

Quantum Computing

Prof. Dr. Michael J. Hartmann

Summer Semester 2024

Contents

1	Overview and Introduction	3
1.1	Organizational things	3
1.2	Motivation	3
1.2.1	History of Quantum Computing	3
1.2.2	Course Content	4
2	Quantum Mechanics	5
2.1	States, Wavefunctions and Operators	5
2.1.1	States and wavefunctions	5
2.1.2	Time evolution and superpositions	6
2.1.3	Operators	7
2.1.4	Measurements	8
2.1.5	Qubits	9
2.2	Composite quantum systems	10
2.2.1	States	10
2.2.2	The exponential Hilbert space	12
2.2.3	Operators	13
2.2.4	Density matrices	14
2.2.5	Dissipation	15
3	Quantum Computing Basics	18
3.1	Models of Quantum Computation	19
3.1.1	Gate model of Quantum Computing	20
3.1.2	Adiabatic Quantum Computing	20
4	Quantum Algorithms for Fault Tolerant Quantum Computers	22
4.1	Deutsch's Problem	22
4.1.1	Deutsch's Problem: formal version	23
4.2	Classical Circuits	27
4.3	Quantum Circuits	27
4.3.1	Gates	28
4.3.2	Some applications of the C-NOT gate	30
4.4	Universal sets of gates	33
4.5	Can a quantum computer do classical computations efficiently?	34
4.6	How hard is it to simulate a quantum computer with a classical computer?	37
4.7	Simon's problem	38
4.8	Quantum Fourier transform	41
4.9	Application: Phase Estimation	45
4.10	Shor's Factoring Algorithm	48
4.10.1	Order or Period Finding	50
4.11	Application: Solving systems of linear equations	51

4.11.1	Applying arbitrary powers of gates	51
4.11.2	Solving systems of linear equations: HHL algorithm (Harrow, Hassidim, Lloyd)	53
5	Quantum Error Correction	56
5.1	Repetition Code	56
5.1.1	Classical repetition code	56
5.1.2	Quantum 3 qubit repetition code	57
5.1.3	General types of errors	59
5.2	Stabilizer Formalism	60
5.3	Surface Codes	61
5.3.1	Toric Code (Kitaev 2006)	61
5.3.2	Planar surface codes	65
6	NISQ Quantum Computing	68
6.1	Today's quantum hardware	68
6.2	Variational algorithms	70
6.3	Quantum Approximate Optimization Algorithm (QAOA)	74
6.4	Quantum Simulation: Computing Quantum Dynamics	77

Chapter 1

Overview and Introduction

1.1 Organizational things

1.2 Motivation

Why should one learn about Quantum Computing?

- It has revolutionary potential.
 - quantum algorithms may eventually make most encryption insecure
 - quantum computers may allow to simulate artificial materials and new drugs → revolution of materials and pharma science
- It's coming.
 - significant progress is being made
 - investment in quantum companies is strongly growing

1.2.1 History of Quantum Computing

1980 First idea by Richard Feynman and Yuri Manin. Feynman's famous quote: *Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy.*

1992 David Deutsch and Richard Jozsa propose first quantum algorithm that can solve a specific problem faster than classical computers.

1994 Peter Shore's algorithm for prime factorization in polynomial time. → Would break *RSA* encryption used in banking, secret services etc.

1995 Ignacio Cirac and Peter Zoller propose trapped ion architecture [Phys. Rev Lett. **74**, 4091].

1999 Demonstration of first superconducting qubit by Nakamura, Pashkin and Tsai, [Nature **398**, 786].

2001 Demonstration of Shor's algorithm to factor 15 using a 7-qubit NMR computer [Nature **414**, 883].

2011 dWave Systems announces first commercial quantum annealer.

- 2016 IBM starts providing cloud access to its quantum hardware.
- 2019 IBM introduces IBM Q System One, its first integrated quantum computing system for commercial use.
- 2019 Google demonstrates computation that is beyond the reach of classical computing technology.
- 2021 IonQ first quantum computing start-up to be listed in Nasdaq
- 2021 IonQ and AQT demonstrate error correcting codes
- 2022 ETH Züürich and Google demonstrate quantum error correction in $d = 3$ (ETH) and $d = 5$ (Google) surface code
- 2024 QuEra, MIT and Harvard demonstration 48 error corrected qubits and perform gates with them

1.2.2 Course Content

- Quantum Mechanics (*states, superpositions, time evolution, operators, measurements, composite systems, dissipation*)
- Quantum computing basics (*models of quantum computation*)
- Quantum Computing for fault tolerant quantum computers (*Deutsch's problem, quantum circuits, quantum Fourier transform, quantum phase estimation, Shor's algorithm, HHL algorithm*)
- Quantum error correction (*repetition code, stabilizer formalism, surface code*)
- NISQ quantum computing (*today's hardware, variational algorithms, quantum simulation*)
- programming quantum computers (*Qiskit*)

Chapter 2

Quantum Mechanics

Here we recap the necessary concepts of quantum mechanics.

2.1 States, Wavefunctions and Operators

2.1.1 States and wavefunctions

In quantum theory a physical system is fully described by a quantum state $|\psi\rangle$.

The space spanned by these states is called Hilbert space (it's big)

The state can be written in a specific basis, e.g. the position basis

$$\langle x|\psi\rangle = \psi(x) \quad (2.1)$$

which is the quantum mechanical wave-function.

The probability for finding a particle at a specific position x is the given by

$$P(x) = |\psi(x)|^2 \quad (2.2)$$

Writing the state in a specific basis is analog to writing a vector in our three dimensional space in a specific basis, e.g.

$$\vec{r} = x\vec{e}_x + y\vec{e}_y + z\vec{e}_z = r\vec{e}_r + r\theta\vec{e}_\theta + r\sin\theta\varphi\vec{e}_\varphi \quad (2.3)$$

The vector can be specified by the three cartesian coordinates x, y and z or by the three spherical coordinates r, θ and φ .

In quantum theory this is very much the same, the state can e.g. be specified by its amplitudes in a position representation $\langle x|\psi\rangle = \psi(x)$ or a momentum representation $\langle p|\psi\rangle = \psi(p)$ (which are related by a Fourier transform).

The notation $|\psi\rangle$ thus always refers to the representation independent, i.e. basis independent, quantity.

2.1.2 Time evolution and superpositions

The dynamics of the wave-function is described by the Schrödinger equation

$$\frac{\partial}{\partial t}\psi(x, t) = -\frac{i}{\hbar}\mathcal{H}(x, p)\psi(x, t) \quad (2.4)$$

where

$$\mathcal{H}(x, p) = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + V(x) \quad (2.5)$$

is the Hamiltonian (here for a single particle in one dimension) in position representation.

the Schrödinger equation (2.4) is linear.

$\Rightarrow$ the sum of 2 solutions is again a solution

$$\begin{aligned} \frac{\partial}{\partial t}\psi_1(x, t) &= -\frac{i}{\hbar}\mathcal{H}(x, p)\psi_1(x, t) \\ \frac{\partial}{\partial t}\psi_2(x, t) &= -\frac{i}{\hbar}\mathcal{H}(x, p)\psi_2(x, t) \\ \Rightarrow \frac{\partial}{\partial t}[\psi_1(x, t) + \psi_2(x, t)] &= -\frac{i}{\hbar}\mathcal{H}(x, p)[\psi_1(x, t) + \psi_2(x, t)] \end{aligned} \quad (2.6)$$

$\Rightarrow$ the sum or superposition of two quantum states is again a valid quantum state

Such superpositions are also known from the behavior of waves, in interference phenomena, e.g. the double slit experiment.

One counter intuitive consequence of such superpositions is that objects can e.g. be in two locations at the same time. This is explained in figure 2.1.

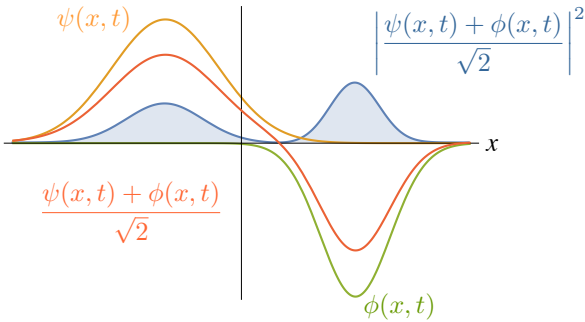


Figure 2.1: The normalized sum of a wave-function $\psi(x, t)$ and a wave-fuction $\phi(x, t)$ is $[(\psi(x, t) + \phi(x, t))/\sqrt{2}]$. It's absolute value squared describes the probability to find the particle at a position x . As in the sketch above this probability can become zero between two non-zero regions. There are thus two disjoint areas where the particle is found with finite probability.

Since the probability distribution for finding the particle at different locations depends on the entire wave-function, see Eq. (2.4), and thus on both areas of non-vanishing probability, this needs to be interpreted as

The particle is in both locations at the same time.

⇒ In a football match, the ball can be in both goals at the same time. Or you could be listening to me here and be on a beach in the Caribbean.

⇒ Why does this never happen?

→ Not fully understood, but one reason is the inevitable interaction with a surrounding.

Nothing is perfectly isolated. There is at least the electromagnetic vacuum around. We will revisit this fact in section 2.2.5

Implications for quantum computing:

- the computational power of quantum computers is rooted in superpositions.
- the inevitable coupling to an environment is what makes building quantum computers so difficult

We can also write this in a representation independent version, where the Schrödinger equation reads,

$$\frac{\partial}{\partial t} |\psi, t\rangle = -\frac{i}{\hbar} H |\psi, t\rangle \quad (2.7)$$

Here we have set $\hbar = 1$, a convention that we will stick to from now on. This simply means that the Hamiltonian H now has dimensions of an angular frequency and no longer an energy.

Clearly a solution equation (2.7) is

$$|\psi, t\rangle = e^{-iH(t-t_0)/\hbar} |\psi, t_0\rangle \quad (2.8)$$

This prompts us to introduce the time evolution operator

$$U(t, t_0) = e^{-iH(t-t_0)/\hbar} \quad (2.9)$$

which is unitary, i.e.

$$U(t, t_0)U^\dagger(t, t_0) = U^\dagger(t, t_0)U(t, t_0) = \mathbb{1} \quad (2.10)$$

Time evolution in quantum mechanics is thus a unitary transformation.

In a quantum computer, the execution of a computation will thus also be a unitary transformation.

Quantum dynamics is reversible: there is a unitary transformation which undoes the time evolution of a system.

⇒ Quantum manipulations are reversible

2.1.3 Operators

If we measure a property of a quantum system, we obtain eigenvalues of the observable corresponding to that property. For example

- measuring the position X , we get the value x with probability $|\langle x|\psi\rangle|^2$
- measuring the orientation of a spin-1/2 degree of freedom, we get $\uparrow$ with probability $|\langle \uparrow|\psi\rangle|^2$ and $\downarrow$ with probability $|\langle \downarrow|\psi\rangle|^2$

Quantum theory uses self-adjoint operators to represent observable quantities.

An operator A is a linear map

$$A|\psi\rangle = |\phi\rangle \quad (2.11)$$

$$A(a_1|\psi_1\rangle + a_2|\psi_2\rangle) = a_1|\phi_1\rangle + a_2|\phi_2\rangle \quad (2.12)$$

It is self-adjoint if

$$A^\dagger = A \quad (2.13)$$

which means its matrix representation is Hermitian

$$A_{j,l}^* = \langle j|A|l\rangle^* = A_{l,j} = \langle l|A|j\rangle \quad (2.14)$$

The eigenvalues of a Hermitian matrix are real and its eigenvectors are mutually orthogonal.

$\Rightarrow$ The eigenstates of a self adjoint operator form a basis of the Hilbert space

$$A|j\rangle = a_j|j\rangle \quad (2.15)$$

$$\langle j|l\rangle = \delta_{j,l} \quad (2.16)$$

$$\sum_j |j\rangle \langle j| = \mathbb{1} \quad (2.17)$$

Here, the first equation means that the states $|j\rangle$ are eigenstates of A , the second equation states that these eigenstates are orthonormal (mutually orthogonal and normalized) and the third equation states that they are complete (their number equals the dimension of the Hilbert space).

Example The Hilbert space of a spin-1/2 degree of freedom has dimension 2. Observables are here the polarizations of the spin in x, y and z direction. They are represented by the Pauli operators σ^x , σ^y and σ^z . We can choose the eigenstates of the spin in z-direction $|\uparrow\rangle$ and $|\downarrow\rangle$ as a basis, where the Pauli operators have matrix representations

$$\sigma^x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \sigma^y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad \sigma^z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad (2.18)$$

Exercise Find the normalized eigenvectors of the σ^x Pauli matrix and show that they are orthogonal and complete.

2.1.4 Measurements

The process of a measurement is very non-trivial in quantum mechanics and is not yet fully understood.

$\rightarrow$ there are big controversies about the interpretation of quantum mechanics

$\rightarrow$ there is however a description of measurements that one can work with ($\rightarrow$ "shut up and calculate" interpretation)

Consider an observable $\mathcal{O}$ that has discrete eigenvalues

$$\mathcal{O}|\nu\rangle = o_\nu|\nu\rangle \quad \text{for } \nu = 1, 2, \dots, n \quad (2.19)$$

Note that $\mathcal{O}$ needs to be Hermitian so that the eigenvalues o_ν are real.

$\Rightarrow$ the eigenstates $|\nu\rangle$ form a basis of the Hilbert space

There could be several eigenstates with the same eigenvalue (degenerate eigenvalues). $\rightarrow$ we assume labels ν such that the $|\nu\rangle$ form a basis.

$\Rightarrow$ Can expand the state before the measurement $|\psi\rangle$ in basis formed by the $|\nu\rangle$.

$$|\psi\rangle = \sum_{\nu=1}^n c_\nu |\nu\rangle \quad (2.20)$$

Measurement If we measure $\mathcal{O}$, we obtain any of its eigenvalues o_ν with probability $|c_\nu|^2$.

The state directly after the measurement is $|\nu\rangle$.

$\rightarrow$ If we repeat the measurement directly (instantly) after the first measurement, we will again find the same eigenvalue o_ν .

$\Rightarrow$ The measurement projects the state $|\psi\rangle$ onto the state $|\nu\rangle$,

$$|\nu\rangle = \frac{|\nu\rangle \langle \nu|}{\sqrt{|c_\nu|^2}} |\psi\rangle \quad (2.21)$$

Note the division by $\sqrt{|c_\nu|^2}$ which ensures the correct norm for the state after the measurement.

Problem The measurement process changes the state but is not described by the Schrödinger equation, c.f. (2.7), unlike other evolutions of states.

2.1.5 Qubits

Computers represent information in elementary units called bits, which can take values 0 and 1.

For example, a computer stores the number 5 as a binary number 101, which requires 3 bits that take the values 1, 0 and 1.

For quantum computers this concept has been generalized to the quantum bit or qubit, a quantum system that has 2 orthonormal states, just like a spin-1/2.

In analogy to the classical bit, the states of the qubit are denoted as

$$|0\rangle \quad \text{and} \quad |1\rangle \quad (2.22)$$

Since the Schrödinger equation is linear, see Eq. (2.4), and superpositions of states are also valid quantum states, the most general state of a qubit is

$$a|0\rangle + b|1\rangle \quad (2.23)$$

where a and b are complex numbers that fulfill

$$|a|^2 + |b|^2 = 1 \quad (2.24)$$

2.2 Composite quantum systems

The interesting and dramatic consequences of superpositions for quantum computing only come into play when considering multiple qubits.

To be able to describe multiple qubits, we first introduce the framework for describing composite quantum systems.

To keep things simple and avoid complicated writing, we restrict ourselves to bipartite quantum systems

2.2.1 States

The Hilbert space of this bipartite system is the direct product of the Hilbert space of subsystem A , $\mathcal{H}_A$ and subsystem B , $\mathcal{H}_B$,

$$\mathcal{H} = \mathcal{H}_A \otimes \mathcal{H}_B \quad (2.25)$$

For isolated subsystems A and B , we can expand their states in some basis of orthonormal states $|j_A\rangle$ and $|j_B\rangle$,

$$|\varphi_A\rangle = \sum_{j=1}^{d_A} a_j |j_A\rangle \quad \text{and} \quad |\varphi_B\rangle = \sum_{j=1}^{d_B} b_j |j_B\rangle \quad (2.26)$$

where $d_A(d_B)$ is the dimension of $\mathcal{H}_A(\mathcal{H}_B)$ and $a_j(b_j)$ are complex expansion coefficients.

A state in the joint Hilbert space could thus be

$$|\varphi\rangle = |\varphi_A\rangle |\varphi_B\rangle \equiv |\varphi_A, \varphi_B\rangle \quad (2.27)$$

We are writing here "could" since not all possible states are of this form as we shall see shortly.

In Eq. (2.27) we also introduced the shorthand notation

$$|j, l\rangle = |j\rangle |l\rangle = |j\rangle \otimes |l\rangle. \quad (2.28)$$

Example For two particles, their joint wave-function in position representation could read

$$\varphi(x_A, x_B) = \varphi_A(x_A)\varphi_B(x_B), \quad (2.29)$$

where x_A and x_B are the positions of the two particles. This wave-function as given in Eq. (2.29) is however special. More generally one can also imagine wave-functions that are not products of the wave-functions of the individual particles

$$\varphi(x_A, x_B) \neq \varphi_A(x_A)\varphi_B(x_B), \quad (2.30)$$

Such wave-functions do exist as well. We will discuss them in more detail later.

Example For two qubits, qubit A could be in the excited state $|1_A\rangle$ and qubit B could be in the ground state $|0_B\rangle$. Their joint state would read

$$|\varphi\rangle = |1_A, 0_B\rangle \quad (2.31)$$

or in the basis of states $|0\rangle$ and $|1\rangle$,

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad (2.32)$$

Since superpositions of states are also possible quantum states, the most general state of the composite Hilbert space $\mathcal{H}$ reads,

$$|\varphi\rangle = \sum_{j=1}^{d_A} \sum_{l=1}^{d_B} c_{j,l} |j_A, l_B\rangle \quad (2.33)$$

Importantly one can not always find a_j and b_l such that

$$\sum_{j=1}^{d_A} \sum_{l=1}^{d_B} c_{j,l} |j_A, l_B\rangle = \sum_{j=1}^{d_A} a_j |j_A\rangle \sum_{l=1}^{d_B} b_l |l_B\rangle \quad (2.34)$$

Definition: States, that can be decomposed into a product of states for their subsystems, as in Eq. (2.27), are called *product states*.

Definition: States, that cannot be decomposed into a product of states for their subsystems, as in Eq. (2.27), are called *entangled*.

Most states are entangled.

Example Two two-level systems are in the state

$$|\Psi\rangle = \sum_{j,l=1}^2 c_{j,l} |j, l\rangle = \frac{1}{\sqrt{2}} (|1, 1\rangle + |2, 2\rangle) \quad (2.35)$$

Note that there is no way of writing this state as a direct product of a system state and environment state,

$$|\Psi\rangle \neq |\varphi_A\rangle \otimes |\varphi_B\rangle \quad (2.36)$$

whatever the states $|\varphi_A\rangle$ and $|\varphi_B\rangle$.

Exercise Show equation (2.36).

Exercise For the product state $|\Psi\rangle = |\varphi_A\rangle \otimes |\varphi_B\rangle$ with $|\varphi_A\rangle = \sum_{j=1}^{d_A} a_j |j\rangle$ and $|\varphi_B\rangle = \sum_{\mu=1}^{d_B} b_\mu |\mu\rangle$, how do the coefficients $c_{j,\mu}$ as given in equation (2.33) read?

Entanglement

Entanglement is an important resource for the speed-up of quantum computation as compared to classical computing.

For an entangled state $|\Psi\rangle$, as given in equation (2.36), what is the state of subsystem A ?

→ There is no definite answer: The state of subsystem A cannot be fully specified.

Note that this property does not arise because we would not have sufficient information about the state. The joint state $|\Psi\rangle$ is a pure state. Although we perfectly know the joint state we cannot fully know the state of a subsystem.

→ Some properties of the joint system $A + B$ cannot be assigned to either of the subsystems A or B . They are not “local”.

2.2.2 The exponential Hilbert space

Here is the central reason why quantum computing is powerful: for a composite quantum system the dimension of the Hilbert space grows exponentially in the number of subsystems.

We have seen that arbitrary superpositions of basis states are possible quantum states.

The most general state for a set of n qubits is:

- for 2 qubits

$$|\psi\rangle = c_{00} |00\rangle + c_{01} |01\rangle + c_{10} |10\rangle + c_{11} |11\rangle \quad (2.37)$$

- for 3 qubits

$$\begin{aligned} |\psi\rangle &= c_{000} |000\rangle + c_{001} |001\rangle + c_{010} |010\rangle + c_{011} |011\rangle \\ &= c_{100} |100\rangle + c_{101} |101\rangle + c_{110} |110\rangle + c_{111} |111\rangle \end{aligned} \quad (2.38)$$

- for 4 qubits

$$\begin{aligned} |\psi\rangle &= c_{0000} |0000\rangle + c_{0001} |0001\rangle + c_{0010} |0010\rangle + c_{0011} |0011\rangle \\ &= c_{0100} |0100\rangle + c_{0101} |0101\rangle + c_{0110} |0110\rangle + c_{0111} |0111\rangle \\ &= c_{1000} |1000\rangle + c_{1001} |1001\rangle + c_{1010} |1010\rangle + c_{1011} |1011\rangle \\ &= c_{1100} |1100\rangle + c_{1101} |1101\rangle + c_{1110} |1110\rangle + c_{1111} |1111\rangle \end{aligned} \quad (2.39)$$

We see that the number of terms in the expansions for these states doubles with each qubit we add.

⇒ For specifying an arbitrary state of n qubits, we need to specify 2^n coefficients $c_{jkl\dots}$

⇒ The Hilbert space of n qubits has dimension 2^n

⇒ If we use n qubits as a quantum computer, they can process 2^n complex numbers in parallel.

300 qubits: 2^{300} is larger than the number of atoms in the (visible) universe.

There are already quantum computers with this number of qubits, but these qubits have imperfections.

Importance of entanglement consider a product state

$$|\psi_p\rangle = (\alpha_1 |0_1\rangle + \beta_1 |1_1\rangle) \otimes (\alpha_2 |0_2\rangle + \beta_2 |1_2\rangle) \otimes \dots \otimes (\alpha_n |0_n\rangle + \beta_n |1_n\rangle) \quad (2.40)$$

⇒ only need to specify $2n$ parameters to identify the state → linear in n and not exponential

⇒ need entangled states to get advantage over classical computation

2.2.3 Operators

An operator acting on the Hilbert space $\mathcal{H}_A$ can be written as

$$\hat{O}_A = \sum_{j,l=1}^{d_A} o_{j,l} |j\rangle \langle l| \quad (2.41)$$

and

$$\hat{O}_A |\varphi_A\rangle = \sum_{j,l=1}^{d_A} o_{j,l} a_l |j\rangle \quad (2.42)$$

is again a state in $\mathcal{H}_A$

For the bipartite system composed of subsystems A and B , we have for a state

$$|\varphi\rangle = \sum_{j=1}^{d_A} \sum_{l=1}^{d_B} c_{j,l} |j_A, l_B\rangle, \quad (2.43)$$

where d_B is the dimension of $\mathcal{H}_B$. and for an operator acting on both, A and B , we have the representation

$$\hat{O} = \sum_{j,l=1}^{d_A} \sum_{\mu,\nu=1}^{d_B} o_{j,\mu;l,\nu} |j, \mu\rangle \langle l, \nu| \quad (2.44)$$

As for states, some operators have product form

$$\hat{O} = \hat{A}_A \otimes \hat{B}_B \quad (2.45)$$

For a product operator $\hat{A}_A \otimes \hat{B}_B$, the matrix representation reads,

$$\langle j, \mu | \hat{A}_A \otimes \hat{B}_B | l, \nu \rangle = \langle j | \hat{A}_A | l \rangle \langle \mu | \hat{B}_B | \nu \rangle \quad (2.46)$$

or

$$\begin{bmatrix} a_{1,1}B & a_{1,2}B & \dots \\ a_{2,1}B & a_{2,2}B & \dots \\ \vdots & \vdots & \end{bmatrix} \quad (2.47)$$

where the $a_{j,l}$ are the matrix elements of $\hat{A}_A$ and B is the matrix representing $\hat{B}_B$, i.e.

$$B = \begin{bmatrix} b_{1,1} & b_{1,2} & \dots \\ b_{2,1} & b_{2,2} & \dots \\ \vdots & \vdots & \end{bmatrix} \quad (2.48)$$

Exercise Write the matrix representation of $\hat{\sigma}_x \otimes \hat{\sigma}_y$ in the basis where the matrix representation of $\hat{\sigma}_z$ is

$$\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad (2.49)$$

2.2.4 Density matrices

For an entangled state $|\psi\rangle$, as given in equation (2.36), what is the state of subsystem A ?

→ There is no definite answer: The state of subsystem A cannot be fully specified.

To find out what state the subsystem S could be in, let us look at an arbitrary “local” observable A_A , which only acts on the Hilbert space $\mathcal{H}_A$,

$$A_A = A \otimes \mathbb{1}_B \quad (2.50)$$

We are interested in its expectation value $\langle \Psi | A_A | \Psi \rangle$ for the state

$$|\psi\rangle = \sum_{j,\mu} c_{j,\mu} |j\rangle |\mu\rangle \quad (2.51)$$

and get

$$\langle \Psi | A_A | \Psi \rangle = \sum_{j,l} \sum_{\mu,\nu} c_{l,\nu}^* c_{j,\mu} \langle l | A | j \rangle \langle \nu | \mu \rangle = \sum_{j,l} \langle l | A | j \rangle \sum_{\mu} c_{l,\mu}^* c_{j,\mu} \quad (2.52)$$

where we have used $\langle \nu | \mu \rangle = \delta_{\mu,\nu}$.

Definition For the state $|\psi\rangle$ we also define the *reduced density matrix*

$$\rho_A = \text{Tr}_B(|\Psi\rangle \langle \Psi|) = \sum_{j,l} \sum_{\mu,\nu,\sigma} c_{l,\nu}^* c_{j,\sigma} |j\rangle \langle \mu | \sigma \rangle \langle l | \langle \nu | \mu \rangle = \sum_{j,l} \sum_{\mu} c_{l,\mu}^* c_{j,\mu} |j\rangle \langle l| \quad (2.53)$$

and calculate the expectation value of A for the reduced density matrix ρ_A ,

$$\text{Tr}_A(A \rho_A) = \sum_{j,l,m} \sum_{\mu} c_{l,\mu}^* c_{j,\mu} \langle m | A | j \rangle \langle l | m \rangle = \sum_{j,l} \langle l | A | j \rangle \sum_{\mu} c_{l,\mu}^* c_{j,\mu} \quad (2.54)$$

Comparing equations (2.52) and (2.54), we see that the expectation value of A_A for the full state $|\psi\rangle$ and the expectation value of A for the reduced density matrix ρ_A are identical,

$$\langle \Psi | A_A | \Psi \rangle = \text{Tr}_A(A \rho_A) \quad (2.55)$$

Hence all information about the dynamics of the subsystem A is contained in the reduced density matrix ρ_A .

Properties of density matrices The reduced density matrix and all density matrices ρ have the following properties:

- they are Hermitian

$$\rho^\dagger = \rho \quad (2.56)$$

- they are not necessarily projectors

$$\rho^2 \neq \rho \quad (2.57)$$

as compared to $(|\psi\rangle \langle \psi|)^2 = |\psi\rangle \langle \psi|$.

- their diagonal elements are positive

$$(\rho_A)_{m,m} = \sum_{\mu} |c_{m,\mu}|^2 \geq 0 \quad (2.58)$$

and sum up to unity

$$\text{Tr}[\rho_A] = \sum_m (\rho_A)_{m,m} = \sum_m \sum_\mu |c_{m,\mu}|^2 = 1. \quad (2.59)$$

The diagonal element $(\rho_A)_{m,m}$ can therefore be interpreted as the probability that the system is in the state $|m\rangle$.

- can diagonalise $\rho \rightarrow$

$$\rho = \sum_{\tilde{k}} p_{\tilde{k}} |\tilde{k}\rangle \langle \tilde{k}| \quad (2.60)$$

$\Rightarrow$

$$\text{Tr}(A\rho) = \sum_{\tilde{k}} p_{\tilde{k}} \langle \tilde{k}|A|\tilde{k}\rangle \quad (2.61)$$

This means the system is in state $|\tilde{k}\rangle$ with probability $p_{\tilde{k}}$ and therefore the expectation value $\text{Tr}(A\rho)$ is the sum of the expectation values $\langle \tilde{k}|A|\tilde{k}\rangle$, weighted by the probabilities $p_{\tilde{k}}$.

Exercise Calculate the reduced density matrix of the system A for the state $|\psi\rangle$ given in equation (2.35). What does this reduced density matrix tell about the state of the system A ?

Evolution

The time evolution of a state $|\psi\rangle$, generated by a Hamiltonian H is

$$|\psi, t\rangle = e^{-iH(t-t_0)} |\psi, t_0\rangle \quad (2.62)$$

and can be written in terms of a unitary time evolution operator

$$U(t, t_0) = e^{-iH(t-t_0)} \longrightarrow |\psi, t\rangle = U(t, t_0) |\psi, t_0\rangle \quad (2.63)$$

Considering the definition of a density matrix in Eq. (2.53),

$$\rho = \sum_{j,l} \rho_{j,l} |j\rangle \langle l| \quad (2.64)$$

it follows that the time evolution of a density matrix is given by,

$$\rho(t) = U(t, t_0) \rho(t_0) U^\dagger(t, t_0) \quad (2.65)$$

2.2.5 Dissipation

It is not possible to isolate a device or a sample that we investigate in an experiment from the rest of the world (universe).

One may see this from a practical perspective: The experiment will be in a laboratory and the device we investigate needs to be held by some frame or table.

Yet, also from a fundamental perspective, everything is surrounded by the vacuum of quantum fields. For this reason, excited atoms emit radiation.

In addition we can only get information about the sample that we investigate if we interact with it. So there will be a coupling to the readout apparatus and observer.

⇒ No quantum system will ever be fully isolated.

⇒ This unavoidably leads to errors in a quantum computation

We thus can view every experiment or quantum computation as a bipartite systems, the system proper S that we are interested in and the "rest of the universe", its environment E , see figure 2.2.



Figure 2.2: The system proper S couples to an environment E via an interaction I .

Note that since we should expect that the system S is entangled with the environment E (the rest of the universe), we need to consider a reduced density matrix ρ_S for its quantum state.

→ the evolution of the system density matrix is a so called *universal dynamical map* (UDM) and can always be written as

$$\rho_S(t_1) = \sum_{\alpha} K_{\alpha}(t_1, t_0) \rho_S(t_0) K_{\alpha}^{\dagger}(t_1, t_0) \quad (2.66)$$

which is also known as the *Kraus decomposition*.

→ since the trace should be conserved for any density matrix, we must have,

$$\sum_{\alpha} K_{\alpha}^{\dagger}(t_1, t_0) K_{\alpha}(t_1, t_0) = \mathbb{1} \quad (2.67)$$

→ *universal dynamical maps* (UDMs) that are Markovian, i.e. where the future evolution only depends on the initial state but not on the evolution in the past (this is often the case), can be written in terms of a differential equation, the *master equation*.

Example The master equation for a qubit is

$$\begin{aligned} \frac{d}{dt} \rho(t) = & -i[H, \rho(t)] \\ & + \frac{\gamma_r}{2} (2\sigma^{-} \rho(t) \sigma^{+} - \sigma^{+} \sigma^{-} \rho(t) - \rho(t) \sigma^{+} \sigma^{-}) \\ & + \frac{\gamma_d}{2} (2\sigma^z \rho(t) \sigma^z - 2\rho(t)) \end{aligned} \quad (2.68)$$

where $\sigma^{-} = |0\rangle \langle 1|$, $\sigma^{+} = |1\rangle \langle 0|$ and we have used that $\sigma^z \sigma^z = \mathbb{1}$.

In equation (2.68):

- the first line describes the unitary evolution according to the Hamiltonian H
- the second line describes a relaxation process, this emerges because the qubit can exchange energy with the environment. Here we have chosen an environment in the ground state so that the qubit only loses energy. This can describe spontaneous emission of radiation.

- the third line describes a dephasing process in which the off-diagonal elements of the density matrix are exponentially damped. This is a consequence of interactions with the environment that shift the energy of the state $|1\rangle$ up and down and make it fluctuate.

The above processes are the main processes through which the environment affects the qubit's capability to store and process information in an unperturbed way.

→ there are typically further errors due to insufficient precision of the mechanisms of the gates and the control pulses.

Exercise For a Hamiltonian

$$H = \omega\sigma^z \tag{2.69}$$

and an initial state

$$|\psi(t_0)\rangle = \frac{1}{\sqrt{2}}(|1\rangle + |0\rangle) \tag{2.70}$$

calculate the time evolution of the density matrix and the expectations values $\langle\sigma^x\rangle$ and $\langle\sigma^z\rangle$.

Chapter 3

Quantum Computing Basics

In this chapter we discuss some ingredients that quantum computing is built on.

- quantum operations are unitary operations $\rightarrow$ they are invertible:

$$|\psi_f\rangle = U |\psi_i\rangle \quad \text{where} \quad U = \mathcal{T}_t \exp \left(-i \int_{t_i}^{t_f} ds H(s) \right) \quad \text{and} \quad U^{-1} = U^\dagger \quad (3.1)$$

Here $\mathcal{T}_t$ is the time ordering operator

Equation (3.1) is the solution to the Schrödinger equation with time dependent Hamiltonian $H(t)$.

$$i\hbar \partial_t |\psi\rangle = H(t) |\psi\rangle \quad (3.2)$$

If we control a quantum system via external, classical controls, as we do in a quantum computer, the evolution of the quantum processor is described by such an equation.

Proof of unitarity: Divide $t_f - t_i$ into small time steps $\delta t = t_j - t_{j-1}$ with $n \gg 1$, $t_n = t_f$ and $t_0 = t_i \Rightarrow$

$$\begin{aligned} U &= \mathcal{T}_t e^{-i \int_{t_0}^{t_1} ds H(s)} \approx \prod_{j=0}^{n-1} e^{-iH(t_j)\delta t} \\ \Rightarrow U^\dagger U &= e^{iH(t_0)\delta t} e^{iH(t_1)\delta t} \dots e^{iH(t_{n-1})\delta t} e^{-iH(t_{n-1})\delta t} \dots e^{-iH(t_0)\delta t} \end{aligned}$$

$\rightarrow$ classical logical operations are not always invertible $\rightarrow$ e.g. the AND operation

- quantum operations can act on all components of a superposition in parallel:

$$U \sum_{j_1, j_2, \dots, j_N=0}^1 c_{j_1, j_2, \dots, j_N} |j_1, j_2, \dots, j_N\rangle = \sum_{j_1, j_2, \dots, j_N=0}^1 c_{j_1, j_2, \dots, j_N} U |j_1, j_2, \dots, j_N\rangle \quad (3.3)$$

$\rightarrow$ even if U only acts on two qubits, say qubits 1 and 2, it affects every component $|j_1, j_2, \dots, j_N\rangle$ since,

$$U |j_l, j_{l+1}\rangle = \sum_{k_l, k_{l+1}=0}^1 \alpha_{k_l, k_{l+1}} |k_l, k_{l+1}\rangle \quad (3.4)$$

- read-out is probabilistic:

For a state $\alpha|0\rangle + \beta|1\rangle$, we will measure 0 with probability $|\alpha|^2$ and 1 with probability $|\beta|^2$, where $|\alpha|^2 + |\beta|^2 = 1$

$\Rightarrow$ single measurement does not tell us anything about values of α and $\beta \Rightarrow$ doesn't tell us anything about the state.

$\rightarrow$ if we measure 0, we only know that $\alpha \neq 0$ but nothing more.

$\rightarrow$ if we measure many times, we can learn $|\alpha|^2$ ($\approx$ number of 0s / number of measurements)

$\Rightarrow$ quantum computations only have probabilistic results $\rightarrow$ a classical output occurs with some frequency

- measurements can influence qubits that are not measured:

consider arbitrary state

$$|\psi\rangle = \sum_{\vec{x} \in \{0,1\}^N} c_{\vec{x}} |\vec{x}\rangle \quad (3.5)$$

and suppose we measure qubits $1, 2, \dots, m$ ($m < N$).

can rewrite this as

$$|\psi\rangle = \sum_{\vec{y} \in \{0,1\}^m} b_{\vec{y}} |y_1, y_2, \dots, y_m\rangle |\phi_{\vec{y}}\rangle \quad \text{with} \quad |\phi_{\vec{y}}\rangle = \sum_{\vec{z} \in \{0,1\}^{N-m}} \tilde{c}_{\vec{z}}(\vec{y}) |z_{m+1}, z_{m+2}, \dots, z_N\rangle \quad (3.6)$$

$\rightarrow$ obtain $y_1, y_2, \dots, y_m$ with probability $|b_{\vec{y}}|^2$ and thus project onto $|y_1, y_2, \dots, y_m\rangle |\phi_{\vec{y}}\rangle$

$\Rightarrow$ can change properties of state of qubits $m, m+1, \dots, N$, c.f. section 2.1.4.

$\rightarrow$ origin of this effect is entanglement between qubits $1 - m$ and qubits $m - N$

These observations lead to several concepts or models of quantum computation. We will here discuss or mention two of them.

3.1 Models of Quantum Computation

Quantum computation can be performed according to multiple models. The most prominent ones are:

- gate based quantum computing
- adiabatic quantum computing
- measurement based quantum computing

Here we will discuss the gate based and the adiabatic approach as these are the most spread and most advanced in terms of implementation.

A good review on measurement based quantum computing is [Nature Physics **5**, 19-26 (2009)].

3.1.1 Gate model of Quantum Computing

This is the most prominent approach and we will focus on it throughout the remainder of the course.

The general concept for algorithms in this model is:

1. encode classical input data into an initial state $\rightarrow$ typically a product state
2. generate superposition of many product states that is suitable for solving the problem $\rightarrow$ typically via Hadamard gates
3. apply sequence of quantum gates (quantum operations) to superposition state $\rightarrow$ does operations on all components of superposition in parallel (the gates are implemented via time dependent Hamiltonians, see Eq. (3.1).)
4. map large superposition to states, where a specific classical output (the solution) occurs with high probability

It is the step 3, which is at the origin of the huge computational power of quantum computers

3.1.2 Adiabatic Quantum Computing

Adiabatic quantum computing makes use of the adiabatic theorem of quantum mechanics.

For a time dependent Hamiltonian $H(t)$, the adiabatic theorem says that a quantum system that is initially prepared in the lowest energy eigenstate of H ($H(t_0) |\phi_0(t_0)\rangle = E_j |\phi_0(t_0)\rangle$) will remain in this eigenstate $|\phi_0(t)\rangle$ provided $H(t)$ has an energy gap between ground and first excited state, and changes slowly enough.

The evolution is then said to be adiabatic.

To use this for computing, suppose we want to find the minimum of a function $f(\vec{x})$ of N binary numbers $x_j \in \{0, 1\}$, i.e. $f : \{0, 1\}^N \rightarrow \mathbb{R}$

$\rightarrow$ can formulate any combinatorial search problem in this way.

now construct the Hamiltonian

$$H_C = \sum_{\vec{x} \in \{0,1\}^N} f(\vec{x}) |\vec{x}\rangle \langle \vec{x}| \quad (3.7)$$

where $|\vec{x}\rangle = |x_1, x_2, \dots, x_N\rangle$.

$\rightarrow H_C$ is often easy to construct using σ^z Pauli operators.

$\Rightarrow$ if we find the ground state of H_C , we have solved the problem.

to get ground state of H_C , adiabatic quantum computation considers the Hamiltonian

$$H_{adia}(t) = \lambda(t)H_C - [1 - \lambda(t)]H_B \quad (3.8)$$

where

- λ is ramped slowly from $\lambda(t=0) = 0$ to the final value $\lambda(t=t_f) = 1$

- the initial state $|\Psi_i\rangle$ is the ground state of $-H_B$, which can be easily prepared

If the ramp of $\lambda(t)$ is slow enough $\rightarrow$ evolution adiabatic: system (quantum chip) will remain in its ground state

$\Rightarrow$ final state $|\Psi_f\rangle$ is the ground state of H_C

$\Rightarrow$ optimization problem solved

Main question: How slow is slow enough?

This depends on the minimal energy gap Δ_E between ground and first excited state that is encountered

$$\Delta_E = \min_{t \in [t_0, t_f]} [E_1(t) - E_0(t)] \quad (3.9)$$

The main question is thus how small this spectral gap Δ_E is.

In the following we will discuss gate based quantum algorithms.

Chapter 4

Quantum Algorithms for Fault Tolerant Quantum Computers

In this chapter we discuss some of the most important quantum algorithms that are known to date.

These algorithms work for quantum computers that work without errors.

In real existing quantum computers the quantum gates do not work perfectly and have errors. Moreover it is very unlikely that these errors can ever be suppressed to a level where they would no longer matter.

Fortunately there is a strategy to correct for errors called "Quantum Error Correction", which we will discuss in chapter 5.

By employing quantum error correction, the algorithms discussed in this chapter can be run even on faulty quantum computers.

Here, we are first going to introduce a very simple quantum algorithm which however contains the essential idea of why quantum computing is powerful.

4.1 Deutsch's Problem

An elementary example of a quantum algorithm that can solve a problem faster than any classical algorithm.

Does not provide a dramatic speedup but contains the main ingredients that make quantum algorithms fast.

The problem addresses a very simple question:

Deutsch's question "Is this switch connected to that light bulb"

Classical Algorithm Test whether the light is on for both positions of the switch.

One can also state this problem more generally and more formally.

4.1.1 Deutsch's Problem: formal version

Consider a function

$$f : \{0, 1\} \rightarrow \{0, 1\} \quad (4.1)$$

Question: Is

$$f(0) = f(1)? \quad \rightarrow \text{"not connected"} \quad (4.2)$$

$$f(0) \neq f(1)? \quad \rightarrow \text{"connected"} \quad (4.3)$$

Classical solution: must evaluate $f(0)$ and $f(1)$.

While the classical answer seems obvious, it is interesting because we need to evaluate two things for only getting one bit of information (there are only two possible answers "yes" or "no", i.e. 1 or 0)

→ Need to "query f twice".

Main result A quantum computer can solve this problem with **only one** query.

The result is quantitatively not very exciting: quantum computers are faster by a factor 2.

It is however qualitatively very exciting: Classically it is very obvious that two queries are needed and there appears no way what so ever to reduce this.

⇒ quantum computers compute in fundamentally different ways.

To explain the quantum algorithm, we will consider a physical realization with an interferometer.

Interferometer algorithm

Consider the interferometer sketched in figure 4.1:

The action of a 50/50 beam splitter is

$$H |j\rangle = \frac{1}{\sqrt{2}} [|0\rangle + (-1)^j |1\rangle] \quad (4.4)$$

The fact that the phase of the second state depends on j is necessary to make the map invertible, i.e. it shouldn't map two distinct inputs onto the same output. In a beam splitter, for one path, the photons acquire a phase shift of π corresponding to a "−" sign.

This is because the Schrödinger equation generates unitary dynamics which is invertible,
⇒ quantum gates are invertible.

The transformation H is thus an invertible, unitary transformation with property

$$H^\dagger H = \mathbb{1} \quad (4.5)$$

The transformation H has a matrix representation that is called Hadamard matrix (hence the " H ")

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (4.6)$$

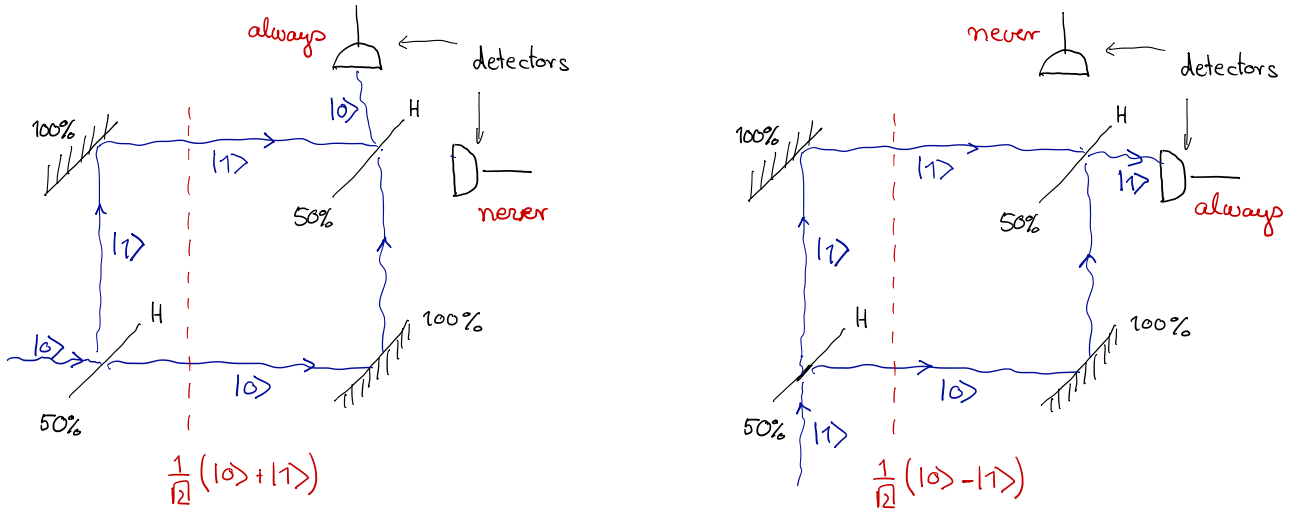


Figure 4.1: Mach-Zehnder interferometer. **Left:** If a photon comes in horizontally, the detector in vertical direction will always fire. Between the two beam-splitters, the state is a symmetric superposition of $|0\rangle$ and $|1\rangle$. **Right:** If a photon comes in vertically, the detector in horizontal direction will always fire. Between the two beam-splitters, the state is an anti-symmetric superposition of $|0\rangle$ and $|1\rangle$.

Hence,

$$H^2 = \mathbb{1} \quad (4.7)$$

where the second Hadamard just describes the effect of the second beam-splitter.

Now let's consider putting a piece of glass into one of the interferometer arms. The situation is sketched in figure 4.2.

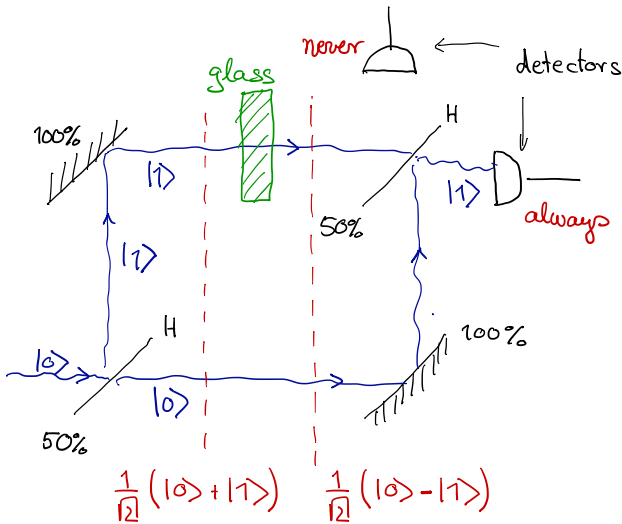


Figure 4.2: Mach-Zehnder interferometer with glass plate. The glass generates a phase shift of π and thus redirects the photon to the other detector.

A glass of the right sickness has the effect of putting a "-" sign as a pre-factor to the state

$$|j\rangle \rightarrow -|j\rangle \quad (4.8)$$

In the arrangement of figure 4.2, the glass thus executes the transformation

$$\frac{1}{\sqrt{2}}[|0\rangle + |1\rangle] \rightarrow \frac{1}{\sqrt{2}}[|0\rangle - |1\rangle] \quad (4.9)$$

Now suppose a machine inserts the glass plates according to the function f introduced in equation (4.1). This generates the transformation

$$|j\rangle \rightarrow (-1)^{f(j)} |j\rangle \quad (4.10)$$

I.e. if $f(0) = 1$ it puts the glass in the lower branch and vice versa.

→ We assume f to be implemented as an oracle (a black box) $\Rightarrow$ machine that puts glass in branches according to f is readily available.

Assuming the input states is $|0\rangle$, this will result in the state

$$|\psi\rangle = \frac{1}{\sqrt{2}}[(-1)^{f(0)} |0\rangle + (-1)^{f(1)} |1\rangle] \quad (4.11)$$

after the glass plates, see in figure 4.3.

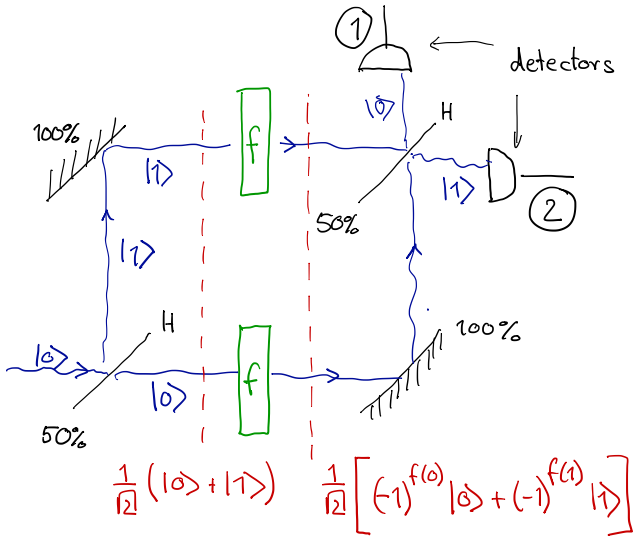


Figure 4.3: Mach-Zehnder interferometer with glass plates in both arms according to the function f .

Thus

$$\text{If } f(0) = f(1) \text{ then } |\psi\rangle \propto \frac{1}{\sqrt{2}}[|0\rangle + |1\rangle] \text{ and detector 1 fires} \quad (4.12)$$

$$\text{If } f(0) \neq f(1) \text{ then } |\psi\rangle \propto \frac{1}{\sqrt{2}}[|0\rangle - |1\rangle] \text{ and detector 2 fires} \quad (4.13)$$

Note that assuming an input state $|1\rangle$ would just exchange the roles of the two detectors, but the protocol will work in both cases.

Thus if the transformation (4.11) is readily implemented, one single application of it solves the problem.

The transformation can be very easily implemented using an ancillary qubit in an arbitrary state $|k\rangle_a$. Let us consider

$$U_f |j\rangle |k\rangle_a \longrightarrow |j\rangle |f(j) \oplus k\rangle_a \quad (4.14)$$

where the XOR operation $\oplus$ means addition modulo 2,

$$j \oplus l = 0 \quad \text{if} \quad j = l \quad (4.15)$$

$$j \oplus l = 1 \quad \text{if} \quad j \neq l \quad (4.16)$$

If we now consider preparing the ancillary qubit in the state $(|0\rangle - |1\rangle)/\sqrt{2}$, we find (ignoring the $1/\sqrt{2}$)

$$U_f |j\rangle (|0\rangle - |1\rangle)_a = |j\rangle |f(j)\rangle_a - |j\rangle |f(j) \oplus 1\rangle_a = \begin{cases} |j\rangle (|0\rangle - |1\rangle)_a & \text{if } f(j) = 0 \\ |j\rangle (|1\rangle - |0\rangle)_a & \text{if } f(j) = 1 \end{cases} \quad (4.17)$$

Hence

$$U_f |j\rangle \frac{1}{\sqrt{2}}(|0\rangle_a - |1\rangle_a) = (-1)^{f(j)} |j\rangle \frac{1}{\sqrt{2}}(|0\rangle_a - |1\rangle_a) \quad (4.18)$$

The ancilla qubit was needed to make the operation invertible (also if $f(0) = f(1)$). Remember that quantum operations are unitary and thus invertible. The map in equation (4.10) on the other hand is not invertible if $f(0) = f(1)$.

We thus have the following algorithm

1. apply Hadamard
2. prepare ancilla qubit in state $(|0\rangle - |1\rangle)/\sqrt{2}$
3. apply U_f
4. apply Hadamard
5. discard ancilla qubit

The important point is that only one application of U_f is needed! (in classical computing we need to evaluate f twice)

We can also present the algorithm as a quantum circuit (as opposed to thinking of beam splitters)

Quantum circuits We first introduce some graphic notation for quantum circuits.

- a horizontal line denotes the time-line of a qubit
- a box in a qubit line denotes a gate acting only on this qubit
→ for example the Hadamard gate is sketched as

$$|j\rangle \longrightarrow \boxed{H} \longrightarrow \frac{1}{\sqrt{2}}(|0\rangle + (-1)^j |1\rangle) \quad (4.19)$$

- a box across two qubit lines denotes a gate acting on these two qubits

$$\begin{array}{c} \text{---} \boxed{U} \text{---} \\ \text{---} \end{array} \quad (4.20)$$

As an example, the action of the empty interferometer can as a circuit be sketched as

$$|0\rangle \longrightarrow \boxed{H} \longrightarrow \boxed{H} \longrightarrow |0\rangle$$

Quantum circuit for solution of Deutsch's problem The circuit for solving Deutsch's problem can be sketched as,

$$\begin{array}{c}
 |0\rangle \text{ --- } [H] \text{ --- } [U_f] \text{ --- } [H] \text{ --- } \text{Measurement} \\
 |1\rangle_a \text{ --- } [H] \text{ --- } [U_f] \text{ --- } \text{---}
 \end{array} \tag{4.21}$$

where the action of U_f is as described in equation (4.14).

The circuit or algorithm (4.21) is called *Deutsch* algorithm.

Importantly this only requires one application of U_f .

It however evaluates f on a suitable *superposition* of the two possibilities and interferes these in a suitable way.

Possible circuits for implementing the unitary U_f are provided in table 4.1.

Conclusion Quantum computer can solve Deutsch's problem with just one query (one application of U_f), which probes both $f(0)$ and $f(1)$ at the same time.

We however don't learn both, $f(0)$ and $f(1)$. We only learn a suitable global property using interference.

Before we discuss more complex quantum algorithms, we will go through a few basics of quantum circuits. We start by discussing classical circuits first.

4.2 Classical Circuits

In classical complexity theory, a Boolean circuit is a finite directed acyclic graph (it doesn't contain circles) with AND, OR, and NOT gates.

- it has n input nodes, which contain the n input bits ($n \geq 0$)
- the internal nodes are AND, OR, and NOT gates
- there are $m \geq 1$ designated output nodes
- it computes some Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$
- AND and NOT gates suffice to compose any Boolean circuit

4.3 Quantum Circuits

Quantum algorithms have the following structure:

- prepare initial state $\rightarrow$ the input
- apply sequence of quantum gates, such as in Eqs. (4.19), (4.20) or (4.21)
- measure state of some qubits $\rightarrow$ the output

Often a feedback loop is also included. This means that the applied gates or initial states are chosen according to the result of a measurement on another qubit.

For the Deutsch algorithm, the circuit is sketched in equation (4.21).

Remarks

- initial state often prepared from

$$|\vec{0}\rangle = |0, 0, \dots, 0\rangle \quad (4.22)$$

by applying gates to it.

Qubits are naturally initialized in $|\vec{0}\rangle$, e.g. by cooling. $|1\rangle$ is the excited state and cooling takes away excitations.

- Why do we chose two-level systems for our qubits?

This is an arbitrary choice. We could use any other. It would not change things significantly.

We now turn to introduce some useful quantum gates.

4.3.1 Gates

A quantum gate is a unitary transformation acting on some qubits.

$$|\tilde{\psi}\rangle = U |\psi\rangle \quad (4.23)$$

Quantum gates can be grouped according to the number of qubits they act on.

See also: https://en.wikipedia.org/wiki/Quantum_logic_gate

We only mention the important gates and start with gates that only act on one qubit.

We use matrix representations according to the following convention

$$|0\rangle \hat{=} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{and} \quad |1\rangle \hat{=} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (4.24)$$

Single qubit gates

- "Hadamard" gate

$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad , \quad |j\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle + (-1)^j |1\rangle) \quad (4.25)$$

- "Pauli-X", "bit-flip" or "Not" gate

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad , \quad |j\rangle \xrightarrow{X} |j \oplus 1\rangle \quad (4.26)$$

- "Pauli-Y" gate

$$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad , \quad |j\rangle \xrightarrow{Y} i(-1)^j |j \oplus 1\rangle \quad (4.27)$$

- "Pauli-Z" gate

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad , \quad |j\rangle \xrightarrow{Z} (-1)^j |j\rangle \quad (4.28)$$

- "S" gate

$$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \quad , \quad |j\rangle \xrightarrow{S} (i)^j |j\rangle \quad (4.29)$$

- "T" gate

$$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix} \quad , \quad |j\rangle \xrightarrow{T} e^{ij\pi/4} |j\rangle \quad (4.30)$$

Two qubit gates

- "Controlled-Not", "C-NOT" or "Controlled-X" gate

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad \begin{array}{c} |j\rangle \\ |k\rangle \end{array} \begin{array}{c} \bullet \\ \oplus \end{array} \begin{array}{c} |j\rangle \\ |k \oplus j\rangle \end{array} \quad (4.31)$$

→ flips the target qubit if the control qubit is in state $|1\rangle$

- "Swap" gate

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \begin{array}{c} |j\rangle \\ |k\rangle \end{array} \begin{array}{c} \times \\ \times \end{array} \begin{array}{c} |k\rangle \\ |j\rangle \end{array} \quad (4.32)$$

→ exchanges the states of both qubits

- "Controlled-Z" or "CZ" gate

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}, \quad \begin{array}{c} |j\rangle \\ |k\rangle \end{array} \begin{array}{c} \bullet \\ \boxed{Z} \end{array} \begin{array}{c} |j\rangle \\ (-1)^j |k\rangle \end{array} \quad (4.33)$$

→ changes sign if both qubits are in state $|1\rangle$ and does nothing otherwise

Exercise Derive the matrices for the C-NOT and CZ gates where the roles of control and target qubit are exchanged. Discuss the difference between C-NOT and CZ gates in this context.

Multi-qubit gates for more qubits There are also quantum gates that act on more than two qubits, particularly three qubit gates, see https://en.wikipedia.org/wiki/Quantum_logic_gate.

- "Toffoli" gate

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad \begin{array}{c} |j\rangle \\ |k\rangle \\ |l\rangle \end{array} \begin{array}{c} \bullet \\ \bullet \\ \oplus \end{array} \begin{array}{c} |j\rangle \\ |k\rangle \\ |l \oplus j \cdot k\rangle \end{array} \quad (4.34)$$

→ flips the target qubit if both control qubits are in state $|1\rangle$

In practice, three-qubit gates are often decomposed into sequences of single- and two-qubit gates.

Circuit for U_f With the above gates, we can now give possible circuits for implementing the unitary U_f that is used in Deutsch's algorithm. These are provided in table 4.1.

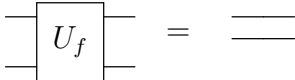
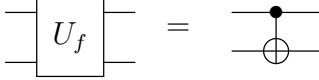
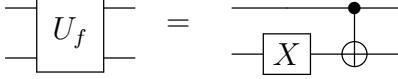
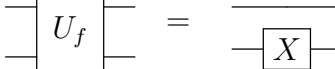
$f(0)$	$f(1)$	circuit for $U_f j\rangle k\rangle_a \rightarrow j\rangle f(j) \oplus k\rangle_a$
0	0	
0	1	
1	0	
1	1	

Table 4.1: Circuits for implementing the unitary U_f for Deutsch's algorithm

4.3.2 Some applications of the C-NOT gate

The C-NOT gate is an entangling gate

$$\left. \begin{array}{c} \alpha |0\rangle + \beta |1\rangle \\ |0\rangle \end{array} \right\} \xrightarrow{\text{C-NOT}} \alpha |00\rangle + \beta |11\rangle \quad (4.35)$$

where the output state cannot be written as a product state, i.e. there are no complex numbers $\alpha_1, \alpha_2, \beta_1, \beta_2$ such that

$$\alpha |00\rangle + \beta |11\rangle \neq (\alpha_1 |0_1\rangle + \beta_1 |1_1\rangle)(\alpha_2 |0_2\rangle + \beta_2 |1_2\rangle) \quad (4.36)$$

Such states are called *entangled*.

We can use the C-NOT gate to transfer a state from one qubit to another.

Quantum state transfer Let us apply a Hadamard gate to qubit 1 at the end of the circuit (4.35).

$$\left. \begin{array}{c} \alpha |0\rangle + \beta |1\rangle \\ |0\rangle \end{array} \right\} \xrightarrow{\text{C-NOT}} \xrightarrow{H} \quad (4.37)$$

The output is now

$$\alpha \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) |0\rangle + \beta \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) |1\rangle = \frac{1}{\sqrt{2}} |0\rangle (\alpha |0\rangle + \beta |1\rangle) + \frac{1}{\sqrt{2}} |1\rangle (\alpha |0\rangle - \beta |1\rangle) \quad (4.38)$$

Now we measure qubit 1:

- in half of the cases we get 0 and the state is $|0\rangle (\alpha |0\rangle + \beta |1\rangle)$
- in the other half of the cases we get 1 and the state is $|1\rangle (\alpha |0\rangle - \beta |1\rangle)$

$\Rightarrow$ if we apply a Z -gate on qubit 2 whenever we measure a 1 for qubit 1,

$$Z(\alpha |0\rangle - \beta |1\rangle) = \alpha |0\rangle + \beta |1\rangle$$

and the final state of qubit two is always

$$\alpha |0\rangle + \beta |1\rangle \quad (4.39)$$

Remember that we use the convention $Z|0\rangle = |0\rangle$ and $Z|1\rangle = -|1\rangle$, see equation (4.28).

The entire circuit can be sketched as

$$\begin{array}{c}
 |\psi\rangle \text{ --- } \bullet \text{ --- } H \text{ --- } \boxed{\text{meter}} \\
 |0\rangle \text{ --- } \oplus \text{ --- } Z^m \text{ --- } |\psi\rangle
 \end{array} \quad (4.40)$$

where

$$\begin{array}{c}
 \boxed{\text{meter}} \\
 \parallel \\
 \boxed{\mathcal{O}^m}
 \end{array} \quad (4.41)$$

symbolizes a measurement of the first qubit with outcome $m = 0, 1$ and a subsequent application of the Pauli gate $\mathcal{O}^m$ ($\mathcal{O} = X, Y$ or Z) to the second qubit.

Since we haven't assumed anything about the amplitudes α and β of the input state of qubit 1, we realize that the circuit transfers an arbitrary input state $|\psi\rangle$ from qubit 1 to qubit 2.

State transfer is nice but not that spectacular. It is not surprising that one can transfer a state from one qubit to another if the qubits interact at some point (and they do during a C-NOT gate).

If combined with suitable entangled states, quantum state transfer can however lead to a much more spectacular application that we will discuss now.

Quantum Teleportation (*Bennett et al. 1993*)

Transfer of a quantum state from one qubit to a distant qubit in a possibly remote location.

- the sender and receiver qubits do not need to interact
- sender and receiver qubit need to be entangled with each other
- need 3 qubits in total: qubits 1 and 2 are in the location of the sender and qubit 3 is in the location of the receiver

Quantum teleportation can be realized via the circuit in figure 4.4, where the conditional gates are as defined in equation (4.41).

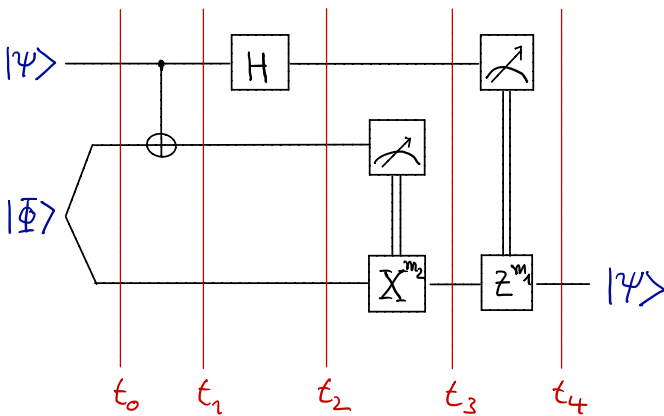


Figure 4.4: Circuit for teleporting an unknown quantum state $|\psi\rangle$. The conditional gates are as defined in equation (4.41).

The input state $|\psi\rangle$ is arbitrary and the input state $|\Phi\rangle$ is the maximally entangled state

$$|\Phi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \quad (4.42)$$

At the times t_0, t_1 and t_2 , we thus have the states

t_0 :

$$\begin{aligned} |\psi\rangle \otimes |\Phi\rangle &= (\alpha|0\rangle + \beta|1\rangle) \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\ &= \frac{\alpha}{\sqrt{2}}|000\rangle + \frac{\beta}{\sqrt{2}}|100\rangle + \frac{\alpha}{\sqrt{2}}|011\rangle + \frac{\beta}{\sqrt{2}}|111\rangle \end{aligned}$$

t_1 :

$$\frac{\alpha}{\sqrt{2}}|000\rangle + \frac{\beta}{\sqrt{2}}|110\rangle + \frac{\alpha}{\sqrt{2}}|011\rangle + \frac{\beta}{\sqrt{2}}|101\rangle$$

t_2 :

$$\begin{aligned} &\frac{\alpha}{2}(|0\rangle + |1\rangle)|00\rangle + \frac{\beta}{2}(|0\rangle - |1\rangle)|10\rangle + \frac{\alpha}{2}(|0\rangle + |1\rangle)|11\rangle + \frac{\beta}{2}(|0\rangle - |1\rangle)|01\rangle \\ &= \frac{1}{2}|00\rangle(\alpha|0\rangle + \beta|1\rangle) + \frac{1}{2}|01\rangle(\beta|0\rangle + \alpha|1\rangle) + \frac{1}{2}|10\rangle(\alpha|0\rangle - \beta|1\rangle) + \frac{1}{2}|11\rangle(-\beta|0\rangle + \alpha|1\rangle) \end{aligned}$$

t_3 : As we discard the measured (middle) qubit, we will drop it in the following equation, $|jl\rangle$ thus denotes the state of qubits 1 and 3,

$$\frac{1}{\sqrt{2}}|0\rangle(\alpha|0\rangle + \beta|1\rangle) + \frac{1}{\sqrt{2}}|1\rangle(\alpha|0\rangle - \beta|1\rangle)$$

where the change in the norm is due to the projection in the measurement, see Eq. (2.21). Note that the third qubit was flipped whenever the second qubit was measured in state $|1\rangle$.

t_4 : As we discard the measured (first) qubit, we will drop it in the following equation, $|j\rangle$ thus denotes the state of qubit 3.

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

We thus find that qubit 3 is in the unknown quantum state that was input into qubit 1.

Some important remarks:

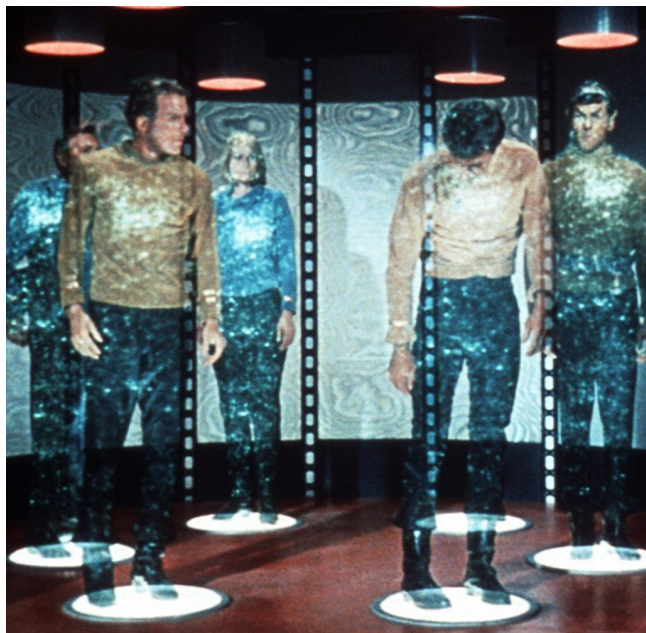
- qubits 2 and 3 need to be prepared in a maximally entangled state at the beginning. They thus need to interact prior to the experiment, but can then be separated.
- the initial state of qubits 2 and 3 is independent of the input in qubit 1
- the time needed to execute the C-NOT and Hadamard gates is independent of the separation of sender and receiver
- Is this in conflict with the theory of relativity? Nothing can travel faster than the speed of light!

→ No: at time t_2 the state seen by the receiver is still fully mixed, i.e. the reduced density matrix of qubit 3 is

$$\rho_3 = \frac{1}{2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

⇒ no information has flown to the receiver yet at t_2

- need to classical communicate the measurement outcomes to the receiver.
→ this can only be done at the speed of light or slower
- essentially could teleport the entire quantum state of a human being and science fiction scenarios could become reality if we were able to implement this.
- such applications would however require transmitting an insanely large amount of classical data.



4.4 Universal sets of gates

A quantum algorithm is a unitary transformation of an input state into an output state.

→ Do we need to implement a different unitary for each quantum algorithm?

Fortunately not: all quantum unitaries can be decomposed into a sequence of single and two qubit gates.

More precisely: For each unitary U , there is a sequence of single- and two-qubit gates that can approximate U arbitrary well.

→ What is the minimal set of gates that we need to be able to implement?

There are many choices of universal sets, i.e. sets of gates, that can be composed to build any algorithm. For example

- $H, T, T^\dagger, CNOT$ with

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix} \quad (4.43)$$

is a universal set.

- $H, X, Z, S, S^\dagger, CNOT$ with

$$S = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/2} \end{bmatrix} \quad (4.44)$$

is not universal.

Note that $S = TT$ so that S is less powerful as it can only lead to algorithms with an even number of T gates.

The difference between S and T gates is in fact more dramatic because the gates H, S and $CNOT$ generate the so called Clifford group.

Clifford Circuits: Clifford circuits are all circuits that can be constructed from the gates H, S and $CNOT$

Gottesmann-Knill Theorem: All quantum algorithms that consist of (i) preparation of initial product states, (ii) Clifford circuits and (iii) measurements of individual qubits can be simulated efficiently on classical computers (in a time that grows polynomially in the size of the input).

→ even though the combination of CNOT and Hadamard gate can generate entanglement, these circuits can be classically simulated

⇒ the presence of entanglement in the states doesn't guarantee a quantum advantage

On the other hand all universal gate sets are equivalent, as the Solovay-Kitaev theorem says.

Solovay-Kitaev Theorem: All finite universal gate sets can efficiently simulate each other. The overhead of going from one to another set can only be polynomial in the size of the input.

⇒ Doesn't matter which finite universal set of gates we implement on the quantum computer we build. → useful as some gates are harder to implement on a specific architecture.

4.5 Can a quantum computer do classical computations efficiently?

Answer: Yes, by translating (transforming) the classical Boolean circuit into a quantum circuit.

There are however two problems:

- quantum gates are unitary and thus reversible
→ need to make circuit reversible first
- quantum circuits are very fragile to decoherence
→ need to minimize exposure to decoherence

Making classical circuit reversible We can make any classical circuit reversible with the following procedure:

Any classical Boolean circuits can be decomposed into AND and NOT gates

NOT is reversible: $\text{NOT} \cdot \text{NOT} = 1$

AND is not reversible: For output 0, there are three possible inputs, 00, 01, and 10. It is thus impossible to determine the input from the output.

Can make classical circuit reversible by using classical Toffoli gates, c.f. Eq. (4.34):

$$\begin{array}{ccc} j & \bullet & j \\ k & \bullet & k \\ l & \oplus & l \oplus j \cdot k \end{array} \quad (4.45)$$

For $l = 0$ will get $j \cdot k$ in 3rd register: $j \cdot k = 1$ only if $j = 1$ and $k = 1 \rightarrow$ AND gate

Toffoli can also make a NOT: for $j = k = 1$ we get $l \rightarrow l \oplus 1 = \neg l$

$\Rightarrow$ Classical computing can be made reversible by using Toffoli gates. Conversion is efficient (i.e. overhead not too bad)

Translating classical into quantum circuits The following steps translate a classical circuit C_f into a quantum circuit U_f (as for example U_f in Deutsch's problem)

Consider a classical circuit C_f that computes a function f :

$$\begin{array}{ccc} x_1 & \text{---} & f_1(\vec{x}) \\ \vdots & & \vdots \\ x_n & \text{---} & f_n(\vec{x}) \end{array} \quad (4.46)$$

Replacing all ANDs with Toffolis leads to a reversible (classical) circuit R_f :

$$\begin{array}{ccc} x_1 & \text{---} & f_1(\vec{x}) \\ \vdots & & \vdots \\ x_n & \text{---} & f_n(\vec{x}) \\ 0 & \text{---} & j_1(\vec{x}) \\ \vdots & & \vdots \\ 0 & \text{---} & j_m(\vec{x}) \end{array} \quad (4.47)$$

where $j_1(\vec{x}), \dots, j_m(\vec{x})$ denotes some m outputs that are not used but depend on the inputs $x_1, \dots, x_n$.

we call it "junk" (hence the "j").

Could convert R_f to quantum circuit

$$\begin{array}{ccc} |x_1\rangle & \text{---} & |f_1(\vec{x})\rangle \\ \vdots & & \vdots \\ |x_n\rangle & \text{---} & |f_n(\vec{x})\rangle \\ |0\rangle & \text{---} & |j_1(\vec{x})\rangle \\ \vdots & & \vdots \\ |0\rangle & \text{---} & |j_m(\vec{x})\rangle \end{array}$$

but the junk output $|j_l(\vec{x})\rangle$ makes the circuit vulnerable to decoherence.

To understand why, let's consider the Deutsch circuit with an additional qubit for the following input

$$\begin{array}{ccccc} |0\rangle & \text{---} & H & \text{---} & H & \text{---} & \text{---} \\ |1\rangle & \text{---} & H & \text{---} & U_f & \text{---} & |-\rangle \\ |0\rangle & \text{---} & & \text{---} & U_f & \text{---} & |j\rangle \end{array} \quad (4.48)$$

After the Hadamards, we have the state $|+\rangle|-\rangle|0\rangle$, where $|\pm\rangle = (|0\rangle \pm |1\rangle)/\sqrt{2}$.

Applying U_f yields

$$U_f |+\rangle|-\rangle|0\rangle = \frac{1}{\sqrt{2}}(-1)^{f(0)}|0\rangle|-\rangle|j(0)\rangle + \frac{1}{\sqrt{2}}(-1)^{f(1)}|1\rangle|-\rangle|j(1)\rangle \quad (4.49)$$

As the 2nd qubit is always in $|-\rangle$ we drop it. Applying the second Hadamard then yields

$$\frac{1}{\sqrt{2}}|0\rangle [(-1)^{f(0)}|j(0)\rangle + (-1)^{f(1)}|j(1)\rangle] + \frac{1}{\sqrt{2}}|1\rangle [(-1)^{f(0)}|j(0)\rangle - (-1)^{f(1)}|j(1)\rangle] \quad (4.50)$$

If $|j(0)\rangle = |j(1)\rangle$, we can drop the junk states and this reduces to the original Deutsch circuit

$$\frac{1}{\sqrt{2}}|0\rangle [(-1)^{f(0)} + (-1)^{f(1)}] + \frac{1}{\sqrt{2}}|1\rangle [(-1)^{f(0)} - (-1)^{f(1)}] \quad (4.51)$$

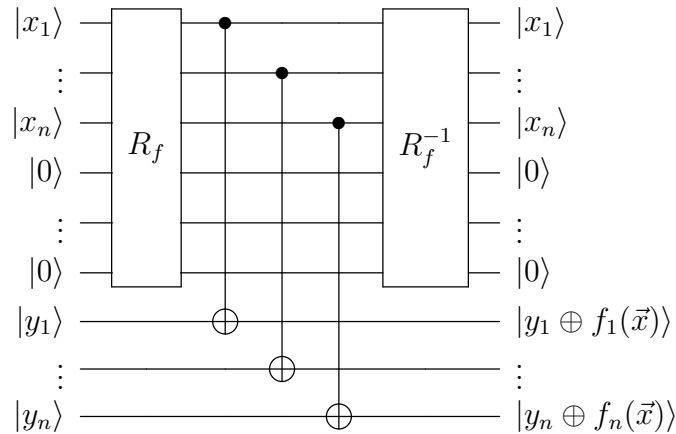
so that we measure 0 if $f(0) = f(1)$ and 1 if $f(0) \neq f(1)$.

If however $\langle j(0)|j(1)\rangle = 0$ and we measure the first qubit, both outcomes have probability $1/2$, independently of f !

$\Rightarrow$ measuring qubit 1 doesn't provide any information as the outcome is completely random.

$\Rightarrow$ Need to make sure that final state of additional qubits is independent of input!

Avoiding junk output The following strategy avoids additional outputs that depend on the input.



Here, applying R_f^{-1} brings the qubits, that do not contribute to the output, back into their input state.

Use $y_1 = y_2 = \dots = y_n = 0$ to get desired output in bottom n qubits.

$\Rightarrow$ Sensibility to "junk" output is avoided.

$\Rightarrow$ There is an efficient and feasible strategy to compute classical circuits on quantum computers.

1. make circuit reversible by using Toffoli gates
2. avoid "junk" output by copying output to ancillas and reversing the circuit on the input qubits afterwards.

4.6 How hard is it to simulate a quantum computer with a classical computer?

This is an important question because it tells us whether quantum computers can be better than classical ones.

If a classical computer can efficiently simulate a quantum computer, it would be as powerful as the quantum computer.

There are mainly two ways to simulate a quantum computer

Schrödinger approach This approach simulates the time evolution of the full quantum state.

At some point in the computation, the quantum processor can be in an arbitrary quantum state. For n qubits, such a state reads

$$|\psi\rangle = \sum_{j_1, j_2, \dots, j_n=0}^1 \alpha_{j_1, j_2, \dots, j_n} |j_1, j_2, \dots, j_n\rangle = \sum_{\vec{j}} \alpha_{\vec{j}} |\vec{j}\rangle \quad (4.52)$$

There are 2^n coefficients $\alpha_{\vec{j}}$.

$\Rightarrow$ from the Schrödinger equation, one gets the 2^n coupled ordinary differential equations,

$$\frac{d}{dt} \alpha_{\vec{j}} = \sum_{\vec{l}} H_{\vec{j}; \vec{l}} \alpha_{\vec{l}} \quad (4.53)$$

where $H_{\vec{j}; \vec{l}} = \langle \vec{j} | H | \vec{l} \rangle$

$\Rightarrow$ A simulation of the evolution of the quantum state can be done by integrating 2^n coupled ordinary differential equations.

$\rightarrow$ Simulating a quantum computer in this way requires an exponential amount of memory.

Feynman approach We can avoid the exponential memory requirement with the Feynman approach.

This is because we do not need to reproduce the entire state to simulate what the quantum computer can tell us.

In a quantum computation we measure the final state. To simulate the quantum computer we thus need to compute the probability for observing a specific bit string $\vec{j}_f$,

$$P(\vec{j}_f) = \left| \langle \vec{j}_f | U | \vec{j}_i \rangle \right|^2 \quad (4.54)$$

where U is the unitary of the algorithm and $|\vec{j}_i\rangle$ the initial state.

Assuming a circuit of size L , we need to compute

$$\langle \vec{j}_f | U | \vec{j}_i \rangle = \langle \vec{j}_f | U_L U_{L-1} \dots U_2 U_1 | \vec{j}_i \rangle \quad (4.55)$$

Inserting identities in the form $\mathbb{1} = \sum_{\vec{l}} |\vec{l}\rangle \langle \vec{l}|$ between each pair of unitaries $U_m U_{m-1}$, we get

$$\begin{aligned} & \langle \vec{j}_f | U_L \left(\sum_{\vec{j}_{L-1}} |\vec{j}_{L-1}\rangle \langle \vec{j}_{L-1}| \right) U_{L-1} \left(\sum_{\vec{j}_{L-2}} |\vec{j}_{L-2}\rangle \langle \vec{j}_{L-2}| \right) U_{L-2} \dots U_2 \left(\sum_{\vec{j}_1} |\vec{j}_1\rangle \langle \vec{j}_1| \right) U_1 |\vec{j}_i\rangle = \\ & = \sum_{\vec{j}_{L-1}} \sum_{\vec{j}_{L-2}} \dots \sum_{\vec{j}_1} \langle \vec{j}_f | U_L |\vec{j}_{L-1}\rangle \langle \vec{j}_{L-1} | U_{L-1} |\vec{j}_{L-2}\rangle \dots \langle \vec{j}_1 | U_1 |\vec{j}_i\rangle \end{aligned} \quad (4.56)$$

The computation thus requires executing $L - 1$ nested loops which add a product of L numbers in each step.

$\Rightarrow$ If $L = \text{poly}(n)$ this requires a memory of size $\text{poly}(n)$. Only need to store all U_l .

Each loop runs over 2^n values!!

$\Rightarrow$ Both simulation approaches require an exponential effort, i.e. the effort grows as 2^n , where n is the number of qubits.

$\Rightarrow$ This effort becomes prohibitively large as n grows.

$\rightarrow$ Currently > 50 qubits and > 14 layers of gates can no longer be simulated.

$\rightarrow$ First experimental demonstration by Google team, see Nature **574**, 505 (2019), confirmed by USTC team in Shanghai in 2021.

$\rightarrow$ Importantly, adding a single qubit doubles the effort of a classical simulation.

$\Rightarrow$ There should be quantum algorithms that solve a problem exponentially faster than any classical algorithm!

$\rightarrow$ We now consider an example of such an algorithm with exponential speedup.

4.7 Simon's problem

Consider an unknown periodic function $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$

I.e. $\exists s$ such that $\forall x, y: f(x) = f(y) \Leftrightarrow y = x \text{ or } y = x \oplus s$.

Here $x \oplus y$ means bit-wise XOR, i.e. $x \oplus y = x_1 \oplus y_1, x_2 \oplus y_2, \dots, x_n \oplus y_n$

Example for f with $s = 110$:

x	$f(x)$
000	101
001	010
010	000
011	110
100	000
101	110
110	101
111	010

Verify: every output occurs twice, e.g. 010 and 100 map to 000, and $010 \oplus 110 = 100$

Task: Find $s \in \{0, 1\}^n$ for unknown f , i.e. need to compute $f(x)$ for each x .

problem might not seem highly relevant, but will guide us to develop highly relevant quantum algorithms

Classical complexity: Classically we need at least $\Omega(2^{n/2})$ function calls.

Notation: $\Omega(f(n))$ means that runtime is at least $k \times f(n)$ as $n \rightarrow \infty$ for some constant k .

Explanation: need to find at least one pair x and y with $f(x) = f(y)$

→ birthday paradox: What is the probability that in a group of n people at least two share the same birthday?

assume m queries → can form $m(m-1) \approx m^2$ pairs (x, y)

for each pair $x \oplus y = c(x, y) \rightarrow$ probability that $c(x, y) = s$ is 2^{-n} as there are 2^n possible bit-strings in $\{0, 1\}^n$

→ need $m^2 2^{-n} \approx 1$ to find solution $\Rightarrow m = 2^{n/2}$

Quantum algorithm Quantum algorithm only needs $\Omega(n)$ evaluations of corresponding U_f .

→ Quantum algorithms needs exponentially less queries.

Run the circuit in Fig. 4.5, where U_f is as for Deutsch's problem, $\Omega(n)$ times to find s .

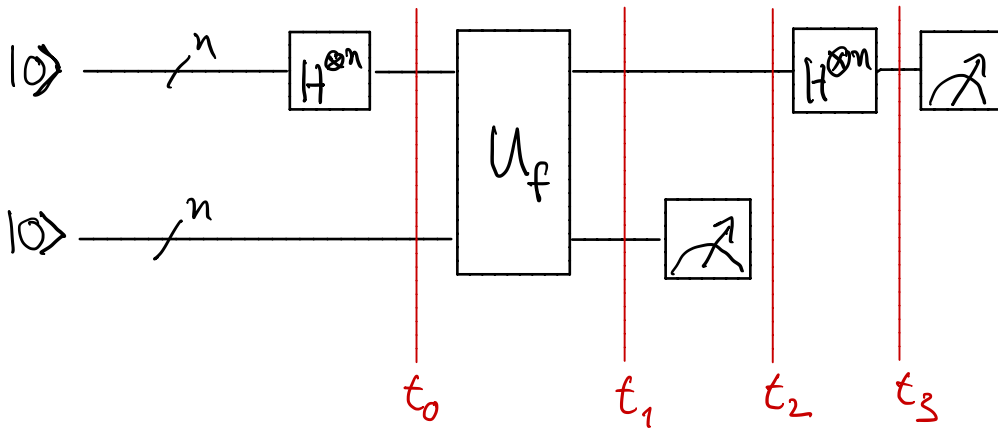


Figure 4.5: Quantum circuit to solve Simon's problem.

The states at the times t_0, t_1, t_2 and t_3 read

$$|\psi(t_0)\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right)^n \otimes |0\rangle^n = \frac{1}{2^{n/2}} \sum_{x \in \{0,1\}^n} |x\rangle \otimes |0\rangle^n \quad (4.57)$$

$$|\psi(t_1)\rangle = \frac{1}{2^{n/2}} \sum_{x \in \{0,1\}^n} |x\rangle \otimes |f(x)\rangle \quad (4.58)$$

For a measurement outcome $f(x)$ in the measurement of the second half of the qubits, the state is projected to

$$|\psi(t_2)\rangle = \frac{|x\rangle + |x \oplus s\rangle}{\sqrt{2}} \quad (\text{or } |x\rangle \text{ if } s = 0) \quad (4.59)$$

Applying the consecutive Hadamards leads to

$$\begin{aligned}
H^n |x\rangle &= \left(\frac{|0\rangle + (-1)^{x_0} |1\rangle}{\sqrt{2}} \right) \left(\frac{|0\rangle + (-1)^{x_1} |1\rangle}{\sqrt{2}} \right) \cdots \left(\frac{|0\rangle + (-1)^{x_{n-1}} |1\rangle}{\sqrt{2}} \right) \\
&= \frac{1}{2^{n/2}} \left(\sum_{y_0 \in \{0,1\}} (-1)^{x_0 y_0} |y_0\rangle \right) \left(\sum_{y_1 \in \{0,1\}} (-1)^{x_1 y_1} |y_1\rangle \right) \cdots \left(\sum_{y_{n-1} \in \{0,1\}} (-1)^{x_{n-1} y_{n-1}} |y_{n-1}\rangle \right) \\
&= \frac{1}{2^{n/2}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle \quad \text{where } x \cdot y = x_0 y_0 + x_1 y_1 + \cdots + x_{n-1} y_{n-1}
\end{aligned} \tag{4.60}$$

$$H^n |x \oplus s\rangle = \frac{1}{2^{n/2}} \sum_{y \in \{0,1\}^n} (-1)^{(x \oplus s) \cdot y} |y\rangle \tag{4.61}$$

where the second equation holds since the calculation in Eq. (4.61) applies for any x .

Running the calculation in Eq. (4.61) for $x = 00000 \dots 0$ also shows that Eq. (4.57) is correct.

Putting the above together we thus get for t_3 ,

$$\begin{aligned}
|\psi(t_3)\rangle &= \frac{1}{2^{(n+1)/2}} \sum_{y \in \{0,1\}^n} [(-1)^{x \cdot y} + (-1)^{(x \oplus s) \cdot y}] |y\rangle \quad \left(\text{or } \frac{1}{2^{n/2}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle \text{ if } s = 0 \right) \\
&= \frac{1}{2^{(n-1)/2}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} \frac{1 + (-1)^{s \cdot y}}{2} |y\rangle
\end{aligned} \tag{4.62}$$

$\Rightarrow$ outcome for measurement of first n qubits is a $y \in \{0,1\}^n$ that satisfies $s \cdot y = 0 \pmod 2$ since, only for $s \cdot y$ even, we have

$$\frac{1 + (-1)^{s \cdot y}}{2} \neq 0. \tag{4.63}$$

All possible outcomes y are equally likely

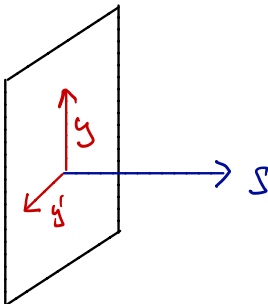
$$P(y) = \frac{1}{2^{n-1}} \tag{4.64}$$

$\Rightarrow$ circuit produces random y that is orthogonal to s .

$\rightarrow$ run circuit multiple times

$\Rightarrow$ each run of circuit produces a y that is orthogonal to s

$\Rightarrow$ after enough runs we have enough y s to determine the hyperplane that is orthogonal to s .



$\Rightarrow$ can determine s with classical post-processing $\rightarrow$ solve linear system of equations

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} s = 0 \pmod{2} \quad (4.65)$$

$\rightarrow$ probability $\bar{P}$ for drawing $n-1$ independent y s in $n-1$ measurements is $\geq 1/4$

$$\bar{P} = \left(1 - \frac{1}{2^{n-1}}\right) \left(1 - \frac{1}{2^{n-2}}\right) \cdots \left(1 - \frac{1}{2}\right) \geq \prod_{k=1}^{\infty} \left(1 - \frac{1}{2^k}\right) \approx 0.29 > \frac{1}{4}$$

$\Rightarrow$ need $\Omega(n)$ runs to get enough y s.

Simon's algorithm relies on cancellation of $+1$ and -1 in sums, see Eq. (4.63) .

Try to generalize to cancellation of arbitrary phases:

4.8 Quantum Fourier transform

Consider Fourier transform on $\mathbb{C}^N$.

For N -level quantum system, can choose basis

$$\{|0\rangle, |1\rangle, |2\rangle, \dots, |N-1\rangle\} \quad (4.66)$$

Consider N th roots of unity

$$\omega = e^{2\pi i/N} \quad (4.67)$$

$$\omega^N = 1 \quad (4.68)$$

$$\sum_{j=0}^{N-1} \omega^{kj} = N\delta_{k,0} \quad \text{for } k \text{ integer} \quad (4.69)$$

Proof $\rightarrow$ geometric series $\sum_{k=0}^{N-1} r^k = \frac{1-r^N}{1-r}$ for $r \neq 1$ and $\sum_{k=0}^{N-1} 1 = N$

Generalize Hadamard basis

$$|\pm\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle) \quad (4.70)$$

to $N \geq 1$ and more general phases $\Rightarrow$

$$|\tilde{y}\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} \omega^{xy} |x\rangle \quad (4.71)$$

These are orthonormal

$$\langle \tilde{y} | \tilde{z} \rangle = \frac{1}{N} \sum_{x=0}^{N-1} \omega^{-xy} \omega^{xz} = \frac{1}{N} \sum_{x=0}^{N-1} e^{-\frac{2\pi i}{N}(y-z)x} = \delta_{y,z} \quad (4.72)$$

and can make N distinct choices of y .

$\Rightarrow \{|\tilde{0}\rangle, |\tilde{1}\rangle, |\tilde{2}\rangle, \dots, |\tilde{N-1}\rangle\}$ form a basis.

$\Rightarrow$ There is a unitary transform from the $|y\rangle$ states to the $|\tilde{y}\rangle$ states

$$FT_N |y\rangle = |\tilde{y}\rangle \quad (4.73)$$

This is called "Quantum Fourier Transform"

$\Rightarrow FT_N^\dagger$ is the inverse of $FT_N \Rightarrow FT_N^\dagger FT_N = \mathbb{1}$

Can expand an arbitrary state in both bases

$$|\psi\rangle = \sum_{x=0}^{N-1} c_x |x\rangle = \sum_{y=0}^{N-1} a_y |\tilde{y}\rangle \quad (4.74)$$

$$\Rightarrow c_x = \sum_{y=0}^{N-1} \frac{a_y}{\sqrt{N}} \omega^{xy} \quad \text{and} \quad a_y = \sum_{x=0}^{N-1} \frac{c_x}{\sqrt{N}} \omega^{-xy} \quad (4.75)$$

Since the transformation (4.73) is unitary, it can be implemented on a quantum computer.

Classical algorithms How expensive are classical algorithms for computing Fourier transforms?

Naively: need to sum N terms for computing each a_y , see Eq. (4.75), hence need $\Omega(N^2)$ steps for all a_y .

There is an algorithm with only $\Omega(N \log N)$ steps called "Fast Fourier Transform" (FFT):

The trick is to decompose the Fourier transform into two parts (N odd):

$$\begin{aligned} a_k &= \sum_{j=0}^{N-1} c_j \omega^{-jk} \\ &= \underbrace{\sum_{j=0}^{\frac{1}{2}(N-1)} c_{2j} \omega^{-2jk}}_{\text{even points}} + \underbrace{\sum_{j=0}^{\frac{1}{2}(N-1)-1} c_{2j+1} \omega^{-(2j+1)k}}_{\text{odd points}} \\ &= \underbrace{\sum_{j=0}^{\frac{1}{2}(N-1)} c_{2j} (\omega^2)^{-jk}}_{\text{I}} + \omega^{-k} \underbrace{\sum_{j=0}^{\frac{1}{2}(N-1)-1} c_{2j+1} (\omega^2)^{-jk}}_{\text{II}} \end{aligned} \quad (4.76)$$

$\rightarrow$ each term, I and II, is a Fourier Transform of length $N/2$ and with $\omega' = \omega^2$ (frequencies double and periods have half the length).

$\rightarrow$ only $N/2$ values for k , but need N values for k (size of original problem)

$\rightarrow$ get remaining values, for $k = (N-1)/2 + 1, \dots, N-1$, from periodicity.

If we do the naive method, the reffort scales as

$$N^2 \quad (4.77)$$

If we divide into odd and even points, we need to compute 2 FTs of length $N/2$. $\rightarrow$ effort scales as

$$2 \left(\frac{N}{2} \right)^2 + N = \frac{N^2}{2} + N \quad (4.78)$$

Where the $+N$ term arises because both FTs need to be added up for each value of k . For the next iteration of dividing the problem into 4 parts, we get

$$2 \left[2 \left(\frac{N}{4} \right)^2 + \frac{N}{2} \right] + N = \frac{N^2}{4} + 2N \quad (4.79)$$

Can iterate the divisions until only individual terms are left $\rightarrow$ effort scales as

$$\frac{N^2}{2^m} + mN = \frac{N^2}{N} + N \log_2(N) = \Omega(N \log_2 N) \quad (4.80)$$

$\rightarrow$ Adding up the individual FTs becomes the dominant effort.

Quantum algorithm The quantum algorithm for a Quantum Fourier Transform works in

$$\Omega(\log^2 N) \quad (4.81)$$

steps, which is an exponential speed-up $\rightarrow \log_2 N$ is exponentially less than N .

There is a price to pay:

$\rightarrow$ output is a state $\rightarrow$ coefficients become amplitudes of basis states $\Rightarrow$ won't know all coefficients

$\rightarrow$ need to use result differently $\rightarrow$ discuss later.

Key reason for advantage: **can represent N states of Eq. (4.66) with $n = \log_2 N$ qubits**
 $\rightarrow N = 2^n$

$$|x\rangle = |x_{n-1}, x_{n-2}, \dots, x_0\rangle \quad \text{such that} \quad x = x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \dots + x_12 + x_0 \quad (4.82)$$

That is $x_{n-1}, x_{n-2}, \dots, x_0$ are the binomial representation of the number x .

Engineer Quantum Fourier Transform recursively using FFT formula (4.76):

$$FT_N = \begin{matrix} |y_{n-1}\rangle \\ |y_{n-2}\rangle \\ \vdots \\ |y_1\rangle \\ |y_0\rangle \end{matrix} \xrightarrow{FT_{N/2}} \begin{matrix} |x_0\rangle \\ |x_1\rangle \\ \vdots \\ |x_{n-2}\rangle \\ |x_{n-1}\rangle \end{matrix} \quad (4.83)$$

where

$$R_j = \begin{bmatrix} 1 & 0 \\ 0 & \exp(2\pi i \times 2^{-j}) \end{bmatrix} \quad (4.84)$$

are controlled phase rotations.

Remarks:

- Sequence $x_0, x_1, \dots, x_{n-1}$ can be flipped.
- Fourier transform on half the length has frequencies that double, i.e. $FT_N(\omega) \leftrightarrow FT_{N/2}(\omega^2)$, see formula (4.76)

Proof of equation (4.83)

- Consider case with $y_0 = 0$:

$\Rightarrow$ controlled R_j -gates do nothing

$\Rightarrow$

$$\begin{aligned}
 FT_N |y_{n-1}, y_{n-2}, \dots, y_1, 0\rangle &= (FT_{N/2} |y_{n-1}, y_{n-2}, \dots, y_1\rangle) \otimes (H |0\rangle) \\
 &= \sqrt{\frac{2}{N}} \sum_{x_0, x_1, \dots, x_{n-2}} (\omega^2)^{(y_1+2y_2+2^2y_3+\dots+2^{n-2}y_{n-1})(x_0+2x_1+2^2x_2+\dots+2^{n-2}x_{n-2})} |x_0, x_1, \dots, x_{n-2}\rangle \\
 &\quad \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \\
 &= \frac{1}{\sqrt{N}} \sum_{x_0, \dots, x_{n-2}} \omega^{(2y_1+2^2y_2+\dots+2^{n-1}y_{n-1})(x_0+2x_1+\dots+2^{n-2}x_{n-2})} (|x_0, \dots, x_{n-2}, 0\rangle + |x_0, \dots, x_{n-2}, 1\rangle) \\
 &= \frac{1}{\sqrt{N}} \sum_{x_0, \dots, x_{n-1}} \omega^{xy} |\vec{x}\rangle
 \end{aligned} \tag{4.85}$$

$$\tag{4.86}$$

where we have used that $y_0 = 0$ and that y is even for $y_0 = 0$. The latter implies for the contribution of $x_{n-1} = 1$, that

$$\omega^{y(x_0+2x_1+\dots+2^{n-2}x_{n-2}+2^{n-1})} = \omega^{y(x_0+2x_1+\dots+2^{n-2}x_{n-2})} \omega^{2^n \frac{y}{2}} = \omega^{y(x_0+2x_1+\dots+2^{n-2}x_{n-2})} \tag{4.87}$$

since

$$\omega^{(2^n)} = (e^{2\pi i/N})^N = e^{2\pi i} = 1 \tag{4.88}$$

- Consider case with $y_0 = 1$:

$\Rightarrow$ need additional pre-factor ω^x :

$$\omega^x = \omega^{x_0+2x_1+\dots+2^{n-1}x_{n-1}} = \underbrace{\left(\prod_{j=0}^{n-2} \omega^{2^j x_j} \right)}_{\prod_{j=0}^{n-2} R_{n-j}} \underbrace{\omega^{2^{n-1} x_{n-1}}}_H \tag{4.89}$$

where we have used

$$\omega^{2^j x_j} = (e^{2\pi i/N})^{2^j x_j} = (e^{2\pi i/2^{n-j}})^{x_j}, \tag{4.90}$$

which is the factor generated by the R_{n-j} controlled phase gate, and that

$$\omega^{2^{n-1} x_{n-1}} = (e^{i\pi})^{x_{n-1}} = (-1)^{x_{n-1}}, \tag{4.91}$$

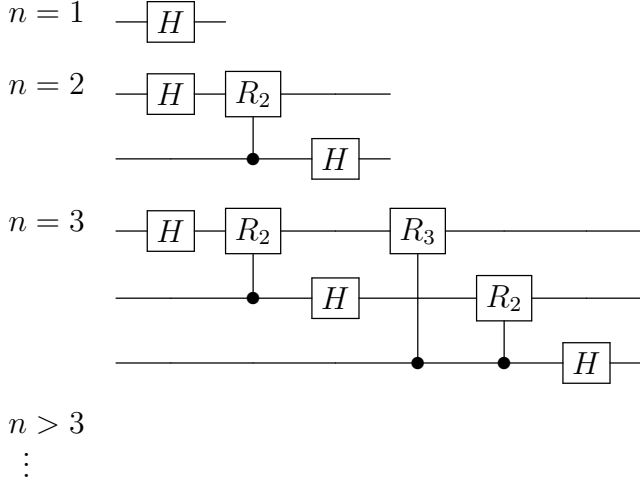
which is the factor generated by the Hadamard gate for $y_0 = 1$.

We can now count the number of necessary gates.

From Eq. (4.83) we get that

$$\begin{aligned} \underbrace{\text{Size}(N)}_{FT_N} &= \underbrace{\text{Size}(N/2)}_{FT_{N/2}} + \underbrace{n}_{R_j \text{ s and } H} = \text{Size}(N/4) + n - 1 + n \\ &= 1 + 2 + \cdots + n - 1 + n = \mathcal{O}(n^2) = \mathcal{O}(\log^2 N) \end{aligned} \quad (4.92)$$

For specific qubit numbers we get the circuits



Note also that

$$FT_N |0^n\rangle = H^{\otimes n} |0^n\rangle \quad (4.93)$$

4.9 Application: Phase Estimation

consider a unitary operator on m qubits U .

let $|u_\nu\rangle$ be an eigenvector of U ,

$$U |u_\nu\rangle = e^{i\theta_\nu} |u_\nu\rangle \quad (4.94)$$

Task find θ_ν .

Why does one want to do this?

- it's part of Shor's algorithm for factoring large numbers into primes
- can be used to solve coupled linear equations $\rightarrow$ appears in many numerical routines $\rightarrow$ useful for quantum machine learning applications $\rightarrow$ HHL algorithm, discuss later
- can be used to compute time evolution of quantum systems

Strategy For fixed number of qubits n , find y such that

$$\omega^y \approx e^{i\theta_\nu} \Rightarrow \theta_\nu \approx 2\pi 2^{-n} y \quad (4.95)$$

is the n bit approximation to θ_ν ($y = y_0 + y_1 2 + y_2 2^2 + \dots y_{n-1} 2^{n-1}$). Have assumed $\theta_\nu \leq 2\pi$.

$\rightarrow n$, the number of qubits, is our machine precision

$m \neq n$, at least not necessarily

If $N\theta_\nu/(2\pi)$ is integer ($N = 2^n$), can find y such that $\theta_\nu = 2\pi 2^{-n} y$

$\Rightarrow$ only for $N\theta_\nu/(2\pi)$ non-integer, use approximation

Suppose we can prepare

$$\frac{1}{2^{n/2}} \sum_{x=0}^{N-1} e^{i\theta_\nu x} |x\rangle \approx \frac{1}{2^{n/2}} \sum_{x=0}^{N-1} \omega^{xy} |x\rangle \quad (4.96)$$

$\Rightarrow$ apply $FT_N^\dagger$ to get $|y\rangle$ ($FT_N^\dagger |x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \omega^{-xy} |y\rangle$) and measure to find y

$$FT_N^\dagger \frac{1}{2^{n/2}} \sum_{x=0}^{N-1} e^{i\theta_\nu x} |x\rangle = \sum_y \underbrace{\left(\frac{1}{N} \sum_x \exp \left[\frac{2\pi i}{N} \left(\frac{N\theta_\nu}{2\pi} - y \right) x \right] \right)}_{f_{\theta_\nu}(y)} |y\rangle \quad (4.97)$$

Note that $N = 2^n$.

Hence for $N\theta_\nu/(2\pi)$ integer,

$$|f_{\theta_\nu}(y)|^2 = 1 \quad \text{for } y = \theta_\nu 2^n / (2\pi) \quad (4.98)$$

$\Rightarrow$ will for sure find $y = \theta_\nu 2^n / (2\pi)$ in measurement.

What does $f_{\theta_\nu}(y)$ look like if $N\theta_\nu/(2\pi)$ is not integer?

Using geometric series, $\sum_{x=0}^{N-1} r^x = (1 - r^N)/(1 - r)$ for $r = e^{i\frac{2\pi}{N}(\frac{N\theta_\nu}{2\pi} - y)} \neq 1$, one finds

$$|f_{\theta_\nu}(y)|^2 = \frac{1}{N^2} \frac{1 - \cos(N\theta_\nu)}{1 - \cos\left(\theta_\nu - \frac{2\pi y}{N}\right)} \quad (4.99)$$

which is strongly peaked at $y = N\theta_\nu/(2\pi)$.

$\rightarrow$ meaning of " $\approx$ " sign in equation (4.95).

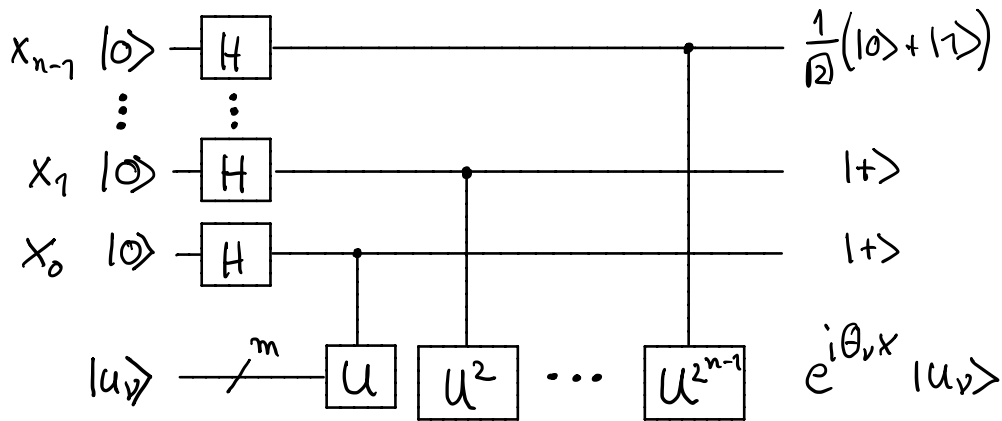
$\rightarrow$ expand $|f_{\theta_\nu}(y)|^2$ around $y_0 = N\theta_\nu/(2\pi)$,

$$|f_{\theta_\nu}(y)|^2 = \frac{1 - \cos(2\pi y_0)}{2\pi^2 (y - y_0)^2} + \frac{1 - \cos(2\pi y_0)}{6N^2} + \mathcal{O}([y - y_0]^2) \quad (4.100)$$

$\rightarrow$ only need to prepare superposition state

$$\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} e^{i\theta_\nu x} |x\rangle \quad (4.101)$$

→ can be done via following circuit



where, after the Hadamards we get an equal superposition of all states x , and the unitary U is defined in equation (4.94).

→ each controlled U generates $e^{i\theta_\nu}$ on $|u_\nu\rangle$ whenever $x_j = 1$.

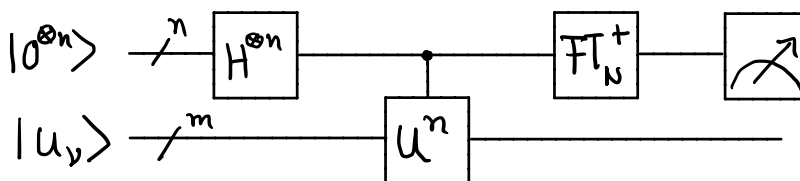
→ using

$$e^{i\theta_\nu x} = e^{i\theta_\nu(x_0 + 2x_1 + 2^2x_2 + \dots + 2^{n-1}x_{n-1})} = \prod_{j=0}^{n-1} e^{i\theta_\nu 2^j x_j} \quad (4.102)$$

we get the final state

$$\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle e^{i\theta_\nu x} |u_\nu\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} e^{i\theta_\nu x} |x\rangle |u_\nu\rangle \quad (4.103)$$

⇒ total phase estimation circuit is



Maybe don't have eigenvector $|u_\nu\rangle$ prepared.

Can we input a general state? Put in a state

$$|\psi\rangle = \sum_{\nu} c_{\nu} |u_{\nu}\rangle \quad (4.104)$$

instead of one $|u_\nu\rangle$.

Output measurements find n -bit approximation to θ_ν with probability $|c_\nu|^2$.

→ to see this, imagine measuring the state $|u_\nu\rangle$ at end of the circuit.

→ assumed that we can measure $|u_\nu\rangle$ (might not be state in computational basis, e.g. $|u_\nu\rangle \neq |10011\dots\rangle$).

→ without measurement can't tell which θ_ν is estimated → only estimate θ_ν with given probability

Quantum versus classical algorithm How do the speeds of algorithms compare?

Quantum algorithm needs n gates (n is number of qubits)

Classical algorithm works with matrix representation of U , which has $2^m \times 2^m$ elements

⇒ quantum algorithm can be exponentially faster

Now let's look at applications of phase estimation.

4.10 Shor's Factoring Algorithm

Each integer N has a unique decomposition as a product of prime numbers $p_1, p_2, \dots$,

$$N = \prod_j p_j \quad (4.105)$$

Finding this decomposition classically is hard → becomes harder quickly as integer number increases

Hardest instances are $N = p_1 p_2$, where $p_1 \sim p_2$ are both as large as possible.

→ Prime factor decomposition is basis for RSA encryption protocol which is

- used for credit cards
- used for email, chats, VPN, ...
- classical secure because the computational effort to break it is exceedingly large → can't factor numbers with more than 1000 bits.

RSA encryption

- **encryption**
 - choose two large prime numbers p and q
 - compute $N = pq$ and $\varphi = (p-1)(q-1)$
 - choose e such that $1 < e < \varphi - 1$ and $\gcd(e, \varphi) = 1$. Here $\gcd$ denotes the greatest common divisor.
 - compute d such that $ed = 1 + r\varphi$ for some integer r (is the same as $ed = 1 \pmod{\varphi}$), there is an efficient algorithm for this.

- common key d shared by sender and receiver (must be secret, i.e. only known to sender and receiver)
- sender encrypts message x ($x < N$) in $y = x^e \bmod N$ (same as $y = x^e + \tilde{r}N$ with $\tilde{r}$ integer)
- sender sends e and N to receiver (can be public)

- **decryption**

- compute

$$y^d = (x^e + \tilde{r}N)^d = x^{ed} \bmod N = x^{1+r\varphi} = x \bmod N = x \quad (4.106)$$

- without knowing d , respectively p and q , x is very hard to compute from y .
- would be easy if one could efficiently find p and q from N

Shor's algorithm finds prime factor decompositions $N = pq$ exponentially faster than classical algorithms

⇒ Implementation of Shor's algorithm would make RSA encryption insecure.

Concept of Shor's algorithm

- reduce factoring to **period finding** of function

$$f(x) = a^x \bmod N \quad (4.107)$$

which is periodic for $a > 1$ and $x > 0$.

- only period finding part of algorithm is done quantum mechanically → needs classical pre- and post-processing (number theory)
- classical part based on existence of an efficient algorithm for finding $\gcd(x, y)$ for two integers x and y .
- main quantum part is to find the period of a specific function
- phase estimation is used to find that period

Algorithmic steps for finding a factor of a number N

1. check that N is not prime, not even, and not of form a^b , where a and b integer.
2. choose random integer $a \in \{2, 3, \dots, N-1\}$
→ if $\gcd(a, N) \neq 1$ ($\gcd$ =greatest common divisor) finished ($p = \gcd(a, N)$, $q = N/p$), else continue
3. compute smallest integer r such that $a^r \equiv 1 \bmod N$, meaning that division by N leaves same rest for both sides, or here that $a^r = kN + 1$ for an integer k .
(exists because $\gcd(a, N) = 1$).
 r is called *order or period of $a \bmod N$* .
This is the **quantum step**
4. if either r is odd (⇒ $a^{r/2}$ is not integer) or $a^{r/2} + 1$ or $a^{r/2} - 1$ are multiples of N → won't find non-trivial factor
 $a^{r/2} - 1 \neq k'N$ since r is the smallest integer to fulfill (3).
→ go to (2)
→ can be shown that this happens in less than half the cases

5. since $a^r \bmod N = 1 \bmod N \Rightarrow a^r - 1 = kN$ for some integer $k > 0$ and $a^{r/2} - 1 \neq k'N$ and $a^{r/2} + 1 \neq k''N$ for k' and k'' integers
 $\Rightarrow \gcd(a^{r/2} - 1, N)$ is a non-trivial factor of N .

Example: $N = 21$ and $a^2 - 1 = 3 \times 21$ for $a = 8$. Hence 21 is a divisor of $a^2 - 1$ but neither of $a + 1$ nor of $a - 1$. $\Rightarrow \gcd(a + 1, N) = 3$ and $\gcd(a - 1, N) = 7$ are divisors of 21.

Here all steps can be done efficiently on classical computers except for step 3

$\Rightarrow$ do step 3 on a quantum computer

$\rightarrow$ use phase estimation for doing step 3

Step 3 is called "order finding".

4.10.1 Order or Period Finding

Let N be a large composite number

choose $a < N$ at random

$\rightarrow$ find the period of

$$f(x) = a^x \bmod N \quad (4.108)$$

$\rightarrow f$ is periodic because of the "mod N "-part

$\rightarrow r$ is the period of f

to implement "order finding", want quantum implementation of f as in equation (4.108).

$\rightarrow$ implement the controlled multiplication gate

$$|x\rangle |b \bmod N\rangle \rightarrow |x\rangle |b a^x \bmod N\rangle \quad (4.109)$$

as U^x with

$$U |x\rangle |b \bmod N\rangle = |x\rangle |b a \bmod N\rangle \quad (4.110)$$

for $b = 1$

$\rightarrow |x\rangle$ is m -qubit control state, $x = \{0, 1, \dots, M-1\}$, $M = 2^m \rightarrow$ precision of phase estimation

$\rightarrow U$ can be e.g. implemented as outlined in section 4.5.

$\rightarrow$ what are eigenstates of U ?

$$U \left[\frac{1}{\sqrt{r}} \sum_{l=0}^{r-1} e^{-2\pi i s \frac{l}{r}} |a^l \bmod N\rangle \right] = \frac{1}{\sqrt{r}} \sum_{l=0}^{r-1} e^{-2\pi i s \frac{l}{r}} |a^{l+1} \bmod N\rangle \quad (4.111)$$

$$= e^{2\pi i s \frac{s}{r}} \frac{1}{\sqrt{r}} \sum_{k=1}^r e^{-2\pi i s \frac{k}{r}} |a^k \bmod N\rangle \quad (4.112)$$

$$= e^{2\pi i s \frac{s}{r}} \left[\frac{1}{\sqrt{r}} \sum_{l=0}^{r-1} e^{-2\pi i s \frac{l}{r}} |a^l \bmod N\rangle \right] \quad (4.113)$$

where we have set $k = l + 1$ and used that the $k = r$ term in the sum is identical to the $k = 0$ term.

→ can use phase estimation to find s/r and then extract r since both, s and r , are integer.

→ there are 2 problems to overcome

problem 1: can't prepare eigenstates of U

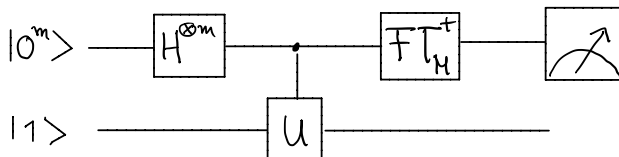
→ prepare superpositions of such states

→ do phase estimation on

$$|1\rangle = |a^0\rangle = \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} \left[\frac{1}{\sqrt{r}} \sum_{l=0}^{r-1} e^{-2\pi i s \frac{l}{r}} |a^l \bmod N\rangle \right] \quad (4.114)$$

⇒ phase estimation will yield s/r for random s → doesn't matter, can extract r for any s

→ circuit for doing this is:



problem 2: phase estimation only provides y/M , an m -bit approximation to s/r

→ for sufficient qubit number $y \gg s$ and $M \gg r$

s/r is rational number

→ try to approximate y/M with continued fractions ⇒ one approximation to y/M will be identical to s/r .

4.11 Application: Solving systems of linear equations

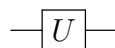
Linear systems of equations appear in many numerical algorithms.

There is a quantum algorithm that can solve such systems of equations exponentially faster than classical algorithms.

Let us first consider a warm-up problem.

4.11.1 Applying arbitrary powers of gates

Consider a unitary U implemented by some circuit.



What is a circuit to implement U^2 ?



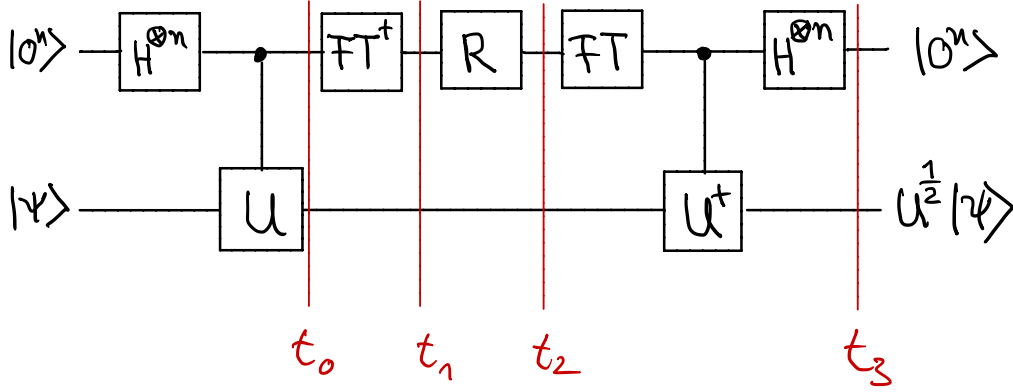
That's easy.

What is a circuit to implement $U^{\frac{1}{2}}$

→ not so easy.

→ can use phase estimation if we have access to controlled U .

→ consider the following circuit



where

$$|\psi\rangle = \sum_j \beta_j |u_j\rangle, \quad \text{is an arbitrary state, and} \quad U |u_j\rangle = e^{i\theta_j} |u_j\rangle \quad (4.115)$$

→ want to implement

$$U^{\frac{1}{2}} |u_j\rangle = e^{i\theta_j/2} |u_j\rangle \quad (4.116)$$

At time t_1 , the state reads [see Eq. (4.97)],

$$\sum_j \beta_j |2^n \theta_j / (2\pi)\rangle |u_j\rangle \quad (4.117)$$

where (as result of phase estimation),

$$|2^n \theta_j / (2\pi)\rangle = |y\rangle \quad \text{for} \quad y \approx \frac{2^n \theta_j}{2\pi} \quad (4.118)$$

to very good accuracy, (amplitudes in components for different θ are negligible).

→ want

$$R |2^n \theta_j / (2\pi)\rangle = e^{i\theta_j/2} |2^n \theta_j / (2\pi)\rangle \quad (4.119)$$

hence choose

$$R |y\rangle = e^{i\pi y / 2^n} |y\rangle \quad (4.120)$$

which can be implemented qubit by qubit ($y = y_0 + y_1 2 + y_2 2^2 + \dots y_{n-1} 2^{n-1}$).

⇒ at time t_2 ,

$$\sum_j \beta_j e^{i\theta_j/2} |2^n \theta_j / (2\pi)\rangle |u_j\rangle \quad (4.121)$$

where the register states $|2^n \theta_j / (2\pi)\rangle$ depend on j

⇒ register is correlated with state $|\psi\rangle$

⇒ need to undo this

reminder of circuit puts register back into $|0\rangle$.

$\Rightarrow$ state at time t_3 ,

$$\sum_j \beta_j e^{i\theta_j/2} |0\rangle |u_j\rangle = |0\rangle U^{\frac{1}{2}} |\psi\rangle \quad (4.122)$$

This follows a general strategy for quantum algorithms

1. create superposition state
2. act with desired unitary in superposition
3. turn superposition back into product state

4.11.2 Solving systems of linear equations: HHL algorithm (Harrow, Hassidim, Lloyd)

System of linear equations: Given a $N \times N$ matrix A and vector $\vec{y}$, find a vector $\vec{x}$ such that,

$$A\vec{x} = \vec{y} \quad (4.123)$$

There are polynomial time classical algorithms, e.g. Gaussian elimination etc.

Coupled linear equations appear in many optimization problems and are therefore very important.

- Assume a quantity f to be optimized depends on parameters $x_1, x_2, \dots, x_n$.
- in each iteration change these parameters a little bit: $\delta x_j = x'_j - x_j$
- the change in f reads

$$\delta f \approx f + \sum_j \frac{\partial f}{\partial x_j} \delta x_j \quad (4.124)$$

- need to solve a system of linear equations to calculate parameter updates δx_j

Quantum version We are given an operator A and a state $|b\rangle = \sum_j b_j |j\rangle$.

amplitudes b_j of state $|b\rangle$ are components of classical vector $\vec{b} = (b_1, b_2, \dots, b_N)^T$.

The goal is to generate a state

$$|x\rangle \propto A^{-1} |b\rangle \quad (4.125)$$

$\Rightarrow$ different quality of answer compare to classical problem

- can't directly access components x_j
- can't copy state $|x\rangle$

Make some assumptions to make life easier (generalizations are easy)

- assume $A^\dagger = A$

$\Rightarrow A|u_j\rangle = \lambda_j |u_j\rangle$ where λ_j is real

- assume eigenvalues are non-negative, i.e. $\kappa^{-1} \leq \lambda_j \leq 1$
 $\rightarrow \kappa$ is ratio of largest to smallest eigenvalue (condition number)
- A is s -sparse, i.e. only s non-zero elements per row.

runtimes of algorithms for

classical $\mathcal{O}(Ns\sqrt{\kappa})$

→ reasonable: takes at least N steps to read in vector $\vec{b}$.

quantum $\mathcal{O}(\log N)$

→ exponentially less than N (roughest classical lower bound)

→ exponentially less than reading in vector $\vec{b}$

Strategy apply phase estimation to

$$U = \exp(2\pi i \kappa A/N) \quad (4.126)$$

and use similar strategy as for $U^{\frac{1}{2}}$ but now for U^{-1}

→ unitary $U = \exp(2\pi i \kappa A/N)$ can be efficiently constructed if A is sparse.

⇒ want to multiply $|2^n \lambda_j / (2\pi)\rangle$ by $1/\lambda_j$.

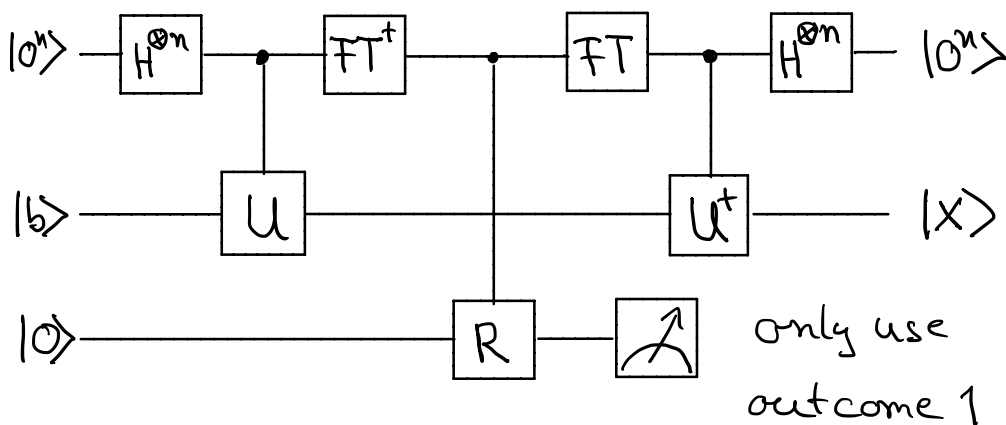
→ can do this with controlled rotation of form

$$|y\rangle |0\rangle \rightarrow |y\rangle \left(\sqrt{1 - \frac{y^2}{2}} |0\rangle + \frac{1}{y} |1\rangle \right) \quad (4.127)$$

followed by a measurement.

→ only use cases where measured $|1\rangle$ → may need to repeat.

⇒ Quantum algorithm has the circuit



consider expansion of input state $|b\rangle$ into eigenstates of U ,

$$|b\rangle = \sum_j \beta_j |u_j\rangle \quad (4.128)$$

state after phase estimation

$$|\psi\rangle = \sum_j \beta_j |\kappa \lambda_j\rangle |u_j\rangle |0\rangle \quad (4.129)$$

→ need to choose $2^n > \kappa$ so that register is large enough to capture all eigenvalues, i.e. $1 < \kappa \lambda_j < 2^n$

state after controlled rotations

$$|\psi\rangle \approx \sum_j \beta_j |\kappa\lambda_j\rangle |u_j\rangle \left(\sqrt{1 - \frac{1}{\kappa^2\lambda_j^2}} |0\rangle + \frac{1}{\kappa\lambda_j} |1\rangle \right) \quad (4.130)$$

→ probability to measure 1:

$$P(1) = \sum_j \frac{|\beta_j|^2}{\kappa^2\lambda_j^2} > \frac{1}{\kappa^2} \quad (4.131)$$

→ need to repeat $O(\kappa^2)$ times

state after measurement

$$|\psi\rangle \approx \frac{1}{\sqrt{P(1)}} \sum_j \frac{\beta_j}{\kappa\lambda_j} |\kappa\lambda_j\rangle |u_j\rangle \quad (4.132)$$

undo phase estimation circuit

$$|\psi\rangle = \frac{1}{\sqrt{P(1)}} \sum_j \frac{\beta_j}{\kappa\lambda_j} |0\rangle |u_j\rangle = |0\rangle \frac{1}{\sqrt{\sum_l \frac{|\beta_l|^2}{\lambda_l^2}}} \sum_j \frac{\beta_j}{\lambda_j} |u_j\rangle = |0\rangle \frac{A^{-1}|b\rangle}{\|A^{-1}|b\rangle\|} \quad (4.133)$$

→ have implemented $A^{-1}|b\rangle$

All algorithms discussed so far need perfect gates.

→ not available in existing devices (will probably never be)

→ how to address this?

Chapter 5

Quantum Error Correction

All the quantum algorithms studied so far need perfectly working gates.

Assume each gate has an error with 1% probability $\rightarrow$ success probability 99%

$\rightarrow$ success probability for algorithm with 1000 gates

$$0.99^{1000} \approx 5 \times 10^{-5} \quad (5.1)$$

$\Rightarrow$ to make large quantum algorithms work we need to correct errors.

$\rightarrow$ error correction codes

To get an idea how this could work, first look at simple classical error correction code

5.1 Repetition Code

5.1.1 Classical repetition code

Classically information is stored in bits, i.e. 0 or 1

to allow for error correction, store one logical bit in three physical bits

$$0_L = 000 \quad \text{and} \quad 1_L = 111 \quad (5.2)$$

The logical bits are also called *code words*.

$\rightarrow$ now the effect of a bit flip error is for example

$$000 \rightarrow 010 \quad \text{or} \quad 111 \rightarrow 110 \quad \text{etc} \quad (5.3)$$

$\rightarrow$ now the signal that an error has occurred is that all three physical bits are no longer the same.

$\rightarrow$ can compute the majority and correct the minority bit,

$$010 \rightarrow 000, \quad 110 \rightarrow 111, \quad \text{etc} \quad (5.4)$$

$[n, k, d]$ **notation** An error correction code is characterized by three parameters,

n the number of physical bits

k the number of encoded or logical bits

d the code distance, i.e. the fewest number of bit flips needed to transform any code-words into another.

→ distance d code can detect errors up to $d - 1$ bit flips and correct errors up to $(d - 1)/2$ bit flips .

logical operations on the code

$$000 \leftrightarrow 111 \quad (5.5)$$

5.1.2 Quantum 3 qubit repetition code

Logical bits in a quantum version we use the logical qubit states (called *codeword basis states*),

$$|0\rangle_L = |0, 0, 0\rangle \quad \text{and} \quad |1\rangle_L = |1, 1, 1\rangle \quad (5.6)$$

Hence a general logical state reads

$$|\psi\rangle_L = \alpha |0\rangle_L + \beta |1\rangle_L = \alpha |0, 0, 0\rangle + \beta |1, 1, 1\rangle \quad (5.7)$$

Errors quantum mechanically there are not only bit flip errors but two types of errors,

- quantum bit flip error

$$|0\rangle \rightarrow |1\rangle \quad \text{or} \quad |1\rangle \rightarrow |0\rangle \quad (5.8)$$

a bit flip error can be written as the action of a Pauli X gate

$$X |0\rangle = |1\rangle \quad \text{and} \quad X |1\rangle = |0\rangle \quad (5.9)$$

- phase flip error

$$|0\rangle \rightarrow |0\rangle \quad \text{but} \quad |1\rangle \rightarrow -|1\rangle \quad (5.10)$$

a bit flip error can be written as the action of a Pauli Z gate

$$Z |0\rangle = |0\rangle \quad \text{and} \quad Z |1\rangle = -|1\rangle \quad (5.11)$$

any single qubit error can be written as a combination of bit and phase flip errors because any single qubit transformation can be written in terms of Pauli operators and

$$Y = iXZ \quad (5.12)$$

→ a bit flip error on the codeword basis states reads, e.g.,

$$|\psi\rangle_L \rightarrow X_2 |\psi\rangle_L = \alpha |0, 1, 0\rangle + \beta |1, 0, 1\rangle \quad (5.13)$$

→ a phase flip error on the codeword basis states reads, e.g.,

$$|\psi\rangle_L \rightarrow Z_2 |\psi\rangle_L = \alpha |0, 0, 0\rangle - \beta |1, 1, 1\rangle \quad (5.14)$$

Error detection → don't want to measure individual qubits → would collapse whole state, e.g.

$$\alpha |0, 1, 0\rangle + \beta |1, 0, 1\rangle \rightarrow |0, 1, 0\rangle \quad (5.15)$$

→ destroys encoded bit

→ measure symmetries, here parity.

- 00 or 11 → even parity
- 01 or 10 → odd parity

for any valid logical state

$$|\psi\rangle_L = \alpha |0\rangle_L + \beta |1\rangle_L = \alpha |0, 0, 0\rangle + \beta |1, 1, 1\rangle \quad (5.16)$$

→ parity of any pair of qubits even $\Leftrightarrow$ no error

$\Rightarrow$ quantum 3 qubit repetition code cannot detect phase flip errors

circuit to measure parity of two qubits



measuring parity is equivalent to measuring $Z \otimes Z$

$$Z \otimes Z |00\rangle = +1 |00\rangle \quad (5.18)$$

$$Z \otimes Z |01\rangle = -1 |01\rangle \quad (5.19)$$

$$Z \otimes Z |10\rangle = -1 |10\rangle \quad (5.20)$$

$$Z \otimes Z |11\rangle = +1 |11\rangle \quad (5.21)$$

→ measuring all parities, $Z_1 Z_2$, $Z_2 Z_3$ and $Z_1 Z_3$, one can tell which physical qubit was flipped:

error	$Z_1 Z_2$	$Z_1 Z_3$	$Z_2 Z_3$
X_1	-1	-1	+1
X_2	-1	+1	-1
X_3	+1	-1	-1

Error correction errors can be corrected by applying the inverses of the error operators

→ for Pauli operators the inverse is the operator itself

→ bit flip on qubit 2 is corrected by applying X_2 .

Logical operations since each logical qubit is encoded in three physical qubits, logical operations (i.e. gates) are not single (physical) qubit gates.

e.g. a logical X -gate

$$X \otimes X \otimes X |1\rangle_L = |0\rangle_L \quad \text{and} \quad X \otimes X \otimes X |0\rangle_L = |1\rangle_L \quad (5.22)$$

$\Rightarrow$

$$X_L = X \otimes X \otimes X \quad (5.23)$$

is a logical X operator

e.g. a logical Z -gate

$$\mathbb{1} \otimes \mathbb{1} \otimes Z |1\rangle_L = -|1\rangle_L \quad \text{or} \quad \mathbb{1} \otimes Z \otimes \mathbb{1} |1\rangle_L = -|1\rangle_L \quad \text{or} \quad \dots \quad (5.24)$$

$\Rightarrow$ there are many possible logical Z operators $\rightarrow$ stabilizer formalism, see section 5.2.

$\Rightarrow$ quantum 3 qubit repetition code cannot detect phase flip errors

Distance of a quantum code The distance of a code is the minimal weight of any logical operator on the code except for the identity. The weight is the number of non-identity single qubit operators in a logical operator.

$\rightarrow$ minimum weight of X_L for 3 qubit repetition code is $d = 3 \Rightarrow$ code can detect up to $d - 1 = 2$ X errors and can correct $(d - 1)/2 = 1$ X error

$\rightarrow$ minimum weight of Z_L for 3 qubit repetition code is 1 $\Rightarrow$ code cannot detect or correct Z errors

There are however quantum error correction codes that can correct also phase flip errors

5.1.3 General types of errors

Why is it enough to consider Pauli errors only?

First note that any single qubit unitary can be written as a linear combination of Pauli matrices and the identity matrix

$$U = c_0 \mathbb{1} + c_x X + c_y Y + c_z Z \quad (5.25)$$

i.e. the Pauli matrices and the identity matrix form a basis for single qubit unitaries

$\rightarrow$ e.g. $\sigma^- = (X - iY)/2$

$\rightarrow$ if c_0, c_x, c_y and c_z are real, then U is also Hermitian.

Moreover, if a error correcting code can correct Pauli errors, like X and Z , it can also correct $Y = iXZ$ and any linear combination of Pauli errors.

$\rightarrow$ consider e.g. the error

$$E = c_x X + c_z Z \quad (5.26)$$

usually one uses ancilla qubits for the error detection Eq. (5.17)

assume we use one ancilla qubit for detecting X errors and another for detecting Z errors

$\Rightarrow$ we have the state

$$|\psi\rangle_L |00\rangle_{\text{anc}} = (\alpha |0\rangle_L + \beta |1\rangle_L) |00\rangle_{\text{anc}} \quad (5.27)$$

the action of the error E now is

$$E |\psi\rangle_L |0\rangle_{\text{anc}} = E(\alpha |0\rangle_L + \beta |1\rangle_L) |00\rangle_{\text{anc}} \quad (5.28)$$

$$= c_x X(\alpha |0\rangle_L + \beta |1\rangle_L) |10\rangle_{\text{anc}} + c_z Z(\alpha |0\rangle_L + \beta |1\rangle_L) |01\rangle_{\text{anc}} \quad (5.29)$$

where the first ancilla detects the X error ($|10\rangle_{\text{anc}}$) and the second the Z error ($|01\rangle_{\text{anc}}$)

→ will detect X error with probability $|c_x|^2$ and Z error with probability $|c_z|^2$

→ in any case can correct the error

- if X error, apply X
- if Z error, apply Z

⇒ can e.g. also apply arbitrary phase errors

$$e^{-i\varphi Z} = \cos(\varphi)\mathbb{1} - i\sin(\varphi)Z \quad (5.30)$$

⇒ ability to correct for Pauli errors is sufficient to correct for any (reasonable) error

→ need formalism to detect Pauli errors X and Z .

5.2 Stabilizer Formalism

The stabilizer formalism specifies a set of operators P_k , with $k = 1, 2, \dots$ such that all logical states $|\psi_j\rangle_L$ are eigenstates with eigenvalue 1, i.e.

$$P_k |\psi_j\rangle_L = |\psi_j\rangle_L \quad \text{for all } j \text{ and } k \quad (5.31)$$

The set of all P_k is called stabilizer and its elements are products of Pauli operators or identities, e.g.

$$P_k = X \otimes X \otimes Y \otimes Z \otimes \mathbb{1} \otimes \dots \quad (5.32)$$

→ for the 3-qubit quantum repetition code, the stabilizer is

$$Z \otimes Z \otimes \mathbb{1}; \quad \mathbb{1} \otimes Z \otimes Z; \quad Z \otimes \mathbb{1} \otimes Z; \quad \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1}; \quad (5.33)$$

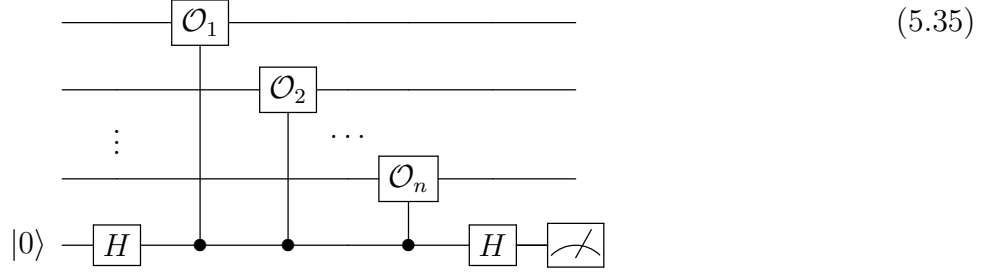
- stabilizer leaves logical states invariant
- stabilizer elements commute
- stabilizer elements form a group (generated by a set of generators S_j) under multiplication
- stabilizer elements are Hermitian operators → can be measured
→ result -1 implies error has occurred

⇒ measurements of stabilizers can be used to detect errors

Measurement of a stabilizer element a n qubit stabilizer element

$$P_k = \mathcal{O}_n \otimes \mathcal{O}_{n-1} \otimes \cdots \otimes \mathcal{O}_1 \quad (5.34)$$

can be measured via the circuit



Two Pauli operators either commute, if they are identical or act on different lattice sites, or anti-commute, if they act on the same lattice site but are non-identical,

$$\sigma^a \sigma^b + \sigma^b \sigma^a = 2\delta_{a,b} \mathbb{1} \quad (5.36)$$

$\Rightarrow$ Pauli error can be detected by stabilizer element that anti-commutes with it

$$P_k \sigma + \sigma P_k = 0 \quad \Rightarrow \quad P_k \sigma |\psi\rangle = (-1) \sigma P_k |\psi\rangle = (-1) \sigma |\psi\rangle \quad (5.37)$$

$\Rightarrow$ the state $\sigma |\psi\rangle$ has eigenvalue -1 for $P_k \Rightarrow$ error

$\Rightarrow$ no need to know or determine the logical states (like e.g. $|0\rangle_L$ and $|1\rangle_L$).

$[n, k, d]$ **stabilizer code** is an error correcting code with $m = n - k$ stabilizer generators S_j , which are mutually commuting products of Pauli operators.

all stabilizer elements can be generated from the stabilizer generators, i.e. for each P_k ,

$$P_k = \prod_j S_j \quad (5.38)$$

2^n states in Hilbert space.

for each stabilizer generator only consider states with eigenvalue +1 $\Rightarrow$ each divides number of states by 2

$\Rightarrow 2^n / 2^m$ possible states $\Rightarrow k = n - m$ logical qubits.

5.3 Surface Codes

5.3.1 Toric Code (Kitaev 2006)

Consider 2d lattice of qubits on surface of a torus, see Fig. 5.1.

There are two types of stabilizer operators associated with vertices and plaquettes of the lattice, see figure 5.2. For vertex j and plaquette k , they read

$$V_j = \prod_{n \text{ next to } j} X_n \quad \text{and} \quad P_k = \prod_{n \text{ around } k} Z_n \quad (5.39)$$

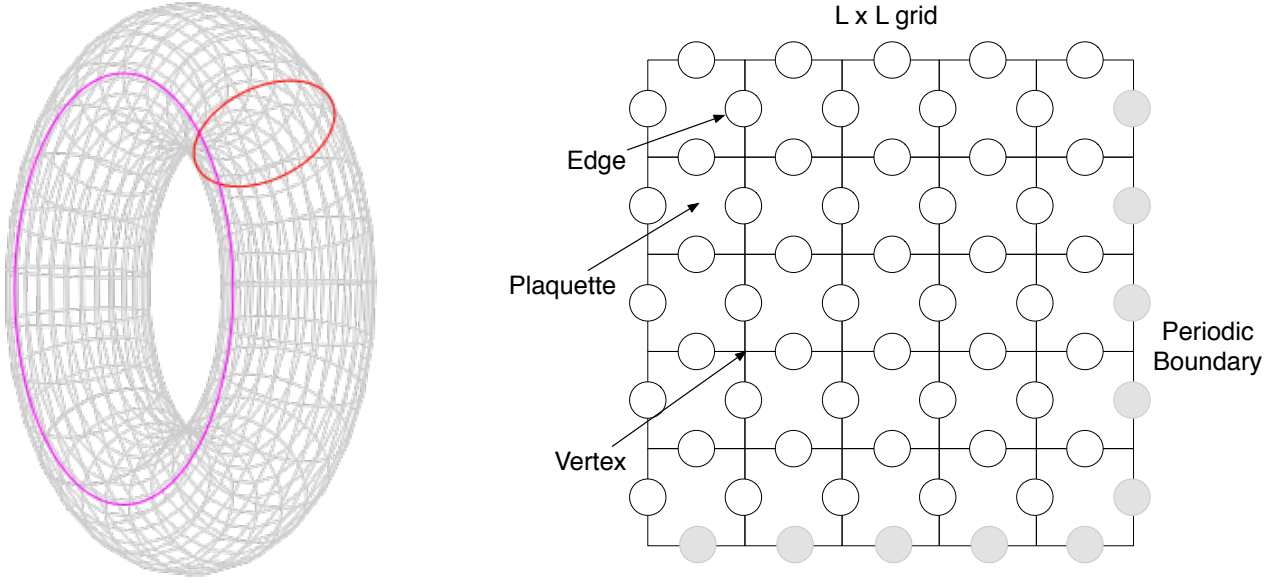


Figure 5.1: **Left** A torus is a 2d surface that has periodic boundary conditions in both directions. There are two lines that cannot be contracted to a point by deformations while staying within the surface. **Right** Lattice grid of qubits on the surface of the torus for the toric code.

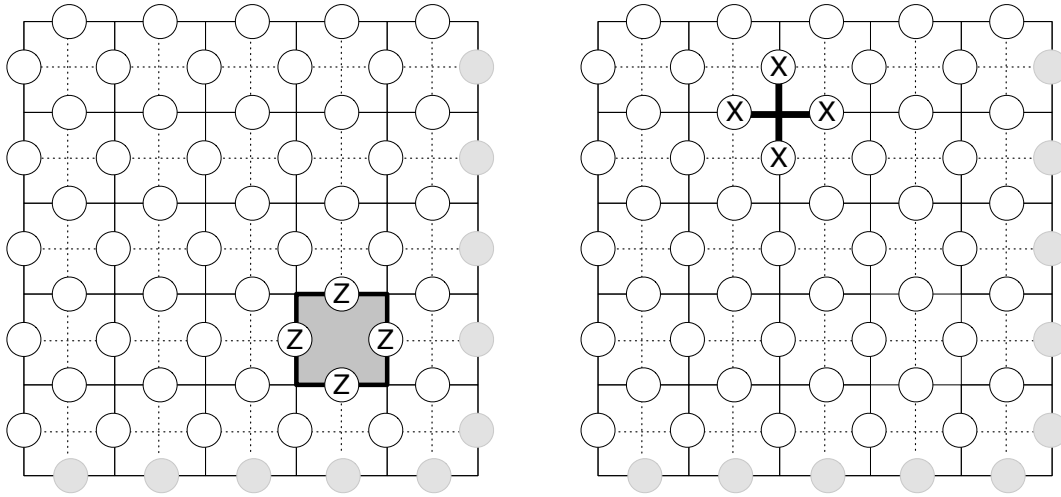


Figure 5.2: Examples of stabilizer generators in the toric code.

→ all stabilizer elements commute, i.e. for all j and l ,

$$[V_j, V_l] = 0, \quad [P_j, P_l] = 0 \quad \text{and} \quad [P_j, V_l] = 0 \quad (5.40)$$

the first two equations hold trivially while the last holds because a vertex and a plaquette operator either have zero or two Pauli operators in common and

$$[X_m X_n, Z_m Z_n] = 0 \quad \text{for all } n, m \quad (5.41)$$

since $X_n Z_n = -iY_n$ and thus $Z_n X_n = iY_n$ and

$$[X_m X_n, Z_m Z_n] = X_m X_n Z_m Z_n - Z_m Z_n X_m X_n = (-1)Y_m Y_n - (-1)Y_m Y_n = 0 \quad (5.42)$$

→ any product of two adjacent stabilizer generators of same type has one common Pauli operator.

→ since $X^2 = Y^2 = Z^2 = \mathbb{1}$, multiplication has the effect shown in Fig. 5.3.

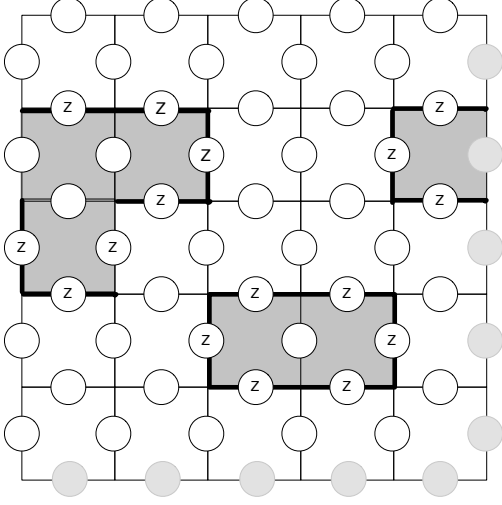


Figure 5.3: Effect of multiplication of adjacent stabilizer elements. The same holds for vertex operators.

→ multiplication of all plaquette operators $\Rightarrow$ no boundary left, same for vertex operators $\Rightarrow$

$$\prod_j V_j = \prod_k P_k = \mathbb{1} \quad (5.43)$$

$\Rightarrow$ for $L \times L$ lattice $L^2 - 1$ independent plaquette and vertex operators $\Rightarrow 2(L^2 - 1)$ stabilizer elements

→ $2L^2$ qubits (say left and bottom one for each plaquette) and $2(L^2 - 1)$ stabilizer elements

→ each additional stabilizer generator S_j divides dimension of subspace of joint eigenstates with eigenvalues ± 1 by 2

Proof:

- for independent stabilizer generators S_j and S_l that are made of Pauli operators, we have $\text{Tr}(S_j) = 0$ and $\text{Tr}(S_j S_l) = 0$, and similarly for any product of independent S_j s
- hence each S_j has half $+1$ eigenvalues and half -1 eigenvalues (both occur in equal numbers).
- the projector on the $+1$ eigenvalue subspace of S_j is $P_j = (\mathbb{1} + S_j)/2$
- hence $\text{Tr}(P_j S_l) = [\text{Tr}(S_l) - \text{Tr}(S_j S_l)]/2 = 0$ meaning that S_l restricted to the $+1$ eigenvalue subspace of S_j also has half $+1$ eigenvalues and half -1 eigenvalues
- iterating over all stabilizer generators yields the statement

→ count number of states, where each stabilizer selects half the states,

$$\frac{2^{2L^2}}{2^{2(L^2-1)}} = 2^2 \quad (5.44)$$

⇒ $2L^2 - 2(L^2 - 1) = 2$ logical qubits

→ writing down logical states $|\psi\rangle_L$ very cumbersome (highly entangled)

→ only need logical operators

Logical operators Need $X_{L,1}, X_{L,2}, Z_{L,1}$ and $Z_{L,2}$ for the logical qubits.

seek to write logical operators as Pauli products, see Eq. (5.32), (is a convention),

$$O_k = X \otimes X \otimes Y \otimes Z \otimes \mathbb{1} \otimes \dots \quad (5.45)$$

one can check that any two Pauli operators either commute, if they are identical or act on different lattice sites, or anti-commute, if they act on the same lattice site but are non-identical, see (5.36)

a logical operator O_k must commute with all stabilizers (needs to leave code subspace invariant), but not be a stabilizer element (must have several eigenvalues for code subspace).

→ consider Z_L as a product of Z operators along irreducible loop (cannot be contracted to point) as in Fig. 5.4.

→ Z_L commutes with all stabilizers, trivially with all P_k and with all V_j since each V_j shares two qubits with Z_L .

→ product of Z s along reducible loop would be equal to product of stabilizer elements ⇒ can't be a logical operator

→ can also form product of Z s along second irreducible loop

→ do the same for products of X s

⇒ get 4 logical operators $X_{L,1}, X_{L,2}, Z_{L,1}$ and $Z_{L,2}$

Error detection or syndrome measurement if a measurement of one stabilizer element yields -1 , an error has occurred

→ Z -error on one qubit → both plaquettes P_k , to which qubit belongs, measure -1

→ X -error on one qubit → both stars V_j , to which qubit belongs, measure -1

Code distance since a string of Pauli operators across the entire lattice is required for a logical operation, an $L \times L$ Toric code has code distance L

Error correction for a single Pauli error → apply same Pauli operators

→ for errors composed of multiple Pauli operators → multiple choices → find best choice via so called *decoder*

→ surface of torus difficult to implement on hardware

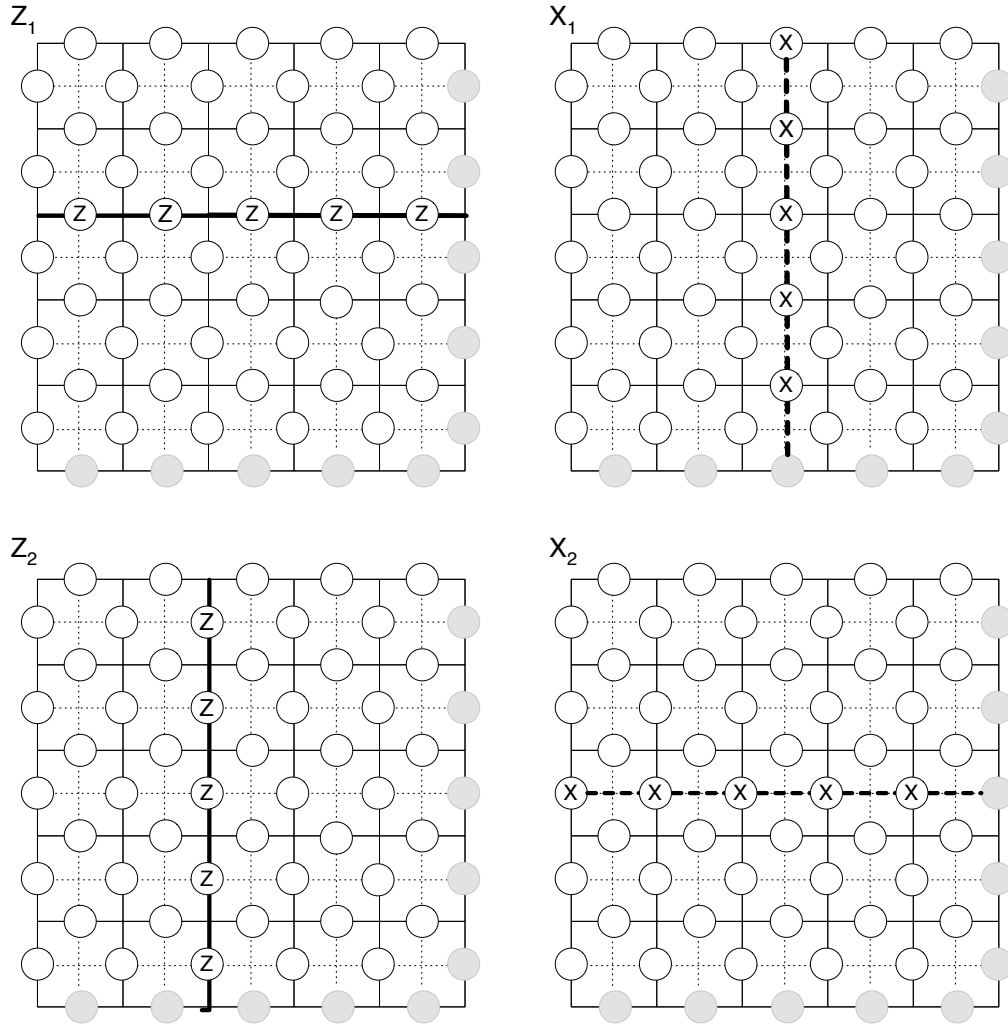


Figure 5.4: Logical operators in the Toric code.

5.3.2 Planar surface codes

For implementation need qubits storing data and ancilla qubits for measurements, see Eq. 5.35.

possible architecture shown in Fig. 5.5.

→ generalization of Toric code, e.g. logical operators are strings of Paulis from one end to the other ...

→ stabilizer elements at boundaries consist of 3 Pauli operators only (in bulk 4 Pauli operators),

→ horizontal boundaries Z -stabilizer, vertical boundaries X -stabilizers

→ circuits to measure stabilizer elements require ancilla qubits, possible circuits for bulk shown in Fig. 5.6.

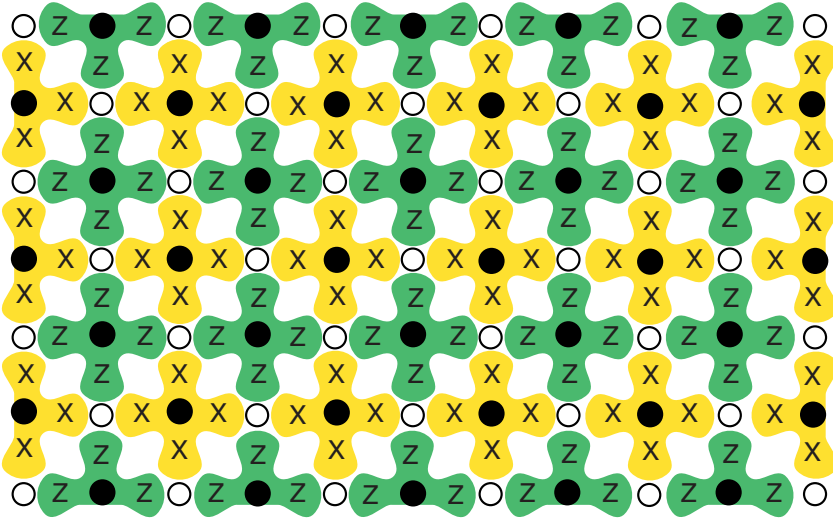


Figure 5.5: Possible architecture for realization of planar surface code with open boundaries. Filled circles are measurement qubits (i.e. ancillae) and empty circles are data qubits. Figure taken from Phys. Rev A **86**, 032324 (2012).

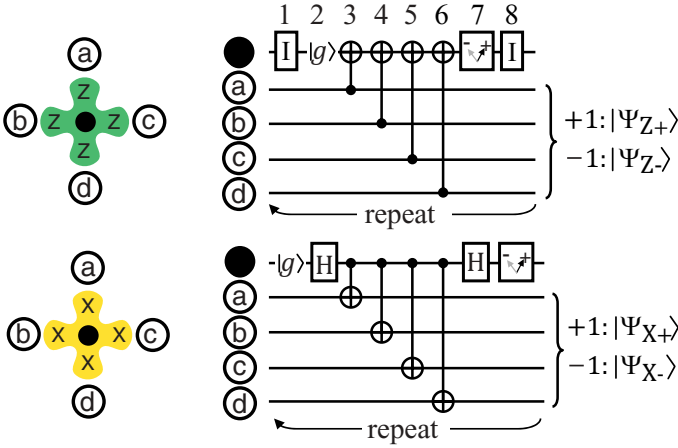


Figure 5.6: Circuits for measurement of Z and X stabilizer generators in bulk of surface code. Figure taken from Phys. Rev A **86**, 032324 (2012).

Small example: surface $[5,1,2]$ code A small example of the surface code with 5 physical qubits, 1 logical qubit and distance 2, can be implemented with 5 data qubits and 4 measurement qubits. The layout is shown in Fig. 5.7.

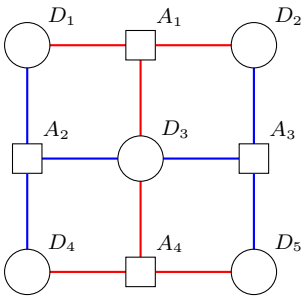


Figure 5.7: Layout of the surface $[5,1,2]$ code. D_j label data qubits and A_k label ancilla qubits.

→ the four stabilizer elements can be taken to read

$$Z_1 Z_2 Z_3, \quad X_2 X_3 X_5, \quad Z_3 Z_4 Z_5 \quad \text{and} \quad X_1 X_3 X_4 \quad (5.46)$$

⇒ all these stabilizer elements mutually commute

→ 5 qubits and 4 stabilizers ⇒ one logical qubit

→ possible choice of logical operators

$$X_L = X_1 X_4, \quad Z_L = Z_1 Z_2 \quad (5.47)$$

these commute with all stabilizers and anti-commute among themselves, $\{X_L, Z_L\} = 0$

Hardware requirements How many qubits are needed for given error probability?

errors can occur in any part of the computation

- in circuit of computation
- in error detection circuit
- in error correction circuit (decoder)

⇒ more qubits → more errors can be detected and corrected but higher probability for error since

- more qubits
- longer error detection circuit
- longer error correction circuit (decoder)

⇒ threshold for physical error per qubit p_{th} below which logical error p_L can be suppressed by increasing lattice size

→ for surface code

$$p_{th} \approx 1\% \quad (5.48)$$

→ need at least ~ 10000 physical qubits per logical qubit, i.e. $\sim 100 \times 100$ data qubits

⇒ need to increase current hardware by factor > 100

recent experiments by ETH Zürich team (code distance 3) [Nature **605**, 669 (2022)] and Google (shows that distance 5 is better than distance 3) [Nature **614**, 676 (2023)]

→ will require time and money.

→ Is there anything useful that can be done with existing hardware

Chapter 6

NISQ Quantum Computing

6.1 Today's quantum hardware

Several computing giants (Google, IBM, Intel, Alibaba, ...), startups (Rigetti, PsiQuantum, IonQ, Alpine Quantum Technologies...) and university labs (OpenSuperQ, ...) are developing quantum computers.

Current hardware (digital quantum computing):

- has up to 50-150 qubits
- allows to execute sequences of ~ 40 layers of gates
Layer: all gates that can be performed in parallel

See figure 6.1 for an illustration and 6.1 for more details on the layers.

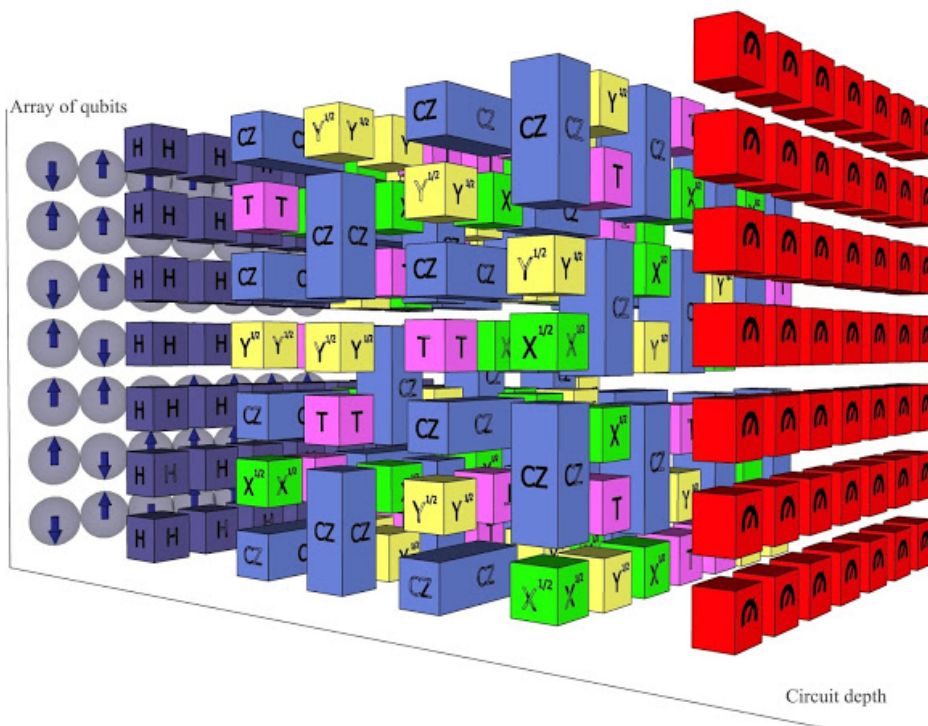


Figure 6.1: The space-time volume of quantum computation, where space indicates the number of qubits and time the number of gates. Source Google AI Blog.

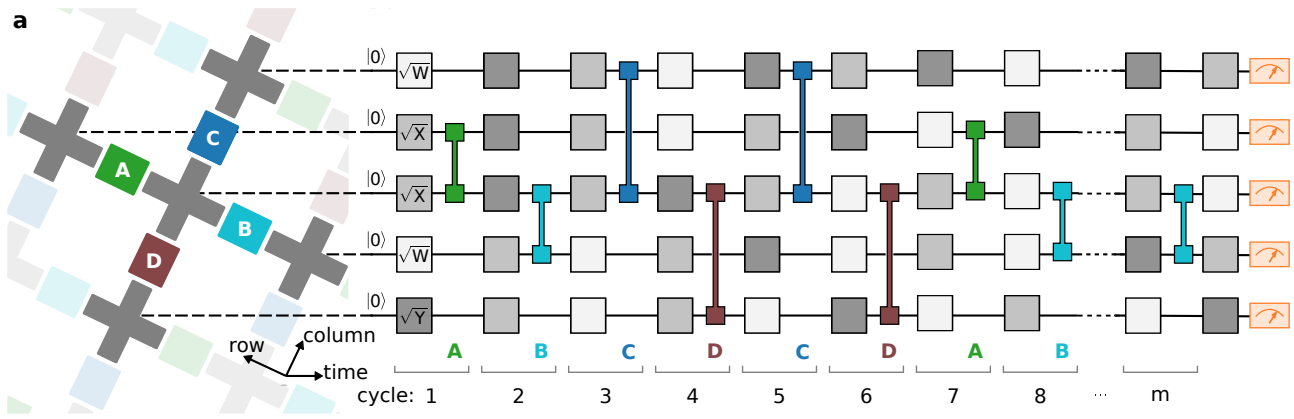


Figure 6.2: Gate sequence for Google’s beyond classical experiment, Nature **574**, 505 (2019). Only two-qubit gates that do not act on the same qubit(s) can be executed in parallel.

Current hardware capabilities The state of the art for different hardware platforms is:

	No. of qubits	fidelity single qubit gates	fidelity two-qubit gates	largest algorithm
trapped ions, see Nature 567 , 61 (2019)	53	99.0%	98.5%	~ 20 single-qubit and ~ 20 two-qubit gates on 7 qubits
superconducting qubits, see Nature 574 , 505 (2019)	54-72	99.85%	99.64%	1113 single-qubit and 430 two-qubit gates on 53 qubits
topological qubits				
quantum dots	2	~ 99%	~ 94%	single two qubit gates

→ table reports average gate fidelities for qubits in a larger processor, except for quantum dots.

→ isolated qubits can reach much higher fidelities (especially for trapped ions).

Error correction Running quantum error correction requires large qubit overhead.

→ would currently require at least 1 000 000 qubit processor or larger

→ current hardware not good enough to run e.g. Shore’s algorithm

→ we are in the ”NISQ” era of **Noisy Intermediate Scale Quantum** computers

⇒ What can we do with NISQ devices?

→ If you find a commercially relevant application you will get a lot of money.

Currently two areas of applications are mainly explored because they are most promising for NISQ use cases.

Quantum chemistry and material science: Full quantum calculations of chemical processes and material properties are very hard.

Here NISQ quantum computers may achieve things that classical computing can't solve. Whether this is the case is an open question.

This has important applications as the development of new materials, chemicals or pharmaceuticals is essentially a trial and error process.

→ one produces a substance and checks what it does (of course experience helps!)

→ with quantum computers we might simulate the working of a substance before we produce it.

Optimization problems Optimization problems are difficult to solve but appear in many areas of logistics and industry.

Fortunately the algorithms that are currently applied in both above areas are very similar.

- chemically stable configurations are low energy configurations

- solutions of optimization problems can be formulated as low energy states of a suitable Hamiltonian.

→ need algorithms to find low energy quantum states or ground states

6.2 Variational algorithms

The goal of a variational algorithm is to find the ground state of a Hamiltonian H , or a good approximation of it,

$$H |GS\rangle = E_0 |GS\rangle \quad (6.1)$$

where E_0 is the lowest eigenvalue of H

It is based on the Ritz-Rayleigh variational principle:

1. choose a parameterization for the quantum state that you want to compute, e.g.

$$|\psi(\vec{\theta})\rangle = \sum_{j=1}^{2^n} f_j(\vec{\theta}) |e_j\rangle \quad (6.2)$$

where the set of the $|e_j\rangle$ is a basis of the Hilbert space of H and $\vec{\theta} = (\theta_1, \theta_2, \dots, \theta_m)^T$ a set of variational parameters.

2. form

$$E(\vec{\theta}) = \frac{\langle \psi(\vec{\theta}) | H | \psi(\vec{\theta}) \rangle}{\langle \psi(\vec{\theta}) | \psi(\vec{\theta}) \rangle} \quad (6.3)$$

3. find the parameters $\theta_1, \theta_2, \dots, \theta_m$ such that $E(\vec{\theta})$ becomes minimal

$$E_{min} = \min_{\theta_1, \theta_2, \dots, \theta_m} [E(\vec{\theta})] \quad (6.4)$$

Note that because the variational state is in the Hilbert space spanned by H , see equation (6.2),

$$E_{min} \geq E_0 \quad (6.5)$$

$\Rightarrow$ the lower we get in equation (6.4), the better the approximation (we can't go too low).

Now how can we run this on a quantum computer?

Raleigh-Ritz variations on a quantum computer To explain how one can run the Raleigh-Ritz variational principle on a near term quantum computer, let us consider an example where

$$H = \sum_{j,l} c_{j,l} Z_j Z_l \quad (6.6)$$

Here Z_j is the Pauli-Z operator for qubit j , which takes on the eigenvalue 1 if the qubit is in the excited state and the eigenvalue -1 if the qubit is in the ground state.

$\Rightarrow$ if we measure the qubits in the z-basis (as we do) we find the eigenvalues z_j of the Z_j .

$\Rightarrow$ we can thus get approximation to the expectation value

$$E_\psi \approx \langle \psi | H | \psi \rangle \quad (6.7)$$

by sampling measurement results.

remember: if we measure an observable Z we obtain one of its eigenvalues z with a specific probability p (see: Copenhagen interpretation)

$\Rightarrow$ if we do $N \gg 1$ measurements and obtain m times the value z^* , then

$$p(z^*) \approx m/N \quad (6.8)$$

$\rightarrow$ If we do N measurements and observe the eigenvalues $z_j(\mu)$ in measurement number μ , then

$$\frac{1}{N} \sum_{\mu=1}^N \sum_{j,l} c_{j,l} z_j(\mu) z_l(\mu) \rightarrow \langle \psi | H | \psi \rangle \quad \text{for } N \rightarrow \infty \quad (6.9)$$

we could also consider a Hamiltonian of the form

$$H = \sum_{\alpha} c_{\alpha} \prod_{\{j_{\alpha}\}} Z_{j_{\alpha}}, \quad (6.10)$$

i.e. a Hamiltonian that is a sum of products of Z_j operators.

Thus, given a Hamiltonian of the form (6.6) or (6.10), we can run the variation as follows.

1. choose a set of unitaries, i.e. gates that can be implemented on your quantum device, which depend on the variational parameters,

$$U_j(\theta_j) \quad \text{with} \quad U_j^\dagger(\theta_j) U_j(\theta_j) = \mathbb{1} \quad \text{for all } j = 1, 2, \dots, m \quad (6.11)$$

2. prepare the quantum state

$$|\psi(\vec{\theta})\rangle = U_m(\theta_m) U_{m-1}(\theta_{m-1}) \dots U_1(\theta_1) |0, 0, \dots, 0\rangle \quad (6.12)$$

on your quantum device. Here $|0, 0, \dots, 0\rangle$ is the state where all qubits are in their ground state.

3. measure $N \gg 1$ times all qubits in the z -basis to get

$$E(\vec{\theta}) = \langle \psi(\vec{\theta}) | H | \psi(\vec{\theta}) \rangle \quad (6.13)$$

according to equation (6.9).

Note that the division by $\langle \psi(\vec{\theta}) | \psi(\vec{\theta}) \rangle$ is absent since all U_j are unitary and the state $|\psi(\vec{\theta})\rangle$ is always normalized.

Importantly, averaging many measurement results makes the approach robust against errors in individual runs. Errors will be averaged out. This is why this can work without error correction.

4. find the parameters $\theta_1, \theta_2, \dots, \theta_m$ such that $E(\vec{\theta})$ becomes minimal

$$E_{min} = \min_{\theta_1, \theta_2, \dots, \theta_m} [E(\vec{\theta})] \quad (6.14)$$

We have thus formulated the Rayleigh-Ritz variational principle as an optimization problem in a quantum classical hybrid algorithm.

- the quantum processor only evaluates the objective function $E(\vec{\theta})$
- the optimization algorithm for choosing $\vec{\theta}$ is done by classical optimizer

We can generalize the approach to Hamiltonians of the form

$$H = \sum_{\alpha} c_{\alpha} \prod_{\{j_{\alpha}\}} Z_{j_{\alpha}} + \sum_{\beta} c_{\beta} \prod_{\{l_{\beta}\}} X_{l_{\beta}}. \quad (6.15)$$

the only extra thing we need to do is rotate the qubits such that $x \rightarrow z$ before measuring to measure the terms $\prod_{\{l_{\beta}\}} X_{l_{\beta}}$.

$$U = \exp\left(-i \frac{\pi}{4} Y\right) \quad (6.16)$$

$$\Rightarrow U X U^{\dagger} = Z, \quad U^{\dagger} Z U = X \quad (6.17)$$

$$\Rightarrow \langle \psi(\vec{\theta}) | X | \psi(\vec{\theta}) \rangle = \langle \psi(\vec{\theta}) | U^{\dagger} Z U | \psi(\vec{\theta}) \rangle \quad (6.18)$$

$\Rightarrow$ applying unitary gate U before measuring Z is equivalent to measuring X

$\rightarrow$ the same can be done for including terms with Y operators.

The minimization in (6.14) needs a classical optimizer. One possible candidate is Gradient Descent

Gradient Descent For a variational quantum algorithm, we need to find $E_{min} = \min_{\theta_1, \theta_2, \dots, \theta_m} [E(\vec{\theta})]$
 $\rightarrow$ need an optimizer for the function $E(\vec{\theta})$.

Gradient descent generates the iteration sequence $\vec{\theta}_1, \vec{\theta}_2, \vec{\theta}_3, \dots$, where

$$\vec{\theta}_{k+1} = \vec{\theta}_k - a \nabla E(\vec{\theta}) \Big|_{\vec{\theta}=\vec{\theta}_k} \quad (6.19)$$

Here $a > 0$ is a step size and

$$\nabla E(\vec{\theta}) = \left[\frac{\partial E(\vec{\theta})}{\partial \theta^{(1)}}, \frac{\partial E(\vec{\theta})}{\partial \theta^{(2)}}, \frac{\partial E(\vec{\theta})}{\partial \theta^{(3)}}, \dots \right]^T \quad (6.20)$$

is the gradient of $E(\vec{\theta}_k)$ ($\theta^{(j)}$ is the component j of $\vec{\theta}$).

$\Rightarrow$ For suitably chosen step size a , this will converge to $E_{min} = \min_{\theta_1, \theta_2, \dots, \theta_m} [E(\vec{\theta})]$.

can also make step size a depend on the step, $a \rightarrow a_k$.

$\rightarrow$ need to get $\nabla E(\vec{\theta}) \Big|_{\vec{\theta}=\vec{\theta}_k}$ from the quantum computer.

$\rightarrow$ one approach is finite differences (assuming we only have one variational parameter)

$$\nabla E(\theta) \approx \frac{E(\theta + h) - E(\theta)}{h} \quad (6.21)$$

for h small enough.

measure $E(\theta)$ via sampling $\Rightarrow$ will have noise $\rightarrow E(\theta) + \delta E$, where δE denotes the error or fluctuation due to noise.

on quantum chip, from measurement results, will get

$$\frac{[E(\theta + h) + \delta E_1] - [E(\theta) + \delta E_2]}{h} \approx \nabla E(\theta) + \frac{\delta E_1 - \delta E_2}{h} \quad (6.22)$$

$\Rightarrow$ can't get information about $\nabla E(\theta)$ for $|\delta E_1 - \delta E_2| \gtrsim h$ (note that δE_1 and δE_2 can be positive or negative)

$\rightarrow$ can cause problems.

for gates of the form $U(\theta_j) = \exp(-i\theta_j \mathcal{O}_j)$, where the operator $\mathcal{O}_j$ only has eigenvalues $+1$ or -1 (e.g. is a Pauli operator), one can use the so called *parameter shift rule*

$$\nabla E(\theta) = E\left(\theta + \frac{\pi}{4}\right) - E\left(\theta - \frac{\pi}{4}\right) \quad (6.23)$$

which we will derive in the exercises.

Next we will show that variational algorithms can be used to solve optimization problems.

6.3 Quantum Approximate Optimization Algorithm (QAOA)

Combinatorial optimization problems consist of clauses for sets of bits (for the moment we consider classical bits)

A clause is a constraint on a subset of bits

A combinatorial optimization problem can be formulated as the minimization of a *cost function* of the form

$$C(z) = - \sum_{\alpha=1}^m C_{\alpha}(z) \quad (6.24)$$

where

- $z = z_1 z_2 \dots z_n$ denotes a bit-string, like 100110100111...
- there are n bits
- each $C_{\alpha}(z)$ is a clause, where $C_{\alpha}(z) = 1$ means the clause is satisfied and $C_{\alpha}(z) = 0$ means it isn't.
- there are m clauses
- the clauses C_{α} typically don't depend on all bits

$\Rightarrow$ the more clauses are satisfied the lower is $C(z)$

Quantum Formulation $\rightarrow$ formulate the cost function $C(z)$ as a Hamiltonian:

$$H_C = - \sum_{\alpha} \sum_{\mu=1}^{m_{\alpha}} J_{\mu} \prod_{j \in I(\mu)} Z_j \quad (6.25)$$

where Z_j is a Pauli- Z operator on qubit j , m_{α} is the number of terms in clause α , J_{μ} is a weight for the term μ in the clause α and $I(\mu)$ the set of all qubit indices that appear in term μ .

Example an example where all clauses just involve pairs of neighboring qubits is

$$H_C = - \sum_{\langle j,l \rangle} Z_j Z_l \quad (6.26)$$

$\rightarrow$ can solve combinatorial optimization problem by finding ground state of H_C ,

$$H_C |\bar{z}_1, \bar{z}_2, \dots, \bar{z}_n\rangle = E_0 |\bar{z}_1, \bar{z}_2, \dots, \bar{z}_n\rangle \quad (6.27)$$

where $\bar{z}_1 \bar{z}_2 \dots \bar{z}_n$ is the bit-string that minimizes the classical cost function $C(z)$.

$\rightarrow$ can now use a variational algorithm as discussed in sec. 6.2 to find the state $|\bar{z}_1, \bar{z}_2, \dots, \bar{z}_n\rangle$ or a good enough approximation.

$\rightarrow$ prepare the quantum state

$$|\psi(\vec{\theta})\rangle = U_k(\theta_k) U_{k-1}(\theta_{k-1}) \dots U_1(\theta_1) |0, 0, \dots, 0\rangle \quad (6.28)$$

and seek $\vec{\theta}$ such that

$$E_{min} = \min_{\theta_1, \theta_2, \dots, \theta_m} [E(\vec{\theta})] \quad \text{where} \quad E(\vec{\theta}) = \langle \psi(\vec{\theta}) | H | \psi(\vec{\theta}) \rangle \quad (6.29)$$

$\Rightarrow$ at minimum will get parameters $\bar{\theta}_1, \bar{\theta}_2, \dots, \bar{\theta}_m$ and state $|\psi(\bar{\theta}_1, \bar{\theta}_2, \dots, \bar{\theta}_m)\rangle$ such that

$$|\langle \bar{z}_1, \bar{z}_2, \dots, \bar{z}_n | \psi(\bar{\theta}_1, \bar{\theta}_2, \dots, \bar{\theta}_m) \rangle| \approx 1 \quad (6.30)$$

$$\text{but not necessarily } |\bar{z}_1, \bar{z}_2, \dots, \bar{z}_n\rangle = |\psi(\bar{\theta}_1, \bar{\theta}_2, \dots, \bar{\theta}_m)\rangle \quad (6.31)$$

$\rightarrow$ What is a good ansatz, i.e. a good set of unitaries $U_m(\theta_m) U_{m-1}(\theta_{m-1}) \dots U_1(\theta_1)$?

QAOA Ansatz The following ansatz is promising:

$$|\psi(\vec{\gamma}, \vec{\beta})\rangle = U_B(\beta_p) U_C(\gamma_p) U_B(\beta_{p-1}) U_C(\gamma_{p-1}) \dots U_B(\beta_1) U_C(\gamma_1) |s\rangle \quad (6.32)$$

where:

- the two alternating unitaries $U_C(\gamma)$ and $U_B(\beta)$ read

$$U_C(\gamma) = e^{-i\gamma H_C} \quad (6.33)$$

$$U_B(\beta) = e^{-i\beta H_B} \quad (6.34)$$

where H_C is as in Eq. (6.25) and

$$H_B = \sum_{j=1}^n X_j \quad (6.35)$$

- note that

$$U_B(\beta) = e^{-i\beta H_B} = \prod_{j=1}^n e^{-i\beta X_j} \quad (6.36)$$

can be composed of single qubit gates

- in turn

$$U_C(\gamma) = e^{-i\gamma H_C} = \prod_{\alpha} \prod_{\mu=1}^{m_{\alpha}} e^{-i\gamma J_{\mu} \prod_{j \in I(\mu)} Z_j} \quad (6.37)$$

$\rightarrow$ need to decompose each $\exp[-i\gamma \prod_{j \in I(\mu)} Z_j]$ into two qubit gates.

- $\vec{\gamma} = (\gamma_1, \gamma_2, \dots, \gamma_p)^T$ and $\vec{\beta} = (\beta_1, \beta_2, \dots, \beta_p)^T$ are the variational parameters.
- increasing the depth p (the number of U_C, U_B alternations) can only improve the approximation (choosing $\gamma_p = \beta_p = 0$ reproduces $p-1$ case)
- the state $|s\rangle$ is a uniform superposition of all possible bit strings

$$|s\rangle = \frac{1}{\sqrt{2^n}} \sum_z |z\rangle \quad (6.38)$$

and can be generated by applying a Hadamard to each qubit

$$|s\rangle = H^{\otimes n} |0, 0, \dots, 0\rangle \quad (6.39)$$

The QAOA ansatz thus

- first generates a superposition of all possible bit-strings z
- U_C multiplies different phases onto all bit-strings
- U_B mixes or reshuffles the amplitudes for all bit-strings

Relation to adiabatic quantum computation Adiabatic quantum computation considers the Hamiltonian

$$H_{adia}(t) = \lambda(t)H_C - [1 - \lambda(t)]H_B \quad (6.40)$$

where

- λ is ramped slowly from $\lambda(t = 0) = 0$ to the final value $\lambda(t = t_f) = 1$
- the initial state $|\Psi_i\rangle$ is the ground state of $-H_B$,

$$|\Psi_i\rangle = \prod_j |+_j\rangle = \prod_j \frac{|0_j\rangle + |1_j\rangle}{\sqrt{2}} \quad (6.41)$$

can be prepared via $|\Psi_i\rangle = H^{\otimes n} |0, 0, \dots\rangle$.

- $|\Psi_i\rangle = |s\rangle$

If the ramp of $\lambda(t)$ is slow enough $\rightarrow$ evolution adiabatic: system (quantum chip) will remain in its ground state

$\Rightarrow$ final state $|\Psi_f\rangle$ is the ground state of H_C

$\Rightarrow$ optimization problem solved

Evolution can be written as

$$|\Psi(t)\rangle = \mathcal{T} \exp \left[-i \int_{t_i}^{t_f} ds H_{adia}(s) \right] |\Psi_i\rangle \quad (6.42)$$

$$\approx \mathcal{T} \exp \left[-i \sum_{j=1}^N \delta t H_{adia}(t_j) \right] |\Psi_i\rangle \quad (6.43)$$

$$\approx \mathcal{T} \prod_{j=1}^N \exp [-i \delta t H_{adia}(t_j)] |\Psi_i\rangle \quad (6.44)$$

$$\approx \mathcal{T} \prod_{j=1}^N \exp [-i \delta t \lambda(t_j) H_C] \exp [-i \delta t (1 - \lambda(t_j)) H_B] |\Psi_i\rangle \quad (6.45)$$

where

- $\mathcal{T}$ is the time ordering operator that puts the terms with the latest times left
- $\delta t = (t_f - t_i)/N$
- the approximation in Eq. (6.43) is good for $N \gg 1$
- the approximation in Eq. (6.44) is the Suzuki Trotter formula

$$\exp[\lambda(H_1 + H_2)] = \exp[\lambda H_1] \exp[\lambda H_2] + \mathcal{O}(\lambda^2) \quad (6.46)$$

for $[H_1, H_2] \neq 0$

- the approximation in Eq. (6.45) is again the Suzuki Trotter formula (for $H_1 = \lambda(t_j)H_C$ and $H_2 = (1 - \lambda(t_j))H_B$).
- Eq. (6.45) is the same as the QAOA ansatz in Eq. (6.32) for $\gamma_j = \delta t \lambda(t_j)$ and $\beta_j = \delta t (1 - \lambda(t_j))$

$\Rightarrow$ for $p \rightarrow \infty$, QAOA will solve the optimization problem

6.4 Quantum Simulation: Computing Quantum Dynamics

The dynamics generated by a quantum many-body Hamiltonian H is hard to simulate classically

$$|\psi(t)\rangle = e^{-iHt} |\psi(0)\rangle \quad (6.47)$$

This can lead to arbitrarily complex states, and it typically does.. For n qubits, such a state reads

$$|\psi\rangle = \sum_{j_1, j_2, \dots, j_n=0}^1 \alpha_{j_1, j_2, \dots, j_n} |j_1, j_2, \dots, j_n\rangle = \sum_{\vec{j}} \alpha_{\vec{j}} |\vec{j}\rangle \quad (6.48)$$

which has 2^n coefficients $\alpha_{\vec{j}}$.

Classically could solve the Schrödinger equation, which leads to the 2^n coupled ordinary differential equations,

$$\frac{d}{dt} \alpha_{\vec{j}} = \sum_{\vec{l}} H_{\vec{j}; \vec{l}} \alpha_{\vec{l}} \quad (6.49)$$

where $H_{\vec{j}; \vec{l}} = \langle \vec{j} | H | \vec{l} \rangle$, see also section 4.6.

Quantum dynamics can be decomposed into gates.

→ assume

$$H = \sum_{\alpha=1}^{N_G} H_{\alpha} \quad (6.50)$$

where

$$U_{\alpha} = \exp(-iH_{\alpha}\delta t) \quad (6.51)$$

are gates that can be implemented on the chosen hardware.

→ U_{α} are typically single- or two-qubit gates.

⇒

$$[H_{\alpha}, H_{\beta}] \neq 0 \quad \text{for } \alpha \neq \beta \quad (6.52)$$

→ can use the Lie-Trotter formula to build gate sequence.

Lie-Trotter formula: For two operators or square matrices A and B , we have

$$\exp(A + B) = \lim_{N \rightarrow \infty} [\exp(A/N) \exp(B/N)]^N \quad (6.53)$$

⇒ for small $\delta t = t/N$, can approximate the time evolution

$$\exp[-i(H_{\alpha} + H_{\beta})t] \approx [\exp(-iH_{\alpha}\delta t) \exp(-iH_{\beta}\delta t)]^N = [U_{\alpha} U_{\beta}]^N \quad (6.54)$$

→ can iterate to get

$$e^{-iHt} = \left[\prod_{\alpha=1}^{N_G} U_{\alpha} \right]^N \quad (6.55)$$

The main challenge for this approach is however that the number of required gates for simulating regimes of interest is often too large for existing hardware.