, b_m) \in \{0,1\}^m$, let $j^{(b)} \in \{0, \ldots, 2^m - 1\}$ be its decimal value, so $j^{(b)} = b_1 2^{m-1} + \cdots + 2^1 b_{m-1} + b_m$. The grid point $x^{(b)}$ is defined as

$$x^{(b)} = \frac{j^{(b)}}{2^m} - \frac{1}{2} + 2^{-m-1}$$

where the term $+2^{-m-1}$ shifts each point by half a cell width; the effect of this is centering all grid points inside the open interval $(-1/2, 1/2)$. The set of 2^m grid points is denoted G_m. The n-dimensional grid is the n-fold Cartesian product

$$G_m^n = \underbrace{G_m \times \cdots \times G_m}_{n \text{ times}}.$$

Since we need m bits to specify a point in each G_m, an element of G_m^n can be represented using $m \times n$ (qu)bits.

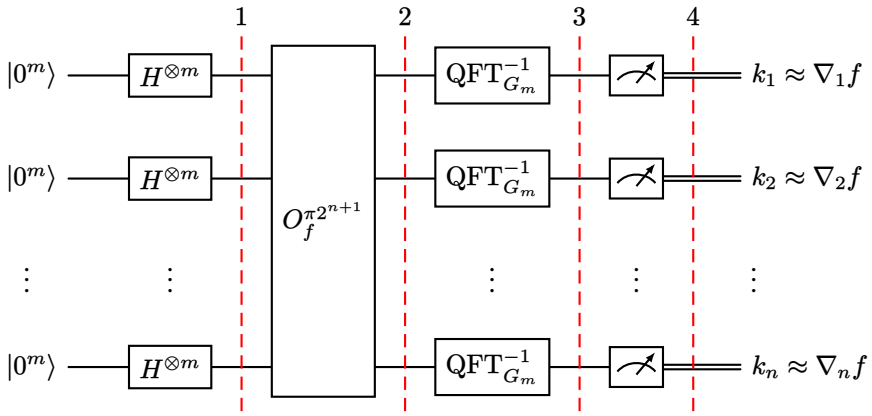
Remark. Think of m as controlling how fine the resolution of the hyper-grid is, which affects the accuracy of the output of Jordan's algorithm.

Recall the (usual) quantum Fourier transform QFT_N acting on integer basis states $|j^{(b)}\rangle$ for $N = 2^m$:

$$\mathrm{QFT}_N\colon |j^{(b)}\rangle \mapsto \frac{1}{\sqrt{2^m}} \sum_{j^{(c)} \in \{0,\ldots,2^m-1\}} e^{2\pi i 2^{-m} j^{(b)} j^{(c)}} |j^{(c)}\rangle .$$

Define the "grid-adapted" Fourier transform QFT_{G_m} which acts on coordinate labels $|x^{(b)}\rangle$:

$$\mathrm{QFT}_{G_m}\colon |x^{(b)}\rangle \mapsto \frac{1}{\sqrt{2^m}} \sum_{x^{(c)} \in G_m} e^{2\pi i 2^m x^{(b)} x^{(c)}} |x^{(c)}\rangle .$$



(1) Prepare n registers containing m qubits each, initialized to $|0^m\rangle$. Apply a Hadamard transform $H^{\otimes m}$ to each register to create a uniform superposition.

For a single m-qubit register starting in $|0^m\rangle$, applying $H^{\otimes m}$ results in

$$H^{\otimes m}|0^m\rangle = \frac{1}{\sqrt{2^m}} \sum_{j=0}^{2^m-1} |j\rangle = \frac{1}{\sqrt{2^m}} \sum_{x \in G_m} |x\rangle$$

where we identify each integer basis state $|j\rangle$ with the corresponding point $x \in G_m$ on the hyper-grid. Since there are n registers, the joint state is the tensor product

$$\bigotimes_{i=1}^{n} \left(\frac{1}{\sqrt{2^m}} \sum_{x_i \in G_m} |x_i\rangle \right) = \frac{1}{\sqrt{|G_m^n|}} \sum_{\mathbf{x} \in G_m^n} |x_1\rangle \cdots |x_n\rangle$$

using $|G_m^n| = (2^m)^n = 2^{mn}$.

(2) Apply the oracle $O_f^{\pi 2^{m+1}}$ with scale factor $S = 2^m$:

$$O_f^{\pi 2^{m+1}} |x_1\rangle \cdots |x_n\rangle = e^{2\pi i 2^m f(x_1,\ldots,x_n)} |x_1\rangle \cdots |x_n\rangle$$

so that the resulting state is

$$\frac{1}{\sqrt{|G_m^n|}} \sum_{\mathbf{x} \in G_m^n} e^{2\pi i 2^m f(\mathbf{x})} |x_1\rangle \cdots |x_n\rangle .$$

Since f is locally affine, we approximate $f(x) \approx g \cdot x + c$, where $g = (g_1, \ldots, g_n)$ is the gradient vector and $g \cdot x = \sum_{i=1}^n g_i x_i$. Substituting this into the exponential term gives

$$e^{2\pi i 2^m f(\mathbf{x})} \approx e^{2\pi i 2^m \left(c + \sum_{i=1}^n g_i x_i\right)} = e^{2\pi i 2^m c} \cdot e^{2\pi i 2^m \sum_{i=1}^n g_i x_i} = e^{2\pi i 2^m c} \cdot \prod_{i=1}^{n} e^{2\pi i 2^m g_i x_i}.$$

Hence, the state is approximately

$$\frac{1}{\sqrt{|G_m^n|}} \sum_{\mathbf{x} \in G_m^n} \left(e^{2\pi i 2^m c} \prod_{i=1}^{n} e^{2\pi i 2^m g_i x_i} \right) |x_1\rangle \cdots |x_n\rangle = e^{2\pi i 2^m c} \bigotimes_{i=1}^{n} \left(\frac{1}{\sqrt{2^m}} \sum_{x_i \in G_m} e^{2\pi i 2^m g_i x_i} |x_i\rangle \right).$$

(3) Apply the inverse grid Fourier transform $\mathrm{QFT}_{G_m}^{-1}$ over each register individually:

$$\mathrm{QFT}_{G_m}^{-1} : |x_i\rangle \mapsto \frac{1}{\sqrt{2^m}} \sum_{k_i \in G_m} e^{-2\pi i 2^m x_i k_i} |k_i\rangle .$$

The action of $\mathrm{QFT}_{G_m}^{-1}$ on register i is

$$\mathrm{QFT}_{G_m}^{-1}\left(\frac{1}{\sqrt{2^m}}\sum_{x_i\in G_m} e^{2\pi i 2^m g_i x_i}\,|x_i\rangle\right) = \frac{1}{\sqrt{2^m}}\sum_{x_i\in G_m} e^{2\pi i 2^m g_i x_i}\left(\frac{1}{\sqrt{2^m}}\sum_{k_i\in G_m} e^{-2\pi i 2^m x_i k_i}|k_i\rangle\right)$$
$$= \sum_{k_i\in G_m}\left(\frac{1}{2^m}\sum_{x_i\in G_m} e^{2\pi i 2^m x_i(g_i-k_i)}\right)|k_i\rangle$$

which is a superposition of states $|k_i\rangle$ with amplitude

$$\frac{1}{2^m}\sum_{x_i\in G_m} e^{2\pi i 2^m x_i(g_i-k_i)}.$$

If $k_i = g_i$, the exponent becomes $e^0 = 1$, so the inner sum becomes $\frac{1}{2^m}\cdot 2^m = 1$. If $k_i \neq g_i$, the terms in the inner sum cancel out because they are complex roots of unity.

(4) Measure each input register i and denote the measurement outcome by k_i. Output $(k_1,\ldots,k_n)$ as the estimation for the gradient.

Lemma 14: The difference $|v_i - g_i|$ between the true and approximate gradient is small, so v_i estimates g_i within precision 2^{2-m} with failure probability bounded by $\frac{1}{3}$.
Let $\tilde{f}\colon G_m^n \to \mathbb{R}$ be an approximation of the true function f with maximum error δ:

$$|\tilde{f}(x) - f(x)| \le \delta \quad \text{for all } x \in G_m^n.$$

Assume we are given access to an oracle U that implements $\tilde{f}$:

$$U\colon\ |x\rangle\,|0\rangle \mapsto |x\rangle\,|\tilde{f}(x)\rangle\,.$$

To run Jordan's algorithm using U, we implement $O_f^{2^{n+1}}$ using one call to U and one call to $U^\dagger$:

(1) Start with $|x\rangle\,|0\rangle$. Apply U to get $|x\rangle\,|\tilde{f}(x)\rangle$.

(2) Apply a phase gate that adds a phase angle, so $|x\rangle\,|\tilde{f}(x)\rangle \mapsto e^{2\pi i 2^m \tilde{f}(x)}\,|x\rangle\,|\tilde{f}(x)\rangle$.

(3) Finally, apply $U^\dagger$ to clear the target register back to $|0\rangle$. The final state is $e^{2\pi i 2^m \tilde{f}(x)}\,|x\rangle\,|0\rangle$.

Corollary 15: When using a noisy function oracle (rather than an ideal oracle), Jordan's algorithm can still compute an approximate gradient $\tilde{g}$ with error bound $\|\tilde{g} - g\|_\infty \le O(\delta/r)$, where the error scales with function noise δ, and success probability of $1-\rho$ achieved via $O(\log(n/\rho))$ oracle queries.
Note: instead of working on $G_m^n \subset (-1/2, 1/2)^n$, the target domain is now scaled to an r-neighborhood around x_0, namely, $x_0 + rG_m^n$.
We would like to apply the above corollary to compute gradients of a convex Lipschitz function. To that end, the function needs to be sufficiently close to a linear function on a small region. Fortunately convex Lipschitz functions have this property.

Lemma 16: Let $S \subseteq \mathbb{R}^n$ be a symmetric set around the origin, so that $S = -S$. Let $\mathrm{conv}(S)$ denote the convex hull of S. If $f\colon \mathrm{conv}(S) \to \mathbb{R}$ is a convex function, $f(0) = 0$, and $|f(s)| \le \delta$ for all $s \in S$, then

$$|f(s')| \le \delta \qquad \text{for all } s' \in \mathrm{conv}(S).$$

Lemma 17: If $r_2 > 0$, $z \in \mathbb{R}^n$ and $f\colon B_1(z, r_2) \to \mathbb{R}$ is convex, then the difference between $f(z+y)$ and its linear approximation at z is bounded by

$$\sup_{y \in B_1(0,r_2)} \left| f(z+y) - f(z) - \left\langle y, \nabla^{(r_2)} f(z) \right\rangle \right| \leq \frac{{r_2}^2 \Delta^{(r_2)} f(z)}{2}.$$

(Quantum analogue of Lemma 10.)

Lemma 18 (main result): Use Jordan's quantum gradient algorithm to construct an approximate subgradient at the origin for a convex Lipschitz function using $O(\log(n/\rho)) = \tilde{O}(1)$ quantum queries.

(Quantum analogue of Lemma 12.)

Proof idea: Our algorithm works as follows:

(1) Instead of evaluating f directly at 0, at which f might be non-differentiable, pick a uniformly random $z \in B_\infty(0, r_1)$, so that f is locally linear at z.

(2) Use Jordan's quantum algorithm to compute an approximate gradient $\tilde{g}$ at z by approximately evaluating $\tilde{f}$ in superposition over a discrete hypergrid of $B_\infty(z, r_2/n)$.

(3) Show that $\tilde{g}$ is an approximate subgradient of f at 0.

8.3 Algorithms for separation using membership queries

Let $K \subseteq \mathbb{R}^n$ be a convex set such that $B(0, r) \subseteq K \subseteq B(0, R)$. Given a membership oracle $\mathrm{MEM}_{\varepsilon,0}(K)$, we will construct a separation oracle $\mathrm{SEP}_{\eta,\rho}(K)$.

Let x be the point we want to separate from K. We first make a membership query to x itself, and receive an answer $x \in B(K, \varepsilon)$ or $x \notin B(K, -\varepsilon)$.

Suppose $x \notin B(K, -\varepsilon)$. Then we need to find a hyperplane that approximately separates x from K, i.e., a vector c such that $c \cdot x > c \cdot y$ for all $y \in K$. Due to the rotational symmetry of the separation problem, for ease of notation, we assume that $x = -\|x\| e_n$ by rotating the coordinate system so that the point x lies along the negative n-th coordinate axis.

Define $h\colon \mathbb{R}^{n-1} \to \mathbb{R} \cup \{\infty\}$ as

$$h(y) = \inf_{(y, y_n) \in K} y_n.$$

Note that any point in $\mathbb{R}^n$ can be split into two parts: the horizontal position $y \in \mathbb{R}^{n-1}$ (the first $n-1$ coordinates), and the vertical height $y_n \in \mathbb{R}$ (the last coordinate), so the point is written as (y, y_n).

Remark. h implicitly depends on x, since we rotated the space such that $x = -\|x\| e_n$.

Geometrically, fix $y \in \mathbb{R}^{n-1}$, look vertically above/below y, find all points $(y, y_n) \in K$. Then $h(y)$ is the lowest vertical height where K begins.

Since K is a convex set, h is a convex function over $\mathbb{R}^{n-1}$.

Lemma 19: h is Lipschitz.

Lemma 20: Compute the value of h using membership queries to K, using binary search with $\tilde{O}(1)$ classical queries to a membership oracle.

Proof idea: $h(y)$ lies in $[-R, -r/2]$ of length $O(R)$. If the membership oracle were perfect, binary search could find $h(y)$ to accuracy δ in $O(\log(R/\delta))$ steps.

However, since the membership oracle is noisy near the boundary, we modify the decision criteria at each step of binary search: Suppose y_n is our current guess for $h(y)$. Query (y, y_n) using membership oracle.

(a) If $(y, y_n) \in B(K, \varepsilon)$, then $y_n \geq h(y) - \delta$.

(b) If $(y, y_n) \notin B(K, -\varepsilon)$, then $y_n \leq h(y) + \frac{2}{3}\delta$.

This binary search also runs in $O(\log(R/\delta))$ query complexity.

Lemma 21: Convert an approximate subgradient of h to a hyperplane that approximately separates x from K, where s is the unit normal vector of the hyperplane.

Theorem 22: Construct a separation oracle using $\tilde{O}(n)$ *classical* queries to a membership oracle.
Proof idea: To separate $x \notin B(K, -\varepsilon)$ from K, the idea is to compute an approximate subgradient of h near the origin and feed it into Lemma 21.

(1) By Lemma 19, h is Lipschitz.

(2) By Lemma 20, we can evaluate h to within error δ using $O(\log(R/\delta))$ queries to membership oracle.

(3) Use Lemma 12 to get an approximate subgradient $\tilde{g}$ with success probability $\geq 1 - \rho$, using $O(n \log(R/\delta))$ queries.

(4) Use Lemma 21 to convert approximate subgradient into unit normal vector s for separating hyperplane.

Theorem 23: Construct a separation oracle using $\tilde{O}(1)$ *quantum* queries to a membership oracle.
Proof idea: Similar to previous theorem, but use Lemma 18 instead of Lemma 12.

8.4 Lower bounds

Results of this section:

- $\Omega(n)$ classical queries to a membership oracle are needed to implement a weak separation oracle.
- $\Omega(n)$ classical (resp. $\Omega(\sqrt{n})$ quantum) queries to a separation oracle are needed to implement a weak optimization oracle; even when we know an interior point in the set.
- $\Omega(n)$ lower bound on the number of classical and/or quantum queries to a separation oracle needed to optimize over the set when we do not know an interior point.

Classical lower bound on the number of MEM queries needed for SEP

Theorem 24: A separation query can provide $\Omega(n)$ bits of information about the convex set K. Since a classical membership query returns a 0 or a 1 and hence can give at most 1 bit of information, this means we need at least $\Omega(n)$ classical membership queries to implement a separation query.
Specifically, it constructs a family of $m = 2^{\Omega(n)}$ convex sets $K_1, \ldots, K_m$ such that querying a separation oracle at any exterior point y uniquely identifies which specific set K_i was queried. Note that i is expressed in $\Omega(n)$ bits.

Lower bound on number of SEP queries for OPT (given an interior point)

We reduce validity to optimization as follows: Suppose we have an algorithm that solves optimization. Then we can solve validity:

- Run optimization algorithm on direction c to find $x^* \in K$.
- Compute the value $\langle c, x^* \rangle$.
- Check if $\langle c, x^* \rangle \geq \gamma$.

Search problem: Given $z \in \{0,1\}^n$ such that either $|z| = 0$ or $|z| = 1$, decide which of the two holds.
$\Omega(n)$ classical queries are needed, $\Omega(\sqrt{n})$ quantum queries are needed (i.e., Grover's quantum search algorithm is optimal; see [Zal99]).

Theorem 25: Any algorithm implementing a weak validity oracle using a strong separation oracle requires $\Omega(n)$ queries if the separation queries are classical, $\Omega(\sqrt{n})$ queries if the separation queries are quantum.

Proof idea: Reduce the search problem to the weak validity problem, then apply the known classical and quantum lower bounds on the search problem.

Lower bound on number of SEP queries for OPT (without interior point)

Finding a point close to K (using separation queries) reduces to OPT, because OPT returns a point close to K.

Theorem 26: $\Omega(n)$ quantum query lower bound for the problem of learning z with first-difference queries: One needs to find an initially unknown n-bit binary string z via a guessing game. For a given guess $g \in \{0,1\}^n$, a query returns the first index in $[n]$ for which the binary strings z and g differ (or it returns $n+1$ if $z = g$). The goal is to recover z with as few guesses as possible.

Theorem 27: Finding a point in an unknown convex set $K \subseteq \mathbb{R}^n$ using strong separation oracles requires $\Omega(n)$ quantum queries, even if we are promised that K contains a ball of constant radius.

Proof idea: Learning z with first-difference queries reduces to finding a point close to K.

Since learning z require $\Omega(n)$ queries, the point-finding problem must also require $\Omega(n)$ SEP queries.

Putting the two reductions together, learning z reduces to OPT. Hence, a lower bound for learning z is a lower bound for OPT.

9 No quantum speedup over gradient descent for non-smooth convex optimization [Gar+20]

Summary

This paper studies whether quantum algorithms can speed up general convex optimization when given access to a first-order oracle, i.e., an oracle that returns the function value and some subgradient. The main result is that, for general non-smooth convex functions, there is no quantum speedup over classical gradient descent in the dimension-independent black-box setting.

Results

Problem setup: Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be a convex function. We want to find $x \in \mathbb{R}^n$ that is ε-close to minimizing the function f. More precisely, let $x^* = \arg\min_{x\in\mathbb{R}^n} f(x)$. Then our goal is to find any $x \in \mathbb{R}^n$ such that $f(x) - f(x^*) \leq \varepsilon$.

In the black-box setting, if we only have access to an oracle computing $f(x)$, this is called **zeroth-order optimization**. In **first-order optimization**, we additionally have access to a subgradient of f. We denote the corresponding first-order oracle by $\mathcal{FO}(f)$.

The starting point of the algorithm is x_0. The complexity depends on the distance from x_0 to a minimizer. Let this distance be R. By translating the problem, we may assume without loss of generality that $x_0 = 0$.

The complexity of our algorithm also depends on how quickly f can change, since the value of f at some point constrains its values at nearby points if the function does not change too rapidly. Let G be an upper bound on the Lipschitz constant of f.

- **Problem 1 (First-order convex minimization):** Let $f\colon \mathbb{R}^n \to \mathbb{R}$ have Lipschitz constant at most G on $B(0,R)$, and let
$$x^* := \arg\min_{x\in B(0,R)} f(x).$$
Then given n, G, R, and $\varepsilon > 0$, the goal is to output $x \in B(0,R)$ such that $f(x) - f(x^*) \leq \varepsilon$, while minimizing the number of queries to f and $\mathcal{FO}(f)$.

  (Since the main result of the paper is a lower bound, the lower bound is stronger if shown for a simple convex set K. So the paper works with the set $K = B(0,R)$, the ℓ_2-ball of radius R around the origin.)

- **Gradient descent** (or in this case **subgradient descent**): algorithm that starts from a point x_0 and takes a small step (governed by a step size η) in the opposite direction of the subgradient at x_0. Intuitively, since each step we move in the direction where f decreases the most, this brings us closer to the minimum.

  Projected subgradient descent: subgradient descent with the added step of projecting the current vector back onto $B(0,R)$ after every step. This ensures we stay inside $B(0,R)$ after every step.

  (1) Start with initial vector $x_0 = 0$.

  (2) Compute x_{t+1} from x_t using the formula
$$x_{t+1} = \mathcal{P}_K(x_t - \eta \cdot g_{x_t})$$
where $\eta > 0$ is the step size (a parameter of the algorithm that we choose), P_K is the projector onto $B(0,R)$.

- **Theorem 2 (Complexity of projected subgradient descent):** The projected subgradient descent algorithm solves Problem 1 using $(GR/\varepsilon)^2$ queries to f and $\mathcal{FO}(f)$.

The important point is that the number of oracle queries does not depend on n. Such algorithms are called **dimension-independent**.

Proof idea: For any $\varepsilon > 0$, set the step size as ε. Run projected gradient descent algorithm for $T \geq \frac{1}{\varepsilon^2}$ steps. Define $\hat{x}_T$ as the average of x_t's for $t = 0, 1, \ldots, T-1$. Then show that $\hat{x}_T$ is the desired minimizer of f.

- **Theorem 3:** Any classical (deterministic or bounded-error randomized) algorithm that solves Problem 1 must make $\Omega((GR/\varepsilon)^2)$ queries to f or $\mathcal{FO}(f)$.

  The proof in this paper uses the following family of functions: for each $z \in \{-1, 1\}^n$, define $f_z : \mathbb{R}^n \to \mathbb{R}$ by

  $$f_z(x_1, \ldots, x_n) = \max_{i \in [n]} z_i x_i,$$

  where $n = O(1/\varepsilon^2)$. Then each f_z is convex with Lipschitz constant 1.

  We show that finding an ε-approximate minimum within $B(0, 1)$ requires $\Omega(n)$ queries to the oracles. This is done by showing with high probability, every query of a randomized algorithm only reveals $O(1)$ bits of information about the string z, but an ε-approximate solution to this problem allows us to reconstruct the string z, which has n bits of information.[6]

  Proof idea:

- **Theorem 4:** There is a quantum algorithm that solves Problem 1 on the class of functions that appear in the classical lower bound of Theorem 3, using $O(GR/\varepsilon)$ queries to the oracle for f.

  This is a quadratic speedup over any classical algorithm (and in particular, over gradient descent).

  Proof idea: Use Belovs' quantum algorithm for Combinatorial Group Testing, which states that given access to an oracle making OR queries to an n-bit string, the n-bit string can be learned in $O(\sqrt{n})$ quantum queries.

- **Theorem 5 (main result):** Any quantum algorithm that solves Problem 1 with high probability must make $\Omega((GR/\varepsilon)^2)$ queries to f or $\mathcal{FO}(f)$ in the worst case.

  Hence, the quantum lower bound is the same lower bound as the classical lower bound.

[6] More formally, let the unknown string $z \in \{0, 1\}^n$, and Q be the number of oracle queries made.
Each query reveals $O(1)$ bits of information about z. After Q queries, we get $O(Q)$ bits about z.
On the other hand, producing an ε-approximate minimizer enables the algorithm to reconstruct z. Thus, the output must contain $\Omega(n)$ bits of information about z. Hence, $O(Q) \geq \Omega(n)$, so $Q = \Omega(n)$.

A Linear Algebra

The basic objects we shall work with are complex vectors $|\psi\rangle \in \mathbb{C}^n$, i.e.,

$$|\psi\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{n-1} \end{pmatrix}$$

for some $\alpha_i \in \mathbb{C}$. The notation $|\cdot\rangle$ is called Dirac notation, named after physicist Paul Dirac, and is simply a useful convention for referring to column vectors. The term $|\psi\rangle$ is read "ket ψ".
The **conjugate transpose** of $|\psi\rangle$ is given by

$$\langle\psi| = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$$

where $\langle\psi|$ is a row vector. The term $\langle\psi|$ is pronounced "bra ψ". This allows us to define how much two vectors "overlap" via their **inner product** defined as

$$\langle\psi|\phi\rangle = \begin{pmatrix} \alpha_0^* & \cdots & \alpha_{n-1}^* \end{pmatrix} \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_{n-1} \end{pmatrix} = \sum_{i=0}^{n-1} \alpha_i^* \beta_i.$$

The inner product satisfies the following properties:

(1) Linearity in second argument:

$$\left(|v\rangle , \sum_i \lambda_i |w_i\rangle \right) = \sum_i \lambda_i (|v\rangle , |w_i\rangle)$$

(2) Conjugate symmetry: $\langle v|w\rangle = \langle w|v\rangle^*$

(3) Positive definiteness: $\langle v|v\rangle \geq 0$, where equality holds if and only if $|v\rangle = 0$

The **norm** of a vector $|v\rangle$ is the non-negative real number

$$\big\| |\psi\rangle \big\| = \sqrt{\langle\psi|\psi\rangle}.$$

A **unit vector** is a vector with norm 1.

Theorem A.1 (Cauchy-Schwarz inequality)**.** *For any* $|\psi\rangle , |\phi\rangle \in \mathbb{C}^n$,

$$|\langle\psi|\phi\rangle| \leq \big\||\psi\rangle\big\| \big\||\phi\rangle\big\| .$$

In finite dimensional complex vector spaces, a **Hilbert space** is an inner product space.

A.1 Orthonormal bases

Two vectors $|\psi\rangle$ and $|\phi\rangle$ are **orthogonal** if $\langle\psi|\phi\rangle = 0$. A set of vectors $\{|\psi_i\rangle\} \subseteq \mathbb{C}^n$ is **orthogonal** if $\langle\psi_i|\psi_j\rangle$ for all $i \neq j$, and **orthonormal** if $\langle\psi_i|\psi_j\rangle = \delta_{ij}$; here, δ_{ij} is the Kroenecker delta, whose value is 1 if $i = j$ and 0 otherwise.
One of the most common bases we use is the **computational basis**, defined for $\mathbb{C}^2$ as

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

The computational basis is easily extended to n-dimensional vectors by defining $|i\rangle \in \mathbb{C}^n$ as having 1 in position i and 0 elsewhere (here, $0 \leq i \leq n-1$):

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix}, \quad \ldots, \quad |n-1\rangle = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}.$$

A.2 Linear maps

A map $\Phi\colon \mathbb{C}^n \to \mathbb{C}^n$ is **linear** if, for any $\sum_i \alpha_i |\psi_i\rangle \in \mathbb{C}^n$,

$$\Phi\left(\sum_i \alpha_i |\psi_i\rangle\right) = \sum_i \alpha_i \Phi(|\psi_i\rangle).$$

The set of linear maps from vector space X to Y is denoted $\mathcal{L}(X, Y)$. We use the shorthand $\mathcal{L}(X)$ to mean $\mathcal{L}(X, X)$.

The action of a linear map $\Phi \in \mathcal{L}(\mathbb{C}^n)$ is fully characterised by its action on a basis for $\mathbb{C}^n$. This leads to a natural representation for Φ in terms of an $n \times n$ matrix, whose i-th column is $\Phi(|i\rangle)$ for $\{|i\rangle\}$ the standard basis for $\mathbb{C}^n$:

$$\begin{pmatrix} \Phi(|0\rangle) & \Phi(|1\rangle) & \cdots & \Phi(|n-1\rangle) \end{pmatrix}.$$

We use both the matrix and linear map views interchangeably, with the application notion clear from context.

A.3 Matrix operations

For a matrix $A \in M_{n\times m}(\mathbb{C})$, we define the following matrices:

- **complex conjugate** A^*: $(A^*)_{ij} = (A_{ij})^*$
- **transpose** $A^\top$: $(A^\top)_{ij} = A_{ji}$
- **conjugate transpose** $A^\dagger = (A^*)^\top$

The trace is a linear map $\mathrm{tr}\colon \mathcal{L}(\mathbb{C}^n) \to \mathbb{C}$ that sums the entries on the diagonal of A, namely,

$$\mathrm{tr}(A) = \sum_{i=1}^{n} a_{ii}.$$

A wonderful property of the trace is that it is *cyclic*, i.e., $\mathrm{tr}(ABC) = \mathrm{tr}(CAB)$. This implies that even if A and B do not commute, i.e., $AB \neq BA$, it nevertheless holds that $\mathrm{tr}(AB) = \mathrm{tr}(BA)$.

A.4 Outer products

For vectors $|\psi\rangle, |\phi\rangle \in \mathbb{C}^n$, the **outer product** is $|\psi\rangle\langle\phi| \in \mathcal{L}(\mathbb{C}^n)$; unlike the inner product, the outer product yields an $n \times n$ matrix. It can be computed straightforwardly via the rules of matrix multiplication. For example,

$$|0\rangle\langle 0| = \begin{pmatrix} 1 \\ 0 \end{pmatrix}\begin{pmatrix} 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad \text{and} \quad |1\rangle|0\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}.$$

More generally, the matrix $|i\rangle\langle j| \in \mathcal{L}(\mathbb{C}^n)$ has a 1 at position (i, j) and zeroes elsewhere. This yields a simple yet neat trick: A matrix $A \in \mathcal{L}(\mathbb{C}^n)$ written in the computational basis can hence be expressed as

$\sum_{ij} a_{ij} |i\rangle |j\rangle$. It is thus easy to evaluate expressions of the form

$$\langle i|A|j\rangle = \langle i| \left(\sum_{i'j'} a_{i'j'} |i'\rangle \langle j'| \right) |j\rangle = \sum_{i'j'} a_{i'j'} \langle i|i'\rangle \langle j|j'\rangle = \sum_{i'j'} a_{i'j'} \delta_{ii'} \delta_{jj'} = a_{ij},$$

where the third equality follows since $\{|i\rangle\}$ forms an orthonormal basis for $\mathbb{C}^n$. In other words, $\langle i|A|j\rangle$ simply rips out entry a_{ij}.

Let $|i\rangle$ be an orthonormal basis for V. Then it satisfies the completeness relation:

$$\sum_i |i\rangle \langle i| = I.$$

To see this, let $|v\rangle \in V$ be arbitrary. Write $|v\rangle = \sum_i v_i |i\rangle$. Note that $\langle i|v\rangle = v_i$, so

$$\left(\sum_i |i\rangle \langle i| \right) |v\rangle = \sum_i |i\rangle \langle i|v\rangle = \sum_i v_i |i\rangle = |v\rangle .$$

A.5 Eigenvalues and eigenvectors

An **eigenvector** of $A \in M_{n\times n}(\mathbb{C})$ is a non-zero vector $|\psi\rangle \in \mathbb{C}^n$ such that

$$A |\psi\rangle = \lambda |\psi\rangle$$

for some $\lambda \in \mathbb{C}$ called an **eigenvalue** corresponding to $|\psi\rangle$. Note that the eigenvalues of A are the roots of the characteristic polynomial $\det(A - \lambda I) = 0$.

Definition A.2. A matrix $A \in M_{n\times n}(\mathbb{C})$ is diagonalisable if

$$A = \sum_{i=1}^{n} \lambda_i |v_i\rangle \langle v_i|$$

where $|v_1\rangle, \ldots, |v_n\rangle$ is an orthonormal set of eigenvectors of A with corresponding eigenvalues $\lambda_1, \ldots, \lambda_n$.

This is called **spectral decomposition** of A.

Equivalently, A can be written as a diagonal matrix

$$\begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}$$

in the basis $|v_1\rangle, \ldots, |v_n\rangle$ of its eigenvectors.

A.6 Adjoints and Hermitian operators

Definition A.3. Let V be a Hilbert space, A be a linear operator on V. The **adjoint** of A is the linear operator $A^\dagger$ such that

$$(|v\rangle, A|w\rangle) = (A^\dagger |v\rangle, |w\rangle)$$

for all $|v\rangle, |w\rangle \in V$.

In a matrix representation of an operator A, the action of the Hermitian conjugation operation is to take the conjugate transpose of A, namely $A^\dagger = (A^*)^\top$.

Definition A.4. An operator A is **Hermitian** (or **self adjoint**) if $A = A^\dagger$. Similarly, a matrix $A \in M_{n\times n}(\mathbb{C})$ is **Hermitian** if $A = A^\dagger$.

An important class of Hermitian operators is the projectors.

Definition A.5. Let V be a vector space, $\dim V = d$. Let W be a subspace of V, with $\dim W = k$. Using the Gram-Schmidt procedure, it is possible to construct an orthonormal basis $|1\rangle, \ldots, |d\rangle$ for V, such that $|1\rangle, \ldots, |k\rangle$ is an orthonormal basis for W. Define the **projector** onto the subspace W by

$$P = \sum_{i=1}^{k} |i\rangle \langle i| .$$

An operator A is **normal** if $AA^\dagger = A^\dagger A$. Similarly, a matrix $A \in M_{n\times n}(\mathbb{C})$ is **normal** if $AA^\dagger = A^\dagger A$.

Theorem A.6 (Spectral decomposition)**.** *Let A be a normal $n \times n$ complex matrix. There exists an orthonormal basis of n-dimensional complex vectors $\{|\psi_1\rangle, \ldots, |\psi_n\rangle\}$ along with complex numbers $\lambda_1, \ldots, \lambda_n$, such that*

$$A = \sum_{i=1}^{n} \lambda_i |\psi_i\rangle \langle\psi_i| .$$

Note that for every $j = 1, \ldots, n$, we have

$$A|\psi_j\rangle = \lambda_j |\psi_j\rangle .$$

That is, each λ_j is an eigenvalue of A, and $|\psi_j\rangle$ is an eigenvector corresponding to that eigenvalue (since $|\psi_j\rangle \neq 0$). This is a consequence of the fact that $\{|\psi_1\rangle, \ldots, |\psi_n\rangle\}$ is orthonormal:

$$A|\psi_j\rangle = \left(\sum_{i=1}^{n} \lambda_i |\psi_i\rangle \langle\psi_i|\right) |\psi_j\rangle = \sum_{i=1}^{n} \lambda_i |\psi_i\rangle \langle\psi_i|\psi_j\rangle = \lambda_j |\psi_j\rangle .$$

Definition A.7. A matrix A is **unitary** if $AA^\dagger = A^\dagger A = I$. Similarly, an operator A is **unitary** if $AA^\dagger = A^\dagger A = I$.

Unitary operators are normal and therefore diagonalisable.

Lemma A.8. *The following are equivalent:*

(1) A is unitary.

(2) A preserves inner product: $\langle Av|Aw\rangle = \langle v|w\rangle$ for all v, w.

(3) A preserves norm: $\|Av\| = \|v\|$ for all v.

(4) $\|Av\| = 1$ if $\|v\| = 1$.

(5) The rows of A form an orthonormal set.

(6) The columns of A form an orthonormal set.

Note that by (4), all eigenvalues of a unitary matrix have absolute value 1, since

$$\||\psi\rangle\| = \|U|\psi\rangle\| = \|\lambda|\psi\rangle\| = |\lambda|\||\psi\rangle\| .$$

Since $|\psi\rangle \neq 0$, we cancel it from both sides to obtain $|\lambda| = 1$.

A.7 Tensor products

Definition A.9. If $A = (A_{ij})$ is an $m \times n$ matrix, and B an $m' \times n'$ matrix, then their **tensor product** is the $mm' \times nn'$ matrix

$$A \otimes B = \begin{pmatrix} A_{11}B & A_{12}B & \cdots & A_{1n}B \\ A_{21}B & A_{22}B & \cdots & A_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1}B & A_{m2}B & \cdots & A_{mn}B \end{pmatrix}$$

where the block at coordinates (i, j) is equal to $A_{ij}B$.

For example,

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} = \left(\begin{array}{cc|cc} 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \\ \hline 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \end{array}\right).$$

Tensor product applies to vectors too, e.g., $|\psi\rangle \otimes |\phi\rangle$ and $\langle\psi| \otimes \langle\phi|$. Note that the tensor product of two numbers (i.e., 1×1 matrices) is itself just a number, and the tensor product of two column vectors is itself a column vector. As a shorthand for $|\psi\rangle \otimes |\phi\rangle$, we write $|\psi\rangle |\phi\rangle$, $|\psi, \phi\rangle$, or $|\psi\phi\rangle$.

Properties of tensor product:

- $c(A \otimes B) = (cA) \otimes B = A \otimes (cB)$ for all scalars c
- $(A \otimes B)^\dagger = A^\dagger \otimes B^\dagger$, and similarly for inverse and transpose
- $A \otimes (B + C) = (A \otimes B) + (A \otimes C)$
- $A \otimes (B \otimes C) = (A \otimes B) \otimes C$
- $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$

A.8 Positive operators

Definition A.10. A **positive operator** A is one for which $\langle\psi|A|\psi\rangle \geq 0$ for all $|\psi\rangle$. A **positive definite operator** A is one for which $\langle\psi|A|\psi\rangle > 0$ for all $|\psi\rangle \neq 0$.

The following theorem shows that any positive operator is automatically Hermitian, and therefore by the spectral decomposition has diagonal representation $\sum_i \lambda_i |i\rangle \langle i|$, with non-negative eigenvalues λ_i.

Theorem A.11. *A positive operator is Hermitian.*

B Computational Complexity

- **P** (for polynomial time) is the class of problems that can be solved in polynomial time by a deterministic computer.
- **BPP** (for bounded-error probabilistic polynomial time) is the class of problems that can be solved in polynomial time by a probabilistic computer.

  It is clear that **BPP** contains **P**, since deterministic computation is a special case of probabilistic computation in which we never consult the source of randomness.
- **BQP** (for bounded-error[7] quantum polynomial) is the class of problems that can be solved in polynomial time by a quantum computer.

  Since a quantum computer can easily generate random bits and simulate a probabilistic classical computer, **BQP** certainly contains **BPP**.

Here we are interested in problems that are in **BQP** but not known to be in **BPP**. The most popular example of such a problem is factoring; see Shor's factoring algorithm.

[7] The phrase "bounded-error" has a precise meaning: the probability of error is at most $\frac{1}{3}$.

References

[BV09] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, 2009.

[Gar+20] Ankit Garg et al. *No quantum speedup over gradient descent for non-smooth convex optimization*. 2020. arXiv: 2010.01801 [cs.DS]. URL: https://arxiv.org/abs/2010.01801.

[JW18] S. Gribling J. van Apeldoorn A. Gilyén and R. de Wolf. "Convex optimization using quantum oracles". In: *Quantum* (2018).

[Jor05] Stephen P. Jordan. "Fast Quantum Algorithm for Numerical Gradient Estimation". In: *Physical Review Letters* 95.5 (2005). ISSN: 1079-7114. DOI: 10.1103/physrevlett.95.050501. URL: http://dx.doi.org/10.1103/PhysRevLett.95.050501.

[Zal99] Christof Zalka. "Grover's quantum searching algorithm is optimal". In: *Physical Review A* 60.4 (Oct. 1999), pp. 2746–2751. ISSN: 1094-1622. DOI: 10.1103/physreva.60.2746. URL: http://dx.doi.org/10.1103/PhysRevA.60.2746.