

Circuits, Algorithms, and Error Correction in Quantum Computing

Introduction to Quantum Computing (U. Cincinnati)

Contents

0	Qubit quantum mechanics	2
0.1	Basic computations in quantum mechanics	2
0.2	Single qubit gates and Pauli matrices	9
0.3	Tensor products	12
0.4	Entanglement	15
I	Quantum Circuits	17
1	Simple circuits	17
1.1	The no-cloning theorem, controlled-NOT and related gates	17
1.2	Quantum teleportation and encryption	21
2	Some basic quantum algorithms	30
2.1	Function gates	31
2.2	Deutsch-Jozsa algorithm	33
2.3	Grover's algorithm	36
II	Cryptosystems and Shor's Algorithm	44
3	Factorization, public key cryptosystems, and the order function	44
3.1	Factorization	45
3.2	RSA cryptosystem	46
3.3	The order function and factorization	47
3.4	Appendix: Modular arithmetic	49
4	Quantum Fourier transform	55
5	Shor's quantum algorithm for the order function	60
5.1	Shor's algorithm	60
5.2	Example: quantum computation of $\text{ord}(10, 21)$	62
5.3	Appendix: Rational approximant algorithms	66
5.4	Appendix: The behavior of Shor's algorithm	68
III	Noise and Error Correction	71

6	The need for quantum error correction	71
7	Models of noise	73
7.1	Classical noise	74
7.2	Density matrices	75
7.3	Quantum operations	82
8	Error correction for noisy channels	89
8.1	Classical majority voting code	89
8.2	Shor's code	91
8.3	Generalizations	101
9	Fault-tolerant quantum computation	103
10	The toric code	104
10.1	Setting up the computational basis	104
10.2	Error syndrome operators	106
10.3	The code subspace	109
10.4	Error correction	111
10.5	Generalization	114
IV	Topological Quantum Computation	116
11	General idea	116
12	Toric code as a model of a topological material	118
12.1	Insensitivity to the detailed microscopic interactions	118
12.2	Quasiparticles of the toric code	119
13	Generalization to non-abelian anyons	122

Much of these notes vaguely follow the treatment and notation of M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information* (Cambridge, 2000), which will be referred to as [NC] when specific page references are needed. Part I covers the basics of quantum gates, circuits, and algorithms, and is loosely based on notes of Marc Cahay and Paul Esposito; it covers topics in chapters 2, 4, 5, and 6 of [NC]. Part II covers the Shor quantum factorization algorithm, and is adapted from notes written by George Purdy; it covers topics in chapter 5 and appendices 4 and 5 of [NC]. Part III covers error correction strategies for dealing with noise in quantum computation, and is adapted from notes written by Philip Argyres; it covers topics in chapters 8 and 10 of [NC]. Part IV is a brief gloss on topological quantum computation, and is loosely drawn from various reviews, notably J. Preskill's *Lecture Notes in Quantum Computation* (<http://www.theory.caltech.edu/~preskill/ph219/topological.pdf>).

0 Qubit quantum mechanics

We review here the basic rules of quantum mechanics as applied to a finite collection of qubits. We emphasize a tensor product formalism that simplifies many of the things we will do from now on in the course. It will be used extensively in the analysis of quantum algorithms later on and is an indispensable tool for the analysis of systems of many qubits.

0.1 Basic computations in quantum mechanics

The language of quantum mechanics is *complex linear algebra*. So in this subsection I quickly and compactly review the essential rules of complex linear algebra and state the rules of quantum mechanics in this language. This consists of three parts: **Hilbert spaces** (complex vector spaces with an inner product), **linear operators** (in particular hermitian, unitary, and projection operators and their spectral representations), and the physical **rules of quantum mechanics**. All the bold faced words and phrases in this subsection are concepts which are essential to any discussion of quantum mechanics, and should be mastered. A short (8-page) article designed specifically to introduce these essentials to computer scientists which you might find helpful is “From Cbits to Qbits: Teaching computer scientists quantum mechanics,” by N. David Mermin, *Am. J. Phys.* **71** (2003) 23-30.

To start, I assume that you are familiar with the basic algorithms of matrix algebra. These are, for square matrices, computing determinants, matrix inverses, eigenvalues, and eigenvectors; and for column vectors the Gram-Schmidt orthogonalization procedure for producing an orthogonal set of vectors with the same span as any given set of linearly independent vectors. For 3×3 and larger matrices or 3-component or larger vectors these algorithms are usually best done with the help of a computer algebra system (*e.g.*, *Mathematica*), but in the 2×2 case you should be able to rapidly carry out these calculations by hand.

Hilbert space review

Recall that an N -dimensional **Hilbert space**, \mathcal{H}_N , is the complex vector space \mathbb{C}^N with an **inner product**. We denote vectors in a Hilbert space by **kets**, $|\psi\rangle$, and their **hermitian conjugates** by **bras**, $(|\chi\rangle)^\dagger = \langle\chi|$, where hermitian conjugation acts by complex conjugation of scalars,

$$(\alpha|\psi\rangle + \beta|\chi\rangle)^\dagger = \alpha^*\langle\psi| + \beta^*\langle\chi|. \quad (1)$$

Here α and β denote arbitrary complex numbers, $\alpha, \beta \in \mathbb{C}$. We denote the inner product by the **bracket** $\langle\chi|\psi\rangle$. Recall that the bracket is a complex number, $\langle\chi|\psi\rangle \in \mathbb{C}$, and satisfies

$$\begin{aligned} \langle\chi|\psi\rangle &= \langle\psi|\chi\rangle^*, & \text{conjugation symmetry,} \\ \langle\psi|(\alpha|\phi\rangle + \beta|\xi\rangle) &= \alpha\langle\psi|\phi\rangle + \beta\langle\psi|\xi\rangle, & \alpha, \beta \in \mathbb{C}, \quad \text{linearity in the ket,} \\ \langle\psi|\psi\rangle &> 0, & \forall |\psi\rangle \neq 0, \quad \text{positive-definiteness.} \end{aligned} \quad (2)$$

The **norm** of a vector is $\|\psi\| := \sqrt{\langle\psi|\psi\rangle}$. The inner product is the complex analog of the dot product of real vectors.

A **qubit** is a vector in a 2-dimensional Hilbert space, \mathcal{H}_2 , of norm 1, i.e., $|\psi\rangle \in \mathbb{C}^2$ and $\langle\psi|\psi\rangle = 1$. Vectors of norm 1 are called **normalized**, and normalized vectors are also called **states**.

We choose an **orthonormal basis** $|0\rangle$ and $|1\rangle$ of \mathcal{H}_2 , which means that $\langle 0|0\rangle = \langle 1|1\rangle = 1$ and $\langle 0|1\rangle = 0$. Writing the basis as $|a\rangle$, $a = 0, 1$, this orthonormality can be written more compactly as

$$\langle a|b\rangle = \delta_{ab}, \quad (3)$$

where δ_{ab} is the Kronecker delta symbol: $\delta_{ab} = 1$ if $a = b$ and is zero otherwise. That $\{|a\rangle, a = 1, 2\}$ is a basis of \mathcal{H}_2 means any vector in \mathcal{H}_2 can be written uniquely as a linear combination,

$$|\psi\rangle = \psi_0|0\rangle + \psi_1|1\rangle = \sum_{a \in \{0,1\}} \psi_a |a\rangle, \quad (4)$$

for some complex numbers ψ_0, ψ_1 . Then

$$\langle\psi|\psi\rangle = \left(\sum_a \psi_a^* \langle a|\right) \left(\sum_b \psi_b |b\rangle\right) = \sum_{ab} \psi_a^* \psi_b \langle a|b\rangle = \sum_{ab} \psi_a^* \psi_b \delta_{ab} = \sum_a \psi_a^* \psi_a = \sum_a |\psi_a|^2. \quad (5)$$

Thus $|\psi\rangle$ is a qubit if it has unit norm, ie,

$$|\psi_0|^2 + |\psi_1|^2 = 1. \quad (6)$$

The ψ_a are the **components** of the state $|\psi\rangle$ in the $\{|0\rangle, |1\rangle\}$ basis, and are given by

$$\langle a|\psi\rangle = \langle a|\left(\sum_b \psi_b |b\rangle\right) = \sum_b \psi_b \langle a|b\rangle = \sum_b \psi_b \delta_{ab} = \psi_a. \quad (7)$$

We also say that ψ_a is the **amplitude** for the qubit $|\psi\rangle$ to be in state $|a\rangle$.

The fact that $\{|0\rangle, |1\rangle\}$ is an orthonormal basis can be encoded in the **completeness relation**

$$1 = \sum_{a \in \{0,1\}} |a\rangle \langle a| = |0\rangle \langle 0| + |1\rangle \langle 1|, \quad (8)$$

where the “1” on the left stands for the identity operator: $|\psi\rangle = 1|\psi\rangle$. Indeed, using the completeness relation, this becomes

$$|\psi\rangle = |0\rangle \langle 0|\psi\rangle + |1\rangle \langle 1|\psi\rangle = \psi_0|0\rangle + \psi_1|1\rangle. \quad (9)$$

There are infinitely many other orthonormal sets of basis vectors in \mathcal{H}_2 and some will be useful shortly.

Exercise 0.1 Show that

$$\begin{aligned} |0'\rangle &= \alpha|0\rangle + \beta|1\rangle \\ |1'\rangle &= \beta^*|0\rangle - \alpha^*|1\rangle \end{aligned} \quad (10)$$

is an orthonormal basis of \mathcal{H}_2 for any complex numbers α, β , such that $|\alpha|^2 + |\beta|^2 = 1$.

The “original” orthonormal basis, $\{|0\rangle, |1\rangle\}$, will be referred to as the **computational basis** from now on.

Orthonormal bases, components, and completeness relations all apply in a similar fashion to general N -dimensional Hilbert spaces, \mathcal{H}_N , not just the 2-dimensional space \mathcal{H}_2 of qubits.

Linear operator review

We now want to learn how to go from one orthonormal basis to another. We will discuss this in the context of qubits, but, again, the discussion generalizes immediately to arbitrary Hilbert spaces. Suppose that we have two sets of orthonormal basis vectors, $\{|0\rangle, |1\rangle\}$ and $\{|0'\rangle, |1'\rangle\}$. We write them as $|a\rangle$, and $|a'\rangle$, $a \in \{0, 1\}$. We are looking for a **linear operator**, $A : \mathcal{H}_2 \rightarrow \mathcal{H}_2$, that maps one basis to the other

$$A|a\rangle = |a'\rangle, \quad a \in \{0, 1\}. \quad (11)$$

Recall that a linear operator (also called a linear transformation) maps vectors to vectors, $A : |\psi\rangle \mapsto |A\psi\rangle = A|\psi\rangle$, in a linear way, $A(\alpha|\psi\rangle + \beta|\chi\rangle) = \alpha A|\psi\rangle + \beta A|\chi\rangle$. By linearity, if we know the action of an operator on a basis, as in (11), then we know it on all states: $A|\psi\rangle = A(\psi_0|0\rangle + \psi_1|1\rangle) = \psi_0 A|0\rangle + \psi_1 A|1\rangle = \psi_0|0'\rangle + \psi_1|1'\rangle := |\psi'\rangle$.

We call the set of complex numbers $A_{ab} := \langle a|A|b\rangle$ the **matrix elements** of A . We can see why by noting that if on an arbitrary state $A|\psi\rangle = |\chi\rangle$, then in components

$$\chi_a := \langle a|\chi\rangle = \langle a|A|\psi\rangle = \langle a|A\left(\sum_b |b\rangle\langle b|\right)|\psi\rangle = \sum_b \langle a|A|b\rangle\langle b|\psi\rangle = \sum_b A_{ab}\psi_b, \quad (12)$$

where we have used the definition (7) and the completeness relation (8). Let us now collect the 4 complex numbers A_{ab} as the 2×2 complex matrix

$$\mathbf{A} = \begin{pmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{pmatrix} \quad (13)$$

and the pairs of complex numbers ψ_a and χ_a as the column vectors

$$\boldsymbol{\psi} = \begin{pmatrix} \psi_0 \\ \psi_1 \end{pmatrix}, \quad \boldsymbol{\chi} = \begin{pmatrix} \chi_0 \\ \chi_1 \end{pmatrix}. \quad (14)$$

Then (12) just becomes the matrix equation $\boldsymbol{\chi} = \mathbf{A}\boldsymbol{\psi}$. Also, the matrix elements of a product of operators $C = AB$ are given by $C_{ab} = \langle a|C|b\rangle = \langle a|AB|b\rangle = \sum_c \langle a|A|c\rangle\langle c|B|b\rangle = \sum_c A_{ac}B_{cb}$ which is simply the matrix expression $\mathbf{C} = \mathbf{A}\mathbf{B}$. Finally, observe that the matrix elements of the hermitian conjugate operator are given by

$$(A^\dagger)_{ab} = \langle a|A^\dagger|b\rangle = \langle b|A|a\rangle^* = A_{ba}^*, \quad (15)$$

which is just $\mathbf{A}^\dagger := (\mathbf{A}^*)^t$, *i.e.*, the complex conjugate transpose matrix. The middle equality in (15) is actually the definition of what is meant by hermitian conjugation of an operator. An important consequence is that hermitian conjugation of products reverses order, so

$$(AB)^\dagger = B^\dagger A^\dagger, \quad (A|\psi\rangle)^\dagger = \langle\psi|A^\dagger, \quad (|\phi\rangle\langle\psi|)^\dagger = |\psi\rangle\langle\phi|. \quad (16)$$

These considerations show that kets and operators can be represented by complex column matrices and complex square matrices, respectively. But it is important to recognize that this association of vectors and operators to matrices *only makes sense relative to a choice of basis*. If one were to choose a different basis, then the matrices would change even though the vector or operator is unaffected. We have emphasized this distinction by making the matrix symbols boldface. But we will not do this from now on, and denote operators and their matrix representations by the same symbols; we thus rely on context to figure out with respect to which basis the matrix elements are computed.

Return now to the special operator A which transforms the $|a\rangle$ orthonormal basis into the $|a'\rangle$ orthonormal basis, (11). From the completeness relation and (11) we have

$$|b'\rangle = \sum_{a \in \{0,1\}} |a\rangle \langle a|b'\rangle = \sum_a |a\rangle \langle a|A|b\rangle := \sum_a |a\rangle A_{ab}. \quad (17)$$

Since $\{|b'\rangle, b' \in \{0,1\}\}$ is an orthonormal basis, (3) gives

$$\delta_{bc} = \langle b'|c'\rangle = \left(\sum_a \langle a|A_{ab}^* \right) \left(\sum_d |d\rangle A_{dc} \right) = \sum_{ad} \delta_{ad} A_{ab}^* A_{dc} = \sum_a A_{ab}^* A_{ac}. \quad (18)$$

These are the matrix components of the operator equation

$$AA^\dagger = A^\dagger A = 1. \quad (19)$$

Exercise 0.2 Verify this.

An operator or matrix satisfying (19) is called **unitary**. Note that A unitary implies $A^{-1} = A^\dagger$. So we have shown that operators that take one orthonormal basis to another are unitary. The converse is also true: all unitary operators transform orthonormal bases to other orthonormal bases. Unitary operators are the complex analogs of rotation operators in real vector spaces. Just as rotations preserve the dot products of vectors, so unitary operators preserve the inner products of kets.

Exercise 0.3 Show that unitary transformations preserve the inner product, *i.e.*, if $A|\psi\rangle = |\psi'\rangle$ and $A|\phi\rangle = |\phi'\rangle$, then $\langle\psi'|\phi'\rangle = \langle\psi|\phi\rangle$.

The unitary operators on qubits (or, equivalently, the 2×2 unitary matrices) have many nice properties and useful representations. They are summarized in section 4.2 (pp 174-177) of [NC], and will be used from time to time in later sections.

Mathematical aside: unitary groups. Consider the set of unitary 2×2 matrices. We call this set $U(2)$. It is easy to show that this set is closed under matrix multiplication, that matrix multiplication is associative, and that the matrix inverse of a unitary matrix is also unitary. We summarize these properties by saying that $U(2)$ is a **group**, the unitary group in 2 dimensions.

There is an important relation between **hermitian** (also known as **self-adjoint**) operators and unitary operators. Recall that a hermitian operator is one which satisfies

$$A^\dagger = A. \quad (20)$$

In quantum mechanics, hermitian operators are also called **observables**. Unlike unitary operators, hermitian operators do not form a group. But hermitian operators and unitary operators both satisfy the **spectral representation theorem**:¹

If A is unitary or hermitian, then it has an orthonormal basis of eigenvectors, $A|x\rangle = \lambda_x|x\rangle$ with $\langle x|y\rangle = \delta_{xy}$, and

$$A = \sum_x \lambda_x |x\rangle\langle x|. \quad (21)$$

The summation is on $x \in \{0, \dots, N-1\}$, where N is the dimension of the Hilbert space.

Though the **spectrum** — the set of eigenvalues $\{\lambda_x, x = 0, \dots, N-1\}$ — is a unique property of A , the orthonormal basis of eigenvectors is not unique. Nevertheless, the spectral representation (21) holds for any choice of orthonormal eigenbasis. Note that matrix elements of A in the orthonormal eigenbasis form a diagonal matrix with the eigenvalues as the diagonal entries.

Another important property of hermitian and unitary operators (which follows fairly easily from the spectral representation theorem) states:

Two hermitian or unitary operators A and B commute if and only if they share an orthonormal basis of eigenvectors.

This is often sloppily summarized as: “commuting operators can be simultaneously diagonalized”.

Exercise 0.4 Show that the eigenvalues of hermitian operators are real and those of unitary operators have norm 1. That is, if $A|\psi\rangle = \lambda|\psi\rangle$, then if A is hermitian show that $\lambda = \lambda^*$ and if A is unitary show that $\lambda\lambda^* = 1$.

Unitary and hermitian operators are closely related by exponentiation. In particular, U is unitary if and only if there is a hermitian matrix, A , such that $U = e^{iA}$.

Proof: If $A = A^\dagger$ then $(e^{iA})^\dagger = e^{-iA^\dagger} = e^{-iA} = (e^{iA})^{-1}$, so $U = e^{iA}$ is unitary. Conversely, a unitary U has the spectral representation $U = \sum_x e^{i\theta_x} |x\rangle\langle x|$, for some real θ_x , since the eigenvalues of U have norm 1. Define $A = \sum_x \theta_x |x\rangle\langle x|$. It is easy to check that $A = A^\dagger$ and, using the orthonormality of the $|x\rangle$ that $A^n = \sum_x (\theta_x)^n |x\rangle\langle x|$ for all positive integers n . Then $e^{iA} := \sum_{n=0}^{\infty} \frac{1}{n!} (iA)^n = \sum_n \frac{i^n}{n!} \sum_x (\theta_x)^n |x\rangle\langle x| = \sum_x \left(\sum_n \frac{i^n}{n!} (\theta_x)^n \right) |x\rangle\langle x| = \sum_x e^{i\theta_x} |x\rangle\langle x| = U$.

An important subset of hermitian operators are the **projection operators**, or **projectors** for short. They are operators satisfying

$$P = P^\dagger \quad \text{and} \quad PP = P. \quad (22)$$

Exercise 0.5 Show that the eigenvalues of a projector can only be 0 or 1.

¹In fact, this theorem holds for the larger set of **normal** operators, which are operators A which satisfy $[A, A^\dagger] = 0$. Clearly hermitian and unitary operators are subsets of the set of normal operators.

It follows from the spectral representation theorem that a projector can be written as

$$P = \sum_{x \in E_1} |x\rangle\langle x|, \quad (23)$$

where $\{|x\rangle, x = 0, \dots, N-1\}$ is an orthonormal eigenbasis of P and $E_1 \subset \{0, \dots, N-1\}$ is the subset for which $|x\rangle$ are eigenvectors with eigenvalue 1. The set $\{|x\rangle, x \in E_1\}$ is an orthonormal basis for a **subspace** of \mathcal{H}_N , called the **eigenspace** of eigenvalue 1, and denoted $V_1 \subset \mathcal{H}_N$. Denote by E_0 the complement of E_1 , *i.e.*, $E_0 := \{x | x \notin E_1\}$. Then $\{|x\rangle, x \in E_0\}$ is an orthonormal basis for another subspace $V_0 \subset \mathcal{H}_N$ which is the eigenspace of P of eigenvalue 0.

Exercise 0.6 By the definition of the subspace V_1 , a general vector $|\psi\rangle \in V_1$ can be written as $|\psi\rangle = \sum_{x \in E_1} \psi_x |x\rangle$ for some complex numbers ψ_x . Show that $|\psi\rangle$ is an eigenvector of P with eigenvalue 1.

Exercise 0.7 For any ket $|\psi\rangle \in \mathcal{H}_N$ show that $P|\psi\rangle \in V_1$, and if $|\psi\rangle \in V_1$, show that $P|\psi\rangle = |\psi\rangle$. Thus P “projects” vectors onto the subspace V_1 .

Exercise 0.8 If P is a projector, show that $Q := 1 - P$ is also a projector, that $QP = PQ = 0$, and that Q projects vectors onto the subspace V_0 .

Exercise 0.9 Show that V_0 and V_1 are orthogonal subspaces, *i.e.*, if $|\psi\rangle \in V_0$ and $|\phi\rangle \in V_1$, then $\langle\psi|\phi\rangle = 0$.

In general, for any diagonalizable operator A (*e.g.*, any hermitian or unitary operator), with distinct eigenvalues $\{\lambda_q\}$ and orthonormal eigenbasis $\{|x\rangle, x = 0, \dots, N-1\}$, we define the sets $E_{\lambda_q} \subset \{0, \dots, N-1\}$ as the subset for which $|x\rangle$ with $x \in E_{\lambda_q}$ are eigenvectors with eigenvalue λ_q . We call $d_{\lambda_q} := |E_{\lambda_q}|$ (*i.e.*, the number of elements in the set E_{λ_q}) the **degeneracy** of the eigenvalue λ_q . The corresponding eigenspace of eigenvalue λ_q is denoted $V_{\lambda_q} \subset \mathcal{H}_N$. Thus the dimension of V_{λ_q} is the degeneracy of λ_q . The spectral representation of A is then

$$A = \sum_q \lambda_q \left(\sum_{x \in E_{\lambda_q}} |x\rangle\langle x| \right) = \sum_q \lambda_q P_{\lambda_q} \quad (24)$$

where we have defined the **projector onto the eigenspace** V_{λ_q} by

$$P_{\lambda_q} := \sum_{i \in E_{\lambda_q}} |i\rangle\langle i|. \quad (25)$$

Exercise 0.10 Show that $[P_{\lambda_p}, P_{\lambda_q}] = 0$ for all p, q , and that $\sum_q P_{\lambda_q} = 1$.

We say that the P_{λ_q} form a complete orthogonal set of projectors.

Exercise 0.11 Show that V_{λ_p} is orthogonal to V_{λ_q} for all $p \neq q$.

The eigenspaces provide an **orthogonal decomposition** of \mathcal{H}_N , which we write as follows. If there are $M \leq N$ distinct eigenvalues, then

$$\mathcal{H}_N = V_{\lambda_1} \oplus V_{\lambda_2} \oplus \cdots \oplus V_{\lambda_M}, \quad (26)$$

where the \oplus denotes the **direct sum** of vector spaces. Note that vector space dimensions add under direct sum, so $N = \sum_{p=1}^M d_{\lambda_p}$.

Exercise 0.12 Consider a 3-dimensional Hilbert space, \mathcal{H}_3 , with orthonormal basis $\{|j\rangle, j = 0, 1, 2\}$ with respect to which the hermitian operator A has matrix elements

$$A = \begin{pmatrix} 17 & -7(1-i)\sqrt{2} & 7i\sqrt{3} \\ -7(1+i)\sqrt{2} & -4 & -7(1-i)\sqrt{6} \\ -7i\sqrt{3} & -7(1+i)\sqrt{6} & 3 \end{pmatrix}.$$

Find the eigenvalues, λ_a , and an orthonormal basis of eigenvectors, $\{|j'\rangle, j = 1, 2, 3\}$, of A . What are the degeneracies of the eigenvalues? Write the matrix elements of the projectors onto the eigenspaces, P_{λ_a} , in the $\{|j\rangle\}$ basis, and verify the spectral representation (24) of A .

Physical interpretation of vectors and operators in quantum mechanics

All the above linear algebra is important because, in a nutshell, vectors in Hilbert spaces encode the physical state of a quantum system, hermitian operators encode the possible experimental observations on those systems, projectors encode the possible outcomes of those experiments, and unitary operators encode the dynamical evolution of systems which are isolated from outside influences (like experimental observation). We will now state this dictionary between quantum physics and linear algebra more precisely. These are often called the **axioms of quantum mechanics**.

- I. The **physical state** of a system is a vector, $|\psi\rangle$, in a Hilbert space which is normalized: $\langle\psi|\psi\rangle = 1$. The overall phase of a state vector is physically unobservable. That is, two normalized vectors $|\phi\rangle$ and $|\psi\rangle$ satisfying $|\phi\rangle = e^{i\theta}|\psi\rangle$ for any real θ describe the same physical state.
- II. An **observable** (or, physically allowed measurement) is a choice of a hermitian operator, M . By the spectral theorem, $M = \sum_a \mu_a P_{\mu_a}$, where μ_a are its eigenvalues and P_{μ_a} are the orthogonal projection operators onto their corresponding eigenspaces.
- III. The only possible **outcomes** of measuring M are one of its eigenvalues. I denote this outcome of this measurement by “ $M = \mu_a$ ”.
- IV. The **probability** of observing a given possible outcome, $M = \mu_a$, of such a measurement is the norm-squared of the projection of the state onto the eigenspace of the eigenvalue μ_a . In formulas, this is

$$\mathcal{P}(M=\mu_a) = \langle\psi|P_{\mu_a}|\psi\rangle = \|P_{\mu_a}|\psi\rangle\|^2. \quad (27)$$

V. Once we observe the outcome $M = \mu_a$, the **state changes as a result of the measurement** from its state $|\psi\rangle$ immediately before the measurement, to a new state $|\psi'\rangle$ immediately after the measurement, given by its normalized projection onto the eigenspace corresponding to the observed eigenvalue. In formulas, this is

$$|\psi\rangle \xrightarrow{\text{meas. } M=\mu_a} |\psi'\rangle = \frac{P_{\mu_a}|\psi\rangle}{\sqrt{\mathcal{P}(M=\mu_a)}} = \frac{P_{\mu_a}|\psi\rangle}{\|P_{\mu_a}|\psi\rangle\|}. \quad (28)$$

VI. The **dynamical evolution** (*i.e.*, change with time) of the state of an isolated system (*i.e.*, when it is not being measured or otherwise interacting with the external world) from an initial state $|\psi\rangle$ to a later state $|\psi'\rangle$ is given by

$$|\psi'\rangle = U|\psi\rangle, \quad (29)$$

where U is a unitary operator that depends on the physical system in question.

Note that the **expectation value** of an observable M on a state $|\psi\rangle$,

$$\langle M \rangle = \langle \psi | M | \psi \rangle, \quad (30)$$

does *not* give the value of a measurement of M ! Rather it gives the **average value** of measurements of M on many identical systems all prepared in the state $|\psi\rangle$. We see this as follows. By definition, the expectation value of a measurement of M is the average of its possible outcomes weighted by their probabilities. Thus, $\langle M \rangle = \sum_a \mu_a \mathcal{P}(M=\mu_a) = \sum_a \mu_a \langle \psi | P_{\mu_a} | \psi \rangle = \langle \psi | (\sum_a \mu_a P_{\mu_a}) | \psi \rangle = \langle \psi | M | \psi \rangle$, proving (30), where we used axioms II-IV.

There is one more rule of quantum mechanics, which governs how larger systems are built up from smaller ones:

VII. The Hilbert space, \mathcal{H} , describing the states of a system which is made up of two **subsystems** which are described by vectors in Hilbert spaces \mathcal{H}_1 and \mathcal{H}_2 is given by the **tensor product**

$$\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2. \quad (31)$$

We will discuss tensor products of Hilbert spaces in more detail below.

0.2 Single qubit gates and Pauli matrices

Talking about the time evolution of a quantum system in terms of “gates” (instead of in terms of a differential equation in time, like the Schrodinger equation) is just a way to simplify the problem: we break the system into small pieces, the qubits, and their interaction into localized pieces which we call gates, and the time evolution into discrete steps. By doing this we effectively map general quantum dynamics onto a quantum analog of a classical computer: qubits replace bits, gates replace gates, and the discretized time step is the analog of the “clock rate” of a classical computer. Thus we can think of “quantum computers” and “quantum algorithms” (*i.e.*, an arrangement of qubits and gates together with a set of

measurements) as a kind of language for describing and exploring general kinds of quantum mechanical evolution.

We have seen that a single qubit is a normalized vector $|\psi\rangle \in \mathcal{H}_2$ which can be written

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle = \sum_{a \in F_2} \alpha_a |a\rangle, \quad \alpha_a \in \mathbb{C}, \quad \sum_{a \in F_2} |\alpha_a|^2 = 1, \quad (32)$$

where $\{|a\rangle, a \in F_2\}$ is some chosen orthonormal basis of \mathcal{H}_2 . Here we have introduced a shorthand notation

$$F_2 \equiv \{0, 1\}, \quad (33)$$

the set with the two elements 0 and 1. (The name comes from the fact that this set of integers is, mathematically, the *field* with two elements, ie, it is closed under addition and multiplication mod 2 and their inverses.) With respect to this basis, called the **computational basis**, we can also represent this qubit as the column matrix

$$|\psi\rangle \leftrightarrow \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix}. \quad (34)$$

In particular, we have $|0\rangle \leftrightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle \leftrightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

A single qubit quantum gate is a unitary operator

$$U : \mathcal{H}_2 \rightarrow \mathcal{H}_2. \quad (35)$$

The action of the unitary operator U on a qubit tells how the qubit changes during a given time interval. The operator U is completely determined by its matrix elements $U_{ab} = \langle a|U|b\rangle$ in an orthonormal basis. This is because its action on any qubit is given by

$$\begin{aligned} U|\psi\rangle &= 1 \cdot U(\alpha|0\rangle + \beta|1\rangle) = \left(|0\rangle\langle 0| + |1\rangle\langle 1|\right)(\alpha U|0\rangle + \beta U|1\rangle) \\ &= (U_{00}\alpha + U_{01}\beta)|0\rangle + (U_{10}\alpha + U_{11}\beta)|1\rangle. \end{aligned} \quad (36)$$

In matrix notation this becomes

$$U|\psi\rangle \leftrightarrow \begin{pmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix}. \quad (37)$$


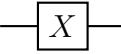
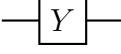
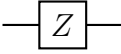
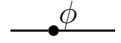
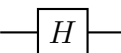
We can also characterize U without referring to a state which it acts on:

$$U = 1 \cdot U \cdot 1 = \left(\sum_a |a\rangle\langle a|\right) U \left(\sum_b |b\rangle\langle b|\right) = \sum_{ab} U_{ab} |a\rangle\langle b| \quad \leftrightarrow \quad \begin{pmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{pmatrix}. \quad (38)$$

In particular,

$$|0\rangle\langle 0| \leftrightarrow \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad |0\rangle\langle 1| \leftrightarrow \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad |1\rangle\langle 0| \leftrightarrow \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}, \quad |1\rangle\langle 1| \leftrightarrow \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}. \quad (39)$$

We now list some of the commonly-used single qubit gates.

Name	symbol	operator	matrix
Identity		$I = 0\rangle\langle 0 + 1\rangle\langle 1 $	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
Pauli-X		$X = 1\rangle\langle 0 + 0\rangle\langle 1 $	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
Pauli-Y		$Y = i 1\rangle\langle 0 - i 0\rangle\langle 1 $	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$
Pauli-Z		$Z = 0\rangle\langle 0 - 1\rangle\langle 1 $	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
Phase		$\phi = 0\rangle\langle 0 + e^{i\phi} 1\rangle\langle 1 $	$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$
Hadamard		$H = \frac{1}{\sqrt{2}} \sum_{a,b \in F_2} (-)^{ab} a\rangle\langle b $	$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

You should be able to easily verify that they are all unitary.

The first four gates, I , X , Y , and Z , are especially useful. Their matrix forms are the identity matrix and the three Pauli matrices, often denoted σ^j , $j = 1, 2, 3$ in the physics literature. It is easy to see that they form a real basis of 2×2 hermitian matrices. That is, any 2×2 hermitian matrix can be written uniquely as a real linear combination of these four matrices. Moreover, these are also unitary operators. Unlike hermitian operators, linear combinations of unitary operators are *not* unitary, in general. Although any unitary operator can be realized as an exponential of a hermitian operator, this is not very useful for building a quantum circuit out of Pauli gates since exponentiation involves an infinite sum of terms. There are a raft of useful Pauli matrix identities — they are always easy to check in the matrix representation. The most basic ones, which are worth memorizing, are

$$\begin{aligned}
1 &= X^2 = Y^2 = Z^2, \\
XY &= -YX, \quad YZ = -ZY, \quad ZX = -XZ, \\
XY &= iZ, \quad YZ = iX, \quad ZX = iY.
\end{aligned} \tag{40}$$

Exercise 0.13 Verify these identities.

The Hadamard gate will also play a prominent role in what follows. Some useful identities which it satisfies are

$$\begin{aligned}
1 &= H^2, \\
H|a\rangle &= \frac{1}{\sqrt{2}} (|0\rangle + (-)^a|1\rangle) \quad \text{for } a \in F_2, \\
H &= \frac{1}{\sqrt{2}}(X + Z), \\
X &= HZH, \quad Z = HXH.
\end{aligned} \tag{41}$$

Exercise 0.14 Verify these identities.

Although we have not exhaustively treated single qubit gates, we have enough to do some interesting things. We now expand our vision to deal with multiple qubit gates.

Exercise 0.15 Show that the following sequence of gates,

$$|0\rangle \xrightarrow{H} \bullet^{2\theta} \xrightarrow{H} \bullet^{\phi+(\pi/2)} \xrightarrow{X} \bullet^{-\theta} \xrightarrow{X} \bullet^{-\theta} \xrightarrow{\quad} |\psi\rangle \quad (42)$$

acting on input $|0\rangle$ gives $|\psi\rangle = \cos\theta|0\rangle + e^{i\phi}\sin\theta|1\rangle$. So this sequence of gates takes us from $|0\rangle$ to a general point on the Bloch sphere.

0.3 Tensor products

The tensor product operation on Hilbert spaces is how we put subsystems together to form larger systems in quantum mechanics. In particular, a quantum computer using n qubits will act a Hilbert space which is the n -fold tensor product of single-qubit Hilbert spaces. For example, a circuit with two qubits of input is, schematically,

$$\begin{array}{c} |\psi\rangle \\ |\phi\rangle \end{array} \xrightarrow{\text{gates}} \begin{array}{c} \text{---} \\ \text{---} \end{array} \quad (43)$$

We write the input state of this circuit as

$$|\psi\rangle \otimes |\phi\rangle \equiv |\psi\rangle|\phi\rangle \equiv |\psi, \phi\rangle. \quad (44)$$

These are three different notations for the same thing, the tensor product of the states. Note that the tensor product is *not* commutative:

$$|\phi\rangle|\psi\rangle \neq |\psi\rangle|\phi\rangle. \quad (45)$$

The order matters! A generally followed convention is that the order of factors in the tensor product from left to right is the same as the order of the qubit lines in a circuit diagram from top to bottom.

Each of the input states is in its own private \mathcal{H}_2 . The space of all 2-qubit states is written

$$\mathcal{H}_2 \otimes \mathcal{H}_2. \quad (46)$$

One important mathematical condition that we impose on this tensor product is that it is bilinear, *i.e.*, it is linear in each of the two qubits separately. In formulas, this says that

$$\begin{aligned} (|\psi\rangle + |\xi\rangle)|\phi\rangle &= |\psi\rangle|\phi\rangle + |\xi\rangle|\phi\rangle, \\ |\psi\rangle(|\phi\rangle + |\xi\rangle) &= |\psi\rangle|\phi\rangle + |\psi\rangle|\xi\rangle, \\ \alpha(|\psi\rangle|\phi\rangle) &= (\alpha|\psi\rangle)|\phi\rangle = |\psi\rangle(\alpha|\phi\rangle). \end{aligned} \quad (47)$$

It follows from this that $\mathcal{H}_2 \otimes \mathcal{H}_2$ is itself a vector space. In particular, if $\{|L_a\rangle, a \in F_2\}$ is a basis of the left \mathcal{H}_2 factor and $\{|R_b\rangle, b \in F_2\}$ is a basis of the right \mathcal{H}_2 factor, then $\{|L_a\rangle \otimes |R_b\rangle \equiv |L_a R_b\rangle, a, b \in F_2\}$ is a basis of $\mathcal{H}_2 \otimes \mathcal{H}_2$. In particular, if we take the computational bases for each \mathcal{H}_2 factor, then the computational basis for $\mathcal{H}_2 \otimes \mathcal{H}_2$ is $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$.

This can easily be extended to an arbitrary number of tensor product factors. The tensor product of n qubits, $\mathcal{H}_2^{\otimes n} = \mathcal{H}_2 \otimes \cdots \otimes \mathcal{H}_2$ n times, has computational basis $\{|a_1 a_2 \cdots a_n\rangle, a_j \in F_2\}$. (Recall the definition (33) of F_2 .) So we see that the dimension of $\mathcal{H}_2^{\otimes n}$ is 2^n . In general, dimensions multiply under tensor products.

Tensor products also inherit an inner product from their factors. In particular, if $|a_1 \cdots a_n\rangle \in \mathcal{H}_2^{\otimes n}$ and $|b_1 \cdots b_n\rangle \in \mathcal{H}_2^{\otimes n}$, then we define

$$\langle a_1 \cdots a_n | b_1 \cdots b_n \rangle = \langle a_1 | b_1 \rangle \cdots \langle a_n | b_n \rangle, \quad (48)$$

and extend this to all states of $\mathcal{H}_2^{\otimes n}$ by linearity.

Exercise 0.16 Show that the computational basis of $\mathcal{H}_2^{\otimes n}$ is an orthonormal basis, ie,

$$\langle a_1 \cdots a_n | b_1 \cdots b_n \rangle = \delta_{a_1 b_1} \cdots \delta_{a_n b_n}. \quad (49)$$

The computation basis vectors, $|a_1 a_2 \cdots a_n\rangle$, are unwieldy to write when n is large. There is a clever shorthand: each basis state is labelled by the sequence of n 0's or 1's — $a_1 a_2 \cdots a_n$ — which can be thought of as an integer in binary notation running from $00 \cdots 0$ up to $11 \cdots 1$. In other words, they run over the integers $x \in \{0, 1, 2, \dots, 2^n - 1\}$. Thus we often will notate the computational basis of $\mathcal{H}_2^{\otimes n}$ by the set $\{|x\rangle, x = 0, \dots, 2^n - 1\}$. For example, when $n = 3$, we write

$$\begin{aligned} |000\rangle &= |0\rangle \\ |001\rangle &= |1\rangle \\ |010\rangle &= |2\rangle \\ |011\rangle &= |3\rangle \\ |100\rangle &= |4\rangle \\ |101\rangle &= |5\rangle \\ |110\rangle &= |6\rangle \\ |111\rangle &= |7\rangle. \end{aligned}$$

Note that in this notation, the orthonormality condition (49) becomes simply $\langle x | y \rangle = \delta_{x,y}$.

Using this notation we can write the general state in $\mathcal{H}_2^{\otimes n}$ as a linear combination of its basis vectors,

$$|\psi\rangle = \sum_{x=0}^{2^n-1} \psi_x |x\rangle, \quad (50)$$

for some complex numbers ψ_x . For $|\psi\rangle$ to be normalized, we must have $\sum_x |\psi_x|^2 = 1$.

Operators on tensor product spaces can be built up from operators on each factor in a similar fashion. For example, if

$$A : \mathcal{H}_2 \rightarrow \mathcal{H}_2, \quad \text{and} \quad B : \mathcal{H}_2 \rightarrow \mathcal{H}_2 \quad (51)$$

are any linear operators acting on a qubit, then a new linear operator, $A \otimes B$ acting on $\mathcal{H}_2 \otimes \mathcal{H}_2$, is defined by the rule

$$\begin{aligned} (A \otimes B)|\psi\rangle \otimes |\phi\rangle &= (A \otimes B)|\psi, \phi\rangle \\ &:= (A|\psi\rangle) \otimes (B|\phi\rangle), \end{aligned} \quad (52)$$

for any states $|\phi\rangle, |\psi\rangle \in \mathcal{H}_2$, and extended by linearity to all states of $\mathcal{H}_2 \otimes \mathcal{H}_2$.

Exercise 0.17 Show that

$$\begin{aligned} (1 \otimes A)|\psi, \phi\rangle &= |\psi\rangle \otimes A|\phi\rangle, \\ (A \otimes 1)|\psi, \phi\rangle &= A|\psi\rangle \otimes |\phi\rangle, \\ (0 \otimes A)|\psi, \phi\rangle &= 0. \end{aligned} \quad (53)$$

Exercise 0.18 Show that if A and B are unitary, that $A \otimes B$ is unitary. Show that if A and B are hermitian, that $A \otimes B$ is hermitian.

In the computational basis, an operator A is given by a 2×2 matrix $A_{ab} = \langle a|A|b\rangle$ with $a, b \in F_2$, and similarly for B ,

$$A \leftrightarrow \begin{pmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{pmatrix}, \quad B \leftrightarrow \begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix}. \quad (54)$$

With respect to the $|ij\rangle$ computational basis of $\mathcal{H}_2 \otimes \mathcal{H}_2$, the tensor product operator $A \otimes B$ is a 4×4 matrix with matrix elements

$$(A \otimes B)_{ab,cd} = \langle a, b|A \otimes B|c, d\rangle = \langle a|A|c\rangle \langle b|B|d\rangle = A_{ac}B_{bd}. \quad (55)$$

So, as a matrix, we have

$$A \otimes B \leftrightarrow \begin{pmatrix} A_{00}B_{00} & A_{00}B_{01} & A_{01}B_{00} & A_{01}B_{01} \\ A_{00}B_{10} & A_{00}B_{11} & A_{01}B_{10} & A_{01}B_{11} \\ A_{10}B_{00} & A_{10}B_{01} & A_{11}B_{00} & A_{11}B_{01} \\ A_{10}B_{10} & A_{10}B_{11} & A_{11}B_{10} & A_{11}B_{11} \end{pmatrix}, \quad (56)$$

which can also be written in 2×2 block form as

$$A \otimes B \leftrightarrow \begin{pmatrix} A_{00}B & A_{01}B \\ A_{10}B & A_{11}B \end{pmatrix}, \quad \text{where} \quad B = \begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix}. \quad (57)$$

Exercise 0.19 Show that

$$X \otimes Y = \begin{pmatrix} 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \\ 0 & -i & 0 & 0 \\ i & 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad Y \otimes X = \begin{pmatrix} 0 & 0 & 0 & -i \\ 0 & 0 & -i & 0 \\ 0 & i & 0 & 0 \\ i & 0 & 0 & 0 \end{pmatrix}. \quad (58)$$

Note that this shows explicitly that $X \otimes Y \neq Y \otimes X$.

This definition and examples are easily generalized from 2 to an arbitrary number, n , qubits. But since the dimension of $\mathcal{H}_2^{\otimes n}$ increases exponentially with n , writing the matrix expressions for operators on tensor product spaces quickly becomes useless.

0.4 Entanglement

The computational basis elements of a tensor product space, like $\mathcal{H}_2^{\otimes n}$, are themselves tensor products of individual vectors in each factor, $|a_1 a_2 \cdots a_n\rangle = |a_1\rangle \otimes |a_2\rangle \otimes \cdots \otimes |a_n\rangle$. The general state in $\mathcal{H}_2^{\otimes n}$ is a linear combination of these basis states. Can the general state also be factorized into a tensor product of states in each factor? The answer is no.

We'll prove this in the case of $\mathcal{H}_2 \otimes \mathcal{H}_2$. First, let's look at a typical example. Consider the state

$$|\beta_{00}\rangle := \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle), \quad (59)$$

often called the **EPR state**. It and related states will play a prominent role in future sections. We ask: Do there exist states $|\phi\rangle, |\psi\rangle \in \mathcal{H}_2$ such that $|\beta_{00}\rangle = |\phi\rangle|\psi\rangle$? The most general possible form of $|\phi\rangle$ and $|\psi\rangle$ are

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad |\psi\rangle = \gamma|0\rangle + \delta|1\rangle, \quad (60)$$

for some complex numbers $\alpha, \beta, \gamma, \delta$. Then

$$\begin{aligned} |\phi\rangle|\psi\rangle &= (\alpha|0\rangle + \beta|1\rangle)(\gamma|0\rangle + \delta|1\rangle) \\ &= \alpha\gamma|00\rangle + \alpha\delta|01\rangle + \beta\gamma|10\rangle + \beta\delta|11\rangle. \end{aligned} \quad (61)$$

Comparing to (59), we see that $|\beta_{00}\rangle = |\phi\rangle|\psi\rangle$ if and only if there is a solution to the equations

$$\alpha\gamma = \beta\delta = 1, \quad \text{and} \quad \alpha\delta = \beta\gamma = 0. \quad (62)$$

But the first equalities imply that none of $\alpha, \beta, \gamma, \delta$ can vanish, while the second equalities imply that at least two of them do vanish. This is a contradiction, so no solution to these equations exist, and the EPR state cannot be factorized.

A multi-qubit state which cannot be factorized into a product of states of each qubit is called an **entangled state**. This property of entanglement has no analog in classical mechanics. It should be thought of as a “new” feature of quantum mechanics, or as a resource that quantum computation can take advantage of that classical computers don't have access to.

We will now show that almost every state in $\mathcal{H}_2 \otimes \mathcal{H}_2$ is entangled. The most general state in $\mathcal{H}_2 \otimes \mathcal{H}_2$ can be written as

$$|\chi\rangle = \zeta_{00}|00\rangle + \zeta_{01}|01\rangle + \zeta_{10}|10\rangle + \zeta_{11}|11\rangle, \quad (63)$$

for some complex numbers ζ_{ij} . The condition that $|\chi\rangle$ is unentangled, ie, that $|\chi\rangle = |\phi\rangle|\psi\rangle$ with $|\phi\rangle$ and $|\psi\rangle$ some 1-qubit states as in (60) is that (63) equals (61), or,

$$\zeta_{00} = \alpha\gamma, \quad \zeta_{01} = \alpha\delta, \quad \zeta_{10} = \beta\gamma, \quad \zeta_{11} = \beta\delta. \quad (64)$$

This can be rewritten as the matrix equation

$$\begin{pmatrix} \zeta_{00} & \zeta_{01} \\ \zeta_{10} & \zeta_{11} \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \begin{pmatrix} \gamma & \delta \end{pmatrix}. \quad (65)$$

Note that the determinant of the right hand side always vanishes. (In fact, the vanishing of this determinant is also a sufficient condition for it being written in this factorized form.) Thus the condition for $|\chi\rangle$ to factorize is that $\zeta_{00}\zeta_{11} - \zeta_{01}\zeta_{10} = 0$. This condition is therefore satisfied by only a 3-complex-dimensional subset of the 4-complex-dimensional space of $\mathcal{H}_2 \otimes \mathcal{H}_2$ states. Thus almost every state in $\mathcal{H}_2 \otimes \mathcal{H}_2$ is entangled.

This can be easily generalized to arbitrary tensor products. There are various quantitative measures of *how* entangled a given state is, ie, how “far” it is from being a factorized state. These measures are generally called **entanglement entropy**. We will not pursue the development of this subject further in this course, but you should be aware that the study of entanglement entropies of various sorts and in various systems is a very large part of current research on quantum information.

Part I

Quantum Circuits

You have had a basic introduction to quantum mechanics, have studied the single qubit in some detail, and have worked with several of the operators that are important in quantum mechanics. Our next goal is to address some of the simple quantum circuits constructed from quantum gates, and then from there, to construct *quantum algorithms*. Quantum algorithms are combinations of quantum circuits with measurements and classical computations to solve certain problems. Even some of the simplest quantum algorithms use many qubits, and so can be complicated to analyze and describe.

1 Simple circuits

1.1 The no-cloning theorem, controlled-NOT and related gates

The no-cloning theorem is a basic no-go statement about possible quantum circuits. In particular, it states that there is no possible quantum circuit that can copy a qubit (or any multi-qubit quantum state, for that matter). By “copy” we mean the following. A quantum copy machine is a circuit that behaves like

$$\begin{array}{ccc} |\psi\rangle & \text{---} & \boxed{U} & \text{---} & |\psi\rangle \\ |0\rangle & \text{---} & & & |\psi\rangle \end{array} \quad (66)$$

That is, it takes a state $|\psi\rangle|0\rangle$ as input and outputs $|\psi\rangle|\psi\rangle$, where $|\psi\rangle$ is *any* state (in a given Hilbert space) and $|0\rangle$ is some fixed reference state.

We now show that this is impossible — ie, that no unitary U exists with this property — by supposing such a unitary U does exist, and finding a contradiction. If U exists, then $U|\psi, 0\rangle = |\psi, \psi\rangle$ and $U|\phi, 0\rangle = |\phi, \phi\rangle$ for any two states $|\psi\rangle$ and $|\phi\rangle$. Take the hermitian conjugate of the second equation and then take the inner product of the two equations to get

$$\langle\phi, 0|U^\dagger U|\psi, 0\rangle = \langle\phi, \phi|\psi, \psi\rangle. \quad (67)$$

The left side is

$$\langle\phi, 0|U^\dagger U|\psi, 0\rangle = \langle\phi, 0|\psi, 0\rangle\langle\phi|\psi\rangle\langle 0|0\rangle = \langle\phi|\psi\rangle, \quad (68)$$

where in the first step we used that U is unitary, and in the last step we used that $|0\rangle$ is normalized. The right side of (67) is

$$\langle\phi, \phi|\psi, \psi\rangle = \langle\phi|\psi\rangle\langle\phi|\psi\rangle = \langle\phi|\psi\rangle^2. \quad (69)$$

Equating the left and right sides we then see that $\langle\phi|\psi\rangle = \langle\phi|\psi\rangle^2$, which only has the solutions

$$\langle\phi|\psi\rangle = 0 \text{ or } 1. \quad (70)$$

But we supposed $|\phi\rangle$ and $|\psi\rangle$ to be *any* two states, so, in particular, $\langle\phi|\psi\rangle$ can be any complex number with norm less than or equal to 1 (since $|\phi\rangle$ and $|\psi\rangle$, as states, are normalized). This contradiction with (70) proves the no-cloning theorem.

Notice that the proof of the theorem does not prevent us from cloning an *orthonormal basis* of Hilbert space. That is, since all states in an orthonormal basis satisfy (70) (by definition), it is perfectly possible to construct a unitary U which clones only these states. Let us demonstrate this for two qubits by defining the **controlled-NOT gate**, which we'll denote " CN ", and whose circuit diagram is

$$CN = \begin{array}{c} \text{---} \bullet \text{---} \\ | \\ \text{---} \oplus \text{---} \end{array} . \quad (71)$$

We can define it by its action on the computational basis,

$$CN|00\rangle = |00\rangle, \quad CN|01\rangle = |01\rangle, \quad CN|10\rangle = |11\rangle, \quad CN|11\rangle = |10\rangle. \quad (72)$$

Note that it does nothing to the first qubit, but if the first qubit is $|1\rangle$ it flips the second qubit. This action on the computational basis can be neatly described diagrammatically as

$$\begin{array}{c} |a\rangle \text{---} \bullet \text{---} |a\rangle \\ | \\ |b\rangle \text{---} \oplus \text{---} |b \oplus a\rangle \end{array} \quad a, b \in F_2. \quad (73)$$

Here we use the \oplus sign to denote addition in F_2 , ie, addition modulo 2:

$$0 \oplus 0 := 0, \quad 0 \oplus 1 := 1, \quad 1 \oplus 0 := 1, \quad 1 \oplus 1 := 0. \quad (74)$$

We say that the flip of the second qubit is controlled by the first qubit. If we have N qubits, labelled by $i = 1, \dots, N$, we might specify a controlled-NOT gate as $C_i N_j$ to mean the flip of the j th qubit is controlled by the i th qubit. Thus, for example,

$$C_1 N_2 = \begin{array}{c} \text{---} \bullet \text{---} \\ | \\ \text{---} \oplus \text{---} \end{array} \quad \text{and} \quad C_2 N_1 = \begin{array}{c} \text{---} \oplus \text{---} \\ | \\ \text{---} \bullet \text{---} \end{array} . \quad (75)$$

Exercise 1.1 Write CN as a 4×4 matrix in the computational basis and verify that it is unitary.

Writing n -qubit gates out as $2^n \times 2^n$ matrices — even for n as small as 2 — is mind-numbing torture. An often better way is to write them as operators. Noting that $|a\rangle\langle b|$ for a, b labelling orthonormal basis vectors is an operator which maps the basis vector $|b\rangle$ to $|a\rangle$, and maps all other basis vectors to zero, it is immediate from (72) that

$$CN = |00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 11| + |11\rangle\langle 10|. \quad (76)$$

Using the hint from (73), this can be written more compactly as

$$CN = \sum_{a,b \in F_2} |a, b \oplus a\rangle \langle a, b|. \quad (77)$$

You should become familiar with these manipulations so that going between the equivalent definitions (72), (73), (76), and (77), becomes an easy exercise.

Now compare the effect of the controlled-NOT gate with input $|a, 0\rangle$,

$$\begin{array}{c} |a\rangle \text{---} \bullet \text{---} |a\rangle \\ |0\rangle \text{---} \oplus \text{---} |a\rangle \end{array} \quad a \in F_2. \quad (78)$$

to that of a putative quantum copier, (66). The difference is that the controlled-NOT gate only has this “copying” output when $|a\rangle$ is a basis state.

Exercise 1.2 Compute the output state of the controlled-NOT gate if the input state is $|\psi, 0\rangle$ where $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ is a general 1-qubit state.

Exercise 1.3 Compute the output state of the controlled-NOT gate if the input state is $|\psi, \phi\rangle$ where $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ and $|\phi\rangle = \gamma|0\rangle + \delta|1\rangle$ are general 1-qubit states.

Some “fun” with controlled-NOT gates. The controlled-NOT gate satisfies a few simple and useful identities.

Exercise 1.4 Show that $(CN)^2 = 1$. That is, show that

$$\begin{array}{c} \bullet \quad \bullet \\ | \quad | \\ \oplus \quad \oplus \end{array} = \text{---} \quad (79)$$

Exercise 1.5 Show that

$$\begin{aligned} C_1 N_2 &= \frac{1}{2}(1 \otimes 1 + Z \otimes 1 + 1 \otimes X - Z \otimes X), \\ C_2 N_1 &= \frac{1}{2}(1 \otimes 1 + 1 \otimes Z + X \otimes 1 - X \otimes Z). \end{aligned} \quad (80)$$

The following combination has its own symbol,

$$\begin{array}{c} \bullet \quad \oplus \quad \bullet \\ | \quad | \quad | \\ \oplus \quad \bullet \quad \oplus \end{array} = \begin{array}{c} \times \\ | \\ \times \end{array}, \quad (81)$$

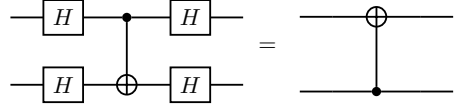
and is called the **swap gate**, and we’ll denote it by the unitary operator U_{swap} . Let us compute its effect on the computational basis.

$$\begin{aligned} U_{\text{swap}}|a, b\rangle &= C_1 N_2 \cdot C_2 N_1 \cdot C_1 N_2 |a, b\rangle \\ &= C_1 N_2 \cdot C_2 N_1 |a, b \oplus a\rangle \\ &= C_1 N_2 |a \oplus b \oplus a, b \oplus a\rangle = C_1 N_2 |b, b \oplus a\rangle \\ &= |b, b \oplus a \oplus b\rangle = |b, a\rangle, \end{aligned} \quad (82)$$

so it swaps the first and second qubits, at least if they are basis states.

Exercise 1.6 Show that $U_{\text{swap}}|\phi, \psi\rangle = |\psi, \phi\rangle$, so the swap gate in fact swaps all tensor product (unentangled) 2-qubit states.

Finally, there is a nice relationship between the C_1N_2 , C_2N_1 , and the Hadamard gates:


(83)

As an operator equation this is

$$(H \otimes H)C_1N_2(H \otimes H) = C_2N_1. \quad (84)$$

We prove this by using the identities (80). Then the left side becomes

$$\begin{aligned} (H \otimes H)C_1N_2(H \otimes H) &= \frac{1}{2}(H \otimes H)(1 \otimes 1 + Z \otimes 1 + 1 \otimes X - Z \otimes X)(H \otimes H) \\ &= \frac{1}{2}(H \otimes H + (HZ) \otimes H + H \otimes (HX) - (HZ) \otimes (HX))(H \otimes H) \\ &= \frac{1}{2}(H^2 \otimes H^2 + (HZH) \otimes H^2 + H^2 \otimes (HXH) - (HZH) \otimes (HXH)) \\ &= \frac{1}{2}(1 \otimes 1 + X \otimes 1 + 1 \otimes Z - X \otimes Z) \\ &= C_2N_1, \end{aligned} \quad (85)$$

where in the second-to-last line we used the identities $H^2 = 1$, $HZH = X$, and $HXH = Z$ from (41), and in the last line we used (80).

Other controlled gates. The controlled-NOT gate can be generalized to any m -qubit gate, U , to give an $(m+1)$ -qubit **controlled-U** gate, denoted by the operator CU , and the circuit diagram


(86)

(Here the $\text{---}/^m$ symbol is a shorthand for m qubits.) The idea is that CN does nothing to the last (bottom) m qubits if the first (control, or top) qubit is in the computational basis state $|0\rangle$, while it performs the U operation on the last m qubits if the control qubit is $|1\rangle$. This can be summarized by the equation

$$CU = (|0\rangle\langle 0|) \otimes 1 + (|1\rangle\langle 1|) \otimes U, \quad (87)$$

where the second factor in the tensor product refers to the m -qubit Hilbert space $\mathcal{H}_2^{\otimes m}$. Thus, the “1” in the first term is the identity operator on $\mathcal{H}_2^{\otimes m}$, and the “ U ” in the second term is the unitary operator U acting on $\mathcal{H}_2^{\otimes m}$.

Exercise 1.7 Show that CU is unitary if U is unitary.

Exercise 1.8 Show that $CN = CX$, ie, that the controlled-NOT gate is the same as the controlled- X gate.

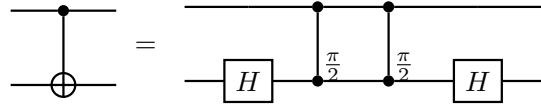
Exercise 1.9 Show that the controlled-phase gate, $C\phi$,


(88)

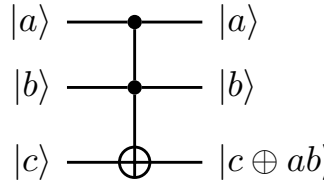
acts on the computational basis as

$$C\phi|a, b\rangle = e^{iab\phi}|a, b\rangle, \quad a, b \in F_2. \quad (89)$$

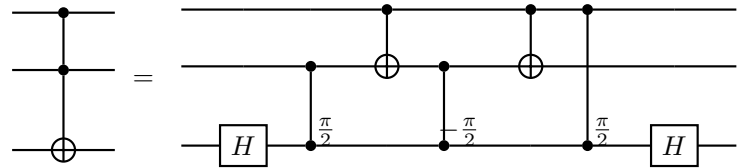
Exercise 1.10 Show that


(90)

We can also control a gate from more than one qubit. For instance, the **Toffoli gate**, denoted CCN , is a NOT gate controlled by two qubits. Its circuit diagram and action on a computational basis is given by


(91)

Exercise 1.11 Show that


(92)

1.2 Quantum teleportation and encryption

We now turn to the simplest quantum circuits which hint at the potential for quantum computation to perform tasks beyond the capability of classical computation. A new feature of these circuits compared to what we've discussed so far is that they involve measurements of qubits. These circuits, called **quantum teleportation** and **quantum encryption** (also called **superdense coding**) are not really examples of quantum computation. They are better described as **quantum information processing** since the interesting aspects of these circuits has to do with communication of information and the conversion of classical to quantum information (ie, classical bits to qubits) and vice versa. They are simple enough (and interesting enough) that they have actually been implemented experimentally, and even commercially.

The teleportation and encryption circuits use the properties of entangled states in a crucial way. So it is convenient to start by analyzing a simple “entangler” gate defined by

$$E := \begin{array}{c} \text{---} \boxed{H} \text{---} \bullet \text{---} \\ \text{---} \oplus \text{---} \end{array} . \quad (93)$$

It is easy enough to compute how E acts on the computational basis,

$$\begin{aligned} |\beta_{ab}\rangle &:= E|a, b\rangle = CN(H \otimes 1)|a, b\rangle \\ &= \frac{1}{\sqrt{2}}CN(|0, b\rangle + (-)^a|1, b\rangle) \\ &= \frac{1}{\sqrt{2}}(|0, b\rangle + (-)^a|1, b \oplus 1\rangle), \end{aligned} \quad (94)$$

where in the second line I used the action of H given in (41) and in the last line I used the action of CN given in (72). We have given a special name, $|\beta_{ab}\rangle$, to these four resulting states because they turn out to be so useful. We write them out explicitly:

$$\begin{aligned} |\beta_{00}\rangle &= \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), \\ |\beta_{01}\rangle &= \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle), \\ |\beta_{10}\rangle &= \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle), \\ |\beta_{11}\rangle &= \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle). \end{aligned} \quad (95)$$

We see that $|\beta_{00}\rangle$ is just the entangled EPR state introduced earlier; the other three are also entangled states.

Exercise 1.12 Calculate the output of $E(|\psi\rangle|0\rangle)$,

$$\begin{array}{c} |\psi\rangle \text{---} \boxed{H} \text{---} \bullet \text{---} \\ |0\rangle \text{---} \oplus \text{---} \end{array} \quad (96)$$

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

Exercise 1.13 Calculate the output of

$$\begin{array}{c} |\psi\rangle \text{---} \boxed{H} \text{---} \bullet \text{---} \bullet \text{---} \\ |0\rangle \text{---} \oplus \text{---} \text{---} \\ |0\rangle \text{---} \oplus \text{---} \end{array} \quad (97)$$

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

Exercise 1.14 Show that

$$E^{-1} := \begin{array}{c} \bullet \text{---} \boxed{H} \text{---} \\ \oplus \text{---} \end{array} \quad (98)$$

is the inverse of E , ie, that $E^{-1}E = 1$.

Measurements in the computational basis. We now add in measurements. In general, we can measure any hermitian operator, M . The possible results of the measurement are the eigenvalues of M , and the probability of a given eigenvalue being observed is given by the length-squared of the component of the state in the eigenspace corresponding to that eigenvalue. It turns out to be enough to consider just one very simple basic measurement, $M = P_1$, on a single qubit. The measurement operator is defined by its action on the computational basis of a single qubit,

$$P_1|a\rangle = a|a\rangle, \quad a \in F_2. \quad (99)$$

Obviously, the eigenvectors of P_1 are the computational basis vectors, $|a\rangle$, $a \in F_2$, and their eigenvalues are just $a \in F_2$, ie, either 0 (for $|0\rangle$) or 1 (for $|1\rangle$). The circuit diagram for such a measurement is

$$\text{---} \boxed{\text{meter}} \text{---} \quad . \quad (100)$$

We call this “measuring in the computational basis.”

If we have a qubit in state $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$, then measuring it in the computational basis will give result “0” with probability $|\alpha_0|^2$ and result “1” with probability $|\alpha_1|^2$, which we’ll describe by writing $\text{Prob}(a=0) = |\alpha_0|^2$, etc. Thus the result of a computational basis measurement is a *classical bit*, $a \in F_2$. We denote the outcome of the measurement diagrammatically by

$$\text{---} \boxed{\text{meter}} \text{---} \quad , \quad (101)$$

where the outcome $a \in F_2$ is written above the meter symbol.

Upon performing a measurement, the qubit state is changed to the eigenvector corresponding to the observed outcome. In other words, if we measure and find “0”, the qubit state becomes $|0\rangle$, and if we observe the result “1”, the qubit state becomes $|1\rangle$. Diagrammatically,

$$\alpha|0\rangle + \beta|1\rangle \text{---} \boxed{\text{meter}} \text{---} |a\rangle \quad . \quad (102)$$

This is sometimes called the “collapse of the wave function”. It should be clear that all the interesting information in the original qubit state — namely, the values of the complex coefficients α and β — is lost upon measurement. For this reason, we are generally not interested in the qubit state after a measurement, and typically discard the qubit from further consideration. The interesting qubit information encoded in α and β is transferred to the *probabilities* of the $a = 0$ or 1 outcomes occurring. (Note that that these probabilities are not notated in the circuit diagram: you will have to compute them for yourself when analyzing a given circuit.) Thus the interesting output of a measurement is the classical bit, $a \in F_2$, encoding the observed outcome of the measurement. For this reason we usually notate a computational basis measurement by

$$\text{---} \boxed{\text{meter}} \text{=} \quad , \quad (103)$$

where the double line on the right denotes a “classical wire” down which the classical bit a is sent.

Now let’s see what happens when we measure many qubits separately in the computational basis,

$$|\chi\rangle \left\{ \begin{array}{l} \text{---} \boxed{\text{meter}} \text{---} \text{=} \\ \text{---} \boxed{\text{meter}} \text{---} \text{=} \end{array} \right. \begin{array}{l} a \\ b \end{array} \quad . \quad (104)$$

When $|\chi\rangle$ is unentangled, ie, factorizes as $|\chi\rangle = |\psi\rangle|\phi\rangle$, it should be easy for you to find the probability of the possible (a, b) outcomes.

Exercise 1.15 Show that if $|\chi\rangle = (\alpha_0|0\rangle + \alpha_1|1\rangle)(\beta_0|0\rangle + \beta_1|1\rangle)$, then $\text{Prob}(a=0 \& b=0) = |\alpha_0\beta_0|^2$, $\text{Prob}(a=0 \& b=1) = |\alpha_0\beta_1|^2$, etc.

Since these probabilities factorize as products of probabilities for the first and second qubit measurements separately, we see that the outcomes of the two measurements are **uncorrelated**. But for a general state,

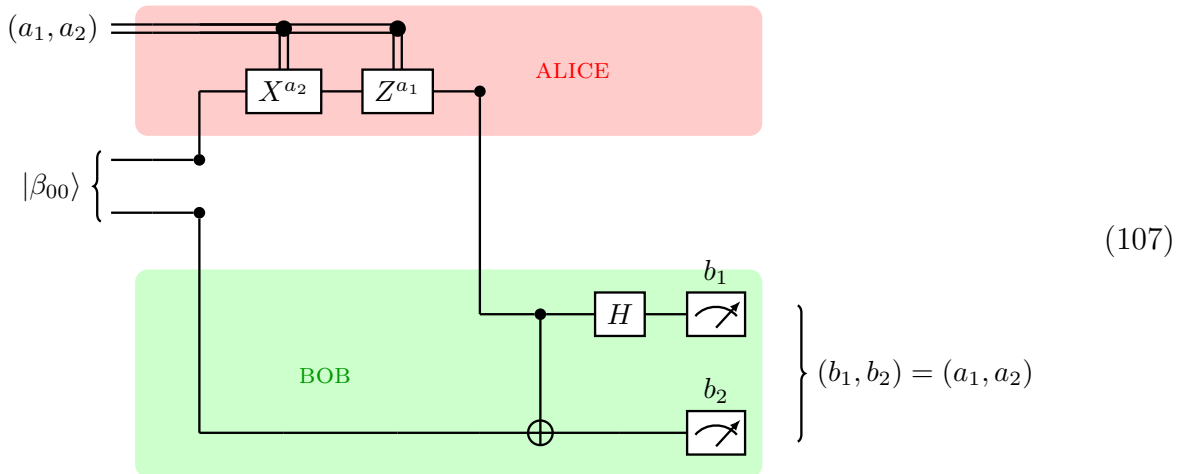
$$|\chi\rangle = \sum_{c,d \in F_2} \chi_{cd} |c, d\rangle, \quad (105)$$

the same analysis tells us that

$$\text{Prob}(a=c \& b=d) = |\chi_{cd}|^2, \quad (106)$$

which does not factorize for an entangled state, and so the two measurements are **correlated**. This is one of the hallmarks of entangled states.

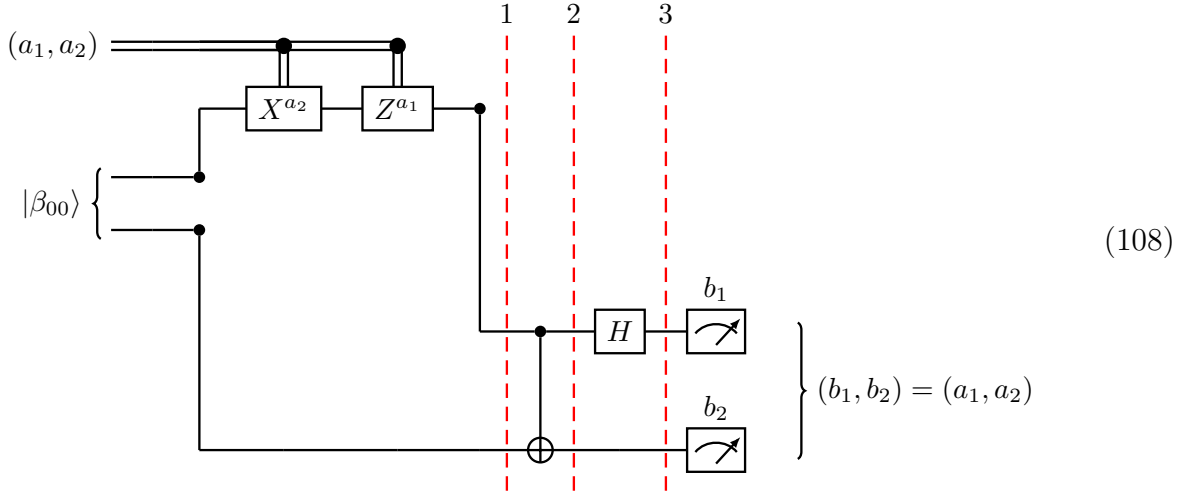
Quantum encryption. The quantum encryption circuit looks like



Here we imagine two widely-separated people, Alice and Bob, who each get one qubit from an entangled pair in the state $|\beta_{00}\rangle$. We show the qubits that Alice and Bob have access to

by the shaded boxes. We imagine that the entangled pair is prepared in that state, say via an entangler gate, at some location, and then the qubits are separated and sent to Alice and Bob. Also, Alice has two classical bits of data, (a_1, a_2) which she wishes to send to Bob by physically transferring her qubit to Bob. Alice uses these two bits to determine what gates (X^{a_2} then Z^{a_1}) she sends her qubit through, before sending her qubit to Bob. The claim, indicated in (107), is that after performing this transfer, if Bob sends his two qubits through the indicated gates and performs a measurement in the computational basis on them, the result he will observe will be Alice's original 2 bits (a_1, a_2) with 100% certainty. The only information communicated between Alice and Bob is through the transfer of Alice's qubit.

Let us now verify that that is, in fact, how this circuit behaves. To do this analysis, it is convenient to name some intermediate states:



We denote by $|\chi_j\rangle$ the state of the qubits at the j th slice. From now on the a, b, c, d, e indices are understood to be in F_2 . The initial state is $|\beta_{00}\rangle = \frac{1}{\sqrt{2}} \sum_b |bb\rangle$, so

$$\begin{aligned}
 |\chi_1\rangle &= Z_1^{a_1} X_1^{a_2} |\beta_{00}\rangle = \frac{1}{\sqrt{2}} \sum_b (Z^{a_1} \otimes 1)(X^{a_2} \otimes 1) |bb\rangle \\
 &= \frac{1}{\sqrt{2}} \sum_b (Z^{a_1} X^{a_2} |b\rangle) \otimes |b\rangle = \frac{1}{\sqrt{2}} \sum_b (Z^{a_1} |b \oplus a_2\rangle) \otimes |b\rangle \\
 &= \frac{1}{\sqrt{2}} \sum_b ((-)^{a_1(b \oplus a_2)} |b \oplus a_2\rangle) \otimes |b\rangle = \frac{1}{\sqrt{2}} \sum_b (-)^{a_1(b \oplus a_2)} |b \oplus a_2, b\rangle.
 \end{aligned} \tag{109}$$

Here we have used the facts that

$$X^a |b\rangle = |b \oplus a\rangle, \quad Z^a |b\rangle = (-)^{ab} |b\rangle, \quad a, b \in F_2. \tag{110}$$

Exercise 1.16 Verify (110).

Since $CN|a, b\rangle = |a, b \oplus a\rangle$, we have for the next slice

$$\begin{aligned}
 |\chi_2\rangle &= CN|\chi_1\rangle = \frac{1}{\sqrt{2}} \sum_b (-)^{a_1(b \oplus a_2)} CN|b \oplus a_2, b\rangle \\
 &= \frac{1}{\sqrt{2}} \sum_b (-)^{a_1(b \oplus a_2)} |b \oplus a_2, b \oplus b \oplus a_2\rangle = \frac{1}{\sqrt{2}} \sum_b (-)^{a_1(b \oplus a_2)} |b \oplus a_2, a_2\rangle,
 \end{aligned} \tag{111}$$

where in the last step we used that $a \oplus b \oplus b = a$ in mod 2 arithmetic. Next, using that $H|a\rangle = \frac{1}{\sqrt{2}}(|0\rangle + (-)^a|1\rangle)$, we get for the final slice

$$\begin{aligned}
|\chi_3\rangle &= (H \otimes 1)|\chi_2\rangle = \frac{1}{\sqrt{2}} \sum_b (-)^{a_1(b \oplus a_2)} (H \otimes 1)|b \oplus a_2, a_2\rangle \\
&= \frac{1}{\sqrt{2}} \sum_b (-)^{a_1(b \oplus a_2)} (H|b \oplus a_2\rangle) \otimes |a_2\rangle = \frac{1}{2} \sum_b (-)^{a_1(b \oplus a_2)} (|0\rangle + (-)^{b \oplus a_2}|1\rangle) \otimes |a_2\rangle \\
&= \frac{1}{2} \sum_c (-)^{a_1 c} (|0\rangle + (-)^c|1\rangle) \otimes |a_2\rangle = \left(\frac{1}{2} \sum_c [(-)^{a_1 c}|0\rangle + (-)^{c(a_1+1)}|1\rangle] \right) \otimes |a_2\rangle,
\end{aligned} \tag{112}$$

where in the second-to-last step we changed variables in the sum to $c = b \oplus a_2$, which is allowed since b and c run over the same range (ie, F_2) whatever the value of a_2 . Now

$$\begin{aligned}
\frac{1}{2} \sum_c [(-)^{a_1 c}|0\rangle + (-)^{c(a_1+1)}|1\rangle] &= \frac{1}{2} [(-)^0|0\rangle + (-)^0|1\rangle] + \frac{1}{2} [(-)^{a_1}|0\rangle + (-)^{a_1+1}|1\rangle] \\
&= \frac{1 + (-)^{a_1}}{2}|0\rangle + \frac{1 - (-)^{a_1}}{2}|1\rangle \\
&= \delta_{a_1,0}|0\rangle + \delta_{a_1,1}|1\rangle = |a_1\rangle,
\end{aligned} \tag{113}$$

so (112) becomes simply the computational basis state

$$|\chi_3\rangle = |a_1\rangle \otimes |a_2\rangle. \tag{114}$$

Therefore, upon measuring this state in the computational basis, we get the result (a_1, a_2) with 100% certainty, verifying the claimed behavior of the quantum encryption circuit.

Exercise 1.17 Show that $\frac{1}{2}(1 + (-)^a) = \delta_{a,0}$ and $\frac{1}{2}(1 - (-)^a) = \delta_{a,1}$.

The encryption circuit shows that, if two people share an entangled pair of qubits, then two classical bits can be encoded in one qubit. This is why this circuit is also referred to as “superdense coding”. The feature of this circuit which is novel compared to classical communication channels is that if a third party, Carol, were to intercept the qubit that Alice sends to Bob, then Carol would gain *no information* about the two bits (a_1, a_2) being encoded.

To see this, say Carol captures the first qubit of the state $|\chi_1\rangle$ and measures it in the computational basis, getting a classical bit $c \in F_2$, as a result. This means that Carol measures the observable $P_1 \otimes 1$, since she is assumed to have no access to Bob’s qubit. Recall that the probability of getting c is the square of the projection of the state onto the eigenspace of the measured eigenvalues. The eigenspace of $P_1 \otimes 1$ with eigenvalue c is the 2-dimensional space spanned by $|c, d\rangle$ for $d \in F_2$.

Exercise 1.18 Check that $|c, d\rangle$ is an eigenstate of $P_1 \otimes 1$ with eigenvalue c for any $d \in F_2$.

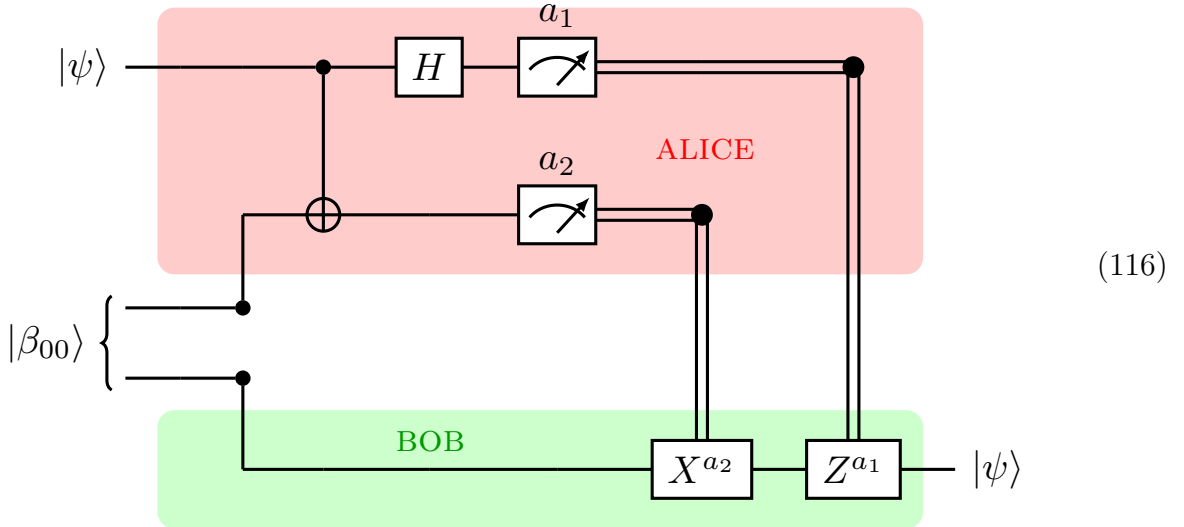
Therefore,

$$\begin{aligned}
\text{Prob}(c) &= \sum_d |\langle c, d | \chi_1 \rangle|^2 = \sum_d \left| \frac{1}{\sqrt{2}} \sum_b (-)^{a_1(b \oplus a_2)} \langle c, d | b \oplus a_2, b \rangle \right|^2 \\
&= \sum_d \left| \frac{1}{\sqrt{2}} \sum_b (-)^{a_1(b \oplus a_2)} \langle c | b \oplus a_2 \rangle \langle d | b \rangle \right|^2 = \sum_d \left| \frac{1}{\sqrt{2}} \sum_b (-)^{a_1(b \oplus a_2)} \delta_{c, b \oplus a_2} \delta_{d, b} \right|^2 \\
&= \sum_d \left| \frac{1}{\sqrt{2}} (-)^{a_1(d \oplus a_2)} \delta_{c, d \oplus a_2} \right|^2 = \left| \frac{1}{\sqrt{2}} (-)^{a_1 c} \right|^2 = \frac{1}{2},
\end{aligned} \tag{115}$$

This result shows that the probability is $1/2$ for the eavesdropper Carol to find either of the two possible $c \in F_2$ outcomes, independent of the values of a_1 and a_2 . Thus Alice's transmitted qubit by itself carries no information about the encoded classical bits. This is therefore a communication method which cannot be decoded *in principle*!

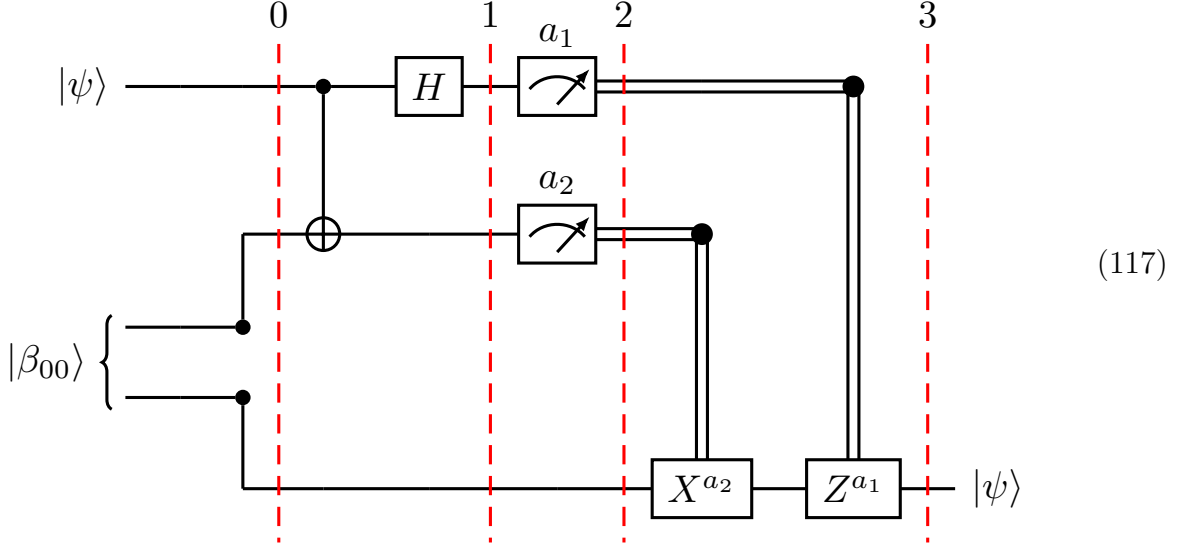
This is very different from the behavior of classical information processing, where communication channels always carry information about the message being transferred. This is our first indication that quantum information processing can do something qualitatively different than what can be done just with classical bits.

Quantum teleportation. We now turn to the quantum teleportation circuit, given by



Here we again imagine widely-separated Alice and Bob each sharing one qubit from an entangled pair in the state $|\beta_{00}\rangle$. Also, Alice has another (unentangled) qubit prepared in some arbitrary state $|\psi\rangle$. The claim, indicated in (116), is that after performing the quantum teleportation, the qubit Bob has ends up in the state $|\psi\rangle$. The only information communicated between Alice and Bob is that Alice sends Bob 2 classical bits of information, (a_1, a_2) , encoding the outcome of two computational basis measurements she performs. Bob uses these two bits to determine what gates (X^{a_2} then Z^{a_1}) he sends his qubit through.

Let us now verify that that is, in fact, how this circuit behaves. We name some intermediate states,



and denote by $|\chi_j\rangle$ the state of the qubits at the j th slice. Write Alice's first qubit state as $|\psi\rangle = \sum_{b \in F_2} \psi_b |b\rangle$. Then the initial state is

$$|\chi_0\rangle = |\psi\rangle |\beta_{00}\rangle = \left(\sum_b \psi_b |b\rangle \right) \left(\frac{1}{\sqrt{2}} \sum_c |cc\rangle \right) = \frac{1}{\sqrt{2}} \sum_{bc} \psi_b |bcc\rangle, \quad (118)$$

and the state at the next slice is

$$\begin{aligned} |\chi_1\rangle &= (H \otimes 1 \otimes 1)(CN \otimes 1) \left(\frac{1}{\sqrt{2}} \sum_{bc} \psi_b |bcc\rangle \right) = \frac{1}{\sqrt{2}} \sum_{bc} \psi_b (H \otimes 1 \otimes 1)(CN \otimes 1) |bcc\rangle \\ &= \frac{1}{\sqrt{2}} \sum_{bc} \psi_b (H \otimes 1 \otimes 1) |b, c \oplus b, c\rangle = \frac{1}{2} \sum_{bc} \psi_b (|0, c \oplus b, c\rangle + (-)^b |1, c \oplus b, c\rangle) \\ &= \frac{1}{2} \sum_{bcd} (-)^{db} \psi_b |d, c \oplus b, c\rangle, \end{aligned} \quad (119)$$

where in the last line we have just written sum in the previous step in a more compact form.

Now Alice performs her two measurements. If she finds (a_1, a_2) as the result, the result will be that the first two qubits will be projected to those values. This projection just means that we should keep only the terms with $d = a_1$ and $c \oplus b = a_2$ in the sum, so

$$|\chi_2\rangle \propto \frac{1}{2} \sum_{bcd} (-)^{db} \psi_b |d, c \oplus b, c\rangle \delta_{a_1, d} \delta_{a_2, b \oplus c}. \quad (120)$$

Here we have indicated only that $|\chi_2\rangle$ is proportional to the right side, because, as always after a projective measurement, we have to re-normalize the resulting vector so that it is a state of norm 1. We can simplify the sum in (120) by noting that

$$a_2 = b \oplus c \quad \text{if and only if} \quad c = a_2 \oplus b, \quad (121)$$

which is easy to check in mod 2 arithmetic. (Just add $\oplus b$ to both sides of the first equation to get $b \oplus c \oplus b = a_2 \oplus b$ and then use the fact that $b \oplus b = 2b = 0 \text{ mod } 2$.) Then we get

$$\begin{aligned}
|\chi_2\rangle &\propto \frac{1}{2} \sum_{bcd} (-)^{db} \psi_b |d, c \oplus b, c\rangle \delta_{a_1, d} \delta_{c, a_2 \oplus b} \\
&\propto \frac{1}{2} \sum_b (-)^{a_1 b} \psi_b |a_1, a_2 \oplus b \oplus b, a_2 \oplus b\rangle \\
&\propto \frac{1}{2} |a_1 a_2\rangle \left(\sum_b (-)^{a_1 b} \psi_b |a_2 \oplus b\rangle \right). \tag{122}
\end{aligned}$$

Exercise 1.19 Show that the norm of $\sum_b (-)^{a_1 b} \psi_b |a_2 \oplus b\rangle$ is 1.

Therefore the normalized state at slice 2 is

$$|\chi_2\rangle = |a_1 a_2\rangle \left(\sum_b (-)^{a_1 b} \psi_b |a_2 \oplus b\rangle \right). \tag{123}$$

We drop the first two qubits, $|a_1 a_2\rangle$, from now on, since they are not entangled and play no further role in the circuit. The final state (slice 3) is then given by

$$\begin{aligned}
|\chi_3\rangle &= Z^{a_1} X^{a_2} \left(\sum_b (-)^{a_1 b} \psi_b |a_2 \oplus b\rangle \right) = \sum_b (-)^{a_1 b} \psi_b Z^{a_1} X^{a_2} |a_2 \oplus b\rangle \\
&= \sum_b (-)^{a_1 b} \psi_b Z^{a_1} |a_2 \oplus b \oplus a_2\rangle = \sum_b (-)^{a_1 b} \psi_b Z^{a_1} |b\rangle \\
&= \sum_b (-)^{a_1 b} \psi_b (-)^{a_1 b} |b\rangle = \sum_b \psi_b |b\rangle = |\psi\rangle. \tag{124}
\end{aligned}$$

This finishes the demonstration that the teleportation circuit has the advertised behavior.

The teleportation circuit shows that, if two people share an entangled pair of qubits, then one qubit can be encoded in two classical bits. The feature of this circuit which is novel compared to classical communication channels is that if a third party, Carol, were to intercept the two classical bits that Alice sends to Bob, then Carol would gain *no information* about the state $|\psi\rangle$ being teleported.

To see this, let us calculate the probability that the classical bits (a_1, a_2) are measured by Alice. Recalling that the probability of getting such a result is the square of the projection of the state onto the eigenspace of the measured eigenvalues. The eigenspace of eigenvalues (a_1, a_2) is the 2-dimensional space spanned by $|a_1, a_2, e\rangle$ for $e \in F_2$.

Exercise 1.20 Check that $|a_1, a_2, e\rangle$ is an eigenstate of both $P_1 \otimes 1 \otimes 1$ and $1 \otimes P_1 \otimes 1$ (which are the two measurements that Alice performs) with eigenvalues (a_1, a_2) for any $e \in F_2$.

Therefore,

$$\begin{aligned}
\text{Prob}(a_1, a_2) &= \sum_e |\langle a_1, a_2, e | \chi_1 \rangle|^2 = \sum_e \left| \frac{1}{2} \sum_{bcd} (-)^{db} \psi_b \langle a_1, a_2, e | d, c \oplus b, c \rangle \right|^2 \\
&= \frac{1}{4} \sum_e \left| \sum_{bcd} (-)^{db} \psi_b \delta_{a_1, d} \delta_{a_2, c \oplus b} \delta_{e, c} \right|^2 = \frac{1}{4} \sum_e \left| \sum_b (-)^{a_1 b} \psi_b \delta_{b, a_2 \oplus e} \right|^2 \\
&= \frac{1}{4} \sum_e |(-)^{a_1(a_2 \oplus e)} \psi_{a_2 \oplus e}|^2 = \frac{1}{4} \sum_e |\psi_{a_2 \oplus e}|^2 = \frac{1}{4} \sum_f |\psi_f|^2 \\
&= \frac{1}{4}.
\end{aligned} \tag{125}$$

In the second-to-last step I changed the summation variable from e to $f = e \oplus a_2$, which is allowed since they run over the same set, F_2 . This result shows that probability is $1/4$ to find any of the four possible (a_1, a_2) bits, independent of the values of a_1 and a_2 . Thus these classical bits carry no information about the state $|\psi\rangle$.

Again, just as with the encryption circuit, this is very different from the behavior of classical information processing. This is our second indication that quantum information processing can do something qualitatively different than what can be done just with classical bits.

2 Some basic quantum algorithms

We now introduce a few simple quantum algorithms which are illustrations of how a quantum computer can solve certain problems more efficiently than classical computers. These algorithms also illustrate a general strategy for constructing quantum algorithms. We outline this strategy first in somewhat heuristic terms, then illustrate it in two specific cases, the **Deutsch-Jozsa algorithm** and **Grover's algorithm**.

The basic strategy is to do the following sequence,

$$\text{initialize} \rightarrow \text{interfere} \rightarrow \text{evaluate } f \rightarrow \text{measure } M.$$

Here by “initialize” we mean prepare the n input qubits in a given unentangled state, usually

$$|0\rangle^{\otimes n} \text{---}^n. \tag{126}$$

By “interfere” we mean transform the input state to a superposition of both $|0\rangle$ and $|1\rangle$ for each qubit. A typical way this is done is

$$\text{---}^n \boxed{H^{\otimes n}} \text{---}^n. \tag{127}$$

Just to be clear, what we mean by the $H^{\otimes n}$ gate is simply a Hadamard gate acting on each

input qubit separately,

$$\begin{array}{c}
 \left. \begin{array}{c}
 \text{---} \boxed{H} \text{---} \\
 \text{---} \boxed{H} \text{---} \\
 \vdots \\
 \text{---} \boxed{H} \text{---}
 \end{array} \right\} n. \quad (128)
 \end{array}$$

Exercise 2.1 If the input state is $|0\rangle^{\otimes n}$, show that the output of $H^{\otimes n}$ is simply

$$\frac{1}{2^{n/2}} \sum_{a_i \in F_2^n} |a_1 a_2 \dots a_n\rangle, \quad (129)$$

ie, an equal superposition of all the computational basis states.

Note that this is an unentangled state. One can interfere the initial state in other ways which involve more complicated phases than just the minus signs that appear in the Hadamard gate. One very common way — used, for example in the Shor factorization algorithm described later in section 5 — is to Fourier transform the initial state,

$$|0\rangle^{\otimes n} \xrightarrow{n} \boxed{QFT} \xrightarrow{n}, \quad (130)$$

as will be described in detail in section 4.

The “evaluate” step means we compute some function, f , which we’ll denote by the n -qubit gate U_f ,

$$\xrightarrow{n} \boxed{U_f} \xrightarrow{n}. \quad (131)$$

We will describe this kind of quantum circuit shortly.

The “measure” step denotes measurement of some or all of the qubits in the computational basis, thus giving an n -classical-bit output, M .

2.1 Function gates

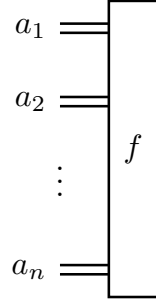
A common kind of map which we want to evaluate is a function, f , from n bits to one bit,

$$f : F_2^n \rightarrow F_2, \quad (132)$$

which just means that given n bits, (a_1, a_2, \dots, a_n) , f computes a single output bit

$$b = f(a_1, \dots, a_n), \quad a_i, b \in F_2. \quad (133)$$

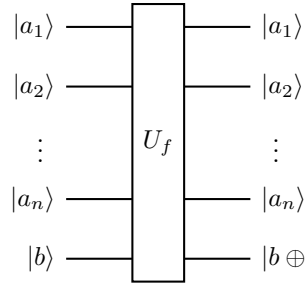
A classical gate or circuit which does this is often written



$$a_i, b \in F_2. \quad (134)$$

But such a gate is not reversible. Furthermore, to implement it as a quantum gate it must also be unitary.

A quantum function gate can be realized by an $(n + 1)$ -qubit gate, U_f , defined by



$$. \quad (135)$$

In particular, if we take the last qubit input state to be $|b\rangle = |0\rangle$, then the last qubit is output in the state $|f(a_1 \dots a_n)\rangle$, so computes the function f .

To see that such a gate is possible, we need to check that it is unitary, $U_f^\dagger U_f = 1$. This is not too hard to show: From (135), we have

$$U_f |a_1 \dots a_n, b\rangle = |a_1 \dots a_n, b \oplus f\rangle, \quad (136)$$

so we immediately read off that

$$U_f = \sum_{a_i, b} |a_1 \dots a_n, b \oplus f\rangle \langle a_1 \dots a_n, b|, \quad (137)$$

and therefore that

$$U_f^\dagger = \sum_{a_i, b} |a_1 \dots a_n, b\rangle \langle a_1 \dots a_n, b \oplus f|. \quad (138)$$

Now compute

$$\begin{aligned}
U_f^\dagger U_f &= \left(\sum_{a_i, b} |(a_i), b\rangle \langle (a_i), b \oplus f(a_i)| \right) \cdot \left(\sum_{c_i, d} |(c_i), d \oplus f(c_i)\rangle \langle (c_i), d| \right) \\
&= \sum_{a_i, b, c_i, d} |(a_i), b\rangle \langle (c_i), d| \left(\langle (a_i), b \oplus f(a_i) | (c_i), d \oplus f(c_i) \rangle \right) \\
&= \sum_{a_i, b, c_i, d} |(a_i), b\rangle \langle (c_i), d| \left(\prod_{i=1}^n \delta_{a_i, c_i} \right) \delta_{b \oplus f(a_i), d \oplus f(c_i)} \\
&= \sum_{a_i, b, d} |(a_i), b\rangle \langle (a_i), d| \delta_{b \oplus f(a_i), d \oplus f(a_i)} = \sum_{a_i, b, d} |(a_i), b\rangle \langle (a_i), d| \delta_{b, d} \\
&= \sum_{a_i, b} |(a_i), b\rangle \langle (a_i), b| = 1,
\end{aligned} \tag{139}$$

thus showing that U_f is indeed unitary. Here in the first step we changed the summation variables in the second (U_f) factor so that they do not clash with the names of the variables used in the first (U_f^\dagger) factor. In the third line we used the orthonormality (3) of the computational basis to evaluate the bracket, and in the next three lines we simplified the sums using the Kronecker deltas. The last step follows from the completeness (8) of the computational basis.

So such a gate is possible. The actual construction of this gate in terms of, say, 1- and 2-qubit gates, depends on the specific function, f , being implemented. Here is a brief outline of one way of constructing such a gate for any given f . First, consider the function $f = \delta((a_i) = 1^n)$ which is defined to be zero for all inputs except for the input $a_1 \dots a_n = 1 \dots 1$ (ie, all 1's), in which case it gives output 1. The gate which does this is the $C^n N$ gate, ie, a controlled-NOT gate where the last qubit is controlled by the first n -qubits. This is thus an n -qubit generalization of the controlled-NOT and Toffoli gates. A circuit for the $C^n N$ gate can be constructed by generalizing the circuits given in (90) and (92). Now consider the function $f = \delta((a_i) = (a_i^*))$ which is zero for all inputs except the input $a_1 \dots a_n = a_1^* \dots a_n^*$, where $a_1^* \dots a_n^*$ is a specific fixed string of n bits. Noting that the input state $|a_1^* \dots a_n^*\rangle$ can be converted to $|1^n\rangle$ by acting on the j th qubit with an $X^{a_j^*+1}$ -gate, we see that a circuit that implements $U_{\delta((a_i)=(a_i^*))}$ is this arrangement of X gates followed by the $C^n N$ gate. Finally, since any function can be written as a sum of such δ -functions, $f(a_1 \dots a_n) = \sum_{(a_i^*)} \delta((a_i) = (a_i^*))$ where the sum is over all strings $(a_1^* \dots a_n^*)$ for which f is 1, we can therefore implement U_f for any f by concatenating these U_δ gates.

For a complicated or random enough function f , the number of gates needed to implement U_f grows with the number of input qubits very fast, as $n^2 \cdot n!$. So usually the step of computing the function, ie, each use of the U_f gate, is the computationally most intensive part of an algorithm.

2.2 Deutsch-Jozsa algorithm

The Deutsch-Jozsa algorithm is a very simple example showing how a quantum computation can solve a problem efficiently which is computationally intensive for a classical computer. The problem the Deutsch-Jozsa algorithm solves is one of a class of problems known as

decision problems, which have the general form: given a function f , determine whether f has a certain property or not. More colloquially, decision problems are ones which ask a yes-or-no question. The Deutsch-Jozsa problem is a fairly artificial example of a decision problem (ie, I do not think it comes up as part of any practical problem), so its interest is mainly as a proof of principle that quantum computation may have an advantage over classical computation in some problems. But the *way* the Deutsch-Jozsa algorithm works is very illuminating: whereas the classical algorithm has to compute the function many times (ie, for many different inputs) to decide whether or not it has the property in question, the quantum algorithm only has to compute the function a single time! The basic idea is that by putting the input state into a superposition of all computational basis states (the “interference” step), by then applying the U_f gate just once we are effectively computing f on *all* its input values at once! Since it is the evaluation of f which is computationally intensive, the quantum algorithm becomes much (exponentially) more efficient than the classical algorithm as n gets large.

The Deutsch-Jozsa problem. Suppose we are given some function $f : F_2^n \rightarrow F_2$ and we know in advance that it is either *constant* or *balanced*. Determine which it is.

To understand what this means, we need to define *constant* and *balanced* functions. A function f is constant if it gives the same output for all inputs,

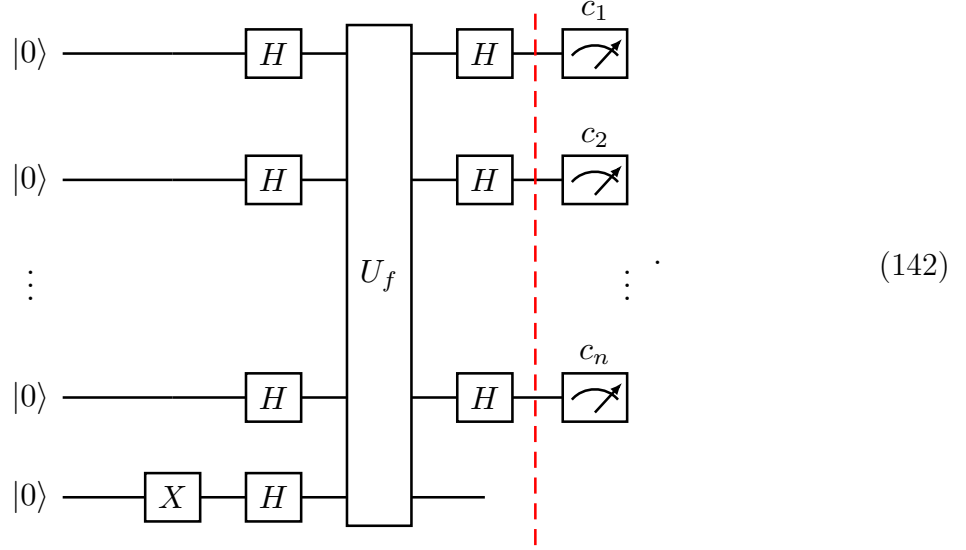
$$f \text{ constant} \Leftrightarrow \begin{cases} \text{either} & f(a_1 \dots a_n) = 0 \quad \text{for all } (a_1 \dots a_n), \\ \text{or} & f(a_1 \dots a_n) = 1 \quad \text{for all } (a_1 \dots a_n). \end{cases} \quad (140)$$

A function is balanced if it is 0 on exactly half its inputs, and therefore 1 on the other half,

$$f \text{ balanced} \Leftrightarrow f(a_1 \dots a_n) = \begin{cases} 0 & \text{for half } (2^{n-1}) \text{ of the } (a_1 \dots a_n)\text{'s,} \\ & \text{and} \\ 1 & \text{for the other half of the } (a_1 \dots a_n)\text{'s.} \end{cases} \quad (141)$$

Classical algorithm. The only way to decide in general whether f is constant or balanced is to evaluate f on half plus 1 of its possible inputs. If these evaluations give all the same output, then f must be constant; otherwise it is balanced. Since there are 2^n possible inputs, this means that we have to evaluate f $2^{n-1} + 1$ times.

Quantum algorithm. Run the circuit



If the measurements give $(c_1 c_2 \dots c_n) = (00 \dots 0)$ (all zeros) then f is constant; otherwise it is balanced. This circuit obviously only uses the U_f gate once, so is exponentially faster than the classical algorithm. Note also that the circuit conforms to the “interfere \rightarrow evaluate \rightarrow measure” pattern advertised earlier.

To see that this circuit works as advertised, compute the state at the slice shown in red:

$$\begin{aligned}
|\psi\rangle &= (H^{\otimes n} \otimes 1) U_f H^{\otimes(n+1)} (1^{\otimes n} \otimes X) |0\rangle^{\otimes n} \\
&= (H^{\otimes n} \otimes 1) U_f H^{\otimes(n+1)} (|0\rangle^{\otimes n} \otimes |1\rangle) \\
&= \frac{1}{2^{n/2}} (H^{\otimes n} \otimes 1) U_f \sum_{a_i \in F_2^n} |a_1 \dots a_n\rangle \otimes \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \\
&= \frac{1}{2^{n/2}} (H^{\otimes n} \otimes 1) \sum_{a_i \in F_2^n} |a_1 \dots a_n\rangle \otimes \frac{1}{\sqrt{2}} (|f(a_i)\rangle - |1 \oplus f(a_i)\rangle) \\
&= \frac{1}{2^{n/2}} (H^{\otimes n} \otimes 1) \sum_{a_i \in F_2^n} (-)^{f(a_i)} |a_1 \dots a_n\rangle \otimes \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \\
&= \frac{1}{2^n} \sum_{a_i, b_i \in F_2^n} (-)^{f(a_i) + \sum_i a_i b_i} |a_1 \dots a_n\rangle \otimes \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle).
\end{aligned} \tag{143}$$

In the fifth line we used the identity

$$|f(a_i)\rangle - |1 \oplus f(a_i)\rangle = (-)^{f(a_i)} (|0\rangle - |1\rangle), \tag{144}$$

and in the last line we used that

$$H^{\otimes n} |a_1 \dots a_n\rangle = \frac{1}{2^{n/2}} \sum_{b_i \in F_2^n} (-)^{\sum_i a_i b_i} |b_1 \dots b_n\rangle. \tag{145}$$

Exercise 2.2 Verify (144) and (145).

Now compute the probability that upon measuring the first n qubits in the computational basis we get the result $(c_1 \dots c_n) = (0 \dots 0)$:

$$\begin{aligned}
 \text{Prob}[(c_1 \dots c_n) = (0 \dots 0)] &= \sum_{d \in F_2} |\langle 0 \dots 0, d | \psi \rangle|^2 \\
 &= \sum_{d \in F_2} \left| \frac{1}{2^n \sqrt{2}} \sum_{a_i \in F_2^n} (-)^{f(a_i)} (\langle d | 0 \rangle - \langle d | 1 \rangle) \right|^2 \\
 &= 2 \cdot \frac{1}{2} \cdot \left| \frac{1}{2^n} \sum_{a_i \in F_2^n} (-)^{f(a_i)} \right|^2.
 \end{aligned} \tag{146}$$

Exercise 2.3 Justify each step in the calculation (146).

Now, the expression inside the absolute value signs in (146) is ± 1 if f is constant (since then all terms in the sum are equal) and 0 if f is balanced (since then half the terms in the sum are $+1$ and half are -1). Thus we have

$$\text{Prob}[(c_1 \dots c_n) = (0 \dots 0)] = \begin{cases} 1 & \text{if } f \text{ is constant,} \\ 0 & \text{if } f \text{ is balanced,} \end{cases} \tag{147}$$

which was the claimed behavior.

2.3 Grover's algorithm

Grover's algorithm is another example showing how a quantum algorithm can solve a problem more efficiently than a classical computer. Grover's algorithm solves a **search problem**: given a function $f : F_2^n \rightarrow F_2$ such that $f = 0$ for all inputs except for a single $x \in F_2^n$ for which $f(x) = 1$, find x . This is called a search problem because it models the problem of finding the object, x , in a large database F_2^n (which has $N = 2^n$ elements) which satisfies some search criterium, encoded in the function f . If the database F_2^n is unstructured (ie, if the function f is complicated or random enough), then a classical search requires the evaluation of f an order $\mathcal{O}(N)$ times. Grover's algorithm performs the search with only an order $\mathcal{O}(\sqrt{N})$ evaluations of f , a quadratic speed-up over the classical search.

The search problem Grover's algorithm solves may seem artificial since it assumes we know in advance that there is only a single element $x \in F_2^n$ satisfying the search criterium $f = 1$. A more reasonable search problem is to find an element satisfying $f = 1$ when there may be multiple such elements in the database. Grover's algorithm can be generalized to this case, and performs the search with order $\mathcal{O}(\sqrt{N/M})$ evaluations of f where M is the number of elements satisfying $f = 1$, and the improved algorithm also determines what M is. This is again a quadratic speed-up over a classical search. We will not describe this generalization of Grover's algorithm here; the generalization is an example of a **quantum counting** algorithm, which are described in section 6.3 of [NC].

We will describe Grover's algorithm for a quantum computer operating on an $(n + 1)$ -qubit state space. This machine therefore has a Hilbert space of dimension $2^{n+1} = 2^n \cdot 2$. We will think of this Hilbert space as the tensor product of two Hilbert spaces $\mathcal{H}_N \otimes \mathcal{H}_2$ of dimensions $N := 2^n$ and 2, respectively. The \mathcal{H}_N represents our database or search space, and the "auxiliary" \mathcal{H}_2 qubit will play a secondary role (and, in fact, can be essentially ignored after the initialization step, though it is necessary to implement the algorithm using quantum gates, ie, unitary operators).

Denote the computational orthonormal basis of \mathcal{H}_N by $|x\rangle$ for $x \in \{0, 1, \dots, N-1\}$ where, as usual, the binary expression for x gives the n -qubit computational basis state. Denote the computational basis of the auxiliary \mathcal{H}_2 qubit by $|a\rangle$, $a \in F_2$. Thus the general basis state of the quantum computer is $|x\rangle|a\rangle$, and a general state of the computer is $|\psi\rangle = \sum_{x=0}^{N-1} \sum_{a=0}^1 C_{xa} |x\rangle|a\rangle$ for some coefficients C_{xa} .

Grover's algorithm

Input: A function $f : F_2^n \rightarrow F_2$ such that $f(x) = 0$ for all $x \in F_2^n$ except one.

Steps: **1.** Prepare the machine in the initial state $|\psi_0\rangle := |0\rangle \otimes |0\rangle$ and apply an $H^{\otimes n} \otimes HX$ gate to put the machine in the uniform superposition of states with all values of $x \pmod N$ in the first factor, and in the state $(|0\rangle - |1\rangle)/\sqrt{2}$ in the second factor:

$$|\psi'_0\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \otimes \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle). \quad (148)$$

2. Apply the U_f gate to the $n + 1$ qubits to get:

$$|\psi_1\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} (-)^{f(x)} |x\rangle \otimes \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle). \quad (149)$$

3. Apply an $H^{\otimes n}(2|0\rangle\langle 0| - I)H^{\otimes n}$ gate to the first n qubits, doing nothing to the auxiliary qubit, to get:

$$|\psi'_1\rangle = \frac{1}{\sqrt{N}} \sum_x \left(2 - (-)^{f(x)} - \frac{4}{N}\right) |x\rangle \otimes \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle). \quad (150)$$

4. Repeat steps **2** and **3** T times, where

$$T = \left\lceil \frac{\pi}{4} \sqrt{N} \right\rceil. \quad (151)$$

Output: Measure the state of the first n qubits with respect to the computational basis. This returns a value for $x \in F_2^n$ such that $f(x) = 1$ with high probability, namely, $P > 1 - \frac{1}{N}$.

We now analyze each of these steps, to verify that the circuit works as advertised.

Step 1 is an initialization step, which in terms of gates is

$$\left. \begin{array}{l} |0\rangle^{\otimes n} \xrightarrow{\text{---}^n} \boxed{H^{\otimes n}} \xrightarrow{\text{---}^n} \\ |0\rangle \xrightarrow{\text{---}} \boxed{X} \xrightarrow{\text{---}} \boxed{H} \end{array} \right\} |\psi'_0\rangle \quad . \quad (152)$$

You should be able to easily check that $|\psi'_0\rangle$ given in (148) is the output of this circuit.

Step 2 is simply an application of the U_f function gate on our $n + 1$ qubits,

$$|\psi'_0\rangle \left\{ \begin{array}{c} \xrightarrow{\text{---}^n} \\ \text{---} \end{array} \right\} \boxed{U_f} \left\{ \begin{array}{c} \xrightarrow{\text{---}^n} \\ \text{---} \end{array} \right\} |\psi_1\rangle \quad . \quad (153)$$

Again, you should be able to easily check that $|\psi_1\rangle$ given in (149) is the output of this circuit using the identity

$$|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle = (-)^{f(x)} (|0\rangle - |1\rangle) \quad (154)$$

that you are familiar with from exercise 4.2. Note that the effect of this gate therefore leaves the state of the last qubit unchanged, and acts only to change the sign of any state $|x\rangle$ of the first n qubits if $f(x) = 1$. The rest of the circuit does not change the last qubit, so it can essentially be ignored from now on. Since, by assumption, $f(x) = 1$ for only one value, $x = x_*$, of x (thus x_* is the output we are trying to get), it follows that the state $|\psi_1\rangle$ looks just like the initial state $|\psi_0\rangle$ except the sign of only that one particular $|x_*\rangle$ state is flipped. We describe this by saying that the coefficient of $|x\rangle$ in $|\psi_1\rangle$ is

$$\text{amplitude}(|x\rangle) := a_x \sim \begin{cases} +1/\sqrt{N} & \text{if } x \neq x_* \\ -1/\sqrt{N} & \text{if } x = x_* \end{cases}, \quad x \in \{0, \dots, N-1\}. \quad (155)$$

Thus we have singled out the desired x_* with this step. However, if we were to measure x at this point, we would find any value of x with equal probability since the probability goes as the amplitude squared.

Step 3 amplifies the amplitude of $|x_*\rangle$ relative to all the other $|x\rangle$'s. It does this by applying the gates

$$|\psi_1\rangle \left\{ \begin{array}{c} \xrightarrow{\text{---}^n} \boxed{H^{\otimes n}} \xrightarrow{\text{---}} \boxed{R} \xrightarrow{\text{---}} \boxed{H^{\otimes n}} \xrightarrow{\text{---}^n} \\ \text{---} \end{array} \right\} |\psi'_1\rangle \quad , \quad (156)$$

where R is the operator $2|0\rangle\langle 0| - I$ acting on the first n qubits. It is easy to check that R is unitary (exercise) so it can be implemented as a quantum gate. It is also not hard

Output is the measurement step. It measures the value of x , which is possible since $\{|x\rangle\}$ is a computational basis for the first n qubits. Since the state of the quantum computer is a superposition of $|x\rangle$ for all values of x , this measurement will not give a particular answer with certainty. Instead, it will give different values of x with different probabilities. So, unlike the Deutsch-Josza algorithm, Grover's algorithm does not give the answer with certainty, but just with some probability P , $0 < P < 1$. The claim is that $P > 1 - \frac{1}{N}$. We will show this shortly.

Practically, what this means is that you run the algorithm, then check the answer — ie, evaluate (classically) whether $f(x) = 1$ for the measured output x — and if $f(x) \neq 1$ then re-run the algorithm. Since the probability of a correct answer is substantial (much greater than $1/2$, in this case) you will only need to run the algorithm on average once or twice to get a correct answer. This kind of algorithm is called a (classical or quantum) **probabilistic algorithm**. Since getting the correct answer just requires running the algorithm at most a few times, this does not change the order-of-magnitude estimate of the computational complexity of the algorithm.

So we have the following tasks left in order to understand Grover's algorithm:

Task 1. Implement the R gate with 1- and 2-qubit gates.

Task 2. Show that the $H^{\otimes n} R H^{\otimes n}$ gate acting on an n -qubit state $\sum_x a_x |x\rangle$ gives $\sum_x a'_x |x\rangle$ with a'_x given in (157).

Task 3. Show that repeated applications of $U_f H^{\otimes n} R H^{\otimes n}$ amplifies the amplitude for $|x_*\rangle$ over all other $|x\rangle$'s.

Task 4. Find the maximum amplitude for $|x_*\rangle$ that can be generated in this way.

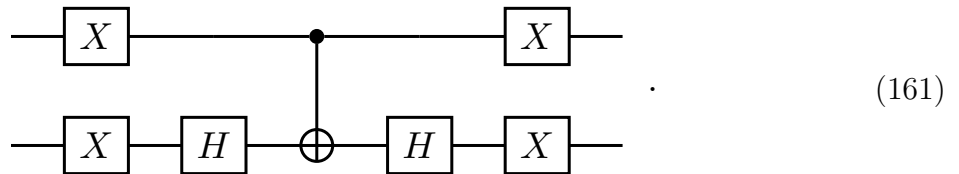
We will tackle these tasks in order.

Task 1: I'll make task 1 easy (on me) by having you do it.

Exercise 2.4 Show that the n -qubit operator $R = 2|0\rangle\langle 0| - I$ is unitary. (Recall that by “ $|0\rangle$ ” here we mean the $|0\rangle^{\otimes n}$ computational basis state.)

This shows that a gate implementing R does in fact exist. We (ie, you) just have to find it now.

Exercise 2.5 Show that for $n = 2$ qubits, R is implemented by the circuit



Exercise 2.6 Generalize the circuit in the last exercise to $n > 2$ qubits.

On to task 2:

Exercise 2.7 Show that $H^{\otimes n}RH^{\otimes n} = 2|\chi\rangle\langle\chi| - I$ where $|\chi\rangle = (1/\sqrt{N})\sum_x |x\rangle$ is the uniform superposition of all computational basis states for n qubits.

We now act on a general n -qubit state with $H^{\otimes n}RH^{\otimes n}$ to find

$$\begin{aligned}
\sum_x a'_x |x\rangle &:= H^{\otimes n}RH^{\otimes n}\left(\sum_x a_x |x\rangle\right) = \sum_x a_x H^{\otimes n}RH^{\otimes n}|x\rangle = \sum_x a_x (2|\chi\rangle\langle\chi| - I)|x\rangle \\
&= \sum_x a_x \left(\frac{2}{N} \sum_{y,z} |y\rangle\langle z| - I\right)|x\rangle = \sum_x a_x \left(\frac{2}{N} \sum_{y,z} |y\rangle\delta_{x,z} - |x\rangle\right) \\
&= \sum_x a_x \left(\frac{2}{N} \sum_y |y\rangle - |x\rangle\right) = 2\frac{1}{N} \left(\sum_x a_x\right) \left(\sum_y |y\rangle\right) - \left(\sum_x a_x |x\rangle\right) \\
&= 2\bar{a} \left(\sum_y |y\rangle\right) - \left(\sum_x a_x |x\rangle\right) = 2\bar{a} \left(\sum_x |x\rangle\right) - \left(\sum_x a_x |x\rangle\right) = \sum_x (2\bar{a} - a_x) |x\rangle.
\end{aligned} \tag{162}$$

Comparing the left and right sides gives the inversion about the mean formula (157).

Exercise 2.8 Justify every step in (162).

Exercise 2.9 Apply (157) to the initial state $|\psi_1\rangle$ given in (149) to find the final state $|\psi_1'\rangle$ given in (150).

Task 3 is more interesting. We saw in step 1 of the algorithm that we start with a state in which the amplitudes of the $|x\rangle$ are all equal, $a_x = 1/\sqrt{N}$. Then in step 2 we flip the sign of only the a_{x_*} amplitude, leaving the rest unchanged. Then in step 3 we invert about the mean which changes the a_x amplitudes for $x \neq x_*$ all in the same way, so they stay equal. Thus the repeated effect of steps 2 and 3 can be simplified by defining the two orthonormal vectors

$$|+\rangle := |x_*\rangle \quad \text{and} \quad |-\rangle := \frac{1}{\sqrt{N-1}} \sum_{x \neq x_*} |x\rangle, \tag{163}$$

which are just the desired state, $|+\rangle$, and the equal superposition of all the undesirable states, $|-\rangle$. Then the effect of the U_f gate is

$$U_f|+\rangle = -|+\rangle, \quad U_f|-\rangle = +|-\rangle, \tag{164}$$

since it just changes the sign of $|x_*\rangle$. You showed in exercise 4.7 that the step 3 gate is $H^{\otimes n}RH^{\otimes n} = 2|\chi\rangle\langle\chi| - I$ where

$$|\chi\rangle = \frac{1}{\sqrt{N}} \sum_x |x\rangle = \frac{1}{\sqrt{N}}|+\rangle + \frac{\sqrt{N-1}}{\sqrt{N}}|-\rangle. \tag{165}$$

A short calculation shows that the effect of the $H^{\otimes n}RH^{\otimes n}$ gate is

$$H^{\otimes n}RH^{\otimes n}|+\rangle = \frac{2-N}{N}|+\rangle + \frac{2\sqrt{N-1}}{N}|-\rangle, \quad H^{\otimes n}RH^{\otimes n}|-\rangle = \frac{2\sqrt{N-1}}{N}|+\rangle + \frac{N-2}{N}|-\rangle. \tag{166}$$

Combining (164) and (166) then shows that the effect of the $H^n R H^n U_f$ gates in each iteration of step 4 of the algorithm is

$$H^n R H^n U_f \begin{pmatrix} |+\rangle \\ |-\rangle \end{pmatrix} = G \begin{pmatrix} |+\rangle \\ |-\rangle \end{pmatrix}, \quad \text{with} \quad G := \frac{1}{N} \begin{pmatrix} N-2 & -2\sqrt{N-1} \\ 2\sqrt{N-1} & N-2 \end{pmatrix}. \quad (167)$$

Exercise 2.10 Verify (166) and (167).

Now we observe a very happy fact: G is a 2-dimensional rotation matrix!

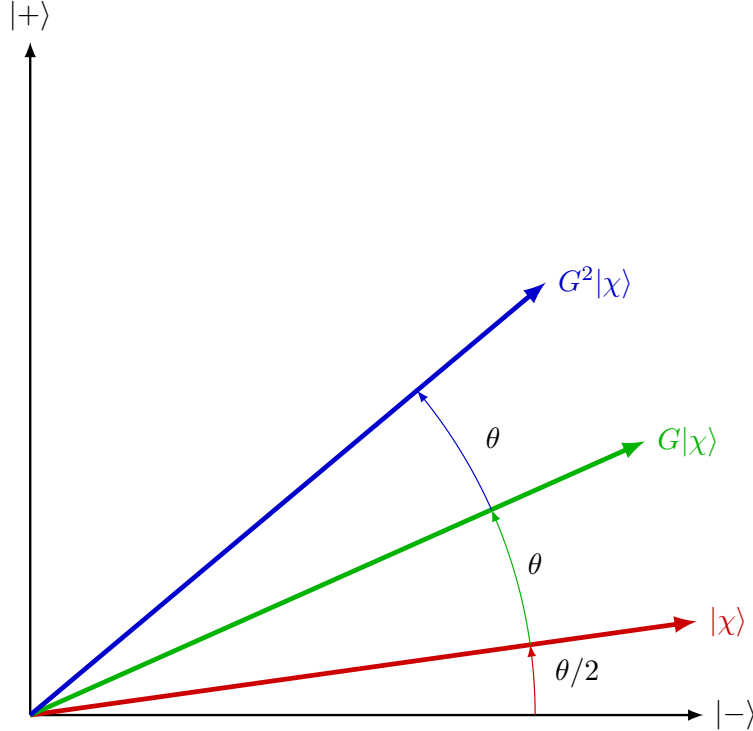
$$G := \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \quad \text{with} \quad \sin \theta := \frac{2\sqrt{N-1}}{N}. \quad (168)$$

Furthermore, the initial vector, $|\psi'_0\rangle$, of step 1 of the algorithm is just the equal superposition of all $|x\rangle$ basis states given in (165), which can be rewritten as

$$|\chi\rangle = \sin \frac{\theta}{2} |+\rangle + \cos \frac{\theta}{2} |-\rangle. \quad (169)$$

Exercise 2.11 Verify (168) and (169), ie, show that $\cos \theta = (N-2)/N$, $\sin \frac{\theta}{2} = 1/\sqrt{N}$, and $\cos \frac{\theta}{2} = \sqrt{N-1}/\sqrt{N}$ if θ is defined by (168).

Thus, in the 2-dimensional plane spanned by $|+\rangle$ and $|-\rangle$ we have the following simple geometrical picture of the action of the iteration step of Grover's algorithm on the n -qubit state:



Each successive iteration of G — ie, step 4 of the Grover algorithm — rotates the state vector by an additional angle θ towards the $|+\rangle$ state, which is the target state, $|x_*\rangle$, whose amplitude we want to amplify.

Task 4, our final task, is now easy. T iterations of the step 4 loop takes the state to $G^T|\chi\rangle$ which is at an angle $(T + \frac{1}{2})\theta$ from the $|-\rangle$ axis. To maximize the $|+\rangle$ amplitude we therefore want this angle to be as close to $\pi/2$ as possible and we therefore should choose

$$T = \text{closest integer to } \left(\frac{\pi}{2\theta} - \frac{1}{2} \right). \quad (170)$$

We can easily determine this T for large N by noting from (168) that $\sin \theta \approx 2/\sqrt{N}$, so the angle θ is small. For small angles $\theta \approx \sin \theta$, so (170) gives $T = \text{closest integer to } (\frac{\pi}{2(2/\sqrt{N})} - \frac{1}{2}) = \lceil \frac{\pi}{4}\sqrt{N} \rceil$ which is the value of T claimed in step 4 of Grover's algorithm, (151).

Note that this value of T is optimal, in the sense that for larger T we start rotating the state vector *away* from the desired $|+\rangle$ (and towards $-|-\rangle$). In fact, we can find a lower bound on the amplitude for $|+\rangle$ at the optimal T by noting from the geometrical picture that at the optimal T the state vector will be within an angle $\theta/2$ of $|+\rangle$. Thus the $|+\rangle$ amplitude will be at least $\cos(\theta/2) = \sqrt{N-1}/\sqrt{N}$, and so the probability of measuring $|+\rangle$ (ie, measuring $x = x_*$) is at least

$$P > \left| \cos \frac{\theta}{2} \right|^2 = 1 - \frac{1}{N}, \quad (171)$$

which was the claimed output behavior of Grover's algorithm.

Part II

Cryptosystems and Shor's Algorithm

In 1995 Peter Shor showed that a quantum computer could be used to factor large integers in polynomial time. In fact, what he showed was a quantum algorithm for computing the “order function” in polynomial time, a problem which is known to be equivalent to that of factoring large integers. No classical algorithm is known for computing the order function in polynomial time. (All known ones take exponential time.) This generated a huge amount of interest in the potential of quantum computing since many public key cryptosystems — encryption methods which currently protect most digital communication from eavesdropping — can be decoded by a third party possessing a polynomial-time factorization algorithm.

The goal of the next three sections is to explain how public key cryptosystems work, and how Shor's quantum algorithm works. The mathematics of cryptosystems is basically that of modular arithmetic, the basic ingredients of which are collected in appendix 3.4, but which can be skipped by the un-interested student. Shor's algorithm is essentially an application of the quantum Fourier transform algorithm, explained in section 4. Section 5 then presents Shor's algorithm and explains how it works in an example. The more technical details needed to rigorously show that the algorithm works as advertised are collected in appendix 5.3 on a (classical) algorithm for computing rational approximants and in appendix 5.4. These appendices can also be skipped by the student satisfied with just a qualitative understanding of the algorithm.

3 Factorization, public key cryptosystems, and the order function

It seems to be a mathematical fact of life that some problems are very hard to solve, but that once the solution is given, it is easy to check that the solution is correct.² It turns out that if a person (“Alice”) has knowledge of the answer of such a problem, she can turn this knowledge into an effective way of exchanging messages (information) with a stranger (“Bob”) (who does not know the answer to the problem) in such a way that any third party (“Carol”) intercepting the message will not be able to read it without first solving the hard problem. This is called public key cryptography, and has become the practical basis for how sensitive (*e.g.*, personal, financial, political) data is transmitted on the web (and with other technologies) today.

The basic trick is to find a way of encoding information using a hard question in such a way that to decode it one needs the answer to the hard question, but such that the encoding procedure is no harder than the process of checking a given answer to the hard question.

²Problems for which it is “easy” to check whether a given answer is correct are known as problems of class **NP**. By “easy” we mean that it takes computational resources (gates, memory, number of steps) which grows at most polynomially in the size of the problem; ie, if it takes N bits to state the problem, then the computational resources needed to check the answer grows as $\mathcal{O}(N^a)$ for some positive exponent a . A problem of size N for which an algorithm solving it exists using computational resources which grow only polynomially in N , is called a problem of class **P**. It is not known whether or not **P** = **NP**.

Given this encoding procedure, Alice can then make publicly available the hard question so that if Bob wants to send her a private message, he just encodes it using Alice's question and sends the result to Alice. Since Alice knows the answer to the hard question, she can decode it easily; but Carol has to do the hard work of solving the hard question.

This section is devoted to describing in some detail a simple and widely-used example of a public key cryptosystem, namely the RSA cryptosystem (named after Rivest, Shamir, and Adleman). It is essentially based on the hard problem of factoring an integer with a large number of digits. Once one has an answer to this problem it is easy to check if it is right: one simply does long division. We then show that the problem of factoring a large integer N is (classically) computationally equivalent to computing the “order function” $\text{ord}(x, N)$ — “the order of $x \pmod{N}$ ”.

In general, cryptosystems can be designed around many different-looking hard-to-solve but easy-to-check problems. In fact, a solution to one of these problems also implies a solution to all others in a wide class of problems. Thus, even though we describe here only the RSA cryptosystem based on the factorization problem, a solution to the factorization problem would imply the ability to “crack” a very large family of public key cryptosystems.

3.1 Factorization

Every positive integer N can be uniquely factorized into its prime factors. We will be interested in integers $N = pq$ which are the product of just two very large primes. We can measure the size of an integer p by the number of its digits, which is approximately³ $\log p$. The factorization of large N is a hard problem in the sense that the fastest known (classical) algorithm for factoring N is the “number field sieve” which takes a typical number of steps $\sim c \exp\{2(\ln N)^{1/3}(\ln \ln N)^{2/3}\}$ where c is some positive constant.

Thus, for example, if $\ln N = 360$ (so $N \sim 10^{156}$), then the number of steps is $\approx 10^{20}$ while if $\ln N = 400$ (so $N \sim 10^{173}$), the number of steps grows to $\approx 10^{21}$. This exponential growth in the number of steps to factorize N is what makes the problem hard: practically, even if your computer could typically factorize 156-digit numbers in a month, then it will take it a couple of years to factorize 173-digit numbers.

(A less sophisticated approach gives an $\exp\{\frac{1}{2} \ln N\}$ estimate of the number of steps. We could simply try to factor N by searching through all primes less than or equal to \sqrt{N} to see if they divide N . By the distribution of primes theorem, there are about $\sqrt{N}/\ln(\sqrt{N})$ primes less than \sqrt{N} . So if we had a list of primes, we would basically have to check on the order of $\sqrt{N}/\ln(\sqrt{N}) = e^{(\ln N)/2}/((\ln N)/2) \sim c \exp\{\frac{1}{2} \ln N - \ln \ln N\}$ long divisions.)

On the other hand, the problem of checking that a given number p is indeed a factor of N is relatively easy: one just needs to use the long division algorithm (basically Euclid's algorithm, reviewed below) to compute the remainder upon dividing N by p . This takes a number of steps which just grows polynomially in the number of digits of N , and in fact are easily and quickly computed on present day desktop computers for numbers with hundreds of

³In a decimal system the number of digits is approximately the log base 10, while in a binary system it is the log base 2. Since $\log_{10}(N) = \log_2(N)/\log_2(10)$ which base is used just gives an overall constant factor. In making estimates for large N , we may ignore these overall factors, and so may not bother to specify the base of the logarithm.

millions of decimal digits (while quick—less than a second—factorization on these machines only occurs up to about 60 decimal digits).

Furthermore, it is easy to generate factorization problems for which you know the answer, but which are beyond the ability of others to solve. For instance, suppose you have enough computing power to factor 60 decimal digit numbers quickly (a few seconds), so that 80 digit factorization takes years. Factorization of random 50 decimal digit numbers on your computer quickly yields many 40+ decimal digit primes. Multiplying two such primes yields an 80+ digit N whose factorization is out of the question for your computer.

Exercise 3.1 Try factorizing $N = 13\,330\,658\,193\,973\,103\,793\,092\,760\,940\,404\,484\,091\,698\,472\,537\,579\,303\,656\,381\,881\,452\,032\,765\,389\,190\,079\,989\,910\,303$ on your desktop until you run out of patience. (For example, the *Mathematica* or *Maple* computer algebra systems have built in integer arithmetic functions with up-to-date factorization algorithms.) Then show that N is divisible by $p = 334\,553\,312\,750\,864\,459\,873\,065\,680\,054\,129\,101\,461\,717$. (It should only take the blink of an eye.) Now generate new 80+ digit N 's which are the product of two 40+ digit primes by the method described above.

(In 2008, the limit on factorization of “general form” numbers on a single PC with a few months of running time was 520 bits, or about 156 decimal digits. With a few dozen PCs this becomes 570 bits, or 171 decimal digits. The 2008 record from RSA challenges was factorization of a 663 bit, or 200 decimal digit, number. Memory, not speed, was the limiting factor.)

3.2 RSA cryptosystem

Let $N = pq$, where p and q are very large primes—*e.g.*, 1000 bit primes. We package the message (usually the key for a conventional cryptosystem) as an integer M with $0 < M < N$ by some conventional method. We choose N so that M is coprime to N . (This is very easy, since N has only the two prime factors p and q , so all but two possible messages shorter than a given N will be acceptable.) We now encrypt M by computing

$$E = M^e \pmod{N}, \quad (172)$$

where e is a randomly chosen number coprime to $\phi(N) = (p-1)(q-1)$. e is called the *encoding key*. Then (we will show that) there is a unique *decoding key* d such that

$$M = E^d \pmod{N}. \quad (173)$$

In fact, we will show that $d = e^{-1} \pmod{\phi(N)}$.

The unusual property of this system is that knowledge of N and e does not imply knowledge of d unless the factorization of $N = pq$ is known, as we will discuss shortly. Thus, if we assume that N is so large that factoring is not feasible, then we have the interesting situation that the encoding and decoding keys e and d are genuinely different. This means, for example, that Alice could post N and e on her website and anyone could send her an encrypted message which only she, possessing d , could decrypt.

[RSA can also be used to send *digital signatures*, whereby Alice sends Bob the message $E = M^d \pmod{N}$ and Bob computes $M = E^e \pmod{N}$ using Alice's *public key* e . If the

message makes sense, then Bob knows that it was sent by someone—*i.e.*, Alice—knowing the secret key d . The message is therefore authentic. When large messages are to be authenticated, then M is the *authenticator* for a conventional *message digest* such as MD15.]

First, let's look at a small example of the RSA recipe: Let $N = pq = 11 \cdot 13 = 143$, so that $\phi(N) = 10 \cdot 12 = 120$. Let the message M be 57, which is coprime to N , and let the encoding key be $e = 37$, which is coprime to $\phi(N)$. Then the decoding key is $d = 13$, since $E = M^e = 57^{37} = 112 \pmod{N}$, and $M = E^d = 112^{13} = 57 \pmod{N}$. Note that d is the inverse of $e \pmod{\phi(N)}$ since $ed = 37 \cdot 13 = 481 = 4 \cdot 120 + 1 = 1 \pmod{\phi(N)}$. This last fact is not a coincidence, as we now explain.

Now let's see what is involved, mathematically, in finding d , given N and e . Combining (172) and (173), we see that $M = M^{ed} \pmod{N}$, from which it follows that $M^{ed-1} = 1 \pmod{N}$ (we chose M coprime to N , so M has an inverse \pmod{N}). Then Euler's theorem tells us that a d which satisfies this is one which satisfies $ed - 1 = \phi(N)$, or, equivalently,

$$d \text{ is the inverse of } e \pmod{\phi(N)}.$$

(The assumption that e was chosen coprime to $\phi(N)$ was to ensure that d exists.) We will see that Euler's algorithm gives an efficient way of computing modular inverses, so the problem of finding d is thus reduced to finding $\phi(N)$ given N .

But it is easy to see that knowledge of N and $\phi(N)$ provides an easy determination of the factors p and q of $N = pq$. For $\phi(N) = (p-1)(q-1) = pq - p - q + 1 = N - p - (N/p) + 1$, so that p is found as the solution of a quadratic equation in terms of N and $\phi(N)$.

Thus, this way of determining d from N and e is computationally as hard as the problem of factorizing N . Since no one knows of any way to find d from N and e without using $\phi(N)$, therefore, as far as we know, knowledge of p and q is required.

3.3 The order function and factorization

In this section we will show how to factor N efficiently on a classical computer if we can compute the *order function*⁴ $\text{ord}(x, N)$ — “the order of $x \pmod{N}$ ”. This will essentially show that factorization and computation of the order function are equally hard problems, computationally. We will see that the order function is a kind of refinement of the Euler function $\phi(N)$, and is closely related to the *discrete logarithm*, the inverse of the discrete exponential.

As far as we know, a classical computer needs about $\exp\{\log N\}$ gates to compute $\text{ord}(x, N)$, *i.e.*, the computation grows exponentially with the input, like the factorization problem. In the next section we will show that there exists an algorithm (Shor's algorithm) whereby a quantum computer can compute the order function with only about $(\log N)^2$ gates, thus showing how the existence of a quantum computer could exponentially speed up the solution to the factorization problem (and thereby render vulnerable classical public key cryptosystems).

Let x be coprime to N . The *order* of $x \pmod{N}$, $k := \text{ord}(x, N)$, is the smallest $k > 0$ such that

$$x^k = 1 \pmod{N}.$$

⁴Also frequently called the *multiplicative order function*.

By Euler’s theorem (reviewed in section 3.4 below) we know that the order exists and does not exceed $\phi(N)$. We will often shorten calling k “the order of $x \pmod{N}$ ” to just “the order of x ” when N is understood from context. Another common usage is to call it “the period of x relative to N ” for reasons we will now explore.

As an example, let’s find $\text{ord}(2, 15)$. The powers of 2 modulo 15 are

n	1	2	3	4	5	6	7	8	9	10	11
2^n	2	4	8	1	2	4	8	1	2	4	8

and so on. Notice that the function $f(n) = 2^n \pmod{15}$ is periodic with period $k = 4$ since $f(n + k) = f(n)$. The order of 2 modulo 15 is this period k —*i.e.*, $\text{ord}(2, 15) = k = 4$. Note that $\phi(15) = 8$, which, in this case, is twice $\text{ord}(2, 15)$.

Exercise 3.2 Find $\text{ord}(2, 13)$ and $\text{ord}(4, 13)$.

Exercise 3.3 Show that for any x coprime to N , $\text{ord}(x, N)$ is the period of the function $f(n) = x^n \pmod{N}$.

The period of a function can be found by taking its Fourier transform. Classically, computing the Fourier transform on a set with N elements takes on the order of $N \log N \sim \log N e^{\log N}$ steps—*i.e.*, exponentially long in the number of input bits $\log N$. Thus the order function (discrete logarithm) and discrete exponentiation bear a similar relationship as factoring and multiplication do: in both cases the first is a hard problem, but the inverse is computationally easy. (We will show that exponentiation in modular arithmetic is computationally easy in subsection 3.4 below.)

By contrast, we will see in the next section that a quantum computer can compute a Fourier transform with only of order $(\log N)^2$ gates, which makes it look hopeful that a quantum computer might be able to compute the order function quickly. The problem with this is that though the quantum computer can compute the Fourier transform quickly, it cannot then access the N elements of the Fourier transform to determine the period! But, by a clever indirect approach it is possible to find the period $\text{ord}(x, N)$ with a number of gates only polynomial in $\log N$. We shall discuss this in detail in section 5.

For the rest of this section, we will simply assume that we have a method of computing $\text{ord}(x, N)$, which we call “an order function oracle”, and will show how it enables us to quickly factor N . Here is the algorithm:

Input: Integer N to be factored.

Output: A nontrivial factor of N .

Steps: 1. Generate a random x in the range $1 < x < N$.

2. Compute $\text{gcd}(x, N) = h$. If $h > 1$, then h is a nontrivial factor of N ; stop.

3. Get $k = \text{ord}(x, N)$ from the order function oracle. If k is odd, return to step 1.

4. Compute $t = x^{k/2} \pmod{N}$. If $t = -1 \pmod{N}$ return to step 1.

5. Compute $h = \text{gcd}(t + 1, N)$. This will be a nontrivial factor of N ; stop.

Steps 1, 2, 4, and 5 are computationally fast (polynomial in $\log N$) because computing the gcd and modular exponential are, as reviewed in subsection 3.4.

It is not hard to see why this algorithm gives a nontrivial factor of N . For if $k = \text{ord}(x, N)$ is even, then $t = x^{k/2} \pmod{N}$ is defined. But from the definition of order, $x^k - 1 = 0 \pmod{N}$, so $t^2 - 1 = (t + 1)(t - 1) = 0 \pmod{N}$. Hence

$$(t + 1)(t - 1) = rN \tag{174}$$

for some r . But then $\text{gcd}(t + 1, N) \neq 1$, since if it were then (174) would imply that $t - 1$ is divisible by N , or $t = x^{k/2} = 1 \pmod{N}$, which would mean that $\text{ord}(x, N) \leq k/2$ in contradiction to the assumption that $\text{ord}(x, N) = k$. Since $\text{gcd}(t + 1, N) \neq 1$, it is a factor of N . But it could still be trivial if $\text{gcd}(t + 1, N) = N$, or equivalently, if $t = -1 \pmod{N}$. This explains the restriction in step 4.

It may be helpful to see this algorithm in action in some examples. Let $N = pq = 11 \cdot 13 = 143$. Choose a random number x in the range $1 < x < N$ with x coprime to N . Say $x = 4$. We use the quantum computer to compute $k = \text{ord}(x, N) = \text{ord}(4, 143) = 30$. (Of course, we don't have a quantum computer; these numbers are very small, so the order is easily computed on a classical computer.) Then we let $t = x^{k/2} = 4^{15} = 12 \pmod{N}$. Then $h = \text{gcd}(t + 1, N) = \text{gcd}(13, 143) = 13$ is indeed a nontrivial factor of N .

With the same N , if we chose instead $x = 10$, though, the algorithm doesn't give us a useful answer, since it fails the condition in step 4. For, $k = \text{ord}(x, N) = \text{ord}(10, 143) = 6$, and $t = x^{k/2} = 10^3 = 142 = -1 \pmod{N}$. Then $\text{gcd}(t + 1, N) = \text{gcd}(143, 143) = \text{gcd}(N, N) = N$, and we have only a trivial factor of N .

Given that just randomly guessing a factor of N is very improbable, the only way this algorithm is successful is for it to reach step 5. For this to happen we need to be lucky in two ways when we choose x :

- (a) $k = \text{ord}(x, N)$ must be even so that $t = x^{k/2} \pmod{N}$ is defined, and
- (b) t must not be $N - 1 = -1 \pmod{N}$.

The 50% theorem states: In the special case when $N = pq$ is the product of two primes, the probability $P(N)$ that a random number x in the range $1 \leq x < N$ satisfies both conditions (a) and (b) above is at least 50 %.

Given this result, and since the probability of the algorithm stopping in step 2 is negligible if the prime factors of N are all large, the probability of reaching step 5 is at least 50% each time we start from step 1. Hence the probability of failing after r times is at most 2^{-r} . So the average number of times through the loop before successfully factoring N is at most $1/(1/2) = 2$.

The 50% theorem is proven in appendix 3.4.

3.4 Appendix: Modular arithmetic

This appendix reviews some elementary concepts from number theory and gives the proof of the 50% theorem.

Greatest common divisors

The *greatest common divisor* of two non-negative integers a, b is denoted $\gcd(a, b)$ and is the largest integer which divides them both. (We define $\gcd(0, a) \equiv a$.) a and b are said to be *coprime*, or equivalently, a is coprime to b (or *vice versa*) if $\gcd(a, b) = 1$.

The gcd can be computed efficiently by *Euclid's algorithm*. This algorithm uses successive long divisions to find a sequence of decreasing remainders. The last positive remainder is the gcd. First let's outline the algorithm, then explain why it works. Suppose we want to find $\gcd(a, b)$ where $a > b$. Then, long division of a by b gives a quotient q_1 and remainder r_1 with $0 \leq r_1 < b$. This can be expressed as $a = bq_1 + r_1$. Euclid's algorithm then says to divide b by r_1 to find a new remainder r_2 and so on:

$$\begin{array}{llll}
 a = bq_1 + r_1 & (0 \leq r_1 < b) & \Rightarrow & \gcd(a, b) = \gcd(b, r_1) \\
 b = r_1q_2 + r_2 & (0 \leq r_2 < r_1) & \Rightarrow & \gcd(b, r_1) = \gcd(r_1, r_2) \\
 r_1 = r_2q_3 + r_3 & (0 \leq r_3 < r_2) & \Rightarrow & \gcd(r_1, r_2) = \gcd(r_2, r_3) \\
 \vdots & \vdots & \vdots & \vdots \\
 r_{n-2} = r_{n-1}q_n + r_n & (0 \leq r_n < r_{n-1}) & \Rightarrow & \gcd(r_{n-2}, r_{n-1}) = \gcd(r_{n-1}, r_n) \\
 r_{n-1} = r_nq_{n+1} + 0 & & \Rightarrow & \gcd(r_{n-1}, r_n) = \gcd(r_n, 0) = r_n.
 \end{array} \tag{175}$$

From the inequalities on the right we see that the r_i form a strictly decreasing sequence, so after at most b steps (typically many fewer: we will find a bound on the number n of steps below) the remainder 0 will appear. The last (smallest) positive remainder, r_n , is the desired gcd.

To see why this is true, note that if $a = bq + r$, then $\gcd(a, b) = \gcd(b, r)$. This is because any number which divides both a and b also divides r , since $r = a - bq$; and conversely, every number which divides b and r also divides a . Therefore the set of all common divisors of a and b is the same as those of b and r , and so their greatest common divisors must be the same.

Now apply this to (175): $\gcd(a, b) = \gcd(b, r_1) = \gcd(r_1, r_2) = \dots = \gcd(r_{n-1}, r_n) = \gcd(r_n, 0) = r_n$, giving the desired answer.

It is not hard to see that Euclid's algorithm is efficient. Suppose a and b are represented by strings of at most L bits; that is, $\log b \leq \log a \lesssim L$. Then none of the quotients q_i or remainders r_i can be more than L bits long, so all computations can be done with L -bit arithmetic. Furthermore, $r_{i+2} \leq r_i/2$.

Proof: Either $r_{i+1} \leq r_i/2$ or $r_{i+1} > r_i/2$. In the first case, since the r_i 's are strictly decreasing, it follows that $r_{i+2} \leq r_i/2$. In the second case, since $r_i = r_{i+1}q_{i+2} + r_{i+2}$, we must have $q_{i+2} = 1$, implying that $r_{i+2} = r_i - r_{i+1} < r_i/2$.

Thus the number n of steps in Euclid's algorithm is bounded above by $n < 2 \log b \sim \mathcal{O}(L)$.

Euclid's algorithm can be refined slightly to give an algorithm for computing integers (positive or negative) A and B such that $Aa + Bb = \gcd(a, b)$. To show this, consider the sequence of remainders in (175). The first can be written $r_1 = a - q_1b = A_1a + B_1b$ (i.e., with $A_1 = 1$ and $B_1 = -q_1$). From the next equation in (175) we then have $r_2 = b - q_2r_1 = b - q_2(A_1a + B_1b) = A_2a + B_2b$. This process can be repeated for the successive remainders until we arrive at $r_n = Aa + Bb$.

Euler totient function

A function which will play an important role in what follows is the *Euler totient function* $\phi(N)$. $\phi(N)$ is defined to be the number of numbers less than N and coprime to N . Thus, if p is prime, $\phi(p) = p - 1$, and if $N = pq$ with p and q prime and not equal, then $\phi(N) = (p - 1)(q - 1)$.

Note that if you know $\phi(N)$, then you know the $N = pq$ factorization (assuming it just has two factors). This is because $N = pq$ implies $p = N/q$, and $\phi(N) = (p - 1)(q - 1) = \left(\frac{N}{q} - 1\right)(q - 1) = N + 1 - q - \frac{N}{q}$ which is a quadratic equation for q given N and $\phi(N)$.

Modular arithmetic

We say that $a = b \pmod{m}$ if m divides $a - b$ (with remainder 0). Equations mod m can be added, subtracted and multiplied. By $b \pmod{m}$ we mean the unique number $B = b \pmod{m}$ such that $0 \leq B < m$.

You have to be a little more careful with modular division. It is only well-defined if the number you are dividing by has a modular inverse. The inverse \pmod{b} of a is defined to be the integer A ($0 < A < b$) such that $Aa = 1 \pmod{b}$. For example, if a is divisible by b —i.e., $a = 0 \pmod{b}$ —then a does not have a modular inverse and you cannot divide by it in modular arithmetic, just as you cannot divide by 0 in integer arithmetic. More generally, the inverse \pmod{b} of a exists if and only if a and b are coprime.

Proof: If the inverse exists, then $Aa = 1 \pmod{b}$ which means that there exists a B such that $Aa + Bb = 1$. As in the proof of Euclid's algorithm, this implies that $\gcd(a, b) = 1$, i.e., that a and b are coprime. The converse follows from Euclid's algorithm, which, as described above, shows the existence of and computes integers A and B such that $Aa + Bb = \gcd(a, b)$. If a and b are coprime, so that $\gcd(a, b) = 1$, then we have $Aa = 1 \pmod{b}$, and so we have an efficient algorithm for computing the inverse \pmod{b} of a as well as a proof of its existence.

The central mathematical object in the RSA cryptosystem is the operation of exponentiation in modular arithmetic,

$$x^k \pmod{m},$$

which is also called the *discrete exponential*. Its value lies in the range $[0, m)$, and it is possible to compute it without at any time dealing with numbers larger than m^2 . We could, for example, compute $x^k \pmod{m}$ recursively as follows:

$$\begin{aligned} x^1 \pmod{m} &= x \pmod{m} \\ \text{for even } k > 1, \quad x^k \pmod{m} &= [x^{k/2} \pmod{m}]^2 \pmod{m} \\ \text{for odd } k > 1, \quad x^k \pmod{m} &= x [x^{k-1} \pmod{m}] \pmod{m}. \end{aligned}$$

It is easy to see that the number of steps in the recursion is $\mathcal{O}(\log k)$.

A key to understanding the properties of the discrete exponential is *Euler's theorem* which states that if a is coprime to n , then

$$a^{\phi(n)} = 1 \pmod{n}.$$

Proof: By definition the number of integers less than and coprime to n is $\phi(n)$. Let $k_1, k_2, \dots, k_{\phi(n)}$ be those numbers. Then the $\phi(n)$ numbers $ak_i \pmod{n}$ must be the same numbers since they are distinct (why?) and coprime to n (why?). Hence

$$\prod_{i=1}^{\phi(n)} (ak_i) = \prod_{i=1}^{\phi(n)} k_i \pmod{n}. \quad (176)$$

But since the k_i are each coprime to n , so is $\prod_i k_i$. Therefore $\prod_i k_i$ has a modular inverse, so we can divide (176) by it to find $a^{\phi(n)} = 1 \pmod{n}$.

A familiar corollary of Euler's theorem is *Fermat's (little) theorem*: Let p be a prime and suppose a is coprime to p . Then $a^{p-1} = 1 \pmod{p}$.

The 50% theorem

We now prove the 50% theorem. The proof is somewhat lengthy since it needs some further results from number theory, but since the techniques used in the proof are basically elementary, we include it here for completeness. The results we need are the Chinese remainder theorem, the notion of primitive roots modulo p , and the primitive root existence theorem. We will briefly explain what these are, and then proceed to the proof of the 50% theorem. The proofs of the Chinese remainder and primitive root existence theorems are included afterwards.

The Chinese remainder theorem states that given coprime m and n , and given a and b , there exists a unique x in the range $0 \leq x < mn$ such that $x = a \pmod{m}$ and $x = b \pmod{n}$. Clearly, given any x , there are a and b such that these two equations are true; hence the converse of the Chinese remainder theorem is also true. This shows, in particular that there is a one-to-one mapping of x 's satisfying the two equations to pairs $(a, b) \pmod{(m, n)}$. This gives rise to a probabilistic way of stating the Chinese remainder theorem: If m and n are coprime, and if a and b are chosen uniformly randomly mod m and mod n respectively, then the unique x such that $x = a \pmod{m}$ and $x = b \pmod{n}$ is distributed uniformly randomly in the set $\{0, 1, \dots, mn - 1\}$.

The primitive roots mod p are those numbers a such that $\text{ord}(a, p) = p - 1$. This implies that if a is a primitive root mod p , then the numbers $\{1, a, a^2 \pmod{p}, \dots, a^{p-2} \pmod{p}\}$ are distinct; hence they must be a permutation of the numbers $\{1, 2, \dots, p - 1\}$. For example: the number $g = 2$ is a primitive root mod p when $p = 13$, as may be seen from the table

n	1	2	3	4	5	6	7	8	9	10	11
2^n	2	4	8	3	6	12	11	9	5	10	7

and 2^{12} is of course $1 \pmod{13}$ by Fermat's little theorem. On the other hand, 4 is not a primitive root, as may be seen from the table

n	1	2	3	4	5	6	7	8	9	10	11
4^n	4	3	12	9	10	1	4	3	12	9	10

The primitive root existence theorem states simply that for every odd prime p , there is a primitive root $g \pmod{p}$.

Given these results, we now turn to

The 50% theorem: Suppose $N = pq$ where p and q are odd primes and $S = \{y | 1 \leq y < N, \gcd(y, N) = 1\}$. Then at least 50% of the integers y in S have even order $k = \text{ord}(y, N)$ and satisfy $y^{k/2} \not\equiv -1 \pmod{N}$.

Proof: Suppose y is in S and has order $\text{ord}(y, N) = r = 2^a h$, where h is odd. Suppose $\text{ord}(y, p) = s = 2^i u$ and $\text{ord}(y, q) = t = 2^j v$, where u and v are odd. Since $N = pq$, it is easy to see that both s and t divide r .

Suppose that r is odd. Since both s and t divide r , they must both be odd. Hence $i = j = a = 0$. Suppose instead that r is even and $y^{r/2} \equiv -1 \pmod{N}$. Then $y^{r/2} \equiv -1 \pmod{p}$, so that $r/2$ is not a multiple of $s = \text{ord}(y, p)$. Since s divides r we must have $i = a$. Similarly $j = a$. To summarize: if $r = \text{ord}(y, N)$ is odd or if r is even and $y^{r/2} \equiv -1 \pmod{N}$, then $i = j = a$.

Let $p - 1 = 2^m x$ where x is odd. From the primitive root existence theorem we know that the nonzero integers mod p are generated by the $p - 1$ powers of some generator g coprime to p . It follows that an integer b will have an odd order with respect to p if and only if b equals g to a power $2^m w$, $1 \leq w \leq x$, and the proportion of such integers is therefore 2^{-m} . Furthermore, precisely those b 's equal to g to a power $2^{m-k} w$, w odd, will have a period with exactly k powers of 2. It follows that w can take odd values from 1 through $2^k x - 1$, and there are precisely $2^{k-1} x$ such integers. That is, the proportion of integers in S whose order with respect to p has exactly k powers of 2 is 2^{k-m-1} .

Now suppose that $p - 1 = 2^m x$ and $q - 1 = 2^n w$ where x and w are odd and $1 \leq m \leq n$. By the probabilistic interpretation of the Chinese remainder theorem, we see that the proportion of integers y in S which have odd period or which have an even period u and $y^{u/2} \equiv -1 \pmod{N}$ is

$$\left(\frac{1}{2}\right)^{m+n} + \sum_{1 \leq k \leq m} \left(\frac{1}{2}\right)^{m+n-2k-2} = \frac{4^m + 2}{2^{m+n} \cdot 3} := f(m, n).$$

The function $f(m, n)$ is greatest when n is as small as possible. Since $1 \leq m \leq n$, we have $f(m, n) \leq f(m, m) = (1 + 2 \cdot 4^{-m})/3 \leq f(1, 1) = 1/2$.

Proof of the Chinese remainder theorem

The proof of the Chinese remainder theorem is simply a computation in modular arithmetic. Since we want $x \equiv a \pmod{m}$, we desire an integer t such that $x = a + tm$. Since we want $x \equiv b \pmod{n}$, we must want $x = a + tm \equiv b \pmod{n}$. Solving this for x , we obtain $tm \equiv b - a \pmod{n}$, or $t \equiv m'(b - a) \pmod{n}$, where m' is the mod n inverse of m . The inverse m' exists because m and n are coprime. Now substituting we obtain $x = a + tm = a + m'm(b - a) \pmod{n}$. To get x in the required range $0 \leq x < mn$, we may have to reduce x modulo mn , so $x = a + m'm(b - a) \pmod{mn}$. Reversing the above steps, we can see that $x \equiv a \pmod{m}$ and $x \equiv b \pmod{n}$.

For example, find x such that $x \equiv 2 \pmod{5}$ and $x \equiv 4 \pmod{7}$. Let $x = 2 + 5t$. Then $2 + 5t \equiv 4 \pmod{7}$, so $5t \equiv 2 \pmod{7}$. But $5^{-1} \equiv 3 \pmod{7}$, so $t \equiv 6 \pmod{7}$. Thus $x = 2 + 5t = 32$. Let's check our solution: $32 \equiv 2 \pmod{5}$ and $32 \equiv 4 \pmod{7}$.

Proof of the primitive root existence theorem

We shall need a number of lemmas and theorems before we can prove this. In particular we will need Lagrange's theorem on the number of solutions to integral polynomial equations mod p , and the property of Euler's function that $\sum_{d|m} \phi(d) = m$.

Lemma: If p is an odd prime, and x is coprime to p , then $\text{ord}(x, p)$ divides $p - 1$.

Proof: Let $s = \text{ord}(x, p)$, and divide $p - 1$ by s to obtain $p - 1 = qs + r$ with $0 \leq r < s$. Then $x^r = x^{p-1}(x^s)^{-q} = 1 \pmod{p}$ by Fermat's little theorem and the definition of s . If $r > 0$ this then implies that $r < s$ is the order of x , contradicting the fact that s is. Therefore $r = 0$, hence s divides $p - 1$.

Lemma: If p is prime and $ab \equiv 0 \pmod{p}$, then either $a \equiv 0 \pmod{p}$ or $b \equiv 0 \pmod{p}$.

Proof: The lemma just says that if p divides ab , then either p divides a or p divides b , which is the definition of prime number.

Lagrange's theorem: If $P(x)$ is a polynomial of degree d with integer coefficients, then the number of solutions to $P(x) \equiv 0 \pmod{p}$ (that are different mod p) does not exceed d if p is prime.

Proof: By induction on d . The result is true for $d = 1$. Let $d > 1$, so $P(x)$ is a polynomial of degree d , and let $P(a) \equiv 0 \pmod{p}$. Then the linear term $x - a$ will divide $P(x)$ exactly. That is, $P(x) = (x - a)Q(x) \pmod{p}$ where $Q(x)$ is a polynomial of degree $d - 1$, and by the induction hypothesis, $Q(x)$ has at most $d - 1$ roots. By the above lemma, $P(x)$ can only be zero if either $x - a$ or $Q(x)$ is zero. Hence $P(x) \equiv 0 \pmod{p}$ has at most d solutions.

Note that the theorem is false if p is not prime. For example $x^2 - 1 \equiv 0 \pmod{15}$ has the four roots 1, 4, 11, and 14.

Lemma: If A and B are positive and coprime, then the AB numbers $m = Ab + Ba$ with $0 \leq a < A$ and $0 \leq b < B$ are distinct modulo AB . Furthermore, if the a 's are confined to the $\phi(A)$ numbers coprime to A and the b 's are confined to the $\phi(B)$ numbers coprime to B , then the resulting $\phi(A)\phi(B)$ numbers are all coprime to AB .

Proof: Suppose that $Ab_1 + Ba_1 \equiv Ab_2 + Ba_2 \pmod{AB}$. Then modulo B we have $Ab_1 \equiv Ab_2 \pmod{B}$, and modulo A we have $Ba_1 \equiv Ba_2 \pmod{A}$. But A and B are coprime, so $b_1 \equiv b_2 \pmod{B}$ and $a_1 \equiv a_2 \pmod{A}$, so $b_1 = b_2$ and $a_1 = a_2$. Hence the numbers $Ab + Ba$ are distinct \pmod{AB} . Furthermore, let $\mu = A\beta + B\alpha$, so $\mu \equiv A\beta \pmod{B}$. If β is coprime to B , then so is $A\beta$ (since A and B are coprime), and hence $\gcd(\mu, B) = \gcd(A\beta + B\alpha, B) = \gcd(A\beta, B) = 1$. Likewise, if α is coprime to A , then so is μ . Therefore, if α is coprime to A and β is coprime to B , we must have that μ is coprime to AB . The lemma follows.

Multiplicative property of ϕ : If $\gcd(A, B) = 1$, then $\phi(AB) = \phi(A)\phi(B)$.

Proof: The $\phi(A)\phi(B)$ numbers μ in the above lemma are coprime to AB and distinct modulo AB . Further, each such μ is equal \pmod{AB} to exactly one integer x in the range $0 < x < AB$. No other of the AB numbers $m = Ab + Ba$ are coprime to AB , for if $\gcd(a, A) > 1$, then m is not coprime to A and therefore not coprime to AB . Similarly, if $\gcd(b, B) > 1$, m is not coprime to AB . The result follows.

Theorem: Let m be written as its prime factorization $m = \prod_p p^c$. Then

$$\phi(m) = \prod_p \phi(p^c) = \prod_p (p - 1)p^{c-1}.$$

Proof: If $m = p^c$ where p is a prime, then the numbers x , $0 < x < m$, that are not prime to m are precisely the multiples of p less than m . There are p^{c-1} of these. Hence $\phi(p^c) = p^c - p^{c-1} = (p-1)p^{c-1}$. The theorem then follows from this and the multiplicative property of ϕ .

Theorem: $\sum_{d|m} \phi(d) = m$. ($d|m$ means d divides m .)

Proof: If $m = \prod p^c$, then the divisors of m are the numbers $d = \prod p^{c'}$, where $0 \leq c' \leq c$ for each p , and

$$\sum_{d|m} \phi(d) = \sum_{p, c'} \prod \phi(p^{c'}) = \prod_p \{1 + \phi(p) + \phi(p^2) + \cdots + \phi(p^c)\},$$

by the multiplicative property of $\phi(m)$. But $1 + \phi(p) + \phi(p^2) + \cdots + \phi(p^c) = 1 + (p-1) + p(p-1) + \cdots + p^{c-1}(p-1) = p^c$, so $\sum_{d|m} \phi(d) = \prod_p p^c = m$.

Theorem: If p is an odd prime, then there are $\phi(p-1)$ primitive elements modulo p .

Proof: Each x , $1 \leq x \leq p-1$, has an order d relative to p which by a previous lemma divides $p-1$. Therefore, if $\psi(d)$ is the number of numbers mod p of order d , then $\sum_{d|p-1} \psi(d) = p-1$, since all x will be counted. But we also have $\sum_{d|p-1} \phi(d) = p-1$ that we proved above. Hence

$$\sum_{d|p-1} \psi(d) = \sum_{d|p-1} \phi(d) = p-1.$$

Now, $\psi(p-1)$ counts the number of primitive elements, so we want to show that $\psi(p-1) = \phi(p-1)$. If we could show that $\psi(d) \leq \phi(d)$ for all d , then it would follow that $\psi(d) = \phi(d)$ for all d , and the theorem would follow.

Suppose $\psi(d) > 0$, then there exists an f such that $\text{ord}(f, p) = d$. Let $f_h = f^h$ for $0 \leq h < d$. Each f_h is a root of the equation

$$x^d = 1 \pmod{p}, \tag{177}$$

since $f^{hd} = 1^h$. The f_h are distinct (mod p), since otherwise $f^h = f^{h'}$ with $h' < h < d$ would imply $f^{h-h'} = 1$ with $0 < h-h' < d$, and thus $\text{ord}(f, p) < d$, a contradiction. Hence by Lagrange's theorem, the f_h 's are all of the roots of equation (177), which means that all the elements of order d are to be found among the f_h 's. Finally, if $\text{ord}(f_h, p) = d$, then $\gcd(h, d) = 1$, since a $k > 1$ dividing both h and d would imply $(f^h)^{d/k} = (f^d)^{h/k} = 1$ and $\text{ord}(f_h, p) \leq d/k < d$. Thus an h such that f_h is of order d must be one of the $\phi(d)$ numbers less than and coprime to d . Therefore $\psi(d) \leq \phi(d)$ and the theorem follows.

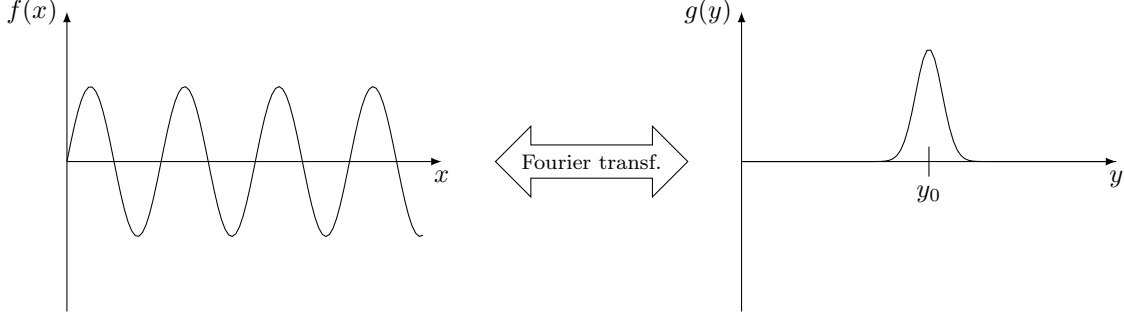
Corollary: If p is an odd prime, then there exists a primitive element g .

4 Quantum Fourier transform

We have seen that the order function basically computes the period of the modular exponential. A basic way to compute the period of a periodic function is to take its Fourier transform. So it should be no surprise that an algorithm for computing the order function will involve computing the Fourier transform. We present here a quantum circuit which implements a version of the Fourier transform on n qubits. Since finding periods has many

other applications beyond factorization, the quantum Fourier transform circuit is interesting in its own right.

The usual Fourier transform has the property that it maps periodic functions — eg, $f(x) \sim \cos(y_0 x)$ or similar — which are non-zero for most values of the variable x , to functions — eg, $\delta(y - y_0)$ — which are concentrated at just a few values of the Fourier transform variable y .



The analog of this behavior for the quantum Fourier transform is that a state which is in an equal superposition with appropriately varying phases of all the computational basis states is transformed to a state which is a single one of the computational basis vectors, and vice versa. We will see how this is useful in our later discussion of the Shor factorization algorithm.

The (classical) Fourier transform is given by

$$g(y) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} dx f(x) e^{ixy}, \quad f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} dy g(y) e^{-ixy}. \quad (178)$$

By plugging one of these expressions into the other, it follows that they are inverses of each other upon using the identity

$$\int_{-\infty}^{\infty} dx f(x) e^{i(y-y')x} = 2\pi \delta(y - y'), \quad (179)$$

or a similar expression with y and x exchanged everywhere. $\delta(y - y')$ is the Dirac delta function, a kind of continuous version of the Kronecker delta symbol, ie, “ $\delta(y - y') \sim \delta_{y,y'}$ ”.

To motivate the definition of the quantum Fourier transform gate, we discretize the classical Fourier transform in (178), so it applies to an n -bit register instead of a continuous function. An n -bit register can take on 2^n different values, which we'll label by indices $j, k \in \{0, 1, \dots, 2^n - 1\}$. Then the discretization consists of the replacements

$$\frac{1}{\sqrt{2\pi}} \int dx \rightarrow \frac{1}{2^{n/2}} \sum_{j=0}^{2^n-1}, \quad f(x) \rightarrow f_j, \quad x \rightarrow \frac{\sqrt{2\pi}}{2^{n/2}} k, \quad (180)$$

and similar substitutions for y and g to give the **discrete Fourier transform**

$$g_k = \frac{1}{2^{n/2}} \sum_{j=0}^{2^n-1} f_j e^{2i\pi jk/2^n}, \quad f_j = \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} g_k e^{-2i\pi jk/2^n}. \quad (181)$$

The discrete analog of (179) which makes the two equations in (181) consistent is the identity

$$\frac{1}{2^n} \sum_{j=0}^{2^n-1} e^{2i\pi j(k-k')/2^n} = \delta_{k,k'}. \quad (182)$$

We can prove this identity by using the fact that if $q^N = 1$, then

$$\frac{1}{N} \sum_{j=0}^{N-1} q^j = \begin{cases} 1 & \text{if } q = 1, \\ 0 & \text{otherwise.} \end{cases} \quad (183)$$

Exercise 4.1 Prove (183) by using the elementary identity $\sum_{j=0}^{N-1} q^j = (q^N - 1)/(q - 1)$ if $q \neq 1$.

Then (182) follows by identifying $q \equiv e^{2i\pi(k-k')/2^n}$, $N \equiv 2^n$, and noting that $q = 1$ if and only if $k = k'$.

This motivates the definition of the quantum Fourier transform gate as a gate, QFT , which acts on the computational basis of an n -qubit Hilbert space as

$$QFT|j\rangle := \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} e^{2i\pi(kj/2^n)} |k\rangle, \quad j \in \{0, 1, \dots, 2^n-1\}. \quad (184)$$

This is just the right hand equation in (181) with the replacements

$$f_j \rightarrow QFT|j\rangle, \quad g_k \rightarrow |k\rangle. \quad (185)$$

As usual, the basis states $|j\rangle$ with $j \in \{0, 1, \dots, 2^n-1\}$ is just a shorthand for the n -qubit basis states $|a_1 a_2 \dots a_n\rangle$ where $j = a_1 a_2 \dots a_n$ in base 2. Explicitly,

$$\begin{aligned} j &:= a_1 a_2 \dots a_n = 2^{n-1} a_1 + 2^{n-2} a_2 + \dots + 2^2 a_{n-2} + 2 a_{n-1} + a_n, \\ k &:= b_1 b_2 \dots b_n = 2^{n-1} b_1 + 2^{n-2} b_2 + \dots + 2^2 b_{n-2} + 2 b_{n-1} + b_n. \end{aligned} \quad (186)$$

We now construct a circuit using only 1- and 2-qubit gates which implements QFT on n qubits. We will see that the total number of gates we need is less than $n(n+2)/2$.

To construct a circuit we need to restate the definition (184) of the Fourier transform gate in terms of the qubit computational basis $|a_1 a_2 \dots a_n\rangle$. This is straightforward except for the factor of $jk/2^n$ appearing in the exponent. Since $e^{2i\pi\ell} = 1$ for ℓ an integer, we only need to evaluate $jk/2^n \bmod 1$ — ie, we only need to keep its fractional part. Note that, using (186),

$$\frac{k}{2^n} = \frac{1}{2} b_1 + \frac{1}{2^2} b_2 + \dots + \frac{1}{2^{n-1}} b_{n-1} + \frac{1}{2^n} b_n \equiv 0.b_1 b_2 \dots b_n, \quad (187)$$

where in the last step we notated the sum as a binary decimal. Then

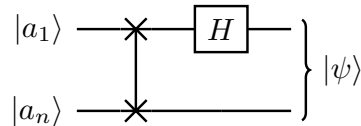
$$\begin{aligned}
j \frac{k}{2^n} &= (2^{n-1}a_1 + \dots + 2^2a_{n-2} + 2a_{n-1} + a_n) \left(\frac{1}{2}b_1 + \frac{1}{2^2}b_2 + \frac{1}{2^4}b_3 + \dots + \frac{1}{2^n}b_n \right) \\
&= \frac{1}{2}a_nb_1 \\
&\quad + \left(\frac{1}{2}a_{n-1} + \frac{1}{2^2}a_n \right) b_2 \\
&\quad + \left(\frac{1}{2}a_{n-2} + \frac{1}{2^2}a_{n-1} + \frac{1}{2^3}a_n \right) b_3 \\
&\quad + \dots \\
&\quad + \left(\frac{1}{2}a_1 + \frac{1}{2^2}a_2 + \dots + \frac{1}{2^n}a_n \right) b_n \\
&= (0.a_n)b_1 + (0.a_{n-1}n_n)b_2 + \dots + (0.a_1 \dots a_n)b_n,
\end{aligned} \tag{188}$$

where we dropped the integer terms in the product. So

$$\begin{aligned}
QFT|a_1 \dots a_n\rangle &= \\
&= \frac{1}{2^{n/2}} \sum_{b_i \in F_2^n} e^{2i\pi[(.a_n)b_1 + (.a_{n-1}a_n)b_2 + \dots + (.a_1 \dots a_n)b_n]} |b_1 \dots b_n\rangle \\
&= \frac{1}{2^{n/2}} \left(\sum_{b_1} e^{2i\pi(.a_n)b_1} |b_1\rangle \right) \otimes \left(\sum_{b_2} e^{2i\pi(.a_{n-1}a_n)b_2} |b_2\rangle \right) \otimes \dots \otimes \left(\sum_{b_n} e^{2i\pi(.a_1 \dots a_n)b_n} |b_n\rangle \right) \\
&= \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2i\pi(.a_n)} |1\rangle \right) \otimes \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2i\pi(.a_{n-1}a_n)} |1\rangle \right) \otimes \dots \otimes \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2i\pi(.a_1 \dots a_n)} |1\rangle \right).
\end{aligned} \tag{189}$$

From this we see that the first output qubit is in a superposition with a phase controlled by the last (a_n) input qubit, and second output qubit is controlled by the last two $(a_{n-1}$ and $a_n)$ inputs, and so on. This indicates that we will need to reverse the order of the input qubits at some point.

Let us focus for the moment on just the first and last qubits, and consider the simple circuit



$$\left. \begin{array}{c} |a_1\rangle \text{---} \text{X} \text{---} \boxed{H} \text{---} \\ |a_n\rangle \text{---} \text{X} \text{---} \end{array} \right\} |\psi\rangle \tag{190}$$

where the first gate is the swap gate defined in (81) which acts on the computational basis by $SWAP|a_1a_n\rangle = |a_na_1\rangle$. Then the output of this circuit is

$$\begin{aligned}
|\psi\rangle &= \frac{1}{\sqrt{2}} \left(|0\rangle + (-)^{a_n} |1\rangle \right) \otimes |a_1\rangle \\
&= \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2i\pi a_n/2} |1\rangle \right) \otimes |a_1\rangle \\
&= \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2i\pi(.a_n)} |1\rangle \right) \otimes |a_1\rangle,
\end{aligned} \tag{191}$$

$$\begin{array}{c}
|a\rangle \text{ --- } \bullet \text{ --- } |a\rangle \\
\quad \quad \quad \downarrow \\
|b\rangle \text{ --- } \boxed{R_j} \text{ --- } e^{2i\pi(ab/2^j)}|b\rangle
\end{array} \quad . \quad (192)$$

$$\begin{array}{c}
|a_1\rangle \text{---} \times \text{---} \bullet \text{---} [H] \text{---} \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2i\pi(\cdot a_n)} |1\rangle \right) \\
|a_2\rangle \text{---} \times \text{---} [H] \text{---} [R_2] \text{---} \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2i\pi(\cdot a_{n-1} a_n)} |1\rangle \right) \\
|a_{n-1}\rangle \text{---} \times \text{---} |a_2\rangle \\
|a_n\rangle \text{---} \times \text{---} |a_1\rangle
\end{array} \quad (193)$$

Now we just repeat this construction for all the other qubits, to get

$$QFT = \text{Diagram} \quad (194)$$

$$\text{number of gates used in } n\text{-qubit } QFT \leq \frac{n(n+2)}{2}. \quad (195)$$

59

5 Shor's quantum algorithm for the order function

We have now reduced the problem of breaking public key encryptions to that of factoring large numbers in “polynomial time” (*i.e.*, in a number of steps polynomial in the number of input bits). And we have reduced the problem of factoring large numbers to that of computing the order function in polynomial time. We will now show how a quantum computer can achieve this by presenting an algorithm due to Peter Shor.

5.1 Shor's algorithm

We will describe Shor's algorithm for a quantum computer operating on a $2n$ -qubit state space, where n will be chosen below. This machine therefore has a Hilbert space of dimension $2^{2n} = 2^n \cdot 2^n$. We will think of this Hilbert space as the tensor product of two Hilbert spaces each of dimension $q = 2^n$. Choose an arbitrary orthonormal basis of each of these spaces which we label by integers $a \pmod{q}$ —*i.e.*, the basis is $\{|a\rangle\}$ with $a = 0, 1, \dots, q-1$. Thus the general basis state of the quantum computer is $|a\rangle|b\rangle$, and a general state of the computer is $|\psi\rangle = \sum_{a,b=0}^{q-1} C_{ab}|a\rangle|b\rangle$ for some coefficients C_{ab} . Assume the machine is initially prepared in the state $|\psi_0\rangle = |0\rangle|0\rangle$.

Shor's algorithm

Inputs: x, N with $\gcd(x, N) = 1$. Also, a small positive integer T .

Output: $r = \text{ord}(x, N)$, *i.e.*, the least positive r such that $x^r = 1 \pmod{N}$.

Steps: 1. Choose a $q = 2^n$ so that $N^2 < q \leq 2N^2$.

2. Put the machine in the uniform superposition of states with all values of $a \pmod{q}$ in the first factor (leaving the second factor alone):

$$|\psi_2\rangle = \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle|0\rangle.$$

3. Next compute $x^a \pmod{N}$ in the second factor:

$$|\psi_3\rangle = \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle|x^a \pmod{N}\rangle. \quad (196)$$

4. Now perform a Fourier transform, mapping the first factor $|a\rangle \rightarrow |c\rangle$ with amplitude $q^{-1/2} \exp\{2\pi iac/q\}$, that is, mapping $|a\rangle \rightarrow \sum_c (1/\sqrt{q}) e^{2\pi iac/q} |c\rangle$:

$$|\psi_4\rangle = \frac{1}{q} \sum_{a=0}^{q-1} \sum_{c=0}^{q-1} e^{2\pi iac/q} |c\rangle|x^a \pmod{N}\rangle. \quad (197)$$

5. Measure the state with respect to the $|a'\rangle|b'\rangle$ basis. This returns a value for both c and $x^a \pmod{N}$ from the state $|\psi_4\rangle$.
6. Round the fraction c/q to the nearest fraction, d/r , (in lowest terms, *i.e.*, $\gcd(d, r) = 1$) having a denominator $r < N$ and satisfying

$$\left| \frac{c}{q} - \frac{d}{r} \right| \leq \frac{1}{2r^2}. \quad (198)$$

7. Compute $h = x^r \pmod{N}$. If $h \neq 1$ return to step 2; otherwise continue.
8. Store the value of r . If there are T values of r stored then continue to step 9; otherwise return to step 2.
9. Choose the least of the stored values of r . It is then $\text{ord}(x, N)$ with probability $P > 1 - (.6)^T$.

Note that steps 1, 6, 7, 8, and 9 are classical, while steps 2, 3, 4, and 5 are the quantum ones. Some comments on the steps:

1. Since $x < N$, the size of the input (*i.e.*, the total number of bits in x and N) is at most $\sim 2 \log N$. In step 1, note that since $q \sim N^2$, the number of bits of q is also $n = \log q \sim 2 \log N$. In particular, the number of qubits $2n$ scales linearly with $\log N$. The small integer T will control the total number of times to repeat the algorithm. The larger T , the more confidence the returned result is correct. (In practice $T = 1$ or 2 is probably ample, as we will discuss later.)
2. Step 2 is the preparation step for the quantum computer. We have seen in earlier lectures how this can be done with a number of quantum gates polynomial in n (or $\log N$) (*e.g.* by applying n Hadamard gates to the first register prepared in the state $|0\rangle|0\rangle$).
3. The computation of $x^a \pmod{N}$ can be done classically with a number of gates polynomial in $\log N$, so it can also be done quantumly in the same way. (Since we are keeping x and N fixed, this can in fact be done reversibly, as can the next step.)
4. A quantum circuit using a polynomial number of gates to perform the Fourier transform was described in previous lectures. It can be done using at most $n(n+4)/2$ gates.
5. In the measurement step, it is actually sufficient to measure just the value of c , but for clarity we will assume that we also measure $x^a \pmod{N}$.
6. Step 6 is a classical computation that can be done in polynomial time using a continued fraction algorithm which we will describe below.
7. Step 7 is a classical computation that can be done in polynomial time and was described earlier. It simply checks whether r is possibly $\text{ord}(x, N)$, and discards it and starts over if it is not.

8. If r passes the test in step 7, then $x^r = 1 \pmod{N}$. But this does not mean that $r = \text{ord}(x, N)$, since the order of x is the *smallest* r satisfying $x^r = 1 \pmod{N}$. In particular, if $r = \text{ord}(x, N)$, then any multiple $r' = mr$ also satisfies $x^{r'} = 1 \pmod{N}$. So step 8 just says to repeat the algorithm until T such possible r 's have been found.
9. The probability bound in step 9 is very crude. As we will discuss below, in practice we can expect virtual certainty with just $T = 1$ or 2 .

Note that steps 7, 8, and 9 essentially just enforce that r will be $\text{ord}(x, N)$ (with some probability which increases for increasing T), so it is clear that this algorithm will eventually yield $\text{ord}(x, N)$. The question is therefore all in how fast (how many steps on average) it takes for it to terminate. It is easy to see that if the value of r returned after step 6 is completely random, then it has only a $1/N$ chance of being $\text{ord}(x, N)$, and so the algorithm would have to run on the order of $N = e^{\log N}$ loops to find the answer, and so would be no advantage over a classical computation.

So the key lies in understanding why the value of r returned after the quantum computation and step 6 has a much higher probability than random chance of being $\text{ord}(x, N)$. We will show how to compute this probability in appendix 5.4 below, and thus assess the time it takes this algorithm to run.

But before we undertake this analysis, it may be helpful to run the algorithm on an example (x, N) to get some intuition for how and why it works. In order to do that, though, we first have to make the classical continued fraction algorithm used in step 6 explicit.

5.2 Example: quantum computation of $\text{ord}(10, 21)$.

Now we are ready to try Shor's algorithm on an example. Say we were trying to use a quantum computer to factor the number 21. Then, following the factorization algorithm (section 3.3), we would want it to compute $\text{ord}(x, 21)$ for some random x coprime to 21. To be definite, let's choose $x = 10$.

Step 1 of Shor's algorithm says to choose $q = 512 = 2^9$ since that is the power of 2 between $N^2 = 21^2 = 441$ and $2N^2 = 882$.

Steps 2 and 3 put the machine in the state (196) where $x^a \pmod{N} = 10^a \pmod{21}$ are computed by the machine to be

a	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	...
10^a	1	10	16	13	4	19	1	10	16	13	4	19	1	10	16	13	4	19	...

Clearly $\text{ord}(10, 21) = 6$. But we can't extract this information directly from the state (196) because by trying to measure the values of $x^a \pmod{N}$ in the second factor, we disturb the state (project it onto the measured value).

So we proceed indirectly, performing the Fourier transform in step 4, giving the state (197). The next step is simply to observe this state by projection to the $|a'\rangle|b'\rangle$ basis, thus measuring the pair of integers c and $x^a \pmod{N}$. So let's compute the probability of measuring any given pair of values. From the table above, we see that the only values of $x^a \pmod{N}$ that will be observed are $\{1, 4, 10, 13, 16, 19\}$, while any possible value of c from 0 to $q - 1 = 511$ seems possible. So if we measured 1 in the second register, then all terms

in sum over a in (197) with $x^a \pmod{N} = 1$ will contribute. But, from the table above, these are all a 's which are multiples of 6, $a = 6m$. And similarly, the amplitude to observe 10 sums over $a = 6m + 1$, to observe 16 has $a = 6m + 2$, to observe 13 has $a = 6m + 3$, to observe 4 has $a = 6m + 4$, and to observe 19 has $a = 6m + 5$. Thus the resulting probabilities of observing c in the first register and 1 in the second, and so forth, are

$$\begin{aligned} P(c, 1) &= \left| \frac{1}{512} \sum_{m=0}^{85} e^{2\pi i(6m)c/512} \right|^2, & P(c, 4) &= \left| \frac{1}{512} \sum_{m=0}^{84} e^{2\pi i(6m+4)c/512} \right|^2, \\ P(c, 10) &= \left| \frac{1}{512} \sum_{m=0}^{85} e^{2\pi i(6m+1)c/512} \right|^2, & P(c, 13) &= \left| \frac{1}{512} \sum_{m=0}^{84} e^{2\pi i(6m+3)c/512} \right|^2, \\ P(c, 16) &= \left| \frac{1}{512} \sum_{m=0}^{84} e^{2\pi i(6m+2)c/512} \right|^2, & P(c, 19) &= \left| \frac{1}{512} \sum_{m=0}^{84} e^{2\pi i(6m+5)c/512} \right|^2. \end{aligned}$$

In each case the m -independent term in the exponent can be dropped since it factors out of the sum and has magnitude 1. Thus we find that, independent of what is measured in the second register, the probability of measuring c in the first register (given that we have also measured the second register) is

$$P(c) = 6 \left| \frac{1}{512} \sum_{m=0}^{84 \text{ or } 85} e^{i\pi 3mc/128} \right|^2.$$

Actually there are two functions, differing slightly by whether the sum goes to $m = 84$ or $m = 85$. A plot of either function looks like figure 1. It is basically zero except for four

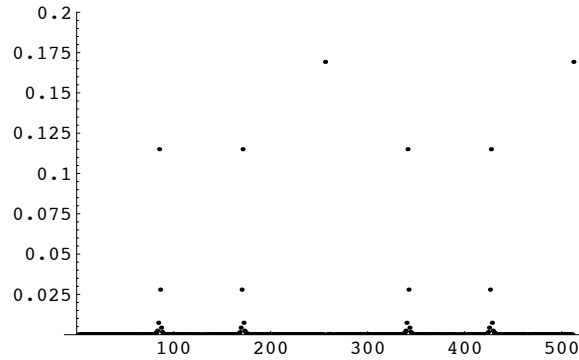


Figure 1: $P(c)$ plotted for values $c = 1, 2, \dots, 512$.

sharp spikes at $c = 85, 171, 341$, and 427 , and two isolated high points at $c = 256$ and 512 (or 0). This is what we expected: we are basically taking the Fourier transform of a periodic function with period 6, so we should find 6 equally-spaced peaks. But since 512 isn't exactly divisible by 6, the peaks can't be perfectly sharp: they actually have some width. In fact, because of this discrete mismatch, the probability never quite vanishes for any of the values of c . This can be more easily seen by plotting the logarithm of $P(c)$, as in figure 2. Here we can see the slight differences between the probability distributions for the sum to $m = 84$ versus the sum to $m = 85$ cases.

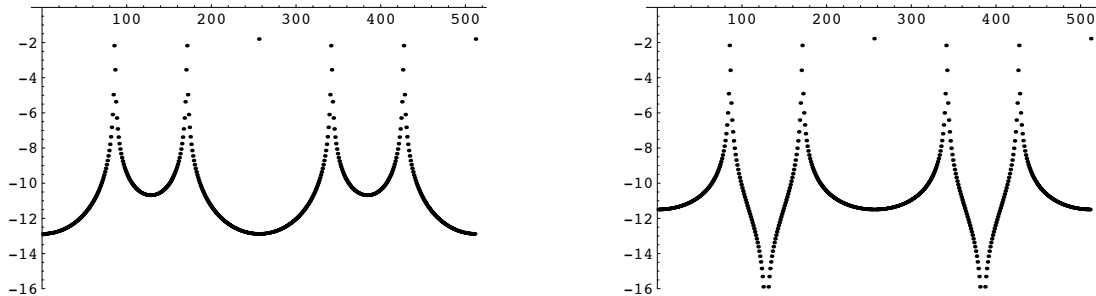


Figure 2: $\text{Log}[P(c)]$ for the 84- and 85-term expressions.

The important thing to notice, though, is clear from figure 1: the probability is exponentially dominated by just a handful of values. In particular, for this example, $c = 0$ or 256 each have probability 17%; $c = 85, 171, 341$, or 427 each have probability 11.5%; $c = 86, 170, 342$, or 426 each have probability 2.8%; $c = 84, 172, 342$, or 428 each have probability 0.7%; and all other values of c combined have a total probability of 6.0%. In fact, we will see below that Shor’s algorithm can fail (*i.e.*, give a wrong answer for the order—recall that the algorithm can fail only by returning a number that is a multiple of the correct $\text{ord}(x, N)$) only for a relatively small set of c ’s which are away from these probable values. This gives Shor’s algorithm a very high probability (more than 99.8% in this example, as we will compute below) of success just taking $T = 1$ in step 8.⁵ But before getting to step 8 of the algorithm, the answer has to pass step 7, which checks if the returned value of r from step 6 satisfies $x^r = 1 \pmod{N}$. It may be that some of the “good” c ’s fail this step.

To see whether and why that happens, let’s now sample some values of c , and run the rest of the algorithm.

The most likely values are $c = 0$ or $c = 256$. These then give $c/q = 0$ or $1/2$. Their closest rational approximations with denominators less than $N = 21$ are just themselves, thus giving $r = 1$ or 2 . Neither of these is the right answer, and indeed step 7 throws them out.

The next most likely are $c \in \{85, 171, 341, 427\}$. Say we measure $c = 427$. Step 6 says to find the nearest fraction d/r to $c/q = 427/512$ having denominator $r < N = 21$. So we run the continued fraction algorithm on c/q finding $427/512 = [0; 1, 5, 42, 2]$. Thus the list of convergents is $\{[0; 1] = 1, [0; 1, 5] = 5/6, [0; 1, 5, 42] = 211/253, [0; 1, 5, 42, 2] = 427/512\}$. Only the first two have denominators less than $N = 21$. Both obey the inequality (198):

$$\begin{aligned} 1/2 = 1/(2 \cdot 1^2) = 1/(2r^2) &\geq |(d/r) - (c/q)| = |(1/1) - (427/512)| = 85/512, \\ 1/72 = 1/(2 \cdot 6^2) = 1/(2r^2) &\geq |(d/r) - (c/q)| = |(5/6) - (427/512)| = 1/1536, \end{aligned}$$

but the second is the closer approximation, so step 6 returns $d/r = 5/6$, or $r = 6$. This passes the check at step 7, and, indeed, is the right answer. The same would happen if $c = 85$.

⁵ This is much higher than the lower bound on the rate of success claimed earlier in step 9 of Shor’s algorithm. This discrepancy is because we will be making only very weak lower bounds on the probabilities in section 5.4. More work can sharpen these bounds considerably, showing that for practical purposes we need only ever take $T = 1$ or 2 .

But if $c = 171$ or 341 , one finds *e.g.*, $171/512 = [0; 2, 1, 170]$ giving the convergents $[0; 2] = 1/2$ which fails (198) and $[0; 2, 1] = 1/3$ which passes (198). Thus step 6 returns $r = 3$. This is again the wrong answer, and is thrown out by step 7.

We can continue to less likely values of c . The result, as should be becoming clear, is that nearby values to $c = 85$ or 427 return the correct answer, whereas values around the other four peaks return the wrong values $r = 2$ or 3 , which are rejected in step 7. How many values around 85 or 427 work? To work, they have to be near enough so that $d/r = 1/6$ or $5/6$ are their best rational approximations satisfying (198). Multiply this relation through by $q = 512$ with $d/r = 1/6$ to get $|c - 85.3| = q/(2r^2) = 512/(2 \cdot 6^2) = 7.1$, and similarly for $d/r = 5/6$. This tells us that only the c 's in the ranges 79 – 92 and 420 – 433 , inclusive, return the correct answer. Summing up the probability of observing these values of c gives 32.7% .

All other values of c give the wrong r at step 6. Of these all are eliminated in step 7 except for those which are multiples of the right r , $r = 6$. Since the algorithm only returns values of $r < N$, the only multiples which could occur are $r = 12$ and $r = 18$. Which values of c could give these wrong answers? Ones for which the closest rational approximation to c/q satisfying (198) have denominator 12 or 18 . These are the fractions $1/12, 5/12, 7/12, 11/12, 1/18, 5/18, 7/18, 11/18, 13/18$, and $17/18$, which correspond to c 's close to $42.7, 213.3, 298.7, 469.3$, for $r = 12$, and $28.4, 142.2, 199.1, 312.9, 369.8, 483.6$ for $r = 18$. The allowed range around these values, by the same reasoning as above, are $\pm q/(2r^2)$ which is ± 1.8 for $r = 12$ and ± 0.8 for $r = 18$. Thus the only values of c which give $r = 12$ or 18 are $c \in \{28\text{--}29, 41\text{--}44, 142\text{--}143, 199, 212\text{--}215, 297\text{--}300, 313, 369\text{--}370, 468\text{--}471, 483\text{--}484\}$. Summing the probabilities for observing these values of c gives just under 0.04% .

Thus, in summary, we have found that step 6 gives the correct r about 33% of the time, an incorrect r which is rejected at step 7 about 67% of the time, and an incorrect r which passes step 7 and therefore gives a wrong answer less than 0.04% of the time. Therefore, on average we have to do the quantum part of the computation 3 times to get an answer that is not rejected at step 7, and of those answers, $3 \cdot (0.04\%) \approx 0.1\%$ are wrong.

We will see in section 5.4 that this basic behavior holds true for arbitrarily large N : on average, the quantum computation will have to be repeated a small number of times (a number which only grows with N as $\log \log N$), and will return a wrong answer only a small percentage of the time. So the correct answer can be found with high confidence by repeating the whole algorithm on a small number, T , times.

Let us emphasize the basic reasons why Shor's algorithm works: (1) The order is the period of the discrete exponential, and the quantum Fourier transform gives a probability distribution which is very sharply peaked at multiples of the period, giving a good probability of observing values near these multiples. (2) And we only need to observe values *near* the multiples because we know that the period is less than N , and so can deduce it by approximating to the closest rational with such a denominator. (3) A substantial fraction of wrong answers can be returned, especially if the correct period itself has many different prime factors, but these can be easily and almost completely screened out since the discrete exponential is easily calculated.

5.3 Appendix: Rational approximant algorithms

This algorithm provides a way of approximating any given rational number, ϕ , by other rationals.

If $\phi = d_0/r_0$ in lowest form ($\gcd(r_0, d_0) = 1$), then divide d_0 by r_0 using the long division algorithm to find $d_0 = d_1 r_0 + r_1$, so $\phi = d_1 + (r_1/r_0) = d_1 + 1/(r_0/r_1)$. Now divide r_0 by r_1 to find $r_0 = d_2 r_1 + r_2$. Thus $\phi = d_1 + 1/(d_2 + 1/(r_1/r_2))$. Clearly this can be continued by now dividing r_1 by r_2 , and so on. At the i th step, we have $r_{i-2} = d_i r_{i-1} + r_i$ and

$$\phi = d_1 + 1/(d_2 + 1/(d_3 + \cdots + 1/(d_{i-1} + 1/(d_i + (r_i/r_{i-1}))) \cdots)).$$

This expansion terminates for any rational ϕ since the series of remainders $\{r_i\}$ is strictly decreasing. If d_0 and r_0 are both $(\log N)$ -bit integers, then the continued fraction expansion can be computed using on the order of $(\log N)^3$ operations—that is, $\log N$ steps each requiring $(\log N)^2$ operations to do the arithmetic.

If you truncate a continued fraction expansion by simply dropping the remainder terms at, say, the i th step,

$$\phi_i := d_1 + 1/(d_2 + 1/(d_3 + \cdots + 1/(d_{i-1} + 1/(d_i)) \cdots)), \quad (199)$$

one gets a series of rational numbers ϕ_i which approximate ϕ more closely as i increases. These truncations, ϕ_i , are called *convergents* of the continued fraction. Since the continued fraction terminates for rational ϕ , we consider ϕ itself as a convergent.

A convenient and standard notation for continued fractions is as a bracketed list of the d_i . Thus (199) will be written

$$\phi_i = [d_1; d_2, d_3, \cdots, d_i].$$

Note that the continued fraction algorithm implies that while d_1 may be an arbitrary integer (positive, negative, or zero), the rest of the d_i are all positive integers. Note also that there is a slight ambiguity in this notation for finite continued fractions, since $[d_1; d_2, \cdots, d_{i-1}, d_i] = [d_1; d_2, \cdots, d_{i-1}, d_i - 1, 1]$ if $d_i > 1$. This follows easily from (199). This ambiguity can be removed by using only the shorter form for the continued fraction; the last d_i in such continued fractions will always be greater than 1.

The useful fact (which we will not prove) about the convergents ϕ_i is that they generate all the *best rational approximations* to ϕ . A best rational approximation to ϕ is a rational number p/q , ($q > 0$, $\gcd(p, q) = 1$) that is closer to ϕ than any approximation with a smaller denominator. The algorithm for listing all the best rational approximants to ϕ is:

1. Use the continued fraction algorithm to form the list of all convergents to ϕ . (This is a finite list if ϕ is rational.)
2. Form new continued fractions from each convergent $[d_1; d_2, \dots, d_i]$ by replacing the last term d_i with all possible d'_i such that $d_i > d'_i \geq d_i/2$.
3. The set of finite continued fractions found from combining those generated in steps 1 and 2 include all the best rational approximations to ϕ . (The ones with $d'_i = d_i/2$ may not be best rational approximations; there is a special rule for this case, but it is complicated so we'll ignore it here.)

Furthermore, if you arrange this list of finite continued fraction in “alphabetical” order of increasing strings of $\{d_i\}$ ’s, then, when reexpressed as fractions $\phi_i = p_i/q_i$ in lowest terms, they will be in order of strictly increasing denominators q_i .

An example will illustrate this. $\phi = 27/32$ has continued fraction $[0; 1, 5, 2, 2]$. So the list of its best rational approximations is

ϕ_i	$[0;1]$	$[0;1,3]$	$[0;1,4]$	$[0;1,5]$	$[0;1,5,1]$	$[0;1,5,2]$	$[0;1,5,2,1]$	$[0;1,5,2,2]$
p_i/q_i	1	3/4	4/5	5/6	6/7	11/13	16/19	27/32
$ \phi - \phi_i \approx$	0.16	0.09	0.04	0.010	0.013	0.0024	0.0016	0
$1/(2q_i^2) \approx$	0.5	0.03	0.02	0.014	0.010	0.0030	0.0014	0.0005

Here we’ve listed the continued fraction form on the first line, the rational form on the second, the approximate value of the difference from ϕ on the third. On the last line, for later use, we’ve also listed the approximate value of $1/(2q_i^2)$. Note that in this list $[0; 1, 5, 1] = 6/7$ is not a best rational approximation since it is further from ϕ than $[0; 1, 5] = 5/6$ which has a smaller denominator. This is an example of a case where $d'_i = d_i/2$ and should be discarded. Thus, from this list and the theorem stating that these are *all* the best rational approximants to ϕ , we can immediately read off the answer to questions like: What is the best rational approximation to $27/32$ with denominator less than 17? Answer: 11/13.

Recall that step 6 of Shor’s algorithm requires finding the closest rational d/r to $\phi = c/q$ with $r < N$. Clearly this question is answered by the continued fraction/best rational approximant algorithm we have just described, and we have seen above that it can be done in polynomial time.

But step 6 has a further condition, namely that the best rational approximation d/r to c/q must also satisfy (198). This is a strong condition which actually simplifies the problem.

To see this, observe that if p and q are coprime integers such that

$$\left| \frac{p}{q} - \phi \right| \leq \frac{1}{2q^2}, \quad (200)$$

then p/q is a best rational approximation to ϕ . This is because the closest another rational p'/q' (not equal to p/q) could be to p/q is $1/qq'$ as shown by putting them over a common denominator. Thus for such rationals with $q' \leq q$, the closest they can come to p/q is always greater than $1/q^2$ and so would violate the inequality (200). Thus all p/q satisfying (200) can be found by the continued fraction/rational approximant algorithm.

Note, however, that the converse to the last paragraph is not true: there can be best rational approximants which do not obey (200). Indeed, comparing the third and fourth lines of the table in the $\phi = 27/32$ example, we see that though $3/4$, $4/5$, and $16/19$ are best rational approximations, they do not satisfy (200).

Finally, there is a theorem (which we will not prove) that the best rational approximants satisfying (200) are convergents of ϕ —though not all convergents satisfy (200). Thus, step 6 of Shor’s algorithm is reduced to just looking through the convergents to ϕ (instead of having to look through all the best rational approximations) and checking which of them with denominator less than N satisfy (200).

5.4 Appendix: The behavior of Shor's algorithm

We compute the probability $P(c, k)$ that we measure the machine to be in the particular state $|c\rangle|x^k \pmod{N}\rangle$ for general x and N . We may assume $0 \leq k < r = \text{ord}(x, N)$. Summing over all possible ways of reaching this state we find

$$P(c, k) = \left| \frac{1}{q} \sum'_a \exp\left(\frac{2\pi i a c}{q}\right) \right|^2$$

where the sum is only over those a , $0 \leq a < q$ such that $x^a = x^k \pmod{N}$. Because $\text{ord}(x, N) = r$, this sum is equivalently over all a satisfying $a = k \pmod{r}$. Writing $a = br + k$, we find that

$$P(c, k) = \left| \frac{1}{q} \sum_{b=0}^{\lfloor (q-k-1)/r \rfloor} \exp\left(\frac{2\pi i (br + k)c}{q}\right) \right|^2.$$

We can ignore the $\exp(2\pi i k c/q)$ factor, as it can be factored out of the sum and has magnitude 1. We can also replace rc with $\{rc\}_q$ where $\{rc\}_q = rc \pmod{q}$ and is in the range $-q/2 < \{rc\}_q \leq q/2$. This leaves us with the expression

$$P(c, k) = \left| \frac{1}{q} \sum_{b=0}^{\lfloor (q-k-1)/r \rfloor} \exp\left(\frac{2\pi i b \{rc\}_q}{q}\right) \right|^2.$$

Since $q > N^2$ while $k < r < N$, the upper limit of the sum is approximately approximated q/r . If $\{rc\}_q$ is small compared to q then changing variables to $t = b/q$, the sum can be further approximated by the integral

$$P(c, k) \sim \left| \int_0^{1/r} dt e^{2\pi i \{rc\}_q t} \right|^2 = \frac{1 - \cos[2\pi \{rc\}_q / r]}{2\pi^2 |\{rc\}_q|^2} \quad \text{if } |\{rc\}_q| \ll q. \quad (201)$$

Using the fact that $1 - \cos x \geq 2x^2/\pi^2$ if $|x| \leq \pi$, this implies the lower bound

$$P(c, k) \geq \frac{4}{\pi^2 r^2} \quad \text{if } |\{rc\}_q| \leq \frac{r}{2}. \quad (202)$$

Note that $|\{rc\}_q| \leq r/2$ automatically ensures the restriction $|\{rc\}_q| \ll q$ needed in the approximation (201).

Now, the restriction $-\frac{r}{2} \leq \{rc\}_q \leq \frac{r}{2}$ is satisfied when there is a d such that $-\frac{r}{2} \leq rc - dq \leq \frac{r}{2}$. Dividing by rq gives

$$\left| \frac{c}{q} - \frac{d}{r} \right| \leq \frac{1}{2q} \leq \frac{1}{2N^2} \leq \frac{1}{2r^2}, \quad (203)$$

since $q > N^2$ and $N > r$. We recognize this to be a necessary condition for d/r to be a best rational approximation to c/q , and hence there is at most one fraction d/r with $r < N$ satisfying the above inequality. We have discussed earlier how a continued fraction expansion of c/q can then determine d/r in polynomial time.

If we have the fraction d/r in lowest terms, and if d happens to be coprime to r , this will give us r , and the algorithm returns the correct answer. We will now count the number of states $|c\rangle|x^k \pmod{N}\rangle$ which enable us to compute r in this way. There are $\phi(r)$ possible values for d coprime to r . The condition for (202) to hold is the first inequality in (203), which when multiplied by q gives $|c - (qd/r)| \leq (1/2)$. This implies there is one value of c for each d . There are also r possible values for $x^k \pmod{N}$ since $r = \text{ord}(x, N)$. Thus there are $r\phi(r)$ states which would enable us to obtain r .

Since we have seen in (202) that each of these states occurs with probability at least $4/(\pi^2 r^2)$, we obtain r with probability at least $4\phi(r)/(\pi^2 r)$. Recall that $\phi(r)$ is just the number of numbers less than r which are coprime to r . It is a deep arithmetic fact (related to the average distribution of primes) that $\phi(r)$, though less than r , grows almost as fast as r for large r . In particular, it is a theorem (which we will not prove) that

$$\frac{\phi(r)}{r} > \frac{C}{\log \log r}$$

for some positive constant C . This shows that we find r at least a $C/\log \log r$ fraction of the time. Since $r < N$, this means that by repeating the measurement only on order $\mathcal{O}(\log \log N)$ times, we are assured of a high probability of success.

But, as we have discussed above, the algorithm can also give a false positive if it returns an r' which is a multiple of the correct r , i.e., $r' = mr$ with $m > 1$. This can never happen for the c 's satisfying the condition in (202) since as we have seen this condition assures that the closest rational approximation to c/q is d/r , and so returns $r' = r$ if $\gcd(d, r) = 1$, or $r' = r/\gcd(d, r)$ if not. In the first case the correct answer is returned, and in the second case $r' < r$, so is rejected by the algorithm in step 7. So the only possible cases where an $r' = mr$ with $m > 1$ could be returned is if the condition on c in (202), or equivalently (203), is violated.

We have seen that there are at most r possible values of c satisfying (203), and there are only r values of $x^k \pmod{N}$ that can be measured, so the total probability of a measurement satisfying (203) is at least $r^2 P(c, k) \geq 4/\pi^2$ by (202). Thus, the probability *not* to observe such a c and k is

$$P_{\text{false}} < 1 - \frac{4}{\pi^2} \approx 0.6. \quad (204)$$

This justifies the probability bound quoted in step 9 of Shor's algorithm. Note, however, that this upper bound on the probability of obtaining a false positive result is very crude. Just because c is outside the range $|\{rc\}_q| \leq r/2$, does not mean that an $r' = mr$ will necessarily be returned. Indeed, the resulting r' 's will be more or less random, and so the great majority of them will be rejected in step 7 of the algorithm because $x^{r'} \neq 1 \pmod{N}$. Indeed, as we discussed in some detail in the example in section 5.2, the false positives can only come from a narrow range of possible values of c , and these c 's occur in regions where the probabilities are very small. A general analysis along these lines, which we will not pursue here, gives much more stringent bounds on P_{false} , which mean in practice that the possibility of false positives can be ignored.

The situation is slightly more complicated than there simply being a better bound on P_{false} . If the order $r = \text{ord}(x, N)$ is so small that $\log r \ll \log N$, then there can actually be a relatively

high probability of a false positive $r' = mr$ being returned. But in that case r' has a small factor (r itself), so is easily factorized, uncovering the correct r . If instead $\log r > (\log N)/B$ for some appropriate constant B (say, $B = 3$), then the offending values of c are very sparse, and one can easily estimate $P_{\text{false}} < 10^{-3}$.

Part III

Noise and Error Correction

6 The need for quantum error correction

Suppose a quantum algorithm uses Q qubits and takes S steps (gates). If the typical time needed to perform an elementary qubit operation is t_s , then the total time to run the algorithm is

$$T \simeq t_s S.$$

The value of t_s is an engineering matter: it depends on the design of quantum gates. But reasonable estimates of the minimum possible t_s follow from quantum mechanics. Qubits are physical 2-state systems with some typical energy difference ΔE between the states. The uncertainty relation $t_s \Delta E \geq \hbar$ implies

$$t_s \gtrsim \frac{\hbar}{\Delta E}.$$

Qubits will (almost certainly) be realized on atomic systems using energy splittings less than (optimistically⁶) $\Delta E \sim 10^{-3}$ eV, implying

$$t_s \gtrsim \frac{\hbar}{10^{-3}\text{eV}} \sim \frac{10^{-15}\text{eV} \cdot \text{s}}{10^{-3}\text{eV}} \sim 10^{-12}\text{s}.$$

So there is no reason, in principle, that quantum computers can't be as fast as present-day computers.

Noise

But, the real time for a computation is *not* given by $T = t_s S$. Real-world computers also experience *noise*, which can cause computations to fail with some probability $1 - \mathcal{P}$. Therefore, on the average, you must run the algorithm $1/\mathcal{P}$ times to get a successful computation. So

$$T = \frac{t_s S}{\mathcal{P}}.$$

Let's estimate \mathcal{P} . To do this we need some information about the noise. "Noise" is just an (unwanted) coupling of the computer to its environment. Since the environment is not under complete control, this coupling has a random nature. (If we could detect a pattern to the noise, we could presumably understand and eliminate its source—these are the "systematic errors" of experiment.) We *assume* that the noise acts *locally* and *independently* on each component of the computer. Then the noise will give some failure probability γ per qubit per second. We write

$$\gamma := \frac{1}{\tau_d},$$

⁶It is hard to make ΔE larger because then there are typically many states in atomic systems with smaller splittings, making it difficult to isolate the states with the larger ΔE from all the other states.

defining the **decoherence time** τ_d as the typical time it takes noise to substantially alter the state of a given qubit.

Since there are Q qubits undergoing S steps each taking t_s seconds, the total probability \mathcal{P} of a successful computation is

$$\mathcal{P} \approx (1 - \gamma t_s)^{QS} \approx e^{-\gamma t_s QS} = e^{-QS t_s / \tau_d}.$$

So, the computation time

$$T \sim \frac{t_s S}{\mathcal{P}} \sim t_s S e^{QS t_s / \tau_d}$$

grows exponentially with the number of gates S .

For example, Shor's factoring algorithm to factor a large number $N \sim 2^L$ (*i.e.*, about L bits long) takes about $S \sim L^3$ steps on $Q \sim L$ qubits in the best implementation. So the total time, with noise, for the computation is $T \sim t_s L^3 e^{L^4 t_s / \tau_d}$. So we can ignore the contribution of the noise if $L^4(t_s/\tau_d) \lesssim 1$. For practical applications we are interested in $L \sim 100$ to 1000 , implying we need $(t_s/\tau_d) \lesssim 10^{-8}$ to 10^{-12} otherwise the quantum algorithm performs no better than a classical algorithm.

At present (~ 2015), $(t_s/\tau_d) \simeq 10^{-2}$ or 10^{-3} is the best we can do.⁷ With present technology it is hard to imagine increasing τ_d by more than 10^3 in the next 10 years, leaving us still far short of a practical quantum computer. And this ignores the question of scalability of any technology to many quantum gates!

Classical error correction

Generally, noise is very small in classical computers using semiconducting devices, $(t_s/\tau_d) \simeq 10^{-17}$, and so can be ignored. But in some applications (*e.g.*, transmitting data over long wires) the noise is substantial. In that case we counteract the noise with **error-correcting codes**.

The simplest such code is the **majority voting code**:

- (1) encode the information by copying it to 2 other bits,
- (2) periodically measure all three bits, compare, and correct: *e.g.*,

$$1 \xrightarrow[\text{(1):encode}]{} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \xrightarrow[\text{noise}]{} \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \xrightarrow[\text{(2):correct}]{} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \xrightarrow[\text{decode}]{} 1.$$

It turns out this strategy can remove the exponential noise factor in T at a relatively minor cost: a constant overall factor increasing SQ . The basic idea is that if p is the error rate due to noise, then encoding reduces the overall error rate to $\mathcal{O}(p^2) < p$ as long as p is less than some *threshold value*.

Quantum error correction

Can this strategy also work for quantum computers?

⁷This is for 2-qubit quantum gates; for 1-qubit gates we can currently do $\sim 10^{-6}$. In 2016 the record for 2-qubit decoherence time was $\tau_d \lesssim 10^{-4}$ s.

There are some obvious problems. Step (1) of the classical error-correction algorithm involves duplicating the classical bits. But the no-cloning theorem says that duplication of qubits is impossible. Step (2) of the classical algorithm involves measuring the bits. But measurements irreversibly change the state of a quantum system.

Nevertheless, Peter Shor showed in 1994 that a quantum analog of error correction does exist. For the past couple of decades, much of the theoretical work on quantum computation has been on error correction.

The goal of the analysis of error correction is to show how to do **fault-tolerant quantum computations** in such a way that one does not exponentially increase the number of gates needed for the computation. The central result (which we will motivate only for the special case of qubit transmission in a noisy channel) is the

Threshold theorem: If you want to carry out a quantum algorithm using N gates, but each gate fails with probability p , then it is possible to design a quantum algorithm to do the computation with an arbitrarily small overall failure probability ϵ by using only $\mathcal{O}(N(\epsilon^{-1} \log N)^x)$ gates for some positive x , as long as $p < p_{\text{th}}$, where p_{th} is some constant threshold probability.

The value of the required threshold probability, p_{th} (and the exponent x) depends on the error-correction code. For example, using the Steane code for error correction (which we will not describe in detail) one needs $p_{\text{th}} = (t_s/\tau_d) \lesssim 10^{-4}$. This value is conceivably within reach of development of current technologies.

7 Models of noise

The goal of this section is to understand how to model the effect of noise on quantum systems. The central result will be to show that on a *single qubit*, the most general noise has the effect of mapping states to **density matrices** as:

$$\text{noise} : |\psi\rangle \mapsto \rho = \sum_k E_k |\psi\rangle \langle \psi| E_k^\dagger$$

with

$$\sum_k E_k E_k^\dagger = I$$

for some

$$E_k := e_{k0}I + e_{kx}X + e_{ky}Y + e_{kz}Z.$$

(Here X , Y , and Z are alternate names for the Pauli matrices σ_x , σ_y , and σ_z , respectively.) To get to this result, first we will briefly review the standard model of classical noise. We next review the density matrix formulation of quantum mechanics in some detail since it is necessary to describe noise. And then we develop the quantum operations formalism which gives rise to the above result.

7.1 Classical noise

Noise is a coupling to an “outside” system about which we have incomplete knowledge. We model its effect on a single classical bit as follows.

A gate acts on the initial state of the bit in a predictable way, which we can describe as a map from the initial state of the bit, $X \in \{0, 1\}$, to its final state, $Y \in \{0, 1\}$. Once noise is added, the behavior of the gate becomes unpredictable, so we can at best only have probabilistic information about the state of the bit. So call the probabilities of the various initial and final states

$$\begin{cases} p_0 &:= \text{prob}(X = 0) \\ p_1 &:= \text{prob}(X = 1) \end{cases} \quad \begin{cases} q_0 &:= \text{prob}(Y = 0) \\ q_1 &:= \text{prob}(Y = 1) \end{cases}$$

Then the action of the gate plus noise can be modeled as a linear map of the initial to final probabilities:

$$q_j = \sum_{i=0}^1 E_{ji} p_i, \quad (205)$$

where the matrix elements E_{ji} are the conditional probabilities

$$E_{ji} = \text{prob}(Y = j | X = i).$$

The probability map (205) can also be written in matrix notation

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

and refer to E as the **evolution matrix** for the gate.

We model the noise as acting *independently* on each gate, so each gate will have its own evolutions matrix, E_1 , E_2 , and so forth:

$$\begin{array}{ccccc} X & \xrightarrow{E_1} & Y & \xrightarrow{E_2} & Z \\ \vec{p} & & \vec{q} & & \vec{r} \end{array}$$

so that in matrix notation

$$\vec{r} = E_2 E_1 \vec{p}.$$

The assumption of independent errors is a *physical* assumption: it should be tested in a given situation. When errors are not independent, but act coherently on many bits, we can still error-correct, but to do so we must use different codes. But we will not discuss these kinds of noise models or error codes further.

Note that even though we are dealing with probabilities, this is *not* quantum mechanics: the probabilities are not inherent, but just reflect our lack of detailed knowledge about the source of the noise.

The evolution matrix E for a single bit is not arbitrary since the probabilities must add to one. Thus

$$\vec{p} = \begin{pmatrix} p \\ 1-p \end{pmatrix}, \quad \vec{q} = \begin{pmatrix} q \\ 1-q \end{pmatrix},$$

where $0 \leq p, q \leq 1$ are the probabilities that $X = 0$ and $Y = 0$ respectively. Likewise, the sum of the two probabilities in $E\vec{p}$ must sum to one, implying

$$E_{12} + E_{22} + (E_{11} - E_{12} + E_{21} - E_{22})p = 1.$$

Since this must be true for all p , it implies $E_{12} + E_{22} = 1$ and $E_{11} + E_{21} = 1$, so we can parameterize the evolution matrix by just two numbers as

$$E := \begin{pmatrix} E_{11} & E_{12} \\ E_{21} & E_{22} \end{pmatrix} = \begin{pmatrix} 1-f & e \\ f & 1-e \end{pmatrix}. \quad (206)$$

Also, $0 \leq q \leq 1$ implies $0 \leq 1 - e + (1 - f - e)p \leq 1$ for all p , implying

$$0 \leq e, f \leq 1.$$

This can be summarized by saying that classical 1-bit noise is characterized by 2 numbers: the probability e for “1” to flip, and an independent probability f for “0” to flip. We want to find a similar characterization of noise for qubits.

7.2 Density matrices

Our challenge is to incorporate both quantum uncertainty and noise uncertainty in one formalism. A state $|\psi\rangle$ gives *complete* information on a system (though only gives statistical predictions). But in realistic situations we usually only have *partial* information; *i.e.*, we can’t be sure exactly which state the system is in.

When we repeat an experiment (or quantum computation) we only know the initial state of the system is

$$|\psi\rangle = \begin{cases} |\psi_1\rangle & \text{with probability } p_1 \\ |\psi_2\rangle & \text{" } p_2 \\ \vdots & \vdots \\ |\psi_n\rangle & \text{" } p_n \end{cases} \quad (207)$$

with $\sum_{a=1}^n p_a = 1$ and $p_a \geq 0$. This kind of probabilistic description of the state of a system is often called a **mixed state** or an **ensemble of states**.⁸ Note that the $|\psi_a\rangle$ are *not* necessarily orthogonal, but we do assume they are all distinct and normalized. If there is only one line in (207), *i.e.* $|\psi\rangle$ is a definite state (with probability 1), then we say that it is a **pure state**.

One might guess (wrongly!) that this state (207) is just a superposition of the $|\psi_a\rangle$ states, say

$$|\psi\rangle \stackrel{?}{=} \sum_{a=1}^n \sqrt{p_a} |\psi_a\rangle. \quad (208)$$

(You should check that this state is normalized if the $|\psi_a\rangle$ are orthonormal.) However, we can see that this guess is wrong in a number of ways. First, on a basic conceptual level

⁸Some people still call it an **incoherent superposition of states**, which is a terribly misleading name, and should be avoided.

(208) cannot be right because it is a definite state, not like (207) which is by definition not a definite state. On a more prosaic level, we can see it is wrong by seeing what the two states, (207) and (208), predict for a measurement.

If we measure an observable M in state $|\psi_a\rangle$, we get the expectation value⁹

$$\langle M \rangle_a = \langle \psi_a | M | \psi_a \rangle. \quad (209)$$

Since, according to (207) we are in state $|\psi_a\rangle$ with probability p_a , the total expectation value will be

$$\langle M \rangle = \sum_a p_a \langle \psi_a | M | \psi_a \rangle. \quad (210)$$

But in the pure state $|\psi\rangle$ of (208) we would have

$$\begin{aligned} \langle M \rangle &\stackrel{?}{=} \langle \psi | M | \psi \rangle = \sum_{a,b} (\sqrt{p_a} |\psi_a\rangle)^\dagger M \sqrt{p_b} |\psi_b\rangle \\ &= \sum_a p_a \langle \psi_a | M | \psi_a \rangle + \sum_{a \neq b} \sqrt{p_a p_b} \langle \psi_a | M | \psi_b \rangle. \end{aligned}$$

The first term agrees with the right answer (210), but the second, interference, term need not vanish, showing that (208) cannot be right.

If $\{|i\rangle\}$ is an orthonormal basis of the Hilbert space, then recall that completeness of the basis means that $I = \sum_i |i\rangle\langle i|$ where I is the identity operator. Also recall that the trace of an operator is defined to be $\text{tr}(X) := \sum_i \langle i | X | i \rangle$. These can be used to rewrite (209) in a suggestive way:

$$\begin{aligned} \langle M \rangle &= \sum_a p_a \langle \psi_a | M | \psi_a \rangle = \sum_a p_a \langle \psi_a | M I | \psi_a \rangle \\ &= \sum_a p_a \langle \psi_a | M \left(\sum_i |i\rangle\langle i| \right) | \psi_a \rangle = \sum_{i,a} p_a \langle \psi_a | M | i \rangle \langle i | \psi_a \rangle \\ &= \sum_{i,a} p_a \langle i | \psi_a \rangle \langle \psi_a | M | i \rangle = \sum_i \langle i | \left(\sum_a p_a |\psi_a\rangle\langle \psi_a| \right) M | i \rangle \\ &= \text{tr} \left[\left(\sum_a p_a |\psi_a\rangle\langle \psi_a| \right) M \right]. \end{aligned} \quad (211)$$

This motivates us to identify the mixed state (207) with the *operator*

$$\boxed{\rho := \sum_{a=1}^n p_a |\psi_a\rangle\langle \psi_a|, \quad \text{with } 0 \leq p_a \leq 1 \quad \text{and} \quad \sum_{a=1}^n p_a = 1.} \quad (212)$$

ρ is called a **density matrix**, and encodes all the quantum-mechanical information about the mixed state. For example, (211) implies

$$\boxed{\langle M \rangle = \text{tr}(\rho M).} \quad (213)$$

⁹Recall the definition and interpretation of the expectation value, discussed in section 0 below equation (30). An examination of the rules of quantum mechanics shows that *all* measurements in quantum mechanics can be written in terms of expectation values, so it is sufficient to focus just on them.

Exercise 7.1 Show that $\text{tr}(AB) = \text{tr}(BA)$ for all A, B .

Exercise 7.2 Show that $\text{tr}(\rho) = 1$ using $\sum_a p_a = 1$.

Exercise 7.3 Show that $\rho = \rho^\dagger$.

Exercise 7.4 Show that ρ is a **positive operator**: $\langle \psi | \rho | \psi \rangle \geq 0$ for all $|\psi\rangle$.

These last three properties actually *define* density matrices.

It is important to note that two different mixed states (207) can give rise to the same density matrix (212). For example, consider the two mixed states

$$|\psi\rangle = \begin{cases} |\psi_1\rangle & \text{with probability } p_1 = 1/2 \\ |\psi_2\rangle & \text{" } p_2 = 1/2 \end{cases}$$

with $|\psi_1\rangle$ and $|\psi_2\rangle$ orthogonal, and

$$|\psi'\rangle = \begin{cases} |\psi_+\rangle & \text{with probability } p_+ = 1/2 \\ |\psi_-\rangle & \text{" } p_- = 1/2 \end{cases}$$

where $|\psi_\pm\rangle := (|\psi_1\rangle \pm |\psi_2\rangle)/\sqrt{2}$. The density matrix for the first state is $\rho = (|\psi_1\rangle\langle\psi_1| + |\psi_2\rangle\langle\psi_2|)/2$. The density matrix for the second is

$$\begin{aligned} \rho' &= \frac{1}{2} \cdot \frac{1}{2} \left(|\psi_1\rangle - |\psi_2\rangle \right) \left(\langle\psi_1| - \langle\psi_2| \right) + \frac{1}{2} \cdot \frac{1}{2} \left(|\psi_1\rangle + |\psi_2\rangle \right) \left(\langle\psi_1| + \langle\psi_2| \right) \\ &= \frac{1}{2} \left(|\psi_1\rangle\langle\psi_1| + |\psi_2\rangle\langle\psi_2| \right) = \rho. \end{aligned}$$

But all possible quantum measurements only depend on ρ . Therefore we learn that these two different-looking mixed states are *physically equivalent*.

What is the density matrix of a system that is actually in a pure state $|\psi\rangle$ 100% of the time? Then the sum in (212) is just one term with $p_a = 1$, giving $\rho = |\psi\rangle\langle\psi|$. So for a pure state ψ we have the correspondence

$$\boxed{|\psi\rangle \longleftrightarrow \rho = |\psi\rangle\langle\psi|.} \quad (214)$$

Note that in this case ρ is a **projection operator** ($\rho^2 = \rho$ and $\rho = \rho^\dagger$) which implies $\text{tr}(\rho^2) = \text{tr}(\rho) = 1$. You can show that for any density matrix,

$$\boxed{\text{tr}(\rho^2) \leq 1, \quad \text{and} \quad \text{tr}(\rho^2) = 1 \text{ iff } \rho \text{ is a pure state.}} \quad (215)$$

This gives a useful way of telling whether a density matrix corresponds to a pure state or not.

To summarize: we can describe both the statistical uncertainty coming from our lack of knowledge of a system together with the inherent quantum uncertainties by describing systems in terms of operators ρ satisfying $\rho^\dagger = \rho \geq 0$ and $\text{tr}\rho = 1$ instead of states. Everything we can do with states $|\psi\rangle$, we can do with the corresponding $\rho = |\psi\rangle\langle\psi|$. But we can also describe mixed states with ρ 's satisfying $\text{tr}\rho^2 < 1$.

Time evolution of density matrices.

$$\begin{aligned}\rho(t) &= \sum_a p_a |\psi_a(t)\rangle\langle\psi_a(t)| = \sum_a p_a U(t) |\psi_a(0)\rangle\langle\psi_a(0)| U^\dagger(t) \\ &= U(t) \left(\sum_a p_a |\psi_a(0)\rangle\langle\psi_a(0)| \right) U^\dagger(t),\end{aligned}$$

implies

$$\boxed{\rho(t) = U(t) \rho(0) U^\dagger(t)} \quad (216)$$

where $U(t)$ is the unitary operator which gives the time evolution of states. Thus, in terms of quantum circuit diagrams, the usual gate action on pure states,

$$|\psi\rangle \text{ --- } \boxed{U} \text{ --- } U|\psi\rangle, \quad (217)$$

becomes

$$\rho \text{ --- } \boxed{U} \text{ --- } U\rho U^\dagger \quad (218)$$

on density matrices.

Measurements with density matrices. I'll just give the results—see [NC] section 2.4 for the details.

If we measure an observable $M = M^\dagger$, recall that the only results that can be observed are the eigenvalues of M . Recall from the **spectral decomposition theorem** (21) that M can be written as a sum of orthogonal projection operators

$$M = \sum_i \lambda_i |\psi_i\rangle\langle\psi_i|, \quad \lambda_i \in \mathbb{R},$$

with $\{|\psi_i\rangle\}$ an ortho-normal basis so that $\langle\psi_i|\psi_j\rangle = \delta_{ij}$. This implies that $M|\psi_j\rangle = \lambda_j|\psi_j\rangle$. So the $\{\lambda_i\}$ are the **eigenvalues** of M , *i.e.*, the set of possible outcomes of measurement of M . (And the $\{|\psi_i\rangle\}$ are a basis of **eigenvectors** of M .) More abstractly, we write

$$M = \sum_i \lambda_i P_i$$

where $P_i := |\psi_i\rangle\langle\psi_i|$ is the **projection operator** onto the eigenspace of the i th eigenvalue λ_i . Hermiticity of M implies that $P_i^2 = P_i$ and $P_i P_j = 0$ if $i \neq j$.

Then, when we measure M in a (mixed) state represented by a density matrix ρ , we find

$$\boxed{\text{prob}(M = \lambda_i) = \text{tr}(P_i \rho)}, \quad (219)$$

and the density matrix changes to

$$\boxed{\rho \rightarrow \rho' = \frac{1}{\text{tr}(P_i \rho)} P_i \rho P_i} \quad (220)$$

if $M = \lambda_i$ is actually observed. (This is for an **ideal** or **non-destructive** measurement.)

Exercise 7.5 Show that when ρ corresponds to a pure state, $\rho = |\psi\rangle\langle\psi|$, that (219) and (220) reduce to the familiar “Born rules” for quantum measurements, namely $\text{prob}(M = \lambda_i) = |\langle\psi|\psi_i\rangle|^2$ and $|\psi\rangle \rightarrow |\psi'\rangle = e^{i\alpha}|\psi_i\rangle$, where $e^{i\alpha}$ is a phase you should compute.

Main example: a single qubit

A single qubit is described by a 2-dimensional Hilbert space with orthonormal basis $\{|0\rangle, |1\rangle\}$. In this basis ρ is a 2×2 hermitian matrix with $\text{tr}\rho = 1$ and positive. Hermiticity implies

$$\rho = \frac{1}{2} \begin{pmatrix} r_0 + r_z & r_x - ir_y \\ r_x + ir_y & r_0 - r_z \end{pmatrix}$$

for real r_0, r_x, r_y, r_z . $\text{tr}\rho = 1$ implies $r_0 = 1$. Positivity implies that all the eigenvalues of ρ are positive. To find the eigenvalues, solve the characteristic equation

$$0 = \det(\rho - \lambda I) = \frac{1}{4} [(1 - 2\lambda)^2 - r_x^2 - r_y^2 - r_z^2] \Rightarrow \lambda = \frac{1}{2} [1 \pm \sqrt{r_x^2 + r_y^2 + r_z^2}].$$

So positivity implies that $r_x^2 + r_y^2 + r_z^2 \leq 1$. Rewrite ρ as

$$\rho = \frac{1}{2} \{I + r_x X + r_y Y + r_z Z\}$$

where

$$X := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \sigma_x, \quad Y := \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = \sigma_y, \quad Z := \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \sigma_z \quad (221)$$

are the *Pauli matrices*. We will use a vector notation $\vec{X} = (X, Y, Z)$, $\vec{r} = (r_x, r_y, r_z)$, so that the general 1-qubit density matrix is parameterized as

$$\boxed{\rho = \frac{1}{2} \{I + \vec{r} \cdot \vec{X}\} \quad \text{with } |\vec{r}| \leq 1.} \quad (222)$$

So, the general density matrix for a single qubit is parametrized by a 3-vector \vec{r} with $|\vec{r}| \leq 1$, the **Bloch ball**, shown in figure 3.

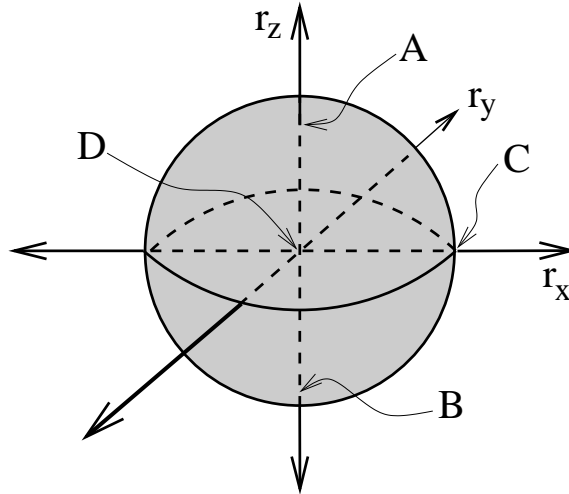


Figure 3: The Bloch ball: any point in the interior or on the boundary corresponds to a 1-qubit ρ . The states corresponding to the points A, B, C, and D are discussed in the notes.

When does ρ correspond to a pure state? ρ is pure if and only if $\text{tr}(\rho^2) = 1$.

Exercise 7.6 Show that for the 1-qubit density matrix $\text{tr}(\rho^2) = (1 + \vec{r} \cdot \vec{r})/2$.

Thus ρ is pure if and only if $|\vec{r}| = 1$. This describes the **Bloch sphere**, the boundary of the Bloch ball. Thus, for example, the “north pole” of the Bloch sphere, $\vec{r} = (0, 0, 1)$ or point A in figure 3, is the density matrix

$$\rho_A = \frac{1}{2}(I + Z) = \frac{1}{2} \begin{pmatrix} 1+1 & 0 \\ 0 & 1-1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \end{pmatrix} = |0\rangle\langle 0|,$$

and therefore corresponds to the pure state $|0\rangle$.

Exercise 7.7 Show that the “south pole” of the Bloch sphere, $\vec{r} = (0, 0, -1)$ or point B in figure 3 corresponding to the density matrix $\rho_B = (I - Z)/2$, is the pure state $|1\rangle$.

Exercise 7.8 Show that the equatorial point $\vec{r} = (1, 0, 0)$, or point C in figure 3 corresponding to the density matrix $\rho_C = (I + X)/2$ is the pure state $(|0\rangle + |1\rangle)/\sqrt{2}$.

What does the center of the ball, $\vec{r} = 0$, correspond to? In this case $\rho = I/2 = (|0\rangle\langle 0| + |1\rangle\langle 1|)/2$. This is a mixed state which is $|0\rangle$ 50% of the time, and $|1\rangle$ 50% of the time. Therefore $\vec{r} = 0$ corresponds to the maximally mixed state, the state with *no information*, or maximum entropy.

Entropy, S , is a measure of the disorder of a system; more entropy means less information is available about the state of the system. One general measure of entropy in quantum mechanics is

$$S = -\text{tr}(\rho \ln \rho).$$

Recall that the logarithm of a matrix is just the inverse of exponentiation, and has an expansion

$$\ln(I + A) = A - \frac{1}{2}A^2 + \frac{1}{3}A^3 - \dots,$$

for A 's with entries small compared to 1.

Exercise 7.9 Show that the entropy of 1-qubit state with density matrix ρ with $|\vec{r}| \ll 1$ is $S \approx \ln 2 - \frac{1}{2}\vec{r} \cdot \vec{r} + \mathcal{O}(r^4)$.

In general, the smaller $|\vec{r}|$, the greater the entropy, so the less information ρ contains about the state of the system.

Exercise 7.10 Show that the general 1-qubit ρ can be written as $\rho = \sum_{a=0}^1 p_a |\psi_a\rangle\langle\psi_a|$ with $p_0 = (1 + |\vec{r}|)/2$ and $p_1 = (1 - |\vec{r}|)/2$, where

$$|\psi_0\rangle := \cos \frac{\theta}{2} |0\rangle + e^{i\phi} \sin \frac{\theta}{2} |1\rangle, \quad |\psi_1\rangle := e^{-i\phi} \sin \frac{\theta}{2} |0\rangle - \cos \frac{\theta}{2} |1\rangle,$$

where (θ, ϕ) are the polar angles of the vector $|\vec{r}|$ (i.e., $r_z = |\vec{r}| \cos \theta$, $r_x = |\vec{r}| \sin \theta \cos \phi$, and $r_y = |\vec{r}| \sin \theta \sin \phi$).

How does noise give rise to density matrices?

If we had complete knowledge of the environment and its interaction with our computer, then there would be no need for density matrices. A computation with noise can be described as

$$\left. \begin{array}{c} |\psi_{\text{comp}}\rangle \\ |\chi_{\text{env}}\rangle \end{array} \right\} \xrightarrow{U} |\Psi_{\text{out}}\rangle \quad (223)$$

where the computer's initial state $|\psi_{\text{comp}}\rangle$ and the environment's initial state $|\chi_{\text{env}}\rangle$ interact and undergo some unitary evolution U (representing the effects of noise and the computer gates) giving some final state $|\Psi_{\text{out}}\rangle$.

Note that $|\Psi_{\text{out}}\rangle$ will, in general, *entangle* the computer with the environment. This just means that the final state need not factorize as a tensor product of $|\text{computer}\rangle \otimes |\text{environment}\rangle$. To write the most general final state, introduce a basis for the state space. In particular, say $\{|a\rangle\}$ is an orthonormal basis of $\mathcal{H}_{\text{comp}}$, the computer's Hilbert space, while $\{|n\rangle\}$ is an orthonormal basis of \mathcal{H}_{env} , the environment's Hilbert space. Then

$$|\Psi_{\text{out}}\rangle = \sum_{a,n} \Psi_{an} |a\rangle |n\rangle$$

for some complex numbers Ψ_{an} satisfying $\sum_{a,n} |\Psi_{an}|^2 = 1$.

Now say we wanted to measure some observable M on the *computer's* state. That means that as an operator on $\mathcal{H}_{\text{comp}} \otimes \mathcal{H}_{\text{env}}$, the observable is of the form $M \otimes I$. Then

$$\begin{aligned} \langle M \otimes I \rangle &= \langle \Psi_{\text{out}} | M \otimes I | \Psi_{\text{out}} \rangle = \left(\sum_{a,n} \Psi_{an}^* \langle a | \langle n | \right) M \otimes I \left(\sum_{b,m} \Psi_{bm} | b \rangle | m \rangle \right) \\ &= \sum_{a,b,n,m} \Psi_{an}^* \Psi_{bm} \langle a | M | b \rangle \langle n | m \rangle = \sum_{a,b,n,m} \Psi_{an}^* \Psi_{bm} \langle a | M | b \rangle \delta_{nm} \\ &= \sum_{a,b} \left(\sum_n \Psi_{an}^* \Psi_{bn} \right) \langle a | M | b \rangle. \end{aligned}$$

We can put this result in a more suggestive form by inserting the completeness relation $I = \sum_c |c\rangle \langle c|$ on the computer states, giving

$$\begin{aligned} \langle M \otimes I \rangle &= \sum_{a,b,c} \left(\sum_n \Psi_{an}^* \Psi_{bn} \right) \langle a | M | c \rangle \langle c | b \rangle = \sum_{a,b,c} \left(\sum_n \Psi_{an}^* \Psi_{bn} \right) \langle c | (|b\rangle \langle a|) M | c \rangle \\ &= \sum_c \langle c | \left[\sum_{a,b} \left(\sum_n \Psi_{an}^* \Psi_{bn} \right) |b\rangle \langle a| \right] \cdot M | c \rangle = \text{tr}_{\mathcal{H}_c}(\rho M) \end{aligned}$$

where we have defined

$$\rho := \sum_{a,b} \left(\sum_n \Psi_{an}^* \Psi_{bn} \right) |b\rangle \langle a|$$

in the last line. Comparing to (213), we identify ρ as the density matrix for the computer, after interaction with the environment. Note that ρ is an operator on \mathcal{H}_c only. You should check that the conditions for a density matrix, namely, $\rho = \rho^\dagger$, $\text{tr}_c(\rho) = 1$, and $\rho \geq 0$, are all satisfied.

Suppose $|\Psi_{\text{out}}\rangle$ were *not* entangled:

$$|\Psi_{\text{out}}\rangle = |\psi'_{\text{comp}}\rangle \otimes |\chi'_{\text{env}}\rangle := \left(\sum_a \psi_a |a\rangle \right) \otimes \left(\sum_n \chi_n |n\rangle \right) = \sum_{a,n} \psi_a \chi_n |a\rangle |n\rangle,$$

with $\sum_a |\psi_a|^2 = \sum_n |\chi_n|^2 = 1$. Then the computer's density matrix becomes

$$\begin{aligned} \rho &= \sum_{a,b,n} \psi_a^* \chi_n^* \psi_b \chi_n |b\rangle \langle a| = \sum_{a,b} \psi_a^* \psi_b \left(\sum_n |\chi_n|^2 \right) |b\rangle \langle a| \\ &= \left(\sum_b \psi_b |b\rangle \right) \left(\sum_a \langle a| \psi_a^* \right) = |\psi'_{\text{comp}}\rangle \langle \psi'_{\text{comp}}|. \end{aligned}$$

Thus we see that if there is no entanglement with the environment, ρ corresponds to a pure state. Recalling that a pure state is one about which we have complete information, we conclude that

$$\text{Noise} = \text{entanglement with the environment.}$$

7.3 Quantum operations

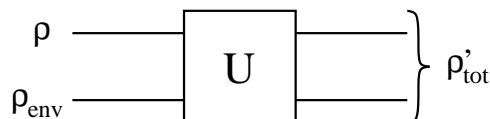
Now that we have the machinery to deal with mixed states (*i.e.*, classical uncertainty) in quantum mechanics, we apply it to modelling noise.

Recall that we modelled the action of noise and gate together on a classical bit by an evolution matrix E taking $\vec{p} \rightarrow \vec{q} = E\vec{p}$ where \vec{p} is the probability distribution of the input bit, and \vec{q} is the probability distribution of the output bit. Analogously, in quantum mechanics, if ρ is the density matrix of the input state of the computer, and ρ' is the density matrix of the output state of the computer, then we want to model the most general effect of any gates plus noise as a map \mathcal{E} taking the input density matrix to the output one:

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

We want to characterize the most general \mathcal{E} allowed by the rules of quantum mechanics. Such \mathcal{E} 's are called **quantum operations** (not to be confused with operators).

Consider the combined system of computer plus environment. It is described by a total Hilbert space $\mathcal{H}_{\text{tot}} = \mathcal{H}_{\text{comp}} \otimes \mathcal{H}_{\text{env}}$, where $\mathcal{H}_{\text{comp}}$ is the Hilbert space of the computer and \mathcal{H}_{env} is the Hilbert space of the environment. Suppose that initially the computer is in a (mixed) state represented by the density matrix ρ , and the environment is in a (mixed) state represented by the density matrix ρ_{env} .



The total evolution of the combined system is by some unitary operator U on \mathcal{H}_{tot} . U encodes the “gate” (the evolution of the computer states), the “noise” (the interaction of computer and environment), and the evolution of the environment itself. Thus the mixed state will be some density matrix ρ'_{tot} on \mathcal{H}_{tot} given by

$$\rho'_{\text{tot}} = U(\rho \otimes \rho_{\text{env}})U^\dagger. \quad (224)$$

We want to extract the final density matrix ρ' for the computer alone from ρ'_{tot} .

We can deduce what it must be by looking at the result of an arbitrary measurement on the computer. Suppose $M = M^\dagger$ is an observable on $\mathcal{H}_{\text{comp}}$. On the total system the observable is $M \otimes I_{\text{env}}$. So the expectation value of M is given by

$$\langle M \rangle = \text{tr}_{\mathcal{H}_{\text{tot}}}(\rho'_{\text{tot}}[M \otimes I_{\text{env}}]).$$

Here the subscript on the trace is to remind us that we are tracing over the whole \mathcal{H}_{tot} Hilbert space. Choose orthonormal bases $\{|a\rangle\}$ for $\mathcal{H}_{\text{comp}}$ and $\{|n\rangle\}$ for \mathcal{H}_{env} . Then $\{|a\rangle \otimes |n\rangle\}$ is an orthonormal basis for \mathcal{H}_{tot} , and an easy calculation gives

$$\begin{aligned} \langle M \rangle &= \sum_{a,n} \langle a | \langle n | (\rho'_{\text{tot}}[M \otimes I_{\text{env}}]) | a \rangle | n \rangle \\ &= \sum_a \langle a | \left(\sum_n \langle n | (\rho'_{\text{tot}}[M \otimes I_{\text{env}}]) | n \rangle \right) | a \rangle \\ &:= \text{tr}_{\mathcal{H}_{\text{comp}}}(\rho' M) \end{aligned} \quad (225)$$

where

$$\boxed{\rho' := \text{tr}_{\mathcal{H}_{\text{env}}}(\rho'_{\text{tot}})}. \quad (226)$$

Here the subscripts on the traces denote **partial traces** over the computer or environment Hilbert spaces. They are defined by

$$\text{tr}_{\mathcal{H}_{\text{comp}}}(A) := \sum_a \langle a | A | a \rangle, \quad \text{tr}_{\mathcal{H}_{\text{env}}}(A) := \sum_n \langle n | A | n \rangle,$$

for any operator A on the total Hilbert space. Thus $\text{tr}_{\mathcal{H}_{\text{comp}}}(A)$ is an operator on \mathcal{H}_{env} , and $\text{tr}_{\mathcal{H}_{\text{env}}}(A)$ is an operator on $\mathcal{H}_{\text{comp}}$. Also, $\text{tr}_{\mathcal{H}_{\text{tot}}}(A) = \text{tr}_{\mathcal{H}_{\text{comp}}}(\text{tr}_{\mathcal{H}_{\text{env}}}(A)) = \text{tr}_{\mathcal{H}_{\text{env}}}(\text{tr}_{\mathcal{H}_{\text{comp}}}(A))$.

Exercise 7.11 Show that a partial trace of a density matrix is a density matrix. That is, show that ρ' defined by (226) satisfies $\rho'^\dagger = \rho' \geq 0$ and $\text{tr}_{\mathcal{H}_{\text{comp}}}\rho' = 1$ if ρ'_{tot} has the same properties on \mathcal{H}_{tot} .

Exercise 7.12 If the unitary evolution operator U acts only on the computer, so $U = U_{\text{comp}} \otimes I_{\text{env}}$, then show that the unitary evolution of the reduced density matrix ρ' is given by $\rho' \rightarrow U_{\text{comp}}\rho'U_{\text{comp}}^\dagger$.

Combining (226) with the evolution formula (224), we conclude

$$\rho' = \text{tr}_{\mathcal{H}_{\text{env}}} [U(\rho \otimes \rho_{\text{env}})U^\dagger] := \mathcal{E}(\rho), \quad (227)$$

giving the general form of a quantum operator \mathcal{E} , since \mathcal{H}_{env} , ρ_{env} , and U are all arbitrary.

We now want to manipulate (227) to get rid of the partial trace over the environment space. For simplicity, assume $\rho_{\text{env}} = |\psi_{\text{env}}\rangle\langle\psi_{\text{env}}|$, a pure state.¹⁰ Also, chose some orthonormal basis $\{|n\rangle\}$ for \mathcal{H}_{env} . Then

$$\begin{aligned}\rho' &= \text{tr}_{\mathcal{H}_{\text{env}}} [U (\rho \otimes |\psi_{\text{env}}\rangle\langle\psi_{\text{env}}|) U^\dagger] = \sum_n \langle n|U (\rho \otimes |\psi_{\text{env}}\rangle\langle\psi_{\text{env}}|) U^\dagger|n\rangle \\ &= \sum_n \langle n|U|\psi_{\text{env}}\rangle \rho \langle\psi_{\text{env}}|U^\dagger|n\rangle,\end{aligned}$$

implying that

$$\boxed{\mathcal{E}(\rho) = \sum_n E_n \rho E_n^\dagger}, \quad (228)$$

where $E_n := \langle n|U|\psi_{\text{env}}\rangle$ are some operators on $\mathcal{H}_{\text{comp}}$, the computer's Hilbert space. The advantage of (228) is that only operators on $\mathcal{H}_{\text{comp}}$ appear, so we need no longer refer to \mathcal{H}_{env} . The operators E_n are called **operation elements** and encode all the relevant information about U and $|\psi_{\text{env}}\rangle$.

The E_n 's cannot be arbitrary, though. Since $\rho' = \mathcal{E}(\rho)$ is a density matrix, it must satisfy $\rho' = (\rho')^\dagger$, $\rho' \geq 0$, and $\text{tr}\rho' = 1$. The first two are easy to check. The third implies

$$1 = \text{tr}(\mathcal{E}(\rho)) = \text{tr}\left(\sum_n E_n \rho E_n^\dagger\right) = \sum_n \text{tr}(E_n \rho E_n^\dagger) = \sum_n \text{tr}(E_n^\dagger E_n \rho).$$

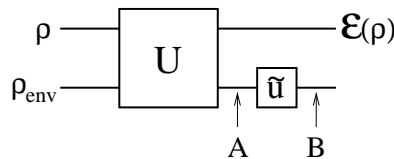
Since this should hold for any computer state ρ , it implies

$$\boxed{\sum_n E_n^\dagger E_n = I.} \quad (229)$$

(Note that, in general, (229) need not hold for E_n and E_n^\dagger in the other order: $\sum_n E_n E_n^\dagger \neq I$.)

In summary: the general effect of noise on a quantum system is to map $\rho \rightarrow \rho' = \mathcal{E}(\rho)$ with \mathcal{E} given by (228) with E_n 's satisfying (229). This is called the **operator sum representation** of a quantum operation. In our derivation of it we only assumed a unitary evolution of the computer plus environment. But it can be generalized to include the effects of possible of measurements done on the system as well; see [NC] section 8.2.

The operation elements E_n are not unique: two different sets of operation elements can give rise to the same quantum operation. Say $\mathcal{E}(\rho) = \sum_n E_n \rho E_n^\dagger$ and $\mathcal{F}(\rho) = \sum_n F_n \rho F_n^\dagger$ are two quantum operations. It can then be shown ([NC] theorem 8.2) that $\mathcal{E}(\rho) = \mathcal{F}(\rho)$ for all ρ if and only if $E_n = \sum_m \tilde{u}_{nm} F_m$ for some unitary matrix \tilde{u}_{nm} . A physical way of understanding this is



¹⁰To generalize, just replace ρ_{env} with a sum over many such states.

where the idea is that $\mathcal{E}(\rho)$ shouldn't depend on whether we observe the environment at time A or time B .

This freedom in the choice of operation elements in the operator sum representation plays an important role in the proofs of the main theorems concerning quantum error-correction. We will not emphasize these proofs here, but wish to proceed a bit more intuitively.

To help build up our intuition, first note that the quantum operations of a special form have a simple physical interpretation. Consider a pure state $|\psi\rangle$. Then $\mathcal{E}(|\psi\rangle) = \sum_n E_n |\psi\rangle \langle \psi| E_n^\dagger$. Define $|\psi_n\rangle$ to be the normalized state that results when E_n acts on $|\psi\rangle$:

$$|\psi_n\rangle := \frac{E_n |\psi\rangle}{\|E_n |\psi\rangle\|}.$$

We can write this as

$$E_n |\psi\rangle = \sqrt{p_n} |\psi_n\rangle, \quad \text{where } \sqrt{p_n} := \|E_n |\psi\rangle\|.$$

Thus the quantum operation takes $|\psi\rangle$ to the density matrix

$$\mathcal{E}(|\psi\rangle) = \sum_n p_n |\psi_n\rangle \langle \psi_n|.$$

Thus we interpret the quantum operation \mathcal{E} as taking $|\psi\rangle$ to $|\psi_n\rangle$ with probability p_n .

Exercise 7.13 Check that $\sum_n p_n = 1$ follows from (229).

Examples: quantum operations on a single qubit

Depolarizing channel or “information loss”.

This is the simplest model of noise:

- probability γ to lose all information: $\rho \rightarrow I/2$,
- probability $1 - \gamma$ to lose no information: $\rho \rightarrow \rho$.

Therefore

$$\rho' = \mathcal{E}_{\text{DC}}(\rho) = \gamma \frac{1}{2} I + (1 - \gamma) \rho.$$

Since ρ' and ρ are density matrices, they can be represented by points in the Bloch ball. Thus $\mathcal{E}(\rho)$ can be thought of as a map of the Bloch ball into itself. For the depolarizing channel, recall that $\rho = I/2$ maps to the origin of the sphere. Then it is not hard to see that \mathcal{E}_{DC} just *shrinks* the Bloch ball, as illustrated in figure 4.

How is \mathcal{E}_{DC} written in the operator sum representation? A useful identity is

$$\frac{1}{2} I = \frac{1}{4} (\rho + X\rho X + Y\rho Y + Z\rho Z) \tag{230}$$

for any 1-qubit ρ , where X , Y , and Z are the Pauli matrices given in (221). They have certain very useful properties that are easily checked:

- $X = X^\dagger$, $Y = Y^\dagger$, $Z = Z^\dagger$ are all hermitian.

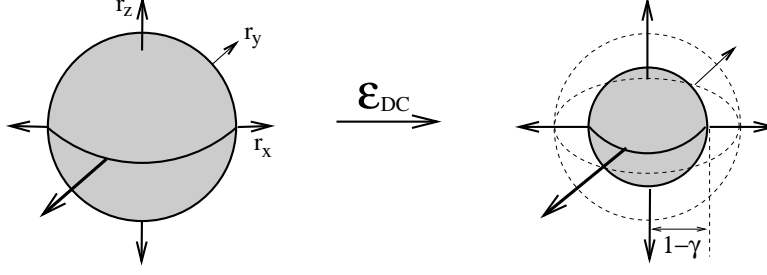


Figure 4: Action of the depolarizing channel operation on the Bloch ball.

- $X^2 = Y^2 = Z^2 = I$, implying that they are all unitary, as well.
- The set $\{X, Y, Z, I\}$ form a complete basis for all 2×2 matrices.
- $XY = -YX = iZ$, $YZ = -ZY = iX$, and $ZX = -XZ = iY$.

X is also called the “bit flip” operator, Z is called the “phase flip” operator, and Y the “bit+phase flip” operator.

Exercise 7.14 Prove (230) by writing $\rho = (I + \vec{r} \cdot X)/2$ with $|\vec{r}| \leq 1$.

Using (230) we have

$$\mathcal{E}_{\text{DC}}(\rho) = \left(1 - \frac{3}{4}\gamma\right) \rho + \frac{1}{4}\gamma (X\rho X + Y\rho Y + Z\rho Z) = \sum_k E_k \rho E_k^\dagger$$

where

$$\{E_k\} = \left\{ \frac{1}{2}\sqrt{4-3\gamma} I, \frac{1}{2}\sqrt{\gamma} X, \frac{1}{2}\sqrt{\gamma} Y, \frac{1}{2}\sqrt{\gamma} Z \right\}.$$

We check that (229) is indeed satisfied by computing $\sum_k E_k^\dagger E_k = \frac{1}{4}(4-3\gamma+\gamma+\gamma+\gamma) I = I$.

Thus the depolarizing channel has the interpretation of doing:

- nothing (I) with probability $(\frac{1}{2}\sqrt{4-3\gamma})^2 = 1 - \frac{3}{4}\gamma$,
- a bit flip (X) " " $(\sqrt{\gamma}/2)^2 = \frac{1}{4}\gamma$,
- a phase flip (Z) " " $(\sqrt{\gamma}/2)^2 = \frac{1}{4}\gamma$,
- a bit+phase flip (Y) " " $(\sqrt{\gamma}/2)^2 = \frac{1}{4}\gamma$.

Note that if we do all four of these with equal probability ($\gamma = 1$) then *all* information is lost: $\mathcal{E}_{\text{DC}} = I/2$.

Amplitude damping channel or “energy loss”.

Suppose a qubit is realised by a physical system with two energy levels. Say $|0\rangle$ is the ground state with energy 0, and $|1\rangle$ is the excited state with energy ΔE . These might be two energy levels of an atom, for example. Often in interaction with the environment, the higher-energy state can decay to the lower one. For the atom this happens by the spontaneous emission of a photon. We can model this by saying that the environment is a 2-state system: \mathcal{H}_{env} has basis $\{|0_e\rangle, |1_e\rangle\}$, where $|0_e\rangle$ is the state with no photon, and $|1_e\rangle$ has 1 photon. Then the unitary interaction between the atom and the environment is described by

$$\begin{aligned} U(|0\rangle|0_e\rangle) &= |0\rangle|0_e\rangle && \text{(ground state does not decay),} \\ U(|1\rangle|0_e\rangle) &= \sqrt{1-\gamma}|1\rangle|0_e\rangle + \sqrt{\gamma}|0\rangle|1_e\rangle && \text{(excited state decays with amplitude } \sqrt{\gamma}\text{).} \end{aligned}$$

Suppose also that $U(|1\rangle|1_e\rangle) = |1\rangle|1_e\rangle$ (excited state doesn’t decay if there is already a photon). Then unitarity of U implies

$$U(|0\rangle|1_e\rangle) = \sqrt{1-\gamma}|0\rangle|1_e\rangle + \sqrt{\gamma}|1\rangle|0_e\rangle \quad \text{(spontaneous absorption).}$$

Thus we have a model of the complete interaction of the system+environment. Assuming the environment starts in the $|0_e\rangle$ state, we can then use the definition of quantum operation (227), to find

$$\begin{aligned} \mathcal{E}_{\text{AD}} &:= \text{tr}_{\mathcal{H}_{\text{env}}} (U [\rho \otimes |0_e\rangle\langle 0_e|] U^\dagger), \\ &= E_0 \rho E_0^\dagger + E_1 \rho E_1^\dagger \end{aligned}$$

with, in the $\{|0\rangle, |1\rangle\}$ basis,

$$E_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{pmatrix}, \quad E_1 = \begin{pmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{pmatrix}.$$

E_0 is the operation element describing no decay, while E_1 is the one describing the decay.

The physical interpretation of γ is thus as the probability of decay of the excited state. In real processes, we usually speak of probability of decay per unit time. For short times, Δt , the probability of decay is given by $\gamma = \Delta t/\tau$, where τ is some characteristic *decay time* of the system. For longer times, $t = N \cdot \Delta t$, the probability of no decay occurring is $1 - \gamma(t) \approx \lim_{N \rightarrow \infty} (1 - (t/N\tau))^N = e^{-t/\tau}$, implying that $\gamma(t) = 1 - e^{-t/\tau}$. In particular, for long times $\gamma \rightarrow 1$ in physical systems.

It is a good exercise to show that \mathcal{E}_{AD} acts on the Bloch ball by squishing it up towards the north pole ($|0\rangle$), as illustrated in figure 5.

Bit flip channel or “classical noise”.

The bit flip channel is like classical noise in which there is probability γ that $|0\rangle \leftrightarrow |1\rangle$ are exchanged:

- probability γ to flip: $|\psi\rangle \rightarrow X|\psi\rangle$,
- probability $1 - \gamma$ to not flip: $|\psi\rangle \rightarrow I|\psi\rangle$.

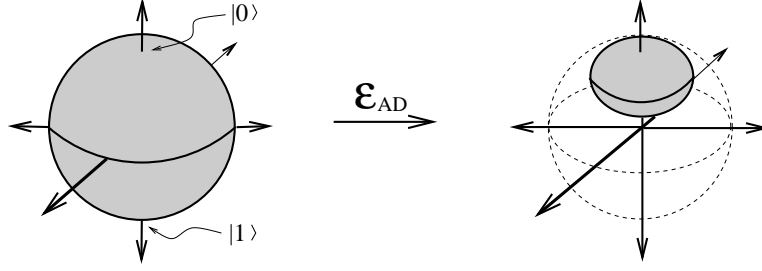


Figure 5: Action of the amplitude damping channel operation on the Bloch ball.

So we choose the operation elements $E_k = \{\sqrt{1-\gamma} I, \sqrt{\gamma} X\}$, giving

$$\rho \rightarrow \rho' = \mathcal{E}_{\text{BF}}(\rho) = (1-\gamma)\rho + \gamma X\rho X.$$

You can show that it acts on the Bloch ball by squeezing it towards a cigar along the x -axis, as in figure 6, where $|\pm\rangle := (|0\rangle \pm |1\rangle)/\sqrt{2}$.

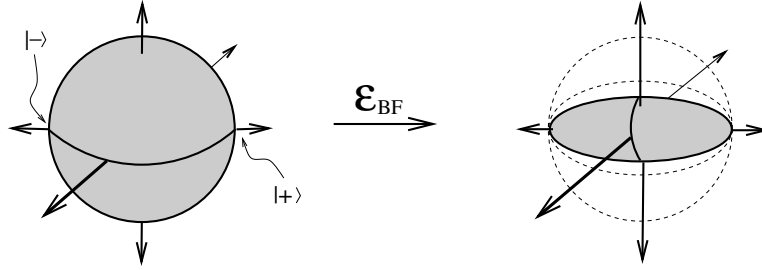


Figure 6: Action of the bit flip channel operation on the Bloch ball.

Phase flip channel or “decoherence”.

This channel is uniquely quantum-mechanical in its effects: it leaves the $|0\rangle$ and $|1\rangle$ pure states unchanged, and only affects the *phase* information in the quantum state:

$$\rho \rightarrow \rho' = \mathcal{E}_{\text{PF}}(\rho) = (1-\gamma)\rho + \gamma Z\rho Z.$$

You can show that it acts on the Bloch ball by squeezing it towards a cigar along the z -axis, as in figure 7.

In many physical implementations phase flip, or decoherence, is the main source of noise, mainly because $|0\rangle$ and $|1\rangle$ are realized as energy eigenstates of different energies. As in the discussion of energy loss, the phase flip probability γ is an exponentially saturating function of time,

$$\gamma \approx 1 - e^{-t/\tau_{\text{dec}}},$$

where τ_{dec} is the **decoherence time** of the system. The statement that phase flip noise is the typically the main source of noise is simply the statement that τ_{dec} is typically the shortest noise channel time scale. For instance, in many proposed physical realizations of qubits, energy conservation often makes the amplitude damping and depolarizing processes take place much more slowly than decoherence processes.

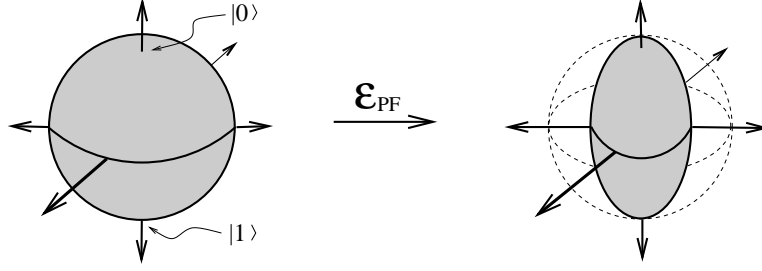


Figure 7: Action of the phase flip channel operation on the Bloch ball.

γ also typically grows with the size of the system, which is why large objects typically act classically: phase decoherence projects them onto classical states. (In terms of the Bloch ball, classical states correspond to the points just along the z -axis.)

General 1-qubit noise

You can show ([NC] section 8.3.2) that the action of the most general noise on a single qubit can be parameterized by 12 parameters. In the Bloch ball representation, if $\rho = \frac{1}{2}(I + \vec{r} \cdot \vec{X})$ and $\rho' = \frac{1}{2}(I + \vec{r}' \cdot \vec{X}) = \mathcal{E}(\rho)$, then $\vec{r}' = O_1 D O_2 \vec{r} + \vec{c}$ where O_i are independent rotation matrices, D is a diagonal scaling matrix, and \vec{c} is a constant translation vector.

8 Error correction for noisy channels

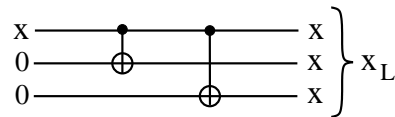
In this section we will discuss algorithms for correcting errors due to noise when sending (qu)bits down a wire.

8.1 Classical majority voting code

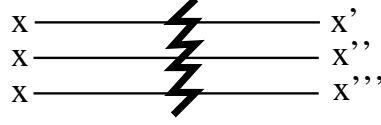
We first start by reviewing the classical situation. For simplicity, consider a symmetric model of noise in a 1-bit channel in which p is the probability of 0 flipping to 1 or the reverse. This is modeled by the classical probability evolution matrix (206) with $e = f = p$. We want to design a classical circuit which transmits the classical bit along this noisy channels, but reduces the probability of a bit flip.

A simple such algorithm is the **majority voting code**, consisting of the steps:

1. **Encode:** Triple the the input, *i.e.* map “0” to “000” and “1” to “111”. We will call 000 “logical 0” and denote it by 0_L ; similarly $111 := 1_L$ is logical 1. A reversible circuit which does the encoding step is



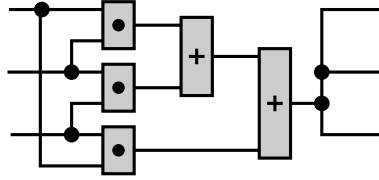
2. **Send:** Send the encoded signal (3 bits for one) down the wire where noise flips bits with probability p . We assume the noise acts independently on each bit:



3. Measure and correct: There are 8 possible outcomes once the noise has acted. Map these back to logical 1 or 0 by “majority voting”

$$\left. \begin{array}{cc} 000 & 111 \\ 001 & 110 \\ 010 & 101 \\ 100 & 011 \end{array} \right\} \longrightarrow \left\{ \begin{array}{cc} 000 & 111 \\ 000 & 111 \\ 000 & 111 \\ 000 & 111 \end{array} \right.$$

realized by the circuit



where “•” stands for an AND gate and “+” stands for an OR gate.

4. Decode: Send $0_L \rightarrow 0$ and $1_L \rightarrow 1$. In this case this just means we should read off any one of the outputs.

Now we can compute the probability of an overall bit flip. Since p is the probability of any of the three bits flipping in step 2, and since the probabilities are independent for each bit, we get

number of flips	0	1	2	3
probability	$(1 - p)^3$	$3p(1 - p)^2$	$3p^2(1 - p)$	p^3

Majority voting works (corrects the error) if there are only 0 or 1 bit flips; but it fails (returns a flipped bit) if there were 2 or 3 bit flips during transmission. Thus, the probability for an overall bit flip is the sum of the probabilities for 2 and 3 flips above, giving

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

The majority voting code is said to be **effective** when it reduces the overall bit-flip probability, *i.e.* when $p' < p$. By the last equation this means p has to satisfy $3p^2 - 2p^3 < p$ which is true for $p < p_{\text{th}} = 1/2$. So, as long as the error rate is less than the **threshold** value, $p_{\text{th}} = 1/2$, the majority voting code reduces the error rate.

Once a code is effective, it can clearly be reiterated to reduce the error rate by as much as is desired. For example, one could triple the logical bits to get logical-logical bits, $0_L \rightarrow 0_L 0_L 0_L := 0_{LL}$, and use majority voting on them to reduce the error rate even further to $p'' \approx 3(p')^2 \approx 27p^4$ (when $p \ll 1$).

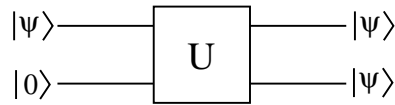
8.2 Shor's code

We want to translate the majority voting code to the quantum context. There are some obvious problems, though:

- Classical encoding involves *copying* the state. But the *no cloning* theorem says that this is impossible in quantum mechanics.
- Classically, *measuring* the signal in order to correct it by majority voting did not (in principle) introduce any new errors. But in quantum mechanics, measurement *irreducibly affects* the qubits, potentially destroying, *e.g.*, their entanglement with other qubits.
- Classically the error (noise) on a single bit was parametrized by just two numbers (the probabilities for 0 to flip to 1 and the probability for 1 to flip to 0). In quantum mechanics, on the other hand, we have just seen that single qubit errors (noise) are parameterized by *twelve* numbers. So there seems to be many more kinds of errors that have to be corrected for.

In the rest of this section we will see how Peter Shor's code (1994) overcomes all these problems.

First, let's review the *no-cloning theorem*, which states: There is no quantum circuit U ($UU^\dagger = I$) which duplicates an arbitrary initial state. That is, the circuit that does



for all $|\psi\rangle$ does not exist.

Proof: Suppose such a U did exist. Then

$$\begin{aligned} U(|\psi\rangle \otimes |0\rangle) &= |\psi\rangle \otimes |\psi\rangle, \\ U(|\phi\rangle \otimes |0\rangle) &= |\phi\rangle \otimes |\phi\rangle, \end{aligned}$$

for any two states $|\psi\rangle$ and $|\phi\rangle$. Take the inner product of these two equations. The left side gives

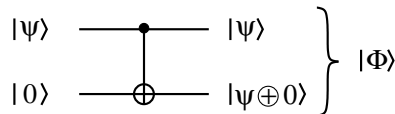
$$(\langle\phi|\langle 0|)U^\dagger U(|\psi\rangle|0\rangle) = (\langle\phi|\langle 0|)I(|\psi\rangle|0\rangle) = \langle\phi|\psi\rangle\langle 0|0\rangle = \langle\phi|\psi\rangle,$$

while the right side is

$$(\langle\phi|\langle\phi|)(|\psi\rangle|\psi\rangle) = \langle\phi|\psi\rangle\langle\phi|\psi\rangle.$$

Comparing left and right sides, we see that we must have $\langle\phi|\psi\rangle = 0$ or 1 , which means that either $|\phi\rangle = |\psi\rangle$ or $|\phi\rangle \perp |\psi\rangle$. But this is a contradiction, since $|\phi\rangle$ and $|\psi\rangle$ were assumed arbitrary.

But, this proof of the no-cloning theorem shows that we *can* clone orthogonal states. In other words, there can exist a circuit that behaves like the one shown above, but only for a basis of orthogonal states, $|\psi\rangle = |0\rangle$ or $|1\rangle$. For example, one such circuit is



But if $|\psi\rangle = a|0\rangle + b|1\rangle$, then the output is $|\Phi\rangle = a|00\rangle + b|11\rangle$, *not* $(a|0\rangle + b|1\rangle) \otimes (a|0\rangle + b|1\rangle)$. Therefore, this circuit *entangles* the output bits, but does not clone them.

Shor's insight was that error-correction could be accomplished using this kind of entanglement to encode qubits: "Fight [noise] entanglement with entanglement."

There are still the two other problems: the effects of measurements, and the many kinds of quantum noise. Let's see how the measurement problem can be dealt with by taking a *simple toy model* of quantum noise. In particular, *assume* that

$$\mathcal{E}(|\psi\rangle) = (1 - p)|\psi\rangle\langle\psi| + pX|\psi\rangle\langle\psi|X,$$

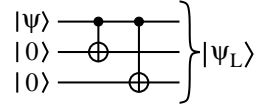
i.e., that the *only* error operator is X , the bit flip, with probability p . (This is completely unrealistic, but since it is similar to the classical noise case, it is a good toy problem to start on.)

3-qubit bit-flip (X) code

Encode:

$$\begin{aligned} |0\rangle &\rightarrow |0_L\rangle := |000\rangle, \\ |1\rangle &\rightarrow |1_L\rangle := |111\rangle, \end{aligned}$$

by entanglement:



Thus, if $|\psi\rangle = a|0\rangle + b|1\rangle$, then $|\psi_L\rangle = a|000\rangle + b|111\rangle$.

Noise acts on the output (entangled) qubits separately. This means that the quantum operator \mathcal{E} acts on the 3 qubits independently:

$$\begin{aligned} \mathcal{E}^3(|\psi_L\rangle\langle\psi_L|) &= \text{"}\mathcal{E} \otimes \mathcal{E} \otimes \mathcal{E}\text{"}(|\psi_L\rangle\langle\psi_L|) \\ &:= (1 - p)^3|\psi_L\rangle\langle\psi_L| \\ &\quad + p(1 - p)^2\{X_1|\psi_L\rangle\langle\psi_L|X_1 + X_2|\psi_L\rangle\langle\psi_L|X_2 + X_3|\psi_L\rangle\langle\psi_L|X_3\} \\ &\quad + p^2(1 - p)\{X_1X_2|\psi_L\rangle\langle\psi_L|X_1X_2 + X_1X_3|\psi_L\rangle\langle\psi_L|X_1X_3 \\ &\quad \quad + X_2X_3|\psi_L\rangle\langle\psi_L|X_2X_3\} \\ &\quad + p^3X_1X_2X_3|\psi_L\rangle\langle\psi_L|X_1X_2X_3. \end{aligned} \tag{231}$$

The first line below (231) expresses the $(1 - p)^3$ probability of no bit flips, the next line expresses the result of a single bit flipping, the next gives 2 bit flips, and the last the possibility of all three bits flipping. Here I have used a notation that will be very useful in what follows:

$$X_1 := X \otimes I \otimes I, \quad X_2 := I \otimes X \otimes I, \quad X_3 := I \otimes I \otimes X.$$

More generally, an expression like $X_iZ_jX_kY_\ell$ will mean the tensor product of 1-qubit operators with X acting on the i th qubit, Z on the j th, X on the k th, Y on the ℓ th, and I on all the rest.

Exercise 8.1 Write out the 8×8 density matrix $\mathcal{E}^3(|\psi_L\rangle\langle\psi_L|)$ in the computational basis $\{|000\rangle, |001\rangle, |010\rangle, |100\rangle, |110\rangle, |101\rangle, |011\rangle, |111\rangle\}$ if $|\psi_L\rangle = a|000\rangle + b|111\rangle$.

The next step is to correct for the errors. Note that our original state $|\psi\rangle$ lived in a 2-dimensional (1 qubit) Hilbert space \mathcal{H} . $|\psi_L\rangle$ lives in an 8-dimensional (3 qubit) Hilbert space \mathcal{H}^3 , but $|\psi_L\rangle = a|000\rangle + b|111\rangle$ is always in the **code subspace** \mathcal{H}_c spanned by $\{|000\rangle, |111\rangle\}$. The central observation needed for correcting errors is that typical errors move $|\psi_L\rangle$ *out* of the code subspace \mathcal{H}_c into the larger \mathcal{H}^3 space.

For example, say that noise flipped the first bit: $|\psi_L\rangle \rightarrow X_1|\psi_L\rangle$. (By (231), this occurs with probability $p(1-p)^2$.) If $|\psi\rangle = a|0\rangle + b|1\rangle$, so that $|\psi_L\rangle = a|000\rangle + b|111\rangle$, then

$$X_1|\psi_L\rangle = aX_1|000\rangle + bX_1|111\rangle = a|100\rangle + b|011\rangle \notin \mathcal{H}_c.$$

In fact, this state is orthogonal to \mathcal{H}_c . Similarly, $X_2|\psi_L\rangle = a|010\rangle + b|101\rangle$ and $X_3|\psi_L\rangle = a|001\rangle + b|011\rangle$ are also orthogonal to \mathcal{H}_c , and to each other. Because the errors are orthogonal, it is easy to cook up a measurement that detects them *without modifying the un-errored state*.

In this case, one way to do it is to measure the operators Z_1Z_2 and Z_2Z_3 . The set of operators one measures to identify an error is called the **error syndrome**. To see why this error syndrome does the job, first consider

$$Z_1Z_2 = Z \otimes Z \otimes I = \begin{pmatrix} 1 & \\ & -1 \end{pmatrix} \otimes \begin{pmatrix} 1 & \\ & -1 \end{pmatrix} \otimes \begin{pmatrix} 1 & \\ & 1 \end{pmatrix},$$

and is easily seen to have eigenvalues ± 1 . Furthermore, \mathcal{H}_c , $X_1\mathcal{H}_c$, $X_2\mathcal{H}_c$, and $X_3\mathcal{H}_c$ are all *eigenspaces* of Z_1Z_2 :

$$\begin{aligned} Z_1Z_2|\psi_L\rangle &= Z_1Z_2(a|000\rangle + b|111\rangle) = a(+)(+)|000\rangle + b(-)(-)|111\rangle = +|\psi_L\rangle, \\ Z_1Z_2(X_1|\psi_L\rangle) &= Z_1Z_2(a|100\rangle + b|011\rangle) = a(-)(+)|100\rangle + b(+)(-)|011\rangle = -X_1|\psi_L\rangle, \\ Z_1Z_2(X_2|\psi_L\rangle) &= Z_1Z_2(a|010\rangle + b|101\rangle) = a(+)(-)|010\rangle + b(-)(+)|101\rangle = -X_2|\psi_L\rangle, \\ Z_1Z_2(X_3|\psi_L\rangle) &= Z_1Z_2(a|001\rangle + b|110\rangle) = a(+)(+)|001\rangle + b(-)(-)|110\rangle = +X_3|\psi_L\rangle. \end{aligned}$$

Similarly for Z_2Z_3 which gives

$$\begin{aligned} Z_2Z_3|\psi_L\rangle &= +|\psi_L\rangle, \\ Z_2Z_3(X_1|\psi_L\rangle) &= +X_1|\psi_L\rangle, \\ Z_2Z_3(X_2|\psi_L\rangle) &= -X_2|\psi_L\rangle, \\ Z_2Z_3(X_3|\psi_L\rangle) &= -X_3|\psi_L\rangle. \end{aligned}$$

This can be summarized in the **error syndrome table**:

Error op. elements	Error syndrome	
	Z_1Z_2	Z_2Z_3
I	+	+
X_1	-	+
X_2	-	-
X_3	+	-

where the rows are the errors (I means “no error”), and the columns are the error syndrome operators we measure, and the entries are the ± 1 eigenvalues on the code space \mathcal{H}_c acted on by the error operation element.

The net result of measuring $\{Z_1Z_2, Z_2Z_3\}$ will be to get two numbers, $\{\pm 1, \pm 1\}$ and to “collapse the wave function”, *i.e.*, to project the state onto the Z_1Z_2 and Z_2Z_3 eigenspace with those eigenvalues. *But since the errors $X_i|\psi_L\rangle$ and $|\psi_L\rangle$ itself are already in Z_1Z_2 and Z_2Z_3 eigenspaces, they are not changed by the projection.*

So, the procedure for correcting the errors is clear:

(1) Measure $\{Z_1Z_2, Z_2Z_3\}$.

(2a) If $\{Z_1Z_2, Z_2Z_3\} = \{+, +\}$, do nothing: $|\psi_L\rangle \rightarrow |\psi_L\rangle$.

(2b) If $\{Z_1Z_2, Z_2Z_3\} = \{+, -\}$, apply X_3 : $X_3|\psi_L\rangle \rightarrow |\psi_L\rangle$.

(2c) If $\{Z_1Z_2, Z_2Z_3\} = \{-, +\}$, apply X_1 : $X_1|\psi_L\rangle \rightarrow |\psi_L\rangle$.

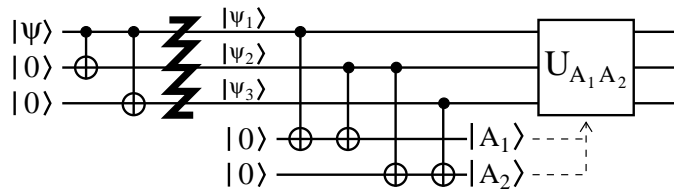
(2d) If $\{Z_1Z_2, Z_2Z_3\} = \{-, -\}$, apply X_2 : $X_2|\psi_L\rangle \rightarrow |\psi_L\rangle$.

If only a single qubit has flipped, we see that we recover the original state in this way.

Note that if *two* qubits flipped, then we do *not* recover the original state, and error correction fails. For example, say the noise takes $|\psi_L\rangle \rightarrow X_1X_2|\psi_L\rangle = a|110\rangle + b|001\rangle$. This is still a $\{Z_1Z_2, Z_2Z_3\}$ eigenstate, but with eigenvalues $\{+, -\}$. So it is mistaken for a single bit flip X_3 error, and our error-correction procedure gives $X_1X_2|\psi_L\rangle \rightarrow X_1X_2X_3|\psi_L\rangle \neq |\psi_L\rangle$. This is just like classical majority voting: it fails if 2 or more bits are flipped by the noise.

Therefore, just as in the classical case, the net probability of error after this correction procedure is $p' = 3p^2(1 - p) + p^3 = 2p^2 - 3p^3$, and therefore the correction is effective if $p' < p$, which occurs for $p < 1/2$.

We now want to represent this quantum error-correction algorithm as a quantum circuit. Also, real measurements are not ideal, so even though $X_i|\psi_L\rangle$ are in Z_jZ_k eigenspaces, a *real* quantum measurement (as opposed to an *ideal* or *non-destructive* one) will generally change $X_i|\psi_L\rangle$. (This is a statement not about quantum mechanics, but about human technological imperfection.) Thus we would like to modify our error-correction procedure so we do not have to do the measurements directly on our signal qubits. This is easily taken care of using **ancilla qubits**:



The jagged line in this circuit indicates where the noise acts. The two additional bottom lines represent the ancilla qubits. The dashed lines mean “measure the ancilla qubits in the computational basis” and the dashed arrow indicates that the error-correcting U gate that is applied depends on the result of that measurement. Finally, I have indicated states $|\psi_{1,2,3}\rangle$ after the noise is just for reference purposes: the state of the 3 signal qubits is really entangled, so is *not* a tensor product of three separate 1-qubit states.

This circuit works as follows, as you can easily read off:

If $|\psi_1\psi_2\rangle = |00\rangle$ or $|11\rangle$, then we measure $|A_1\rangle = |0\rangle$ and the Z_1Z_2 eigenvalue is $+1$.

If $|\psi_1\psi_2\rangle = |01\rangle$ or $|10\rangle$, then we measure $|A_1\rangle = |1\rangle$ and the Z_1Z_2 eigenvalue is -1 .

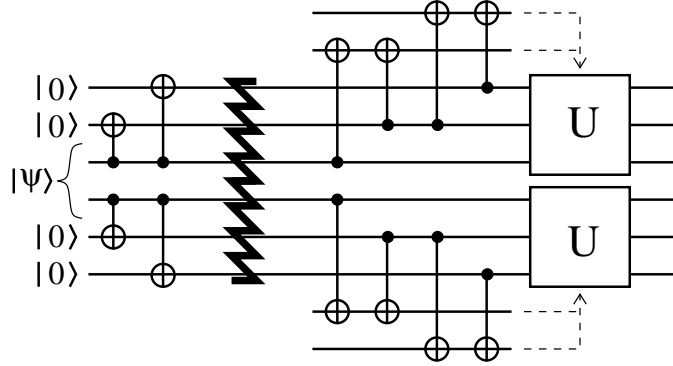
If $|\psi_2\psi_3\rangle = |00\rangle$ or $|11\rangle$, then we measure $|A_2\rangle = |0\rangle$ and the Z_2Z_3 eigenvalue is $+1$.

If $|\psi_2\psi_3\rangle = |01\rangle$ or $|10\rangle$, then we measure $|A_2\rangle = |1\rangle$ and the Z_2Z_3 eigenvalue is -1 .

Therefore, the error-correcting gate should be

$$U_{A_1A_2} = \begin{cases} I & \text{if } A_1 = 0, A_2 = 0, \\ X_1 & \text{if } A_1 = 1, A_2 = 0, \\ X_2 & \text{if } A_1 = 1, A_2 = 1, \\ X_3 & \text{if } A_1 = 0, A_2 = 1. \end{cases}$$

As the following circuit makes obvious, $|\psi\rangle$ (the input) could be entangled with any number of other qubits, and this entanglement would not be affected by the error-correction procedure. So, for example, the circuit for error-correcting bit-flip errors in two entangled qubits is simply the double of the 1-qubit circuit:



Exercise 8.2 Show that if p'_1 is the probability that the 1-qubit error-correction circuit fails to correct bit-flip noise errors, then the probability for the above 2-qubit error-correction circuit to fail is $p'_2 = 2p'_1 - (p'_1)^2$.

3-qubit phase-flip (Z) code

So far all this was only for correcting bit-flip channel errors. We have seen that there are many other kinds of quantum noise. For example, there is also the phase-flip channel (error operator = Z) which had no classical analog. Let's see how to correct for that.

So suppose $\mathcal{E}(|\psi\rangle) = (1 - p)|\psi\rangle\langle\psi| + pZ|\psi\rangle\langle\psi|Z$. Recall that $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ flips the sign of $|1\rangle$. How do we correct for this error? Recall $X = HZH$ where $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ is the Hadamard gate. Define the $|\pm\rangle$ basis by

$$|\pm\rangle := \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle),$$

and note that $\{|\pm\rangle\}$ is the X eigenbasis, and that Z acts on it as

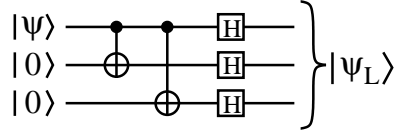
$$Z|+\rangle = |-\rangle, \quad Z|-\rangle = |+\rangle.$$

In other words, Z acts in the $\{|\pm\rangle\}$ basis like X does in the $\{|0\rangle, |1\rangle\}$ basis. This makes it clear how to proceed.

Encode:

$$\begin{aligned} |0\rangle &\rightarrow |0_L\rangle := |+++\rangle, \\ |1\rangle &\rightarrow |1_L\rangle := |--\rangle, \end{aligned}$$

by entanglement:



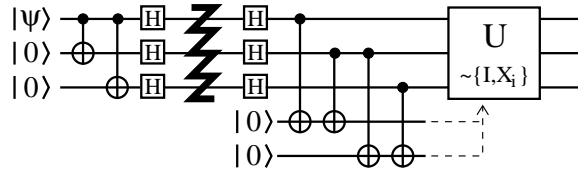
Thus, if $|\psi\rangle = a|0\rangle + b|1\rangle$, then $|\psi_L\rangle = a|+++\rangle + b|--\rangle$.

Decode: after transmission, to switch back to the $\{|000\rangle, |111\rangle\}$ basis of \mathcal{H}_c , just act with Hadamard gates on the three qubits. The net effect of this is to sandwich the noise between two Hadamard gates (on each qubit), thus effectively changing the Z_i error operations to $X_i = HZ_iH$. This has simply “rotated” the error from phase-flips to bit-flips. So the rest of the error correction is the same as in the bit-flip case:

Measure error syndrome: $\{Z_1Z_2, Z_2Z_3\}$ just as before.

Correct single errors: again in the same way by applying X_i to the i th bit flip.

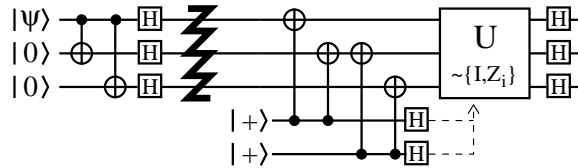
The complete circuit thus looks like



Note that we can make equivalent circuits by moving the Hadamard gates around using the identities $H^2 = I$, $HXH = Z$, as well as the 2-qubit identity

$$\begin{array}{c} \text{---} \text{H} \text{---} \bullet \text{---} \text{H} \text{---} \\ \text{---} \text{H} \text{---} \oplus \text{---} \text{H} \text{---} \end{array} = \begin{array}{c} \text{---} \oplus \text{---} \\ \text{---} \bullet \text{---} \end{array}$$

Exercise 8.3 Use the above identities to show that the phase-flip error-correcting circuit is equivalent to



9-qubit X and Z code (Shor code)

Now we can finally describe **Shor’s code**: a simple code that simultaneously protects against bit-flip *and* phase-flip errors. We will then see the pleasant surprise of quantum error correction: a code that corrects bit-flip (X) and phase-flip (Z) errors, actually automatically corrects *arbitrary* single-qubit quantum errors!

Encode: in 2 steps. First, encode

$$\begin{aligned} |0\rangle &\rightarrow |+++\rangle, \\ |1\rangle &\rightarrow |--\rangle, \end{aligned}$$

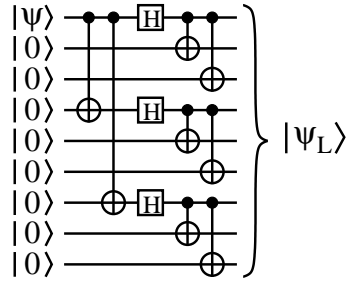
then encode each of these three qubits by

$$\begin{aligned} |+\rangle &\rightarrow \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle), \\ |-\rangle &\rightarrow \frac{1}{\sqrt{2}}(|000\rangle - |111\rangle). \end{aligned}$$

The net result is a 9-qubit encoding:

$$\begin{aligned} |0\rangle &\rightarrow |0_L\rangle := \frac{1}{2\sqrt{2}}(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle), \\ |1\rangle &\rightarrow |1_L\rangle := \frac{1}{2\sqrt{2}}(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle). \end{aligned}$$

This is accomplished by the circuit



So we are encoding 1 qubit $\in \mathcal{H}$ in a 2-dimensional code subspace $\mathcal{H}_c \subset \mathcal{H}^9$ of a $2^9 = 512$ -dimensional space! So there should be plenty of room for the 18 types of noise (9 X 's and 9 Z 's) we are interested in correcting to appear in orthogonal subspaces of \mathcal{H}^9 , and to find error syndromes with these subspaces as eigenspaces.

The bit-flip errors: X_1, X_2, \dots, X_9 are easy to fix. For example,

$$\begin{aligned} X_5|0_L\rangle &= \frac{1}{2\sqrt{2}}(|000\rangle + |111\rangle)(|010\rangle + |101\rangle)(|000\rangle + |111\rangle), \\ X_5|1_L\rangle &= \frac{1}{2\sqrt{2}}(|000\rangle - |111\rangle)(|010\rangle - |101\rangle)(|000\rangle - |111\rangle), \end{aligned}$$

so is an eigenstate of Z_4Z_5 and Z_5Z_6 , with

$$\begin{aligned} Z_4Z_5(X_5|\psi_L\rangle) &= -X_5|\psi_L\rangle, \\ Z_5Z_6(X_5|\psi_L\rangle) &= -X_5|\psi_L\rangle. \end{aligned}$$

So if we measure $Z_4Z_5 = Z_5Z_6 = -1$, we know that the X_5 bit-flip error occurred, and can correct it. In general, measuring $\{Z_1Z_2, Z_2Z_3, Z_4Z_5, Z_5Z_6, Z_7Z_8, Z_8Z_9\}$ will detect all single bit-flip errors.

The phase-flip errors: Z_1, Z_2, \dots, Z_9 are actually easier to fix in this encoding. Notice that the phase-flips $\{Z_1, Z_2, Z_3\}$ all have the same effect on \mathcal{H}_c , *e.g.*,

$$\begin{aligned} Z_1|0_L\rangle &= \frac{1}{2\sqrt{2}} (|000\rangle - |111\rangle) (|000\rangle + |111\rangle) (|000\rangle + |111\rangle), \\ Z_2|0_L\rangle &= \frac{1}{2\sqrt{2}} (|000\rangle - |111\rangle) (|000\rangle + |111\rangle) (|000\rangle + |111\rangle). \end{aligned}$$

Thus we only have to detect a phase flip on bits 1–3, 4–6, or 7–9. Just as X_1X_2 and X_2X_3 would do the job for bits 1–3, so $\{X_1X_2X_3X_4X_5X_6, X_4X_5X_6X_7X_8X_9\}$ are sufficient to detect all single phase-flips.

Thus, the set

$$\{Z_1Z_2, Z_2Z_3, Z_4Z_5, Z_5Z_6, Z_7Z_8, Z_8Z_9, X_1X_2X_3X_4X_5X_6, X_4X_5X_6X_7X_8X_9\}$$

are the error syndrome to tell us if a single bit-flip or phase-flip error has occurred. Measuring these operators will not disturb the state because all single-qubit bit-flip or phase-flip errors are eigenstates of *all* these operators. This can be checked explicitly. Or, since it is already clear that bit-flips are eigenstates of the Z_iZ_j 's and phase-flips of the $\prod_i X_i$, we just have to check that they are also eigenstates of each other. A necessary condition for this is that the Z^2 and X^6 operators commute, so that they have simultaneous eigenvalues. The commutators indeed vanish, since

$$[Z_1Z_2, X_1X_2] = Z_1X_1Z_2X_2 - X_1Z_1X_2Z_2 = (iY_1)(iY_2) - (-iY_1)(-iY_2) = 0,$$

and similarly for the others.

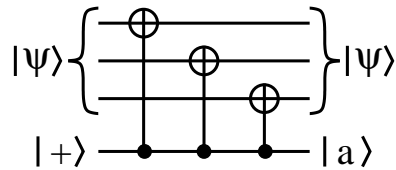
So, our error-correction syndrome table is:

Error op. elements	Error syndrome							
	Z_1Z_2	Z_2Z_3	Z_4Z_5	Z_5Z_6	Z_7Z_8	Z_8Z_9	$X_1 \cdots X_6$	$X_4 \cdots X_9$
I								
X_1	—							
X_2	—	—						
X_3		—						
X_4			—					
X_5			—	—				
X_6				—				
X_7					—			
X_8					—	—		
X_9						—		
Z_1, Z_2, Z_3							—	
Z_4, Z_5, Z_6							—	—
Z_7, Z_8, Z_9								—
$iY_1 = X_1Z_1$	—						—	
$iY_2 = X_2Z_2$	—	—					—	
$iY_3 = X_3Z_3$		—					—	
$iY_4 = X_4Z_4$			—				—	—
$iY_5 = X_5Z_5$			—	—			—	—
$iY_6 = X_6Z_6$				—			—	—
$iY_7 = X_7Z_7$					—			—
$iY_8 = X_8Z_8$					—	—		—
$iY_9 = X_9Z_9$						—		—

All the eigenvalues are ± 1 ; we only show the ones which are -1 for clarity. (The last 9 lines with the Y_i 's will be discussed below).

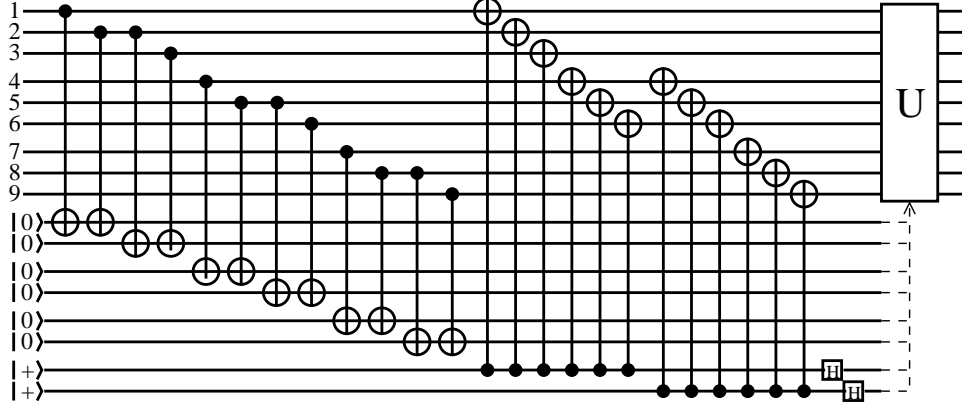
The circuit for making the measurement and correcting the errors is not hard to find.

Exercise 8.4 Show that the following circuit



gives the output shown with $|a\rangle = |\pm\rangle$ if $|\psi\rangle \in \left\{ \frac{1}{\sqrt{2}}(|000\rangle \pm |111\rangle), \frac{1}{\sqrt{2}}(|001\rangle \pm |110\rangle), \frac{1}{\sqrt{2}}(|010\rangle \pm |101\rangle), \frac{1}{\sqrt{2}}(|100\rangle \pm |011\rangle), \right\}$, where the plus or minus signs are correlated.

Using the result of the above exercise, it is straight forward to check that the circuit



does the job, where U means “act with the error operator determined by the values of the measurements, as given by the table on the previous page.”

The Y errors: So far we have shown that we can correct for X and Z errors. But there are many other errors. We will now show that Shor’s code also automatically corrects for Y errors.

Recall that $iY = ZX$, so up to an overall phase a Y error is just an X error followed by a Z error (on the same qubit!). Our error-correcting code worked for X_j and Z_j errors because the $X_j\mathcal{H}_c$ and $Z_j\mathcal{H}_c$ subspaces were all eigenspaces of the error syndromes $\{Z_1Z_2, \dots, X_4 \cdots X_9\} := \{F_n\}$ operators:

$$F_n X_j \mathcal{H}_c = \pm X_j \mathcal{H}_c \quad \text{and} \quad F_n Z_j \mathcal{H}_c = \pm Z_j \mathcal{H}_c. \quad (232)$$

Furthermore, it is also true that

$$F_n X_j = \pm X_j F_n \quad \text{and} \quad F_n Z_j = \pm Z_j F_n \quad (233)$$

as operator equations (*i.e.*, acting on any state). (In fact, since $F_n \mathcal{H}_c = +\mathcal{H}_c$, (232) follows from (233).) Equation (233) is true by virtue of the algebra of Pauli matrices. In particular, it follows from the easily checked operator relations $X_j Z_j = -Z_j X_j$, $X_i X_j = X_j X_i$, $Z_i Z_j = Z_j Z_i$ for any i, j , and $X_i Z_j = Z_j X_i$ for $i \neq j$.

Now it is easy to prove that $Y_j \mathcal{H}_c$ are also eigenspaces of the syndrome operators F_n by using (233):

$$F_n Y_j \mathcal{H}_c = i F_n Z_j X_j \mathcal{H}_c = \pm i Z_j X_j F_n \mathcal{H}_c = \pm i Z_j X_j \mathcal{H}_c = \pm Y_j \mathcal{H}_c.$$

The resulting error syndrome table for the Y_j is shown on page 58.

Exercise 8.5 Consider the state $|\chi\rangle := X_1 Z_1 |0_L\rangle = \frac{1}{2\sqrt{2}} (|100\rangle - |011\rangle) (|000\rangle + |111\rangle) (|000\rangle + |111\rangle)$. (*I.e.*, this state has suffered a combined bit- and phase-flip error on qubit 1.) Show that, indeed, $Z_1 Z_2 |\chi\rangle = -|\chi\rangle$, $Z_2 Z_3 |\chi\rangle = |\chi\rangle$, and $X_1 X_2 X_3 X_4 X_5 X_6 |\chi\rangle = -|\chi\rangle$.

General 1-qubit errors: We now show that Shor’s code actually corrects *all possible* 1-qubit errors. Recall that the general error is

$$\mathcal{E}(|\psi\rangle) = \sum_k E_k |\psi\rangle \langle \psi| E_k^\dagger \quad \text{such that} \quad \sum_k E_k^\dagger E_k = I,$$

which we saw could be interpreted as saying that the error $|\psi\rangle \rightarrow E_k|\psi\rangle$ occurs with probability $p_k = \langle\psi|E_k^\dagger E_k|\psi\rangle$. The most general error operator on a single qubit is a 2×2 matrix, so can be written as a linear combination of $\{I, X, Y, Z\}$:

$$E_k = e_{k0}I + e_{kx}X + e_{ky}Y + e_{kz}Z$$

where $\{e_{k0}, e_{kx}, e_{ky}, e_{kz}\}$ are complex numbers. That means that the most general possible error is a linear combination of the “errors” I , X , Z , and $iY = XZ$. (The I “error” is actually no error at all.)

Say qubit 1 in the state $|\psi_L\rangle$ in the Shor encoding experiences the general error $(E_k)_1 = E_k \otimes I^{\otimes 8}$. Then *measuring* the error syndrome operators $\{F_n\} = \{Z_1Z_2, X_1 \cdots X_6, \dots\}$ will project the total state

$$(E_k)_1|\psi_L\rangle = e_{k0}|\psi_L\rangle + e_{kx}X_1|\psi_L\rangle + e_{ky}Y_1|\psi_L\rangle + e_{kz}Z_1|\psi_L\rangle$$

onto the orthogonal eigenspaces of the measured F_n results. For example, say we measure $Z_1Z_2 = -1$ and $X_1X_2 \cdots X_6 = -1$, and all the rest of the $F_n = +1$. This is the error syndrome for a Y_1 error. The eigenspace of this measurement is $Y_1\mathcal{H}_c$, which is orthogonal to all the other errors (because they are F_n eigenspaces), and in particular to \mathcal{H}_c , $X_1\mathcal{H}_c$, and $Z_1\mathcal{H}_c$. Thus, after the measurement, the total state will be projected onto $Y_1\mathcal{H}_c$:

$$(E_k)_1|\psi_L\rangle \rightarrow \frac{P_{Y_1\mathcal{H}_c}(E_k)_1|\psi_L\rangle}{\|P_{Y_1\mathcal{H}_c}(E_k)_1|\psi_L\rangle\|} = \frac{e_{ky}Y_1|\psi_L\rangle}{\|e_{ky}Y_1|\psi_L\rangle\|} = Y_1|\psi_L\rangle,$$

Then when we apply the error correction, Y_1 , to this state according to the error-syndrome table on page 99 we get

$$Y_1(Y_1|\psi_L\rangle) = Y_1^2|\psi_L\rangle = |\psi_L\rangle,$$

and we have corrected the error! It should be clear that there was nothing special about this example: precisely the same computation works for every error syndrome eigenvalues measured. This constitutes a proof that Shor’s code corrects *any* 1-qubit error.

8.3 Generalizations

Though Shor’s code does the job, it is useful to generalize and abstract a method for describing error-correcting codes efficiently.

What is the smallest possible code?

The number of qubits used to do the encoding, n , is used to label quantum codes. For example, Shor’s code is a 9-qubit code. We want to know how many qubits n are necessary to protect against 1-qubit errors.

To correct arbitrary errors we need enough “room” in the total Hilbert space so that all errors will be in orthogonal subspaces. Say we have encoded 1 qubit with n qubits. Then $\dim\mathcal{H}_c = 2$ and $\dim\mathcal{H} = 2^n$. For example, for the Shor code, $n = 9$, so $\dim\mathcal{H} = 2^9 = 512$ is the total encoding Hilbert space, while the **logical subspace**, or **quantum code subspace**, has $\dim\mathcal{H}_c = 2$.

There are $3n$ possible 1-qubit errors: X, Y, Z for each of the n encoding qubits. There is also the possibility of “no error”: the I operator on all qubits. Thus there are a total of $1 + 3n$ independent error operators, E_n .

We want to choose $\mathcal{H}_c \subset \mathcal{H}$ such that $E_a \mathcal{H}_c$ are orthogonal for all a . Thus we must have $(\dim \mathcal{H}_c) \cdot (\#E_a) \leq \dim \mathcal{H}$, which implies $2(1 + 3n) \leq 2^n$, or

$$n \geq 5.$$

Thus the minimum number of qubits needed is 5. Such a code has been found, and is described in the text.

(Note that the same argument works for classical linear codes, except there is only one independent error—bit flip—instead of 3, giving $2 \cdot (1 + n) \leq 2^n$ which implies $n \geq 3$, with $n = 3$ the simple majority-voting code.)

It is easy to extend the above bound to more general codes. Say we encoded k qubits with n qubits, so $\dim \mathcal{H}_c = 2^k$ and $\dim \mathcal{H} = 2^n$, and we want a code to correct all errors on up to t qubits simultaneously. Then, the number of independent errors is

$$N(t) = \sum_{j=0}^t 3^j \binom{n}{j},$$

since there are $\binom{n}{j}$ ways to choose j qubits out of n that are affected by errors, and on each those j qubits there are 3 independent errors (X, Y , or Z). So, for all errors to be orthogonal in \mathcal{H} , we need $(\dim \mathcal{H}_c) \cdot N(t) \leq \dim \mathcal{H}$, which implies

$$\sum_{j=0}^t 3^j \binom{n}{j} \leq 2^{n-k}.$$

This is called the **quantum Hamming bound**.

The derivation of the quantum Hamming bound assumed the code was **nondegenerate** (which we won’t define here; see [NC]). It turns out, though, that you can’t defeat the Hamming bound using degenerate codes. Indeed, one can prove the **quantum Singleton bound**,

$$n \geq k + 2(d - 1),$$

which is true for all codes. Here we have introduced

$$d := 2t + 1,$$

which is called the **code distance**.

One uses the same notation for quantum codes as one does for **classical linear codes**. That is, we speak of an “ $[n, k, d]$ code” when it corrects $(d - 1)/2$ errors on k qubits by encoding them with n qubits. So, for example, Shor’s code is a $[9, 1, 3]$ code.

Conditions for error recovery

Theorems 10.1 and 10.2 of [NC] characterize when error recovery is possible. They are equivalent to the following

Theorem: Given a set of linearly independent error operators $\{E_a\}$ and an orthonormal basis $\{|i\rangle\}$ of the code space \mathcal{H}_c , then the E_a can be corrected if and only if

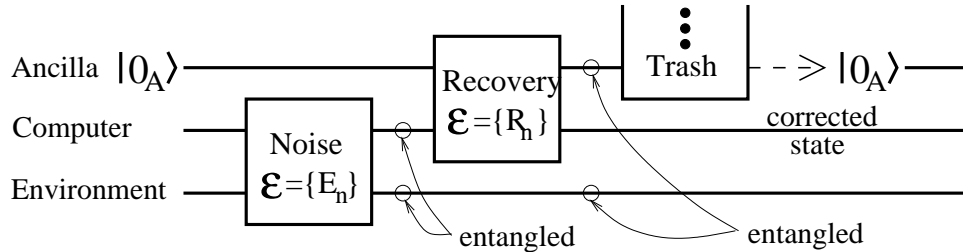
$$\langle j|E_a^\dagger E_b|i\rangle = C_{ab}\delta_{ij} \quad (234)$$

for all a, b, i, j where the C_{ab} are some complex numbers independent of i and j . (Note that $C_{ab}^* = C_{ba}$.)

The proof of this theorem is interesting because it is constructive: it actually gives a method for correcting errors, and gives some physical insight into how error correction works. It basically shows how, given (234), one can choose a basis for the error operators which diagonalizes C_{ab} , and in this basis how one can unambiguously diagnose the errors by performing orthogonal measurements. Arbitrary linear combinations of the errors E_a are also automatically recovered by the same arguments that we used for the Shor code.

Since the proof of this theorem is really just a formalization of what we did to check that the Shor code worked, we will not reproduce the details here; see the discussion in [NC] if you are interested. However a couple of physical points are made clearer in the general proof which were not so clear in our discussion of the Shor code:

1. In the proof of the theorem, the error recovery is modelled as a general quantum operation. In particular, no mention is made of whether the ancilla qubits were actually measured or not. Thus, the measurement step is *not necessary*. But the ancilla are necessary.
2. This gives the following general picture of how error correction works: Noise is entanglement with the environment, and increases the entropy of the system. We can't decrease the total entropy (eliminate the noise), but error recovery instead *shifts* it to entanglement of the environment with the ancilla qubits. To repeat the procedure, you have to “dump” the ancilla system, *i.e.*, reset it to a starting state, $|0_A\rangle$, unentangled with computer *or* environment. This is summarized in the figure below.



9 Fault-tolerant quantum computation

So far we have only discussed how to reduce noise in the transmission of qubits. But how about noise in the gates, *etc.*, used in a quantum computer and in the noise-correction algorithm itself? In the words of the inventors of fault-tolerant quantum computation (Mosca, Jozsa, Steane, Ekert):

At this point the reader still should not feel altogether happy about building this house of cards. Although we have introduced corrective measures, what if they themselves are faulty, as they must be in any real system?

It turns out (see the last sections of chapter 10 of [NC]) that analogs of the noisy channel error-correction idea described in the last section can be applied at every stage of a quantum computation: bit preparation, ancilla measurement, quantum gates, as well as simple qubit propagation. The net result is that if the sources of noise in all these steps is (1) sufficiently independent, and (2) sufficiently small (less than some finite threshold), then the error rate can be reduced arbitrarily at a cost of increasing the total number of bits and gates by a factor at most polynomial in the number of (non-error-corrected) gates. (This is the *threshold theorem* quoted on page 73, above.)

The final picture of quantum computation that emerges is summed up in metaphoric language by MJSE:

The “realistic” quantum computer looks very different from the idealized noise-free one. The latter is a silent shadowy beast at which we must never look until it has finished its computations, whereas the former is a bulky thing at which we “stare” all the time, via our error-detecting devices, yet in such a way as to leave unshackled the shadowy logical machine lurking within it.

10 The toric code

We now introduce another quantum error-correction scheme due to A. Kitaev (A. Kitaev, “Quantum computation: algorithms and error correction,” *Russian Math. Surveys* **52** (1997) 1191; A. Kitaev, in O. Hirota, A. Holevo, C. Caves (eds.), *Quantum Communication, Computing and Measurement*, (Plenum, NY, 1997).) called the *toric code* for reasons which will become obvious. It encodes 2 qubits with N qubits, so has code and total Hilbert spaces of dimensions

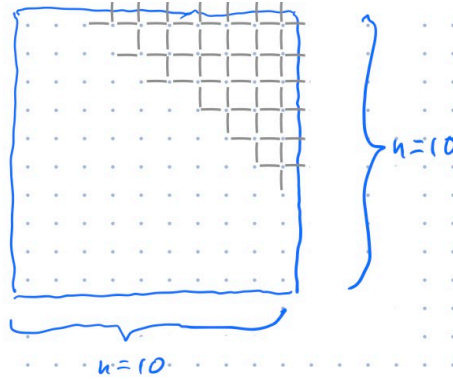
$$\dim(\mathcal{H}_c) = 2^2, \quad \text{and} \quad \dim(\mathcal{H}) = 2^N. \quad (235)$$

It corrects against up to $\sqrt{N/2}$ simultaneous arbitrary 1-qubit errors. Even though, for large N , it is far from saturating the quantum Singleton bound mentioned in section 8.3, it does have the highest known error threshold. Also, it is conceptually simple, and leads to the notion of *topological quantum computing* which is the idea of building a quantum computer which *automatically* (i.e., naturally, as part of its atomic structure) error-corrects! This idea will be briefly outlined in part IV below.

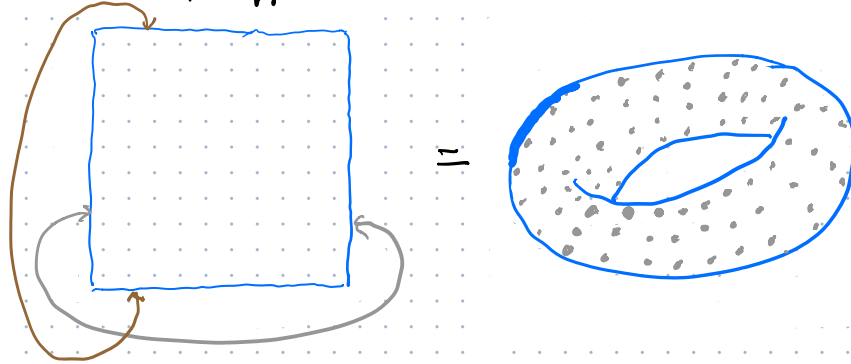
10.1 Setting up the computational basis

To describe the toric code, do not think in terms of a quantum circuit diagram, but rather of an arrangement of qubits in space:

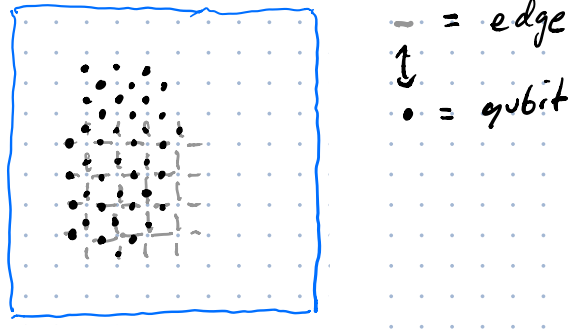
- Imagine an $n \times n$ square lattice of points, which we’ll call *vertices*. The line segments connecting nearest vertices are the *edges*, shown in gray. (I didn’t draw in all the edges.)



- Identify opposite sides of the square. Then the vertices can be thought of as living on a *torus* (e.g., the surface of a donut).



- Put one qubit on each *edge*. Thus the total number of qubits is $N = 2n^2$.



From now on, we just draw the lattice vertices with the understanding that the qubits are located on the edges.

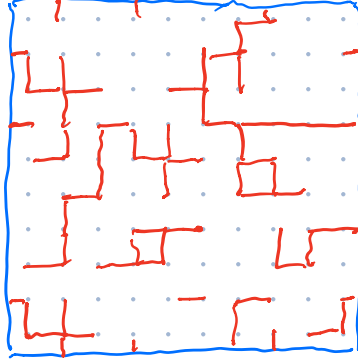
Label each edge by an index $\ell \in \{1, \dots, 2n^2\}$ in some way. Denote the qubit Hilbert space basis of the ℓ th edge by

$$\mathcal{H}_\ell = \{|0\rangle_\ell, |1\rangle_\ell\}. \quad (236)$$

We will notate these computational basis states as

$$|0\rangle_\ell \leftrightarrow |, \quad |1\rangle_\ell \leftrightarrow |, \quad (237)$$

i.e., we don't color the ℓ th edge for $|0\rangle$, and we color it red for $|1\rangle$. Thus a general computational basis state of $\mathcal{H} = \bigotimes_{\ell=1}^{2n^2} \mathcal{H}_\ell$ is



i.e., a set of 0's or 1's placed on each edge.

10.2 Error syndrome operators

Now we want to define a set of error syndrome operators,

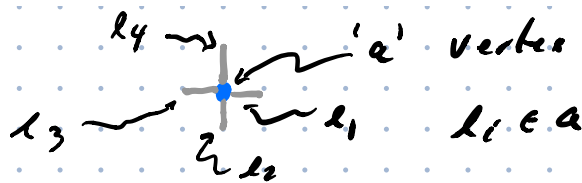
$$F_n \doteq \{V_a, P_a\}, \quad (238)$$

which are called *vertex operators* and *plaquette operators*.

Vertex operators V_a

Label the vertices by $a, b, \dots \in \{1, \dots, n^2\}$, and write

$$\ell \in a \quad \text{means} \quad \text{edge } \ell \text{ ends on vertex } a. \quad (239)$$



Then we define the vertex operator associated to vertex a by

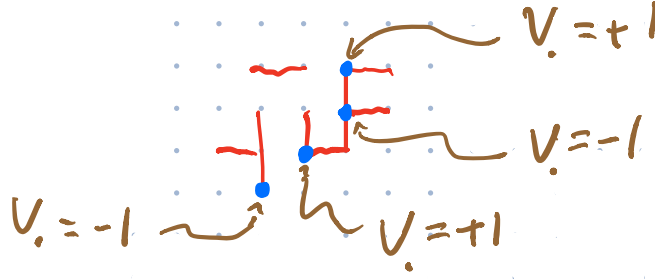
$$V_a \doteq \prod_{\ell \in a} Z_\ell = Z_{\ell_1} Z_{\ell_2} Z_{\ell_3} Z_{\ell_4}, \quad (240)$$

where Z_ℓ is the Z -gate acting on the ℓ th qubit.

It then easily follows (you should check this!) that

$$\begin{aligned} V_a V_b &= V_b V_a, \\ (V_a)^2 &= I, \\ \text{eigenvalues}(V_a) &\in \{\pm 1\}. \end{aligned} \quad (241)$$

In the computational basis a $+1$ eigenstate of V_a is any state with an even number of red edges ($|1\rangle$ states) $\in a$, while a -1 eigenstate is any state with an odd number of red edges $\in a$.



The computational basis is a simultaneous eigenbasis of all the V_a operators (which is possible because they all commute).

It then follows that

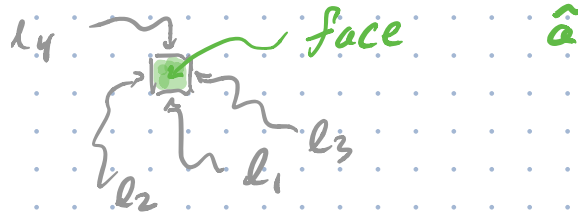
$$\prod_{a=1}^{n^2} V_a = I \quad (242)$$

since each Z_ℓ appears exactly twice in the product and squares to the identity. This is the only identity among the vertex operators, so we see that there are $n^2 - 1$ independent V_a operators.

Plaquette operators $P_{\hat{a}}$

Call each square made by four edges a *plaquette* (or *face*) of the lattice, label them by $\hat{a}, \hat{b}, \dots \in \{1, \dots, n^2\}$, and write

$$\ell \in \hat{a} \quad \text{means} \quad \text{edge } \ell \text{ is an edge of plaquette } \hat{a}. \quad (243)$$



Then we define the plaquette operator associated to plaquette \hat{a} by

$$P_{\hat{a}} \doteq \prod_{\ell \in \hat{a}} X_\ell = X_{\ell_1} X_{\ell_2} X_{\ell_3} X_{\ell_4}, \quad (244)$$

where X_ℓ is the Pauli X -gate acting on the ℓ th qubit.

It then easily follows that

$$\begin{aligned} P_{\hat{a}} P_{\hat{b}} &= P_{\hat{b}} P_{\hat{a}}, \\ (P_{\hat{a}})^2 &= I, \\ \text{eigenvalues}(P_{\hat{a}}) &\in \{\pm 1\}, \end{aligned} \quad (245)$$

and that

$$\prod_{a=1}^{n^2} P_{\hat{a}} = I \quad (246)$$

since each X_ℓ appears exactly twice in the product and squares to the identity. This is the only identity among the plaquette operators, so we see that there are $n^2 - 1$ independent $P_{\hat{a}}$ operators.

The $P_{\hat{a}}$ plaquette operator acts on the computational basis states by flipping the color of the edges of the \hat{a} plaquette and leaving all the other edges unchanged. For instance

$$\begin{aligned} P_{\hat{a}} \left(\begin{array}{c} \text{green square with red top and right edges} \end{array} \right) &= \begin{array}{c} \text{green square with red top and left edges} \end{array} \\ P_{\hat{a}} \left(\begin{array}{c} \text{green square with red top and left edges} \end{array} \right) &= \begin{array}{c} \text{green square with red top and right edges} \end{array} \end{aligned} \quad (247)$$

where the green-shaded plaquette is the \hat{a} plaquette. Thus the \hat{a} plaquette eigenstates are equal \pm superpositions of computational basis states with edges of the \hat{a} plaquette flipped. For example

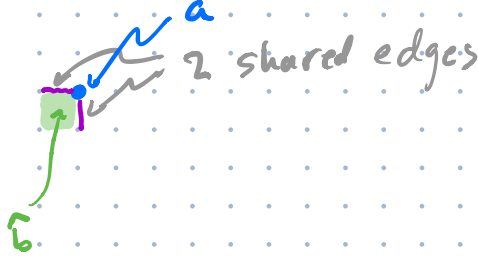
$$\begin{aligned} P_{\hat{a}} \frac{1}{\sqrt{2}} \left(\begin{array}{c} \text{green square with red top and right edges} \\ + \\ \text{green square with red top and left edges} \end{array} \right) &= + \frac{1}{\sqrt{2}} \left(\begin{array}{c} \text{green square with red top and left edges} \\ + \\ \text{green square with red top and right edges} \end{array} \right) \\ P_{\hat{a}} \frac{1}{\sqrt{2}} \left(\begin{array}{c} \text{green square with red top and right edges} \\ - \\ \text{green square with red top and left edges} \end{array} \right) &= - \frac{1}{\sqrt{2}} \left(\begin{array}{c} \text{green square with red top and left edges} \\ - \\ \text{green square with red top and right edges} \end{array} \right) \end{aligned} \quad (248)$$

Vertex and plaquette operators commute

That is,

$$[V_a, P_{\hat{b}}] = 0 \quad \text{for all } a \text{ and } \hat{b}. \quad (249)$$

If vertex a and plaquette \hat{b} have no edge in common, then this is obvious. If they share an edge, then they must share exactly 2 edges,



In this case

$$\begin{aligned} [V_a, P_{\hat{b}}] &\propto [Z_{\ell_1} Z_{\ell_2}, X_{\ell_1} X_{\ell_2}] = Z_{\ell_1} Z_{\ell_2} X_{\ell_1} X_{\ell_2} - X_{\ell_1} X_{\ell_2} Z_{\ell_1} Z_{\ell_2} \\ &= Z_{\ell_1} X_{\ell_1} Z_{\ell_2} X_{\ell_2} - X_{\ell_1} Z_{\ell_1} X_{\ell_2} Z_{\ell_2} = (-X_{\ell_1} Z_{\ell_1})(-X_{\ell_2} Z_{\ell_2}) - X_{\ell_1} Z_{\ell_1} X_{\ell_2} Z_{\ell_2} = 0. \end{aligned}$$

Since all $F_n \in \{V_a, P_{\hat{a}}\}$ commute, there is an orthonormal basis of simultaneous eigenstates of all the F_n . There are $(n^2 - 1) + (n^2 - 1) = 2n^2 - 2$ independent F_n , so there are 2^{2n^2-2} different possible patterns of ± 1 eigenvalues of the $\{F_n\}$. But there are total of $2n^2$ qubits, and so a 2^{2n^2} -dimensional state space. We conclude that each simultaneous eigenspace has dimension $2^2 = 4$.

Thus the $\{F_n\}$ are the error syndrome for 2 qubits encoded in $2n^2$ qubits.

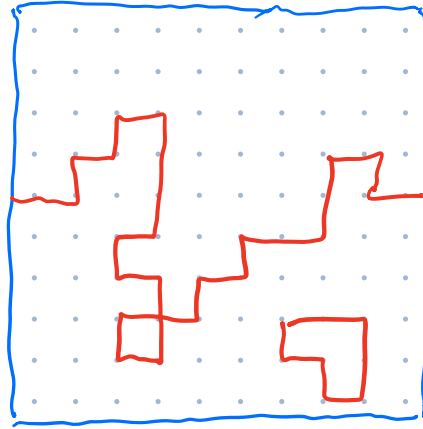
10.3 The code subspace

In particular, the subspace with all $V_a = P_a = +1$ is 4-dimensional. We call this subspace the *code subspace*. We now construct an orthonormal basis of the code subspace.

Since the $V_a = +1$ for all a , there must be an even number of red edges at each vertex,

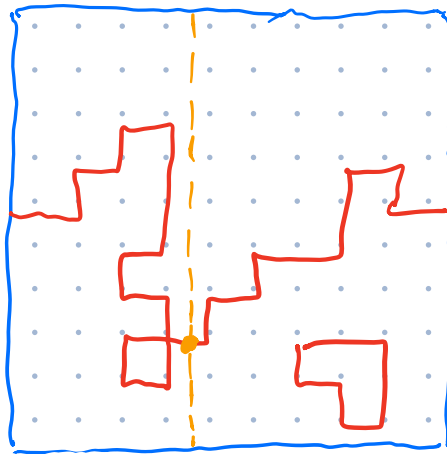
$$\in \left\{ \begin{array}{c} \bullet \\ \cdot \\ \cdot \\ \cdot \end{array} , \begin{array}{c} \color{red}{\mid} \\ \bullet \\ \cdot \\ \cdot \end{array} , \begin{array}{c} \color{red}{\mid} \\ \color{red}{\mid} \\ \bullet \\ \cdot \end{array} , \begin{array}{c} \color{red}{\mid} \\ \color{red}{\mid} \\ \color{red}{\mid} \\ \bullet \end{array} , \begin{array}{c} \color{red}{\mid} \\ \color{red}{\mid} \\ \color{red}{\mid} \\ \color{red}{\mid} \end{array} , \begin{array}{c} \color{red}{\mid} \\ \color{red}{\mid} \\ \color{red}{\mid} \\ \color{red}{\mid} \end{array} , \begin{array}{c} \color{red}{\mid} \\ \color{red}{\mid} \\ \color{red}{\mid} \\ \color{red}{\mid} \end{array} , \begin{array}{c} \color{red}{\mid} \\ \color{red}{\mid} \\ \color{red}{\mid} \\ \color{red}{\mid} \end{array} \right\}$$

Therefore for any computational basis state in the code subspace, the red edges form closed (but perhaps self-intersecting) loops,



(250)

Note that a closed red loop can loop all the way around the torus by, for example, going off to the right edge and then continuing at the corresponding point at the left edge, as in the figure. A property of configurations of closed loops is that any vertical or horizontal line wrapping the torus will intersect the loops either an even or odd number of times and this *parity* (even or odd) does not depend on where you place the lines. For example, in the configuration of the last figure, the vertical orange line at the fourth link from the left intersects the red loops just once,



(251)

If one shifted the orange line one link to the left, it would intersect the loops 5 times, still an odd number. We say that the configuration has vertical parity “ $n_x = 1$ ”, meaning any

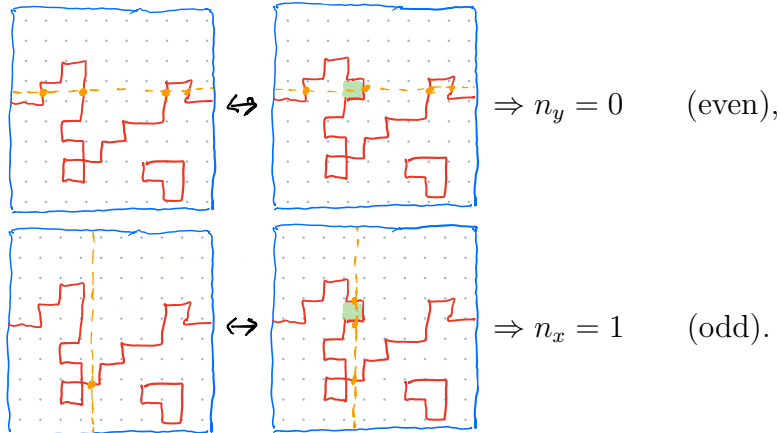
vertical line intersects it an odd number of times. This is equivalent to saying that the red loops *wrap* the horizontal direction a net odd number of times. You should check that in this example any horizontal line intersects the red loops an even number of times, or, equivalently, that the red loops wrap the vertical a net even number of times (namely 0). We say, then, that this configuration has “ $n_y = 0$ ”.

So far we have only written the (many) states which are +1 eigenstates of all the vertex operators. We now want to find superpositions of these states which are also +1 eigenstates of all the plaquette operators. Recall that $P_{\hat{a}} = +1$ eigenstates are given by equal superpositions of computational basis states with the red edges of the \hat{a} plaquette “flipped”, as in (247) and (248). Then we can build a state with $P_{\hat{a}} = +1$ for all \hat{a} by taking, for instance, the particular $V_a = +1$ state (250) and flipping *all* its plaquettes in all combinations, giving a state

$$|10\rangle_L = \frac{1}{2^{n^2/2}} \left(\text{[Diagram 1]} + \text{[Diagram 2]} + \text{[Diagram 3]} + \dots \right). \quad (252)$$

The green plaquettes indicate those where a flip relative to the original state is performed. All possible flips of 1, 2, 3, ..., up to n^2 plaquettes need to be performed and summed to get a state with $P_{\hat{a}} = +1$ for all \hat{a} . This means there are 2^{n^2} terms in the sum, hence the normalization factor.

The flipping operation does not change the n_x and n_y parities of the original state. For example,



Thus each state constructed by averaging all possible plaquette flips will have definite $n_x \in \mathbb{Z}_2$ and $n_y \in \mathbb{Z}_2$ parities. We denote these states by $|n_x, n_y\rangle_L$, as shown in (252). Furthermore, it is not too hard to see that these four possible values of (n_x, n_y) are the only properties of the subspace of all $V_a = +1$ states which are invariant under the flipping operation. It follows that, no matter which all $V_a = +1$ state we start with, the averaging over all flips constructs just one of four *code states*

$$|n_x, n_y\rangle_L \in \{|00\rangle_L, |01\rangle_L, |10\rangle_L, |11\rangle_L\}. \quad (253)$$

We can summarize this as saying that our 2-qubit code space is spanned by

$$\begin{aligned}
|00\rangle_L &\propto \left(\begin{array}{|c|} \hline \cdot & \cdot & \cdot & \cdot \\ \hline \end{array} + \begin{array}{|c|} \hline \cdot & \color{red}{\blacksquare} & \cdot & \cdot \\ \hline \end{array} + \dots \right) \sim \text{no wraps} \\
|10\rangle_L &\propto \left(\begin{array}{|c|} \hline \color{red}{\rule{1cm}{0.4pt}} & \cdot & \cdot & \cdot \\ \hline \end{array} + \begin{array}{|c|} \hline \cdot & \color{red}{\blacksquare} & \cdot & \cdot \\ \hline \end{array} + \dots \right) \sim \text{1-wrap x-dir} \\
|01\rangle_L &\propto \left(\begin{array}{|c|} \hline \cdot & \cdot & \color{red}{\rule{0.4pt}{1cm}} & \cdot \\ \hline \end{array} + \begin{array}{|c|} \hline \cdot & \color{red}{\blacksquare} & \cdot & \cdot \\ \hline \end{array} + \dots \right) \sim \text{1-wrap y-dir} \\
|11\rangle_L &\propto \left(\begin{array}{|c|} \hline \color{red}{\rule{1cm}{0.4pt}} & \color{red}{\rule{0.4pt}{1cm}} & \cdot & \cdot \\ \hline \end{array} + \begin{array}{|c|} \hline \cdot & \color{red}{\blacksquare} & \cdot & \cdot \\ \hline \end{array} + \dots \right) \sim \text{1-wrap x+y-dirs}
\end{aligned}$$

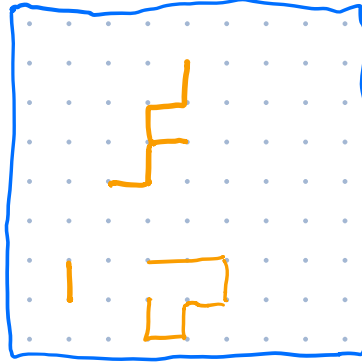
Exercise 10.1 Show that the code states are orthogonal.

10.4 Error correction

Just as we did for the Shor code, we will first look at how to correct X_ℓ and Z_ℓ errors, and then will move on to general errors.

X errors

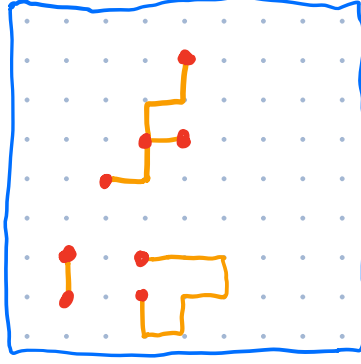
Let's say that an X error ("bit flip") occurs on bits $\ell \in XE$ where XE is a subset of all the edges. (" XE " stands for " X errors".) For example, the set of X errors XE might look like



where the edges where an X error occurs are colored orange. Then, if we start with a state $|\psi\rangle_L$ in the code subspace, the effect of the X errors is to move it to the state

$$|\psi\rangle_L \rightarrow \left(\prod_{\ell \in XE} X_\ell \right) |\psi\rangle_L. \quad (254)$$

We can detect these errors by measuring all the V_a 's. We find $V_a = -1$ on the vertices at the "ends" of strings of X errors:

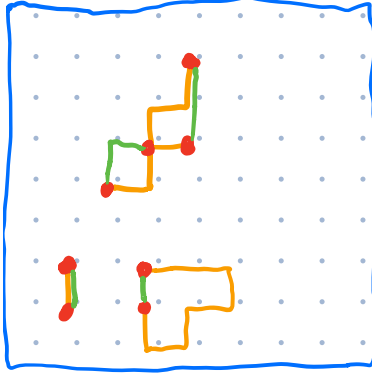


where the red dots denote vertices with $V_a = -1$. This is because $V_a = \prod_{\ell \in a} Z_\ell$, and $Z_\ell X_\ell = -X_\ell Z_\ell$. So if an odd number of X_ℓ 's act for $\ell \in a$, then

$$V_a(\prod X_\ell)|\psi\rangle_L = -(\prod X_\ell)V_a|\psi\rangle_L = -(\prod X_\ell)|\psi\rangle_L. \quad (255)$$

So any configuration of X error strings with end points is detectable by measuring the V_a 's.

We can then *correct* these errors by then acting with X_ℓ 's on *any* string of edges connecting the $V_a = -1$ vertices. For example



where the green edges are where we act with X_ℓ 's to correct the errors. This works because any *closed* path of X_ℓ 's acts as the *identity* on the code space:

$$\begin{array}{|c|} \hline \text{Grid with orange path} \\ \hline \end{array} \doteq \begin{array}{|c|} \hline \text{Empty grid} \\ \hline \end{array} \doteq |\psi\rangle_L. \quad (256)$$

To see this, look at any given computational basis state contributing to $|\psi\rangle_L$, e.g.,

$$|10\rangle_L \supset \begin{array}{|c|} \hline \text{Grid with red path} \\ \hline \end{array} \quad (257)$$

(where red edges denote qubits in state $|1\rangle$ as usual). Then closed strings of X_ℓ 's acting on this state gives

$$\left(\prod_{\ell} X_{\ell}\right)|10\rangle_L \supset \text{[Diagram: A square lattice with a red path and a yellow closed loop]} = \text{[Diagram: A square lattice with a red path]} \subset |10\rangle_L \quad (258)$$

which is just another term in $|\psi\rangle_L$ since it gives another configuration with a closed string of $|1\rangle$ (i.e., red) edges.

But this argument breaks down if the closed string of X_ℓ 's stretches across the whole $n \times n$ square of qubits — i.e., wraps around a cycle of the torus. For example,

$$\left(\prod_{\ell} X_{\ell}\right)|10\rangle_L \supset \text{[Diagram: A square lattice with a red path and a yellow closed loop wrapping around the right edge]} = \text{[Diagram: A square lattice with a red path]} \subset |11\rangle_L \quad (259)$$

which changes the logical state. Thus we cannot correct for X -errors if there are so many of them that they form a string around either (or both) cycles of the torus. But we need at minimum n such errors to wrap the torus, so *we can correct in this way up to n arbitrary simultaneous X errors.*

Z errors

This is very similar to the X errors. Call the set of bits where a Z error occurs ZE . Mark an edge $\ell \in ZE$ by drawing a purple interval bisecting it perpendicularly, e.g.,

$$\text{[Diagram: A square lattice with a purple path]} = \left(\prod_{\ell \in ZE} Z_{\ell}\right) |\psi\rangle_L. \quad (260)$$

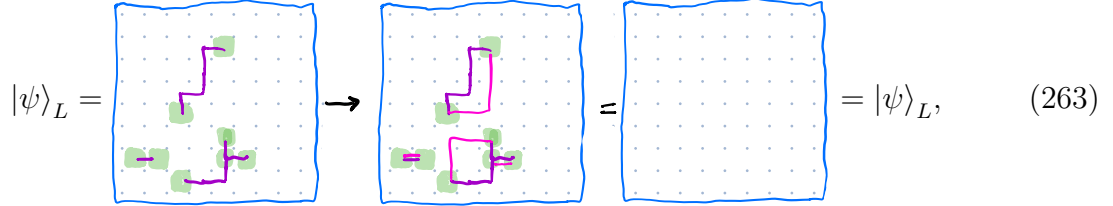
Detect these errors by measuring all the $P_{\hat{a}}$'s. You find $P_{\hat{a}} = -1$ on the plaquettes at the “ends” of purple strings of Z errors,

$$\text{[Diagram: A square lattice with a purple path and green shaded plaquettes at its ends]} \quad (261)$$

where the green plaquettes mark those where $P_{\hat{a}} = -1$. This is because if there is an odd number of Z_ℓ errors with $\ell \in \hat{a}$ then

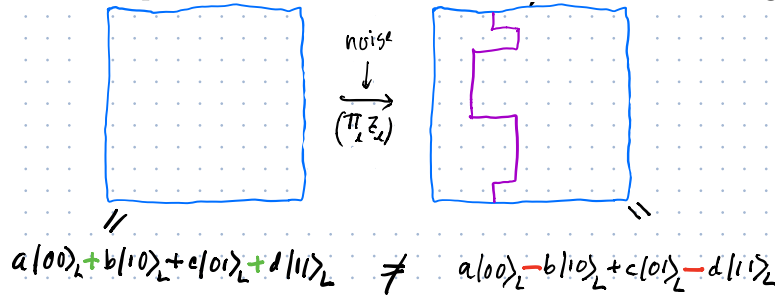
$$P_{\hat{a}}(\prod Z_\ell)|\psi\rangle_L = -(\prod Z_\ell)P_{\hat{a}}|\psi\rangle_L = -(\prod Z_\ell)|\psi\rangle_L. \quad (262)$$

Correct these errors by acting with *any* Z_ℓ 's to close the purple paths:



$$|\psi\rangle_L = \text{[Lattice with path]} \rightarrow \text{[Lattice with closed loop]} = |\psi\rangle_L, \quad (263)$$

where the last equality follows by a similar argument as for closed loops of X errors. Again, this doesn't work if the loop of Z errors stretches around one or both edges of the torus:



$$a|00\rangle_L + b|10\rangle_L + c|01\rangle_L + d|11\rangle_L \neq a|00\rangle_L - b|10\rangle_L + c|01\rangle_L - d|11\rangle_L$$

So, again, we can correct for all up to n simultaneous Z errors.

Y errors

Exactly as with the discussion of Shor's code, since $Y_\ell = iX_\ell Z_\ell$, and since we can correct for X_ℓ and Z_ℓ errors independently, we automatically correct for up to n simultaneous Y errors.

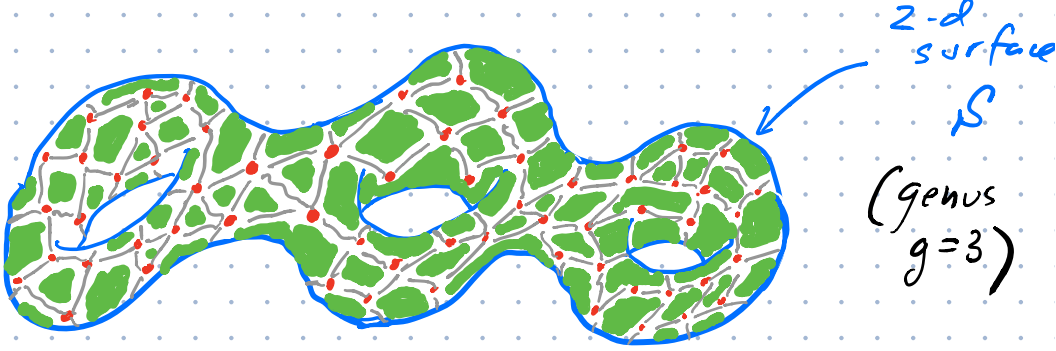
E_k errors

General errors are given by error operators $E_k = aI + bX + cY + dZ$. By the same argument as in Shor's code, if our code corrects for X , Y , and Z errors, it also automatically corrects for E_k errors.

10.5 Generalization

So, to summarize, we have shown how to encode 2 qubits in $2n^2$ qubits by thinking of them as corresponding to the edges of an $n \times n$ square lattice with opposite sides identified. Then the error syndromes and error correction procedures of this toric code were “local” on this lattice, meaning that they involved only products of 1-qubit gates acting on a few edges close to a given edge corresponding to the qubit where the noise occurred.

None of this actually depended on having a square lattice. It works just as well for any 2-dimensional “graph” of qubits, i.e., for any arrangement of qubits on a surface, e.g.,



where the gray edges correspond to the qubits, the red dots correspond to the vertex operators, and the green faces correspond to the plaquette operators. The code subspace with $P_a = V_a = +1$ for all vertices and plaquettes encodes $2g$ qubits where

$$\begin{aligned} g &= \text{genus of surface } S \\ &= \text{number of "handles" of } S \\ &= 1 - \frac{1}{2}\chi \end{aligned}$$

where χ is the *Euler characteristic* of S . The Euler characteristic can be defined in terms of the graph of qubits by

$$\chi \doteq V - E + F,$$

where V is the number of vertices, E is the number of edges (qubits), and F is the number of plaquettes (faces).

Thus this class of “surface” error-correction codes does not depend on the local structure of the graph of qubits, but only on the global *topology* (i.e., g or χ) of the surface. This insensitivity of these codes to the exact details of how the encoding qubits are arranged is the inspiration for the next topic.

Part IV

Topological Quantum Computation

There is a completely different strategy for defeating noise in quantum computers: instead of correcting for it, try to reduce it to insubstantial levels from the very beginning. Such a strategy is pursued in *topological quantum computation*, which tries to use certain special quantum systems in which quantum entanglement is very robust against noise. This was first proposed by A. Kitaev, “Fault-tolerant quantum computation by anyons,” *Ann. Phys. (NY)* **303** (2003) 2.

11 General idea

The basic idea is to find a system in which some of the quantum information is carried by degrees of freedom which are insensitive to localized perturbations of the system, but instead only care about global (large distance scale) configurations of the system. Such degrees of freedom are called topological degrees of freedom. An intuitive example is the number of handles (the “genus”) of a closed 2-dimensional surface: local deformations of the surface will change its exact shape, but not change the overall number of handles on the surface unless something drastic (discontinuous, like tearing the surface) happens.

Remarkably, systems with such topological degrees of freedom do exist in nature. For example, at low enough temperatures and high enough magnetic fields and at long distances, such a topological phase of matter is found in the *fractional quantum Hall effect* (at certain specific values of the magnetic field). We will see below that the toric code can be recast as an example of such a *topological material*.

The low-energy excitations of this system, its *quasiparticles*, act as very massive particles that are constrained to move in two spatial dimensions. A configuration of n well-separated quasiparticles carries with it a Hilbert (sub)space, \mathcal{H}_T , of topological degrees of freedom, analogous to the code subspace of the toric code. So a given such a configuration will “carry” a corresponding state $|\psi\rangle \in \mathcal{H}_T$ (which depends on the history of how the quasiparticle configuration was created). The quasiparticles can be measured and manipulated locally in space without affecting the associated topologically protected state $|\psi\rangle$.

But by manipulating the quasiparticle configuration (e.g., moving them around) by large amounts the state $|\psi\rangle$ can be changed. For example, the positions of a pair of identical quasiparticles can be interchanged by moving them along some paths γ which keeps them far apart. The result is that their final state in \mathcal{H}_T will be given by $|\psi'\rangle = U(\gamma)|\psi\rangle$ where $U(\gamma)$ is a unitary operator which may depend on the paths along which the quasiparticles were interchanged. But since small motions of the quasiparticles do not affect the state, $U(\gamma)$ can only depend on the *topological* properties of the paths. In two dimensions these topological properties are measured by the *braiding* of the paths. This means that the possible ways they can go around one another fall into discrete topological classes. For example, if two quasiparticles start out in a specific configuration, and end up in the same configuration, in between they could get there in an infinite number of topologically distinct ways: (0) neither particle moves at all; (1) particle 2 stays put, but particle 1 moves around it once

in a counterclockwise manner; (2) particle 2 stays put, but particle 1 moves around it twice in a counterclockwise manner; ...; (-1) particle 2 stays put, but particle 1 moves around it once in a clockwise manner; ...; see the figure.

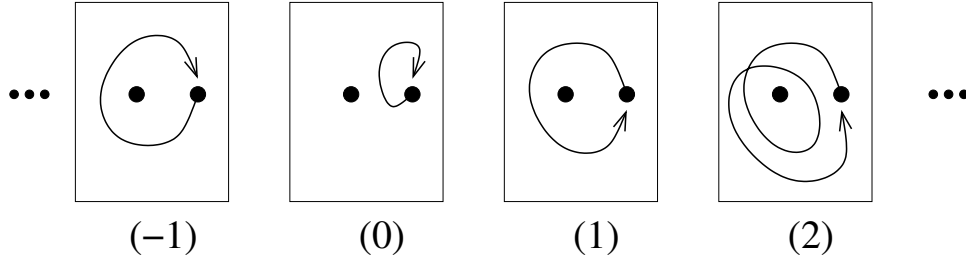


Figure 8: Different braidings of two quasiparticles in the plane.

For identical quasiparticles, just as with identical particles in three dimensions, their wave functions have definite statistics under interchange of the particles. For example, identical fermions, like electrons, in three dimensions, obey Fermi statistics, which means that under interchange of a pair of electrons, the state of the whole system gains a minus sign. This phase of the wave function under interchange is an example of a topologically-protected aspect of the quantum evolution of the system. Unfortunately, this simple boson/fermion phase can't be used to do any interesting quantum computations. However, the statistics of two-dimensional systems is potentially much richer, because now there is not just interchange of identical quasiparticles, but there is also the topological class of the path along which the interchange takes place, as illustrated with the braidings above. Quasiparticles with these more general *braid statistics* are called *anyons* (because they're not constrained to be only bosons or fermions anymore).

If one has a 2d topological material whose anyons have complicated enough braid statistics, one can imagine performing quantum computations as follows:

- First, create a configuration of well-separated anyons. This prepares a state $|\psi\rangle \in \mathcal{H}_T$.
- Next, move the anyons along some paths which avoid one another but braid them in some pattern, arriving back at the initial configuration. Their state will be changed to $|\psi'\rangle = U(\text{braid})|\psi\rangle$. Thus U is the analog of a quantum circuit which depends only on the braiding chosen.
- Finally, measure the state $|\psi'\rangle$.

These steps — preparation, controlled evolution, and measurement — are all the elements needed for a quantum computation. If the anyon braid statistics are sufficiently rich, then one can design arbitrary quantum circuits by choosing sufficiently complicated braidings of a sufficient number of anyons. Furthermore, all the steps in this quantum computation are automatically protected from noise by the topologically protected nature of the \mathcal{H}_T code subspace.

In what follows, we will try to flesh out this idea, first using the toric code as a model of a topological material, and then later generalizing to topological materials with more complicated anyonic braid statistics.

12 Toric code as a model of a topological material

We can make this more concrete by thinking of the large set of encoding qubits used in the toric code as the microscopic states of a macroscopic material. For instance, all the $N = 2n^2$ qubits of the toric code might be electron spin states on the surface of a metal in the shape of a donut. Even if it is macroscopic, so $N \sim 10^{20}$, it would still only encode 2 code qubits.

Now suppose the interaction energy among these N encoding qubits is such that noise is automatically suppressed. In other words, it is “designed” so that you pay a large energy cost to add 1-qubit noise to the system. For example, here is a Hamiltonian (energy function) for the toric code encoding qubits which does the job:

$$H = -E_0 \left(\sum_a V_a + \sum_{\hat{a}} P_{\hat{a}} \right). \quad (264)$$

E_0 is some positive constant with the dimensions of energy. It is called the *energy gap* of the system, for reasons that will become clear. Since the eigenvalues of V_a and $P_{\hat{a}}$ are all ± 1 and they all commute, the possible eigen-energies of H are

$$E = -E_0 \left(\sum_a (\pm 1) + \sum_{\hat{a}} (\pm 1) \right). \quad (265)$$

Clearly the lowest-energy states are those with $V_a = P_{\hat{a}} = +1$ for all vertices a and plaquettes \hat{a} . They have energy $E = -2n^2 E_0$. Note that these ground states are the toric code subspace, so they are precisely the 4-dimensional subspace with basis $\{|00\rangle_L, |01\rangle_L, |10\rangle_L, |11\rangle_L\}$.

Flipping a qubit to make some V_a or $P_{\hat{a}} = -1$ costs an energy $2E_0$, so if the energy gap E_0 can be made large enough, noise will be automatically suppressed. Physically, the probability of a transition costing an energy E_0 is typically suppressed by an exponential factor $e^{-\beta E_0}$, where the energy scale

$$k_B T \doteq \frac{1}{\beta} \quad (266)$$

is called the *noise temperature*. (Here k_B is the Boltzmann constant which is the conversion factor between temperature and energy.) It is often the actual temperature of the system. So for a given energy gap E_0 , noise in such a system can be exponentially suppressed by making the temperature small enough.

Note that the Hamiltonian is *local* on the lattice of encoding qubits, because $P_{\hat{a}}$ and V_a are products of operators acting on nearest neighbor qubits. This is typical of actual materials, and gives some hope that such a system can be realized.

12.1 Insensitivity to the detailed microscopic interactions

The crucial property of this system is that even though it has a macroscopic number of degrees of freedom, it nevertheless has a four-fold degenerate ground state, and all other states have energies “gapped” from those ground states — i.e., they have energies greater than the ground state by at least $2E_0$. This may seem like an incredibly fine-tuned property:

real materials will have all sorts of defects (dirt) which would be expected to completely destroy this ground state degeneracy. But, surprisingly, this is not the case: the ground state degeneracy and the energy gap are robust features of this system. In particular, if you perturb the Hamiltonian by *any* local operator,

$$H \rightarrow H + \delta H, \quad \text{where} \quad \delta H = \sum_a \epsilon_a \mathcal{O}_a, \quad \text{with} \quad \frac{1}{N} \sum_a |\epsilon_a| < E_0, \quad (267)$$

then the ground states $\{|n_x, n_y\rangle_L\}$ remain degenerate to exponential accuracy, and there is still an energy gap $\sim E_0$ protecting against local noise. In (267) the local “dirt” operators, \mathcal{O}_a , at each site are assumed to have matrix elements with the unperturbed ground states of order 1, and their ϵ_a coefficients are assumed to be small compared to the energy gap E_0 on average, as shown. This amounts to saying that although the dirt can be random and pervasive, it is assumed not to be “too big”.

The reason for this stability of the ground state degeneracy and gap follows from our previous analysis of the toric code, where we found that we needed an accumulation of at least $\sim \sqrt{N/2}$ local operator actions (“errors”) to change the code state. Thus δH needs to act $\sim \sqrt{N/2}$ times to modify the ground state energy, implying it lifts the degeneracy of the code subspace by an amount

$$\Delta E \sim \left(\frac{\epsilon}{E_0} \right)^{\sqrt{N/2}} E_0. \quad (268)$$

(This argument and estimate come from perturbation theory.) Even if $\epsilon/E_0 = 1/2$ on average, this means the energy splitting is exponentially suppressed by the size of the system, $\Delta E \lesssim 2^{-\sqrt{N/2}} E_0$. Note that $\sqrt{N/2}$ is proportional to the linear size of the system L divided by the atomic spacing a , so for macroscopic systems, we can easily have $\sqrt{N/2} \approx 10^{10}$. This means that ΔE is *extremely* small, and the ground state remains effectively degenerate.

(Usually in physics when we have such degeneracies there is a symmetry responsible for enforcing it. In this case classical 20th century symmetries do not enforce the ground state degeneracy. Instead there is a more intrinsically quantum mechanical effect at play that is increasingly viewed as a kind of quantum generalization of the very notion of “symmetry” in physics.)

This means that we don’t need to exquisitely engineer our topological computer material: it just has to be “close enough”. This gives hope that such materials might actually exist in nature.

In fact they do! Since around 1990 it was realized that certain states of matter have this kind of property. They are known as *topological phases* of matter, or as materials with *topological order*. The best-known examples are simply the surface electron states of a metal in a strong magnetic field at very low temperatures. An indication of the topological order is the observed *quantum Hall effect* behavior of these materials. By now many different examples of topological phases are known.

12.2 Quasiparticles of the toric code

So, there are materials that “automatically” protect the code qubits from noise. But how do we do computations with them? In other words, what are the analogs of quantum gates

and circuits for such materials?

In these topological phases, the gap — the finite, positive energy price, E_0 , that one pays to excite the system above its ground states — can be used as a tool to control the specific state of the code subspace — the 4-dimensional subspace of ground states in the toric code case.

Consider an energy eigenstate in which $V_{a_0} = -1$ for one vertex a_0 , and otherwise $V_a = P_a = +1$ for all the other vertices and all the plaquettes. This is a state localized at vertex a_0 with energy $2E_0$ above the ground state. Physicists often call such states of spatially localized energy *quasi-particles*. We will call this a “V-particle at a_0 ”. This is an eigenstate of our topological Hamiltonian (264), so it is stable: once created, it just stays at a_0 . On the other hand, the energy of the V-particle does not depend on its location a_0 . So, by some process (adiabatically adding and removing energy from the system), we can move the V-particle around at will. Note that we detect and interact with a V-particle at a_0 by measuring or acting with X or Z operators acting on qubits on edges ending at a_0 . But we have seen that repeated such “errors” or “noise” do not change the code state $|\psi\rangle_L \in \mathcal{H}_L$ irreversibly: the code state can be recovered as long as there is not a (macroscopic) string of errors stretching around one or both cycles of the torus.

We can make this concrete as follows. Consider adding localized energy to flip a single qubit at edge ℓ . This acts on the code state as $|\psi\rangle_L \rightarrow X_\ell |\psi\rangle_L$, and has the effect of creating two neighboring V-particles. The energy needed to create this pair (to flip the bit) is $4E_0$.

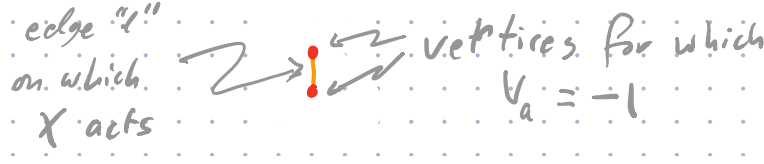
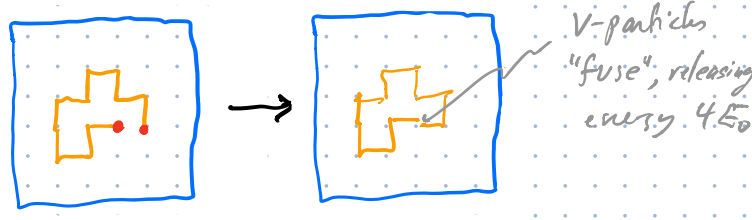


Figure 9: Creating a pair of V-particles by flipping a bit.

(V-particles can only be created in pairs because of the constraint (242).)

We can move the V-particles around by acting on neighboring edges, ℓ' with $X_{\ell'}$ bit flips at no cost in energy. A series of successive such bit flips will move the V-particles along some path. If we bring them back together, we can “fuse” or cancel them, gaining back energy $4E_0$.



This leaves the original state acted on by a closed path of spin-flips

$$|\psi\rangle_L \rightarrow \prod_{\ell \in \text{loop}} X_\ell |\psi\rangle_L. \quad (269)$$

But as we showed earlier, this does not change the code state,

$$\prod_{\ell \in \text{loop}} X_\ell |\psi\rangle_L = \left[\text{Diagram: A 5x5 grid of dots with a blue square boundary and an orange closed path in the center} \right] = \left[\text{Diagram: A 5x5 grid of dots with a blue square boundary} \right] = |\psi\rangle_L, \quad (270)$$

as long as the closed path of spin flips does not wrap one or both homology cycles of the torus,

$$|n_x, n_y\rangle_L = \left[\text{Diagram: A 5x5 grid of dots with a blue square boundary} \right] \neq \left[\text{Diagram: A 5x5 grid of dots with a blue square boundary and an orange path that wraps around the right and bottom edges} \right] = |n_x \oplus 1, n_y\rangle_L. \quad (271)$$

Thus pair-creating V-particles, moving one around a non-contractible cycle of the torus, and then fusing them acts as a “gate” transforming the logical qubits.

Entirely analogously, we can implement a $|\psi\rangle_L \rightarrow Z_\ell |\psi\rangle_L$ phase flip. This creates two neighboring “P-particles” — localized states where $P_{\hat{a}_0} = -1$ for a face \hat{a}_0 . Again, moving one P-particle around a cycle of the torus and annihilating it gives a phase transformation of $|\psi\rangle_L$. Similarly, adding energy to the system by acting on a qubit with Y , $|\psi\rangle_L \rightarrow Y_\ell |\psi\rangle_L \propto X_\ell Z_\ell |\psi\rangle_L$, creates a pair of “PV-particles” — which can be thought of as a bound state of a P-particle and a V-particle.

For topological matter realizing the toric code, these are all the independent excitations at our disposal, so we can only make gates by pair-creating P-, V-, or PV-particles, moving them around, and then annihilating them. It turns out that the set of 2-qubit gates (i.e., acting on the code space) that one obtains in this way is not rich enough to permit general quantum computations. Essentially, the problem is two-fold: pair-creating any number of these quasiparticles does not change the dimension (i.e., 4) of the code subspace, \mathcal{H}_L ; and, relatedly, moving any of these particles along a path looping around another particle (i.e., braiding the particle paths) only gives simple phases on the code state. For instance, under interchange

$$E_j E_k |\psi\rangle_L \xrightarrow{\text{exchange}} e^{i\theta_{jk}} E_k E_j |\psi\rangle_L, \quad (272)$$

where $E_j \in \{P, V, PV\}$ are the different possible excitations. In the case of the toric code, the exchange phases are

$$e^{i\theta_{jk}} = \begin{pmatrix} +1 & +1 & +1 \\ +1 & +1 & +1 \\ +1 & +1 & -1 \end{pmatrix}, \quad (273)$$

where the rows/columns refer to P, V, and PV left to right and top to bottom. This means that P- and V-particles just act like bosons and PV-particles just act like fermions.

When quasiparticle braiding just gives simple phases as in (272), even if they are more general than just ± 1 as in the toric code case, we say the quasiparticles are *abelian anyons*. Topological matter with just abelian anyons is not rich enough to permit general quantum computations.

13 Generalization to non-abelian anyons

We now describe a more general kind of topological matter which has *nonabelian anyon* quasiparticles. These kinds of matter can, in principle, be used to build fault-tolerant quantum computers. This appealing idea has garnered much theoretical attention.

“In principle” because this has not been realized in practice yet: physical realizations of such systems are beyond current technological capabilities to create and manipulate. As of 2013, only a few potentially non-abelian anyon topological phases have experimentally been observed and none have been definitively confirmed to have non-abelian anyon excitations. For example, the $\nu = 12/5$ fractional quantum Hall system is thought to give a realization of such nonabelian statistics. Furthermore, its braid matrices (defined below) can be realized as rotations by multiples of $\pi/5$ (along with phases) around various directions in the Hilbert space of multiple quasiparticles. These braid matrices do not form a finite group, but are dense in the set of all unitary transformations, so can in principle be used to approximate any desired unitary transformation to any given accuracy, so can in principle form a basis for building an automatically fault-tolerant quantum computer. But, experimentally, the $\nu = 12/5$ fractional quantum Hall system is very hard to reliably observe, even at very low temperatures (10^{-2} K), let alone manipulate.

There are other types of systems (besides fractional quantum Hall systems) which support nonabelian anyon quasiparticles and could be used for topological quantum computers. Prominent among them are the so-called $p + ip$ superconductors and topological insulators. Also, evidence of the quantum Hall effect (though not yet of the fractional quantum Hall effect) has recently (2007) been observed at room temperature in graphene. It is an active area of research to locate suitable systems with nonabelian braid statistics and braid matrices which are dense in the set of unitary matrices. Only time will tell whether this interesting idea will give rise to practical quantum computers.

On the theoretical side, one can construct models of systems with nonabelian anyons along the lines of the way we interpreted the toric code as a model of topological matter. But they quickly become mathematically complicated, so we will not describe them here, and will instead just concentrate on an “axiomatic” description of the properties of their anyonic quasiparticles. (The interested student can look at Kitaev’s original article referenced at the beginning of this part for a description of a relatively simple set of such models.)

Quasiparticles come in various “types”, which we will label by a, b, c , etc. These are the analog of the V-, P-, and PV-particles of the toric model. A configuration of n well-separated anyons of types $\{\text{types}\} = \{a_1, a_2, \dots, a_n\}$ in our topological material is associated with a topologically protected subspace of the Hilbert space, which we’ll call $V_{\{\text{types}\}}$. This space does not depend on the positions of the anyons, so only depends on the list of types of

anyons present. One performs local measurements on the system to determine the $\{\text{types}\}$ of anyons present. Measurement of these anyon-type observables projects the state vector onto the $V_{\{\text{types}\}}$ subspace but does not affect the state within that subspace, as per the usual measurement rules of quantum mechanics. Indeed, due to the topological nature of the material, states in $V_{\{\text{types}\}}$ are protected from all local measurements or any other local interactions with the environment. In the case of the toric code, $V_{\{\text{types}\}} \simeq \mathcal{H}_L$, the 4-dimensional code subspace, irrespective of the number and types of the quasiparticles present. This is generally not the case for systems with nonabelian anyons, where typically the dimension of $V_{\{\text{types}\}}$ grows with the number of anyons present. We will see an explicit example of this with the so-called “Fibonacci anyon” below.

In particular, for these systems there is no need to place the system on a torus (or other topologically nontrivial surface) in order to perform quantum computations. For instance, if the system is realized on a simple disk, its ground state with no anyons will be non-degenerate (i.e., a one-dimensional subspace, described by a single basis state). But upon pair-creating a set of anyons, the protected subspace $V_{\{\text{types}\}}$ can be multi-dimensional.

There are three basic processes that anyons can undergo:

- fission (pair creation),
- braiding (moving them around in the plane),
- fusion (pair annihilation).

Fission and fusion are inverses of each other. These three processes also correspond to the basic steps of a quantum computation. Successive fissions create from the ground state (by appropriately adding energy to the system in a localized way) a configuration of n anyons with $\{\text{types}\}$ in a specific state $|\psi\rangle \in V_{\{\text{types}\}}$. This corresponds to preparing the initial state of the qubits in a quantum circuit. Braiding the $\{\text{types}\}$ anyons by dragging them along some specified paths (again, done by appropriately adding energy to the system in a localized way) changes the state to a new state $|\psi'\rangle = U(\text{braid})|\psi\rangle \in V_{\{\text{types}\}}$, where $U(\text{braid})$ is a unitary operator determined by the $\{\text{types}\}$ and the braiding (the topological equivalence class of the anyons paths). This corresponds to acting on the initial state of the qubits by a sequence of quantum gates realizing the unitary transformation $U(\text{braid})$. Successively fusing the the anyons, by bringing them together and removing the released energy, reduces the number of anyons and can change their types. Measuring the types of the anyons remaining after fusion corresponds to measuring the final state of the quantum circuit qubits in the computational basis.

So what remains is to describe the rules for the effect fusion (or fission) and braiding of anyons has on the $V_{\{\text{types}\}}$ Hilbert spaces.

We assume a finite set of anyon types, $a, b, c, \dots \in \{1, t_1, t_2, \dots, t_n\}$. It is convenient to include in this list a type for the “identity anyon”, 1, which corresponds to the absence of an anyon. Creation of a set of anyons from the ground state corresponds to a process $1 \rightarrow \{a, b, c, \dots\}$, and the inverse process, $\{a, b, c, \dots\} \rightarrow 1$, to their annihilation. More generally, by bringing a set of anyons together and removing the released energy, we may have a process like $\{a, b, c, \dots\} \rightarrow d$ in which we say that anyons of types $\{a, b, c, \dots\}$ *fuse* to an anyon of type d . The inverse process in which we add energy to anyon d to produce a set

of separated anyons of types $\{a, b, c, \dots\}$ is called fission. Since fission is fusion in reverse, we will refer to them both as fusion from now on.

Just as to a set $\{a, b, c, \dots\}$ of anyons created from the ground state we have a Hilbert space $V_{abc\dots}$ — the $V_{\{\text{types}\}}$ introduced above — we will have associated to a general fusion process $\{a, b, c, \dots\} \leftrightarrow d$ a Hilbert space which we'll call $V_{abc\dots}^d$. Thus $V_{abc\dots} \equiv V_{abc\dots}^1$ in this notation, since a single anyon of type 1 describes the system with no anyons, i.e., the ground state. It is convenient to introduce a graphical notation for the fusion of lines as in figure 10. One reads these figures as depicting the paths in space-time that anyons take as they are



Figure 10: Diagrammatic depiction of vector spaces associated with fusing anyons.

fused. For instance, thinking of time as running from the top of the diagram to the bottom, the left diagram shows anyons a and b approaching one another and fusing to form anyon d , and so forth. These graphs represent the Hilbert spaces V_{ab}^d , V_{abc}^d , etc. The fusion of two anyons into one will be our central focus, and we'll call the V_{ab}^c Hilbert spaces *fusion spaces*.

A basic structure is then encoded in the *fusion algebra* of pairs of anyons,

$$a \times b = \sum_c N_{ab}^c c, \quad N_{ab}^c \doteq \dim(V_{ab}^c). \quad (274)$$

Here “ \times ” is a formal multiplication of anyon types defined by this equation. Since they are dimensions of vector spaces, the N_{ab}^c are non-negative integers. If $N_{ab}^c = 0$, this just means that fusing a with b cannot give c . Since two anyons must fuse into *some* anyon type (since we have included the identity anyon type “1” which corresponds to no anyon), there must be some c for which $N_{ab}^c > 0$ for every choice of a and b . Since the identity anyon is the same as no anyon, we must also have that

$$1 \times a = a \times 1 = a, \quad (275)$$

for any a , or, equivalently, that $N_{1a}^c = N_{a1}^c = \delta_a^c$. We can interpret $N_{ab}^c > 1$ as counting the multiplicity of distinct “ways” that a and b can fuse to c . What may be less obvious is what is meant by the summation over anyon types c on the right side of (274). This just reflects the fact that if the anyon type c which is the outcome of the fusion process is not measured then all we can say is that the state of the system will be in the direct sum, $\oplus_c V_{ab}^c$, of fusion spaces.

Likewise, we can also specify the vector spaces V_{abc}^d for fusing three lines, and so forth as shown in figure 10, and define $N_{abc}^d = \dim(V_{abc}^d)$, etc. But these higher fusion spaces are determined by the two-anyon fusion spaces, for we can think of fusing three anyons as a process in which first two of the anyons fuse, and that fused anyon then fuses with the remaining anyon. This is illustrated in figure 11(a), where the right diagram shows a and b fusing to f , and then f and c fusing to d . Since the identity of the intermediate anyon f

is not measured, we should sum over all possible f . Thus the figure depicts the equivalence among fusion spaces

$$V_{abc}^d \simeq \bigoplus_f \left(V_{ab}^f \otimes V_{fc}^d \right). \quad (276)$$

But the middle fusion diagram in figure 11(a) shows that the $abc \rightarrow d$ fusion can be obtained by first fusing $bc \rightarrow e$ and then $ae \rightarrow d$, giving the equivalence

$$V_{abc}^d \simeq \bigoplus_e \left(V_{ae}^d \otimes V_{bc}^e \right). \quad (277)$$

The equivalence of the right sides of (276) and (277) means there is an isomorphism¹¹ between them,

$$F_{abc}^d : \bigoplus_f \left(V_{ab}^f \otimes V_{fc}^d \right) \rightarrow \bigoplus_e \left(V_{ae}^d \otimes V_{bc}^e \right), \quad (278)$$

called the *associator* isomorphism between fusing 3 anyons in different orders.



Figure 11: (a) The associator isomorphism F . (b) The braid isomorphism R .

The associator satisfies a stringent consistency condition of the form $F^3 = F^2$, called the pentagon identity, and shown in figure 12(a). Deriving its precise form by putting in anyon labels is left as an exercise. There can be many inequivalent solutions to the pentagon identity for the associator F for a given fusion algebra (274).

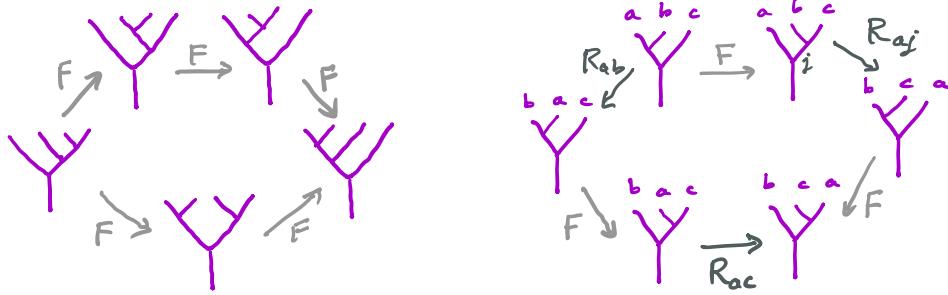


Figure 12: (a) The pentagon identity satisfied by the associator F . (b) The hexagon identity satisfied by the associator and the braid matrix R .

The fusion product (274) is commutative, since fusing anyon a with anyon b is the same as fusing b with a , as we can simply exchange their positions before fusing them while still

¹¹ “Isomorphism” just means an invertible linear map between the two vector spaces. Since they are Hilbert spaces, the isomorphism must, in addition, preserve inner products.

keeping them far apart. Thus $V_{ab}^c \simeq V_{ba}^c$ are isomorphic. This is shown in figure 11(b). But, as shown in figure 8, there are topologically inequivalent ways of exchanging their positions — the different braidings of their paths. This means that to specify the isomorphism in figure 11(b) we need to specify a specific topological class of paths exchanging a and b . For instance, we can choose this to be a simple counterclockwise exchange of a with b in the x - y plane when looking down on them from the positive z -axis. We call the resulting isomorphism,

$$R_{ab}^c : V_{ab}^c \rightarrow V_{ba}^c, \quad (279)$$

the *braid* isomorphism. Notice that R_{ba}^c need not be the inverse of R_{ab}^c , since successive counterclockwise interchanges results in a non-trivial braiding. This reflects the possibility of fractional statistics of anyons in 2 dimensions.

The braid and associator isomorphisms satisfy a “hexagon identity” of the form $RFR = FRF$, shown in figure 12(b). A fusion algebra with identity and this braid structure is called a *modular tensor category* in the mathematics literature. It can be shown that for each anyon type a there is a unique “anti-type” \bar{a} such that $a \times \bar{a} = 1 + \dots$ ($a = \bar{a}$ is allowed.) Also it can be shown that for a given fusion rule, N_{ab}^c , there is only a finite set of inequivalent solutions for F and R satisfying the pentagon and hexagon identities. Furthermore, every such solution of topological anyons can be realized by some topological quantum mechanical model. (For instance, it has been shown that every such solution is realized by the anyons of some (2+1)-dimensional *Chern-Simons gauge theory*, a class of well-studied quantum field theories.)

When $N_{ab}^c = 1$, the R_{ab}^c braid isomorphism is an isomorphism between two 1-dimensional Hilbert spaces. Thus R_{ab}^c can only be a complex phase. If anyons a and b can only fuse to anyon c , then this phase can be thought of as an intrinsic property of a and b — called their exchange statistics. But if a and b can fuse to more than one type of anyon, say c and d , then R_{ab}^c and R_{ab}^d can be different phases. In this case we say that these anyons have *non-abelian statistics*. On the other hand, when $N_{ab}^c > 1$, the R_{ab}^c braid isomorphism is now an $N_{ab}^c \times N_{ab}^c$ unitary matrix. In this case we also say that these anyons have non-abelian statistics. So, to summarize, whenever $\sum_c N_{ab}^c > 1$ we can have non-abelian anyons.

I’ll finish by briefly describing two of the simplest examples of modular tensor categories — i.e., anyon fusion and braiding algebras — one with abelian statistics and one with non-abelian statistics.

$\mathbb{Z}_2 \times \mathbb{Z}_2$ fusion rule. Consider the case of four types of anyon, $\{1, a, b, c\}$, satisfying the fusion algebra

$$\begin{array}{c|ccc} \times & a & b & c \\ \hline a & 1 & c & b \\ b & c & 1 & a \\ c & b & a & 1 \end{array}, \quad (280)$$

where the row and column corresponding to multiplication with the identity anyon is left out, since it is always given by (275). The fusion algebra (280) is $N_{ab}^c = \delta_{a \cdot b}^c$ where $a \cdot b$ is multiplication in the group $\mathbb{Z}_2 \times \mathbb{Z}_2$. Since only a single anyon type appears as the result

of the fusion product and all the N_{ab}^c are either 0 or 1, it follows all the non-trivial fusion spaces are 1-dimensional, i.e., equivalent to \mathbb{C} . So this is an example of anyons with abelian statistics.

There are four inequivalent solutions of the pentagon and hexagon identities for R and F . It is a good exercise to write out these identities and construct all the solutions; in doing this you should bear in mind that two solutions that look different may in fact be equivalent after an appropriate change of basis of the V_{ab}^c spaces (which can only be by phases in this case since they are all 1-dimensional). Since the anyons all have abelian statistics, all the R are just phases. One solution is the obvious solution in which all $R = 1$ (and $F = 1$ as well); this is called the $\mathbb{Z}_2 \times \mathbb{Z}_2$ group solution. The other three solutions have braiding phases

$$\begin{array}{c|ccc} R & a & b & c \\ \hline a & 1 & 1 & 1 \\ b & 1 & 1 & 1 \\ c & 1 & 1 & -1 \end{array} \quad \begin{array}{c|ccc} R & a & b & c \\ \hline a & -1 & -1 & -1 \\ b & -1 & -1 & -1 \\ c & -1 & -1 & -1 \end{array} \quad \begin{array}{c|ccc} R & a & b & c \\ \hline a & i & 1 & -1 \\ b & 1 & -i & -1 \\ c & -1 & -1 & 1 \end{array} \quad (281)$$

(toric code) (three fermion) (double semion)

The first, where a and b are bosons and c is a fermion, is the toric code (with a and b denoting the V-particles and P-particles, and c denoting the PV-particles). The three fermion model is the one where all three anyons have fermion statistics. The double semion model is so-named because the a and b anyons are semions, i.e, have statistical phase which is the square-root of fermionic statistics.

Fibonacci anyon. This is the simplest example of non-abelian anyon statistics. It has just two types, 1 and a (which are their own anti-anyons) satisfying

$$a \times a = 1 + a. \quad (282)$$

There is a unique solution to the pentagon and hexagon identities (up to isomorphism) given by

$$\begin{aligned} R_{aa}^1 &= e^{4\pi i/5}, & R_{aa}^a &= -e^{2\pi i/5}, \\ F_{1aa}^a &= \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, & F_{aaa}^a &= \frac{1}{\varphi} \begin{pmatrix} 1 & \sqrt{\varphi} \\ \sqrt{\varphi} & -1 \end{pmatrix}, \end{aligned} \quad (283)$$

where $\varphi = (1 + \sqrt{5})/2$ is the golden ratio. (The other braid and associator isomorphisms are the identity.) a thus has something like fifth-root of unity anyonic statistics but with different phases depending on the fusion channel. This is an example of non-abelian anyonic statistics. Denote by $N_{a^n}^1 = \dim(V_{a^n \dots a}^1)$ the dimension of the vector space for fusing n anyons to the ground state. By (282) it follows that $N_{a^1}^1 = N_{a^2}^1 = 1$ and it is easy to see that $N_{a^n}^1 = N_{a^{n-1}}^1 + N_{a^{n-2}}^1$. Thus the $N_{a^n}^1$ are the *Fibonacci numbers*, and $\lim_{n \rightarrow \infty} N_{a^n}^1 \sim \varphi^n / \sqrt{5}$. Interpreting $N_{a^n}^1$ as the dimension of the Hilbert space of n Fibonacci anyons created from the ground state, this motivates identifying the *quantum dimension* of a Fibonacci anyon as the irrational number $\varphi \approx 1.62$. Fibonacci anyons can be realized as (a subsector of) the $SU(2)_3$ Chern-Simons theory, or the related Yang-Lee edge singularity (a non-unitary 2d rational conformal field theory).