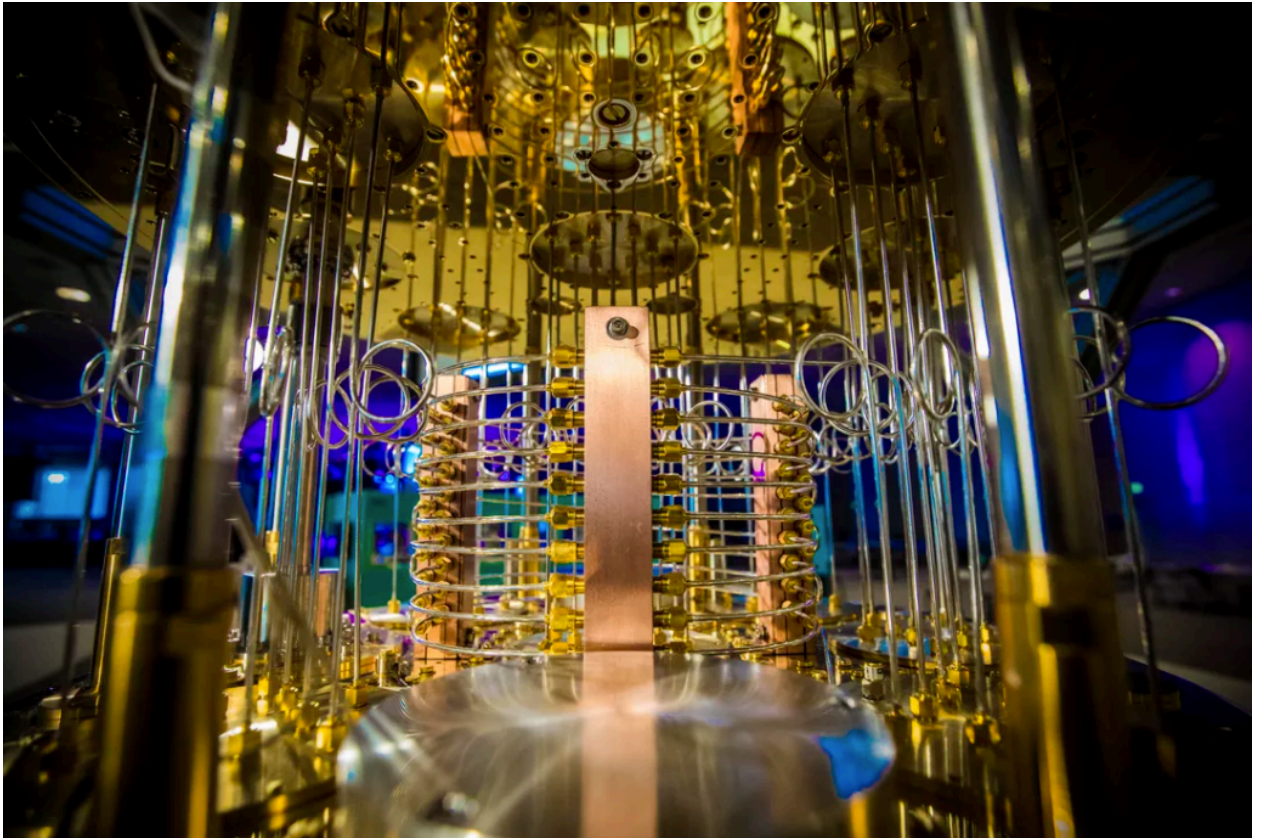


Quantum Information and Quantum Computation

Summer Of Science

Full Term Report



Vibhav Aggarwal

(Dept. of Computer Science and Engineering)

Mentor : Neeraj Sohani

Contents

0	Introduction	3
1	Quantum bits (qubits)	4
2	Introduction to Quantum Mechanics	5
2.1	Linear Algebra	5
2.1.1	Inner Products	5
2.1.2	Eigenvectors and eigenvalues	6
2.1.3	Adjoint and Hermitian operators	6
2.1.4	Tensor products	7
2.1.5	Operator functions	8
2.1.6	The commutator and anti-commutator	9
2.1.7	The polar and singular value decompositions	10
2.2	The postulates of quantum mechanics	10
2.2.1	State space	10
2.2.2	Evolution	10
2.2.3	Quantum measurement	11
2.2.4	Composite systems	12
2.3	The density operator	13
2.4	EPR paradox and the Bell inequality	14
3	Models for Computation	16
3.1	Turing machines	16
3.2	Circuits	18
4	Quantum circuits	21
4.1	Single qubit operations	21
4.2	Controlled operations	21
4.3	Measurement	22
4.4	Universal quantum gates	23
4.4.1	Two-level unitary gates are universal	23
4.4.2	Single qubit and CNOT gates are universal	24
5	The quantum Fourier transform and its applications	26
5.1	The quantum Fourier transform	26
5.2	Phase estimation	27
5.3	Applications	29
5.3.1	Application: order-finding	29
5.3.2	Application: factoring	29
5.3.3	Application : Period-finding	31
6	Distance measures for quantum information	32
6.1	Distance measures for classical information	32
6.2	How close are two quantum states?	33
6.2.1	Trace	33
6.2.2	Fidelity	34

0 Introduction

Quantum computation and quantum information is the study of the information processing tasks that can be accomplished using quantum mechanical systems. It is relatively a newer field of research and much work is going on to actually make a functional quantum computer. Big tech companies like Google, Microsoft, Hitachi, Mitsubishi, Nokia, etc. are investing huge amounts of money in research and development of quantum computers.

But why? What is so special about this field that so many people are interested in it? Turns out, quantum computers offer an essential speed advantage over classical computers. This speed advantage is so significant that many researchers believe that no conceivable amount of progress in classical computation would be able to overcome the gap between the power of a classical computer and the power of a quantum computer!

For example, the RSA cryptosystem used for transferring data in secure manner is just based on the fact that it is very difficult for a classical computer to factorize large numbers (order of 2048 bits) quickly. Till date there is simply no known algorithm to that efficiently on a classical computer. However, there is an algorithm known as Shor's algorithm which can just do that really quickly on a quantum computer.

There are many other algorithms based upon quantum computers and one class of such algorithms is the *quantum search algorithms*. The quantum search algorithm solves the following problem: Given a search space of size N , and no prior knowledge about the structure of the information in it, we want to find an element of that search space satisfying a known property. How long does it take to find an element satisfying that property? Classically, this problem requires approximately N operations, but the quantum search algorithm allows it to be solved using approximately \sqrt{N} operations.

Another important use is *quantum simulation*. Simulation of quantum systems on classical machines is difficult and the space requirement grows exponentially with increasing number of components in the system. But in case of quantum computers, this growth is linear and therefore they can be used in quantum chemistry to simulate large molecules and study the inter-atomic interactions.

1 Quantum bits (qubits)

The *bit* is the fundamental concept of classical computation. It has only two possible states: 0 and 1. Any information can be represented by a combination of bits. Using n bits, a total of 2^n different messages can be represented/conveyed.

Quantum computation and quantum information are built upon an analogous concept, the *quantum bit*, or *qubit* for short. Qubits are mathematical objects with certain properties. While it is true that qubits, like bits, are realized as actual physical systems, we are going to treat them as abstract mathematical objects.

A qubit has a state, just like a bit, represent by $|\psi\rangle$. Two possible states are $|0\rangle$ and $|1\rangle$. However it can also have a state which is a linear combination of these two. Thus:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$$

The numbers α and β are complex numbers. Put another way, the state of a qubit is a vector in a two-dimensional complex vector space. The special states $|0\rangle$ and $|1\rangle$ are known as computational basis states, and form an *orthonormal* basis for this vector space.

Since α and β can take infinitely many different complex values, one might be tempted to think that infinite different messages can be conveyed using a single qubit! But there's a catch. When we make a measurement of a qubit, its state *collapses* into either $|0\rangle$ with probability $|\alpha|^2$ or $|1\rangle$ with probability $|\beta|^2$. So every time we observe only one of the two possible states. Naturally, $|\alpha|^2 + |\beta|^2 = 1$.

Geometrically, we can interpret this as the condition that the qubit's state be normalized to length 1. Thus, in general a qubit's state is a unit vector in a two-dimensional complex vector space.

Since α and β are complex numbers with the only constraint that $|\alpha|^2 + |\beta|^2 = 1$, we may represent the state of qubit as:

$$|\psi\rangle = e^{i\gamma} \left(\cos \frac{\theta}{2} |0\rangle + e^{i\phi} \sin \frac{\theta}{2} |1\rangle \right)$$

In fact, we can ignore the factor $e^{i\gamma}$ because it has no observable effects. Thus we can effectively write:

$$|\psi\rangle = \cos \frac{\theta}{2} |0\rangle + e^{i\phi} \sin \frac{\theta}{2} |1\rangle$$

The numbers θ and ϕ define a point on the unit three-dimensional sphere, as shown in the figure below. This sphere is often called the *Bloch sphere*.

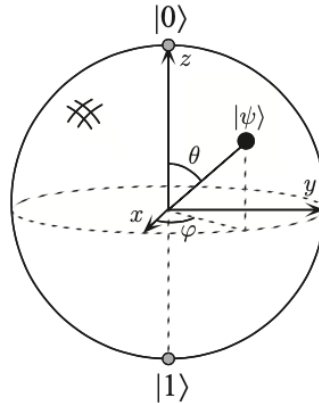


Figure 1: Bloch sphere

2 Introduction to Quantum Mechanics

To understand the concepts of quantum information and computing, a thorough understanding of quantum mechanics is required. Thus we must first get familiar with the mathematics involved and later we will see the postulates of quantum mechanics.

2.1 Linear Algebra

Linear algebra is the study of vector spaces and of linear operations on those vector spaces. A good understanding of quantum mechanics is based upon a solid grasp of elementary linear algebra.

Definition 2.1. A *spanning set* for a vector space is a set of vectors $|v_1\rangle, \dots, |v_n\rangle$ such that any vector $|v\rangle$ in the vector space can be written as a linear combination $|v\rangle = \sum_i a_i |v_i\rangle$ of vectors in that set.

Definition 2.2. A *linear operator* between vector spaces V and W is defined to be any function $A: V \rightarrow W$ which is linear in its inputs,

$$A\left(\sum_i a_i |v_i\rangle\right) = \sum_i a_i A(|v_i\rangle)$$

The most convenient way to understand linear operators is in terms of their *matrix representation*. Suppose $A: V \rightarrow W$ is a linear operator between vector spaces V and W . Suppose $|v_1\rangle, \dots, |v_m\rangle$ is a basis for V and $|w_1\rangle, \dots, |w_n\rangle$ is a basis for W . Then for each j in the range $1, \dots, m$, there exist complex numbers A_{1j} through A_{nj} such that

$$A|v_j\rangle = \sum_{i=1}^n A_{ij} |w_i\rangle$$

The $n \times m$ matrix whose entries are the values A_{ij} is said to form a matrix representation of the operator A . Matrix representation is very helpful in the way that applying operator A to a vector $|v\rangle$ is equivalent to multiplying the matrix of A by the column vector representation of $|v\rangle$.

The Pauli matrices

Four extremely useful matrices are the Pauli matrices. They are defined as:

$$\begin{aligned} \sigma_0 \equiv I &\equiv \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \sigma_1 \equiv \sigma_x \equiv X &\equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \\ \sigma_2 \equiv \sigma_y \equiv Y &\equiv \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} & \sigma_3 \equiv \sigma_z \equiv Z &\equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \end{aligned}$$

2.1.1 Inner Products

An *inner product* is a function which takes as input two vectors $|v\rangle$ and $|w\rangle$ from a vector space and produces a complex number as output. For the time being, it will be convenient to write the inner product of $|v\rangle$ and $|w\rangle$ as $(|v\rangle, |w\rangle)$ although the standard quantum mechanical notation is $\langle v|w\rangle$.

A function (\cdot, \cdot) from $V \times V$ to C is an inner product if it satisfies the requirements that:

1. (\cdot, \cdot) is linear in the second argument,

$$\left(|v\rangle, \sum_i \lambda_i |w_i\rangle\right) = \sum_i \lambda_i (|v\rangle, |w_i\rangle)$$

2. $(|v\rangle, |w\rangle) = (|w\rangle, |v\rangle)^*$
3. $(|v\rangle, |v\rangle) \geq 0$ with equality if and only if $|v\rangle = 0$

A vector space having an inner product is called an *inner product space*. For example, \mathbb{C}^n has an inner product defined by

$$((y_1, \dots, y_n), (z_1, \dots, z_n)) = \sum_i y_i^* z_i = \begin{bmatrix} y_1^* & \dots & y_n^* \end{bmatrix} \begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix}$$

In the finite dimensional complex vector spaces that come up in quantum computation and quantum information, a *Hilbert space* is *exactly the same thing* as an inner product space. From now on we may use the two terms interchangeably.

Definition 2.3. Vectors $|v\rangle$ and $|w\rangle$ are *orthogonal* if their inner product is zero.

Definition 2.4. The *norm* of a vector is defined as,

$$\| |v\rangle \| = \sqrt{\langle v | v \rangle}$$

Definition 2.5. A *unit vector* is a vector $|v\rangle$ such that $\| |v\rangle \| = 1$.

2.1.2 Eigenvectors and eigenvalues

Definition 2.6. An *eigenvector* of a linear operator A on a vector space is a non-zero vector $|v\rangle$ such that $A|v\rangle = v|v\rangle$, where v is a complex number known as the *eigenvalue* of A corresponding to $|v\rangle$.

Definition 2.7. A *diagonal representation* for an operator A on a vector space V is a representation $A = \sum_i \lambda_i |i\rangle \langle i|$, where the vectors $|i\rangle$ form an orthonormal set of eigenvectors for A , with corresponding eigenvalues λ_i . Diagonal representations are sometimes also known as *orthonormal decompositions*.

Definition 2.8. An operator is said to be *diagonalizable* if it has a diagonal representation.

2.1.3 Adjoint and Hermitian operators

Definition 2.9. Suppose A is any linear operator on a Hilbert space, V . There exists a unique operator A^\dagger on V such that for all vectors $|v\rangle, |w\rangle \in V$, $(|v\rangle, A|w\rangle) = (A^\dagger |v\rangle, |w\rangle)$. The linear operator A^\dagger is known as the *adjoint* or *Hermitian conjugate* of the operator A .

It can be proved that $A^\dagger = (A^*)^T$

Definition 2.10. An operator A is said to be *normal* if $AA^\dagger = A^\dagger A$.

Definition 2.11. An operator A is said to be *unitary* if $AA^\dagger = I$.

Theorem 2.1. (Spectral decomposition) Any normal operator M on a vector space V is diagonal with respect to some orthonormal basis for V . Conversely, any diagonalizable operator is normal.

Exercise 2.1. Show that a normal matrix is Hermitian if and only if it has real eigenvalues.

Solution. (\implies) Let $A|v\rangle = \lambda|v\rangle$. We have,

$$\begin{aligned} (|v\rangle, A|v\rangle) &= (A^\dagger |v\rangle, |v\rangle) \\ (|v\rangle, \lambda|v\rangle) &= (\lambda|v\rangle, |v\rangle) \\ \lambda(|v\rangle, |v\rangle) &= \lambda^*(|v\rangle, |v\rangle) \\ \lambda &= \lambda^* \end{aligned}$$

(\Leftarrow) By the spectral theorem, if A is normal, then it is diagonalizable. Hence, $A = UDU^\dagger$ for some unitary operator matrix U and a diagonal matrix D containing eigenvalues of A . Taking adjoint on both sides,

$$\begin{aligned} A^\dagger &= (UDU^\dagger)^\dagger \\ &= UD^*U^\dagger \\ &= UDU^\dagger \\ &= A \end{aligned}$$

Exercise 2.2. Show that all eigenvalues of a unitary matrix have modulus 1, that is, can be written in the form $e^{i\theta}$ for some real θ .

Solution.

$$\begin{aligned} U|v\rangle &= \lambda|v\rangle \\ \Rightarrow \langle v|U^\dagger &= \langle v|\lambda^* \end{aligned}$$

By multiplying the above equations, we get

$$\begin{aligned} \langle v|U^\dagger U|v\rangle &= \langle v|\lambda^* \lambda|v\rangle \\ \langle v|v\rangle &= \|\lambda\|^2 \langle v|v\rangle \\ \|\lambda\|^2 &= 1 \\ \|\lambda\| &= 1 \end{aligned}$$

Definition 2.12. A positive operator A is defined to be an operator such that for any vector $|v\rangle$, $(|v\rangle, A|v\rangle)$ is a real, non-negative number.

Exercise 2.3. Show that a positive operator is necessarily Hermitian.

Solution. Let

$$B = \frac{A + A^\dagger}{2} \qquad C = \frac{-iA + iA^\dagger}{2}$$

Then B and C are Hermitian and $A = B + iC$ Hence

$$\langle v|A|v\rangle = \langle v|B|v\rangle + i\langle v|C|v\rangle$$

Any Hermitian X can be represent as $X = \sum_i \lambda_i |i\rangle \langle i|$ where $|i\rangle$ are its eigenvectors and λ_i are corresponding eigenvalues. For any vector $|v\rangle$, $\langle v|X|v\rangle = \sum_i \lambda_i \langle v|i\rangle \langle i|v\rangle$ which is always real. Hence $\langle v|C|v\rangle = 0$ for all vectors $|v\rangle$ and this combined with the fact that C is Hermitian yields that C is identically 0. Therefore, A must be Hermitian.

2.1.4 Tensor products

The *tensor product* is a way of putting vector spaces together to form larger vector spaces. This construction is crucial to understanding the quantum mechanics of multiparticle systems.

In layman terms, tensor product of two vectors *from different vector spaces* is simply the "concatenation" of these vectors (i.e. placing next to each other).

Suppose V and W are vector spaces of dimension m and n respectively. For convenience we also suppose that V and W are Hilbert spaces. Then $V \otimes W$ (read 'V tensor W') is an mn dimensional vector space. The elements of $V \otimes W$ are linear combinations of 'tensor products' $|v\rangle \otimes |w\rangle$ of elements $|v\rangle$ of V and $|w\rangle$ of W . In particular, if $|i\rangle$ and $|j\rangle$ are orthonormal bases for the spaces V and W then $|i\rangle \otimes |j\rangle$ is a basis for $V \otimes W$.

Tensor product can be defined for operators too. Let A and B be linear operators acting on vector spaces V and W respectively. Then the operator $A \otimes B$ acting on $V \otimes W$ is defined as

$$(A \otimes B)(|v\rangle \otimes |w\rangle) \equiv (A|v\rangle) \otimes (B|w\rangle)$$

The inner products on the spaces V and W can be used to define a natural inner product on $V \otimes W$. Define

$$\left(\sum_i a_i |v_i\rangle \otimes |w_i\rangle, \sum_j b_j |v'_j\rangle \otimes |w'_j\rangle \right) \equiv \sum_{ij} a_i^* b_j \langle v_i | v'_j \rangle \langle w_i | w'_j \rangle$$

It can be shown that the function so defined is a well-defined inner product.

Now we discuss the *Kronecker product* which is a convenient matrix representation of $A \otimes B$. Suppose A is an $m \times n$ matrix, and B is a $p \times q$ matrix. Then we have the matrix representation:

$$A \otimes B = \overbrace{\begin{bmatrix} A_{11}B & A_{12}B & \dots & A_{1n}B \\ A_{21}B & A_{22}B & \dots & A_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1}B & A_{m2}B & \dots & A_{mn}B \end{bmatrix}}^{nq} \Bigg\}^{mp}$$

We use $|\psi\rangle^{\otimes k}$ to denote $|\psi\rangle$ tensored with itself k times.

2.1.5 Operator functions

There are many important functions which can be defined for operators and matrices. Generally speaking, given a function f from the complex numbers to the complex numbers, it is possible to define a corresponding matrix function on normal matrices by the following construction. Let $A = \sum_a a |a\rangle \langle a|$ be a spectral decomposition for a normal operator A . Define $f(A) \equiv \sum_a f(a) |a\rangle \langle a|$. This procedure can be used, for example, to define the square root of a positive operator, the logarithm of a positive-definite operator, or the exponential of a normal operator.

Exercise 2.4. (Exponential of the Pauli matrices) Let \vec{v} be any real, three-dimensional unit vector and θ a real number. Prove that

$$\exp(i\theta \vec{v} \cdot \vec{\sigma}) = \cos(\theta)I + i \sin(\theta) \vec{v} \cdot \vec{\sigma},$$

where $\vec{v} \cdot \vec{\sigma} = \sum_{i=1}^3 v_i \sigma_i$.

Solution. Note that $\sigma_i \sigma_j = \begin{cases} I, & \text{if } i = j \\ -\sigma_j \sigma_i, & \text{if } i \neq j \end{cases}$

Hence, $(\vec{v} \cdot \vec{\sigma})^2 = (v_1^2 + v_2^2 + v_3^2)I = I$

$$\begin{aligned} \exp(i\theta \vec{v} \cdot \vec{\sigma}) &= \sum_{k=0}^{\infty} \frac{(i\theta \vec{v} \cdot \vec{\sigma})^k}{k!} \\ &= \sum_{k=0}^{\infty} \frac{(i\theta \vec{v} \cdot \vec{\sigma})^{2k}}{(2k)!} + \sum_{k=0}^{\infty} \frac{(i\theta \vec{v} \cdot \vec{\sigma})^{2k+1}}{(2k+1)!} \\ &= \sum_{k=0}^{\infty} \frac{(-1)^k (\theta)^{2k} I}{(2k)!} + i \vec{v} \cdot \vec{\sigma} \sum_{k=0}^{\infty} \frac{(-1)^k (\theta)^{2k+1}}{(2k+1)!} \\ &= \cos(\theta)I + i \sin(\theta) \vec{v} \cdot \vec{\sigma} \end{aligned}$$

An important matrix function is the *trace* of a matrix.

Definition 2.13. The trace of A is defined to be the sum of its diagonal elements,

$$\text{tr}(A) \equiv \sum_i A_{ii}$$

The following properties can easily be proved for the trace of a matrix:

1. $\text{tr}(AB) = \text{tr}(BA)$
2. $\text{tr}(A + B) = \text{tr}(A) + \text{tr}(B)$
3. $\text{tr}(zA) = z \text{tr}(A)$

2.1.6 The commutator and anti-commutator

Definition 2.14. The *commutator* between two operators A and B is defined to be

$$[A, B] = AB - BA$$

Definition 2.15. The *anti-commutator* between two operators A and B is defined to be

$$\{A, B\} = AB + BA$$

If $[A, B] = 0$, we say that A *commutes* with B . Similarly, If $\{A, B\} = 0$, we say that A *ant-commutes* with B .

Theorem 2.2. (Simultaneous diagonalization theorem) Suppose A and B are Hermitian operators. Then $[A, B] = 0$ if and only if there exists an orthonormal basis such that both A and B are diagonal with respect to that basis. We say that A and B are *simultaneously diagonalizable* in this case.

Proof. It can be easily verified that if A and B are diagonal in the same orthonormal basis then $[A, B] = 0$. To show the converse, let $|a, j\rangle$ be an orthonormal basis for the eigenspace V_a of A with eigenvalue a ; the index j is used to label possible degeneracies. Note that

$$AB|a, j\rangle = BA|a, j\rangle = aB|a, j\rangle$$

and therefore $B|a, j\rangle$ is an element of the eigenspace V_a . Let P_a denote the projector onto the space V_a and define $B_a \equiv P_a B P_a$. It is easy to see that the restriction of B_a to the space V_a is Hermitian on V_a , and therefore has a spectral decomposition in terms of an orthonormal set of eigenvectors which span the space V_a . Let's call these eigenvectors $|a, b, k\rangle$, where the indices a and b label the eigenvalues of A and B_a , and k is an extra index to allow for the possibility of a degenerate B_a . Note that $B|a, b, k\rangle$ is an element of V_a , so $B|a, b, k\rangle = P_a B|a, b, k\rangle$. Moreover we have $P_a|a, b, k\rangle = |a, b, k\rangle$, so

$$B|a, b, k\rangle = P_a B P_a|a, b, k\rangle = B_a|a, b, k\rangle = b|a, b, k\rangle$$

It follows that $|a, b, k\rangle$ is an eigenvector of B with eigenvalue b , and therefore $|a, b, k\rangle$ is an orthonormal set of eigenvectors of both A and B , spanning the entire vector space on which A and B are defined. That is, A and B are simultaneously diagonalizable. \square

2.1.7 The polar and singular value decompositions

The *polar* and *singular value* decompositions are useful ways of breaking linear operators up into simpler parts. In particular, these decompositions allow us to break general linear operators up into products of unitary operators and positive operators. While we don't understand the structure of general linear operators terribly well, we do understand unitary operators and positive operators in quite some detail. The polar and singular value decompositions allow us to apply this understanding to better understand general linear operators.

Theorem 2.3. (Polar decomposition) Let A be a linear operator on a vector space V . Then there exists unitary U and positive operators J and K such that

$$A = UJ = KU,$$

where the unique positive operators J and K satisfying these equations are defined by $J \equiv \sqrt{A^\dagger A}$ and $K \equiv \sqrt{AA^\dagger}$. Moreover, if A is invertible then U is unique.

Corollary 2.4. (Singular value decomposition) Let A be a square matrix. Then there exist unitary matrices U and V , and a diagonal matrix D with non-negative entries such that

$$A = UDV$$

The diagonal elements of D are called the singular values of A .

2.2 The postulates of quantum mechanics

Quantum mechanics is a mathematical framework for the development of physical theories. On its own quantum mechanics doesn't tell you what laws a physical system must obey, but it does provide a mathematical and conceptual framework for the development of such laws. In the next few sections we give a complete description of the basic postulates of quantum mechanics. These postulates provide a connection between the physical world and the mathematical formalism of quantum mechanics.

2.2.1 State space

Postulate 1. Associated to any isolated physical system is a complex vector space with inner product (that is, 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.

Quantum mechanics does *not* tell us, for a given physical system, what the state space of that system is, nor does it tell us what the state vector of the system is. Figuring that out for a *specific* system is a difficult problem for which physicists have developed many intricate and beautiful rules.

The simplest quantum mechanical system, and the system which we will be most concerned with, is the *qubit*. A qubit has a two-dimensional state space. Suppose $|0\rangle$ and $|1\rangle$ form an orthonormal basis for that state space. Then an arbitrary state vector in the state space can be written

$$|\psi\rangle = a|0\rangle + b|1\rangle$$

where a and b are complex numbers. The condition that $|\psi\rangle$ be a unit vector, $\langle\psi|\psi\rangle = 1$, is therefore equivalent to $|a|^2 + |b|^2 = 1$. The condition $\langle\psi|\psi\rangle = 1$ is often known as the *normalization condition* for state vectors.

2.2.2 Evolution

How does the state, $|\psi\rangle$, of a quantum mechanical system change with time? The following postulate gives a prescription for the description of such state changes.

Postulate 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_1 is related to the state $|\psi'\rangle$ of the system at time t_2 by a unitary operator U which depends only on the times t_1 and t_2 ,

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

Just as quantum mechanics does not tell us the state space or quantum state of a *particular* quantum system, it does not tell us which unitary operators U describe real world quantum dynamics. Quantum mechanics merely assures us that the evolution of any closed quantum system may be described in such a way. An obvious question to ask is: what unitary operators are natural to consider? In the case of single qubits, it turns out that *any* unitary operator at all can be realized in realistic systems.

2.2.3 Quantum measurement

We postulated that closed quantum systems evolve according to unitary evolution. The evolution of systems which don't interact with the rest of the world is all very well, but there must also be times when the experimentalist and their experimental equipment – an external physical system in other words – observes the system to find out what is going on inside the system, an interaction which makes the system no longer closed, and thus not necessarily subject to unitary evolution. To explain what happens when this is done, we introduce Postulate 3, which provides a means for describing the effects of measurements on quantum systems.

Postulate 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 then the probability that result m occurs is given by

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

and the state of the system after the measurement is

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

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:

$$1 = \sum_m p(m) = \sum_m \langle \psi | M_m^\dagger M_m | \psi \rangle$$

A simple but important example of a measurement is the *measurement of a qubit in the computational basis*. This is a measurement on a single qubit with two outcomes defined by the two measurement operators $M_0 = |0\rangle\langle 0|$, $M_1 = |1\rangle\langle 1|$. Observe that each measurement operator is Hermitian, and that $M_0^2 = M_0$, $M_1^2 = M_1$. Thus the completeness relation is obeyed, $I = M_0^\dagger M_0 + M_1^\dagger M_1 = M_0 + M_1$. Suppose the state being measured is $|\psi\rangle = a|0\rangle + b|1\rangle$. Then the probability of obtaining measurement outcome 0 is

$$p(0) = \langle \psi | M_0^\dagger M_0 | \psi \rangle = \langle \psi | M_0 | \psi \rangle = |a|^2$$

Similarly, the probability of obtaining the measurement outcome 1 is $p(1) = |b|^2$. The state after measurement in the two cases is therefore

$$\begin{aligned} \frac{M_0 |\psi\rangle}{|a|} &= \frac{a}{|a|} |0\rangle \\ \frac{M_1 |\psi\rangle}{|b|} &= \frac{b}{|b|} |1\rangle \end{aligned}$$

The multipliers like $a/|a|$, which have modulus one, can effectively be ignored because they don't have any observable effect, so the two post-measurement states are effectively $|0\rangle$ and $|1\rangle$.

2.2.4 Composite systems

Postulate 4. The state space of a composite physical system is the tensor product of the state spaces of the component physical systems. Moreover, if we have systems numbered 1 through n , and system number i is prepared in the state $|\psi_i\rangle$, then the joint state of the total system is $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \dots \otimes |\psi_n\rangle$

Exercise 2.5. Show that the average value of the observable $X_1 Z_2$ for a two qubit system measured in the state $(|00\rangle + |11\rangle)/2$ is zero.

Solution. Let $|\psi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$

The average value of operator $X_1 Z_2$ is given by $\langle\psi| X_1 Z_2 |\psi\rangle$

$$\begin{aligned} \langle\psi| X_1 Z_2 |\psi\rangle &= \langle\psi| X_1 \left(\frac{|00\rangle - |11\rangle}{\sqrt{2}} \right) \\ &= \left(\frac{\langle 00| + \langle 11|}{\sqrt{2}} \right) \left(\frac{|10\rangle - |01\rangle}{\sqrt{2}} \right) \\ &= \frac{1}{2} (\langle 00|10\rangle - \langle 00|01\rangle + \langle 11|10\rangle - \langle 11|01\rangle) \\ &= 0 \end{aligned}$$

(since the four states $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ are orthogonal)

Postulate 4 also enables us to define one of the most interesting and puzzling ideas associated with composite quantum systems – *entanglement*.

Formally, any multiple qubit state which cannot be factorized into single qubit states is called an *entangled state*.

For example, consider the state $|\psi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$. We are now going to show that it is entangled by contradiction.

Let $|\psi\rangle = |a\rangle |b\rangle$, where $|a\rangle$ and $|b\rangle$ are single qubit states. Further let $|a\rangle = \alpha_1 |0\rangle + \beta_1 |1\rangle$ and $|b\rangle = \alpha_2 |0\rangle + \beta_2 |1\rangle$. Then,

$$\frac{|00\rangle + |11\rangle}{\sqrt{2}} = |\psi\rangle = |a\rangle |b\rangle = \alpha_1 \alpha_2 |00\rangle + \alpha_1 \beta_2 |01\rangle + \beta_1 \alpha_2 |10\rangle + \beta_1 \beta_2 |11\rangle$$

Therefore, we have,

$$\begin{aligned} \alpha_1 \alpha_2 &= \frac{1}{\sqrt{2}}, \\ \alpha_1 \beta_2 &= 0, \\ \beta_1 \alpha_2 &= 0, \\ \beta_1 \beta_2 &= \frac{1}{\sqrt{2}} \end{aligned}$$

By multiplying first and fourth equations and second and third equations, we get two different values for the product $\alpha_1 \alpha_2 \beta_1 \beta_2$ which is clearly a contradiction. Hence, the given state is entangled.

2.3 The density operator

The density operator is an alternate and a convenient way to represent the state of composite systems especially when the exact state of the system is not known. Suppose a quantum system is in one of a number of states $|\psi_i\rangle$, where i is an index, with respective probabilities p_i . We shall call $p_i, |\psi_i\rangle$ an ensemble of pure states. The density operator for the system is defined by the equation :

$$\rho \equiv \sum_i p_i |\psi_i\rangle \langle \psi_i|$$

The density operator is also known as *density matrix*.

It turns out that all the postulates of quantum mechanics can be reformulated in terms of the density operator language. Suppose, for example, that the evolution of a closed quantum system is described by the unitary operator U . If the system was initially in the state $|\psi_i\rangle$ with probability p_i then after the evolution has occurred the system will be in the state $U|\psi_i\rangle$ with probability p_i . Thus, the evolution of the density operator is described by the equation

$$\rho = \sum_i p_i |\psi_i\rangle \langle \psi_i| \rightarrow \sum_i p_i U |\psi_i\rangle \langle \psi_i| U^\dagger = U^\dagger \rho U$$

General properties of the density operator

Theorem 2.5. (Characterization of density operators) 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)** ρ has trace equal to one.
2. **(Positivity condition)** ρ is a positive operator.

Proof. Suppose $\rho = \sum_i p_i |\psi_i\rangle \langle \psi_i|$ is a density operator. Then

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

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

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

so the positivity condition is satisfied. Conversely, suppose ρ is any operator satisfying the trace and positivity conditions. Since ρ is positive, it must have a spectral decomposition

$$\rho = \sum_j \lambda_j |j\rangle \langle j|$$

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 . \square

This theorem provides a characterization of density operators that is intrinsic to the operator itself: we can *define* a density operator to be a positive operator ρ which has trace equal to one.

2.4 EPR paradox and the Bell inequality

The story behind EPR paradox is a fascinating one. EPR stands for Einstein–Podolsky–Rosen and it is a thought experiment which was proposed to show that quantum mechanics is an incomplete description of nature - a common belief of scientist those days. However nearly thirty years after the EPR paper was published, an experimental test was proposed that could be used to check whether or not the picture of the world which EPR were hoping to force a return to is valid or not. It turns out that Nature experimentally invalidates that point of view, while agreeing with quantum mechanics. The key to this experimental invalidation is a result known as *Bell's inequality*.

Imagine we perform the following experiment. Charlie prepares two particles. It doesn't matter how he prepares the particles, just that he is capable of repeating the experimental procedure which he uses. Once he has performed the preparation, he sends one particle to Alice, and the second particle to Bob.

Once Alice receives her particle, she performs a measurement on it. Imagine that she has available two different measurement apparatuses, so she could choose to do one of two different measurements. These measurements are of physical properties which we shall label P_Q and P_R , respectively. Alice doesn't know in advance which measurement she will choose to perform. Rather, when she receives the particle she flips a coin or uses some other random method to decide which measurement to perform. We suppose for simplicity that the measurements can each have one of two outcomes, $+1$ or -1 . Suppose Alice's particle has a value Q for the property P_Q . Q is assumed to be an objective property of Alice's particle, which is merely revealed by the measurement, much as we imagine the position of a tennis ball to be revealed by the particles of light being scattered off it. Similarly, let R denote the value revealed by a measurement of the property P_R .

Similarly, suppose that Bob is capable of measuring one of two properties, P_S or P_T , once again revealing an objectively existing value S or T for the property, each taking value $+1$ or -1 . Bob does not decide beforehand which property he will measure, but waits until he has received the particle and then chooses randomly. The timing of the experiment is arranged so that Alice and Bob do their measurements at the same time (or, to use the more precise language of relativity, in a causally disconnected manner). Therefore, the measurement which Alice performs cannot disturb the result of Bob's measurement (or vice versa), since physical influences cannot propagate faster than light.

We are going to do some simple algebra with the quantity $QS + RS + RT - QT$. Notice that

$$QS + RS + RT - QT = (Q + R)S + (R - Q)T$$

Because $R, Q = \pm 1$ it follows that either $(Q + R)S = 0$ or $(R - Q)T = 0$. In either case, it is easy to see that $QS + RS + RT - QT = \pm 2$. Suppose next that $p(q, r, s, t)$ is the probability that, before the measurements are performed, the system is in a state where $Q = q, R = r, S = s$, and $T = t$. These probabilities may depend on how Charlie performs his preparation, and on experimental noise. Letting $E(\cdot)$ denote the mean value of a quantity, we have

$$\begin{aligned} E(QS + RS + RT - QT) &= \sum_{qrst} p(q, r, s, t)(qs + rs + rt - qt) \\ &\leq \sum_{qrst} p(q, r, s, t) \times 2 \\ &= 2 \end{aligned}$$

Thus we obtain the *Bell inequality*

$$E(QS) + E(RS) + E(RT) - E(QT) \leq 2$$

This result is also often known as the CHSH inequality after the initials of its four discoverers. It is part of a larger set of inequalities known generically as Bell inequalities, since the first was found by John Bell.

What can we learn from Bell's inequality? For physicists, the most important lesson is that their deeply held commonsense intuitions about how the world works are wrong. The world is not locally realistic. Most physicists take the point of view that it is the assumption of realism which needs to be dropped from our worldview in quantum mechanics, although others have argued that the assumption of locality should be dropped instead. Regardless, Bell's inequality together with substantial experimental evidence now points to the conclusion that either or both of locality and realism must be dropped from our view of the world if we are to develop a good intuitive understanding of quantum mechanics.

3 Models for Computation

How do you define an *algorithm*? In its most basic terms, an algorithm is simply a sequence of tasks that one can perform to get a desired result. For example, in school we learn the long division method to find square root of integers. That's an algorithm.

In the theory of computation, a computer algorithm is also similar in meaning. In 1930s the fundamental notions of the modern theory of algorithms, and thus of computation, were introduced, by Alonzo Church, Alan Turing, and other pioneers of the computer era. This work arose in response to a profound challenge laid down by the great mathematician David Hilbert in the early part of the twentieth century. Hilbert asked whether or not there existed some algorithm which could be used, in principle, to solve all the problems of mathematics. Hilbert expected that the answer to this question, sometimes known as the *entscheidungsproblem*, would be yes.

Amazingly, the answer to Hilbert's challenge turned out to be no: there is no algorithm to solve all mathematical problems. To prove this, Church and Turing had to solve the deep problem of capturing in a mathematical definition what we mean when we use the intuitive concept of an algorithm. In so doing, they laid the foundations for the modern theory of algorithms, and consequently for the modern theory of computer science.

Here we will describe two models of computation - Turing machines and circuit model.

3.1 Turing machines

The basic elements of a Turing machine are illustrated in the following figure. A Turing machine contains four main elements: (a) a *program*, rather like an ordinary computer; (b) a *finite state control*, which acts like a stripped-down microprocessor, coordinating the other operations of the machine; (c) a *tape*, which acts like a computer memory; and (d) a *read- write tape-head*, which points to the position on the tape which is currently readable or writable. We now describe each of these four elements in more detail.

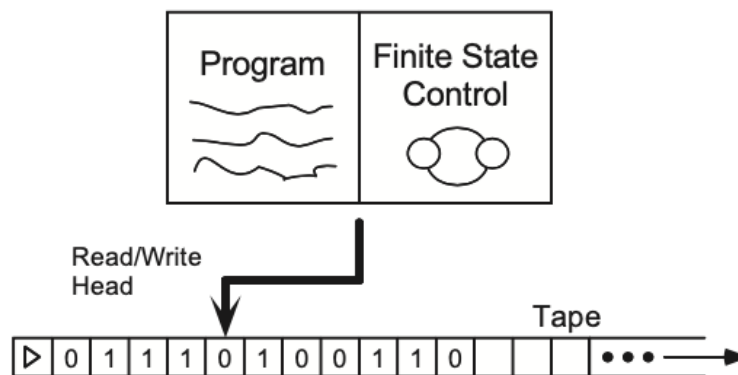


Figure 2: Main elements of a Turing machine

The finite state control for a Turing machine consists of a finite set of internal states, q_1, \dots, q_m . The number m is allowed to be varied; it turns out that for m sufficiently large this does not affect the power of the machine in any essential way, so without loss of generality we may suppose that m is some fixed constant. The best way to think of the finite state control is as a sort of microprocessor, coordinating the Turing machine's operation. It provides temporary storage off-tape, and a central place where all processing for the machine may be done. In addition to the states q_1, \dots, q_m , there are also two special internal states, labelled q_s and q_h . We call these the starting state and the halting state, respectively. The idea is that at the beginning of the computation, the Turing machine is in the starting state q_s . The execution of the

computation causes the Turing machine's internal state to change. If the computation ever finishes, the Turing machine ends up in the state q_h to indicate that the machine has completed its operation.

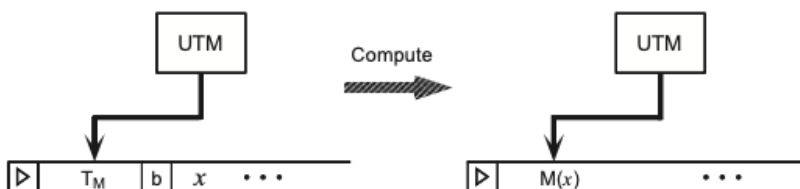
The Turing machine tape is a one-dimensional object, which stretches off to infinity in one direction. The tape consists of an infinite sequence of tape squares. We number the tape squares $0, 1, 2, 3, \dots$. The tape squares each contain one symbol drawn from some alphabet, Γ , which contains a finite number of distinct symbols. For now, it will be convenient to assume that the alphabet contains four symbols, which we denote by $0, 1, b$ (the 'blank' symbol), and an arrow, to mark the left hand edge of the tape. Initially, the tape contains an arrow at the left hand end, a finite number of 0s and 1s, and the rest of the tape contains blanks. The read-write tape-head identifies a single square on the Turing machine tape as the square that is currently being accessed by the machine.

A *program* for a Turing machine is a finite ordered list of program lines of the form $\langle q, x, q', x', s \rangle$. The first item in the program line, q , is a state from the set of internal states of the machine. The second item, x , is taken from the alphabet of symbols which may appear on the tape, Γ . The way the program works is that on each machine cycle, the Turing machine looks through the list of program lines in order, searching for a line $\langle q, x, \cdot, \cdot, \cdot \rangle$, such that the current internal state of the machine is q , and the symbol being read on the tape is x . If it doesn't find such a program line, the internal state of the machine is changed to q_h , and the machine halts operation. If such a line is found, then that program line is executed. Execution of a program line involves the following steps: the internal state of the machine is changed to q' ; the symbol x on the tape is overwritten by the symbol x' , and the tape-head moves left, right, or stands still, depending on whether s is -1 , $+1$, or 0 , respectively. The only exception to this rule is if the tape-head is at the leftmost tape square, and $s = -1$, in which case the tape-head stays put.

It turns out that the Turing machine model of computation can be used to compute an enormous variety of functions. For example, it can be used to do all the basic arithmetical operations, to search through text represented as strings of bits on the tape, and many other interesting operations. Surprisingly, it turns out that a Turing machine can be used to simulate all the operations performed on a modern computer! Indeed, according to a thesis put forward independently by Church and by Turing, the Turing machine model of computation completely captures the notion of computing a function using an algorithm. This is known as the Church–Turing thesis: *The class of functions computable by a Turing machine corresponds exactly to the class of functions which we would naturally regard as being computable by an algorithm.*

The Universal Turing Machine

We've described Turing machines as containing three elements which may vary from machine to machine - the initial configuration of the tape, the internal states of the finite state control, and the program for the machine. A clever idea known as the Universal Turing Machine (UTM) allows us to fix the program and finite state control once and for all, leaving the initial contents of the tape as the only part of the machine which needs to be varied. The Universal Turing Machine (see the figure below) has the following property. Let M be any Turing machine, and let T_M be the Turing number associated to machine M . Then on input of the binary representation for T_M followed by a blank, followed by any string of symbols x on the remainder of the tape, the Universal Turing Machine gives as output whatever machine M would have on input of x . Thus, the Universal Turing Machine is capable of simulating any other Turing machine!

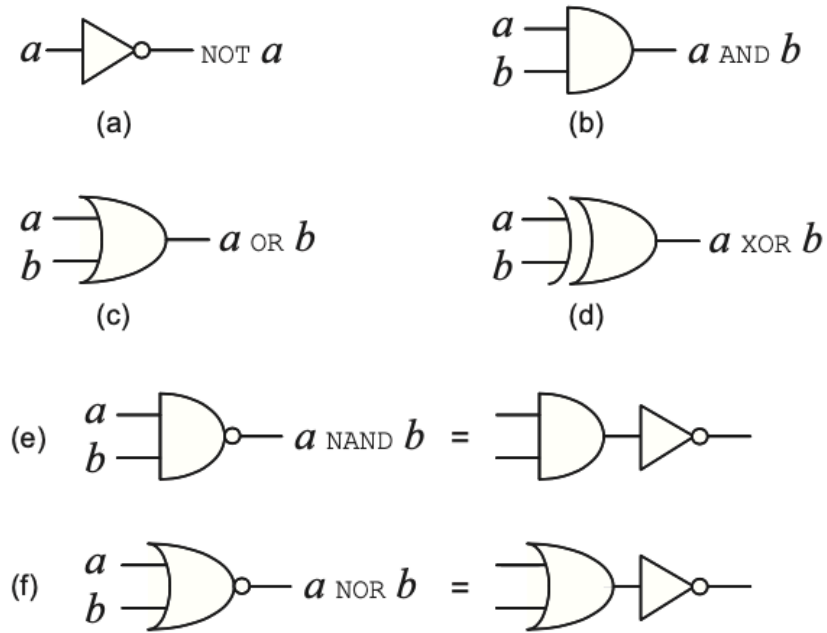


3.2 Circuits

Turing machines are rather idealized models of computing devices. Real computers are finite in size, whereas for Turing machines we assumed a computer of unbounded size. Here we investigate an alternative model of computation, the circuit model, that is equivalent to the Turing machine in terms of computational power, but is more convenient and realistic for many applications. In particular the circuit model of computation is especially important as preparation for our investigation of quantum computers.

A circuit is made up of *wires* and *gates*, which carry information around, and perform simple computational tasks, respectively. More generally, a circuit may involve many input and output bits, many wires, and many logical gates. A *logic gate* is a function $f : \{0, 1\}^k \rightarrow \{0, 1\}^l$ from some fixed number k of input bits to some fixed number l of output bits.

The following figure shows the circuit representation of the elementary logic gates.



Now, let's look at a simple example of a circuit which adds two n bit integers. The basic element in this circuit is a smaller circuit known as a half-adder. A half-adder takes two bits, x and y , as input, and outputs the sum of the bits $x \oplus y$ modulo 2, together with a carry bit set to 1 if x and y are both 1, or 0 otherwise.

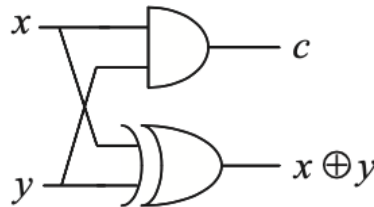


Figure 3: Half-adder circuit

Two cascaded half-adders may be used to build a full-adder, as shown in the following figure. A full-adder takes as input three bits, x , y , and c . The bits x and y should be thought of as data to be added, while c is

a carry bit from an earlier computation. The circuit outputs two bits. One output bit is the modulo 2 sum, $x \oplus y \oplus c$ of all three input bits. The second output bit, c' , is a carry bit, which is set to 1 if two or more of the inputs is 1, and is 0 otherwise.

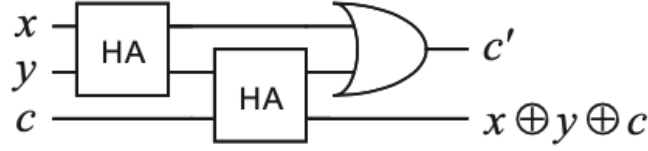


Figure 4: Full-adder circuit

By cascading many of these full-adders together we obtain a circuit to add two n -bit integers, as illustrated in the figure for the case $n = 3$.

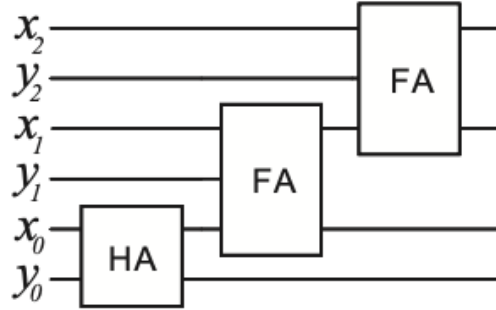


Figure 5: Addition circuit for two three-bit integers

We claimed earlier that just a few fixed gates can be used to compute any function $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ whatsoever. We will now prove this for the simplified case of a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ with n input bits and a single output bit. Such a function is known as a *Boolean function*, and the corresponding circuit is a *Boolean circuit*. The general universality proof follows immediately from the special case of Boolean functions. The proof is by induction on n . For $n = 1$ there are four possible functions: the identity, which has a circuit consisting of a single wire; the bit flip, which is implemented using a single *NOT* gate; the function which replaces the input bit with a 0, which can be obtained by *AND*ing the input with a work bit initially in the 0 state; and the function which replaces the input with a 1, which can be obtained by *OR*ing the input with a work bit initially in the 1 state.

To complete the induction, suppose that any function on n bits may be computed by a circuit, and let f be a function on $n + 1$ bits. Define n -bit functions f_0 and f_1 by $f_0(x_1, \dots, x_n) \equiv f(0, x_1, \dots, x_n)$ and $f_1(x_1, \dots, x_n) \equiv f(1, x_1, \dots, x_n)$. These are both n -bit functions, so by the inductive hypothesis there are circuits to compute these functions.

It is now an easy matter to design a circuit which computes f . The circuit computes both f_0 and f_1 on the last n bits of the input. Then, depending on whether the first bit of the input was a 0 or a 1 it outputs the appropriate answer. A circuit to do this is shown in the following figure. This completes the induction.

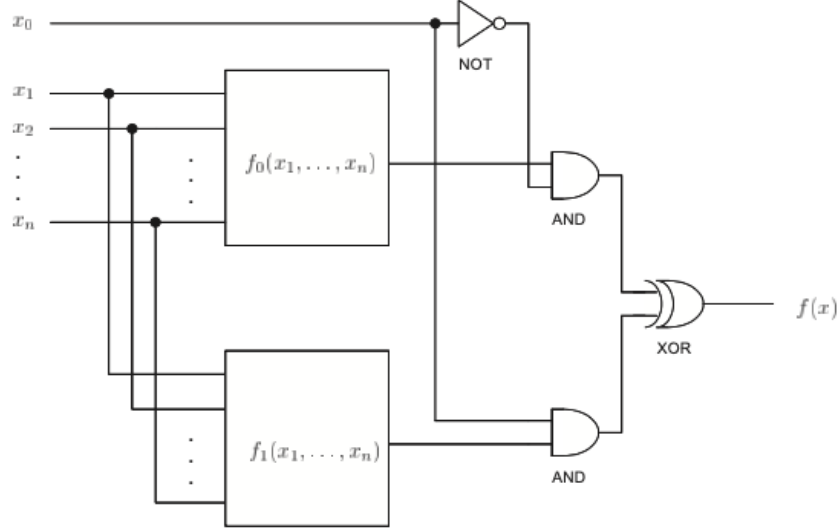


Figure 6: Circuit to compute an arbitrary function f on $n + 1$ bits, assuming by induction that there are circuits to compute the n -bit functions f_0 and f_1 .

Let's return from our brief quantum digression, to the properties of classical circuits. We claimed earlier that the Turing machine model is equivalent to the circuit model of computation. In what sense do we mean the two models are equivalent? On the face of it, the two models appear quite different. The unbounded nature of a Turing machine makes them more useful for abstractly specifying what it is we mean by an algorithm, while circuits more closely capture what an actual physical computer does.

The two models are connected by introducing the notion of a *uniform circuit family*. A circuit family consists of a collection of circuits, $\{C_n\}$, indexed by a positive integer n . The circuit C_n has n input bits, and may have any finite number of extra work bits, and output bits. The output of the circuit C_n , upon input of a number x of at most n bits in length, is denoted by $C_n(x)$. We require that the circuits be consistent, that is, if mn and x is at most m bits in length, then $C_m(x) = C_n(x)$. The function computed by the circuit family $\{C_n\}$ is the function $C(\cdot)$ such that if x is n bits in length then $C(x) = C_n(x)$. For example, consider a circuit C_n that squares an n -bit number. This defines a family of circuits C_n that computes the function, $C(x) = x^2$, where x is any positive integer.

In this section we saw the two models of computation and realized that the circuit model is a realistic one. In the next section we will learn about quantum circuits which are at the heart of realizing a quantum computer.

4 Quantum circuits

In the previous chapter, we discussed the classical models of computation including classical circuits. In this chapter, we will discuss about the quantum circuits which lie at the heart of quantum computation. There are two main ideas introduced in this chapter. First, we explain in detail the fundamental model of quantum computation, the quantum circuit model. Second, we demonstrate that there exists a small set of gates which are *universal*, that is, any quantum computation whatsoever can be expressed in terms of those gates.

4.1 Single qubit operations

As we have already seen previously that a qubit is a vector $|\psi\rangle = a|0\rangle + b|1\rangle$ parameterized by two complex numbers satisfying $|a|^2 + |b|^2 = 1$. Operations on a qubit must preserve this norm, and thus are described by 2×2 unitary matrices. It should be noted that *all* unitary matrices are valid gates for a qubit. Here are some commonly used single qubit gates:







Hadamard		$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Pauli-X		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Phase		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$

Figure 7: Names, symbols, and unitary matrices for the common single qubit gates.

4.2 Controlled operations

'If A is true, then do B'. This type of *controlled operation* is one of the most useful in computing, both classical and quantum.

Suppose we have a single qubit gate U . A *Controlled- U* gate is a two qubit gate which takes in an extra qubit called control qubit which is either $|0\rangle$ or $|1\rangle$. If it is $|1\rangle$, then the gate U is applied on the other qubit (target qubit). However, if it is $|0\rangle$, then the other qubit is not changed. Mathematically, $|c\rangle|t\rangle \rightarrow |c\rangle U^c|t\rangle$ where $U^c = U$ if $c = |1\rangle$ and $U^c = I$ if $c = |0\rangle$. It is represented as follows.

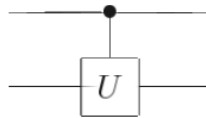


Figure 8: Controlled- U operation

If U is the NOT gate, then we get what is called the CNOT gate. It is a very useful gate as we will see later and has the following matrix representation.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

The advantage of CNOT gate that any general controlled- U gate can easily be derived CNOT gate by using the circuit given below.

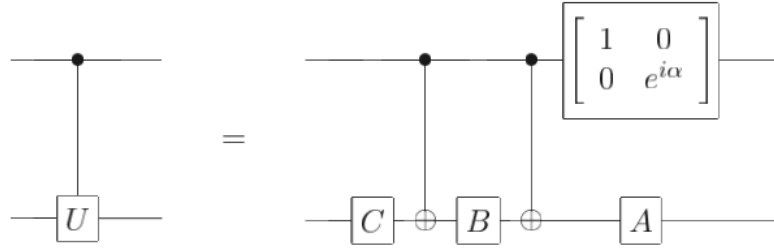


Figure 9: Circuit implementing the controlled- U operation for single qubit gate U . α , A , B and C satisfy $U = e^{i\alpha} A \times B \times C$, $ABC = I$.

4.3 Measurement

A final element used in quantum circuits, almost implicitly sometimes, is measurement. In circuit diagrams, a measurement is represented using the 'meter' symbol. In the theory of quantum circuits it is conventional to not use any special symbols to denote more general measurements, because they can always be represented by unitary transforms with ancilla qubits followed by projective measurements.

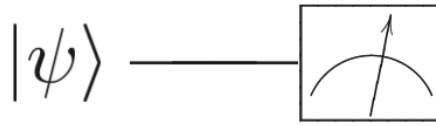


Figure 10: Symbol for measurement on a single qubit.

There are two important principles that it is worth bearing in mind about quantum circuits.

1. **Principle of deferred measurement:** Measurements can always be moved from an intermediate stage of a quantum circuit to the end of the circuit; if the measurement results are used at any stage of the circuit then the classically controlled operations can be replaced by conditional quantum operations.
2. **Principle of implicit measurement:** Without loss of generality, any unterminated quantum wires (qubits which are not measured) at the end of a quantum circuit may be assumed to be measured.

4.4 Universal quantum gates

A set of gates is said to be universal for quantum computation if any unitary operation may be approximated to arbitrary accuracy by a quantum circuit involving only those gates.

We now describe three universality constructions for quantum computation.

4.4.1 Two-level unitary gates are universal

Consider a unitary matrix U which acts on a d -dimensional Hilbert space. In this section we explain how U may be decomposed into a product of two-level unitary matrices; that is, unitary matrices which act non-trivially only on two-or-fewer vector components. The essential idea behind this decomposition may be understood by considering the case when U is 3×3 , so suppose that U has the form

$$U = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & j \end{bmatrix}$$

We will find two-level unitary matrices U_1, U_2, U_3 such that

$$U_3 U_2 U_1 U = I$$

It follows that

$$U = U_1^\dagger U_2^\dagger U_3^\dagger$$

U_1, U_2 and U_3 are all two-level unitary matrices, and it is easy to see that their inverses, U_1^\dagger, U_2^\dagger and U_3^\dagger are also two-level unitary matrices.

Use the following procedure to construct U_1 : if $b = 0$ then set

$$U_1 \equiv \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

If $b \neq 0$ then set

$$U_1 \equiv \begin{bmatrix} \frac{a^*}{\sqrt{|a|^2+|b|^2}} & \frac{b^*}{\sqrt{|a|^2+|b|^2}} & 0 \\ \frac{b}{\sqrt{|a|^2+|b|^2}} & \frac{-a}{\sqrt{|a|^2+|b|^2}} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Note that in either case U_1 is a two-level unitary matrix, and when we multiply the matrices out we get

$$U_1 U = \begin{bmatrix} a' & d' & g' \\ 0 & e' & h' \\ c' & f' & j' \end{bmatrix}$$

The key point to note is that the middle entry in the left hand column is zero. Now apply a similar procedure to find a two-level matrix U_2 such that $U_2 U_1 U$ has no entry in the bottom left corner. That is, if $c' = 0$ we set

$$U_2 \equiv \begin{bmatrix} a'^* & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

while if $c' \neq 0$ then we set

$$U_2 \equiv \begin{bmatrix} \frac{a'^*}{\sqrt{|a'|^2+|c'|^2}} & 0 & \frac{a'^*}{\sqrt{|a'|^2+|c'|^2}} \\ 0 & 1 & 0 \\ \frac{c'}{\sqrt{|a'|^2+|c'|^2}} & 0 & \frac{-a'}{\sqrt{|a'|^2+|c'|^2}} \end{bmatrix}$$

In either case, when we carry out the matrix multiplication we find that

$$U_2 U_1 U = \begin{bmatrix} 1 & d'' & g'' \\ 0 & e'' & h'' \\ 0 & f'' & j'' \end{bmatrix}$$

Since U , U_1 and U_2 are unitary, it follows that $U_2 U_1 U$ is unitary, and thus $d'' = g'' = 0$ since the first row of $U_2 U_1 U$ must have norm 1. Finally, set

$$U_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & e''^* & h''^* \\ 0 & f''^* & j''^* \end{bmatrix}$$

It is now easy to verify that $U_3 U_2 U_1 U = I$, and thus $U = U_1^\dagger U_2^\dagger U_3^\dagger$, which is a decomposition of U into two-level unitaries.

More generally, suppose U acts on a d -dimensional space. Then, in a similar fashion to the 3×3 case, we can find two-level unitary matrices U_1, \dots, U_{d-1} such that the matrix $U_{d-1} U_{d-2} \dots U_1 U$ has a one in the top left hand corner, and all zeroes elsewhere in the first row and column. We then repeat this procedure for the $d-1$ by $d-1$ unitary submatrix in the lower right hand corner of $U_{d-1} U_{d-2} \dots U_1 U$, and so on, with the end result that an arbitrary $d \times d$ unitary matrix may be written

$$U = V_1 \dots V_k$$

where the matrices V_i are two-level unitary matrices, and $k \leq (d-1) + (d-2) + \dots + 1 = d(d-1)/2$.

4.4.2 Single qubit and CNOT gates are universal

Suppose U is a two-level unitary matrix on an n qubit quantum computer. Suppose in particular that U acts non-trivially on the space spanned by the computational basis states $|s\rangle$ and $|t\rangle$, where $s = s_1 \dots s_n$ and $t = t_1 \dots t_n$ are the binary expansions for s and t . Let \tilde{U} be the non-trivial 2×2 unitary submatrix of U ; \tilde{U} can be thought of as a unitary operator on a single qubit.

Our immediate goal is to construct a circuit implementing U , built from single qubit and CNOT gates. To do this, we need to make use of *Gray codes*. Suppose we have distinct binary numbers, s and t . A Gray code connecting s and t is a sequence of binary numbers, starting with s and concluding with t , such that adjacent members of the list differ in exactly one bit. For instance, with $s = 101001$ and $t = 110011$ we have the Gray code

$$\begin{array}{cccccc} 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \end{array}$$

Let g_1 through g_m be the elements of a Gray code connecting s and t , with $g_1 = s$ and $g_m = t$. Note that we can always find a Gray code such that $m \leq n+1$ since s and t can differ in at most n locations.

Now, the first step is to swap the states $|g_1\rangle$ and $|g_2\rangle$. Suppose g_1 and g_2 differ at the i th digit. Then we accomplish the swap by performing a controlled bit flip on the i th qubit, conditional on the values of the other qubits being identical to those in both g_1 and g_2 . Next we use a controlled operation to swap $|g_2\rangle$ and $|g_3\rangle$. We continue in this fashion until we swap $|g_{m-2}\rangle$ with $|g_{m-1}\rangle$. The effect of this sequence of $m-2$ operations is to achieve the operation

$$\begin{array}{l} |g_1\rangle \rightarrow |g_{m-1}\rangle \\ |g_2\rangle \rightarrow |g_1\rangle \\ |g_3\rangle \rightarrow |g_2\rangle \\ \dots \\ |g_{m-1}\rangle \rightarrow |g_{m-2}\rangle \end{array}$$

All other computational basis states are left unchanged by this sequence of operations. Next, suppose g_{m1} and g_m differ in the j th bit. We apply a controlled- \tilde{U} operation with the j th qubit as target, conditional on the other qubits having the same values as appear in both g_m and g_{m1} . Finally, we complete the U operation by undoing the swap operations: we swap $|g_{m-1}\rangle$ with $|g_{m-2}\rangle$, then $|g_{m-2}\rangle$ with $|g_{m-3}\rangle$ and so on, until we swap $|g_2\rangle$ with $|g_1\rangle$.

We see that implementing the two-level unitary operation U requires at most $2(n1)$ controlled operations to swap $|g_1\rangle$ with $|g_{m-1}\rangle$ and then back again. Each of these controlled operations can be realized using $O(n)$ single qubit and CNOT gates; the controlled- \tilde{U} operation also requires $O(n)$ gates. Thus, implementing U requires $O(n^2)$ single qubit and CNOT gates. We saw in the previous section that an arbitrary unitary matrix on the $2n$ -dimensional state space of n qubits may be written as a product of $O(2^{2n}) = O(4^n)$ two-level unitary operations. Combining these results, we see that an arbitrary unitary operation on n qubits can be implemented using a circuit containing $O(n^2 4^n)$ single qubit and CNOT gates.

5 The quantum Fourier transform and its applications

In the introduction, we stated that quantum computers can factor numbers much faster than a classical computer. To put this in perspective, finding the prime factorization of an n -bit integer is thought to require $\exp(O(n^{1/3} \log^{2/3} n))$ operations using the best classical algorithm known, the so-called number *field sieve*. In contrast, a quantum algorithm can accomplish the same task using $O(n^2 \log n \log \log n)$ operations. That is, a quantum computer can factor a number *exponentially* faster than the best known classical algorithms.

But now the question arises, what other tasks a quantum computer can do more efficiently than a classical computer? In this chapter, we talk about the *quantum Fourier transform* which is the key ingredient for quantum factoring and many other interesting quantum algorithms.

5.1 The quantum Fourier transform

A very useful and common way to solve a problem in computer science is to break it down or *transform* it into other problem(s) whose solution is known. A great discovery of quantum computation has been that some such transformations can be computed much faster on a quantum computer than on a classical computer.

One such transformation is the discrete Fourier transform. In the usual mathematical notation, the discrete Fourier transform takes as input a vector of complex numbers, x_0, \dots, x_{N-1} where the length N of the vector is a fixed parameter. It outputs the transformed data, a vector of complex numbers y_0, \dots, y_{N-1} , defined by

$$y_k \equiv \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i j k / N}$$

The quantum Fourier transform is exactly the same transformation, although the conventional notation for the quantum Fourier transform is somewhat different. The quantum Fourier transform on an orthonormal basis $|0\rangle, \dots, |N-1\rangle$ is defined to be a linear operator with the following action on the basis states,

$$|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i j k / N} |k\rangle$$

Equivalently, the action on an arbitrary state may be written

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

where the amplitudes y_k are the discrete Fourier transform of the amplitudes x_j . Now we shall demonstrate the unitarity of the Fourier transform by constructing a manifestly unitary quantum circuit computing the Fourier transform.

In the following, we take $N = 2^n$, where n is some integer, and the basis $|0\rangle, \dots, |2^n - 1\rangle$ is the computational basis for an n qubit quantum computer. It is helpful to write the state $|j\rangle$ using the binary representation $j = j_1 j_2 \dots j_n$. More formally, $j = j_1 2^{n-1} + j_2 2^{n-2} + \dots + j_n 2^0$. It is also convenient to adopt the notation $0.j_1 j_2 \dots j_n$ to represent the binary fraction $j_1/2 + j_2/4 + \dots + j_n/2^{n-l+1}$.

With a little algebra the quantum Fourier transform can be given the following useful *product representation*:

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

The equivalence of the product representation and the definition follows from some elementary algebra:

$$\begin{aligned}
|j\rangle &\rightarrow \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2\pi i j k / 2^n} |k\rangle \\
&= \frac{1}{2^{n/2}} \sum_{k_1=0}^1 \cdots \sum_{k_n=0}^1 e^{2\pi i j (\sum_{l=1}^n k_l 2^{-l})} |k_1 \dots k_n\rangle \\
&= \frac{1}{2^{n/2}} \sum_{k_1=0}^1 \cdots \sum_{k_n=0}^1 \bigotimes_{l=1}^n e^{2\pi i j k_l 2^{-l}} |k_l\rangle \\
&= \frac{1}{2^{n/2}} \bigotimes_{l=1}^n \left[\sum_{k_l=0}^1 e^{2\pi i j k_l 2^{-l}} |k_l\rangle \right] \\
&= \frac{1}{2^{n/2}} \bigotimes_{l=1}^n \left[|0\rangle + e^{2\pi i j 2^{-l}} |1\rangle \right] \\
&= \frac{\left(|0\rangle + e^{2\pi i 0 \cdot j_n} |1\rangle \right) \left(|0\rangle + e^{2\pi i 0 \cdot j_{n-1} j_n} |1\rangle \right) \cdots \left(|0\rangle + e^{2\pi i 0 \cdot j_1 j_2 \dots j_n} |1\rangle \right)}{2^{n/2}}
\end{aligned}$$

The product representation makes it easy to derive an efficient circuit for the quantum Fourier transform. Such a circuit is shown in the figure. The gate R_k denotes the unitary transformation

$$R_k \equiv \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i / 2^k} \end{bmatrix}$$

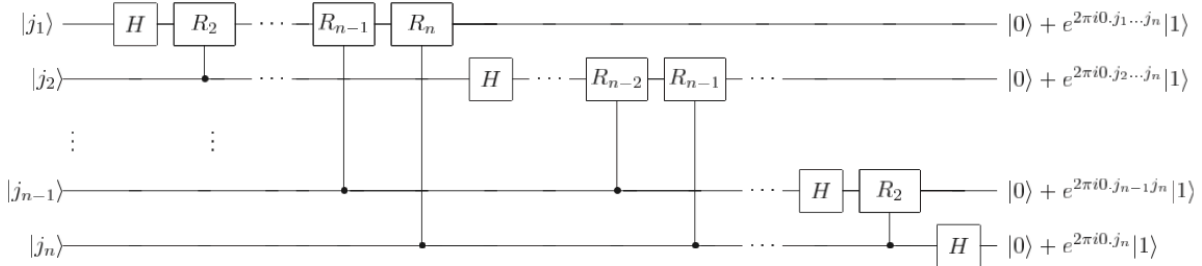


Figure 11: Efficient circuit for the quantum Fourier transform.

5.2 Phase estimation

The Fourier transform is the key to a general procedure known as *phase estimation*, which in turn is the key for many quantum algorithms. Suppose a unitary operator U has an eigenvector $|u\rangle$ with eigenvalue $e^{2\pi i \phi}$, where the value of ϕ is unknown. The goal of the phase estimation algorithm is to estimate ϕ . To do this, we will assume that we already have available some *black boxes* capable of preparing the state $|u\rangle$ and performing the controlled- U^{2^j} operation, for suitable non-negative integers j .

The quantum phase estimation procedure uses two registers. The first register contains t qubits initially in the state $|0\rangle$. How we choose t depends on two things: the number of digits of accuracy we wish to have in our estimate for ϕ , and with what probability we wish the phase estimation procedure to be successful. The second register begins in the state $|u\rangle$, and contains as many qubits as is necessary to store $|u\rangle$. Phase estimation is performed in two stages. First, we apply the circuit shown in the figure. The circuit begins by applying a Hadamard transform to the first register, followed by application of controlled- U operations on

the second register, with U raised to successive powers of two. The final state of the first register is easily seen to be:

$$\frac{1}{2^{t/2}} \left(|0\rangle + e^{2\pi i 2^{t-1} \phi} |1\rangle \right) \left(|0\rangle + e^{2\pi i 2^{t-2} \phi} |1\rangle \right) \dots \left(|0\rangle + e^{2\pi i 2^0 \phi} |1\rangle \right) = \frac{1}{2^{t/2}} \sum_{k=0}^{2^t-1} e^{2\pi i \phi k} |k\rangle$$

We omit the second register from this description, since it stays in the state $|u\rangle$ throughout the computation.

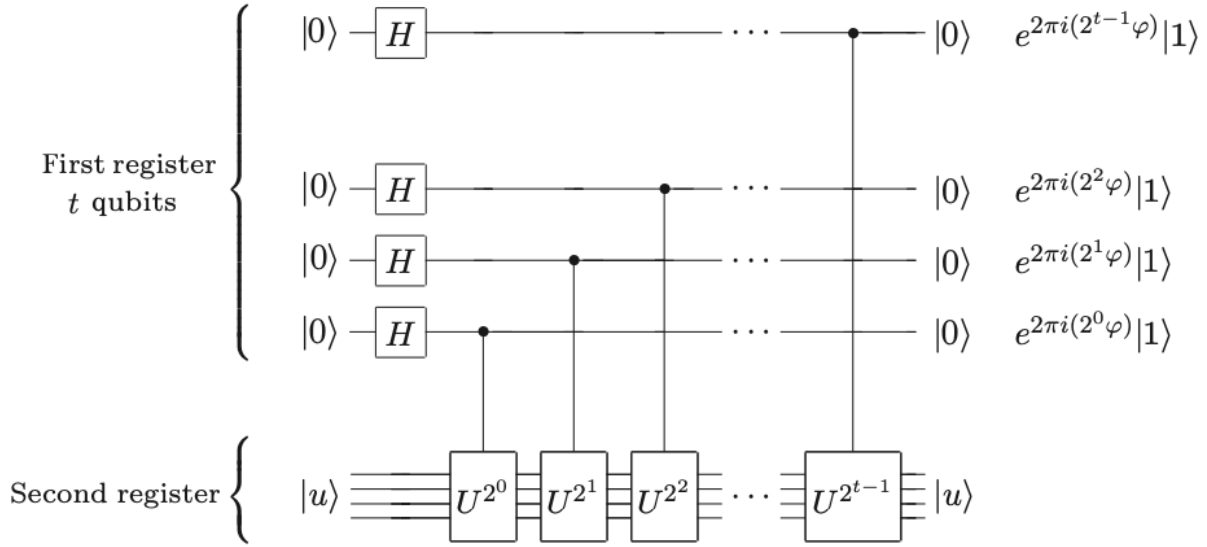


Figure 12: The first stage of the phase estimation procedure.

The second stage of phase estimation is to apply the inverse quantum Fourier transform on the first register. This is obtained by reversing the circuit for the quantum Fourier transform, and can be done in $\Theta(t^2)$ steps. The third and final stage of phase estimation is to read out the state of the first register by doing a measurement in the computational basis. We will show that this provides a pretty good estimate of ϕ . An overall schematic of the algorithm is shown below.

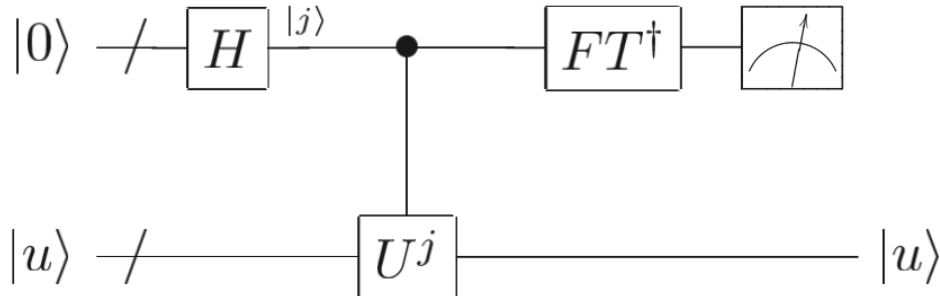


Figure 13: Schematic of the overall phase estimation procedure.

To see why phase estimation works, suppose ϕ may be expressed exactly in t bits, as $\phi = 0.\phi_1 \dots \phi_t$. Then the state resulting from the first stage of phase estimation may be rewritten

$$\frac{1}{2^{t/2}} \left(|0\rangle + e^{2\pi i 0.\phi_t} |1\rangle \right) \left(|0\rangle + e^{2\pi i 0.\phi_{t-1}\phi_t} |1\rangle \right) \dots \left(|0\rangle + e^{2\pi i 0.\phi_1 \dots \phi_t} |1\rangle \right)$$

The second stage of phase estimation is to apply the inverse quantum Fourier transform. But comparing the previous equation with the product form for the Fourier transform, we see that the output state from the second stage is the product state $|\phi_1 \dots \phi_t\rangle$. A measurement in the computational basis therefore gives us ϕ exactly!

5.3 Applications

5.3.1 Application: order-finding

For positive integers x and N , $x < N$, with no common factors, the *order* of x modulo N is defined to be the least positive integer, r , such that $x^r = 1 \pmod{N}$. The order-finding problem is to determine the order for some specified x and N . Order-finding is believed to be a hard problem on a classical computer, in the sense that no algorithm is known to solve the problem using resources polynomial in the $O(L)$ bits needed to specify the problem, where $L \equiv \lceil \log(N) \rceil$ is the number of bits needed to specify N . In this section we explain how phase estimation may be used to obtain an efficient quantum algorithm for order-finding.

The quantum algorithm for order-finding is just the phase estimation algorithm applied to the unitary operator

$$U |y\rangle \equiv |xy \pmod{N}\rangle$$

A simple calculation shows that the states defined by

$$|u_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left\{\frac{-2\pi i s k}{r}\right\} |x^k \pmod{N}\rangle,$$

for integer $0 \leq s \leq r-1$ are eigenstates of U , since

$$\begin{aligned} U |u_s\rangle &= \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left\{\frac{-2\pi i s k}{r}\right\} |x^{k+1} \pmod{N}\rangle \\ &= \exp\left\{\frac{2\pi i s}{r}\right\} |u_s\rangle \end{aligned}$$

Using the phase estimation procedure allows us to obtain, with high accuracy, the corresponding eigenvalues $\exp(2\pi i s/r)$, from which we can obtain the order r with a little bit more work.

5.3.2 Application: factoring

The *factoring problem* goes as follows : given any integer N , find all of its prime factors and the exponent with which they appear. This factoring problem turns out to be equivalent to the order-finding problem in the sense that a fast algorithm for order-finding can easily be turned into a fast algorithm for factoring.

The reduction of factoring to order-finding proceeds in two basic steps. The first step is to show that we can compute a factor of N if we can find a non-trivial solution $x \not\equiv \pm 1 \pmod{N}$ to the equation $x^2 = 1 \pmod{N}$. The second step is to show that a randomly chosen y co-prime to N is quite likely to have an order r which is even, and such that $y^{r/2} \not\equiv \pm 1 \pmod{N}$, and thus $x \equiv y^{r/2} \pmod{N}$ is a non-trivial solution to $x^2 = 1 \pmod{N}$.

Theorem 5.1. Suppose N is an L bit composite number, and x is a non-trivial solution to the equation $x^2 = 1 \pmod{N}$ in the range $1 \leq x \leq N$, that is, neither $x = 1 \pmod{N}$ nor $x = N-1 = -1 \pmod{N}$. Then at least one of $\gcd(x-1, N)$ and $\gcd(x+1, N)$ is a non-trivial factor of N that can be computed using $O(L^3)$ operations.

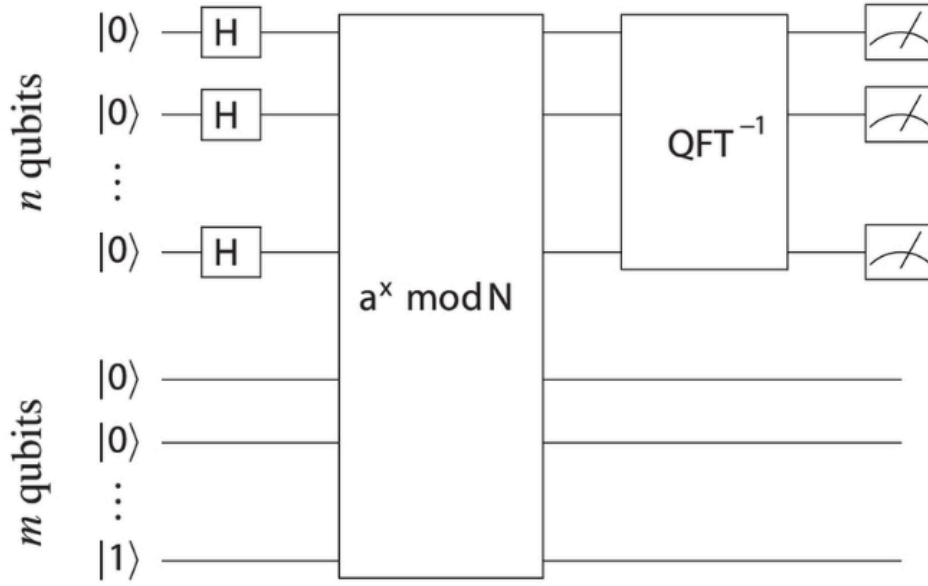


Figure 14: Circuit diagram for order finding algorithm.

Theorem 5.2. Suppose $N = p_1^{\alpha_1} \dots p_m^{\alpha_m}$ is the prime factorization of an odd composite positive integer. Let x be an integer chosen uniformly at random, subject to the requirements that $1 \leq x \leq N - 1$ and x is co-prime to N . Let r be the order of x modulo N . Then

$$p(r \text{ is even and } x^{r/2} \not\equiv -1 \pmod{N}) \geq 1 - \frac{1}{2^m}$$

Above theorems can be combined to give an algorithm which, with high probability, returns a non-trivial factor of any composite N . All the steps in the algorithm can be performed efficiently on a classical computer except (so far as is known today) an order-finding 'subroutine' which is used by the algorithm. By repeating the procedure we may find a complete prime factorization of N . The algorithm is summarized below.

Algorithm: Reduction of factoring to order-finding

Inputs: A composite number N .

Outputs: A non-trivial factor of N .

Runtime: $O((\log N)^3)$ operations. Succeeds with probability $O(1)$.

Procedure:

1. If N is even, return the factor 2.
2. Determine whether $N = a^b$ for integers $a \geq 1$ and $b \geq 2$, and if so return the factor a .
3. Randomly choose x in the range 1 to $N - 1$. If $\gcd(x, N) > 1$ then return the factor $\gcd(x, N)$.
4. Use the order-finding subroutine to find the order r of x modulo N .
5. If r is even and $x^{r/2} \not\equiv -1 \pmod{N}$ then compute $\gcd(x^{r/2} - 1, N)$ and $\gcd(x^{r/2} + 1, N)$, and test to see if one of these is a non-trivial factor, returning that factor if so. Otherwise, the algorithm fails.

5.3.3 Application : Period-finding

Consider the following problem. Suppose f is a periodic function producing a single bit as output and such that $f(x+r) = f(x)$, for some unknown $0 < r < 2L$, where $x, r \in \{0, 1, 2, \dots\}$. Given a quantum black box U which performs the unitary transform $U|x\rangle|y\rangle \rightarrow |x\rangle|y \oplus f(x)\rangle$ (where \oplus denotes addition modulo 2) how many black box queries and other operations are required to determine r ? Note that in practice U operates on a finite domain, whose size is determined by the desired accuracy for r . Here is a quantum algorithm which solves this problem using one query, and $O(L^2)$ other operations:

Algorithm: Period-finding

Inputs: (1) A black box which performs the operation $U|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$, (2) a state to store the function evaluation, initialized to $|0\rangle$, and (3) $t = O(L + \log(1/\epsilon))$ qubits initialized to $|0\rangle$.

Outputs: The least integer $r > 0$ such that $f(x+r) = f(x)$.

Runtime: One use of U , and $O(L^2)$ operations. Succeeds with probability $O(1)$.

Procedure:

- | | | |
|----|--|---|
| 1. | $ 0\rangle 0\rangle$ | initial state |
| 2. | $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{x=0}^{2^t-1} x\rangle 0\rangle$ | create superposition |
| 3. | $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{x=0}^{2^t-1} x\rangle f(x)\rangle$
$\approx \frac{1}{\sqrt{r}2^t} \sum_{\ell=0}^{r-1} \sum_{x=0}^{2^t-1} e^{2\pi i \ell x / r} x\rangle \hat{f}(\ell)\rangle$ | apply U |
| 4. | $\rightarrow \frac{1}{\sqrt{r}} \sum_{\ell=0}^{r-1} \widehat{\ell/r}\rangle \hat{f}(\ell)\rangle$ | apply inverse Fourier transform to first register |
| 5. | $\rightarrow \widehat{\ell/r}$ | measure first register |
| 6. | $\rightarrow r$ | apply continued fractions algorithm |

6 Distance measures for quantum information

What does it mean to say that two items of information are similar? What does it mean to say that information is preserved by some process? These questions are central to a theory of quantum information processing, and the purpose of this chapter is the development of distance measures giving quantitative answers to these questions. Motivated by our two questions we will be concerned with two broad classes of distance measures, *static measures* and *dynamic measures*. Static measures quantify how close two quantum states are, while dynamic measures quantify how well information has been preserved during a dynamic process.

6.1 Distance measures for classical information

One way of quantifying the distance between two strings of bits is the *Hamming distance*, defined to be the number of places at which two bit strings are not equal. For example, the bit strings 00010 and 10011 differ in the first and last place, so the Hamming distance between them is two. Unfortunately, the Hamming distance between two objects is simply a matter of labeling, and *a priori* there aren't any labels in the Hilbert space arena of quantum mechanics!

The first measure is the *trace distance*, defined by the equation:

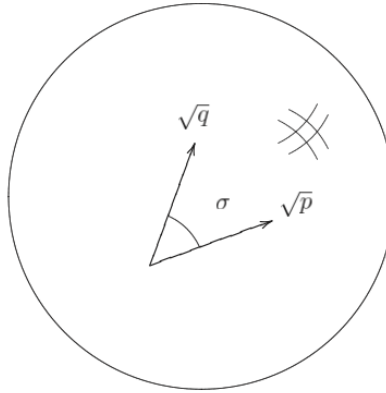
$$D(p_x, q_x) \equiv \frac{1}{2} \sum_x |p_x - q_x|$$

This quantity is sometimes known as the L_1 distance or *Kolmogorov distance*. The trace distance turns out to be a metric on probability distributions, so the use of the term 'distance' is justified.

A second measure of distance between probability distributions, the *fidelity* of the probability distributions $\{p_x\}$ and $\{q_x\}$, is defined by

$$F(p_x, q_x) \equiv \sum_x \sqrt{p_x q_x}$$

The fidelity is a very different way of measuring distance between probability distributions than is the trace distance. The fidelity is just the inner product between vectors with components $\sqrt{p_x}$ and $\sqrt{q_x}$, which lie on a unit sphere.



$$F(p, q) = \sqrt{p} \cdot \sqrt{q} = \cos(\sigma)$$

Figure 15: Geometric interpretation of the fidelity as the inner product between vectors $\sqrt{p_x}$ and $\sqrt{q_x}$ lying on a unit sphere.

6.2 How close are two quantum states?

6.2.1 Trace

Analogous to the classical case, we begin by defining the *trace distance* between quantum states ρ and σ ,

$$D(\rho, \sigma) \equiv \frac{1}{2} \text{tr} |\rho - \sigma|$$

where as per usual we define $|A| \equiv \sqrt{A^\dagger A}$ to be the positive square root of $A^\dagger A$. Notice that the quantum trace distance generalizes the classical trace distance in the sense that if ρ and σ commute then the (quantum) trace distance between ρ and σ is equal to the classical trace distance between the eigenvalues of ρ and σ . More explicitly, if ρ and σ commute they are diagonal in the same basis,

$$\rho = \sum_i r_i |i\rangle \langle i|; \quad \sigma = \sum_i s_i |i\rangle \langle i|,$$

for some orthonormal basis $|i\rangle$. Thus

$$\begin{aligned} D(\rho, \sigma) &= \frac{1}{2} \text{tr} \left| \sum_i (r_i - s_i) |i\rangle \langle i| \right| \\ &= D(r_i, s_i) \end{aligned}$$

Theorem 6.1. Let $\{E_m\}$ be a POVM, with $p_m \equiv \text{tr}(\rho E_m)$ and $q_m \equiv \text{tr}(\sigma E_m)$ as the probabilities of obtaining a measurement outcome labeled by m . Then

$$D(\rho, \sigma) = \max_{\{E_m\}} D(p_m, q_m)$$

where the maximization is over all POVMs $\{E_m\}$.

Proof. Note that

$$D(p_m, q_m) = \frac{1}{2} \sum_m |\text{tr}(E_m(\rho\sigma))|$$

Using the spectral decomposition we may write $\rho\sigma = QS$, where Q and S are positive operators with orthogonal support. Thus $|\rho\sigma| = Q + S$, and

$$\begin{aligned} |\text{tr}(E_m(\rho\sigma))| &= |\text{tr}(E_m(Q - S))| \\ &\leq |\text{tr}(E_m(Q + S))| \\ &\leq |\text{tr}(E_m|\rho - \sigma|)| \end{aligned}$$

Thus

$$\begin{aligned} D(p_m, q_m) &\leq \frac{1}{2} \sum_m \text{tr}(E_m|\rho\sigma|) \\ &= \frac{1}{2} \text{tr} |\rho - \sigma| \\ &= D(\rho, \sigma) \end{aligned}$$

where we have applied the completeness relation for POVM elements, $\sum_m E_m = I$.

Conversely, by choosing a measurement whose POVM elements include projectors onto the support of Q and S , we see that there exist measurements which give rise to probability distributions such that $D(p_m, q_m) = D(\rho, \sigma)$. \square

6.2.2 Fidelity

A second measure of distance between quantum states is the *fidelity*. The fidelity of states ρ and σ is defined to be

$$F(\rho, \sigma) = \text{tr} \sqrt{\rho^{1/2} \sigma \rho^{1/2}}$$

Theorem 6.2. (Uhlmann's theorem) Suppose ρ and σ are states of a quantum system Q . Introduce a second quantum system R which is a copy of Q . Then

$$F(\rho, \sigma) = \max_{|\psi\rangle, |\phi\rangle} |\langle \psi | \phi \rangle|$$

where the maximization is over all purifications $|\psi\rangle$ of ρ and $|\phi\rangle$ of σ into RQ .

Theorem 6.3. (Monotonicity of the fidelity) Suppose \mathcal{E} is a trace-preserving quantum operation. Let ρ and σ be density operators. Show that

$$F(\mathcal{E}(\rho), \mathcal{E}(\sigma)) \geq F(\rho, \sigma)$$

Proof. Let $|\psi\rangle$ and $|\phi\rangle$ be purifications of ρ and σ into a joint system RQ such that $F(\rho, \sigma) = |\langle \psi | \phi \rangle|$. Introduce a model environment E for the quantum operation, \mathcal{E} , which starts in a pure state $|0\rangle$, and interacts with the quantum system Q via a unitary interaction U . Note that $U |\psi\rangle |0\rangle$ is a purification of $\mathcal{E}\rho$, and $U |\phi\rangle |0\rangle$ is a purification of $\mathcal{E}\sigma$. By Uhlmann's theorem it follows that

$$\begin{aligned} F(\mathcal{E}\rho, \mathcal{E}\sigma) &\geq |\langle \psi | \langle 0 | U^\dagger U | \phi \rangle |0\rangle| \\ &= |\langle \psi | \phi \rangle| \\ &= F(\rho, \sigma) \end{aligned}$$

□