Hiu Yung Wong

Introduction to Quantum Computing

From a Layperson to a Programmer in 30 Steps

Springer

Introduction to Quantum Computing

Hiu Yung Wong

Introduction to Quantum Computing

From a Layperson to a Programmer in 30 Steps

Springer

Hiu Yung Wong
San José State University
San José, CA, USA

ISBN 978-3-030-98338-3 ISBN 978-3-030-98339-0 (eBook)
https://doi.org/10.1007/978-3-030-98339-0

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

To my Family

Preface

Quantum computing sounds fancy but also scary. Is it possible to learn quantum computing rigorously without years of training in mathematics, physics, and computer science? This book tries to address this question. For any serious learner who has some basic training in science or engineering at the junior level, they should find this textbook useful to prepare themselves to start a serious quantum computing journey within a few months.

This book is based on an introduction to quantum computing class I teach at San José State University. The target audiences are senior and master's students who do not have a strong background in linear algebra and physics. I designed the course materials by working backward from the final goal, namely being able to understand the nuts and bolts, derive the equation, and perform numerical substitution in common algorithms such as the Quantum Phase Estimation algorithm and Shor's algorithm. The materials are derived until some basic matrix and vector operations.

This book is suitable for self-study or as a textbook for a one-semester or one-quarter introduction class at senior or master's level. After spending 60–90 hours of reading, students are expected to be able to understand the critical concepts of quantum computing, construct and run quantum computing circuits, and interpret the results. For self-study students, it is recommended to read the book and do the exercise for 3–4 hours per week for about 20 weeks (about 1–2 chapters per week).

There are two parts in this book. Part I teaches linear algebra with an emphasis on quantum computing applications. Part II teaches quantum computing gates and algorithms. If the readers have not been exposed to quantum computing and do not have a very strong background in linear algebra, it is recommended to read from the first chapter to the final chapter in sequence. Part I is "entangled" with Part II. Many important quantum computing concepts are introduced in Part I. It is not recommended to start with Part II even if the readers already have some background in linear algebra. Part I of this book can also be used to enhance the students' understanding of linear algebra as standalone teaching material.

A few didactic approaches are used throughout the book. First, fundamental concepts are repeated when it is used. Such an "immersion" approach helps emphasize the critical aspects of the concepts. For example, the concept of basis,

which is the main source of confusion in most calculations, is repeated throughout the book to remind the readers about the importance of knowing which basis is being used in the calculations. Second, background knowledge is introduced only when it is really needed. For example, the Bloch sphere and the general unitary gate are only introduced right before the quantum phase estimation, almost at the end of the book. Third, numerical examples are used throughout the book and the calculations are explained clearly to avoid ambiguities. For readers who are not well trained in abstract linear algebra, numerical substitution is the best way to make sure they understand the meaning of each equation. Fourth, real quantum computers (IBM-Q) are used to implement some of the algorithms, and students are encouraged to write their own simple simulator using Google Co-lab. Fifth, I try to use many analogies to help the readers understand better the absurd concepts in quantum computing. Finally, *do not get scared by the lengthy equation blocks. Most equation blocks can be written in just one or two lines, but I deliberately show the tiny steps to help the reader to follow the flow.*

For schools that do not have the resources to create materials to start an introduction to quantum computing class, teaching material can be created by following the flow of this textbook.

Finally, I need to acknowledge the students in my EE225 class in Spring 2020 and Fall 2021 for taking my class seriously.

For any comments, please send an email to intro.qc.wong@gmail.com.

San José, CA, USA Hiu Yung Wong
January 2022

Contents

Part I
Linear Algebra for Quantum Computing

Chapter 1
The Most Important Step to Understand Quantum Computing

1.1 Learning Outcomes

Know when to believe and when to question.

1.2 How to Learn Quantum Computing

From the moment you started reading this book, I trust that you want to learn quantum computing and you are interested in quantum computing. I would not waste your time by spending even a paragraph to depict the rosy picture of quantum computing.

This book is for people who are interested in learning quantum computing to the level that they can describe some algorithms by showing on paper how each quantum qubit evolves. Of course, if one can show that on paper, one can also program it. And if one can do so, one is ready to have a better appreciation of the power of quantum computing and its limitations and, needless to say, the important concepts in quantum computing such as superposition, entanglement, quantum parallelism, quantum supremacy, etc.

This book does not require any solid knowledge in quantum mechanics or linear algebra.

But "no pain, no gain." While I try to write it in a way to guide you step by step to achieve the final goal—being able to describe how each qubit evolves in quantum algorithms, you are expected to follow the flow of the book closely and seriously.

The most difficult part of learning quantum mechanics is due to the fact that we do not have firsthand experience. Have you ever questioned why $1 + 1 = 2$? If you have not, probably this is because you had firsthand experience. You do get two apples if each of your parents gives you one apple. What if you did not have such an experience and you are a very serious learner? I believe you will spend days

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_1

and nights thinking about the philosophy behind $1 + 1 = 2$. But very often, in our learning experience, we do not question what we are taught and we just believe it. I did not question the methodology of performing division when my primary school teacher taught us. I learned it and believed it, and indeed it is useful for the rest of my life. If I had spent days and nights questioning it when I was only 7 years old, I probably would have got lost and given up my study. Do you agree? And now I do understand the division methodology after many more years of training and I agree the methodology is correct. So, one important element of learning is to believe. To be successful in reading this book, I want you to believe what I say.

I am not a cult leader. I only need you to believe me when you are reading this book. I hope you can challenge and question me once you have finished the book. To avoid being "brainwashed" and to retain your curiosity, I would like you to note down everything that you cannot accept easily. The document may start with:

> Today I started learning quantum computing. The following is a list of concepts and methodologies I don't really understand. The book might be incomplete, inaccurate, or even wrong. For now, I use them as tools. But I need to use them very carefully.

Are you ready? Let us move one step forward.

Chapter 2
First Impression

2.1 Learning Outcomes

Have an idea of how quantum computing is different from classical computing; Gain first impressions of the keywords in quantum computing.

2.2 What Is Quantum Computing?

Qubit, Superposition, and Entanglement: In a classical computer, a bit is represented by a physical quantity with two **states** to represent 0 and 1. It can be the voltage in a circuit (e.g. 5 V for 1 and 0 V for 0). It can be the magnetization direction in a magnetic drive (e.g. up for 1 and down for 0). It can be the charge density in the charge storage layer in the flash memory (e.g. low for 1 and high for 0). In a quantum computer, the information is represented by **qubits**. Like the classical bit, each qubit still has two distinct states and we say that they are **orthogonal** to each other because they cannot exist with 100% certainty at the same time. We also call them the **basis vectors** and I will call them **basis states** most of the time. We can still label them as "0" and "1." In addition to the two distinct states, a qubit may also have many other states which are the **superposition** of the two distinct states (or the basis states). This is what is fundamentally different from the classical bit. In these states, the two distinct (orthogonal) basis states are allowed to exist at the same time with $<100\%$ certainty.

To illustrate the meaning of superposition, we can use an interesting analogy. Try to physically and classically spin a coin on the table (not to be confused with electron spin). The coin has a head (to represent 0) and a tail (to represent 1) and they are the two orthogonal states. When it is spinning, we cannot tell whether it is head or tail (both have $<100\%$ certainty). We can say the coin is in a state of the superposition of head and tail. When it stops spinning, it will be either head or tail (with 100%

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_2

certainty). We say the superposition state **collapses** to either the head state or the tail state. This, of course, is not a real qubit and you cannot build a quantum computer out of many spinning coins. So do not work too hard to understand. On the other hand, after finishing this book, I hope you will be able to appreciate why this is a good analogy and how it is different from a real quantum qubit. Later, I will tell you another analogy that is more romantic.

Superposition is one of the fundamental operation principles in quantum computing. Superposition allows us to *perform a calculation on multiple basis states at the same time*, which is not possible in classical computing. However, it is not a stranger to us. We learned superposition in wave theory in which two wavelets can superimpose on each other. In electromagnetism, light can be polarized in any direction. Any linear polarization is just a linear combination of horizontal and vertical polarization. If we represent the horizontal polarization as a horizontal vector and the vertical polarization as a vertical vector, any linear polarization, $\vec{V}$, is just a linear combination of the two vectors. I can write

$$\begin{aligned} \textit{Final Polarization} = \ & \textit{this much Horizontal Polarization} + \\ & \textit{that much Vertical Polarization} \end{aligned} \tag{2.1}$$

or more succinctly

$$\vec{V} = \alpha \hat{V}_x + \beta \hat{V}_y \tag{2.2}$$

where α and β are just some numbers (coefficients) and $\hat{V}_x$ and $\hat{V}_y$ are the horizontal and vertical polarization unit vectors (which are basis vectors in this case), respectively. Do not be scared by these notations. They are just notations. You just need to accept it and then you will not be afraid of it. $\hat{V}_x$ and $\hat{V}_y$ are the orthogonal states in the system. There is a genius called Dirac and he decided to write in this way

$$|V\rangle = \alpha\,|V_x\rangle + \beta\,\left|V_y\right\rangle \tag{2.3}$$

or we can also write in this way

$$\begin{aligned} |\textit{Final Polarization}\rangle = \ & \textit{this much}\,|\textit{Horizontal Polarization}\rangle + \\ & \textit{that much}\,|\textit{Vertical Polarization}\rangle \end{aligned} \tag{2.4}$$

This is called the **Dirac bra–ket notation**. The power of the Dirac bra–ket notation will be discussed later. However, I would like you to appreciate that there is no difference between these 4 ways of writing the superposition and they contain the same amount of information.

	Classical Computing	Quantum Computing
Representation of Information	0, 1, ***(bit)*** (voltage level)	a|0>+b|1> ***(qubit)*** (superposition of basis states)
Operation Principles	Classical Boolean Algebra	**Superposition, Entanglement**, Interference
Logic Gate	Usually non-reversible	Unitary, Reversible
Robustness	Very robust, immune to noise even at 400K	Very sensitive to noise, operating at mK, low fault tolerance
Measurement	Can copy, deterministic	No-cloning, statistical
Hardware	Transistor	Trapped Ions, Josephson Junction, Topological Insulator, Photonics, Defect Center, Electron Spin, Neutral Atoms
Information size	Process 1 n-bit state at a time	May process 2^n n-bit state at a time

Fig. 2.1 Comparison between classical and quantum computing

Another fundamental principle in quantum computing is the use of **entanglement**. We are not ready to discuss this yet. But you need to put it in your must-know list.

Figure 2.1 contrasts the difference between classical and quantum computing. We have discussed the first two items, and let us discuss the rest.

Reversible Computing In classical computing, logic gates are non-reversible. For example, in a NAND gate (Fig 2.2), when the two inputs are 1 and 0, the output is 1. When the two inputs are both 0, the output is also 1. Therefore, if you measure the output and find it to be 1, you cannot deduce the inputs. In quantum computing, every operation has to be **reversible** (except measurement). If I know the output and the type of gate, I can deduce the inputs. In Fig 2.2, we see that the output of a quantum gate (CNOT which we will learn later) has a one-to-one correspondence to the input. If you tell me the outputs, I can deduce the inputs as long as I know it is a CNOT gate. This is because quantum mechanics requires nature to evolve in a "**unitary**" way, and as a result, every quantum gate has to be unitary and thus reversible (we will discuss the mathematical meaning of unitary later). We can also regard every quantum gate as a rotation operator for a vector (this is an example where you just need to believe me). If a vector can be rotated in one direction, naturally, we expect that we should be able to rotate it back (reversible). The most important consequence of nature being unitary is that the overlap (inner product) of

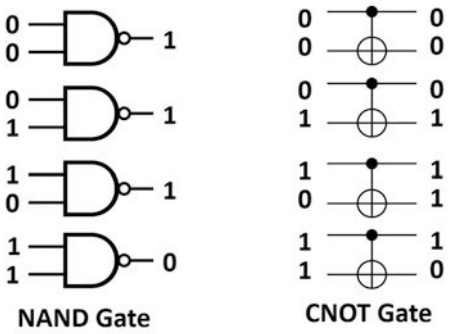


Fig. 2.2 Classical gates (left) are usually non-reversible, but quantum gates (right) have 1-to-1 correspondence between the output and input

two vectors is preserved if they are rotated. This is just like the projection of one edge of a triangle to another edge does not change no matter how you rotate the triangle as a whole.

Robustness A classical computer is very robust against noise. We can have billions of transistors working well even at 400 K. However, a quantum computer is very sensitive to noise. Most of them need to work in the mK range. 1 mK is 1/1000 K. Liquid helium (L-He) temperature is 4.2 K. This means that the quantum computer chip needs to operate in an environment a hundred times cooler than the L-He. This is because the thermal noise is smaller at a lower temperature. Thermal noise is proportional to temperature. Thermal noise energy is about 25 meV at 300 K, 0.35 meV at 4.2 K, and 0.0083 mV at 100 mK. 1 eV is the energy an electron will gain when it travels across 1 V potential difference. Unlike the classical bit which is implemented using a physical bit with large energy separation between states (e.g. 1 V for 1 and 0 V for 0 which results in an energy separation of 1 eV for an electron), qubits are usually implemented with a physical qubit with small state energy separation (e.g. the separation of a spin qubit's up and down is about 0.2 meV). Therefore, low temperature is needed so that the thermal energy (noise) is much less than the state separation energy. To further mitigate the impact of noise, error correction and signal processing are very important in quantum computing. We will not discuss error correction codes in this book. You definitely want to learn them to be a serious quantum computing programmer.

Measurement and Collapse When you perform a classical measurement, you actually copy the output to somewhere else (e.g. to your brain or a piece of paper). Inside the computer, you still copy it to a register or memory. In quantum computing, there is a theory called the **No Cloning Theory**. This theory prevents us from cloning all quantum states (certain states can be cloned but not all). Therefore, you cannot copy a quantum state from one place to another place. You might have heard of **quantum teleportation** (no worry if you have not and we will discuss it later). It seems that you can "copy" something from one place to another in no time. But in quantum teleportation, when we "copy" a state, we need to destroy the state at the

source. So actually, it is not a copy action and that is why it is called teleportation during which a state is transferred from a place to another.

There is one more subtle thing about measurement in quantum computing. When you measure a superposition state, the state will collapse to one of the basis states (0 or 1) and this is completely random. In the spinning coin analogy, a measurement is like you are bluntly stopping it from spinning and you do not know whether it will end up in head or tail. However, it is not completely random. The chance of collapsing to a certain basis state depends on the values of α and β which we mentioned earlier (Eq. (2.3)). Then how a quantum computer is going to be useful if even the measurement is random? In some algorithms, we design it so that eventually it will give us a certain basis state for sure. Then, based on the output (0 or 1), we can determine the information we are trying to get. This is just like if it rains, I am sure there are clouds in the sky. Feeling the rain is the measurement and knowing there are clouds in the sky is the information I want to get. In some algorithms, it has a high chance of giving us the correct solution (e.g. factorization). We can verify the solution easily using a classical computer (e.g. multiplication of factors). Through a few iterations, we can get the result we want. Well, in principle, it is possible you never get the solution in the second category, but you need to be extremely unlucky (or lucky).

Implementation of Qubits To implement a bit in classical computers, we can use any classical switches. We have used mechanical switches, vacuum tubes, and nowadays, transistors. In a quantum computer, we need something small so that quantum phenomena can exhibit. It also needs to have two distinguishable quantum states as the basis states (0 and 1). For example, a spin up electron and a spin down electron have different energies under an external magnetic field. We can use spin up to represent 0 and spin down to represent 1. According to quantum mechanics, the electron can be in a superposition of spin up and spin down (you need to trust me again if this is not clear to you). Therefore, the spin of an electron under an external magnetic field can be used to implement a qubit. There are other possible qubit implementations such as trapped ions, superconducting qubit using Josephson Junction, topological insulator, photons, defect centers in diamond or silicon carbide. We will not discuss them in this book. But I would like you to understand that there are trade-offs between them. These include the size of the device, the speed, the coherence time (a very important concept describing how fast the qubit information will lose), scalability (i.e. can we use it to build a computer with many qubits eventually?), etc.

Information Size Superposition is a fancy concept but nothing special if we accept it like how we accept the fact that the wavelets can superimpose on each other. The power of superposition is the amount of information one can store when we have more than 1 qubit. How many possible states can we store with n physical bits in a classical computer? It is 2^n states. Of course, you can only store one of them at a time. For example, with 2 physical bits, you can store either the state 3 (11 in binary), 2 (10 in binary), or 1 (01 in binary), or 0 (00 in binary). For quantum computers, each of the classical states becomes the basis state and they are orthogonal to each

other (i.e. they cannot exist at the same time with 100% certainty). So, you have 2^n basis states if you have n qubits. That means the number of basis states grows exponentially with n. Of course, the n-qubit system can also be in a superposition state of the 2^n basis states. For example, if there are two photon polarization qubits, Eq. (2.3) becomes

$$|V\rangle = \alpha\,|V_{x1}V_{x2}\rangle + \beta\,\left|V_{x1}V_{y2}\right\rangle + \gamma\,\left|V_{y1}V_{x2}\right\rangle + \delta\,\left|V_{y1}V_{y2}\right\rangle \tag{2.5}$$

where α, β, γ, and δ are just some numbers and V_{x1}, V_{y1}, V_{x2}, and V_{y2} are the horizontal and vertical polarization vectors of the first and the second photons, respectively. You see that with 2-qubits, we have four basis vectors now, namely, $|V_{x1}V_{x2}\rangle$, $\left|V_{x1}V_{y2}\right\rangle$, $\left|V_{y1}V_{x2}\right\rangle$, and $\left|V_{y1}V_{y2}\right\rangle$. Again, do not be scared by the bra–ket notation. They are nothing but just labels for vectors and states. Please believe me and get used to it.

Equation (2.5) is succinct, but it will be more instructive if we can write it in the form of Eq. (2.4),

$$\begin{aligned}
&|\textit{System State}\rangle = \\
&\alpha\,|\textit{photon1 horizontally polarized and photon2 horizontally polarized}\rangle \\
&+\beta\,|\textit{photon1 horizontal polarized and photon2 vertically polarized}\rangle \\
&+\gamma\,|\textit{photon1 vertically polarized and photon2 horizontally polarized}\rangle \\
&+\delta\,|\textit{photon1 vertically polarized and photon2 vertically polarized}\rangle
\end{aligned} \tag{2.6}$$

Again, everything in the Dirac bra–ket is just a description of the state. The description can be a mathematical notation like in Eq. (2.5) or English like in Eq. (2.6) or if you want, in Cantonese. You may also write them as $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. From this, you see they do correlate with the classical states (i.e. all the classical states become the corresponding basis states in a quantum computer). So, why this is powerful? If you have 100 qubits, you can form a system of 2^{100} basis vectors. This is more than the number of atoms in the universe. If you can process the 100 qubits at the same time (all you need is just a 100-qubit gate), you are processing the 2^{100} coefficients at once. This is the so-called **quantum parallelism**. This is not possible in classical computers because if we need at least 1 atom to store the coefficient, there are not even enough atoms to store the coefficients and needless to think about processing the state.

2.3 Summary

We finished our second "step." This "step" is pretty shaky. I showed you many important concepts in quantum computing. I do not expect you to fully understand them. In the following "steps," we will start some serious calculations and we will

keep reinforcing the critical concepts we have learned. Before moving forward, let us do some exercises. I will keep reiterating the important concepts and analogies. I expect you do not need to look back to this step often.

Problems

2.1 Quantum Computing Account

Please go to IBM Quantum Experience and register an account, https://quantum-computing.ibm.com/.

2.2 My First Quantum Circuit

Try to create a circuit. Click on IBM Quantum Composer and drag the gates to form the following circuit (Fig. 2.3).

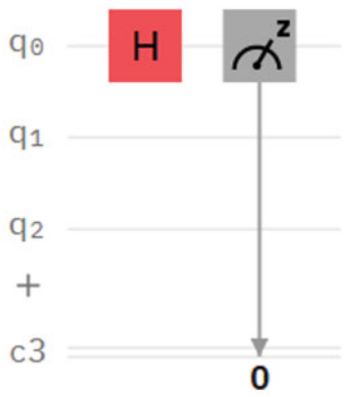
The first icon is a quantum gate called Hadamard gate and the second icon is a measurement. The input is from the left ($q0$). Do not worry if you do not understand what it is at this moment. But good job! You have created your first quantum circuit!

2.3 My First Quantum Computing

Submit the job. Click on the result when it is finished. You see some like this. You may submit to a simulator or real quantum hardware (Fig. 2.4).

We started with a state in $|0\rangle$ (a basis state). After the Hadamard gate, it becomes a superposition of $|0\rangle$ and $|1\rangle$, i.e. $\sim |0\rangle + |1\rangle$ (we will learn later). Then we performed a measurement on the output and the superposition state collapses to either $|0\rangle$ or $|1\rangle$. The histogram shows that about 50% of the chance we obtain $|0\rangle$ and another 50% of the chance we obtain $|1\rangle$. Cool! We have done our first quantum computation. This job was submitted to a real quantum computer, called "ibmq_casablanca." It is a 5-qubit quantum computer. When you read this book, I am not sure if this computer is still there. But definitely, you will find other more powerful ones if it has retired by then.

Fig. 2.3 Problem 2.2



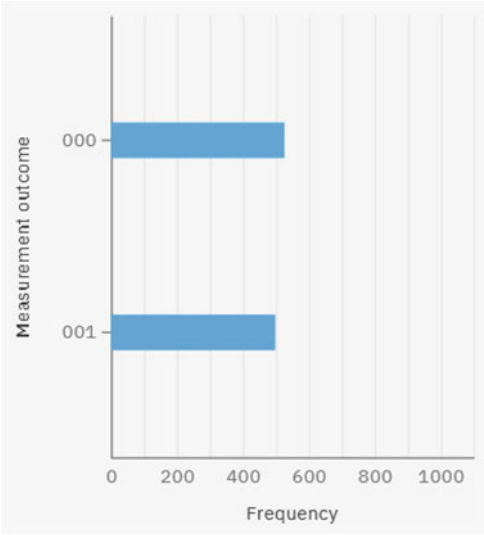


Fig. 2.4 Problem 2.3

Chapter 3
Basis, Basis Vectors, and Inner Product

3.1 Learning Outcomes

Understand the concept of basis; Be familiar with vector inner product; Reinforce the concept of vector representations; Be comfortable when dealing with hyperspaces that we cannot feel.

3.2 A Romantic Story

As promised, here I will tell you a romantic story. However, I need to admit that this is half real and half made-up. Also, *this analogy is not completely correct in quantum mechanics*. We will give a more rigorous explanation of the critical concepts later. Therefore, if you do not understand fully, do not worry too much. After moving forward with more "steps," you will appreciate the rights and wrongs in this analogy.

Once upon a time, there was a boy who liked a girl. But he did not know if the girl liked him. The girl treated him pretty well, but he was not sure if she just treated him as a good friend or she did like him. The boy had heard about quantum mechanics but did not understand too well. For him, the state of the girl was

$$|State\ of\ the\ girl\rangle = \alpha\,|Yes\rangle + \beta\,|No\rangle \tag{3.1}$$

Definitely, $|Yes\rangle$ (she likes him) and $|No\rangle$ (she does not like him) are two orthogonal basis vectors that never occur at the same time. For the boy, the girl is in a state which is a superposition of $|Yes\rangle$ and $|No\rangle$. Well, this is not quantum mechanics. But let us pretend it is, so we can have a romantic story. The boy dared not ask because asking is a measurement. The girl's state would collapse to either $|Yes\rangle$ or $|No\rangle$. If it collapses to $|No\rangle$, that would be the end of the world for him.

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_3

After waiting for months, the boy lost his patience and he decided to ask (which is a measurement by asking "do you like me?"). Unfortunately, the state collapsed to $|No\rangle$! Well, I have not told you, *if a state collapses to one of the basis states, it will not change no matter how many times you measure it again*. That is the problem. Once the answer is "No," it is "No" forever. If you measure (ask) again, the answer is still "No." Poor boy!

Fortunately, the boy did not collapse psychologically. He turned his attention to study and understood quantum mechanics better. He realized that in a quantum system, even if the state has collapsed to a basis state of a certain basis, when you measure in another basis, it will collapse to a basis state of the new basis, which is a superposition of the old basis states. For him, the old basis was formed by the $|Yes\rangle$ or $|No\rangle$ basis states of the *affection basis*. Let us make this clear and rewrite Eq. (3.1) as

$$|State\ of\ the\ girl\rangle = \alpha\,|Yes_L\rangle + \beta\,|No_L\rangle \tag{3.2}$$

where $|Yes_L\rangle$ and $|No_L\rangle$ are the answers to the question "Do you like me?". The girl's state can also be expressed as a superposition of the basis states on another basis. For example, this new basis can have the basis states $|Yes\rangle$ and $|No\rangle$ to the question "Do you want to work on a project with me?". Let us label the basis states as $|Yes_P\rangle$ and $|No_P\rangle$. Thus,

$$|State\ of\ the\ girl\ after\ saying\ No\rangle = |No_L\rangle = \gamma\,|Yes_P\rangle + \delta\,|No_P\rangle \tag{3.3}$$

This is because a girl who does not like him may or may not want to work on a project with him. So $|No_L\rangle$ can coexist with $|Yes_P\rangle$ or $|No_P\rangle$. They are not orthogonal to each other. Therefore, after a measurement in the new basis (let us call it the *project basis*, but do not confuse with the important concept of *state projection* to be discussed later), the state may collapse to $|Yes_P\rangle$. That is, the girl agrees to work on a project with him. Similarly, since $|Yes_P\rangle$ is not orthogonal to $|Yes_L\rangle$ or $|No_L\rangle$, we may have

$$|Yes_P\rangle = \epsilon\,|Yes_L\rangle + \zeta\,|No_L\rangle \tag{3.4}$$

Now what if he tried again to measure in the old basis (affection basis), i.e. ask her if she likes him, there is a chance he will get $|Yes_L\rangle$! Here is the magic. After the first time he measured (ask) in the old affection basis and got $|No_L\rangle$, no matter how many times he asks again, the answer is still "No." But if he tries to measure again on a new basis (the project basis), he might get $|Yes_P\rangle$, which is a superposition of $|Yes_L\rangle$ and $|No_L\rangle$. If he then measures again in the old basis, i.e. asking "do you like me?", he might get $|Yes_L\rangle$! ! This is possible in reality, is not that? She might be impressed by him more after the project. And they lived happily ever after.

You probably do not understand everything I mentioned. Think about this at your leisure. Let us move forward to have a more mathematical description.

Action Item There are lots of issues with this analogy. Write down what you think and later find out what is correct and what is incorrect.

3.3 Vectors in Quantum Computing

After all, *quantum computing is just a manipulation of vectors through operations in the vector space*. We prepare many vectors in a special way and manipulate them through special **quantum gates** (operations) and we get the answers we want. And as mentioned earlier, these operations are also just *rotations of vectors in hyperspaces*.

On a 2D plane (Fig. 3.1), we know a vector can be described by its x and y coordinates. For example, in Fig. 3.1, we have a vector $\vec{v}$ and it has x-component equal to 3 and y-component equal to 2. How do we represent this vector? We have the liberty to write it in any of the following forms:

$$\vec{v} = 3\hat{x} + 2\hat{y} = 3 \textit{ units of } x + 2 \textit{ units of } y = \begin{pmatrix} 3 \\ 2 \end{pmatrix} \tag{3.5}$$

where $\hat{x}$ and $\hat{y}$ are the unit vectors in the x and y directions. You see that it really does not matter how I write $\hat{x}$ and $\hat{y}$. I wrote them as "unit of x" and "unit of y" and you still understand what it is about. I even did not write them when I wrote 3 and 2 in the column form. As long as we agree that the top (bottom) of the column represents the number of x(y)-component, we still represent the vector correctly and unambiguously.

$\hat{x}$ and $\hat{y}$ are just the *basis vectors* we discussed in the previous chapter. In quantum computing, vector and state are the same. I can also write $\vec{v}$ in Dirac's bra–ket notation, as well:

$$|\vec{v}\rangle = 3\,|\hat{x}\rangle + 2\,|\hat{y}\rangle \tag{3.6}$$

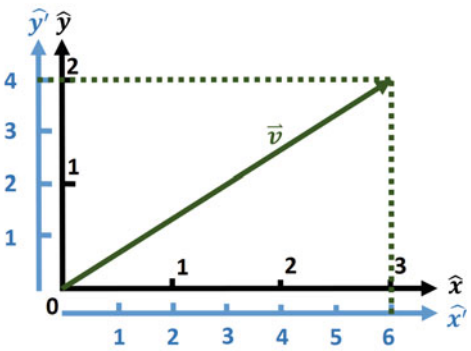


Fig. 3.1 Representations of $|\vec{v}\rangle$ in two different bases

Definitely, you can apply all the concepts we discussed earlier on vectors. I can say $\vec{v}$ is just a *superposition* of $\hat{x}$ and $\hat{y}$.

I can also choose another set of basis vectors, $\hat{x'}$ and $\hat{y'}$. $\hat{x'}$ and $\hat{y'}$ are only half of the lengths of $\hat{x}$ and $\hat{y}$, respectively. How do I represent $\vec{v}$ now? It can be written as

$$|\vec{v}\rangle = 3\,|\hat{x}\rangle + 2\,|\hat{y}\rangle = 3\left(2\left|\hat{x'}\right\rangle\right) + 2\left(2\left|\hat{y'}\right\rangle\right) = 6\left|\hat{x'}\right\rangle + 4\left|\hat{y'}\right\rangle \tag{3.7}$$

As mentioned, as long as I know I am using the basis vectors, $\hat{x'}$ and $\hat{y'}$, I can just write them in the column form,

$$\vec{v} = \begin{pmatrix} 6 \\ 4 \end{pmatrix} \tag{3.8}$$

Now comparing the column forms in Eq. (3.5) and Eq. (3.8), they are different but they represent the same vector. Therefore, it is very important to keep in mind which basis we are using. This is nothing new to you. For instance, your father asks you "how many dozens of eggs have you bought?", and you answer 1. Later your mother asks you "how many eggs have you bought?", and you answer 12. They are different numbers, but they represent exactly the same thing. They both are correct and you are just using different bases (one with basis vector = 1-egg and the other with basis vector = 12-egg).

So far, the basis and basis vectors can be visualized easily. But even if it cannot be visualized, the concept is the same. When we study quantum computing, we need to accept some concepts we do not have experience with and we also need to borrow concepts we have experience with.

If I tell you there is a physical system with "spin up" and "spin down" as the basis states (basis vectors) and even you do not know what "spin up" and "spin down" mean, you can still use your knowledge in vectors for computation. You know that a vector in this basis must be a superposition of the basis vectors and can be written as, for example,

$$|\textit{spin vector}\rangle = 3\,|\textit{spin up}\rangle + 2\,|\textit{spin down}\rangle = \begin{pmatrix} 3 \\ 2 \end{pmatrix} \tag{3.9}$$

3.4 Vector Space

All vectors on a plane can be expressed as a linear combination of the basis vectors $\hat{x}$ and $\hat{y}$ with the form in Eq. (3.5). We say that the basis vectors form or **span** that vector space. Now if I have a vector that is out of the plane, can it be expressed as a linear combination of $\hat{x}$ and $\hat{y}$? The answer is no. We say that this out-of-plane vector is in another vector space. Such a concept is very natural to us. You bought a dozen eggs. You say you have 12 1-eggs. The basis vector is 1-egg and it spans

the whole vector space of eggs. Any number of eggs can be represented as a linear combination of 1-egg (here we only have 1 basis vector). If you have a dozen apples, you cannot say you have 12 1-eggs because 1-egg does not span the vector space of apples. In English, when we disagree, we often say "we are not talking to each other" or "we are not on the same page." Basically, we are saying we do not have the same vector space and we are not in the same universe. You have your own vector space and I have my own vector space. No matter how I transform myself, I cannot understand you.

If we agree that 1-apple and 1-egg individually span their corresponding space, it is not difficult to understand that $|spin\ up\rangle$ and $|spin\ down\rangle$ span a vector space that is different from the one spanned by $\hat{x}$ and $\hat{y}$. There can be a vector called $\begin{pmatrix}3\\2\end{pmatrix}$ in the spin vector space and there can also be a vector called $\begin{pmatrix}3\\2\end{pmatrix}$ in the x–y vector space. But they are different and not talking to each other because they have different basis vectors and they are in different spaces. This is just like 12 1-eggs are different from 12 1-apples although they both are 12.

Let me give one more example. This book can be treated as a vector in a vector space spanned by the English alphabet and mathematical symbols (let us ignore the figures, tables, etc. for simplicity). And actually, any English book can be considered as a vector in the English–math vector space. But if you pick a book in Japanese, they cannot be represented as a combination of the basis vectors (i.e. "A," "a," "B," "b," $\cdots$) in this English–math vector space. So, they are in another vector space. But what if I construct a new vector space by using Japanese characters, English alphabets, and mathematical symbols as the basis vectors? Now all Japanese and English books are in this new and larger vector space.

3.5 Inner Products of Vectors

We know that vector and state are equivalent. We can represent every physical system state as a vector. We will review some of the important operations of vectors and their physical meanings. The first one is the **inner product**, which is also called the "**dot product**" or "**scalar product'**', for reasons that will be obvious soon. This is a very fundamental and critical concept. What is the physical meaning of the inner product? It is **projection**. Projection is an important concept in quantum mechanics.

In every measurement, we can say we try to do an operation to project a superposition state to one of the basis states. For example, when the boys asked (a measurement) the girl if she liked him and she answered no, the wavefunction is collapsed (projected) into the $|No_L\rangle$ basis state. Although not everyone agrees on this meaning of measurement, we adapt this concept in this book.

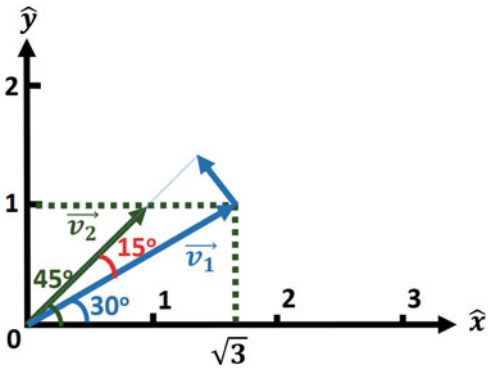


Fig. 3.2 Graphical representation of the inner product of $\vec{v}_1$ and $\vec{v}_2$. Note that the final result needs to be scaled by $|\vec{v}_2|$

Let us look at a simple example. In Fig. 3.2, there are two vectors, $\vec{v}_1$ and $\vec{v}_2$. We can represent them as

$$|\vec{v}_1\rangle = \sqrt{3}\,|\hat{x}\rangle + 1\,|\hat{y}\rangle \tag{3.10}$$

$$|\vec{v}_2\rangle = 1\,|\hat{x}\rangle + 1\,|\hat{y}\rangle \tag{3.11}$$

The angle between $\vec{v}_1$ and the x-axis (which has the same direction as $\hat{x}$) is 30°. The angle between $\vec{v}_2$ and the x-axis is 45°. So, the angle between $\vec{v}_1$ and $\vec{v}_2$ is 15°. From trigonometry, the inner product of the two vectors is

$$\vec{v}_1 \cdot \vec{v}_2 = |\vec{v}_1||\vec{v}_2| \cos 15^\circ \tag{3.12}$$

where $|\vec{v}_1|$ and $|\vec{v}_2|$ are the magnitudes (lengths) of $\vec{v}_1$ and $\vec{v}_2$, respectively. Here Dirac's bra–ket notation is not used for a reason. I will explain later. Using the Pythagorean theorem, $|\vec{v}_1| = \sqrt{\sqrt{3}^2 + 1^2} = 2$ and $|\vec{v}_2| = \sqrt{1^2 + 1^2} = \sqrt{2}$. Therefore,

$$\vec{v}_1 \cdot \vec{v}_2 = |\vec{v}_1||\vec{v}_2| \cos 15^\circ = 2\sqrt{2} \cos 15^\circ \approx 2.732 \tag{3.13}$$

This is the length if you project $\vec{v}_1$ onto $\vec{v}_2$ and scaled by $|\vec{v}_2|$. Therefore, the inner product is telling us how much the two vectors have in common after being scaled by their magnitudes (lengths). Obviously, if two vectors are from two different vector spaces, we expect their inner product to be zero because they do not have anything in common or overlap. For example, a vector in the spin space has zero inner product with the vector in the 2D space.

There is an easier way to perform the inner product. This is something you must trust me before you fully understand and you can only use it if the basis vectors are **orthonormal** (just note this down, we will discuss later). Since we can write all

vectors in the column form if the basis vectors are known, Eq. (3.10) and Eq. (3.11) become

$$|\vec{v}_1\rangle = \sqrt{3}\,|\hat{x}\rangle + 1\,|\hat{y}\rangle = \begin{pmatrix}\sqrt{3}\\ 1\end{pmatrix} \tag{3.14}$$

$$|\vec{v}_2\rangle = 1\,|\hat{x}\rangle + 1\,|\hat{y}\rangle = \begin{pmatrix}1\\ 1\end{pmatrix} \tag{3.15}$$

To perform the dot product of $\vec{v}_1 \cdot \vec{v}_2$, firstly, I will rotate the column form of the first vector anti-clockwise by 90°, so that it becomes a **row vector**. And then I will perform what we learn in matrix multiplication. I will multiply the vectors element-wise and sum them up. This is the procedure,

$$\vec{v}_1 \cdot \vec{v}_2 = \begin{pmatrix}\sqrt{3} & 1\end{pmatrix}\begin{pmatrix}1\\ 1\end{pmatrix} = \sqrt{3}\times 1 + 1 \times 1 = \sqrt{3} + 1 \approx 2.732 \tag{3.16}$$

The answer is exactly the same as in Eq. (3.13). Again, this is just how we do row and column multiplication in matrices.

In general, if $\vec{v}_1 = \begin{pmatrix}a\\ b\end{pmatrix}$ and $\vec{v}_2 = \begin{pmatrix}c\\ d\end{pmatrix}$, we have

$$\vec{v}_1 \cdot \vec{v}_2 = \begin{pmatrix}a^* & b^*\end{pmatrix}\begin{pmatrix}c\\ d\end{pmatrix} = a^* \times c + b^* \times d \tag{3.17}$$

where a^* and b^* are the **complex conjugate** of a and b, respectively. I have not been explicit. The coefficients in the linear combination of basis vectors can be **complex numbers** (if you want to review some of the properties of complex numbers, you may read the first three paragraphs in Sect. 25.2). We need to accept this as we do not have a simple direct daily experience. Although it is as weird as having $12 + 2i$ 1-apples, it should not stop us from adopting it and moving forward. The conjugate of a complex number can be obtained by negating the imaginary number. As examples, $(1 + 2i)^* = 1 - 2i$ and $(1 - 2i)^* = 1 + 2i$. What about $\sqrt{3}^*$? Yes, it is just $(\sqrt{3} + 0i)^* = \sqrt{3} - 0i = \sqrt{3}$. So the complex conjugate of a real number is just itself. That is why we have Eq. (3.16).

The inner product of a vector to itself has a special meaning, i.e. square of the length of the vector, L^2. It is not difficult to see this for a vector in a 2D plane.

$$\vec{v}_1 \cdot \vec{v}_1 = \begin{pmatrix}a^* & b^*\end{pmatrix}\begin{pmatrix}a\\ b\end{pmatrix} = a^* \times a + b^* \times b \tag{3.18}$$

where a and b are real because the vector is on a 2D plane. Therefore, $\vec{v}_1 \cdot \vec{v}_1 = a^2 + b^2 = L^2$. And by Pythagoras' theorem, this is just the square of the length of

the vector (e.g. in Fig. 3.2, $a = \sqrt{3}$, $b = 1$, and $L = 2$ for $\vec{v}_1$). This is true, as a definition, for any vectors with higher dimensions and complex coefficients.

3.6 Higher Dimensional Vector Space

When there are two basis vectors, it spans a **two-dimensional (2D) vector space**, just like $\hat{x}$ and $\hat{y}$ span a plane. If we add $\hat{z}$, the third basis vector in the height direction, they then span a **3D** space. If we have n basis vectors, I expect they will span an n-**D** space. This might be difficult to understand. But think about an ant that cannot jump and is put on a 1000 km by 1000 km plane since it was born. I did some calculations and most ants will not be able to reach the edge of the plane within their lifetime unless they invent the car. For the ant, the universe is just a 2D space. It probably will argue with us that the world is 2D and it is very difficult to understand a 3D universe. Indeed, it is also very difficult for us to tell the ant how it feels in a 3D world. So, although I cannot feel a 4-D or n-D world, from the experience of the ant, I know that a higher dimension is possible even I cannot feel it just like how I would tell the ant that the mathematical operations involving $\hat{z}$ are the same as those for $\hat{x}$ and $\hat{y}$. And I also know that many mathematical operations can be generalized to higher dimensions. Here is where again I want you to accept it if you still do not feel comfortable with it. You will understand it one day.

If you accept what I said, then in an n-D space, any vector, $\vec{V}_a$, must be a linear combination of the n basis vectors.

$$\vec{V}_a = a_0\hat{v_0} + a_1\hat{v_1} + a_2\hat{v_2} + \cdots + a_{n-1}\hat{v_{n-1}} \tag{3.19}$$

where a_0 to a_{n-1} are the n coefficients and $\hat{v_0}$ to $\hat{v_{n-1}}$ are the n basis vectors. Note that the coefficients may be complex numbers. And for another vector, $\vec{V}_b$, in the same space, we have

$$\vec{V}_b = b_0\hat{v_0} + b_1\hat{v_1} + b_2\hat{v_2} + \cdots + b_{n-1}\hat{v_{n-1}} \tag{3.20}$$

where b_0 to b_{n-1} are the n coefficients which, again, can be real or complex. The inner product or dot product of the two vectors is

$$\begin{aligned}\vec{V}_a \cdot \vec{V}_b &= \begin{pmatrix} a_0{}^* & a_1{}^* & \cdots & a_{n-1}{}^* \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \end{pmatrix} \\ &= a_0{}^* \times b_0 + a_1{}^* \times b_1 + \cdots + a_{n-1}{}^* \times b_{n-1} \end{aligned} \tag{3.21}$$

And we can write it in a more succinct form which you need to be familiarized with,

$$\vec{V}_a \cdot \vec{V}_b = \sum_{i=0}^{n-1} {a_i}^* \times b_i \tag{3.22}$$

which means that we first perform element-wise multiplications to obtain ${a_i}^* \times b_i$ (there are n multiplications from $i = 0$ to $i = n - 1$) and then sum them together.

Like the 2D case, the length of the vector, L, is defined as

$$L^2 = \vec{V}_a \cdot \vec{V}_a = \sum_{i=0}^{n-1} {a_i}^* \times a_i \tag{3.23}$$

3.7 Summary

In this chapter, we familiarized ourselves with the concept that vectors are states and vectors are linear combinations of basis vectors. We realize that the basis vectors can be something very abstract (e.g. spin up) or absurd (1-apple). The coefficients in the linear combination can be complex numbers. We also reviewed the meaning and methodology of calculating the inner product of two vectors. All these concepts apply to higher dimensions. The most important thing is that we have learned how to deduce or believe in something we do not have experience with by finding the pattern based on our daily experience. We can move forward only if we believe in these. If you still do not feel comfortable with what I said, either read it again or write it down, so you will investigate later after finishing this book. Let us do a few exercises, so I can "brainwash" you further to let you get used to the absurdity.

Problems

3.1 Google Colab and Python

We need to do a lot of matrix operations in this book. I will choose Google Colab, in which you can run Python easily. Since IBM Quantum Experience also uses Python, it will save your efforts. Please open a Google Colab account here https://colab.research.google.com/.

3.2 Vector Inner Product Using Python

We use the NumPy library to perform vector and matrix operations. To import the library, type

```
import numpy as np
```

Then let us implement Eq. (3.16)

```
v1=np.array([[np.sqrt(3)],[1]])
v2=np.array([[1],[1]])
inner_result=np.vdot(v1,v2)
print(inner_result)
```

Type Shift–Enter to let it run. Do you see the same result? Note that we use $np.vdot$ instead of $np.dot$ because we need to use complex vector dot products in quantum computing.

Try also $print(v1.shape)$, which will give you (2, 1). It means we constructed a 2×1 matrix, which is a column vector with 2 elements.

3.3 Inner Product with Complex Coefficient 1

Now, let us try $\vec{v_1} = \begin{pmatrix} i \\ 1 \end{pmatrix}$ and $\vec{v_2} = \begin{pmatrix} -i \\ 1 \end{pmatrix}$. Firstly, try to use hand calculation. Then use Python to check if it is correct. Note that in the NumPy library, you need to put i as j. For example, $5 + i$ should be entered as $5 + 1j$ (putting the "1" is necessary). Think about that before reading the code below:

```
v1=np.array([[1j],[1]])
v2=np.array([[-1j],[1]])
inner_result=np.vdot(v1,v2)
print(inner_result)
```

3.4 Inner Product with Complex Coefficient 2

Now, let us try $\vec{v_1} = \begin{pmatrix} 1 + 2i \\ 1 \end{pmatrix}$ and $\vec{v_2} = \begin{pmatrix} 1 - i \\ 1 \end{pmatrix}$.

Chapter 4
Orthonormal Basis, Bra–Ket Notation, and Measurement

4.1 Learning Outcomes

Understand that orthonormal bases and normalized vectors are used in quantum computing; Have a deeper understanding of Bra–Ket notation; Understand the meaning of superposition coefficient in measurement.

4.2 Orthonormal Basis

We have seen many different types of **basis**. Each basis has its set of **basis vectors**. For example, $\hat{x}$ and $\hat{y}$ **span** a plane and they form the basis of a 2D plane. Any vector on the 2D plane can be represented by a **linear combination** of the basis vectors, $\hat{x}$ and $\hat{y}$ (e.g. Eq. (3.6)). Let me remind you again, while we are talking about vectors, it is equivalent to the states of a quantum system. *That means that states are vectors and basis states are basis vectors.* If we add $\hat{z}$, then $\hat{x}$, $\hat{y}$, and $\hat{z}$ are the basis vectors and they span the 3D space. Any 3D vector can be represented by a linear combination of $\hat{x}$, $\hat{y}$, and $\hat{z}$. And I told you, even we do not know how an n-D space looks like, we can deduce that in an n-D space, there are n basis vectors and they span the whole n-D space. The basis vectors can be labeled as $\hat{v_0}$ to $\hat{v_{n-1}}$ and any vector in this space can be represented as a linear combination of $\hat{v_0}$ to $\hat{v_{n-1}}$ (e.g. Eq. (3.19)).

I also mentioned some weird bases. For example, the space of the number of eggs has a basis vector of 1-egg. I also told you something not exactly correct and too romantic. For example, the basis vectors of the space of a girl's state can be $|Yes_P\rangle$ and $|No_P\rangle$ or $|Yes_L\rangle$ and $|No_L\rangle$. If you feel comfortable with these, I think you will accept the fact that an electron's spin state (vector) must be a linear combination of some basis states (vectors). And again, I want you to believe me that one of the possible sets of basis vectors are $|spin\ up\rangle$ and $|spin\ down\rangle$ (e.g. Eq. (3.9)). Now

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_4

can you tell me what is the dimension of the spin space? Yes, it is 2D because it has 2 basis vectors. So, while the spin space sounds fancy, it is actually as simple as a 2D real space in some sense. However, this is again not exact. In a real 2D space, the coefficients in the linear combinations are all real numbers (e.g. Eq. (3.14)). In a spin space, the coefficient can be complex numbers (e.g. Eq. (3.17)). We do not have experience in the spin space. So, often, we need to use the real 2D space to understand some basics and then deduce what happens in the spin space. This is just like that we try to understand 1D, 2D, and 3D space properties first and then deduce how an n-D space will look like.

One type of bases is of particular importance and we will use it throughout this book. It is called the **orthonormal basis**. They are also important because they make our life much easier by making the derivation much simpler. Let me just tell you the definition first. An orthonormal basis is defined as

$$\hat{v_i} \cdot \hat{v_j} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases} = \delta_{ij} \tag{4.1}$$

where $\hat{v_i}$ and $\hat{v_j}$ are any two basis vectors. i and j can be any number from 0 to $n-1$, if it is an n-D space. For example, if it is our regular 3D space, we may set $\hat{v_0} = x$, $\hat{v_1} = y$, and $\hat{v_2} = z$. So, $\hat{v_i}$ and $\hat{v_j}$ can be x and y, or x and z, or y and z, or y and x, or z and x, or z and y. $\hat{v_i}$ and $\hat{v_j}$ can also be x and x, or y and y, or z and z because it is ok to have $i = j$. Equation (4.1) says that if the inner product of every basis vector with itself is one (i.e. it is **normalized**, with $|L| = 1$ in Eq. (3.23)) and the inner product with other basis vector is zero (i.e. they are **orthogonal**), then this set of basis vectors and the basis formed by this set of basis vectors are **orthonormal** (*orthogonal and normalized*). Remember what is the meaning of the inner product of two vectors? It tells us if they have anything in common and if they contain a part of each other and if there is any "projection" of one vector onto the other. Therefore, when the basis vectors are orthonormal, it means all of them have no common part.

Before moving forward, let us also remind ourselves of the meaning of δ_{ij}. This is called the **Kronecker delta**. It is just a notation. It means that if $i = j$, it is one. If $i \neq j$, then, it is zero.

It is easy to think in our 2D real space. What is $\hat{x} \cdot \hat{y}$? It is zero because they are perpendicular to each other. $\hat{x}$ does not contain any part of $\hat{y}$ and $\hat{y}$ does not contain any part of $\hat{x}$. So if you want to date a boy, you should tell him that you have a non-zero inner product with him. When $i = j$, e.g. $\hat{v_0} \cdot \hat{v_0} = \hat{x} \cdot \hat{x}$, $\hat{x}$ projects onto itself. Does it have a common part with itself? And of course, it contains 100% of itself. As shown in Fig. 4.1, we might have two different bases for a 2D plane. The basis formed by basis vectors $\hat{x}$ and $\hat{y}$ is orthonormal and that formed by $\hat{x}$ and $\hat{y'}$ is not. Why the basis formed by $\hat{x}$ and $\hat{y'}$ is not? This is just based on our knowledge in trigonometry that the angle between $\hat{x}$ and $\hat{y'}$ is 45°. And by recalling that $\vec{v_1} \cdot \vec{v_2} = |\vec{v_1}||\vec{v_2}| \cos\theta$, where θ is the angle between the two vectors (e.g. Eq. (3.12)). Since $\cos 45°$ is non-zero, this gives us a non-zero inner product like in

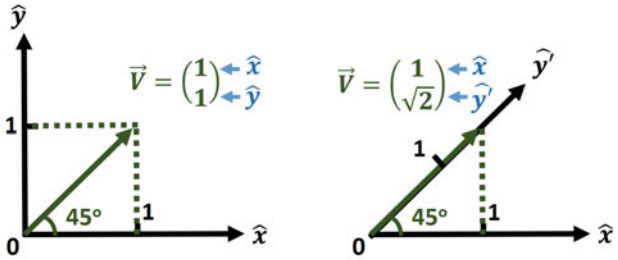


Fig. 4.1 Two different sets of basis vectors, orthonormal (left) and non-orthonormal (right)

Eq. (3.12). Any vector on the 2D plane can be formed by a linear combination of either set of basis vectors. For example, $\vec{V}$ can be represented as

$$\left|\vec{V}\right\rangle = 1\left|\hat{x}\right\rangle + 1\left|\hat{y}\right\rangle = 1\left|\hat{x}\right\rangle + \sqrt{2}\left|\hat{y'}\right\rangle \tag{4.2}$$

The $\hat{x}$ and $\hat{y'}$ basis is very difficult to deal with. In quantum computing, we will always use an orthonormal basis. What if we are given a basis that is not orthonormal? We will try to transform it into orthonormal. There are methods to do this such as the Gram-Schmidt process. But we will not spend time on this as this is not essential for our understanding of quantum computing.

Remember I told you how to do inner product using row and column forms (Eqs. (3.14)–(3.17)) and this is only valid if you are working on an orthonormal basis? Since $\hat{x} = \begin{pmatrix}1\\0\end{pmatrix}$ and $\hat{y} = \begin{pmatrix}0\\1\end{pmatrix}$, we can check if $\hat{x}$ and $\hat{y}$ are orthogonal by observing that their inner product is

$$\hat{x} \cdot \hat{y} = \begin{pmatrix}1 & 0\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix} = 1 \times 0 + 0 \times 1 = 0 \tag{4.3}$$

Note that we write the vectors in column form by agreeing that we are using the $\hat{x}$ and $\hat{y}$ basis vectors. Since this is an orthonormal basis, such operation is valid. And indeed, $\hat{x} \cdot \hat{y} = 0$, which does not contradict our initial assumption that $\hat{x}$ and $\hat{y}$ are orthogonal.

Can I do the same thing for $\hat{x}$ and $\hat{y'}$? The answer is no. If you try to repeat what we did in Eq. (4.3), by using the basis of $\hat{x}$ and $\hat{y'}$, you also get 0. Note that $\hat{y'} = \begin{pmatrix}0\\1\end{pmatrix}$ in the $\hat{x}$ and $\hat{y'}$ basis. But it is wrong as $\hat{x}$ and $\hat{y'}$ are not orthogonal and the answer contradicts the assumption that $\hat{x}$ and $\hat{y'}$ are not orthogonal because we are using the wrong mathematics. If this is still not very clear to you, that is fine. But you must remind yourself in the rest of the book that we will only work on orthonormal bases.

4.3 Bra–Ket Notation

We have talked about **ket** already.

For every vector $\vec{V}_a$ (or this is just the quantum state), it really does not matter how I write it as long as it is clear. I can represent it as $\vec{V}_a$, "Vector a," $\left|\vec{V}_a\right\rangle$, $|V_a\rangle$, $|a\rangle$, $|vector\ a\rangle$, $|that\ vector\ I\ just\ mentioned\ to\ you\rangle$, etc.

If we write it in the form of $|\ \rangle$, then it is Dirac's bra–ket notation. $|\ \rangle$ is just a vector and we can perform any vector operations on it (and we have been doing this "gradually" in the last few chapters). For example, for $\vec{V}_a = \vec{V}_b + \vec{V}_c$, we can write as

$$\begin{aligned} \left|\vec{V}_a\right\rangle &= \left|\vec{V}_b\right\rangle + \left|\vec{V}_c\right\rangle \\ |a\rangle &= |b\rangle + |c\rangle \end{aligned} \tag{4.4}$$

I hope you can get used to this notation. It is just a notation. And I tell you this is a great notation. If you trust me, just accept it as your best friend. Let us try another example. How do you represent $\vec{V}_a = m\vec{V}_b + n\vec{V}_c$ in bra–ket notation? This should be

$$\begin{aligned} \left|\vec{V}_a\right\rangle &= m\left|\vec{V}_b\right\rangle + n\left|\vec{V}_c\right\rangle \\ |a\rangle &= m\,|b\rangle + n\,|c\rangle \end{aligned} \tag{4.5}$$

This is just telling us that *"vector a is m times of vector b plus n times of vector c."*

How about the **bra**? It still represents a vector and it has the form of $\langle\ |$. Again, I can represent $\vec{V}_\alpha = \vec{V}_\beta + \vec{V}_\gamma$ in the bra form as

$$\begin{aligned} \left\langle\vec{V}_\alpha\right| &= \left\langle\vec{V}_\beta\right| + \left\langle\vec{V}_\gamma\right| \\ \langle\alpha| &= \langle\beta| + \langle\gamma| \end{aligned} \tag{4.6}$$

And similarly, represent $\vec{V}_\alpha = k\vec{V}_\beta + l\vec{V}_\gamma$ as

$$\begin{aligned} \left\langle\vec{V}_\alpha\right| &= k\left\langle\vec{V}_\beta\right| + l\left\langle\vec{V}_\gamma\right| \\ \langle\alpha| &= k\,\langle\beta| + l\,\langle\gamma| \end{aligned} \tag{4.7}$$

Why do we need two forms of representation? Actually, the set of vectors in the *bra-representation* is the same set of vectors in the *ket-representation* and they form a **dual space**. Each vector in the ket space can find a corresponding vector in the bra space. The method to find the correspondence is by performing a transpose operation and then complex conjugation. If you forgot, a transpose operation is

just to swap the row and column indices in a matrix. For a column vector, we can imagine it as rotating the column anti-clockwise to form the row vector in a transpose operation. It is easier to understand by looking at an example. If a vector $\vec{v_1}$ in the ket space is

$$|v_1\rangle = |1\rangle = \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 1+2i \\ i \end{pmatrix} \tag{4.8}$$

the transpose of $\vec{v_1}$ is $\begin{pmatrix} a & b \end{pmatrix}$. After performing complex conjugation, it becomes $\begin{pmatrix} a^* & b^* \end{pmatrix}$, and we can write

$$\langle v_1| = \langle 1| = \begin{pmatrix} a^* & b^* \end{pmatrix} = \begin{pmatrix} 1-2i & -i \end{pmatrix} \tag{4.9}$$

Here you see that although the vector in the bra space is still a vector, we write it in a row form and it has different coefficients from the ket vector if the coefficients are complex numbers. Its relationship to its corresponding vector in the ket space is by conjugating the coefficients.

Do you recall how we do the inner product of two vectors? Let me repeat (Eq. (3.17))

$$\vec{v_1} \cdot \vec{v_2} = \begin{pmatrix} a^* & b^* \end{pmatrix} \begin{pmatrix} c \\ d \end{pmatrix} = a^* \times c + b^* \times d \tag{4.10}$$

What have you discovered? You see that this is just the same as

$$\langle v_1| \ |v_2\rangle = \begin{pmatrix} a^* & b^* \end{pmatrix} \begin{pmatrix} c \\ d \end{pmatrix} = a^* \times c + b^* \times d \tag{4.11}$$

We can further simplify the notation to $\langle v_1|v_2\rangle$. Now based on how it is written, I believe you understand why this is called **bra–ket notation**, which is just a little modification from the word "bracket." So, the inner product of two vectors is just the product between the first vector in the bra-representation and the second one in the ket notation. From now on, we will only use bra–ket and row/column representations. You will find them to be extremely useful and intuitive.

Let us look at another example. This is a 3D example.

$$|v_a\rangle = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} \quad \text{and} \quad |v_b\rangle = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix}$$

$$\langle v_b|v_a\rangle = \begin{pmatrix} b_0^* & b_1^* & b_2^* \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = a_0 b_0^* + a_1 b_1^* + a_2 b_2^* \tag{4.12}$$

Here, we transpose $|v_b\rangle$ into a row vector for the bra-representation and then conjugate its coefficients to perform inner product with $|v_a\rangle$, which is still in the ket-representation.

4.4 Quantum Mechanical State, Superposition, and Measurement

We should not get lost in the vector operations. Let us try to have a deeper understanding of the quantum mechanical state before moving forward to more complex vector operations. What is a vector? Every vector is a state of a physical system. Let us look at an example. I have a physical system called electron spin. And you are given that it has 2 basis vectors, which are $|spin\ up\rangle$ and $|spin\ down\rangle$. And now, without ambiguity, I can call them $|\uparrow\rangle$ and $|\downarrow\rangle$ also. First of all, the basis vectors themselves are vectors and also quantum mechanical states. Just like in the 2D plane case, $\left|\hat{x}\right\rangle$ and $\left|\hat{y}\right\rangle$ are also vectors. Of course, they have a special identity called basis vectors because we choose them to represent other vectors as linear combinations of them. We can write the basis states as follows:

$$|\uparrow\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{and} \quad |\downarrow\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{4.13}$$

Since they are the basis states, we really *do not need* to know what they exactly are. This is just like in the old time we know everything is formed by atoms (the basis) and we did not know what formed atoms, and we still can describe chemistry through our knowledge of the atoms. Or as mentioned earlier, the number of eggs (a vector) is a linear combination of the basis vector 1-egg and we do not really need to know what is the meaning of 1-egg and if an egg has a shell and yolk. We just take them for granted and there is nothing wrong to treat basis as an assumption or given. We take it for granted that $\left|\hat{x}\right\rangle$ and $\left|\hat{y}\right\rangle$ are the basis vectors of a 2D plane because we have experience in this. But it is still true that $\left|\hat{x}\right\rangle$ and $\left|\hat{y}\right\rangle$ are the basis vectors of a 2D plane even we are the creatures that cannot move. Therefore, it is not difficult to accept the fact that $|\uparrow\rangle$ and $|\downarrow\rangle$ can form the basis vectors of a spin space as long as you trust the physicist who told us that this is true. Therefore, any spin state of an electron can be written as

$$|\Psi\rangle = \alpha\,|\uparrow\rangle + \beta\,|\downarrow\rangle \tag{4.14}$$

Again, we say that any electron spin state is a superposition, which we have been calling "linear combination," of $|\uparrow\rangle$ and $|\downarrow\rangle$. In 2D space, what is the meaning of the coefficients in the superposition or linear combination? If a vector is $|\vec{v}_1\rangle = 1\left|\hat{x}\right\rangle + 1\left|\hat{y}\right\rangle$, we say that it has 1 x-component and 1 y-component. But in quantum mechanics, it has a deeper meaning.

In Eq. (4.14), the spin state $|\Psi\rangle$ is a linear combination of $|\uparrow\rangle$ and $|\downarrow\rangle$. It is still true that it has $\alpha\,|\uparrow\rangle$ component and $\beta\,|\downarrow\rangle$ component. But we know that $|\uparrow\rangle$ and $|\downarrow\rangle$ must be orthogonal and cannot exist at the same time with 100% certainty. Let us recall what we learned about the $|Head\rangle$ and $|Tail\rangle$ of a spinning coin (again not to confuse this classical spin with electron spin). They are orthogonal to each other. To know which state it is, we need to perform a **measurement**. In the spinning coin analogy, I just need to stop it from spinning and the superposition state will collapse to one of the basis states as we mentioned. How about the electron spin? What we can do is to apply a magnetic field in the vertical direction and measure its energy. We will not discuss this in detail. But as a superposition state, we know that when we perform a measurement, the spin state $|\Psi\rangle$ will collapse to one of the basis states, namely, $|\uparrow\rangle$ or $|\downarrow\rangle$. This is completely random. However, the probability of collapsing to $|\Psi\rangle$ is *proportional to* $|\alpha|^2$ and $|\beta|^2$, respectively. And this is the important meaning of the coefficients in the superposition. This is due to the **Born rule** which is an essential part of the **Copenhagen interpretation**.

Let us make this more general. For a state $|\Psi\rangle$ in an n-D space, which can be represented as

$$|\Psi\rangle = a_0\,|V_0\rangle + a_1\,|V_1\rangle + \cdots + a_{n-1}\,|V_{n-1}\rangle = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} \tag{4.15}$$

where $|V_i\rangle$ are the basis states, when we perform a measurement, $|\Psi\rangle$ will randomly collapse to one of the basis states, e.g. $\left|V_j\right\rangle$ with a probability *proportional to* $|a_j|^2$.

Let us recall the meaning of $|a_j|^2$. This is just the square of the magnitude of a_j. It equals $a_j^* a_j$, where a_j^* is the complex conjugate of a_j. For example, if $a_j = 1+i$, then $a_j^* = 1-i$ and $a_j^* a_j = (1+i)(1-i) = 1\times 1 + i\times 1 + 1\times -i + i\times -i = 2$. And indeed, the length of a_j is $\sqrt{2}$ and thus $|a_j|^2 = 2$.

If a vector is **normalized**, which will be used throughout quantum computing, it means it has a unit length. The length of a vector is defined by the inner product of itself (Eq. (3.23)). For example, the inner product of $|V\rangle$ is

$$\begin{aligned} \langle V|V\rangle &= \left(a_0^*\ a_1^* \cdots a_{n-1}^*\right) \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} \\ &= a_0 a_0^* + a_1 a_1^* + \cdots + a_{n-1} a_{n-1}^* \\ &= \sum_{i=0}^{n-1} a_i a_i^* = \sum_{i=0}^{n-1} |a_i|^2 = 1 \end{aligned} \tag{4.16}$$

$\langle V|V\rangle = 1$ because it is normalized. What is the probability that it will collapse to the basis state $|V_i\rangle$? Yes, it is proportional to $|a_i|^2$. But we know that it must collapse to one of the basis states during measurement. Then it means that the probability that it will collapse to $|V_i\rangle$, i.e. P_i, is just $|a_i|^2$ (the proportional factor is 1). That is why normalized vectors are used for convenience. In summary, if a vector (or state) is normalized, the probability that it will collapse to the basis state $|V_i\rangle$ equals $|a_i|^2$

$$P_i = a_i a_i^* = |a_i|^2 \tag{4.17}$$

If you are given a vector, $|V\rangle$, as in the numerical example after Eq. (4.15), that is not normalized, we need to normalize it first. Since the definition of the normalized vector is that the summation of the square of the magnitude of coefficient is equal to one, we can just divide the vector by this sum and it will be normalized.

$$\begin{aligned} \left|V'\right\rangle &= \frac{|V\rangle}{\sqrt{\sum_{i=0}^{n-1}|a_i|^2}} \\ &= \frac{a_0}{\sqrt{\sum_{i=0}^{n-1}|a_i|^2}}|V_0\rangle + \frac{a_1}{\sqrt{\sum_{i=0}^{n-1}|a_i|^2}}|V_1\rangle + \cdots + \frac{a_{n-1}}{\sqrt{\sum_{i=0}^{n-1}|a_i|^2}}|V_{n-1}\rangle \end{aligned} \tag{4.18}$$

It looks complicated, but $\sqrt{\sum_{i=0}^{n-1}|a_i|^2}$ is just the length of $|V\rangle$, i.e. $L = \sqrt{\langle V|V\rangle}$. So we can easily prove that $\left|V'\right\rangle$ is normalized (with length = 1) as the following:

$$\left\langle V'\middle|V'\right\rangle = \frac{\langle V|}{\sqrt{\sum_{i=0}^{n-1}|a_i|^2}}\frac{|V\rangle}{\sqrt{\sum_{i=0}^{n-1}|a_i|^2}} = \frac{\langle V|V\rangle}{LL} = 1 \tag{4.19}$$

4.5 Summary

If you forget everything else, you must at least agree and remember that we need to use orthonormal bases and normalized vectors in quantum computing. The bra–ket notation is a very important and useful notation and I hope you start getting used to it and love it. You will see its power soon. Finally, the probability of a normalized quantum state collapsing to a certain basis state is equal to the square of the magnitude of the corresponding coefficient. Let us do some exercise to get used to bra–ket notation and normalization.

Problems

4.1 Normalization by Hand
Normalize vector $|\vec{v_1}\rangle = 2\,|\uparrow\rangle + 1\,|\downarrow\rangle$ by hand.

4.2 Normalization Using Python
Check using Google Colab. You can use the scikit-learn library to do so:

```
import numpy as np
v1=np.array([[2],[1]])
print(v1)
from sklearn.preprocessing import normalize
v2 = v1/np.linalg.norm(v1)
print(v2)
```

4.3 Normalization by Hand with Complex Coefficient
Normalize vector $|\vec{v_1}\rangle = 2\,|\uparrow\rangle + (1+i)\,|\downarrow\rangle$ by hand.

4.4 Normalization Using Python with Complex Coefficient
Verify with Google Colab. Remember you need to use $1 + 1\,j$ to represent $1 + i$.

4.5 Vectors on a 2D Plane
On a 2D plane, draw the vectors $\left|\vec{x'}\right\rangle = \frac{1}{\sqrt{2}}(|\hat{x}\rangle + |\hat{y}\rangle)$ and $\left|\vec{y'}\right\rangle = \frac{1}{\sqrt{2}}(|\hat{x}\rangle - |\hat{y}\rangle)$

Write the column forms of $\left|\vec{x'}\right\rangle$ and $\left|\vec{y'}\right\rangle$. Show that $\left|\vec{x'}\right\rangle$ and $\left|\vec{y'}\right\rangle$ are orthonormal, i.e. $\left\langle\vec{x'}\middle|\vec{x'}\right\rangle = \left\langle\vec{y'}\middle|\vec{y'}\right\rangle = 1$ and $\left\langle\vec{x'}\middle|\vec{y'}\right\rangle = \left\langle\vec{y'}\middle|\vec{x'}\right\rangle = 0$. You have derived a new orthonormal basis for a 2D plane. But this is not surprising, right? As $\left|\vec{x'}\right\rangle$ and $\left|\vec{y'}\right\rangle$ can be obtained by just rotating $|\vec{x}\rangle$ and $|\vec{y}\rangle$ by 45°.

Chapter 5
Changing Basis, Uncertainty Principle, and Bra–Ket Operations

5.1 Learning Outcomes

Know how to represent vectors in different bases; Have a feeling on the origin of the uncertainty principle; Be more familiar with Bra–Ket operations.

5.2 Basis and Uncertainty Principle

We talked about the electron spin space which can be spanned by the $|\uparrow\rangle$ and $|\downarrow\rangle$ basis vectors *if we choose to do so*. It is important to realize that $|\uparrow\rangle$ and $|\downarrow\rangle$ are *not* real space direction vector like $\hat{z}$ and $-\hat{z}$. They are not direction vectors. It just happens that since we can measure them by applying a magnetic field in the z-direction, we call them "up" and "down," but they do not have a direct relationship with the up and down vectors in the real space. They are just as abstract as my earlier analogy that the basis states of the girl's state are $|Yes_L\rangle$ and $|No_L\rangle$. This is very important and even if you do not fully understand at this moment, you must remind yourself that $|\uparrow\rangle$ and $|\downarrow\rangle$ are not $\hat{z}$ and $-\hat{z}$. Therefore, you will not have the question of why $|\uparrow\rangle \neq -|\downarrow\rangle$ and why $|\uparrow\rangle$ and $|\downarrow\rangle$ are orthogonal.

Now, I can also choose another basis with basis vectors as $|spin\ left\rangle$ and $|spin\ right\rangle$. For simplicity, I write them as $|\leftarrow\rangle$ and $|\rightarrow\rangle$ like what we did for $|\uparrow\rangle$ and $|\downarrow\rangle$. Again, these two vectors have nothing to do with $\hat{x}$ and $-\hat{x}$. $|\leftarrow\rangle$ and $|\rightarrow\rangle$ are *not* real space direction vectors. The reason I call it left and right is just because, to distinguish them, I need to apply a magnetic field in the x-direction to perform a measurement.

Here, I need to say again that $|\uparrow\rangle$ / $|\downarrow\rangle$ ($|\leftarrow\rangle$ / $|\rightarrow\rangle$) have no direct relationship to the up/down (left/right) directions in our real 3D space. They are called so just because, in the experiment, they are the states we will get when we apply a vertical

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_5

(horizontal) magnetic field during measurements. You need to believe me for now if you are not familiar with the spin qubit physics.

We will also rename the basis vectors, which is completely legitimate, as they are just names. We set $|\uparrow\rangle = |0\rangle$, $|\downarrow\rangle = |1\rangle$, $|\leftarrow\rangle = |+\rangle$, and $|\rightarrow\rangle = |-\rangle$. If your name is William, your friends often call you Bill. As long as it does not cause confusion, calling $|\uparrow\rangle = |0\rangle$ is legitimate. Therefore, the $|\uparrow\rangle / |\downarrow\rangle$ basis becomes the $|0\rangle / |1\rangle$ basis and the $|\leftarrow\rangle / |\rightarrow\rangle$ basis becomes the $|+\rangle / |-\rangle$ basis. This change is to reinforce the basis concept and also to make the notation consistent with the theories we will introduce in the following chapters.

While I keep reminding you that the $|0\rangle / |1\rangle$ basis vectors or the $|+\rangle / |-\rangle$ basis vectors in electron spin space are not direction vectors, unfortunately, I need to borrow the direction vectors to introduce some important concepts because we are very familiar with the real space and the 2D real space is easy to understand. Therefore, let us pretend $|0\rangle$ and $|1\rangle$ are just the horizontal and vertical unit vectors as shown in Fig. 5.1. Note that, for now, I will not call horizontal and vertical unit vectors in a real 2D space as $\hat{x}$ and $\hat{y}$, respectively, because this can cause confusion with the spin directions. Let us call the horizontal and vertical unit vectors in the real 2D space in Fig. 5.1 as $\hat{\zeta}$ and $\hat{\eta}$, respectively.

Representing the spin space using the 2D real space is incomplete or even wrong. For example, any vector in the 2D real space only has real coefficients when they are represented as the linear combination of $\hat{\zeta}$ and $\hat{\eta}$, but the vector in the spin space could have complex coefficients when they are represented as the linear combination of $|0\rangle$ and $|1\rangle$, i.e. $|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, where α and β may be complex numbers. For illustration, let us *assume* α and β are real, and then we can treat them as vectors on a 2D plane.

On the 2D plane, I can also have another set of basis vectors which are just equivalent to rotating the $|0\rangle / |1\rangle$ basis vectors clockwise by 45°. We call them $|+\rangle / |-\rangle$ basis as shown in Fig. 5.1.

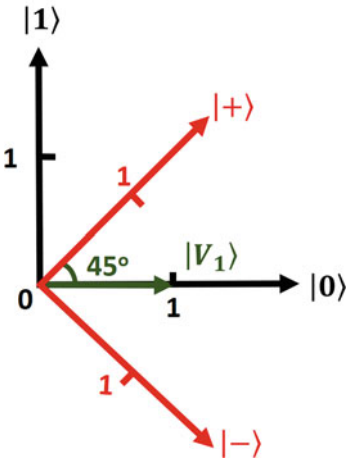


Fig. 5.1 The two bases ($|0\rangle / |1\rangle$ and $|+\rangle / |-\rangle$) represented in a fictitious 2D real space (which is *incorrect*). This is only done for illustration purpose

How do I represent vector $|V_1\rangle$ in the $|0\rangle$ / $|1\rangle$ basis. By observation, I see that it has a horizontal component of 1 and a vertical component of 0. Therefore, we have

$$|V_1\rangle = 1\,|0\rangle + 0\,|1\rangle = |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \tag{5.1}$$

We need to make sure everyone knows that we are using the $|0\rangle$ / $|1\rangle$ basis when we write it in the column form so that everyone knows it has 1 part of $|0\rangle$ and none, i.e. 0 part, of $|1\rangle$. If we perform a measurement on $|V_1\rangle$ by applying a magnetic field in the $\hat{z}$ direction, we definitely will get $|0\rangle$ because $|\alpha|^2 = 1$. *Here let us take a pause to clarify something very important.* The $\hat{z}$ direction is the direction in real space. It is not perpendicular to the fictitious 2D plane we have borrowed to illustrate the concept in Fig. 5.1. Make sure you understand this. If we measure again, it is still $|0\rangle$. So, when measuring in the $|0\rangle$ / $|1\rangle$ basis, we have 100% certainty that the electron is spinning up, i.e. at state $|0\rangle$ or $|\uparrow\rangle$. For the same state, how is it represented in the $|+\rangle$ / $|-\rangle$ basis? By using some trigonometry in Fig. 5.1, we found that it is

$$|V_1\rangle = \frac{1}{\sqrt{2}}(|+\rangle + |-\rangle) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix} \tag{5.2}$$

For the same vector, when it is represented in different bases, it has different column forms. This reaffirms the importance of making it clear which basis is being used. The column form in Eq. (5.2) tells us that the vector has $\frac{1}{\sqrt{2}}$ part of $|+\rangle$ and also $\frac{1}{\sqrt{2}}$ part of $|-\rangle$. Now, if I want to measure the state using the $|+\rangle$ / $|-\rangle$ basis, I will apply a magnetic field in the $\hat{x}$ direction (again, $\hat{x}$ is in the x direction in real space not the horizontal direction in Fig. 5.1). What do I get? There are 50% of chance I will get $|+\rangle$ (spin left) and 50% of chance I will get $|-\rangle$ (spin right) because $|\alpha|^2 = |\beta|^2 = (\frac{1}{\sqrt{2}})^2 = 0.5$.

For $|V_1\rangle$, I can obtain its state with 100% certainty (i.e. 100% $|0\rangle$), but it is completely uncertain to me whether it is spinning left or spinning right in the $|+\rangle$ / $|-\rangle$ basis (chances are 50/50). This means that I cannot measure the spin up/down and spin left/right with 100% accuracy *simultaneously*. This is called the **Heisenberg Uncertainty Principle**.

We will have a deeper mathematical explanation of the Heisenberg Uncertainty Principle later. This is related to the fact that the two measurements are **incompatible**. But it is amazing that we can appreciate this principle by just using simple vector manipulations on a 2D plane using the concepts of *superposition* and *collapse of the state*.

Similarly, in quantum mechanics, the measurement of momentum and position is incompatible. Therefore, we cannot measure the momentum and position of a particle precisely at the same time. Note that measurement of the momentum means to collapse the state into a momentum basis vector, which happens to be the momentum in our daily experience, and measurement of position means to

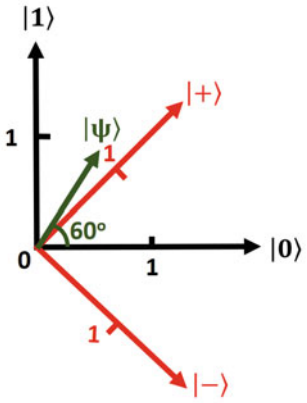


Fig. 5.2 $|\Psi\rangle$ in the fictitious 2D real space (which is *incorrect*)

collapse the particle state into a position basis vector, which is the position in our daily experience. It will be difficult to show this like what we did for $|+\rangle / |-\rangle$ and $|0\rangle / |1\rangle$ on a 2D plane. But our understanding of the uncertainty principle using $|+\rangle / |-\rangle$ and $|0\rangle / |1\rangle$ makes us more convinced that there is a valid reason why we cannot measure the momentum and position with 100% certainty at the same time. It should also be noted that the more certain we have on one measurement, the less we have on another measurement.

Does this sound familiar? Yes, the boy we mentioned earlier did try to measure the girl's state in the $|Yes_P\rangle$ and $|No_P\rangle$ basis after he obtained $|No_L\rangle$ in the first measurement. The girl's state is just like $|V_1\rangle$. $|0\rangle$ is just like the $|No_L\rangle$ state. And the $|+\rangle / |-\rangle$ basis is just like the $|Yes_P\rangle / |No_P\rangle$ basis.

Let us try an example.

Example 5.1 (Changing Basis Example) In Fig. 5.2, $|\Psi\rangle$ can be represented in the $|0\rangle / |1\rangle$ basis as the following because its angle to the horizontal axis is 60°:

$$|\Psi\rangle = \frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle \tag{5.3}$$

How do we represent it in the $|+\rangle / |-\rangle$ basis? We can use trigonometry, but this is not useful to us when the problem cannot be mapped onto a fictitious 2D plane. The trick is to represent $|0\rangle / |1\rangle$ basis vectors as the linear combinations of $|+\rangle / |-\rangle$ basis vectors and then use substitution.

Firstly, we can represent $|0\rangle$ and $|1\rangle$ as

$$|0\rangle = \frac{1}{\sqrt{2}}|+\rangle + \frac{1}{\sqrt{2}}|-\rangle \quad \text{and} \quad |1\rangle = \frac{1}{\sqrt{2}}|+\rangle - \frac{1}{\sqrt{2}}|-\rangle \tag{5.4}$$

To get Eq. (5.4), i.e. the relationship between two sets of basis vectors, we need to use trigonometry in this case. However, this is often given in quantum mechanics

through other means (e.g. Laws of Physics). Then we substitute Eq. (5.4) into Eq. (5.3) and obtain

$$\begin{aligned} |\Psi\rangle &= \frac{1}{2}\left(\frac{1}{\sqrt{2}}|+\rangle + \frac{1}{\sqrt{2}}|-\rangle\right) + \frac{\sqrt{3}}{2}\left(\frac{1}{\sqrt{2}}|+\rangle - \frac{1}{\sqrt{2}}|-\rangle\right) \\ &= \frac{1}{\sqrt{2}}\left(\frac{1}{2} + \frac{\sqrt{3}}{2}\right)|+\rangle + \frac{1}{\sqrt{2}}\left(\frac{1}{2} - \frac{\sqrt{3}}{2}\right)|-\rangle \end{aligned} \tag{5.5}$$

By doing so, we have changed the basis from $|0\rangle / |1\rangle$ to $|+\rangle / |-\rangle$. The coefficients are $\frac{1}{\sqrt{2}}(\frac{1}{2} + \frac{\sqrt{3}}{2})$ and $\frac{1}{\sqrt{2}}(\frac{1}{2} - \frac{\sqrt{3}}{2})$ for $|+\rangle$ and $|-\rangle$, respectively. In other words, we have represented the same quantum state as a superposition of the $|+\rangle / |-\rangle$ basis states instead of the $|0\rangle / |1\rangle$ basis states.

It is also instructive to represent the $|+\rangle / |-\rangle$ as a linear combination of $|0\rangle / |1\rangle$. Please try it. The answer will be given in the next chapter.

5.3 More Bra–ket Operations

Now let us go back to the bra–ket operation. I would like to tell you some rules just like how an elementary school teacher teaches the students addition and multiplication. The rules are definitions and cannot be proved. Therefore, do not think too hard if you do not know how to "derive." Please just trust me and use it. On the other hand, you should try to see and feel the logic within and compare it to your experience in a real 3D space.

First of all, as mentioned earlier, for any vector in the ket space, there is a corresponding vector in the bra space. This is called **Dual Correspondence**. This is just like everyone will find their image in a mirror. Of course, this image is not the same as that person. We discussed earlier already that it needs to go through transpose and conjugate operations. In the following, the symbol $\Leftrightarrow$ means direct correspondence.

$$\langle a| \Leftrightarrow |a\rangle \tag{5.6}$$

For example, if $|\alpha\rangle = \begin{pmatrix} i \\ 1 \end{pmatrix}$, then $\langle\alpha| = \begin{pmatrix} -i & 1 \end{pmatrix}$.

We have learned that we can scale a vector $|\alpha\rangle$ by a constant (may be complex number) c. And this is still a vector, $c\,|\alpha\rangle$; for example, $2\begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \end{pmatrix}$. What is the corresponding vector in the bra space? It is equivalent to multiplying the complex conjugate of c, i.e. c^*, to the bra version of $|\alpha\rangle$, i.e. $\langle\alpha|$.

$$c^*\,\langle a| \Leftrightarrow c\,|a\rangle \tag{5.7}$$

For example, if $c = i$, $c\,|\alpha\rangle = i\begin{pmatrix} i \\ 1 \end{pmatrix} = \begin{pmatrix} -1 \\ i \end{pmatrix}$. Its corresponding vector in the bra space is $\begin{pmatrix} -1 & -i \end{pmatrix}$. And indeed $c^*\,\langle a| = -i\begin{pmatrix} -i & 1 \end{pmatrix} = \begin{pmatrix} -1 & -i \end{pmatrix}$, which is the bra version of $c\,|\alpha\rangle$ by definition (Eq. (4.9)). Therefore, this rule makes sense. Or in the mirror image analogy, when you lift your left arm, the image lifts its right arm. "Left" and "right" behave like c and c^* in some sense.

Similarly, if we represent a vector, $|\gamma\rangle$, as a linear combination of other vectors in the ket space, we can find the corresponding vector in the bra space as

$$\langle\gamma| = a^*\,\langle\alpha| + b^*\,\langle\beta| \Leftrightarrow |\gamma\rangle = a\,|\alpha\rangle + b\,|\beta\rangle \tag{5.8}$$

There is another very useful rule and it is about the inner product. First of all, is an inner product of two vectors a vector or a scalar (pure number)? It is a scalar. Therefore, there is no dual space for it (or we may say its dual space is just itself). However, there is a rule related to the dual space for describing the relationship between $\langle\alpha|\beta\rangle$ and $\langle\beta|\alpha\rangle$. $\langle\alpha|\beta\rangle$ means the projection of vector β on vector α and $\langle\beta|\alpha\rangle$ means the projection of vector α on vector β. Or $\langle\alpha|\beta\rangle$ means the dot product of bra space α with ket space β and $\langle\beta|\alpha\rangle$ means the dot product of bra space β with ket space α. Are they the same? Not necessarily and they obey this rule.

$$\langle\alpha|\beta\rangle = \langle\beta|\alpha\rangle^* \tag{5.9}$$

Again, note that they are numbers, not vectors. Equation (5.9) is not difficult to prove. Let us use a 2D example to demonstrate. If $|\alpha\rangle = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix}$ and $|\beta\rangle = \begin{pmatrix} b_0 \\ b_1 \end{pmatrix}$, then $\langle\alpha| = \begin{pmatrix} a_0^* & a_1^* \end{pmatrix}$ and $\langle\beta| = \begin{pmatrix} b_0^* & b_1^* \end{pmatrix}$, and therefore,

$$\begin{aligned} \langle\alpha|\beta\rangle &= a_0^* b_0 + a_1^* b_1 \\ \langle\beta|\alpha\rangle &= b_0^* a_0 + b_1^* a_1 \end{aligned} \tag{5.10}$$

And clearly, they differ by a conjugate operation only. Therefore, Eq. (5.9) makes sense.

This is a very important equation and I hope you can memorize it. There is a special case when $\alpha = \beta$. This means $\langle\alpha|\alpha\rangle = \langle\alpha|\alpha\rangle^*$. This is just the inner product of vector α to itself and it says that the result equals its complex conjugate. Since only a real number equals its complex conjugate, it means the *inner product of any vector to itself must be real*. This is also expected because the length of a vector is better to be real.

With the new knowledge in bra–ket operations, let us revisit the concept of orthonormality and normalization one more time.

If two vectors are orthogonal to each other, their inner product is zero.

$$\langle\alpha|\beta\rangle = 0 \tag{5.11}$$

If a vector is normalized, its inner product is 1.

$$\langle \alpha | \alpha \rangle = 1 \tag{5.12}$$

To normalize a vector, Eq. (4.18) can be succinctly written as

$$\left| V' \right\rangle = \frac{|V\rangle}{\sqrt{\langle V|V\rangle}} \tag{5.13}$$

5.4 Summary

The most important thing we learned in this chapter is that vectors can be represented on different bases. These different bases might not be compatible with each other. Therefore, we cannot measure both of them accurately at the same time. We used a fictitious 2D plane to illustrate this uncertainty principle. We also learned some important bra–ket operations and started seeing the power of bra–ket notation in simplifying equations.

5.5 More About Python

I would like to spend some more time discussing how to set up vectors and perform operations in Python. This is instructive because if you understand the details, you understand vector and thus matrix operations better.

To implement vector $\vec{v_1} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$, we can use numpy library as mentioned in the earlier exercise. For example,

```
import numpy as np
v1=np.array([[1j],[2]])
print(v1)
```

It gives this column vector.

```
[[0.+1.j]
[2.+0.j]]
```

If you print its shape, you have

```
print(v1.shape)
(2, 1)
```

I can normalize it by using the function "np.linalg.norm,"

```
v2 = v1/np.linalg.norm(v1)
print(v2)
[[0. +0.4472136j]
[0.89442719+0.j ]]
```

To prove it is normalized, I can perform vector dot product

```
print(np.vdot(v2,v2))
(0.9999999999999999+0j)
```

You see that the vector dot product does the transpose and also conjugation implicitly for you already. We can also create the bra version of v2 by using transpose and conjugation,

```
v2_t=np.conjugate(np.transpose(v2))
print(v2_t)
print(v2_t.shape)
[[0. -0.4472136j 0.89442719-0.j ]]
(1, 2)
```

It gives a row vector of 1×2, and compared to v2, you do see it has a negative sign in front of 0.4472136j.

Now, I cannot use the vector dot product, np.vdot. I need to use the regular dot product, np.dot to perform normality check.

```
print(np.dot(v2_t,v2))
[[1.+0.j]]
```

You see we get the same result. Although the data types are different, we will ignore them for now.

Problems

5.1 Basis Change by Hand

Vector $|\vec{v}_1\rangle = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ is in the $|+\rangle$ / $|-\rangle$ basis. Represent it in the $|0\rangle$ / $|1\rangle$ basis. Try to find $|+\rangle$ and $|-\rangle$ in terms of $|0\rangle$ and $|1\rangle$ and perform the substitution. To check your answer, convert it back to $|+\rangle$ / $|-\rangle$ using the equations introduced in this chapter and you should get back $\begin{pmatrix} 1 \\ 2 \end{pmatrix}$.

5.2 Vector Normalization by Hand and Using Python

Normalize $|\vec{v}_1\rangle = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ in $|+\rangle$ / $|-\rangle$ basis. Use Google Colab to verify your answer.

5.3 Measurement Probability

For the $|\vec{v}_1\rangle$ in the last question, what is the probability to find $|+\rangle$ in the $|+\rangle$ / $|-\rangle$ basis measurement?

5.4 Vector Normalization by Hand and Using Python 2

Normalize $|\vec{v}_1\rangle$ in the $|0\rangle$ / $|1\rangle$ basis. What is the probability to find $|0\rangle$ in the $|0\rangle$ / $|1\rangle$ basis measurement? Use Google Colab to verify your answer.

5.5 Basis Change and Measurement

If I get $|1\rangle$ in the measurement in Problem 5.4, what is the probability of getting $|+\rangle$ if I measure again in the $|+\rangle$ / $|-\rangle$ basis measurement? (hint: after getting $|1\rangle$, how to represent it in the $|+\rangle$ / $|-\rangle$ basis?)

Chapter 6
Observables, Operators, Eigenvectors, and Eigenvalues

6.1 Learning Outcomes

Understand the connections between operator matrices and observables; Able to find the eigenvalues and eigenvectors of a matrix.

6.2 Observables and Operators

Any quantum mechanical state is a vector and it is a linear combination of the basis vectors in a given basis. How do we find the basis? What is a reasonable basis? Let me give you a simplified and *incomplete* view first. This is related to the so-called **observables**. Observables are something we can observe and measure. For example, the position, momentum, and spin of an electron are its observables. We say that if we measure a state, it will collapse to one of the basis states corresponding to an observable. Then it means that the basis state (vector) must be a value we can understand classically. For example, the position is an observable for an electron. How do we measure the position of an electron? We can check where it strikes the electron detector. When we measure its position (detected by the detector), its state collapses to one of the basis states, which is just a position vector and may be $(x, y, z) = (1.1, 0, -10)$. We can call this basis vector any name such as $|1.1, 0, -10\rangle$. We can also measure the spin in the $\hat{z}$ direction. Since we are talking about measurement, we need to be specific on how we will do the measurement. To measure in the $\hat{z}$ direction, I need to apply a magnetic field in the $-\hat{z}$ direction, and through some processes (e.g. photon emission but let us do not worry about the details now), we measure its energy. If it has lower energy, I know it collapses to the $|\uparrow\rangle$ state. If it has higher energy, I know it collapses to the $|\downarrow\rangle$ state. I can also measure its spin in the $\hat{x}$ direction, but this time I need to apply the magnetic field in

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_6

the $\hat{x}$ direction. If the energy level measured is low, I know it collapses to the $|\leftarrow\rangle$ state. If the energy level measured is high, I know it collapses to the $|\rightarrow\rangle$ state.

We see that every measurement corresponds to an observable and each measurement result must be a basis state corresponding to that observable/measurement. Therefore, it is very normal to choose the possible measurement results to form the basis states. For the position observable, the basis contains the basis states of (x, y, z), and since x, y, z can take any values, it has an infinite number of basis states. This looks very weird, but this is exactly what quantum mechanics is about. The electron can be in a superposition of all the basis states. That means it can be inside your body or at the edge of the universe at the same time (but with different coefficients and thus probabilities). When you measure it, it collapses to one basis state, i.e. one position. That is why people argue that the Moon is only there when you look at it (a measurement). Otherwise, it is everywhere in the universe. Of course, this does not really happen because, in a macroscopic object, coherence is usually destroyed and quantum mechanics converges to classical mechanics. So do not be too romantic.

Similarly, for the measurement of the electron energy under a magnetic field in the $-\hat{z}$ direction, we can use $|\uparrow\rangle$ and $|\downarrow\rangle$ as the basis vectors. Do we have more basis vectors for this? No, because physics tells us that there are only two energy levels we can measure using this method. This is also true if we try to measure the energies when we apply a magnetic field in the $\hat{x}$ direction and we will form a basis using $|\leftarrow\rangle$ and $|\rightarrow\rangle$. Note that for the same electron state, we can represent it as a linear combination of $|\uparrow\rangle$ and $|\downarrow\rangle$ or $|\leftarrow\rangle$ and $|\rightarrow\rangle$ as discussed earlier.

In summary, *a basis formed by the basis states corresponding to an observable is a convenient choice in the formulation of the quantum computing problems.*

There are two formalisms in quantum mechanics. One is the **wave quantum mechanics** by Erwin Schrödinger and you probably have heard of **Schrödinger Equation**, which describes how a quantum state evolves. In this formalism, we describe the physics by differential equations and differential operators. However, they are not convenient. And in certain types of problems, they are not intuitive. We will not use this approach here. The other formalism is the **matrix quantum mechanics**. It was proposed independently by Werner Heisenberg. It has exactly the same content as the wave quantum mechanics including Schrödinger equation, but all in matrix form. This is very useful in quantum computing. *It is particularly powerful when we combine it with Dirac's bra–ket notation*. We have been using matrix quantum mechanics since the beginning. *We represent quantum mechanical states as vectors*. Of course, for each vector in either row or column form, we need to specify clearly which basis we are using.

But how do we describe the evolution of a quantum state? This is represented by an **operator**. An operator *operates* on a vector. It transforms the vector from one to another. You probably have learned computer graphics in which we use matrices to transform an image which is nothing but just a collection of real space vectors. The transformation can be stretching and rotation, etc. In quantum computing, if you remember, we say that it is nothing but just the rotation of vectors (states) in the hyperspace. In the matrix quantum mechanics, *operators are represented as*

matrix. We will discuss how to construct an operator later. Now, let us understand its properties first.

6.3 Eigenvalues and Eigenvectors

There is a special type of operators that correspond to some observables/measurements. For example, the operator corresponding to the measurement of electron spin in the $\hat{z}$ direction is $given$ as

$$\boldsymbol{\sigma}_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{6.1}$$

This is not exactly the operator for measuring the energy and is correct up to a scaling factor. But for simplicity, we will use it in most of the following text. Again, this is given by the physicist, so please take it as it is now, but you will see that it makes sense. Firstly, this is a 2×2 matrix. It has two rows and two columns. Let us recall that we index the elements in the matrix by using row-column numbering and it starts from 0 to $n - 1$ if there are n rows and columns. For example, the element (0, 1) is in row 0 and column 1 and it is 0. You need to make sure you are familiar with this convention. Let us try to operate on the two possible measurable outputs, i.e. $|\uparrow\rangle = |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|\downarrow\rangle = |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

$$\begin{aligned} \boldsymbol{\sigma}_z |\uparrow\rangle &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \times 1 + 0 \times 0 \\ 0 \times 1 + -1 \times 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |\uparrow\rangle \\ \boldsymbol{\sigma}_z |\downarrow\rangle &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \times 0 + 0 \times 1 \\ 0 \times 0 + -1 \times 1 \end{pmatrix} = - \begin{pmatrix} 0 \\ 1 \end{pmatrix} = - |\downarrow\rangle \end{aligned} \tag{6.2}$$

We see that both $|\uparrow\rangle$ and $|\downarrow\rangle$ are unchanged when acted by $\boldsymbol{\sigma}_z$, with $|\uparrow\rangle$ and $|\downarrow\rangle$ scaled by 1 and -1, respectively. Note that while it scales $|\downarrow\rangle$ by -1, the vector is still the same in quantum mechanics. Scaling a vector by -1 is just the same as prepending a phase factor to that vector which will be discussed later.

Equation (6.2) tells us that $|\uparrow\rangle$ and $|\downarrow\rangle$ are the **eigenstates** of the operator $\boldsymbol{\sigma}_z$ with **eigenspins** 1 and -1, respectively. We may also say $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ are the **eigenvectors** of the matrix $\boldsymbol{\sigma}_z$ with **eigenvalues** 1 and -1, respectively. Both sentences are equivalent.

Note that when we say $\boldsymbol{\sigma}_z$ corresponds to the measurement of the spin in the $\hat{z}$ direction, it does *NOT* mean that we use $\boldsymbol{\sigma}_z$ to perform the measurement. A measurement is a collapsing process of the wavefunction and is completely random.

It only means that the eigenvalues of $\boldsymbol{\sigma}_z$ are those possible values that can be measured.

In general, for an operator A *which does not need to correspond to any observables/measurements*, if it operates on a vector $|a\rangle$ and results in $a\,|a\rangle$, then we say $|a\rangle$ and a are the eigenvector and eigenvalue of A, respectively.

$$\boldsymbol{A}\,|a\rangle = a\,|a\rangle \tag{6.3}$$

Note that I call it $|a\rangle$ just to show that it is a vector with eigenvalue a. If you remember, we can call it any name. I can call it $\boldsymbol{A}\,|Sam\rangle = a\,|Sam\rangle$ as well. That also means when we apply the operator to its eigenvectors, it only *scales* the vector by its eigenvalues. It does not perform other transformations like rotation. This is why the eigenvector is so special.

What if I operate $\boldsymbol{\sigma}_z$ on $|\leftarrow\rangle$ and $|\rightarrow\rangle$ (i.e. $|+\rangle$ and $|-\rangle$)? Since we are using matrix representation, as we have been emphasizing, we need to make sure which basis we want to use so that we know what components we have in the column vector (e.g. Fig. 5.1). Let us still use the $|\uparrow\rangle$ and $|\downarrow\rangle$ (i.e. $|0\rangle$ and $|1\rangle$) as the basis. Then we should represent $|\leftarrow\rangle$ and $|\rightarrow\rangle$ as the linear combinations of $|0\rangle$ and $|1\rangle$. We did that in the exercise of the previous chapters already. Or, from Fig. 5.1, we see that

$$\begin{aligned} |+\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} \\ |-\rangle &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix} \end{aligned} \tag{6.4}$$

Example 6.1 Show that $|+\rangle$ and $|-\rangle$ are not the eigenvectors of $\boldsymbol{\sigma}_z$.

$$\begin{aligned} \boldsymbol{\sigma}_z\,|+\rangle &= \begin{pmatrix}1 & 0\\0 & -1\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\times 1 + 0\times 1\\0\times 1 + -1\times 1\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix} = |-\rangle \\ \boldsymbol{\sigma}_z\,|-\rangle &= \begin{pmatrix}1 & 0\\0 & -1\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\times 1 + 0\times -1\\0\times 1 + -1\times -1\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} = |+\rangle \end{aligned} \tag{6.5}$$

You see that $\boldsymbol{\sigma}_z$ transforms, and in other words **rotates** $|+\rangle$ to $|-\rangle$ and vice versa. Therefore, $|+\rangle$ and $|-\rangle$ are not the eigenvalues of $\boldsymbol{\sigma}_z$.

We know $|+\rangle$ and $|-\rangle$ are the eigenvectors of the operator corresponding to the measurement of electron spin in the $\hat{x}$ direction. Let us call it $\boldsymbol{\sigma_x}$. What is the matrix form of it then? This can be found by using the definition of Eq. (6.3). It is lengthy

but I hope you can follow it closely because this will reinforce the understanding of several critical concepts. We can write

$$\begin{aligned}
\boldsymbol{\sigma_x}\,|+\rangle &= \begin{pmatrix} a & b \\ c & d \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} a \times 1 + b \times 1 \\ c \times 1 + d \times 1 \end{pmatrix} \\
&= \frac{1}{\sqrt{2}} \begin{pmatrix} a+b \\ c+d \end{pmatrix} = \frac{\alpha}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \alpha\,|+\rangle \\
\boldsymbol{\sigma_x}\,|-\rangle &= \begin{pmatrix} a & b \\ c & d \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} a \times 1 + b \times -1 \\ c \times 1 + d \times -1 \end{pmatrix} \\
&= \frac{1}{\sqrt{2}} \begin{pmatrix} a-b \\ c-d \end{pmatrix} = \frac{\beta}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \beta\,|-\rangle
\end{aligned} \tag{6.6}$$

Here, $\boldsymbol{\sigma_x}$ is assumed to be $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ and we need to find a, b, c, and d, which are complex numbers. The first equation assumes $|+\rangle$ is the eigenvector of $\boldsymbol{\sigma_x}$ with eigenvalue α and the second equation assumes $|-\rangle$ is the eigenvector of $\boldsymbol{\sigma_x}$ with eigenvalue β. Again, why we or why can we represent $|+\rangle$ as $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ and $|-\rangle$ as $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix}$? This is because we are using the $|0\rangle$ / $|1\rangle$ basis, so I can use Eq. (6.4). Equating the entries of the last two terms, we get four equations,

$$\begin{aligned}
a + b = \alpha \quad \text{and} \quad c + d = \alpha \\
a - b = \beta \quad \text{and} \quad c - d = -\beta
\end{aligned} \tag{6.7}$$

This is not enough to solve for a, b, c, d, α, and β. But physics tells us that the two eigenvalues must be 1 and -1. This is because, in the empty space, there is no difference between up/down and left/right (we call this *isotropic*). If measuring in the $\hat{z}$ direction we get either 1 or -1, we should expect to get 1 or -1 also if we measure in the $\hat{x}$ direction. Therefore, we can set $\alpha = 1$ and $\beta = -1$. There are some important properties for $\boldsymbol{\sigma_x}$ to obey to help us find a, b, c, and d, but we will only discuss later. For now, you can check that $a = d = 0$ and $b = c = 1$ are valid solutions. Therefore,

$$\boldsymbol{\sigma_x} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{6.8}$$

Now I want you to ask yourselves one more time. Which basis are we using when we write $\boldsymbol{\sigma_x} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$? Correct, it is the $|0\rangle$ / $|1\rangle$ basis. If we write in $|+\rangle$ / $|-\rangle$ basis, how would it look like? It will be

$$\boldsymbol{\sigma_{x,|+\rangle/|-\rangle}} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{6.9}$$

Do you still remember in Eqs. (3.6) and (3.7) we also write the same vector with different column representations in different bases? This is the same for operators. I actually can write Eq. (6.9) without a second thought. Why? Again, this is because the space is isotropic. If we use the basis formed by the eigenvectors of $\boldsymbol{\sigma_x}$ (i.e. $|+\rangle$ / $|-\rangle$), then it is exactly the same as $\boldsymbol{\sigma_z}$ when $|0\rangle$ / $|1\rangle$ is used as the basis. Please take a pause and try your best to understand. This is very important to understand quantum computing algorithms.

6.4 Eigenvalues and Eigenvectors Finding and Phase Factor

Finally, let us practice how to find eigenvectors and eigenvalues. Previously, you were given the eigenvectors and eigenvalues of $\boldsymbol{\sigma_x}$ and you found $\boldsymbol{\sigma_x}$. Now assume $\boldsymbol{\sigma_x}$ is given as in Eq. (6.8); how do we find its eigenvalues and eigenvectors?

Again we set up the equations as in Eq. (6.6) and assume the eigenvector to be $\begin{pmatrix} e \\ f \end{pmatrix}$ and the eigenvalue to be γ. We have

$$\begin{aligned} \boldsymbol{\sigma_x}\,|eigenvector\rangle &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} e \\ f \end{pmatrix} = \begin{pmatrix} 0\times e + 1\times f \\ 1\times e + 0\times f \end{pmatrix} = \begin{pmatrix} f \\ e \end{pmatrix} \\ &= \gamma\,|eigenvector\rangle = \gamma\begin{pmatrix} e \\ f \end{pmatrix} \end{aligned} \tag{6.10}$$

Therefore,

$$\begin{aligned} f &= \gamma e \\ e &= \gamma f \end{aligned} \tag{6.11}$$

Substituting the second equation into the first one, we obtain $\gamma^2 = 1$. Therefore, the eigenvalues, γ, are ± 1. This is the same as given in the previous part.

Remember only normalized vectors are meaningful because the sum of the probabilities of collapsing into each state needs to be 1? Therefore, we need $|e|^2 + |f|^2 = 1$.

When $\gamma = 1$, from Eq. (6.11) , $e = f$, and therefore $|e|^2+|f|^2 = |e|^2+|e|^2 = 1$, and $e = f = \frac{1}{\sqrt{2}}$ is a solution.

When $\gamma = -1$, from Eq. (6.11) , $e = -f$, and therefore $|e|^2 + |f|^2 = |e|^2 + |-e|^2 = 1$, and $e = -f = \frac{1}{\sqrt{2}}$ is a solution.

Therefore, the corresponding eigenvectors are $\frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}$ (i.e. $|+\rangle$) and $\frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix}$ (i.e. $|-\rangle$), respectively.

Since e and f may be complex numbers, there are actually infinite numbers of solutions (eigenvectors). However, every solution is just different from $|+\rangle$ and $|-\rangle$ by a **phase factor**. Phase factor is very important in quantum computing which is used to achieve **interference**. A phase factor is a complex number with a magnitude equal to 1. Therefore, it can be written as $e^{i\theta}$, where θ is the phase. As a reminder, it can be expanded as

$$e^{i\theta} = \cos\theta + i\sin\theta \tag{6.12}$$

However, a phase factor applying to a standalone state has no physical significance. This is because all physical observables are related to the inner products of vectors and the phase factor will disappear. We will discuss more later, but let us have an initial feeling on it.

What is the inner product of any vector $|v\rangle$ to itself? Yes, it is $\langle v|v\rangle$. What if $|v\rangle$ is scaled by a phase factor $e^{i\theta}$? It becomes $e^{i\theta}|v\rangle$. What is the inner product of this new vector to itself? From Eq. (5.7), the bra version of the new vector becomes $(e^{i\theta})^* \langle v| = e^{-i\theta}\langle v|$. Therefore the inner product is $e^{i\theta}|v\rangle\, e^{-i\theta}\langle v| = \langle v|v\rangle$ because $e^{i\theta}e^{-i\theta} = 1$. Therefore, adding a phase factor does not affect the inner product.

6.5 Remarks on Eigenvectors, Basis, and Measurement

It is important to point out that if $|v\rangle$ is an eigenvector of $\boldsymbol{A}$, then $c|v\rangle$, which means $|v\rangle$ is scaled by any complex number c, is also an eigenvector of $\boldsymbol{A}$. This is trivial because

$$\begin{aligned}\boldsymbol{A}|v\rangle &= \alpha|v\rangle\\ c\boldsymbol{A}|v\rangle &= c\alpha|v\rangle\\ \boldsymbol{A}(c|v\rangle) &= \alpha(c|v\rangle)\end{aligned} \tag{6.13}$$

which still satisfies the definition of eigenvector in Eq. (6.3). This also explains why it is sufficient to only work with normalized vectors.

For matrix $\boldsymbol{A}$, is it possible to have two different eigenvectors, $|v\rangle$ and $|u\rangle$, which are *not* a scalar scaling of the others ($|v\rangle \neq c\,|u\rangle$), having the same eigenvalue, α? i.e.

$$\begin{aligned} \boldsymbol{A}\,|v\rangle &= \alpha\,|v\rangle \\ \boldsymbol{A}\,|u\rangle &= \alpha\,|u\rangle \end{aligned} \tag{6.14}$$

Yes, it is possible and, in such case, the eigenvectors (i.e. quantum eigenstates) are **degenerated**. Such situation is called **degeneration**. And if this happens, any linear combination of these two vectors, $|\Psi\rangle = c\,|v\rangle + d\,|u\rangle$, is also an eigenvector of the matrix and we say the two eigenvectors span an **eigenspace** of the system. It can be shown easily.

$$\begin{aligned} \boldsymbol{A}\,|\Psi\rangle &= \boldsymbol{A}(c\,|v\rangle + d\,|u\rangle) \\ &= c\boldsymbol{A}\,|v\rangle + d\boldsymbol{A}\,|u\rangle \\ &= c\alpha\,|v\rangle + d\alpha\,|u\rangle \\ &= \alpha(c\,|v\rangle + d\,|u\rangle) \\ &= \alpha\,|\Psi\rangle \end{aligned} \tag{6.15}$$

We will not use the concept of degeneracy in this book. But I think this is something very important that you should be aware of.

The matrix corresponding to the energy observable has a very important role in the matrix form of the Schrödinger equation. Usually, it is the so-called the **Hamiltonian** that determines how a state (vector) evolves (transforms or rotates) with time. For example, $\boldsymbol{\sigma_x}$ and $\boldsymbol{\sigma_z}$ play an important role in rotating the spin of an electron.

Finally, I want to emphasize again that, for a matrix corresponding to an observable, it does not perform a measurement when you apply the matrix to a vector. As you already seen in Eq. (6.5), it does not collapse $|-\rangle$ to either $|0\rangle$ or $|1\rangle$. By applying a matrix corresponding to an observable to a state (vector), we only rotate that vector.

6.6 Summary

Operators are represented as matrices and they determine the evolution of a quantum state. The matrices rotate the state in the hyperspace. However, they only scale and do not rotate a special group of vectors, namely the eigenvectors of the matrices. We usually use the observable eigenvectors to form the basis in quantum computing for convenience. And by definition, their eigenvalues are the classical measurable values in our daily lives. Therefore, it is important to know the definition of eigenvectors and eigenvalues and the methodologies to find them. Let us practice more.

Problems

6.1 Finding Eigenvectors by Hand

This is a tedious exercise. But I want you to go through to practice the skill of finding eigenvectors. We can also measure the electron spin in $\hat{y}$ direction by setting up an external magnetic field in the $\hat{y}$ direction. It is given that the corresponding operator is $\boldsymbol{\sigma}_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$. Again we are using the $|0\rangle$ / $|1\rangle$ basis. Find the eigenvalues and eigenvectors in the $|0\rangle$ / $|1\rangle$ basis.

6.2 Finding Eigenvectors Using Python

Use Google Colab to check your answer. What you need is the function to find the eigenvectors and eigenvalues. The following code first builds the matrix and then find the eigenvectors and eigenvalues:

```
import numpy as np
sigmay=np.array([[0,-1j],[1j,0]])
print(sigmay)
eigenvalues, eigenvectors = np.linalg.eig(sigmay)
print(eigenvalues)
print(eigenvectors)
```

Chapter 7
Pauli Spin Matrices, Adjoint Matrix, and Hermitian Matrix

7.1 Learning Outcomes

Be familiar with Pauli matrices and Pauli vector and their properties; Be familiar with matrix-vector multiplications; Understand the importance of adjoint and Hermitian matrices.

7.2 Pauli Spin Matrices

Any quantum mechanical state is a vector. I repeated this important concept many times already. We can decompose the state (i.e. the vector) into a linear combination of some basis vectors that span the space of interest. In this process, the basis states are usually given. It can be as simple as the position vectors that we understand well or as abstract as $|\uparrow\rangle$ and $|\downarrow\rangle$ that we never have had experience with. It can be as complicated as $|Alive\rangle$ and $|Dead\rangle$ in Schrödinger's cat, which is even not understood well in philosophy. The good thing is that once we have taken the basis states for granted and we are sure they are **complete** (i.e. it spans the space of interest) and obey certain properties, we do not need to understand what they are at all. We can just use them and may call them any names. This is something I would like you to remember whenever you feel confused with the physical meaning of the basis vectors. Just like in a classical computer, you still can be an expert in computing algorithms even you do not know how the "0" and "1" are physically implemented.

However, some bases are more convenient and meaningful than others. For example, the eigenvectors of the matrices corresponding to observables are commonly used as the basis states (see the previous chapter). This is because after all, we need to retrieve information from the quantum computer through measurement and the states must collapse to one of the basis states, which is one of the eigenstates of the

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_7

matrices corresponding to the observables. In a quantum computer, we commonly use observables with 2 eigenstates, which means it spans a 2D space (again, not the 2D space on your table). This is because it is easier to implement quantum computers using two-level systems. That is the reason we still use "bit" in the qubit, as bit stands for "binary digit" like in classical computing. It is possible to use 3-level systems for quantum computing too, but this is less practical. One of the common two-level systems is the electron spin. We say that to perform a spin measurement, we need to apply an external magnetic field. The magnetic field can be applied in the $\hat{x}$, $\hat{y}$, and $\hat{z}$ directions. Each of the setups and measurements corresponds to a matrix in matrix quantum mechanics. They are called **Pauli Spin Matrices**,

$$\boldsymbol{\sigma}_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{7.1}$$

$$\boldsymbol{\sigma}_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{7.2}$$

$$\boldsymbol{\sigma}_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{7.3}$$

Again, I need to emphasize that the matrices are the representations of the operators in the matrix quantum mechanics. The operators are NOT for measurement as a measurement is a random process. It just happens that these matrices have eigenvalues equal to the possible measurement outcomes (see the previous chapter for more details). We have derived their basis vectors earlier. But I need to ask you again when they are expressed in Eqs. (7.1)–(7.3), which basis are we using? Yes, it is the $|0\rangle$ / $|1\rangle$ basis, i.e. the eigenvectors of $\boldsymbol{\sigma}_z$. This is a question you always need to ask yourself when performing quantum computing. Another question you need to keep asking yourself is to identify the symbols in the equations and algorithms as scalars (real or complex), vectors (then you know it is a state), or matrices (then you know it is an operator).

Paul matrices have some important properties. First of all,

$$\boldsymbol{\sigma}_x^2 = \boldsymbol{\sigma}_y^2 = \boldsymbol{\sigma}_z^2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \boldsymbol{I} \tag{7.4}$$

This seems to be very simple. However, it needs to be very clear what exactly each symbol and operation means. Do not jump over because if you cannot do this manually, you will get lost in quantum computing algorithms. The purpose of this book is to make sure you know the nuts and bolts of quantum computing instead of some superficial and handwaving arguments.

Example 7.1 Show that $\boldsymbol{\sigma}_x^2 = \boldsymbol{I}$.

What is $\boldsymbol{\sigma}_x^2$? It is $\boldsymbol{\sigma}_x\boldsymbol{\sigma}_x$, which is a matrix–matrix multiplication. Therefore, $\boldsymbol{\sigma}_x^2$ is neither a scalar nor a vector but a matrix. This is how it is calculated:

$$\begin{aligned}\boldsymbol{\sigma}_x^2 = \boldsymbol{\sigma}_x\boldsymbol{\sigma}_x &= \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\\ &= \begin{pmatrix}0\times 0+1\times 1 & 0\times 1+1\times 0\\ 1\times 0+0\times 1 & 1\times 1+0\times 0\end{pmatrix} = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} = \boldsymbol{I}\end{aligned} \tag{7.5}$$

What is $\boldsymbol{I}$? Clearly, it is a matrix and $\boldsymbol{I} = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}$. This is called the **identity matrix**. It is just like the "1" in scalar multiplications. Any scalar multiplied by "1" is unchanged and any matrix ($\boldsymbol{M}$) or vector ($|v\rangle$) multiplied by $\boldsymbol{I}$ is unchanged. For example,

$$\boldsymbol{I}\,|v\rangle = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\begin{pmatrix}a\\ b\end{pmatrix} = \begin{pmatrix}a\\ b\end{pmatrix} = |v\rangle \tag{7.6}$$

$$\boldsymbol{I}\boldsymbol{M} = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\begin{pmatrix}c & d\\ e & f\end{pmatrix} = \begin{pmatrix}c & d\\ e & f\end{pmatrix} = \boldsymbol{M} \tag{7.7}$$

7.3 Commutation and Anti-commutation

There are other two important properties with Pauli matrices. The first one is about their commutation property.

$$\boldsymbol{\sigma}_i\boldsymbol{\sigma}_j \neq \boldsymbol{\sigma}_j\boldsymbol{\sigma}_i \text{ if } i \neq j \tag{7.8}$$

where i and j can be x, y, or z.

We can also write it as

$$\left[\boldsymbol{\sigma}_i, \boldsymbol{\sigma}_j\right] = \boldsymbol{\sigma}_i\boldsymbol{\sigma}_j - \boldsymbol{\sigma}_j\boldsymbol{\sigma}_i \neq 0 \text{ if } i \neq j \tag{7.9}$$

where $\left[\boldsymbol{\sigma}_i, \boldsymbol{\sigma}_j\right]$ is called the **commutator bracket** or just **commutator**.

For example, if $i = x$ and $j = z$, then it means $\boldsymbol{\sigma}_x\boldsymbol{\sigma}_z \neq \boldsymbol{\sigma}_z\boldsymbol{\sigma}_x$ and we say the Pauli matrices **do not commute** with each other. Matrix is an operator that rotates a vector in the hyperspace. We apply the matrix to a vector from the left. For example, $\boldsymbol{\sigma}_x\boldsymbol{\sigma}_z\,|v\rangle$ means we first apply $\boldsymbol{\sigma}_z$ to rotate $|v\rangle$ and then $\boldsymbol{\sigma}_x$ to further rotate the *rotated vector*. Therefore, the equation tells us that applying $\boldsymbol{\sigma}_z$ and then $\boldsymbol{\sigma}_x$ is *different* from applying $\boldsymbol{\sigma}_x$ and then $\boldsymbol{\sigma}_z$. In our daily life, some operations commute with each other. For example, stepping forward and laterally (without turning) commute with each other. If you step forward 10 steps and then step laterally 10

steps (without turning), it is the same as stepping laterally 10 steps and then stepping forward 10 steps. However, if turning is an operation, then stepping forward 10 steps and then turning right is different from turning right first and then stepping forward 10 steps. So turning right and stepping forward do not commute with each other. Non-commuting operators also mean their measurements are **incompatible**. This is the **Heisenberg Uncertainty Principle** due to which we cannot measure x and z direction spins precisely at the same time (we do not explain the reason here but just believe me).

Let us keep practicing matrix multiplication and improving our understanding of the symbols by showing $\boldsymbol{\sigma_x\sigma_z \neq \sigma_z\sigma_x}$.

Example 7.2 Show that $\boldsymbol{\sigma_x\sigma_z \neq \sigma_z\sigma_x}$.

$$\begin{aligned}\boldsymbol{\sigma_x\sigma_z} &= \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}\\ &= \begin{pmatrix}0\times 1+1\times 0 & 0\times 0+1\times -1\\ 1\times 1+0\times 0 & 1\times 0+0\times -1\end{pmatrix} = \begin{pmatrix}0 & -1\\ 1 & 0\end{pmatrix}\\ \boldsymbol{\sigma_z\sigma_x} &= \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\\ &= \begin{pmatrix}1\times 0+0\times 1 & 1\times 1+0\times 0\\ 0\times 0+-1\times 1 & 0\times 1+-1\times 0\end{pmatrix} = \begin{pmatrix}0 & 1\\ -1 & 0\end{pmatrix}\end{aligned} \tag{7.10}$$

Therefore, $\boldsymbol{\sigma_x\sigma_z \neq \sigma_z\sigma_x}$.

The second property is their **anti-commutation** property,

$$\{\boldsymbol{\sigma_i}, \boldsymbol{\sigma_j}\} = \boldsymbol{\sigma_i\sigma_j} + \boldsymbol{\sigma_j\sigma_i} = 2\delta_{ij}\boldsymbol{I} \tag{7.11}$$

Again, it is very important to understand each symbol thoroughly. $\{\boldsymbol{\sigma_i}, \boldsymbol{\sigma_j}\}$ is the **anti-commutator** and defined in Eq. (7.11). It is different from commutator by just changing "−" to "+." δ_{ij}, again, is the **Kronecker delta**, which is one if $i = j$ and zero if $i \neq j$. So is it a scalar, a vector, or a matrix? Yes, it is just a scalar (and a real number either 1 or 0). This makes sense because anti-commutator is a matrix and $\boldsymbol{I}$ is the identity matrix, so δ_{ij} must be a scalar like the number "1." Let us take $\{\boldsymbol{\sigma_x}, \boldsymbol{\sigma_z}\}$ as an example to get more insight. We already know $\boldsymbol{\sigma_x\sigma_z} = \begin{pmatrix}0 & -1\\ 1 & 0\end{pmatrix}$ and $\boldsymbol{\sigma_z\sigma_x} = \begin{pmatrix}0 & 1\\ -1 & 0\end{pmatrix}$ from Eq. (7.10), and therefore,

$$\begin{aligned}\{\boldsymbol{\sigma_x}, \boldsymbol{\sigma_z}\} &= \boldsymbol{\sigma_x\sigma_z} + \boldsymbol{\sigma_z\sigma_x}\\ &= \begin{pmatrix}0 & -1\\ 1 & 0\end{pmatrix} + \begin{pmatrix}0 & 1\\ -1 & 0\end{pmatrix} = \begin{pmatrix}0 & 0\\ 0 & 0\end{pmatrix} = 2\delta_{xz}\boldsymbol{I}\end{aligned} \tag{7.12}$$

Here, we introduced the **zero matrix**, $\mathbf{0}$. It is also called the **null matrix**. The result is correct because $\delta_{xz} = 0$ as $x \neq z$. We say that the three Pauli matrices, $\boldsymbol{\sigma_x}$, $\boldsymbol{\sigma_y}$, and $\boldsymbol{\sigma_z}$ anti-commute with one another because their anti-commutators are zero.

7.4 Spin Operator in Arbitrary Direction

What is the spin operator in arbitrary directions? This corresponds to setting up the magnetic field in an arbitrary direction in the 3D space we are living in. I will show you the answer and it is very confusing. However, I would like you to understand every symbol and operation. With this, I think you will be pretty familiar with the concepts of basis, vector, and matrix and you are ready to do quantum computing after being introduced to a few more essential concepts.

And again, this spin operator, when expressed as a matrix in the matrix quantum mechanics, has the eigenvalues corresponding to the possible measurement outcomes when we measure the spin in that particular direction. This is just like $\boldsymbol{\sigma_z}$ is the operator matrix with eigenvalues corresponding to the possible outcomes when performing spin measurement in the $\hat{z}$ direction.

The spin operator, $\boldsymbol{\sigma_{\vec{n}}}$, in direction $\vec{n}$ is given by

$$\boldsymbol{\sigma_{\vec{n}}} = \vec{n} \cdot \vec{\boldsymbol{\sigma}} \tag{7.13}$$

This is very confusing, but we need to understand it thoroughly. $\vec{n}$ is a unit vector in a certain direction in the real 3D space that we are living in. We may represent it as

$$\vec{n} = n_x \hat{x} + n_y \hat{y} + n_z \hat{z} \tag{7.14}$$

where $\hat{x}$, $\hat{y}$, and $\hat{z}$ are the unit vectors in the x, y, and z directions in the real space as depicted in Fig. 7.1. $\vec{n}$ is a vector in our real space. It can also be written as $|\vec{n}\rangle$ or $\begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix}$ as long as we know that the basis vectors are $\hat{x}$, $\hat{y}$, and $\hat{z}$ (or $|\hat{x}\rangle$, $|\hat{y}\rangle$, and $|\hat{z}\rangle$). Equation (7.14) and Fig. 7.1 tell us that $\vec{n}$ is a linear combination of $\hat{x}$, $\hat{y}$, and $\hat{z}$ with coefficients n_x, n_y, and n_z, respectively. We have repeated such concepts many times and only this time it is so easy to understand because it is in the real 3D space that we live in. Since $\vec{n}$ is a unit vector, which means it is **normalized**, $\langle \vec{n} | \vec{n} \rangle = 1$, and based on the inner product definition, it means $\begin{pmatrix} n_x & n_y & n_z \end{pmatrix} \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} = n_x^2 + n_y^2 + n_z^2 = 1$.

Then how about $\vec{\boldsymbol{\sigma}}$? Is it a matrix like $\boldsymbol{\sigma_z}$? But it has an arrow on the top, is it a

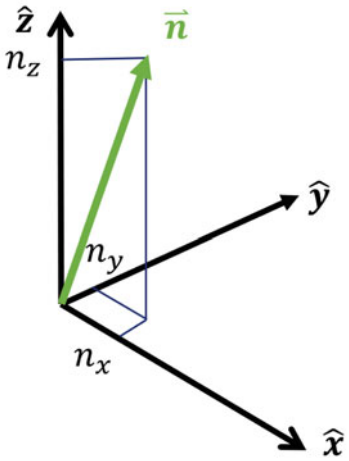


Fig. 7.1 Decomposition of a unit vector $\vec{n}$ in the 3D real space

vector? It is a vector but the coefficients are matrix. Let us write down the definition of $\vec{\boldsymbol{\sigma}}$.

$$\begin{aligned}\vec{\boldsymbol{\sigma}} &= \boldsymbol{\sigma}_x\hat{x} + \boldsymbol{\sigma}_y\hat{y} + \boldsymbol{\sigma}_z\hat{z} \\ &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\hat{x} + \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}\hat{y} + \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\hat{z}\end{aligned} \tag{7.15}$$

This is new. But there is no reason you will reject the idea of having a vector with matrices as the coefficients as long as the mathematician tells you that this is useful even you do not know why. Indeed this is useful and it is called the **Pauli vector**. There is no problem for you to accept a linear combination with real coefficients as in $\vec{n}$ because you have experience with that. You were also able to accept coefficients to be complex as we have been discussing (otherwise you would have stopped reading). Then if you trust the mathematician, I hope you can accept the fact that the coefficients can be matrices, too. Since $\vec{\boldsymbol{\sigma}}$ is a vector, I can write it in column form also as long as I know the basis vectors are $\hat{x}$, $\hat{y}$, and $\hat{z}$,

$$\vec{\boldsymbol{\sigma}} = \begin{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \\ \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \end{pmatrix} \tag{7.16}$$

And of course, I can write it as $|\vec{\boldsymbol{\sigma}}\rangle$, too. Therefore, $\boldsymbol{\sigma}_{\vec{n}}$, the spin operator, is just an inner product between the vector $\vec{n}$, which you have been seeing since middle

school, and the vector $\vec{\boldsymbol{\sigma}}$, which you are so surprised to see but is completely valid. Let us expand the terms to find $\boldsymbol{\sigma}_{\vec{\boldsymbol{n}}}$ to make sure it is indeed a matrix.

$$\begin{aligned}
\boldsymbol{\sigma}_{\vec{\boldsymbol{n}}} &= \vec{n} \cdot \vec{\boldsymbol{\sigma}} = \langle \vec{n} | \vec{\boldsymbol{\sigma}} \rangle \\
&= \begin{pmatrix} n_x & n_y & n_z \end{pmatrix} \begin{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \\ \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \end{pmatrix} \\
&= n_x \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} + n_y \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} + n_z \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \\
&= \begin{pmatrix} 0 & n_x \\ n_x & 0 \end{pmatrix} + \begin{pmatrix} 0 & -in_y \\ in_y & 0 \end{pmatrix} + \begin{pmatrix} n_z & 0 \\ 0 & -n_z \end{pmatrix} \\
&= \begin{pmatrix} n_z & n_x - in_y \\ n_x + in_y & -nz \end{pmatrix} \qquad (7.17)
\end{aligned}$$

Let us check the result with a special case. What if $\vec{n}$ is pointing at the $\hat{y}$ direction? Then it means $n_x = 0$, $n_y = 1$, and $n_z = 0$. Substituting the values into Eq. (7.17), we get $\boldsymbol{\sigma}_{\vec{\boldsymbol{n}}} = \begin{pmatrix} 0 & 0 - i1 \\ 0 + i1 & -0 \end{pmatrix} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$ which is just $\boldsymbol{\sigma_y}$.

I want to ask you the question again. Which basis are we using when we write the operators (matrices) and states (vectors) in these equations? Correct, they are in the $|0\rangle$ / $|1\rangle$ basis.

I hope this exercise reinforces your understanding of vector space and you will not confine yourselves by preoccupied concepts. Of course, we always need to make sure the new ideas are mathematically rigorous by using trustable sources.

7.5 Relationship Between Spin Direction and Real 3D Space

As mentioned earlier, the basis states of the spin spaces (e.g. $|\uparrow\rangle$, $|\downarrow\rangle$, $|\leftarrow\rangle$, $|\rightarrow\rangle$) are not the same as the real 3D space basis (i.e. $\hat{x}$, $\hat{y}$, and $\hat{z}$). They belong to different spaces and they do not talk to one another. If you do not bother to understand that, it is perfectly fine as a programmer. All you need to do is to remember that they are not the same. And you can call $|\uparrow\rangle$ as $|0\rangle$ to avoid any confusion.

But I would like to explain a little bit more why there is confusion and how the study of spin is related to the real 3D space. Let us use electron spin as an example. In an isotropic space, electrons with different spin states, $|\Psi\rangle$, have the same energy. When a magnetic field is applied in a certain direction (e.g. $\hat{z}$), two particular states

become the eigenstates (i.e. basis vectors) of the new system with that magnetic field. We call these two states $|\uparrow\rangle$ and $|\downarrow\rangle$. These two states have different energies under this external magnetic field and that is how we distinguish these two states. You see that we can just call these two states $|0\rangle$ and $|1\rangle$ or $|happy\rangle$ and $|sad\rangle$. The reason we call it $|\uparrow\rangle$ and $|\downarrow\rangle$ is because these two states are the eigenstates of a system formed by the external magnetic field in the $\hat{z}$ direction. Moreover, it also corresponds to the classical spin direction which will give the corresponding energy change. Therefore, we may say that the basis states such as $|\uparrow\rangle$, $|\downarrow\rangle$, $|\leftarrow\rangle$, $|\rightarrow\rangle$ do have a relationship to the direction in the real space. However, any linear combination of them loses the connection to the real 3D space. For example, $|\Psi\rangle = \frac{1}{\sqrt{2}}(|\uparrow\rangle + |\downarrow\rangle)$ does *not* correspond to $\hat{z} + \hat{-z}$.

7.6 Adjoint and Hermitian Matrices

We say every vector in the ket space has a corresponding vector in the bra space. An operator acts on a vector by rotating it in the hyperspace and the result is still a vector (after all, this is just a matrix-vector multiplication, which should result in a vector). If so, what will happen to its corresponding vector in the bra space? It is just like if I rotate myself 90° to the left, what is the corresponding operation to my mirror image? It is rotating the image 90° to its "right." Based on this analogy, you see that it is not trivial to know what the corresponding operator in the bra space is for a given operator in the ket space.

Mathematicians tell us that the corresponding operator in the bra space is its **adjoint matrix**. If the matrix is $\boldsymbol{A}$, we denote its adjoint matrix as $\boldsymbol{A}^\dagger$. The way to find the adjoint matrix of $\boldsymbol{A}$ is the same as finding the corresponding vector in the bra space for a given vector in the ket space, i.e. by applying conjugation, $\boldsymbol{A}^*$, and transpose, $(\boldsymbol{A}^*)^T$, operations.

Let us review again how we label the elements in a matrix. We use row-column indexing. For example, the i-th row and j-th column element of matrix $\boldsymbol{A}$ is written as A_{ij}. Then the adjoint matrix of $\boldsymbol{A}$, i.e. $\boldsymbol{A}^\dagger$, has its j-th row and i-th column element, i.e., $A^\dagger_{ji}$, equals the complex conjugate of the i-th row and j-th column element of $\boldsymbol{A}$, i.e. A^*_{ij}. Note that $A^\dagger_{ji}$ and A^*_{ij} are just scalars (complex numbers) and therefore are not bold. In summary,

$$\begin{aligned} \boldsymbol{A}^\dagger &= \boldsymbol{A}^{*T} \\ A^\dagger_{ji} &= A^*_{ij} \end{aligned} \tag{7.18}$$

Let us look at an example. If $\boldsymbol{A} = \begin{pmatrix} i & i \\ 0 & 1 \end{pmatrix}$, then $\boldsymbol{A}^* = \begin{pmatrix} -i & -i \\ 0 & 1 \end{pmatrix}$ and $(\boldsymbol{A}^*)^T = \begin{pmatrix} -i & 0 \\ -i & 1 \end{pmatrix}$. Therefore, $\boldsymbol{A}^\dagger = \begin{pmatrix} -i & 0 \\ -i & 1 \end{pmatrix}$. By choosing $i = 0$ and $j = 1$ in Eq. (7.18),

$A^*_{01} = -i$ and $A^\dagger_{10} = -i$ as expected. Note that the row and column numbering starts from 0.

How do we operate the matrix on a bra vector? While an operator operates on ket vector from the left, it operates from the right on the bra vector. If $|v\rangle = \begin{pmatrix} i \\ 1 \end{pmatrix}$, $\boldsymbol{A}\,|v\rangle = \begin{pmatrix} i & i \\ 0 & 1 \end{pmatrix}\begin{pmatrix} i \\ 1 \end{pmatrix} = \begin{pmatrix} -1+i \\ 1 \end{pmatrix}$. The corresponding vector of $\boldsymbol{A}\,|v\rangle$ in the bra space is thus $\begin{pmatrix} -1-i & 1 \end{pmatrix}$, after taking the transpose and conjugation operations. This is just like after I (i.e. $|v\rangle$) turned right by 90° (applying $\boldsymbol{A}$) and my image in the mirror is now $\begin{pmatrix} -1-i & 1 \end{pmatrix}$. I can also apply $\boldsymbol{A}^\dagger$ to $\langle v|$ directly from the *right*. Since $\langle v| = \begin{pmatrix} -i & 1 \end{pmatrix}$, $\langle v|\,\boldsymbol{A}^\dagger = \begin{pmatrix} -i & 1 \end{pmatrix}\begin{pmatrix} -i & 0 \\ -i & 1 \end{pmatrix} = \begin{pmatrix} -1-i & 1 \end{pmatrix}$. This is just like rotating the mirror image directly by 90° left. The results are the same.

What if we take the adjoint of the adjoint of a matrix? That is to perform the adjoint operation (conjugation and transpose) two times? It is easy to see it reverts to itself.

$$(\boldsymbol{A}^\dagger)^\dagger = \boldsymbol{A} \tag{7.19}$$

If a matrix's adjoint matrix equals itself, this matrix is **self-adjoint** or **Hermitian**. That is,

$$\boldsymbol{A}^\dagger = \boldsymbol{A} \quad \text{and} \quad A^\dagger_{ji} = A_{ji} \tag{7.20}$$

combining with Eq. (7.18), then

$$A^*_{ij} = A_{ji} \tag{7.21}$$

It means that after performing the transpose and complex conjugation, the matrix is still the same. This is similar to a real number, in which after performing complex conjugation, it still equals itself. The Hermitian matrix is very important in quantum computing and we will discuss it more in the next chapter.

7.7 Summary

There is a lot of mathematics in this chapter. However, it is necessary in order to understand quantum computing. I showed you step by step how to work on them. Most importantly, you need to remind yourselves all the time to distinguish scalars, vectors, and matrices in the symbols. We even show that the coefficients can be a matrix such as in the Pauli vector. Pauli matrices are very important in quantum mechanics and computing and I would like you to memorize them if you can. We

also talked about the adjoint matrix and Hermitian matrix. They are very important like the bra–ket dual correspondence and real number, respectively.

Problems

7.1 Pauli Matrix Multiplication by Hand and Using Python

Use hand calculation to show that $\boldsymbol{\sigma}_y^2 = \boldsymbol{I}$. Then verify with Colab using the following codes. Note that *np.matmul* is for matrix multiplication.

```
import numpy as np
sigmay=np.array([[0,-1j],[1j,0]])
print(sigmay)
print(np.matmul(sigmay, sigmay))
```

7.2 Pauli Matrix Multiplication by Hand and Using Python

Repeat the previous question for $\boldsymbol{\sigma}_z^2 = \boldsymbol{I}$ using both hand calculation and Colab.

7.3 Proof of Hermitianity

Show that $\boldsymbol{\sigma}_y$ is Hermitian. Hints: find $\boldsymbol{\sigma}_y^{*T}$ and show that it is the same as $\boldsymbol{\sigma}_y$. Verify using Colab.

7.4 Operation of Paul Matrix on Bra and Ket

Find $\boldsymbol{\sigma}_y |0\rangle$, $\langle 0| \boldsymbol{\sigma}_y$. Are they the same? Why? Verify using Colab. You can use *np.dot* for matrix-vector multiplication. The following shows how to do $\boldsymbol{\sigma}_y |0\rangle$.

```
zerospin=np.array([[1],[0]])
print(zerospin.shape)
np.dot(sigmay,zerospin)
```

Chapter 8
Operator Rules, Real Eigenvalues, and Projection Operator

8.1 Learning Outcomes

Understand the rules of manipulating operators in the *bra* and *ket* spaces; Appreciate the importance of Hermitian matrices and their relationship to observables; Have an intuitive understanding of projection operators; Understand the difference between inner and outer products.

8.2 Operator Rules in the Bra-ket Notation

We know that an operator is a matrix, e.g. $\boldsymbol{M}$. An operator in quantum computing rotates the vector (a quantum state), $|\alpha\rangle$, in a hyperspace. Therefore, when it operates on a vector, it gives another vector (i.e. a rotated vector), $|\alpha'\rangle$. Naturally, an operator obeys matrix operation rules. We already tried that but let us write down the equation so that we are more familiar with the notations.

$$|\alpha'\rangle = \boldsymbol{M}(|\alpha\rangle) = \boldsymbol{M}\,|\alpha\rangle \tag{8.1}$$

If there are two operators, $\boldsymbol{M}$ and $\boldsymbol{N}$, that give the same final vector after rotating the same vector, $|\alpha\rangle$, for all $|\alpha\rangle$ in the space, then they must be the same,

$$\boldsymbol{M}\,|\alpha\rangle = \boldsymbol{N}\,|\alpha\rangle\,, \quad \forall\,|\alpha\rangle \quad \Rightarrow \quad \boldsymbol{M} = \boldsymbol{N} \tag{8.2}$$

This is true unless the space only contains $|0\rangle$. $|0\rangle$ is the so-called **null vector**. It is just like the zero in real number that anything multiplied by it will be a $|0\rangle$. But remember it is still a vector. For example, in a 2-D space (not necessarily the real

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_8

2D space on the table), $|0\rangle = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. In a 4-D space, $|0\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$. What are their corresponding bra vectors? Yes, they are $\langle 0| = \begin{pmatrix} 0 & 0 \end{pmatrix}$ and $\langle 0| = \begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}$ in the 2-D and 4-D spaces, respectively. As you can see, any matrix multiplied by it will result in a null vector.

We may also have a **null operator**, $\mathbf{0}$, which gives null vector after operating on any vector, $|\alpha\rangle$. This is the **zero matrix** mentioned in the last chapter.

$$\mathbf{0}\,|\alpha\rangle = |0\rangle \tag{8.3}$$

What does it look like in matrix form? It has all elements equal to zero. In a 2D space, $\mathbf{0} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$.

As a matrix, the operator should obey distribution law also.

$$\boldsymbol{M}(C_\alpha\,|\alpha\rangle + C_\beta\,|\beta\rangle) = C_\alpha \boldsymbol{M}\,|\alpha\rangle + C_\beta \boldsymbol{M}\,|\beta\rangle \tag{8.4}$$

where C_α and C_β are scalars (any complex numbers). Again, it is very important to keep track of the meaning of each symbol and ask yourself if it is a scalar, vector, or matrix and if it is still a scalar, vector or matrix after multiplications. This equation tells us that applying $\boldsymbol{M}$ to (i.e. rotating) the linear combination of two vectors, $C_\alpha\,|\alpha\rangle + C_\beta\,|\beta\rangle$, is the same as rotating them individually first ($\boldsymbol{M}\,|\alpha\rangle$ and $\boldsymbol{M}\,|\beta\rangle$), and then perform the linear combination of the rotated vectors.

Due to dual correspondence, a vector in the ket space, $|\alpha\rangle$, like yourself in the real space, has a corresponding vector in the bra space, $\langle\alpha|$, like your image in the mirror. Then the rotated vector in the ket space, $|\alpha'\rangle$, like your body rotated right, must also have a corresponding vector in the bra space, $\langle\alpha'|$, like your mirror image rotated left. We can represent what I say as

$$\langle\alpha|\,\boldsymbol{M}^\dagger = \langle\alpha'| \quad \Leftrightarrow \quad \boldsymbol{M}\,|\alpha\rangle = |\alpha'\rangle \tag{8.5}$$

You can see that I need to apply the **adjoint of the operator**, i.e. $\boldsymbol{M}^\dagger$, *from the right* to the vector in the bra space. This is just the rule and you need to trust me that this is how it works. But why does it have to be applied from the right? This is the requirement for matrix operations. We showed an example in the last chapter already. And this is because $\langle\alpha|$ is a row vector, we can only multiply a matrix from the right. Note that based on Eq. (7.19), we can also say $\boldsymbol{M}$ is the adjoint of $\boldsymbol{M}^\dagger$ because $\boldsymbol{M} = (\boldsymbol{M}^\dagger)^\dagger$. *So the operators in the ket and bra spaces are the adjoint of each other.*

The vector in the bra space should also obey the distribution law. Let us look at the bra version of Eq. (8.4).

$$(C_\alpha^* \langle\alpha| + C_\beta^* \langle\beta|)\boldsymbol{M}^\dagger = C_\alpha^* \langle\alpha| \boldsymbol{M}^\dagger + C_\beta^* \langle\beta| \boldsymbol{M}^\dagger \tag{8.6}$$

Finally, we need to study a very important rule, the **associative axiom**. I would like you to follow closely because this is a very important step to understand the details of operators in the bra-ket notation. The associative axiom can be written as this equation,

$$(\langle\alpha|)(\boldsymbol{M}\,|\beta\rangle) = (\langle\alpha|\,\boldsymbol{M})(|\beta\rangle) \tag{8.7}$$

The left-hand side is the inner product of two vectors, $|\alpha\rangle$ and $\boldsymbol{M}\,|\beta\rangle$, and the product is a scalar. $\boldsymbol{M}\,|\beta\rangle$ is a vector because it is the rotation of $|\beta\rangle$ in the hyperspace by the operator $\boldsymbol{M}$. To do inner product, based on the rule, we multiply the bra version of $|\alpha\rangle$, i.e. $\langle\alpha|$, with the ket version of $\boldsymbol{M}\,|\beta\rangle$. Let's look at an example. Let $|\beta\rangle = \begin{pmatrix} i \\ 1 \end{pmatrix}$, $\boldsymbol{M} = \begin{pmatrix} i & i \\ 0 & 1 \end{pmatrix}$, $|\alpha\rangle = \begin{pmatrix} 2 \\ i \end{pmatrix}$. Then, $\boldsymbol{M}\,|\beta\rangle = \begin{pmatrix} i & i \\ 0 & 1 \end{pmatrix}\begin{pmatrix} i \\ 1 \end{pmatrix} = \begin{pmatrix} -1+i \\ 1 \end{pmatrix}$ and, thus, $(\langle\alpha|)(\boldsymbol{M}\,|\beta\rangle) = \begin{pmatrix} 2 & -i \end{pmatrix}\begin{pmatrix} -1+i \\ 1 \end{pmatrix} = -2+i$, which is a scalar complex number as expected. Note that $\langle\alpha|$ is $\begin{pmatrix} 2 & -i \end{pmatrix}$ after taking the transpose and conjugation of $|\alpha\rangle = \begin{pmatrix} 2 \\ i \end{pmatrix}$.

The right-hand side is also the inner product of two vectors. It is the multiplication of the bra space vector $\langle\alpha|\,\boldsymbol{M}$ and the ket space vector $|\beta\rangle$, and the product is a scalar. Let's continue the numerical example. $\langle\alpha|\,\boldsymbol{M} = \begin{pmatrix} 2 & -i \end{pmatrix}\begin{pmatrix} i & i \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 2i & i \end{pmatrix}$, which is still a vector in the bra space (row vector). So, $(\langle\alpha|\,\boldsymbol{M})(|\beta\rangle) = \begin{pmatrix} 2i & i \end{pmatrix}\begin{pmatrix} i \\ 1 \end{pmatrix} = -2+i$, which is a scalar and the same as the left-hand-side.

It is important to note that we do not use $\boldsymbol{M}^\dagger$ in this process because we are not performing any ket-space to bra-space mapping. We are purely using the associative rule to find the inner product. You may also ask yourself what is the corresponding ket space vector for $\langle\alpha|\,\boldsymbol{M}$ to test your understanding of the bra-ket notation. And yes, based on Eq. (8.5), it is $\boldsymbol{M}^\dagger\,|\alpha\rangle$.

The associative axiom in Eq. (8.7) tells us that we may apply $\boldsymbol{M}$ to the left first or the right first and the result is the same.

What is the corresponding bra space vector of $\boldsymbol{M}\,|\beta\rangle$? Firstly, since $\boldsymbol{M}\,|\beta\rangle$ is a vector, I can write it as $|\boldsymbol{M}\beta\rangle$. We have emphasized many times that this is just a way to represent a vector. As long as it is clear, we can write in any form we like. The way I am writing here is clear to me that this vector is a vector after applying $\boldsymbol{M}$ to $|\beta\rangle$. The left-hand side of Eq. (8.7) can also be written as:

$$(\langle\alpha|)(\boldsymbol{M}\,|\beta\rangle) = \langle\alpha|\boldsymbol{M}\beta\rangle \tag{8.8}$$

And the corresponding bra space vector must be $\langle \boldsymbol{M}\beta|$ by definition. Now, we still remember from Eq. (5.9), $\langle\alpha|\beta\rangle = \langle\beta|\alpha\rangle^*$, therefore,

$$\langle\alpha|\boldsymbol{M}\beta\rangle = \langle\boldsymbol{M}\beta|\alpha\rangle^* \tag{8.9}$$

But we also know that due to dual correspondence, the corresponding bra vector of $\boldsymbol{M}\,|\beta\rangle$ is also $\langle\beta|\,\boldsymbol{M}^\dagger$. So the right-hand side of Eq. (8.9) can be written as $((\langle\beta|\,\boldsymbol{M}^\dagger)\,|\alpha\rangle)^*$. Now due to the associative axiom, we can remove the parentheses, as the evaluation order does not matter. Then we have

$$\langle\alpha|\boldsymbol{M}|\beta\rangle = \left\langle\beta\middle|\boldsymbol{M}^\dagger\middle|\alpha\right\rangle^* \tag{8.10}$$

This equation tells us that if you apply complex conjugation, swap the vectors in the bra and ket space, and replace the operator with its adjoint, you get the same answer!

I hope you are familiar with the notation of $\langle\alpha|\boldsymbol{M}|\beta\rangle$ now. This is just the multiplication of a row vector to a matrix and then to a column vector from the left to the right in the form of

$$\begin{pmatrix}\blacksquare & \cdots & \blacksquare\end{pmatrix}\begin{pmatrix}\blacksquare & \cdots & \blacksquare\\ \vdots & \ddots & \vdots\\ \blacksquare & \cdots & \blacksquare\end{pmatrix}\begin{pmatrix}\blacksquare\\ \vdots\\ \blacksquare\end{pmatrix}$$

8.3 Eigenvalues of Hermitian Matrix

We have gone through some very complex math for operators. I hope you treat this seriously because this is essential for you to understand quantum computing. And please always ask yourselves how the scalar, vector, and matrix evolve to another scalar, vector, and matrix in various operations. In the last chapter, we say Hermitian matrices are very important in quantum computing. This is because Hermitian matrices have real eigenvalues. We already mentioned that matrices correspond to operators and some of them even correspond to measurements (e.g. Pauli spin matrices) and observables. The measured values are the values we read in our daily life and must be real numbers (e.g. ± 1). Therefore, all operators corresponding to observables and measurements must be Hermitian. Now let us prove that the eigenvalues of a Hermitian matrix must be real.

If a matrix $\boldsymbol{M}$ is Hermitian, it means $\boldsymbol{M}^\dagger = \boldsymbol{M}$. If λ_i and $|i\rangle$ are its i-th eigenvalue and eigenvector, respectively, by definition, we have

$$\boldsymbol{M}\,|i\rangle = \lambda_i\,|i\rangle \tag{8.11}$$

By finding the inner product of vector $\boldsymbol{M}|i\rangle$ (again, this is the rotation of $|i\rangle$ by $\boldsymbol{M}$ and it is a vector) and $|i\rangle$, Eq. (8.11) becomes

$$\langle i|\boldsymbol{M}|i\rangle = \langle i|\lambda_i|i\rangle = \lambda_i \langle i|i\rangle \tag{8.12}$$

I can pull λ_i out because it is just a scalar. What is $\langle i|i\rangle$? It is just 1 if *we assume $\boldsymbol{M}$'s eigenvectors are normalized*. But I also know, $\langle i|\boldsymbol{M}|i\rangle = \left\langle i\middle|\boldsymbol{M}^\dagger\middle|i\right\rangle^*$ due to Eq. (8.10), where I swap the vectors in the bra and ket space, take the adjoint of $\boldsymbol{M}$ and then take the complex conjugate of the resulting scalar. However, since M is Hermitian, $\left\langle i\middle|\boldsymbol{M}^\dagger\middle|i\right\rangle^* = \langle i|\boldsymbol{M}|i\rangle^*$, therefore,

$$\langle i|\boldsymbol{M}|i\rangle = \langle i|\boldsymbol{M}|i\rangle^* \tag{8.13}$$

Due to Eq. (8.12), we thus have $\lambda_i = \lambda_i^*$. And λ_i must be real as it equals to its complex conjugate.

8.4 Copenhagen Interpretation/Born Rule and Projection Operator

We showed that the operator that corresponds to an observable is required to be Hermitian to be meaningful so that it will have real eigenvalues. This is related to the **Copenhagen Interpretation** we discussed earlier. It states that

If an observable corresponding to a self-adjoint operator (i.e. Hermitian) $\boldsymbol{M}$ with discrete spectrum is measured in a system with normalized state Ψ, then

(1) The result will be one of the eigenvalues λ_i of $\boldsymbol{M}$.
(2) The probability is $\langle\Psi|\boldsymbol{P_i}|\Psi\rangle$, where $\boldsymbol{P_i}$ is projection operator to project any state to the subspace of $|i\rangle$.

This looks complicated but you understand it well already. First of all, it just reiterates that an observable must correspond to a Hermitian operator and it must have real eigenvalues. The system may be in an arbitrary quantum state $|\Psi\rangle$ (and we assumed it is normalized, i.e. $\langle\Psi|\Psi\rangle = 1$). We can measure the system using a measurement method corresponding to an operator $\boldsymbol{M}$ (e.g. the spin measurement in the z-direction corresponds to $\boldsymbol{\sigma}_z$). Here we assume the operator has N discrete spectrum of eigenvalues, λ_i, where i runs from 0 to $N-1$. For example, the eigenvalues of $\boldsymbol{\sigma}_z$ are either $+1$ or -1 (i.e. $N = 2$, $\lambda_0 = +1$, $\lambda_1 = -1$), so it is discrete. As a side note, if what we are measuring is location x by using the $\boldsymbol{x}$ operator, it can be any real number and it is continuous.

The first statement tells us that when we perform a measurement corresponding to $\boldsymbol{M}$, it must collapse to an eigenstate, $|i\rangle$, corresponding to one of the eigenvalues, λ_i, of $\boldsymbol{M}$ (e.g. either spin up or down in the $\boldsymbol{\sigma}_z$ case). We know this well and we have discussed using analogies of spinning coins or the girl's state earlier. The second

statement tells us the probability of a certain state we will get. We learned that the probability $|\Psi\rangle$ will collapse to $|i\rangle$ is the magnitude square of the coefficient of $|i\rangle$ when $|\Psi\rangle$ is expressed as the linear combination of the basis states formed by the eigenstates of $\boldsymbol{M}$. For example, we can rewrite Eq. (4.15) as

$$|\Psi\rangle = a_0\,|0\rangle + a_1\,|1\rangle + \cdots + a_{N-1}\,|N-1\rangle \tag{8.14}$$

The probability to get state $|i\rangle$ is just $|a_i|^2$. This is very easy to find when we are using the eigenstates of $\boldsymbol{M}$ as the basis states to expand/represent $|\Psi\rangle$ because it is just the square of the magnitude of the coefficient. In the above statement, it gives a more general equation to find the probability, i.e. $\langle\Psi|\boldsymbol{P_i}|\Psi\rangle$, by introducing the **projection operator**, $\boldsymbol{P_i}$. This will be very useful when $|\Psi\rangle$ is not expressed as a linear combination of its eigenstates (i.e. another basis is used). The projection operator is just a matrix. What it does is to project a state into a subspace. In our real 3D space, a building has a 3D nature that has height, width, and depth. We can project its shadow on the floor, maybe the 2D x-y plane, or on a vertical wall, maybe the 2D y-z plane. The floor and the wall are the subspaces of the 3D space we live in. To do so, we need to adjust the light source accordingly, which corresponds to two different *projection operations*. The concept is the same in linear algebra. $|\Psi\rangle$ is a vector in a high dimensional space. We can project it to a lower dimension space using an appropriate projection operator. Here $\boldsymbol{P_i}$ is the projection operator that will project an arbitrary vector $|\Psi\rangle$ to the subspace spanned by $|i\rangle$. The equation to form the projection operator is very simple and the projection operator that projects a vector to a subspace formed by $|i\rangle$ is given by

$$\boldsymbol{P_i} = |i\rangle\,\langle i| \tag{8.15}$$

It looks very simple but also very confusing. What is the meaning of a ket on the left of a bra? Is it a scalar, vector, or matrix? We already say $\boldsymbol{P_i}$ is a projection operator, so it must be a matrix. Therefore, a ket multiplied by a bra is a matrix and this is called the **outer product** or **tensor product** of vectors. This is an important concept and I will show you that it is very easy to construct on a certain basis. Let us write it in matrix form but then we need to choose a set of basis states first. The easiest one is to choose the eigenstates of $\boldsymbol{M}$ as the basis states. Then by definition,

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} \quad \cdots \quad |N-1\rangle = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix} \tag{8.16}$$

So $|i\rangle$ has the i-th entry non-zero in a column vector and $\langle i|$ has the i-th entry non-zero in a row vector. Let's assume $N = 4$ and $i = 2$, we have

$$\begin{aligned} \boldsymbol{P_i} &= \boldsymbol{P_2} = |2\rangle\langle 2| \\ &= \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \end{aligned} \tag{8.17}$$

If you try to derive it, you see that $\boldsymbol{P_i}$ is a matrix with zero elements everywhere except the i-th diagonal element.

Let us see how $\boldsymbol{P_i}$ accomplish the projection task and firstly I want to show you the power of Dirac notation. Let us apply $\boldsymbol{P_i}$ to $|\Psi\rangle$ in Eq. (8.14),

$$\boldsymbol{P_i}\,|\Psi\rangle = \boldsymbol{P_i}(a_0\,|0\rangle + a_1\,|1\rangle + \cdots + a_{N-1}\,|N-1\rangle) \tag{8.18}$$

We can distribute $\boldsymbol{P_i}$ into each term due to the distribution rule in Eq. (8.4),

$$\begin{aligned} \boldsymbol{P_i}\,|\Psi\rangle &= \boldsymbol{P_i}(a_0\,|0\rangle + a_1\,|1\rangle + \cdots + a_{N-1}\,|N-1\rangle) \\ &= a_0\boldsymbol{P_i}\,|0\rangle + a_1\boldsymbol{P_i}\,|1\rangle + \cdots + a_{N-1}\boldsymbol{P_i}\,|N-1\rangle \\ &= a_0\,|i\rangle\langle i|\;|0\rangle + a_1\,|i\rangle\langle i|\;|1\rangle + \cdots + a_{N-1}\,|i\rangle\langle i|\;|N-1\rangle \end{aligned} \tag{8.19}$$

Due to the association rule, $a_j\,|i\rangle\langle i|\;|j\rangle = a_j\,|i\rangle(\langle i|\;|j\rangle) = a_j\,|i\rangle\langle i|j\rangle$ for $j = 0$ to $N-1$. Since the basis states are supposed to be orthonormal (Eqs. (5.11) and (5.12)), only $\langle i|i\rangle$ is non-zero. Therefore, $\boldsymbol{P_i}\,|\Psi\rangle = a_i\,|i\rangle\langle i|\;|i\rangle = a_i\,|i\rangle$. You can see that the projection operation removes all the orthogonal components and only keeps the component in the subspace spanned by $|i\rangle$. Then we apply the bra vector $\langle\Psi|$ from the left,

$$\begin{aligned} \langle\Psi|\boldsymbol{P_i}|\Psi\rangle &= \langle\Psi|(\boldsymbol{P_i}|\Psi\rangle) = \langle\Psi|(a_i|i\rangle) \\ &= (a_0^*\langle 0| + a_1^*\langle 1| + \cdots + a_{N-1}^*\langle N-1|)a_i\,|i\rangle \\ &= a_0^*a_i\langle 0|i\rangle + a_1^*a_i\langle 1|i\rangle + \cdots + a_{N-1}^*a_i\langle N-1|i\rangle \\ &= a_i^*a_i = |a_i|^2 \end{aligned} \tag{8.20}$$

Here, we expand the bra vector $\langle\Psi|$ as a linear combination of the basis vectors in their bra form. Or you can think of the expression as the bra form of Eq. (8.14). Then we apply the distribution and orthogonality rules to get the final expression. You see that $\langle\Psi|\boldsymbol{P_i}|\Psi\rangle$ does give the probability of measuring $|i\rangle$, i.e. $|a_i|^2$.

What if we apply $\boldsymbol{P_i}$ twice? Intuitively, if we project the vector, $|\Psi\rangle$, on a basis vector and then project that resulting vector, $\boldsymbol{P_i}\,|\Psi\rangle$, on the same basis again, we expect no change. Indeed,

$$\begin{aligned}\boldsymbol{P_i}\,\boldsymbol{P_i}\,|\Psi\rangle &= \boldsymbol{P_i}\,|i\rangle\,\langle i|\,|\Psi\rangle = |i\rangle\,\langle i|\;\;|i\rangle\,\langle i|\,|\Psi\rangle \\ &= |i\rangle\,(\langle i|\;\;|i\rangle)\,\langle i|\,|\Psi\rangle = |i\rangle\,(1)\,\langle i|\,|\Psi\rangle = \boldsymbol{P_i}\,|\Psi\rangle \end{aligned} \tag{8.21}$$

where we used the fact that it is an orthonormal basis and thus $\langle i|i\rangle = 1$. Therefore,

$$\boldsymbol{P_i}\,\boldsymbol{P_i} = \boldsymbol{P_i} \tag{8.22}$$

8.5 Summary

Again we went through a lot of mathematics in this chapter. We are almost equipped with all the skills necessary to understand quantum computing and perform simulations. We know how to apply operators in the bra and ket spaces. In particular, we find that the associative axiom is very useful to simplify an expression, through which we can perform the trivial operation first. We proved Hermitian matrix has real eigenvalues. We also introduced the projection operator and outer product concepts. These two concepts are very useful in the future "steps". Let's practice more and enhance our understanding of the math involved.

Problems

8.1 Find Eigenvalues using Colab

Find the eigenvectors and eigenvalues of $\boldsymbol{\sigma_x}$ using colab. You may use this code. But please ask yourself, which basis are we using when we express $\boldsymbol{\sigma_x}$ in the following form?

```
import numpy as np
sigmax=np.array([[0,1],[1,0]])
print(sigmax)
eigenvalues, eigenvectors = np.linalg.eig(sigmax)
print(eigenvalues)
print(eigenvectors)
```

8.2 Construct Projection Operator

Now, use hand calculation, derive the projection operators to project to the eigenstates of $\boldsymbol{\sigma_z}$.

8.3 Use Projection Operator to find Measurement probability

Use the projector equation to find the probability of measuring spin up and down when it is applied to the eigenvectors of σ_x Hints: you should get 50% for all of them.

8.4 Vector normalization by hand and using python 2

Check Problem 8.3 using co-lab. Hints:

```
Use np.transpose for transpose
Use np.dot for multiplication, including outer product
Use np.linalg.eig to find eigenvalues
```

Chapter 9
Eigenvalue, Matrix Diagonalization and Unitary Matrix

9.1 Learning Outcomes

Understand the meaning of matrix diagonalization and its equivalence to finding eigenvalues and eigenvectors; able to find eigenvalues and eigenvectors; understand the importance of unitary matrix and its properties.

9.2 Eigenvalues and Matrix Diagonalization

We have learned how to find the eigenvalues and eigenvectors in Chap. 6. Here I would like to reinforce the concept by showing a more general example and then introduce the concept of matrix diagonalization. This is related to the change of basis which we have discussed earlier and will go into more details later.

For any matrix, $\boldsymbol{A}$, with dimension $n \times n$, it can be written as the following with n rows and n columns.

$$\boldsymbol{A} = \begin{pmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,n-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1,0} & a_{n-1,1} & \cdots & a_{n-1,n-1} \end{pmatrix} = (a_{i,j}) \tag{9.1}$$

We index the matrix elements using the row-column convention. We may also write it as $(a_{i,j})$ to indicate that the matrix has the i-th row and j-th column element equals to $a_{i,j}$. We know these well.

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_9

To find its eigenvalues, λ_i, and eigenvectors, $|i\rangle$, we need to solve the following equation,

$$\boldsymbol{A}\,|i\rangle = \lambda_i\,|i\rangle \tag{9.2}$$

Let me remind you again, the eigenvector of a matrix is so special because when you apply the matrix to that vector, the vector is not rotated but just scaled by a scalar. This is what Eq. (9.2) tells us. As a reminder, $|i\rangle$ is a column vector. For example, $|i\rangle$ may be $\begin{pmatrix}1\\2\\0\\5\end{pmatrix}$ if $n = 4$. In general, $|i\rangle = \begin{pmatrix}\alpha_0\\\alpha_1\\\vdots\\\alpha_{n-1}\end{pmatrix}$ for an n-dimensional vector. Therefore, Eq. (9.2) can be written as

$$\boldsymbol{A}\,|i\rangle = \begin{pmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,n-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1,0} & a_{n-1,1} & \cdots & a_{n-1,n-1} \end{pmatrix} \begin{pmatrix}\alpha_0\\\alpha_1\\\vdots\\\alpha_{n-1}\end{pmatrix} = \lambda_i \begin{pmatrix}\alpha_0\\\alpha_1\\\vdots\\\alpha_{n-1}\end{pmatrix} \tag{9.3}$$

The right-hand side is the multiplication of a scalar and a vector. But we can also write it as a matrix-vector multiplication,

$$\begin{aligned}\boldsymbol{A}\,|i\rangle &= \begin{pmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,n-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1,0} & a_{n-1,1} & \cdots & a_{n-1,n-1} \end{pmatrix} \begin{pmatrix}\alpha_0\\\alpha_1\\\vdots\\\alpha_{n-1}\end{pmatrix} \\ &= \begin{pmatrix} \lambda_i & 0 & \cdots & 0 \\ 0 & \lambda_i & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_i \end{pmatrix} \begin{pmatrix}\alpha_0\\\alpha_1\\\vdots\\\alpha_{n-1}\end{pmatrix} = \lambda_i \boldsymbol{I}\,|i\rangle \end{aligned} \tag{9.4}$$

We see that a scalar multiplied to a vector acts the same as a diagonal matrix with the scalar value as the diagonal element multiplied to that vector. This is because $\lambda_i\,|i\rangle = \lambda_i \boldsymbol{I}\,|i\rangle$, where $\boldsymbol{I}$ is the identity matrix. If we move the right-hand side term to the left, we have

$$(\boldsymbol{A} - \lambda_i \boldsymbol{I})\,|i\rangle = \begin{pmatrix} a_{0,0} - \lambda_i & a_{0,1} & \cdots & a_{0,n-1} \\ a_{1,0} & a_{1,1} - \lambda_i & \cdots & a_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1,0} & a_{n-1,1} & \cdots & a_{n-1,n-1} - \lambda_i \end{pmatrix} \begin{pmatrix}\alpha_0\\\alpha_1\\\vdots\\\alpha_{n-1}\end{pmatrix} = \begin{pmatrix}0\\0\\\vdots\\0\end{pmatrix} \tag{9.5}$$

The right-hand side is the *null vector* $\mathbf{0}$. We need to equate the left to the right and there are n equations. For the j-th row, we need to equate

$$(a_{j,0})\alpha_0 + (a_{j,1})\alpha_1 + \cdots + (a_{j,j} - \lambda_j)\alpha_j + \cdots + (a_{j,n-1})\alpha_{n-1} = 0 \quad (9.6)$$

Here you need to trust me if you forgot or have not learned how to solve for λ_i and α_i's. If all α_i's equal to zero, i.e. $|i\rangle = 0$, that is a valid solution but this is useless to us. We want to find non-zero $|i\rangle$ which will be the eigenvectors of $\boldsymbol{A}$. We have n equations and n variables (α_0 to α_{n-1}) (note that λ_i is not a variable in this perspective). The theory tells us that we need to make the determinant of the matrix in Eq. (9.5) zero.

$$|\boldsymbol{A} - \lambda_i \boldsymbol{I}| = \begin{vmatrix} a_{0,0} - \lambda_i & a_{0,1} & \cdots & a_{0,n-1} \\ a_{1,0} & a_{1,1} - \lambda_i & \cdots & a_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1,0} & a_{n-1,1} & \cdots & a_{n-1,n-1} - \lambda_i \end{vmatrix} = 0 \quad (9.7)$$

This is equivalent to what we did in Chap. 6. Let us repeat what we did in Chap. 6 but use the new equations to make sure we are familiar with the process. We want to find the eigenvalues and eigenvectors of $\boldsymbol{\sigma_x}$. Since $\boldsymbol{\sigma_x} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, the equation we need to solve is

$$\begin{vmatrix} a_{0,0} - \lambda_i & a_{0,1} \\ a_{1,0} & a_{1,1} - \lambda_i \end{vmatrix} = \begin{vmatrix} 0 - \lambda_i & 1 \\ 1 & 0 - \lambda_i \end{vmatrix} = 0 \quad (9.8)$$

The determinant is thus $-\lambda_i \times -\lambda_i - 1 \times 1$. Therefore, the equation becomes,

$$\lambda_i^2 - 1 = 0 \quad (9.9)$$

And thus, $\lambda_i = \pm 1$. We can assign, $\lambda_0 = 1$ and $\lambda_1 = -1$. This is what we got in Chap. 6. Now, I know the eigenvalues. I can use them in Eq. (9.5) to find the eigenvectors. For $\lambda_0 = 1$, we have

$$\begin{pmatrix} a_{0,0} - \lambda_0 & a_{0,1} \\ a_{1,0} & a_{1,1} - \lambda_0 \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} = \begin{pmatrix} 0 - 1 & 1 \\ 1 & 0 - 1 \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad (9.10)$$

This gives us two equations,

$$\begin{aligned} (0 - 1)\alpha_0 + (1)\alpha_1 &= 0 \\ (1)\alpha_0 + (0 - 1)\alpha_1 &= 0 \end{aligned} \quad (9.11)$$

This is the same as Eq. (6.11) where $e = \alpha_0$, $f = \alpha_1$, and $\gamma = 1$. So we will not solve it again. The answer is $|+\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}$ for $\lambda_0 = 1$. And for $\lambda_0 = -1$, the eigenvector is $|-\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix}$. Note that I call the vector corresponding to eigenvalue +1, i.e. λ_0, $|+\rangle$ and that corresponding to eigenvalue -1, i.e. λ_1, $|-\rangle$. I could have called them $|0\rangle$ and $|1\rangle$, respectively, but I do not want to confuse it with the basis vectors of $\boldsymbol{\sigma_z}$.

Let me ask you the question again, which set of basis vectors are we using when we performed the derivations? Yes, we have been using the basis vectors of $\boldsymbol{\sigma_z}$. For example, $\begin{pmatrix}\alpha_0\\\alpha_1\end{pmatrix}$ means $\alpha_0|0\rangle + \alpha_1|1\rangle$ instead of, for example, $\alpha_0|+\rangle + \alpha_1|-\rangle$.

We can also represent $\boldsymbol{\sigma_x}$ in the basis formed by its eigenvectors, $|+\rangle$ and $|-\rangle$. We know $|+\rangle = 1\times|+\rangle + 0\times|-\rangle = \begin{pmatrix}1\\0\end{pmatrix}$ and $|-\rangle = 0\times|+\rangle + 1\times|-\rangle = \begin{pmatrix}0\\1\end{pmatrix}$. How would $\boldsymbol{\sigma_x}$ look like? We already mentioned this in Eq. (6.9),

$$\boldsymbol{\sigma_{x,|+\rangle/|-\rangle}} = \begin{pmatrix}1 & 0\\0 & -1\end{pmatrix} \tag{9.12}$$

We did not mention earlier but do you see that this is a diagonal matrix with the eigenvalues being the diagonal elements? In general, if we represent a matrix using its eigenvectors as the basis vectors, it will be a diagonal matrix with the eigenvalues as the diagonal elements. This process is called **diagonalization**.

$$\boldsymbol{A} = \begin{pmatrix}\lambda_0 & 0 & \cdots & 0\\0 & \lambda_1 & \cdots & 0\\\vdots & \vdots & \ddots & \vdots\\0 & 0 & \cdots & \lambda_{n-1}\end{pmatrix}_{\textit{In the basis formed by A's eigenvectors}} \tag{9.13}$$

But how do we do that? We need to find its eigenvectors and eigenvalues first. Therefore, the diagonalization of a matrix is equivalent to finding its eigenvectors and eigenvalues.

It is also important to note that while the matrix and vector representations change with basis, the eigenvalues are kept the same. This is very important.

It seems that you have not learned anything new by spending so much effort to reach here. The purposes actually have 3 folds, namely (1) to reinforce the skill of finding eigenvectors and eigenvalues and see it from a more general perspective, (2) to reinforce the concept of representing matrix in different bases, and (3) to understand the significance of diagonal matrix and its relationship to its eigenvectors.

9.3 Unitary Matrix

In quantum computing and quantum mechanics, the **unitary matrix** is very important. This is as important as the **Hermitian matrix**. Let us remind ourselves what a matrix is. It is an operator that rotates the vector (state). A Hermitian matrix is a matrix that is equal to its adjoint matrix, i.e. $\boldsymbol{A}^\dagger = \boldsymbol{A}$, where $\boldsymbol{A}^\dagger$ is the adjoint matrix of $\boldsymbol{A}$ by performing the complex conjugate and transpose operations. The Hermitian matrix is important because it has real eigenvalues and therefore, all observables must correspond to Hermitian matrices. *The unitary matrix is important because it preserves the inner product of vectors when they are transformed together by a unitary matrix*. It also *preserves the length of a vector*. This is very important because it will preserve the probability amplitude of a vector in quantum computing so that it is always 1. Let us start with the definition and understand why it is important.

A unitary matrix, $\boldsymbol{U}$, is defined as a matrix whose adjoint, $\boldsymbol{U}^\dagger$, equals to its inverse, $\boldsymbol{U}^{-1}$ (instead of to itself as in the Hermitian matrix).

$$\boldsymbol{U}^\dagger = \boldsymbol{U}^{-1} \tag{9.14}$$

If we multiply both sides by $\boldsymbol{U}$ from the right or the left, respectively, we get

$$\begin{aligned} \boldsymbol{U}^\dagger \boldsymbol{U} &= \boldsymbol{I} \\ \boldsymbol{U}\boldsymbol{U}^\dagger &= \boldsymbol{I} \end{aligned} \tag{9.15}$$

where $\boldsymbol{I}$ is the identity matrix. This is because any matrix multiplied by its inverse must equal the identity matrix by definition.

The definition is very simple and I hope you can memorize it. Why this is important? Assume there are two vectors, $|f\rangle$ and $|g\rangle$. Their inner product is $\langle f|g\rangle$ (i.e. the bra version of $|f\rangle$ times the ket version of $|g\rangle$). What happens if we operate $\boldsymbol{U}$ on the vectors (i.e. rotate them in the hyperspace)? We expect their inner product to be the same, right? For example, your table's corner angle should be the same if you rotate your table (recall that the inner product is related to the angle between the two edges in the real space). If after rotation, the table angle changes, it means the table is distorted. This is not something we want. Or, if two basis vectors are orthogonal to each other (meaning they will not appear at the same time during a measurement), I hope they are still orthogonal after being rotated together. Otherwise, it means they can exist at the same time after rotation when it is measured. This is just like in the rotating coin analogy, $|head\rangle$ and $|tail\rangle$ cannot appear at the same time during measurement. It looks strange if they can appear at the same time during measurement just because they have been rotated in the space together due to the rotation of the Earth.

After applying $\boldsymbol{U}$, $|f\rangle$ and $|g\rangle$ become $\boldsymbol{U}|f\rangle$ and $\boldsymbol{U}|g\rangle$, respectively. Remember, $\boldsymbol{U}|f\rangle$ and $\boldsymbol{U}|g\rangle$ are two new vectors. Their inner product is the multiplication of the bra version of $\boldsymbol{U}|f\rangle$ on the ket version of $\boldsymbol{U}|g\rangle$. We have learned that the bra

version of $\boldsymbol{U}\,|f\rangle$ is $\langle f|\,\boldsymbol{U}^\dagger$. Therefore, the inner product of the two rotated vectors is $\left\langle f\middle|\boldsymbol{U}^\dagger\boldsymbol{U}\middle|g\right\rangle$. But due to the association rule, I can evaluate $\boldsymbol{U}^\dagger\boldsymbol{U}$ first and this is just $\boldsymbol{I}$ based on Eq. (9.15). Therefore,

$$\left\langle f\middle|\boldsymbol{U}^\dagger\boldsymbol{U}\middle|g\right\rangle = \langle f|\boldsymbol{I}|g\rangle = \langle f|g\rangle \tag{9.16}$$

You see that the inner product is conserved if the rotation matrix is unitary. *In quantum computing, when we operate a qubit, we need to make sure the operation matrix is unitary*. The unitary matrix is also used to describe the *evolution* of a quantum state as a function of time.

Now, I would like to introduce the important properties of the columns and rows in a unitary matrix. Assume

$$\boldsymbol{U} = \begin{pmatrix} b_{0,0} & b_{0,1} & \cdots & b_{0,n-1} \\ b_{1,0} & b_{1,1} & \cdots & b_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n-1,0} & b_{n-1,1} & \cdots & b_{n-1,n-1} \end{pmatrix} \tag{9.17}$$

where it has n columns and n rows.

Then Eq. (9.15) becomes

$$\begin{aligned} \boldsymbol{U}^\dagger\boldsymbol{U} &= \begin{pmatrix} b^*_{0,0} & b^*_{1,0} & \cdots & b^*_{n-1,0} \\ b^*_{0,1} & b^*_{1,1} & \cdots & b^*_{n-1,1} \\ \vdots & \vdots & \ddots & \vdots \\ b^*_{0,n-1} & b^*_{1,n-1} & \cdots & b^*_{n-1,n-1} \end{pmatrix} \begin{pmatrix} b_{0,0} & b_{0,1} & \cdots & b_{0,n-1} \\ b_{1,0} & b_{1,1} & \cdots & b_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n-1,0} & b_{n-1,1} & \cdots & b_{n-1,n-1} \end{pmatrix} \\ &= \boldsymbol{I} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \end{aligned} \tag{9.18}$$

Note that we performed transpose and conjugation operations to obtain $\boldsymbol{U}^\dagger$. The i-th row and j-th column element of $\boldsymbol{I}$ is just the inner product of the i-th row vector of $\boldsymbol{U}^\dagger$ and the j-th column vector of $\boldsymbol{U}$. For example,

$$\boldsymbol{I}_{0,1} = 0 = b^*_{0,0}b_{0,1} + b^*_{1,0}b_{1,1} + \cdots + b^*_{n-1,0}b_{n-1,1} \tag{9.19}$$

Or in general,

$$\boldsymbol{I}_{i,j} = b^*_{0,i}b_{0,j} + b^*_{1,i}b_{1,j} + \cdots + b^*_{n-1,i}b_{n-1,j} = \delta_{i,j} \tag{9.20}$$

In the above equation, since only the diagonal element in $\boldsymbol{I}$ is non-zero and equal to one, therefore, $\delta_{i,j}$ is used.

If we treat every column of $\boldsymbol{U}$ as a vector and called the i-th column $|v_i\rangle$, we have

$$|v_i\rangle = \begin{pmatrix} b_{0,i} \\ b_{1,i} \\ \vdots \\ b_{n-1,i} \end{pmatrix} \quad \text{and} \quad \langle v_i| = \begin{pmatrix} b^*_{0,i} & b^*_{1,i} & \cdots & b^*_{n-1,i} \end{pmatrix} \tag{9.21}$$

Here we take the transpose and conjugate to get the bra version of $|v_i\rangle$. And Eq. (9.20) is just the inner product of the column vectors of $\boldsymbol{U}$, i.e.

$$\boldsymbol{I}_{i,j} = b^*_{0,i} b_{0,j} + b^*_{1,i} b_{1,j} + \cdots + b^*_{n-1,i} b_{n-1,j} = \langle v_i | v_j \rangle \tag{9.22}$$

However, for the identity matrix, only the diagonal elements are one and the others are zero. That means

$$\langle v_i | v_j \rangle = \delta_{i,j} \tag{9.23}$$

Again, this Kronecker delta tells us that the inner products are zero if they are not from the same column and one if they are from the same column (i.e. same vector). Therefore, *the column vectors of any unitary matrix are* ***orthonormal*** *to each other*. This is a very useful property. And this is the same for the row vectors in the unitary matrix.

9.4 Summary

We derived another equation to find the eigenvalues of a matrix. This is equivalent to what we learned in Chap. 6 but is sometimes more convenient. We see that if we can represent a matrix in the basis formed by its eigenvectors, the matrix is diagonal and the diagonal elements are just its eigenvalues. We then introduced the unitary matrix. The unitary matrix is important in quantum computing because it preserves the inner products of any two vectors if they are transformed by the unitary matrix. It also preserves the magnitude of a vector. Moreover, the rows and columns of the unitary matrix are orthonormal to each other.

We can also prove that for an orthonormal set of basis vectors after it is transformed by a matrix, $\boldsymbol{U}$, if it is still orthonormal, then $\boldsymbol{U}$ must be unitary. We will use this to check if a quantum oracle is unitary in the future. We will prove this in the Problems.

Problems

9.1 Orthonormality of Unitary Matrix Row Vectors

Prove that the row vectors of a unitary matrix are orthonormal.

9.2 Finding the Determinants of Pauli Matrices by Hand

Find the determinants of $\boldsymbol{\sigma_x}$, $\boldsymbol{\sigma_y}$, and $\boldsymbol{\sigma_z}$. What do you notice?

9.3 Finding the Determinants of Pauli Matrices using Colab

Confirm your answers in Problem 9.2 using Colab. This is a sample code.

```
import numpy as np
M=np.array([[1,0], [0,-1]])
print(np.linalg.det(M))
```

9.4 Proof of Unitarity

For an orthonormal set of basis, $|v_i\rangle$, we have $\langle v_i | v_j \rangle = \delta_{i,j}$ (Eq. (9.23)). We transform it using a matrix $\boldsymbol{A}$, i.e. they become $\boldsymbol{A}|v_i\rangle$. Prove that if after the transformation, they are still orthonormal, then $\boldsymbol{A}$ is unitary. Hints: Express the basis vectors in column form and each of them has only 1 entry. Express $\boldsymbol{A}$ in its matrix form (like Eq. (9.17)). The fact that after the transformation, they are still orthonormal implies that the columns of $\boldsymbol{A}$ are orthonormal and thus $\boldsymbol{A}$ is unitary.

Chapter 10
Unitary Transformation, Completeness, and Construction of Operator

10.1 Learning Outcomes

Able to perform unitary transformation; able to construct unitary transformation matrix from the given bases; be prepared to use the completeness equation for quantum computing; able to construct operator from the given eigenvectors and eigenvalues.

10.2 Unitary Transformation

We have discussed the importance of the unitary matrix because it is used in quantum computing very often. The definition of a unitary matrix, $\boldsymbol{U}$, is that it satisfies $\boldsymbol{U}^{\dagger}\boldsymbol{U} = \boldsymbol{I}$. This is not very interesting until we showed that when two vectors (states) are transformed by the same unitary matrix, their inner product is unchanged (preserved) (Eq. (9.16)). If all the vectors (states) are transformed by the same unitary vector, it is equivalent to transforming the coordinates in the opposite "direction". This might sound confusing. Let us remind ourselves what transformation is. It is nothing but just the rotation of vectors (states) in a hyperspace. So, if we rotate all vectors in the same way, it is equivalent to rotating the coordinates or the hyperspace in the opposite way. This is just like if there is only the Earth in an empty space, you can say the plane is flying around a fixed Earth or the Earth is rotating, in the opposite direction, under a stationary plane.

Naturally, a transformation by a unitary matrix is called a **unitary transformation**.

If unitary transformation is the most important when we transform all vectors together, then it is also the most important when we perform coordinates (basis) transformation, but in the opposite direction. And indeed, whenever we talk about unitary transformation in this book, we apply it to coordinate transformation.

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_10

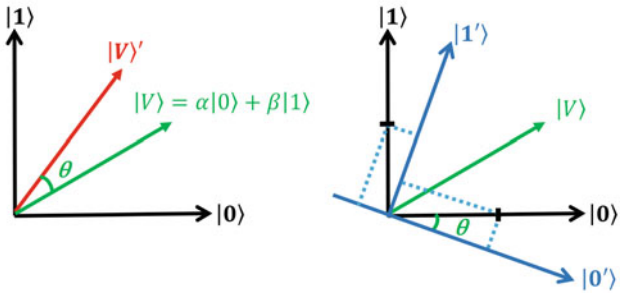


Fig. 10.1 Vector $|V\rangle$ rotated anti-clockwise by θ (left) is equivalent to the basis vectors $|0\rangle$ / $|1\rangle$ rotated clockwise by θ

Let us look at an example of unitary transformation in Fig. 10.1.

This looks like a 2D plane. Indeed, it is a 2D plane and you can call $|0\rangle$ and $|1\rangle$, $|\hat{x}\rangle$ and $|\hat{y}\rangle$, if you like. But I deliberately label it as $|0\rangle$ and $|1\rangle$ so that you are instilled with the concept that transformation of quantum states in hyperspace which we cannot see is just similar to rotating vectors in the 2D plane. So, you can say I am giving you a fake example. However, everything I am telling you is valid except the visualization is wrong.

We know any vector $|V\rangle$ can be represented as a linear combination (i.e. superposition) of the basis states and we write it as

$$|V\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix} \tag{10.1}$$

I need to remind you that we are borrowing the 2D plane for visualization and $|0\rangle$ and $|1\rangle$ are orthogonal to each other and they cannot co-exist with 100% certainty at the same time. When we perform a measurement, $|V\rangle$ will collapse to either $|0\rangle$ or $|1\rangle$ with probabilities $|\alpha|^2$ and $|\beta|^2$, respectively. Again, I want to remind you that when we write it as $\begin{pmatrix}\alpha\\ \beta\end{pmatrix}$, we know that we are using $|0\rangle$ and $|1\rangle$ as the basis states.

Now if we rotate (transform) $|V\rangle$ anti-clockwise by θ (Left of Fig. 10.1), it becomes $|V'\rangle$. How do we represent it in the *old basis* in terms of the old basis states $|0\rangle$ and $|1\rangle$? By using trigonometry, we can find the new representation. But here I want to present another view. Instead of rotating $|V\rangle$ anti-clockwise by θ, we can also rotate the basis (i.e. its basis states $|0\rangle$ and $|1\rangle$) clockwise by θ (Right of Fig. 10.1). The representation of $|V\rangle$ by the new basis states $|0'\rangle$ and $|1'\rangle$ will then be equal to the representation of $|V'\rangle$ by the old basis states $|0\rangle$ and $|1\rangle$. So, here I will apply a coordinate (basis) transformation. All I need to know is how to represent $|0\rangle$ and $|1\rangle$ in terms of $|0'\rangle$ and $|1'\rangle$ and we have done something similar before in Eq. (5.4) when we introduced the concept of basis change. And yes, the

coordinate transformation is just another name for the basis change. From Fig. 10.1 (right), using trigonometry,

$$
\begin{aligned}
|0\rangle &= \cos\theta \left|0'\right\rangle + \sin\theta \left|1'\right\rangle = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} \\
|1\rangle &= -\sin\theta \left|0'\right\rangle + \cos\theta \left|1'\right\rangle = \begin{pmatrix} -\sin\theta \\ \cos\theta \end{pmatrix}
\end{aligned} \tag{10.2}
$$

I also wrote $|0\rangle$ and $|1\rangle$ in column form. Which basis are we using now? Yes, it is the $\left|0'\right\rangle$ and $\left|1'\right\rangle$ basis. We need to be very careful on which basis we are using when writing vectors in column/row forms. I can then substitute Eq. (10.2) into Eq. (10.1).

$$
\begin{aligned}
|V\rangle &= \alpha\,|0\rangle + \beta\,|1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}_{|0\rangle/|1\rangle} \\
&= \alpha(\cos\theta \left|0'\right\rangle + \sin\theta \left|1'\right\rangle) + \beta(-\sin\theta \left|0'\right\rangle + \cos\theta \left|1'\right\rangle) \\
&= (\alpha\cos\theta - \beta\sin\theta) \left|0'\right\rangle + (\alpha\sin\theta + \beta\cos\theta) \left|1'\right\rangle \\
&= \begin{pmatrix} \alpha\cos\theta - \beta\sin\theta \\ \alpha\sin\theta + \beta\cos\theta \end{pmatrix}_{|0'\rangle/|1'\rangle}
\end{aligned} \tag{10.3}
$$

Now you see for the same $|V\rangle$, it can have two different column forms because it is represented in two different bases. In this process, we have found a matrix to transform the representation in the old basis to the new basis,

$$
\begin{pmatrix} \alpha\cos\theta - \beta\sin\theta \\ \alpha\sin\theta + \beta\cos\theta \end{pmatrix}_{|0'\rangle/|1'\rangle} = \boldsymbol{U} \begin{pmatrix} \alpha \\ \beta \end{pmatrix}_{|0\rangle/|1\rangle} \tag{10.4}
$$

where

$$
\boldsymbol{U} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \tag{10.5}
$$

Let us check if $\boldsymbol{U}$ is unitary, i.e. $\boldsymbol{U}^\dagger\boldsymbol{U} = \boldsymbol{I}$.

$$
\begin{aligned}
\boldsymbol{U}^\dagger\boldsymbol{U} &= \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}^\dagger \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \\
&= \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \\
&= \begin{pmatrix} \cos\theta\cos\theta + \sin\theta\sin\theta & -\cos\theta\sin\theta + \sin\theta\cos\theta \\ -\sin\theta\cos\theta + \cos\theta\sin\theta & \sin\theta\sin\theta + \cos\theta\cos\theta \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \boldsymbol{I}
\end{aligned} \tag{10.6}
$$

Finally, it is important to remind ourselves that this transformation matrix is to *rotate a vector by θ anti-clockwise or rotate the basis by $-\theta$ (or θ clockwise)*.

We can also see this in another perspective. If you already know the representation of $|V\rangle$ in the $|0\rangle$ / $|1\rangle$ basis (i.e. Eq. (10.1)), you can find the representation of $|V\rangle$ in a newly given basis, $|0'\rangle$ / $|1'\rangle$ by using the matrix $\boldsymbol{U}$ given in Eq. (10.5).

10.3 Construction of Unitary Transformation Matrices

How do we construct the transformation matrix? We can use trigonometry if it is a 2D or 3D real space like what we just did. How do we do that for a higher dimension? I will not derive it but I will show you the method. It is not difficult to memorize and I hope you can memorize it and this will make your life easier when you do quantum computing.

Let the basis vectors in the old basis be $|0\rangle, |1\rangle, \ldots, |n-1\rangle$ and those in the new basis be $|0'\rangle, |1'\rangle, \ldots, |n-1'\rangle$ in an n-dimensional space. The transformation matrix to represent a vector in the new basis is given by

$$\boldsymbol{U} = \begin{pmatrix} \langle 0'|0\rangle & \langle 0'|1\rangle & \cdots & \langle 0'|n-1\rangle \\ \langle 1'|0\rangle & \langle 1'|1\rangle & \cdots & \langle 1'|n-1\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle n-1'|0\rangle & \langle n-1'|1\rangle & \cdots & \langle n-1'|n-1\rangle \end{pmatrix} \tag{10.7}$$

The i-th row j-th column element is just the inner product of the i-th new basis vector and the j-th old basis vector, $\langle i'|j\rangle$. Let us check and practice using the previous 2D case.

Example 10.1 Derive Eq. (10.5) using Eq. (10.7).

Based on Eq. (10.2),

$$\begin{aligned} \langle 0'|0\rangle &= \langle 0'| \ (\cos\theta\,|0'\rangle + \sin\theta\,|1'\rangle) = \cos\theta \\ \langle 0'|1\rangle &= \langle 0'| \ (-\sin\theta\,|0'\rangle + \cos\theta\,|1'\rangle) = -\sin\theta \\ \langle 1'|0\rangle &= \langle 1'| \ (\cos\theta\,|0'\rangle + \sin\theta\,|1'\rangle) = \sin\theta \\ \langle 1'|1\rangle &= \langle 1'| \ (-\sin\theta\,|0'\rangle + \cos\theta\,|1'\rangle) = \cos\theta \end{aligned} \tag{10.8}$$

Here we have used the fact that $|0'\rangle$ and $|1'\rangle$ are orthonormal and used the distributive property. Therefore,

$$\boldsymbol{U} = \begin{pmatrix} \langle 0'|0\rangle & \langle 0'|1\rangle \\ \langle 1'|0\rangle & \langle 1'|1\rangle \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \tag{10.9}$$

which is the same as Eq. (10.5).

Now I want to show you that even though the example is incorrect visually, it does gives us some intuitions into the spin qubit transformation we will learn later. When $\theta = 45°$ (and this is the case in Fig. 5.1), $\boldsymbol{U}$ becomes,

$$\begin{aligned}\boldsymbol{U} &= \begin{pmatrix}\cos\theta & -\sin\theta \\ \sin\theta & \cos\theta\end{pmatrix} \\ &= \begin{pmatrix}\cos 45° & -\sin 45° \\ \sin 45° & \cos 45°\end{pmatrix} = \begin{pmatrix}\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\end{pmatrix}\end{aligned} \tag{10.10}$$

Recall that the eigenvectors of $\boldsymbol{\sigma_z}$ are $|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}$ and $|1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$ and the eigenvectors of $\boldsymbol{\sigma_x}$ are $|+\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}$ and $|-\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix}$ (see Sect. 6.4). If a quantum state was represented in the $|0\rangle / |1\rangle$ basis, what is the transformation matrix to represent it in the $|+\rangle / |-\rangle$ basis? We can use Eq. (10.7),

$$\begin{aligned}\boldsymbol{U} &= \begin{pmatrix}\langle -|0\rangle & \langle -|1\rangle \\ \langle +|0\rangle & \langle +|1\rangle\end{pmatrix} \\ &= \begin{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1 & -1\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix} & \frac{1}{\sqrt{2}}\begin{pmatrix}1 & -1\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix} \\ \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix} & \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}\end{pmatrix} = \begin{pmatrix}\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\end{pmatrix}\end{aligned} \tag{10.11}$$

You see that the transformation matrix from the $\boldsymbol{\sigma_z}$ basis to the $\boldsymbol{\sigma_x}$ basis is just the same as Eq. (10.10) which is obtained by using the fake 2D plane analogy.

10.4 Completeness of Basis

We already said that in quantum computing, the basis needs to be orthonormal. One of the main reasons is that it makes our life much easier when deriving equations (e.g. it allows us to use $\langle 1|0\rangle = 0$). We showed that in many places including when we derived Eq. (10.8). Another important requirement for the basis is its **completeness**. This is trivial but important. This just means that the basis should span the whole space we are interested in. For example, a basis composed of $|\hat{x}\rangle$ and $|\hat{y}\rangle$ is complete for a real 2D space but incomplete for a real 3D space. Every vector in the real 2D space can be represented as a linear combination of $|\hat{x}\rangle$ and $|\hat{y}\rangle$, but some vectors in the real 3D space cannot (e.g. the vector $2|\hat{z}\rangle$). When the basis is complete, I can decompose any vector in the space of interest into a linear combination of the basis vectors. This is trivial and we have been taking this for granted since the beginning. For example, in an n-D space, a vector $|V\rangle$ can be

described as the linear combination of a basis composed of basis vectors $|i\rangle$, where i runs from 0 to $n-1$.

$$|V\rangle = \sum_{i=0}^{n-1} \alpha_i |i\rangle \tag{10.12}$$

where α_i are scalars (complex numbers) and describe "how much" $|i\rangle$ vector $|V\rangle$ has. How to find α_j (which is one of the α_i inside the summation)? With Dirac's bra–ket notation, this is very straightforward and we just need to perform the inner product of $|j\rangle$ and $|i\rangle$ and take advantage of the orthonormal property of the basis vectors,

$$\langle j|V\rangle = \sum_{i=0}^{n-1} \alpha_i \langle j|i\rangle = \sum_{i=0}^{n-1} \alpha_i \delta_{ji} = \alpha_j \tag{10.13}$$

Or we can think in another way, we can project $|V\rangle$ on to $|j\rangle$ to find the resulting vector. Equation (8.15) tells us that the projection operator is just $\boldsymbol{P_j} = |j\rangle\langle j|$. So now I will perform the project operation and this is just prepending the projection operator on $|V\rangle$.

$$\begin{aligned} \boldsymbol{P_j}|V\rangle &= |j\rangle\langle j|V\rangle \\ &= \sum_{i=0}^{n-1} \alpha_i |j\rangle\langle j|i\rangle = \sum_{i=0}^{n-1} \alpha_i |j\rangle \delta_{ji} = \alpha_j |j\rangle \end{aligned} \tag{10.14}$$

Note $\langle j|i\rangle$ is a scalar and we can move it freely to the front inside the summation. It gives the same result as Eq. (10.13) except it gives the projected vector instead of just the "amount" of the component $|j\rangle$. I can also substitute $\alpha_j = \langle j|V\rangle$ from Eq. (10.13), but rename j to i (because what we want to substitute is $|i\rangle$), into Eq. (10.12),

$$\begin{aligned} |V\rangle &= \sum_{i=0}^{n-1} \alpha_i |i\rangle \\ &= \sum_{i=0}^{n-1} \langle i|V\rangle |i\rangle \\ &= \sum_{i=0}^{n-1} |i\rangle\langle i|V\rangle \\ &= \left(\sum_{i=0}^{n-1} |i\rangle\langle i|\right)|V\rangle \end{aligned} \tag{10.15}$$

We used the fact that $\langle i|V\rangle$ is a number and moved it to the end from line 2 to line 3. And then we used the associative axiom in line 4. Note that $\sum_{i=0}^{n-1} |i\rangle\langle i|$ is a matrix. Since it gives $|V\rangle$ when it is multiplied to an arbitrary vector $|V\rangle$, it means it must be the identity matrix.

$$\sum_{i=0}^{n-1} |i\rangle\langle i| = \boldsymbol{I} \tag{10.16}$$

This is a very important result of completeness. It tells us that the outer products of each basis to itself sum up to an identity matrix. We will use this property in many derivations in the future.

Let us do an example. For the space spanned by the eigenvectors of $\boldsymbol{\sigma_x}$, we have

$$\begin{aligned}\sum_{i=0}^{n-1} |i\rangle\langle i| &= |-\rangle\langle -| + |+\rangle\langle +| \\ &= \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1 & -1\end{pmatrix} + \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} + \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\end{pmatrix} \\ &= \frac{1}{2}\begin{pmatrix}1 & -1\\-1 & 1\end{pmatrix} + \frac{1}{2}\begin{pmatrix}1 & 1\\1 & 1\end{pmatrix} = \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix} = \boldsymbol{I}\end{aligned} \tag{10.17}$$

10.5 Construct Operator from Eigenvalues and Eigenvectors

We are very familiar with bra–ket operations and finding eigenvectors and eigenvalues from a given matrix now. What if you are given the eigenvectors and eigenvalues of an operator, how do we construct the operator. This happens often because eigenvalues are what we really can feel and measure. If an operator has eigenvalues λ_i and eigenvectors $|i\rangle$, where i runs from 0 to $n-1$, the operator, $\boldsymbol{A}$, is given by,

$$\boldsymbol{A} = \sum_{i=0}^{n-1} \lambda_i |i\rangle\langle i| \tag{10.18}$$

We will not prove this. But when the matrix is represented in the basis formed by its eigenvectors, this is just the diagonal matrix given in Eq. (9.13). Let us try to understand what each symbol in the equation means and how to operate. Let us consider $\boldsymbol{\sigma_z}$. Its eigenvectors are $|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}$ and $|1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$ with eigenvalues of 1 and -1, respectively. Then it means

$$\begin{aligned}\boldsymbol{\sigma_z} &= \sum_{i=0}^{2-1} \lambda_i \left|i\right\rangle \left\langle i\right| \\ &= \lambda_0 \left|0\right\rangle \left\langle 0\right| + \lambda_1 \left|1\right\rangle \left\langle 1\right| \\ &= 1 \times \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \end{pmatrix} + -1 \times \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \end{aligned} \tag{10.19}$$

This is exactly the matrix form of $\boldsymbol{\sigma_z}$. Again, which basis are we using when writing the matrix and vectors in this way? Yes, in the basis of the eigenstates of $\boldsymbol{\sigma_z}$.

10.6 Summary

The concept that we have been emphasizing since the beginning is the representation of states in different bases. This is just like, for the same meaning, I can say "He is very handsome" in English or "keui hou leng jai" in Cantonese. We have to be very meticulous in representing vectors in different bases. And to transform the vector from one representation to another representation, we can use a unitary matrix, which essentially rotates the basis in the hyperspace and allows us to represent the vector in a new basis. The matrix is the dictionary and the rotation is the transformation from English to Cantonese. We also discussed the concept of completeness and also how to construct a matrix from given eigenvalues and eigenvectors. They will be proved to be very useful in quantum computing later. Right now, let us practice the skills.

Problems

10.1 Finding Unitary Matrix by Hand
Construct the matrix for the transformation from the basis of $\boldsymbol{\sigma_y}$ to $\boldsymbol{\sigma_x}$. In other words, finding the unitary matrix to *re-represent* a vector in the $\boldsymbol{\sigma_y}$ eigenvector basis representation in the $\boldsymbol{\sigma_x}$ eigenvector basis representation.

10.2 Completeness of Basis
Show that the eigenvectors of $\boldsymbol{\sigma_x}$ form a complete basis (*Hint*: see Eq. (10.17)).

10.3 Construction of Matrix
Construct the $\boldsymbol{\sigma_x}$ by using it eigenvalues and eigenvectors.

10.4 Colab Verification
Use Colab to verify the results from Exercises 10.1 to 10.3.

Chapter 11
Hilbert Space, Tensor Product, and Multi-Qubit

11.1 Learning Outcomes

Have an idea that Hilbert space is just an extension of the real space; understand that tensor product is a way to construct a higher-dimensional Hilbert space from lower ones; appreciate the power of quantum computing due to the tensor product of qubits; familiar with important tensor product operations.

11.2 Hilbert Space

Even if we do not understand the **Hilbert Space**, we still can understand the materials in this book as long as you trust me and follow the rules exactly. But understanding a little bit of the Hilbert Space will make us look cool in front of those who do not know. More importantly, of course, it helps us appreciate the rules and prepares us for more advanced topics. You will find that this is the fundamental of all the rules we have discussed. It is the backdrop of the stage of quantum computing.

A Hilbert space is a complete linear space with an inner product defined. The word "complete" here means it converges in certain operations and it is not the same as the completeness concept we learned earlier. I just want you to be aware of this and we will not explain more. Then a Hilbert space is just a linear space with an inner product defined. It is just a generalization of our real 3D space. So we can loosely regard a Hilbert space as a space with any dimensions which allows complex coefficients. This is what we have been seeing so far. In this space, it has vectors and it has a well-defined inner product, $\langle f|g\rangle$, of any pair of vectors, $|f\rangle$ and $|g\rangle$, and for any complex number, c, with the following properties

$$\langle f|f\rangle \geq 0 \tag{11.1}$$

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_11

$$\langle cf|g\rangle = c^* \langle f|g\rangle \tag{11.2}$$

$$\langle f|cg\rangle = c \langle f|g\rangle \tag{11.3}$$

$$\langle f|g\rangle = \langle g|f\rangle^* \tag{11.4}$$

$$\langle f+h|g\rangle = \langle f|g\rangle + \langle h|g\rangle \tag{11.5}$$

Before I explain them, let us take a moment to understand what are the meanings of the symbols in Eqs. (11.1)–(11.5). Firstly, ask yourselves which are scalars (numbers), vectors (states), and matrices (operators). Here, c is a scalar and $|f\rangle$ and $|g\rangle$ are vectors. Anything with $\langle\ |\ \rangle$ are inner products and, thus, scalars. For example, $\langle f+h|g\rangle$ means the inner product of $|g\rangle$ with $|f+h\rangle$, where $|f+h\rangle$ is just a vector due to the sum of another two vectors $|f\rangle$ and $|h\rangle$.

I think you have realized that we have learned them before. Equation (11.1) just tells us that the inner product of a vector to itself has to be positive or zero. This is because $\langle f|f\rangle$ is just the square of the length of the vector $|f\rangle$. Eq. (3.23) is a special case for this in the real space. And if $|f\rangle$ is normalized, we have Eq. (4.16). Equation (11.2) tells us that if we scale $|f\rangle$ by a scalar c and then take it bra version to multiply to $|g\rangle$, it is the same as multiplying $|f\rangle$'s bra version on $|g\rangle$ followed by multiplying the complex conjugate of c. This is just equivalent to Eq. (5.7) due to the definition of dual correspondence. But no complex conjugate is needed if the c is used to scale the second vector, which only appears in its ket version in Eq. (11.3). Equation (11.4) is the same as Eq. (5.9), meaning that projecting $|g\rangle$ on $|f\rangle$ gives a scalar which is the complex conjugate of the projection of $|f\rangle$ on $|g\rangle$. Equation (11.5) is just the distribution properties of a linear operation that we have been using and taken for granted.

So we did not learn anything new. What we have learned is that what we have been doing is based on some very solid mathematical ground and I hope this will make you feel more confident in moving forward.

11.3 Expansion of Hilbert Space and Tensor Product

While we have not learned anything new, this prepares us for a very important concept, namely the tensor product. If there is one electron, I know that its spin forms a Hilbert space with the basis vectors of spinning up or down (i.e. $|0\rangle$ or $|1\rangle$). I say it is a Hilbert space because it fulfills the complete requirement and the requirements in Eqs. (11.1)–(11.5). You can check so by checking that every vector in this space, which is just a linear combination of the basis states, $\alpha|0\rangle + \beta|1\rangle$, fulfills the requirements. We have done many exercises and are pretty sure they do obey those properties. As a reminder, we may also use $|+\rangle$ / $|-\rangle$ as the basis but it is still the *same* Hilbert space (just different representations). Let me call this space $\mathcal{H}_1$.

Now, what if I also want to describe the position of the electron. The position itself also forms a Hilbert space with the basis vectors being any point

in the real space. We have discussed this earlier. The state of the electron in this space is a superposition of the basis vectors. That is why in principle, the electron in a water molecule in the cloud has a finite probability to be in my cup as $|Location\ state\ of\ electron\rangle = \alpha\,|At\ a\ point\ in\ the\ cloud\rangle + \beta\,|At\ a\ point\ in\ my\ cup\rangle + \cdots$, although β can be very small. This is just to remind you of the concept of superposition and basis states. Let me call this space, $\mathcal{H}_2$.

Now, just trust me, the spin Hilbert space and the position Hilbert space can form a bigger (higher-dimensional) space, $\mathcal{H}_3$. The way to form it is by using the **tensor product** denoted as $\otimes$,

$$\mathcal{H}_3 = \mathcal{H}_1 \otimes \mathcal{H}_2 \tag{11.6}$$

I have been asking you to understand the meaning of each symbol since the beginning to make sure you do not get lost. Now, besides scalars, vectors, and matrices, you need to also check for the "space." This is very important. You do not know the meaning of $\otimes$ yet, I will show you its operations later. Please take it for granted now. If the larger space is a tensor product of two smaller spaces, then how do we form the vector in the larger space from the two smaller spaces? We also use the tensor product. If $|f\rangle_1$ and $|g\rangle_2$ are two vectors in $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively, it can form a vector $|h\rangle_3$ in $\mathcal{H}_3$ through

$$|h\rangle_3 = |f\rangle_1 \otimes |g\rangle_2 \tag{11.7}$$

The tensor product is what makes quantum computing so powerful. Note that the spin space has *no* interaction with the position space. You can treat them as two quantities in two different universes. They never talk to each other. As a result, the basis vectors in the spin space and the position space can form a new basis in $\mathcal{H}_3$ through permutation. Let us understand this using another example. Assume we have two electrons. The first electron has its own spin space with basis vectors $|0\rangle_1$ and $|1\rangle_1$. Note that I use a subscript to denote that the basis vectors are of the first electron or in Hilbert space $\mathcal{H}_1$. The state of the first electron again can be a linear combination of the basis states,

$$|f\rangle_1 = \alpha_1\,|0\rangle_1 + \beta_1\,|1\rangle_1 \tag{11.8}$$

Similarly, for the second electron, it has its own spin space $\mathcal{H}_2$ and basis vectors $|0\rangle_2$ and $|1\rangle_2$ and any spin states of the second electron $|g\rangle_2$ can be represented as

$$|g\rangle_2 = \alpha_2\,|0\rangle_2 + \beta_2\,|1\rangle_2 \tag{11.9}$$

To describe the system of the two electrons which have individual spins, we can use a tensor product, $\mathcal{H}_3 = \mathcal{H}_1 \otimes \mathcal{H}_2$. What are the basis states of this new system

with two electrons? It will be the permutation through tensor product of their basis states,

$$\begin{aligned}
|0\rangle_3 &= |0\rangle_1 \otimes |0\rangle_2 = |0\rangle_1 |0\rangle_2 \\
|1\rangle_3 &= |0\rangle_1 \otimes |1\rangle_2 = |0\rangle_1 |1\rangle_2 \\
|2\rangle_3 &= |1\rangle_1 \otimes |0\rangle_2 = |1\rangle_1 |0\rangle_2 \\
|3\rangle_3 &= |1\rangle_1 \otimes |1\rangle_2 = |1\rangle_1 |1\rangle_2
\end{aligned} \tag{11.10}$$

Here it says, $\mathcal{H}_3$ has four basis states and they are just the permutation of electron 1's and electron 2's basis states. For example, the fourth basis state is called $|3\rangle_3$. As mentioned many times, this is just the name. As long as you know what it is referring to, you can call it any name. You can call it $|The\ fourth\ state\ of\ \mathcal{H}_3\rangle_3$ or $|The\ last\ state\rangle_{two-electron\ system}$ or $|11\rangle_3$ in which the binary representation of 3 is used. They all refer to the tensor product states due to the spin down basis state of electron 1 and the spin down basis state of electron 2. Similarly, the tensor product of two vectors is nothing (so far) but just *an ordering of two vectors*. Therefore, I can omit $\otimes$ in the expression of the tensor product when there is no confusion.

How to represent an arbitrary state $|h\rangle_3$ in $\mathcal{H}_3$? Again it must be a linear combination of the basis vectors in $\mathcal{H}_3$. Therefore,

$$|h\rangle_3 = \alpha_3 |0\rangle_3 + \beta_3 |1\rangle_3 + \gamma_3 |2\rangle_3 + \delta_3 |3\rangle_3 \tag{11.11}$$

where α_3, β_3, γ_3, and δ_3 are the complex coefficients.

Since it really does not matter how I label the basis states, by substituting Eq. (11.10), I can rewrite Eq. (11.11) as

$$\begin{aligned}
|h\rangle_3 &= \alpha_3 |0\rangle_1 \otimes |0\rangle_2 + \beta_3 |0\rangle_1 \otimes |1\rangle_2 + \gamma_3 |1\rangle_1 \otimes |0\rangle_2 + \delta_3 |1\rangle_1 \otimes |1\rangle_2 \\
&= \alpha_3 |0\rangle_1 |0\rangle_2 + \beta_3 |0\rangle_1 |1\rangle_2 + \gamma_3 |1\rangle_1 |0\rangle_2 + \delta_3 |1\rangle_1 |1\rangle_2 \\
&= \alpha_3 |0\rangle |0\rangle + \beta_3 |0\rangle |1\rangle + \gamma_3 |1\rangle |0\rangle + \delta_3 |1\rangle |1\rangle \\
&= \alpha_3 |00\rangle + \beta_3 |01\rangle + \gamma_3 |10\rangle + \delta_3 |11\rangle
\end{aligned} \tag{11.12}$$

I omitted the subscript eventually because I know the first and second number refers to the first and second electron, respectively. I can also write it in column form,

$$|h\rangle_3 = \begin{pmatrix} \alpha_3 \\ \beta_3 \\ \gamma_3 \\ \delta_3 \end{pmatrix} \tag{11.13}$$

As usual, ask yourselves, when it is written in this way, which basis are we using?

11.4 Multi-Qubits

We have discussed the meaning of qubit. It stands for *quantum bit*. It refers to a vector space with two basis vectors and it is 2D. We are very familiar with this already. The vector in this vector space can be represented by a qubit, which is the linear combination of the basis vectors. And the basis vectors can be $|Yes_L\rangle / |No_L\rangle$, $|Yes_P\rangle / |No_P\rangle$, or $|Head\rangle / |Tail\rangle$ for those not so correct analogies or $|0\rangle / |1\rangle$ and $|+\rangle / |-\rangle$ in a single spin system. We just discussed that we can form a larger system with two qubits and the number of basis vectors becomes four. Since the tensor product is used to construct a larger system from its subsystems through the permutation of basis vectors, the number of basis states grows exponentially and this is the power of quantum computing. Figure 11.1 shows how the number of basis vectors grows with the number of qubits.

For a single qubit, we call it a $\mathbb{C}^2$ space, which means it is a 2D complex vector space ($\mathbb{C}$ means complex). When there are 2 qubits, the space is $\mathbb{C}^4$ as it is 4-dimensional due to the tensor product of two $\mathbb{C}^2$ spaces. Similarly, if there are n qubits, it is a tensor product of n $\mathbb{C}^2$ spaces and the dimension is 2^n instead of n. If I have 10 qubits, the dimension is now 1024 and there are 1024 basis states! A vector (state) in the 10-qubit system will be a superposition of 1024 basis states which will enable quantum parallelism as you will see later.

We also see that the basis vectors of multi-qubit systems have some special patterns. Since it is just the permutation of the basis vectors of single-qubits, then it is just the permutation of 0's and 1's. Therefore, they are just the binary numbers from 0 to $2^n - 1$ for n-qubits. This helps us to memorize the basis vectors and also allows us to write them succinctly in decimal numbers in the kets. For example, the

Number of Qubits	Number of basis vectors	Basis Vectors	Space
1	2	$\|0\rangle, \|1\rangle$	$\mathbb{C}^2$
2	4	$\|0\rangle\|0\rangle, \|0\rangle\|1\rangle, \|1\rangle\|0\rangle, \|1\rangle\|1\rangle$ Or: $\|00\rangle, \|01\rangle, \|10\rangle, \|11\rangle$ Or: $\|0\rangle, \|1\rangle, \|2\rangle, \|3\rangle$	$\mathbb{C}^4 = \mathbb{C}^2 \otimes \mathbb{C}^2$
3	8	$\|0\rangle\|0\rangle\|0\rangle, \|0\rangle\|0\rangle\|1\rangle, \cdots, \|1\rangle\|1\rangle\|1\rangle$ Or: $\|000\rangle, \|001\rangle, \cdots, \|111\rangle$ Or: $\|0\rangle, \|1\rangle, \cdots, \|7\rangle$	$\mathbb{C}^8 = \mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \mathbb{C}^2$
n	2^n	$\|0\rangle, \|1\rangle, \cdots, \|2^n\text{-}1\rangle$	$\mathbb{C}^{2^n} = \mathbb{C}^2 \otimes \cdots \otimes \mathbb{C}^2 = (\mathbb{C}^2)^{n\otimes}$

Fig. 11.1 Basis vectors of multi-qubit systems

permutation of 3 basis vectors of $|1\rangle$ is $|1\rangle\,|1\rangle\,|1\rangle$ and I can write it as $|111\rangle$ or $|7\rangle$ without ambiguity. However, I need to be very careful about the number of qubits a system has if I write it in decimal. This is because, for a 3-qubit system, $|7\rangle = |111\rangle$ but in a 5-qubit system, $|7\rangle = |00111\rangle$ where there are two extra electrons.

Moreover, we need to be very careful how each qubit is ordered. That is to make sure we know which qubit is treated as the **Most Significant Bit** (MSB) and which is the **Least Significant Bit** (LSB). For the spins of two electrons, A and B, I may write electron A on the left (as the MSB) or on the right (as the LSB). In the former case, the tensor product state due to electron A spinning up and electron B spinning down is $|0\rangle_A\,|1\rangle_B = |0\rangle\,|1\rangle = |1\rangle$. In the latter case, it becomes $|1\rangle_B\,|0\rangle_A = |1\rangle\,|0\rangle = |2\rangle$. Therefore, just like we need to make sure which basis we are using when we write the vectors in their column form, we also need to make sure how the subspaces are ordered if we ignore the subscript in a tensor product state.

Let us use Eqs. (11.7)–(11.13) to understand vector tensor product from another perspective, i.e. in their column forms.

$$|h\rangle_3 = |f\rangle_1 \otimes |g\rangle_2$$

$$\begin{pmatrix} \alpha_3 \\ \beta_3 \\ \gamma_3 \\ \delta_3 \end{pmatrix} = \begin{pmatrix} \alpha_1 \\ \beta_1 \end{pmatrix} \otimes \begin{pmatrix} \alpha_2 \\ \beta_2 \end{pmatrix} \tag{11.14}$$

The column vector form clearly shows us how two $\mathbb{C}^2$ can form a $\mathbb{C}^4$ vector. But how do we find the $\begin{pmatrix} \alpha_3 \\ \beta_3 \\ \gamma_3 \\ \delta_3 \end{pmatrix}$ for the given $\begin{pmatrix} \alpha_1 \\ \beta_1 \end{pmatrix}$ and $\begin{pmatrix} \alpha_2 \\ \beta_2 \end{pmatrix}$? This is not difficult if we combine Eq. (11.14) with Eqs. (11.8) and (11.9)

$$\begin{aligned} |h\rangle_3 &= \alpha_3\,|0\rangle_1 \otimes |0\rangle_2 + \beta_3\,|0\rangle_1 \otimes |1\rangle_2 + \gamma_3\,|1\rangle_1 \otimes |0\rangle_2 + \delta_3\,|1\rangle_1 \otimes |1\rangle_2 \\ &= \alpha_3\,|0\rangle_1\,|0\rangle_2 + \beta_3\,|0\rangle_1\,|1\rangle_2 + \gamma_3\,|1\rangle_1\,|0\rangle_2 + \delta_3\,|1\rangle_1\,|1\rangle_2 \\ &= |f\rangle_1 \otimes |g\rangle_2 \\ &= (\alpha_1\,|0\rangle_1 + \beta_1\,|1\rangle_1)(\alpha_2\,|0\rangle_2 + \beta_2\,|1\rangle_2) \\ &= \alpha_1\alpha_2\,|0\rangle_1\,|0\rangle_2 + \alpha_1\beta_2\,|0\rangle_1\,|1\rangle_2 + \beta_1\alpha_2\,|1\rangle_1\,|0\rangle_2 + \beta_1\beta_2\,|1\rangle_1\,|1\rangle_2 \end{aligned} \tag{11.15}$$

Here we have used the distribution rule of tensor product to be introduced in Eq. (11.20). We can equate the coefficients of each basis in line 2 and line 5 and obtain

$$\alpha_3 = \alpha_1\alpha_2; \quad \beta_3 = \alpha_1\beta_2; \quad \gamma_3 = \beta_1\alpha_2; \quad \delta_3 = \beta_1\beta_2 \tag{11.16}$$

Therefore, this is just a permutation of the coefficients through scalar multiplication. So, we can also do this in column vector using the following steps

$$\begin{aligned}\begin{pmatrix}\alpha_3\\ \beta_3\\ \gamma_3\\ \delta_3\end{pmatrix} &= \begin{pmatrix}\alpha_1\\ \beta_1\end{pmatrix} \otimes \begin{pmatrix}\alpha_2\\ \beta_2\end{pmatrix}\\ &= \begin{pmatrix}\alpha_1\begin{pmatrix}\alpha_2\\ \beta_2\end{pmatrix}\\ \beta_1\begin{pmatrix}\alpha_2\\ \beta_2\end{pmatrix}\end{pmatrix} = \begin{pmatrix}\alpha_1\alpha_2\\ \alpha_1\beta_2\\ \beta_1\alpha_2\\ \beta_1\beta_2\end{pmatrix}\end{aligned} \tag{11.17}$$

We will discuss more in the following chapters.

11.5 More About Tensor Product in Hilbert Space

I would like to show you some more rules about the tensor product in the Hilbert space. It is not necessary to memorize them but we will use them in future derivations. They are actually intuitive.

If $\mathcal{H}_3 = \mathcal{H}_1 \otimes \mathcal{H}_2$, what vectors does $\mathcal{H}_3$ contains? It contains the linear combination of the tensor products of the vectors in $\mathcal{H}_1$ and $\mathcal{H}_2$.

$$|h\rangle = \sum_{j=1}^{n} |f_i\rangle \otimes \left|g_j\right\rangle \tag{11.18}$$

For example, n can be 1 and $|h\rangle$ can be $|0\rangle\,|1\rangle$ or $|1\rangle\,|1\rangle$. Or n can be 2, then $|h\rangle$ may be $i\,|0\rangle\,|1\rangle + |1\rangle\,|0\rangle$, if $|f_1\rangle = i\,|0\rangle$, $|f_2\rangle = |1\rangle$, $|g_1\rangle = |1\rangle$, and $|g_2\rangle = |0\rangle$.

You may also scale the vector in their subspace first and then perform tensor product or vice versa. The result is the same,

$$c\,|f\rangle \otimes |g\rangle = |cf\rangle \otimes |g\rangle = |f\rangle \otimes |cg\rangle \tag{11.19}$$

It also obeys distribution rule, from both right and left,

$$\begin{aligned}(|f\rangle + |e\rangle) \otimes |g\rangle &= |f\rangle \otimes |g\rangle + |e\rangle \otimes |g\rangle\\ |g\rangle \otimes (|f\rangle + |e\rangle) &= |g\rangle \otimes |f\rangle + |g\rangle \otimes |e\rangle\end{aligned} \tag{11.20}$$

Finally, if we perform inner product in their respective $\mathcal{H}_1$ and $\mathcal{H}_2$ spaces first and then perform the tensor product, it is the same as forming tensor product first and then perform inner product in the $\mathcal{H}_3$ space.

$$\langle h_1|h_2\rangle = \left\langle \sum_{j=1}^{n_1} \left|f_j\right\rangle \otimes \left|g_j\right\rangle \middle| \sum_{l=1}^{n_2} |e_l\rangle \otimes |k_l\rangle \right\rangle = \sum_{j=1}^{n_1}\sum_{l=1}^{n_2} \left\langle f_j|e_l\right\rangle \left\langle g_j|k_l\right\rangle \quad (11.21)$$

Here, $|h_1\rangle$ and $|h_2\rangle$ are the vectors in the $\mathcal{H}_3$ space. They are formed by the linear combination of tensor products of $\left|f_j\right\rangle$ and $|e_l\rangle$ in the $\mathcal{H}_1$ space and $\left|g_j\right\rangle$ and $|k_l\rangle$ in the $\mathcal{H}_2$. This shows again the subspaces are not "talking to each other" because $\left|f_j\right\rangle$ and $\left|g_j\right\rangle$ only form inner products with $|e_l\rangle$ and $|k_l\rangle$, respectively.

11.6 Summary

We have learned about the Hilbert space. It is not necessary to memorize all its properties but it is important to appreciate that many of its properties are similar to the 3D space we are living in and many of the operations are familiar to us in the previous chapters already. We also learned the skills of constructing higher-dimensional Hilbert space using the tensor product, in particular, the procedure to produce a vector (state) in the higher-dimensional space using the given lower-dimensional vectors. These concepts are very important to understanding quantum computing algorithms. We are almost there.

Problems

11.1 Computing Tensor Product

Let $|f\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ and $|g\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} \end{pmatrix}$, find $|f\rangle_1 \otimes |g\rangle_2$ and $|g\rangle_1 \otimes |f\rangle_2$. Are they the same? Here you see the order is very important.

11.2 Computing Tensor Products of Three Vectors

Find $\begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. What is the dimension of this space?

11.3 Distribution Law

Check Eq. (11.20) by substituting $|f\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, $|e\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|g\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} \end{pmatrix}$ to both sides.

11.4 Tensor Products using Colab

Use Colab to verify the results from Problems 11.1 to 11.3. For Problem 11.1, you may use this code

```
import numpy as np
g = np.array([[0],[1]])
h = np.array([[1/np.sqrt(2)],[1j/np.sqrt(2)]])
print(np.kron(g,h))
print(np.kron(h,g))
```

11.5 Inner Product of Tensor Products

Check Eq. (11.21) but setting $n_1 = n_2 = 1$ (i.e. only one term) and $|f\rangle = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, $|e\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, $|g\rangle = \begin{pmatrix} i \\ i \end{pmatrix}$, and $|k\rangle = \begin{pmatrix} i \\ 1 \end{pmatrix}$.

Chapter 12
Tensor Product of Operators, Partial Measurement, and Matrix Representation in a Given Basis

12.1 Learning Outcomes

Be more skillful in tensor product operations; understand how to perform tensor product for matrices; understand the meaning of partial measurement and normalization after measurement; understand the meaning of the operator matrix elements in a given basis.

12.2 Tensor Product of Vectors in General Form

We already discussed how to perform tensor product for two qubits (two vectors from two $\mathbb{C}^2$ spaces). Let us try to see how to perform tensor product for two vectors from arbitrary/higher-dimensional spaces, which is just a natural extension of what we have learned. Let $\vec{V} = V_0\hat{e}_0 + V_1\hat{e}_1 + \cdots + V_{n-1}\hat{e}_{n-1}$ in an n-dimensional space and $\vec{W} = W_0\hat{f}_0 + W_1\hat{f}_1 + \cdots + W_{m-1}\hat{f}_{m-1}$ in a m-dimensional space. $\hat{e}_0, \cdots, \hat{e}_{n-1}$ and $\hat{f}_0, \cdots, \hat{f}_{m-1}$ are the basis vectors in the corresponding space.

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_12

I write it in regular vector form but definitely, I can write it in bra–ket notation or column form, too. I hope you feel very comfortable about this now. That is

$$\left|\vec{V}\right\rangle = V_0\left|\hat{e}_0\right\rangle + V_1\left|\hat{e}_1\right\rangle + \cdots + V_{n-1}\left|\hat{e}_{n-1}\right\rangle = \begin{pmatrix} V_0 \\ V_1 \\ \vdots \\ V_{n-1} \end{pmatrix}$$

$$\left|\vec{W}\right\rangle = W_0\left|\hat{f}_0\right\rangle + W_1\left|\hat{f}_1\right\rangle + \cdots + W_{m-1}\left|\hat{f}_{m-1}\right\rangle = \begin{pmatrix} W_0 \\ W_1 \\ \vdots \\ W_{m-1} \end{pmatrix} \tag{12.1}$$

Again, we need to remind ourselves which basis we are using when we write the vectors in the column form. For $\left|\vec{V}\right\rangle$, $\hat{e}_0, \ldots, \hat{e}_{n-1}$ are used and for $\left|\vec{W}\right\rangle$, $\hat{f}_0, \ldots, \hat{f}_{m-1}$ are used. If we perform a tensor product of $\left|\vec{V}\right\rangle \otimes \left|\vec{W}\right\rangle$, what do we get?

$$\begin{aligned} &\left|\vec{V}\right\rangle \otimes \left|\vec{W}\right\rangle \\ &= \left(V_0\left|\hat{e}_0\right\rangle + V_1\left|\hat{e}_1\right\rangle + \cdots + V_{n-1}\left|\hat{e}_{n-1}\right\rangle\right) \otimes \left(W_0\left|\hat{f}_0\right\rangle + W_1\left|\hat{f}_1\right\rangle + \cdots + W_{m-1}\left|\hat{f}_{m-1}\right\rangle\right) \\ &= V_0W_0\left|\hat{e}_0\right\rangle\left|\hat{f}_0\right\rangle + V_0W_1\left|\hat{e}_0\right\rangle\left|\hat{f}_1\right\rangle + \cdots + V_1W_0\left|\hat{e}_1\right\rangle\left|\hat{f}_0\right\rangle + \\ &\quad \cdots + V_{n-1}W_{m-1}\left|\hat{e}_{n-1}\right\rangle\left|\hat{f}_{m-1}\right\rangle \end{aligned} \tag{12.2}$$

Note that $\left|\hat{e}_i\right\rangle\left|\hat{f}_j\right\rangle = \left|\hat{e}_i\right\rangle \otimes \left|\hat{f}_j\right\rangle$. So how many new basis vectors do we have now? They are $\left|\hat{e}_0\right\rangle\left|\hat{f}_0\right\rangle$, $\left|\hat{e}_0\right\rangle\left|\hat{f}_1\right\rangle$, ..., $\left|\hat{e}_1\right\rangle\left|\hat{f}_0\right\rangle$, ..., $\left|\hat{e}_{n-1}\right\rangle\left|\hat{f}_{m-1}\right\rangle$. They are the permutation of the basis vectors from the two spaces and there are $n \times m$ new basis vectors. So the new space is $n \times m$ -dimensional. We can see this from the column vector multiplication in a tensor product we learned in the previous chapter also,

$$\left|\vec{V}\right\rangle \otimes \left|\vec{W}\right\rangle = \begin{pmatrix} V_0 \\ V_1 \\ \vdots \\ V_{n-1} \end{pmatrix} \otimes \begin{pmatrix} W_0 \\ W_1 \\ \vdots \\ W_{m-1} \end{pmatrix}$$

$$= \begin{pmatrix} V_0 \begin{pmatrix} W_0 \\ \vdots \\ W_{m-1} \end{pmatrix} \\ \vdots \\ V_{n-1} \begin{pmatrix} W_0 \\ \vdots \\ W_{m-1} \end{pmatrix} \end{pmatrix} = \begin{pmatrix} V_0 W_0 \\ \vdots \\ V_0 W_{m-1} \\ \vdots \\ V_{n-1} W_0 \\ \vdots \\ V_{n-1} W_{m-1} \end{pmatrix} \tag{12.3}$$

It is clear that the final column vector has $n \times m$ entries. Again when we write $\left|\vec{V}\right\rangle \otimes \left|\vec{W}\right\rangle$ in this column form, what basis are we using? Yes, we are using the new basis with basis vectors $|\hat{e}_0\rangle \left|\hat{f}_0\right\rangle$, $|\hat{e}_0\rangle \left|\hat{f}_1\right\rangle$, …, $|\hat{e}_1\rangle \left|\hat{f}_0\right\rangle$, …, $|\hat{e}_{n-1}\rangle \left|\hat{f}_{m-1}\right\rangle$. Although these all look trivial, I hope you appreciate their connections and meanings.

Finally, I just want to let you know that sometimes the tensor product is written in matrix form to make it more compact. We will not do this in the rest of the book but you might see this elsewhere. Please make sure you understand that this is **NOT** an operator. For example,

$$\left|\vec{V}\right\rangle \otimes \left|\vec{W}\right\rangle = \begin{pmatrix} V_0 \\ \vdots \\ V_{n-1} \end{pmatrix} \begin{pmatrix} W_0 \cdots W_{m-1} \end{pmatrix} = \begin{pmatrix} V_0 W_0 & \cdots & V_0 W_{m-1} \\ \vdots & \ddots & \vdots \\ V_{n-1} W_0 & \cdots & V_{n-1} W_{m-1} \end{pmatrix} \tag{12.4}$$

In this type of notation, the second vector is written in row form. And the basis vector for the i-th row and j-th column element in the matrix is $|\hat{e}_i\rangle \left|\hat{f}_j\right\rangle$ (again, it is very important to know which basis vector each element corresponds to). If this looks trivial to you, I think you have a good understanding of the basis vectors now. Finally, let me emphasize one more time, Eq. (12.4) is NOT an operator and we will not use this representation in the rest of the book.

12.3 Tensor Product of Operators

In each Hilbert space, it has its own operators. Physically, if there are two electrons, there will be two separate operators used to manipulate the spin of the electrons (i.e. to rotate the vector in the $\mathbb{C}^2$ space). We use a magnetic field to manipulate the spin of the electron. So we need two magnetic fields to manipulate the spin of two electrons. In the $\mathbb{C}^2$ spaces, it just means that there are two matrices to rotate the vectors, one in each space and each corresponds to a different set of basis vectors. If we consider the two electrons as a whole system, we can use the tensor product to construct the state in the larger system. How do we construct an operator

to describe the manipulation of the combined vector? That is how do we describe the two magnetic fields used to control the two electrons as a whole? We still use the tensor product.

Assume $\mathcal{H}_3 = \mathcal{H}_1 \otimes \mathcal{H}_2$, and $\boldsymbol{U_1}$ and $\boldsymbol{U_2}$ are two operators in the two $\mathbb{C}^2$ spaces of $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively. Assume,

$$\boldsymbol{U_1} = \begin{pmatrix} a & c \\ b & d \end{pmatrix}; \quad \boldsymbol{U_2} = \begin{pmatrix} e & g \\ f & h \end{pmatrix} \tag{12.5}$$

The corresponding operator in the $\mathcal{H}_3$ space is $\boldsymbol{U} = \boldsymbol{U}_1 \otimes \boldsymbol{U}_2$, where

$$\begin{aligned} \boldsymbol{U} = \boldsymbol{U}_1 \otimes \boldsymbol{U}_2 &= \begin{pmatrix} a & c \\ b & d \end{pmatrix} \otimes \begin{pmatrix} e & g \\ f & h \end{pmatrix} \\ &= \begin{pmatrix} a \begin{pmatrix} e & g \\ f & h \end{pmatrix} & c \begin{pmatrix} e & g \\ f & h \end{pmatrix} \\ b \begin{pmatrix} e & g \\ f & h \end{pmatrix} & d \begin{pmatrix} e & g \\ f & h \end{pmatrix} \end{pmatrix} = \begin{pmatrix} ae & ag & ce & cg \\ af & ah & cf & ch \\ be & bg & de & dg \\ bf & bh & df & dh \end{pmatrix} \begin{matrix} |00'\rangle \\ |01'\rangle \\ |10'\rangle \\ |11'\rangle \end{matrix} \end{aligned} \tag{12.6}$$

Please treat this as a definition and just accept the methodology of finding the tensor product of matrices. I will not prove here. I also labeled the basis states next to the matrix so you know which basis vector each row corresponds to. I also label the second electron state with a "′". Let us try an example and try to gain more insights.

Assume both electrons have an up-spin, i.e. $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|0'\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$. If we consider each electron individually and *apply operators on the vectors in the individual* $\mathbb{C}^2$ *spaces*,

$$\begin{aligned} \boldsymbol{U}_1 |0\rangle &= \begin{pmatrix} a & c \\ b & d \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix} = a |0\rangle + b |1\rangle \\ \boldsymbol{U}_2 |0'\rangle &= \begin{pmatrix} e & g \\ f & h \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} e \\ f \end{pmatrix} = e |0'\rangle + f |1'\rangle \end{aligned} \tag{12.7}$$

We can then form a tensor product to see how the final vector looks like in the $\mathbb{C}^4$ space,

$$\begin{aligned} \boldsymbol{U}_1 |0\rangle \otimes \boldsymbol{U}_2 |0'\rangle &= (a |0\rangle + b |1\rangle) \otimes (e |0'\rangle + f |1'\rangle) \\ &= \begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} e \\ f \end{pmatrix} = \begin{pmatrix} a \begin{pmatrix} e \\ f \end{pmatrix} \\ b \begin{pmatrix} e \\ f \end{pmatrix} \end{pmatrix} = \begin{pmatrix} ae \\ af \\ be \\ bf \end{pmatrix} \begin{matrix} |00'\rangle \\ |01'\rangle \\ |10'\rangle \\ |11'\rangle \end{matrix} \end{aligned} \tag{12.8}$$

Again, I highlighted the corresponding basis states of each element in the two-electron system. Now, I can also use $\boldsymbol{U} = \boldsymbol{U}_1 \otimes \boldsymbol{U}_2$ and operate directly on the tensor product of $|0\rangle$ and $|0'\rangle$ (the initial states). Firstly, I need to create the tensor product of $|0\rangle$ and $|0'\rangle$,

$$|0\rangle \otimes |0'\rangle = |00'\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ 0 \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \begin{matrix} |00'\rangle \\ |01'\rangle \\ |10'\rangle \\ |11'\rangle \end{matrix} \tag{12.9}$$

Then apply $\boldsymbol{U}$ to $|00'\rangle$, i.e. to *apply an operator on a vector in the* $\mathbb{C}^4$ *space.*

$$\boldsymbol{U}|00'\rangle = \begin{pmatrix} ae & ag & ce & cg \\ af & ah & cf & ch \\ be & bg & de & dg \\ bf & bh & df & dh \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} ae \\ af \\ be \\ bf \end{pmatrix} \begin{matrix} |00'\rangle \\ |01'\rangle \\ |10'\rangle \\ |11'\rangle \end{matrix} \tag{12.10}$$

The result is exactly the same as Eq. (12.8). That makes us more confident in the method of constructing the tensor product of operators.

In general, the tensor product of an n-dimensional operator and a m-dimensional operator is an $n \times m$-dimensional operator.

$$\begin{aligned} &\begin{pmatrix} a_{0,0} & \cdots & a_{0,n-1} \\ \vdots & \ddots & \vdots \\ a_{n-1,0} & \cdots & a_{n-1,n-1} \end{pmatrix} \otimes \begin{pmatrix} b_{0,0} & \cdots & b_{0,m-1} \\ \vdots & \ddots & \vdots \\ b_{m-1,0} & \cdots & b_{m-1,m-1} \end{pmatrix} \\ &= \begin{pmatrix} a_{0,0} \begin{pmatrix} b_{0,0} & \cdots & b_{0,m-1} \\ \vdots & \ddots & \vdots \\ b_{m-1,0} & \cdots & b_{m-1,m-1} \end{pmatrix} & \cdots & a_{0,n-1} \begin{pmatrix} b_{0,0} & \cdots & b_{0,m-1} \\ \vdots & \ddots & \vdots \\ b_{m-1,0} & \cdots & b_{m-1,m-1} \end{pmatrix} \\ \vdots & \ddots & \vdots \\ a_{n-1,0} \begin{pmatrix} b_{0,0} & \cdots & b_{0,m-1} \\ \vdots & \ddots & \vdots \\ b_{m-1,0} & \cdots & b_{m-1,m-1} \end{pmatrix} & \cdots & a_{n-1,n-1} \begin{pmatrix} b_{0,0} & \cdots & b_{0,m-1} \\ \vdots & \ddots & \vdots \\ b_{m-1,0} & \cdots & b_{m-1,m-1} \end{pmatrix} \end{pmatrix} \\ &= \begin{pmatrix} a_{0,0}b_{0,0} & \cdots & a_{0,n-1}b_{0,m-1} \\ \vdots & \ddots & \vdots \\ a_{n-1,0}b_{m-1,0} & \cdots & a_{n-1,n-1}b_{m-1,m-1} \end{pmatrix} \end{aligned} \tag{12.11}$$

12.4 Partial Measurement

If I use a certain method to measure the spin of both electrons in the $\mathbb{C}^4$ space, based on what we have learned so far, it will randomly collapse to one of the basis states. For example, for the vector in Eq. (11.12), $|h\rangle_3 = \alpha_3 |00\rangle + \beta_3 |01\rangle + \gamma_3 |10\rangle + \delta_3 |11\rangle$, if I measure $|h\rangle_3$, the probability it will collapse to $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$ is $|\alpha_3|^2$, $|\beta_3|^2$, $|\gamma_3|^2$, and $|\delta_3|^2$, respectively. To enhance our understanding of the concept of different spaces, we can also find it by using the projection operator. For example, to find the probability to collapse to $|10\rangle$ state, we can use the projector $\boldsymbol{P}_{|10\rangle} = |10\rangle \langle 10|$ (see Eqs. (8.15)) and (8.20),

$$\begin{aligned}
&\langle h | \boldsymbol{P}_{|10\rangle} | h \rangle \\
&= (\alpha_3^* \langle 00| + \beta_3^* \langle 01| + \gamma_3^* \langle 10| + \delta_3^* \langle 11|) |10\rangle \langle 10| (\alpha_3 |00\rangle + \beta_3 |01\rangle + \gamma_3 |10\rangle + \delta_3 |11\rangle) \\
&= \gamma_3^* \langle 10| \ |10\rangle \langle 10| \gamma_3 |10\rangle \\
&= \gamma_3^* \gamma_3 = |\gamma_3|^2
\end{aligned} \tag{12.12}$$

This is exactly what we expect. Here we have used the orthonormal properties of the basis vectors. We can say $|10\rangle$ is orthogonal to $|00\rangle$, $|01\rangle$, and $|11\rangle$ because they are different basis vectors. Or we can use the fact that $|10\rangle = |1\rangle \otimes |0\rangle$, then, for example,

$$\begin{aligned}
\langle 11| \ |10\rangle &= (\langle 1|_1 \otimes \langle 1|_2)(|1\rangle_1 \otimes |0\rangle_2) \\
&= \langle 1|_1 |1\rangle_1 \times \langle 1|_2 |0\rangle_2 = 1 \times 0 = 0
\end{aligned} \tag{12.13}$$

Here, when we perform the inner product, we can do it in their individual space. *This is because there is no overlap between the vectors in different spaces.*

We may also ask what is the probability to collapse to $|00\rangle$ *or* $|01\rangle$ states and we can use the projector $\boldsymbol{P}_{|10\rangle|01\rangle} = |00\rangle \langle 00| + |01\rangle \langle 01|$ to find it. Similarly, we will get $|\alpha_3|^2 + |\beta_3|^2$.

We may also only perform a measurement on one of the electrons, e.g. electron 1. Let us remind ourselves that the vector (state) for electron 1 can be written as $|f\rangle_1 = \alpha_1 |0\rangle_1 + \beta_1 |1\rangle_1$ and that for electron 2 as $|g\rangle_2 = \alpha_2 |0\rangle_2 + \beta_2 |1\rangle_2$. If after measurement, electron 1 collapses to $|0\rangle_1$, it will become $|f\rangle_1 = \alpha_1 |0\rangle_1$ (Not exactly correct because I have not normalized the state) and only $|00\rangle$ and $|01\rangle$ will remain in the whole system. This is called the **partial measurement**. What is the probability that electron 1 will collapse to $|0\rangle_1$ in this partial measurement? Since after electron collapses to $|0\rangle_1$, only $|00\rangle$ and $|01\rangle$ are left in the *whole* system, so it should be the same as asking "what is the probability to collapse to the $|00\rangle$ *or* $|01\rangle$ states" after the measurement? Therefore, it is $|\alpha_3|^2 + |\beta_3|^2$.

Let us use a second method to check if this is true by using the projector operator, $\boldsymbol{P}_{|0\rangle_1} = |0\rangle_1 \langle 0|_1$ to confirm.

$$\begin{aligned}
&\langle h | \boldsymbol{P}_{|0\rangle_1} | h \rangle \\
&= (\alpha_3^* \langle 00| + \beta_3^* \langle 01| + \gamma_3^* \langle 10| + \delta_3^* \langle 11|) |0\rangle_1 \langle 0|_1 (\alpha_3 |00\rangle + \beta_3 |01\rangle + \gamma_3 |10\rangle + \delta_3 |11\rangle) \\
&= \alpha_3^* \langle 00| \ |0\rangle_1 \langle 0|_1 \alpha_3 |00\rangle + \beta_3^* \langle 01| \ |0\rangle_1 \langle 0|_1 \beta_3 |01\rangle \\
&= |\alpha_3|^2 + |\beta_3|^2
\end{aligned} \tag{12.14}$$

This is the same as expected. Note that we need to calculate all the cross terms but due to the orthonormal property of the basis vectors and also basis vectors in spaces 1 and 2 do not have overlap with each other, the calculation is greatly simplified. For example, $\alpha_3^* \langle 00| \ |0\rangle_1 \langle 0|_1 \alpha_3 |00\rangle = \alpha_3^* \langle 0|_1 \langle 0|_2 \ |0\rangle_1 \langle 0|_1 \alpha_3 |0\rangle_1 |0\rangle_2 = |\alpha_3|^2 \langle 0|_1 |0\rangle_1 \langle 0|_2 |0\rangle_2 |0\rangle_1 |0\rangle_1 = |\alpha_3|^2$.

The third way to look into this is that we can also see this as electron 1 firstly collapses to $|0\rangle$ and then performs tensor product with $|g\rangle_2$ to obtain the vector in the $\mathbb{C}^4$ space,

$$\begin{aligned}
|h\rangle_3 &= \alpha_1 |0\rangle_1 \otimes (\alpha_2 |0\rangle_2 + \beta_2 |1\rangle_2) \\
&= \alpha_1\alpha_2 |0\rangle_1 |0\rangle_2 + \alpha_1\beta_2 |0\rangle_1 |1\rangle_2 \\
&= \alpha_3 |0\rangle_1 |0\rangle_2 + \beta_3 |0\rangle_1 |1\rangle_2
\end{aligned} \tag{12.15}$$

However, this is not enough. We need to normalize the vector which is required so the probabilities of obtaining $|0\rangle_1 |0\rangle_2$ and $|0\rangle_1 |1\rangle_2$ sum up to 1. To do this, we use Eq. (4.18),

$$|h'\rangle_3 = \frac{|h\rangle_3}{\sqrt{|\alpha_3|^2 + |\beta_3|^2}} = \frac{\alpha_3 |0\rangle_1 |0\rangle_2 + \beta_3 |0\rangle_1 |1\rangle_2}{\sqrt{|\alpha_3|^2 + |\beta_3|^2}} \tag{12.16}$$

In this example, you see that a partial measurement of electron 1 does not affect the state of electron 2. We can still express the total vector as a tensor product between a vector in space 1, $|0\rangle_1$, and a vector in space 2, $\alpha_2 |0\rangle_2 + \beta_2 |1\rangle_2$. Later, we will see this is not always the case when there is *entanglement*.

12.5 Matrix Representation in a Given Basis

We know that in different bases, a vector has different representations in column or row forms. For example, in Fig. 12.1, $|V\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ in the $|0\rangle$ / $|1\rangle$ basis (black) but $|V\rangle = |+\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ in the $|+\rangle$ / $|-\rangle$ basis (red).

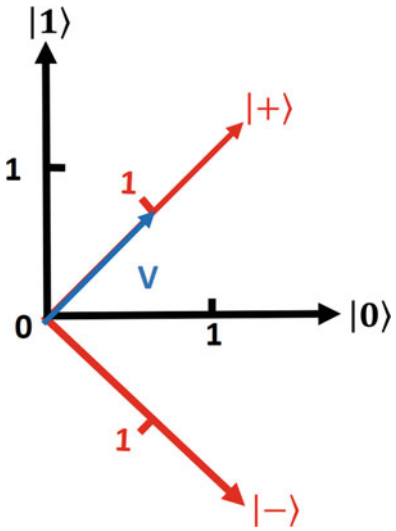


Fig. 12.1 Representations of $|V\rangle$ in two different bases

Naturally, the operator should also have different representations in different bases. How do we represent an operator, **H**, in a given basis? Assume the basis has basis vectors $|\alpha_k\rangle$ where k running from 0 to $n-1$ in an n-dimensional space. The i-th row and j-th column element is given by

$$H_{i,j} = \langle \alpha_i | \boldsymbol{H} | \alpha_j \rangle \tag{12.17}$$

Note that $H_{i,j}$ is a scalar (complex number). You only need to know this equation. Let us try an example just to show that this is correct. For example, what is the representation of $\boldsymbol{\sigma_x}$ in the basis form by the eigenvectors of $\boldsymbol{\sigma_z}$? Firstly, the eigenvectors of $\boldsymbol{\sigma_z}$ form a 2D space. The two eigenvectors are $|\alpha_0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|\alpha_1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. So the matrix must be 2×2, with k running from 0 to 1. For example, $H_{0,0} = \langle \alpha_0 | \boldsymbol{\sigma_x} | \alpha_0 \rangle$. But we need to know $\boldsymbol{\sigma_x}$ in order to evaluate this equation. So, it is a chicken and egg problem. But this equation is very useful in many derivations. And this is a good exercise to check if we understand the concepts well. Since $\boldsymbol{\sigma_x} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, we have

$$\begin{aligned} \boldsymbol{\sigma_x} &= \begin{pmatrix} \langle \alpha_0 | \boldsymbol{\sigma_x} | \alpha_0 \rangle & \langle \alpha_0 | \boldsymbol{\sigma_x} | \alpha_1 \rangle \\ \langle \alpha_1 | \boldsymbol{\sigma_x} | \alpha_0 \rangle & \langle \alpha_1 | \boldsymbol{\sigma_x} | \alpha_1 \rangle \end{pmatrix} \\ &= \begin{pmatrix} (1\ 0) \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} & (1\ 0) \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ (0\ 1) \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} & (0\ 1) \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{aligned} \tag{12.18}$$

I hope this exercise has reinforced your understanding of the matrix-vector operation and made the basis concept even clearer.

Now, I would like to derive Eq. (12.17). This is a very good exercise. Firstly, for any matrix, I can multiply it by the identity operator at will.

$$\boldsymbol{H} = \boldsymbol{I}\boldsymbol{H}\boldsymbol{I} \tag{12.19}$$

Substituting Eq. (10.16), $\boldsymbol{I} = \sum_{i=0}^{n-1} |i\rangle \langle i|$, it becomes

$$\begin{aligned}\boldsymbol{H} &= \left(\sum_{i=0}^{n-1} |i\rangle \langle i|\right) \boldsymbol{H} \left(\sum_{j=0}^{n-1} |j\rangle \langle j|\right) \\ &= \sum_{j=0}^{n-1}\sum_{i=0}^{n-1} |i\rangle \langle i| \boldsymbol{H} |j\rangle \langle j| \\ &= \sum_{j=0}^{n-1}\sum_{i=0}^{n-1} |i\rangle \langle j| \langle i| \boldsymbol{H} |j\rangle \end{aligned} \tag{12.20}$$

Note that $\langle i| \boldsymbol{H} |j\rangle$ is just a scalar and we have also used the associative axiom in the derivation. What is $|i\rangle \langle j|$? It is just

$$|i\rangle \langle j| = \begin{pmatrix} 0 \\ \vdots \\ 1(i-\text{th}) \\ \vdots \end{pmatrix} \begin{pmatrix} 0 \cdots 1(j-\text{th}) \cdots \end{pmatrix} \tag{12.21}$$

which means it will create a matrix with zero everywhere except the i-th row and j-th column and then it is multiplied by $\langle i| \boldsymbol{H} |j\rangle$. All these matrices are added to form the final $\boldsymbol{H}$. Therefore, it is true that $H_{i,j} = \langle \alpha_i | \boldsymbol{H} | \alpha_j \rangle$.

12.6 Summary

We are more familiar with tensor product operation on vector and matrix now. Sometimes we can also decompose the vector in a higher-dimensional space to the tensor product of two vectors in two lower-dimensional spaces. But this is not always possible and we will see that when we discuss entanglement. We also studied partial measurement which is the measurement of a sub-space vector. And we learned how to construct a matrix from a given basis.

Problems

12.1 Hand Calculation of Tensor Product

Let $|f\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ and $|g\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} \end{pmatrix}$, find $\boldsymbol{\sigma_x}\,|f\rangle_1$ and $\boldsymbol{\sigma_x}\,|g\rangle_2$. Then find the tensor product of the resulting vector (i.e. $\boldsymbol{\sigma_x}\,|f\rangle_1 \otimes \boldsymbol{\sigma_x}\,|g\rangle_2$). Now find $|f\rangle_1 \otimes |g\rangle_2$ and construct $\boldsymbol{\sigma_x} \otimes \boldsymbol{\sigma_x}$. Show that $\boldsymbol{\sigma_x} \otimes \boldsymbol{\sigma_x}(|f\rangle_1 \otimes |g\rangle_2)$ gives the same result.

12.2 Tensor Product Calculation using Colab

Use Colab and the function $np.kron$ to verify Problem 12.1.

Part II
Quantum Computing: Gates and Algorithms

Chapter 13
Quantum Register and Data Processing, Entanglement, the Bell States, and EPR Paradox

13.1 Learning Outcomes

Understand the similarity and difference between a quantum register and a classical register; be more familiar with the tensor and inner products between multiple qubits; better understanding on why the quantum computer is powerful; understand entanglement and the Bell States; appreciate Einstein–Podolsky–Rosen (EPR) paradox.

13.2 Quantum Register

In a classical computer, a register is a memory to store the classical bits of information. The size of the register determines the number of bits it can store. For example, a 4-bit register can store 4 bits of information, such as $(1001)_2$. The bit on the left is called the **Most Significant Bit** (MSB) and the bit on the right is called the **Least Significant Bit** (LSB). Note that $(1001)_2$ is the binary representation of the value being stored and the binary representation is indicated by the subscript "$_2$". We know that the MSB represents $1 \times 2^3 = 8$ and the least significant bit represents $1 \times 2^0 = 1$. The decimal value of this representation is $1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 9$. In general, an n-bit register can store n binary digits, a_i, where i runs from 0 to $n - 1$. Of course, a_i can only be "0" or "1" as it is binary. The decimal value stored in an n-bit register can be found using the following equation.

$$(a_{n-1}a_{n-2}\cdots a_0)_2 = a_{n-1} \times 2^{n-1} + a_{n-2} \times 2^{n-2} + \cdots + a_0 \times 2^0 = \sum_{i=0}^{n-1} a_i \times 2^i \tag{13.1}$$

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_13

While you can easily imagine how to implement an n-bit classical register physically (e.g. using 4 coins as a 4-bit register with the "head" and "tail" storing the "0' and "1', respectively), a register can be an abstract concept and we can use them in the computing algorithm without mentioning or even not knowing its physical implementation. Similarly, we can discuss and use **quantum register** in quantum algorithm without knowing how to implement a quantum register. And most of the time, a quantum register is only a *grouping of qubits* and it is not necessarily a memory as in the classical case. Therefore, in the rest of this book, we will just treat a register as a convenient way to group the qubits.

A quantum register is just a natural extension of the concept of a classical register. An n-bit classical register stores n-bit of information while an n-qubit quantum register stores n-qubit of information. For a 1-qubit register, it stores 1-qubit of information that represents a **vector** in the $\mathbb{C}^2$ space (see Sect. 11.4). We can write it as $|a\rangle$ which, in general, is a linear combination of the basis vectors, $|0\rangle$ and $|1\rangle$. That is, $|a\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, with which we are very familiar already. We see that when $|a\rangle = |0\rangle$ or $|a\rangle = |1\rangle$, the quantum register stores exactly the same information as the classical one, i.e. $(0)_2$ or $(1)_2$. *This is the most direct relationship between the quantum register and the classical register*. **If the quantum register is only allowed to store the basis states, it is the same as a classical register!**

As mentioned, *a quantum register is just a grouping of qubits*, all the knowledge we learned earlier is directly applicable to the qubits stored in a quantum register. For a 4-qubit quantum register, it stores the information of 4 qubits, which is a vector in the $\mathbb{C}^{2^4} = \mathbb{C}^{16}$ space. This is obtained by performing the tensor products of 4 individual qubits, each from a $\mathbb{C}^2$ space.

$$\mathbb{C}^{16} = \mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \mathbb{C}^2 = \mathbb{C}^{2^4} \tag{13.2}$$

In this register, it can store any vectors that are the linear combinations of the basis states. And the basis states are just the possible values that can be stored in a classical register. Table 13.1 shows the basis states of a 4-qubit register and their relationship to the values that can be stored in a classical register.

The *vector* or *state*, $|\Psi\rangle$ that can be stored in a 4-qubit register is therefore a linear combination of the basis states listed in Table 13.1.

$$\begin{aligned} |\Psi\rangle &= a_0\,|0\rangle \otimes |0\rangle \otimes |0\rangle \otimes |0\rangle + a_1\,|0\rangle \otimes |0\rangle \otimes |0\rangle \otimes |1\rangle + \cdots + a_{15}\,|1\rangle \otimes |1\rangle \otimes |1\rangle \otimes |1\rangle \\ &= a_0\,|0\rangle_{10} + a_1\,|1\rangle_{10} + \cdots + a_{15}\,|15\rangle_{10} \end{aligned} \tag{13.3}$$

Here we see the power of the quantum register. Instead of being able to store *only* the basis states (which is equivalent to the capability of a classical register), it can store the linear combination of the basis states. And *the number of basis states, thus the dimension of the* ***Hilbert space****, grows exponentially with the number of qubits!*

Table 13.1 Relationship between storable values in a 4-bit classical register and the basis states in a 4-qubit quantum register

Value can be stored in a classical register	Basis states in a quantum register
$(0000)_2 = 0$	$\|0\rangle \otimes \|0\rangle \otimes \|0\rangle \otimes \|0\rangle = \|0\rangle\|0\rangle\|0\rangle\|0\rangle = \|0000\rangle = \|0\rangle_{10}$
$(0001)_2 = 1$	$\|0\rangle \otimes \|0\rangle \otimes \|0\rangle \otimes \|1\rangle = \|0\rangle\|0\rangle\|0\rangle\|1\rangle = \|0001\rangle = \|1\rangle_{10}$
$(0010)_2 = 2$	$\|0\rangle \otimes \|0\rangle \otimes \|1\rangle \otimes \|0\rangle = \|0\rangle\|0\rangle\|1\rangle\|0\rangle = \|0010\rangle = \|2\rangle_{10}$
$\vdots$	$\vdots$
$(1111)_2 = 15$	$\|1\rangle \otimes \|1\rangle \otimes \|1\rangle \otimes \|1\rangle = \|1\rangle\|1\rangle\|1\rangle\|1\rangle = \|1111\rangle = \|15\rangle_{10}$

For a 4-qubit register, it stores a vector in the $2^4 = 16$ dimensional space. For an n-qubit register, it stores a vector in the 2^n dimensional space.

$$\begin{aligned}|\Psi\rangle &= a_0\,|00\cdots0\rangle + a_1\,|00\cdots1\rangle + \cdots + a_{2^{n-1}}\,|11\cdots1\rangle \\ &= a_0\,|0\rangle_{10} + a_1\,|1\rangle_{10} + \cdots + a_{2^{n-1}}\left|2^{n-1}\right\rangle_{10} \\ &= \sum_{i=0}^{2^{n-1}} a_i\,|i\rangle_{10} \\ &= \sum_{i=0}^{2^{n-1}} a_i\,|i\rangle \end{aligned} \tag{13.4}$$

We drop the subscript "$_{10}$" in the last line when there is no ambiguity and we know anything inside the ket is just a name.

The basis states in the 2^n-dimensional vector space are orthonormal to each other. This is because for two basis states $|a\rangle = |a_{n-1}a_{n-2}\cdots a_0\rangle$ and $|b\rangle = |b_{n-1}b_{n-2}\cdots b_0\rangle$ where a_i and b_i can only take either "0" or "1", based on Table 13.1, we have

$$\begin{aligned}\langle a|b\rangle &= \langle a_{n-1}a_{n-2}\cdots a_0|\,|b_{n-1}b_{n-2}\cdots b_0\rangle \\ &= \langle a_{n-1}|b_{n-1}\rangle\,\langle a_{n-2}|b_{n-2}\rangle\cdots\langle a_0|b_0\rangle \end{aligned} \tag{13.5}$$

$\langle a|b\rangle$ must be 0 if $a \neq b$ as they are different in at least one binary digit or 1 if $a = b$.

Let us work on one more example to understand the notations in Eq. (13.3) better. To make it simple, we will only use 2-qubit. A 2-qubit register stores a state in the $\mathbb{C}^4$ Hilbert space which is a tensor product space of two 1-qubit systems ($\mathbb{C}^2 \otimes \mathbb{C}^2$). A vector example in this space is,

$$|\Psi\rangle = 0.5\,|0\rangle + 0.5\,|1\rangle + 0.5\,|2\rangle + 0.5\,|3\rangle \tag{13.6}$$

Now we understand $|0\rangle$, $|1\rangle$, $|2\rangle$, and $|3\rangle$ are the basis vectors and they are written in decimal representation. In binary form they are $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$ which are the tensor products of the basis states of two $\mathbb{C}^2$ spaces, i.e. $|0\rangle \otimes |0\rangle$, $|0\rangle \otimes |1\rangle$, $|1\rangle \otimes |0\rangle$, and $|1\rangle \otimes |1\rangle$. The probability of finding the wavefunction to collapse to $|2\rangle$ is $0.5^2 = 0.25$ (i.e. the square of the magnitude of the corresponding coefficient, Eq. (4.17)). But what is the physical meaning of this? It is the same as finding the system to collapse to $|1\rangle \otimes |0\rangle$. For example, if this is a two-electron spin system, we will find that the first electron collapses to the spin down state and the second electron collapses to the spin up state.

These are important notions and we will use them a lot in the quantum algorithms.

13.3 Quantum Data Processing

You see that an n-bit register can store tremendous information, i.e. a vector in the 2^n dimensional space. Here I want to give a **fake** example to show the power of quantum data processing. If I want to calculate a function, $f(x)$, for $x = 0, 1, 2, 3$ in a classical computer. I need to do it 4 times by calculating $f(0)$, $f(1)$, $f(2)$, and $f(3)$. I also need to store $x = 0, 1, 2, 3$ in 4 registers before the calculation. However, in quantum computing, I just need to form a vector $|\Psi\rangle$, which is a linear superposition of $|0\rangle$, $|1\rangle$, $|2\rangle$, and $|3\rangle$ as in Eq. (13.6) and store it in one 4-qubit quantum register. And I will try to find a mechanism with a linear property so that I get

$$\begin{aligned} f(|\Psi\rangle) &= f(0.5\,|0\rangle + 0.5\,|1\rangle + 0.5\,|2\rangle + 0.5\,|3\rangle) \\ &= 0.5 f(|0\rangle) + 0.5 f(|1\rangle) + 0.5 f(|2\rangle) + 0.5 f(|3\rangle) \end{aligned} \tag{13.7}$$

Then all the computations can be done at once!

In summary, the power of quantum computing has two folds. Firstly, it only requires n-qubit to store a 2^n-dimensional space vector. If $n = 100$, we only need, for example, 100 electrons. But in a classical computer, to store a 2^{100}-dimensional space vector, we need to store its 2^{100} complex coefficients. This requires way more than the amount of classical memory we can imagine. Secondly, with the linearity of quantum mechanics, we can perform the calculation of the basis states simultaneously. If $n = 100$, we can perform 2^{100} calculations at one shot but this may take forever in a classical computer. This is the famous **quantum parallelism**. And based on Eq. (13.7), we see that this is due to the linearity and **superposition** properties in quantum mechanics.

In most important quantum algorithms, we utilize these two properties to perform gigantic calculations without the need for an enormous amount of hardware and by exploiting quantum parallelism. However, we should also realize that even we can store 2^n coefficients and perform 2^n computations easily in a quantum computer, it is difficult to read them due to *wavefunction collapse* after a single measurement. As

a result, most quantum algorithms exploit the interference between the basis states to obtain the desired results. You probably do not understand what I mean now. I just want you to remember that while quantum parallelism seems powerful, to fetch information to be useful *in our classical world*, we need to apply some careful tricks which will be obvious when we discuss the quantum algorithms.

13.4 Entanglement and Bell States

Besides *superposition* and *interference*, **entanglement** is also a very heavily used concept and property in quantum computing algorithms. We know that every higher-dimensional Hilbert space, e.g. $\mathcal{H}_3$, can be constructed by the tensor products of some lower-dimensional spaces, e.g. $\mathcal{H}_2$ and $\mathcal{H}_1$. For example, $\mathbb{C}^4 = \mathbb{C}^2 \otimes \mathbb{C}^2$. Any vector in the higher-dimensional space, e.g. $|h\rangle$, can be expressed as the *linear combination* of the tensor products of lower-dimensional space vectors (Eq. (11.18)), e.g. $|f_i\rangle$ and $\left|g_j\right\rangle$. For convenience, the equation is repeated here:

$$\begin{aligned} |h\rangle &= \sum_{j=1}^{n} |f_i\rangle \otimes \left|g_j\right\rangle \\ &= |f_1\rangle \otimes |g_1\rangle + |f_2\rangle \otimes |g_2\rangle + \cdots + |f_n\rangle \otimes |g_n\rangle \end{aligned} \tag{13.8}$$

It is possible that for some higher-dimensional vectors, they can be expressed as just one single tensor product of two lower-dimensional vectors, each from a lower-dimensional space. That is

$$|h\rangle = |f\rangle \otimes |g\rangle \tag{13.9}$$

In this case, n in Eq. (13.8) equals 1. These vectors are **unentangled**, which is not special. They are called **separable** or **product** states. However, there are some higher-dimensional vectors *cannot* be expressed as a single product of two lower-dimensional vectors and must be expressed as the linear combination of them as in Eq. (13.8) with $n > 1$. Then we say the lower-dimensional vectors are **entangled**, which means that they cannot be separated. We also say that a higher-dimensional vector is *entangled* and cannot be unentangled to be represented as a simple tensor product of two lower-dimensional vectors.

Let us try an example in the $\mathbb{C}^4 = \mathbb{C}^2 \otimes \mathbb{C}^2$ space. Firstly, let us try to construct a vector in $\mathbb{C}^4$ using the tensor product of two vectors in the $\mathbb{C}^2$ spaces. Let $|f\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|g\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, we can construct

$$\begin{aligned} |h\rangle &= |f\rangle \otimes |g\rangle \\ &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{aligned}$$

$$= \frac{1}{2}(1 \times 1\,|0\rangle \otimes |0\rangle + 1 \times -1\,|0\rangle \otimes |1\rangle + 1 \times 1\,|1\rangle \otimes |0\rangle + 1 \times -1\,|1\rangle \otimes |1\rangle)$$

$$= \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle) \tag{13.10}$$

Is the vector $|h\rangle = \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)$ entangled? No, because we just showed that it can be expressed as a single tensor product of two lower-dimensional vectors $|f\rangle$ and $|g\rangle$, even it looks so lengthy.

How about the vector $\left|\Phi^+\right\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$? This is less lengthy than $|h\rangle$. This is also a vector in the $\mathbb{C}^4$ space. If we write out all basis, it is $\left|\Phi^+\right\rangle = \frac{1}{\sqrt{2}}(1\,|00\rangle + 0\,|01\rangle + 0\,|10\rangle + 1\,|11\rangle) = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\0\\1\end{pmatrix}$. Can we express it as a tensor product of two vectors from the $\mathbb{C}^2$ spaces? We will assume it can and $\left|\Phi^+\right\rangle = |m\rangle \otimes |n\rangle$, where $|m\rangle = a_0\,|0\rangle + a_1\,|1\rangle$ and $|n\rangle = b_0\,|0\rangle + b_1\,|1\rangle$, then

$$\begin{aligned}\left|\Phi^+\right\rangle &= |m\rangle \otimes |n\rangle \\ &= (a_0\,|0\rangle + a_1\,|1\rangle) \otimes (b_0\,|0\rangle + b_1\,|1\rangle) \\ &= a_0 b_0\,|00\rangle + a_0 b_1\,|01\rangle + a_1 b_0\,|10\rangle + a_1 b_1\,|11\rangle\end{aligned} \tag{13.11}$$

But $\left|\Phi^+\right\rangle = \frac{1}{\sqrt{2}}(1\,|00\rangle + 0\,|01\rangle + 0\,|10\rangle + 1\,|11\rangle)$, so the coefficients of the corresponding basis must be the same as those in Eq. (13.11). Therefore,

$$\begin{aligned}a_0 b_0 &= 1 \\ a_0 b_1 &= 0 \\ a_1 b_0 &= 0 \\ a_1 b_1 &= 1\end{aligned} \tag{13.12}$$

From the second equation, either a_0 or b_1 needs to be 0. Then it means either $a_0 b_0 = 0$ or $a_1 b_1 = 0$ which contradicts the first or the fourth equations, respectively. That means our assumption that $\left|\Phi^+\right\rangle = |m\rangle \otimes |n\rangle$ is wrong and it is impossible to find $|m\rangle$ and $|n\rangle$ such that their tensor product is $\left|\Phi^+\right\rangle$. *Therefore,* $\left|\Phi^+\right\rangle$ *is entangled.*

$\left|\Phi^+\right\rangle$ is special and it is also one of the four Bell states. In the $\mathbb{C}^4$ space, since it is 4-dimensional, it has four basis states. If we think of it as the space of two electron spins, it is natural to construct the basis to be $|\uparrow\uparrow\rangle$, $|\uparrow\downarrow\rangle$, $|\downarrow\uparrow\rangle$, and $|\downarrow\downarrow\rangle$, or in other words, $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. These are convenient basis states. But we can also

create a new basis using the so-called **Bell states**. The Bell states are all entangled and are very useful in certain applications. The Bells states are

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\0\\1\end{pmatrix}; \quad |\Phi^-\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\0\\-1\end{pmatrix}$$
$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\1\\0\end{pmatrix}; \quad |\Psi^-\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\-1\\0\end{pmatrix} \tag{13.13}$$

Firstly, let us ask ourselves again when the Bell states are written in the column vectors in Eq. (13.13), which basis are we using? Yes, it is the basis of $|00\rangle/|01\rangle/|10\rangle/|11\rangle$. The Bell states are all *entangled* and we will prove the rest of them in the Problems.

Let us explore a special property of an entangled state. Suppose we have two electron spins and they form an entangled state, Φ^+, in a system (Fig. 13.1) at time $t = 0\ year$ on the Earth. I then send one of the electrons to my alien friend 100 light-years away (i.e. it is so far that it takes the light 100 years to reach there) and I assume the *state is not destroyed and the two electrons are still entangled*. Let us say I am able to send the electron at half of the speed of the light. So after 200 years, at $t = 200\ year$, my friend receives the electron (No worry, I am still alive because I expect my life expectancy is about 500 years old when the technology is good enough for me to make an alien friend 100 light-years away). At $time = 201\ year$, I decide to measure the spin of my electron by applying a magnetic field in the $\hat{z}$ direction and then measure its energy. What will I get? Of course, I will get either

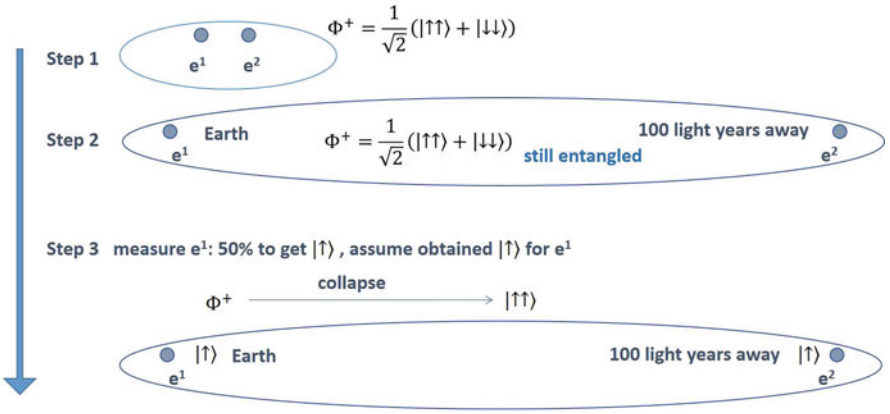


Fig. 13.1 Measurement of an entangled pair of electrons

$|\uparrow\rangle$ or $|\downarrow\rangle$ because these are the two eigenstates of the system with the external magnetic field. And I know that after a measurement, it will randomly collapse to either $|\uparrow\rangle$ or $|\downarrow\rangle$. However, $\left|\Phi^{+}\right\rangle = \frac{1}{\sqrt{2}}(|\uparrow\uparrow\rangle + |\downarrow\downarrow\rangle)$. If my electron's wavefunction collapses to $|\uparrow\rangle$ (or $|\downarrow\rangle$), the whole system must collapse to $|\uparrow\uparrow\rangle$ (or $|\downarrow\downarrow\rangle$) because this is the only non-zero magnitude basis state that contains $|\uparrow\rangle$ (or $|\downarrow\rangle$). This also means that my friend's electron wavefunction will collapse to the same spin state as my electron even my alien friend does not do the measurement and even it takes the light 100 years to tell the other electron what happened to my electron.

This looks like we can transmit information instantaneously and at a speed faster than light. This is related to the **Einstein–Podolsky–Rosen (EPR)** paradox which we will discuss later. However, my experiment does not contradict general relativity which states that information and matters cannot travel faster than light. After more thought, you see that this does not transmit any useful information. For example, I cannot use this information to do something that violates the causality. Assume I buy a pair of gloves from the supermarket, and assume the left and right gloves should have the same color of either blue or yellow. Assume I am color blind and I send one of them to my friends in Hong Kong from San Francisco. When they open the box, they found that it is yellow and they know immediately my glove is yellow while I even do not know what I am wearing as I am color blind. Does this violate general relativity? Of course not. This is just a result of logic. Is this the same in the scenario depicted in Fig. 13.1? To be frank, I am not sure.

13.5 Einstein–Podolsky–Rosen (EPR) Paradox

Albert Einstein who won his Nobel prize due to photon quantization (to explain the photoelectric effect) does not believe in quantum mechanics. He thinks God does not play dice. The randomness in quantum mechanics makes him very uncomfortable. With Podolsky and Rosen, they came up with a paradox to show that quantum mechanics is either wrong or *incomplete*.

This is related to the entanglement experiment I just did with my alien friend. They modified by adding a few more steps. But this time, my alien friend is visiting me so I do not need to wait for hundreds of years. And we are not trying to prove that quantum mechanics will violate general relativity. Let me repeat what was done first. After I sent an electron in the entangled pair of $\left|\Phi^{+}\right\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ to my alien friend in the hotel, I measure my electron in the $|0\rangle$ / $|1\rangle$ basis. I assume I get $|0\rangle$. Then the state of the two-electron system becomes $|00\rangle$. I call my friend and say I got $|0\rangle$. My friend then knows with a 100% certainty that the electron in the hotel must be $|0\rangle$ due to entanglement. Then, my friend decides to perform a measurement on the hotel electron in the $|+\rangle$ / $|-\rangle$ basis. We know $|0\rangle = \frac{1}{\sqrt{2}}(|+\rangle + |-\rangle)$. So the electron must collapse to either $|+\rangle$ or $|-\rangle$. Assume it collapses to $|+\rangle$. Now my friend also knows with a 100% certainty that the electron is in the $|+\rangle$ state. So my friend knows with 100% certainty that the electron is in the $|+\rangle$ and $|0\rangle$ states.

But is not that due to the Heisenberg Uncertainty principle, we cannot measure $|+\rangle$ and $|0\rangle$ states with a 100% certainty as these two bases are incompatible (see Chap. 5)? That means either the quantum mechanics is wrong or incomplete (with hidden variables we still have not discovered).

Einstein is thus not convinced that quantum mechanics is correct. There are experiments related to this (which is to prove the so-called **Bell's Inequality**) and they show that quantum mechanics "looks" correct. It only looks correct because there is a possibility that the experiment is not 100% well controlled. However, the technology has been improving that the chance that it is not well controlled becomes smaller and smaller which gives us more and more confidence that quantum mechanics is correct.

We do not want to spend too much time understanding if the paradox can be solved. But if you understand why the paradox makes Einstein uncomfortable, you do understand many of the important concepts we have studied so far.

13.6 Summary

After learning the basic linear algebra in the previous "steps," we are ready to appreciate the power of quantum register and quantum data processing easily in this "step." We find that quantum computing is powerful because the information it can store goes up exponentially with the number of qubits (2^n) and the number of calculations it can perform simultaneously goes up also exponentially. This is possible due to the superpositions of basis states. We then discuss entanglement and Bell states which are entangled states in the $\mathbb{C}^4$ space. We find that entanglement has a very special property that can easily lead us to believe the information can travel faster than the speed of the light. It even confused Albert Einstein with their EPR paradox. But our goal is not to debate with the giants. As long as we can understand why it looks confusing, it is a big "step" for us already.

Problems

13.1 Power of Quantum Register

At the time of writing, the whole world has less than 10^{21} byte of memory. If we need two bytes to store one coefficient of a quantum state (assume one for the real part and one for the imaginary part), what is the maximum size of the quantum register state the world's memory can store?

13.2 Bell States

Prove all Bell states are entangled.

13.3 Profiting using Entanglement

There is a strange trillionaire living on Pluto. She invests based on the measurement of electron spin every month. If it is a spin up, she will invest in the gold on the Earth. If it is spin down, she will invest in the silver on the Earth. An investor was able to entangle his electron with her electron to be measured next month secretly. Once the measurement is performed, the investor knows that his electron will collapse to the same spin states and he will buy the corresponding commodity before anyone else who receives the result through a radio wave from Pluto. Is this really useful? In other words, is this a riskless investment for the investor on the Earth?

13.4 Inner Product between Two Spaces

What is the inner product between the basis states of two electron spins? For example, what is the inner products of $|0\rangle_1$ and $|0\rangle_0$?

Chapter 14
Concepts Review, Density Matrix, and Entanglement Entropy

14.1 Learning Outcomes

Have a deeper understanding of the linear algebra and quantum mechanics concepts and skills; able to understand and apply the basic concepts and skills in advanced examples; understand the difference between pure and mixed states; and know how to calculate the density matrix and entanglement entropy.

14.2 Concepts Review Using Entanglement

We are done with the basic linear algebra and quantum mechanics that are necessary for us to understand quantum computing algorithms. Here, I would like to use an example of entanglement to review some of the most critical concepts and skills we have learned before moving to the next "steps."

The question we want to ask is that if there is a Bell state on a certain basis, is it possible to express it on another basis so that it is *not entangled*? Or if it is entangled in a certain basis, is it always entangled no matter how you change the basis?

Let us take $\left|\Psi^{-}\right\rangle$ as an example. From Eq. (13.13), $\left|\Psi^{-}\right\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 0 \\ 1 \\ -1 \\ 0 \end{pmatrix}$. This is a 4-dimensional space vector. Which basis are we using when we write it in this column form? We are using the basis formed by $|00\rangle/|01\rangle/|10\rangle/|11\rangle$ in the $\mathbb{C}^4$ Hilbert

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_14

space, which are the *tensor products* of two $\mathbb{C}^2$ spaces with basis vectors, $|0\rangle$ and $|1\rangle$. For convenience, I will call this $\{01\}$ basis. Therefore, we can also write

$$\begin{aligned} \left|\Psi^-\right\rangle_{\{01\}} &= \frac{1}{\sqrt{2}}(0\,|00\rangle + 1\,|01\rangle - 1\,|10\rangle + 0\,|11\rangle) \\ &= \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \end{aligned} \tag{14.1}$$

Here I write $\left|\Psi^-\right\rangle$ as $\left|\Psi^-\right\rangle_{\{01\}}$ because I want to make it clear that this vector is expressed in the basis formed by $|00\rangle/|01\rangle/|10\rangle/|11\rangle$ in the $\mathbb{C}^4$ space. We already proved in the Problems in the last chapter that $\left|\Psi^-\right\rangle$ is an entangled state and cannot be expressed as a single tensor product of two vectors from the $\mathbb{C}^2$ spaces. However, is it possible to have a Bell state on another basis (let me call it $\{\alpha\beta\}$), which is entangled, such that it is *not* entangled in the $\{01\}$ basis?

$\mathbb{C}^2$ is a 2D space with complex coefficients. Like what we did earlier, we can use a real 2D plane to illustrate how to *change the basis* as shown in Fig. 14.1. This figure is wrong but is intended to give you a relevant idea. In general, the new basis vectors $|\alpha\rangle$ and $|\beta\rangle$ can be expressed as the linear combinations (*superposition*) of the old basis vectors, $|0\rangle$ and $|1\rangle$:

$$\begin{aligned} |\alpha\rangle &= a\,|0\rangle + b\,|1\rangle \\ |\beta\rangle &= c\,|0\rangle + d\,|1\rangle \end{aligned} \tag{14.2}$$

where a, b, c, and d are complex numbers. As we have been doing, the basis vectors need to be *orthonormal*. Therefore, $\langle\alpha|\alpha\rangle = \langle\beta|\beta\rangle = 1$ and $\langle\alpha|\beta\rangle = \langle\beta|\alpha\rangle = 0$. $|\beta\rangle$ has two complex numbers. To determine $|\beta\rangle$, we need to fix 4 values (real and imaginary parts of the two complex numbers). Equations $\langle\beta|\alpha\rangle = 0$ and $\langle\beta|\beta\rangle = 1$ help us fix two of them and it still has two numbers undetermined (i.e. still has two degrees of freedom). One of them can be captured as the overall *phase factor*, which

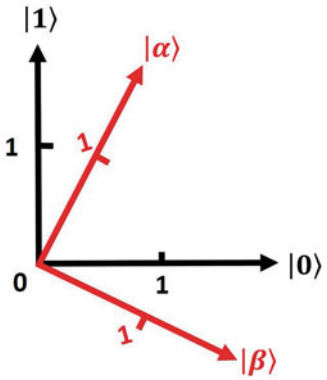


Fig. 14.1 Changing basis states in the $\mathbb{C}^2$ space using a fake real 2D space illustration

is immaterial (see the discussion after Eq. (6.12)) because when we calculate any physical quantities, both the *bra* and *ket* versions of the vector are involved and the global phase factors in the *bra* and *ket* cancel each other.

Therefore, if $|\alpha\rangle$ is given, there is only *one* value left to be determined in $|\beta\rangle$. We may express $|\beta\rangle$ as

$$|\beta\rangle = e^{i\theta}(b^*|0\rangle - a^*|1\rangle) = \begin{pmatrix} e^{i\theta}b^* \\ -e^{i\theta}a^* \end{pmatrix} \tag{14.3}$$

This is given to you. If you cannot see why right the way, this is normal. But we can check if it is correct. First of all, it only has one extra parameter, which is θ as required. Then let us check if it is normalized.

$$\begin{aligned} \langle\beta|\,|\beta\rangle &= \begin{pmatrix} e^{-i\theta}b & -e^{-i\theta}a \end{pmatrix} \begin{pmatrix} e^{i\theta}b^* \\ -e^{i\theta}a^* \end{pmatrix} \\ &= e^{-i\theta}be^{i\theta}b^* + (-e^{-i\theta}a)(-e^{i\theta}a^*) = bb^* + aa^* = 1 \end{aligned} \tag{14.4}$$

Note that we have used transpose and complex conjugation to construct the bra version of $|\beta\rangle$. Moreover, since $\langle\alpha|\alpha\rangle = 1$, we have used the fact $aa^* + bb^* = 1$.

Now let us check if $|\beta\rangle$ is orthogonal to $|\alpha\rangle$.

$$\begin{aligned} \langle\beta|\,|\alpha\rangle &= \begin{pmatrix} e^{-i\theta}b & -e^{-i\theta}a \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} \\ &= e^{-i\theta}ba + (-e^{-i\theta}a)(b) = 0 \end{aligned} \tag{14.5}$$

Therefore, Eq. (14.3) is a correct representation of $|\beta\rangle$ when $|\alpha\rangle$ is given and $|\beta\rangle$ and $|\alpha\rangle$ form a pair of orthonormal basis from which we will construct the new $|\Phi^-\rangle$.

Up to now, I have only shown you how to construct a new basis with a minimal number of parameters. This is nothing new to you but just to review some important concepts and skills. Now we have two different bases in a $\mathbb{C}^2$ space, namely, $\{01\}$ and $\{\alpha\beta\}$. They are related to each other by Eq. (14.2). We can then construct a $\mathbb{C}^4$ space using either two $\{01\}$ spaces or two $\{\alpha\beta\}$ spaces. The new space will have four basis vectors. The straightforward sets are $|00\rangle/|01\rangle/|10\rangle/|11\rangle$ if it is formed by $\{01\}$ basis or $|\alpha\alpha\rangle/|\alpha\beta\rangle/|\beta\alpha\rangle/|\beta\beta\rangle$ if it is formed by $\{\alpha\beta\}$ basis. *Note that they are the same space but just have a different basis or they are "viewed" at different angles.*

How do we represent them in column forms? If we are using $|00\rangle/|01\rangle/|10\rangle/|11\rangle$, they are represented as

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}_{\{01\}} ; \quad |01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}_{\{01\}} ; \quad |10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}_{\{01\}} ; \quad |11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}_{\{01\}} \tag{14.6}$$

How about in the $|\alpha\alpha\rangle/|\alpha\beta\rangle/|\beta\alpha\rangle/|\beta\beta\rangle$ basis? It is the same, except we want to label the basis carefully to avoid confusion.

$$|\alpha\alpha\rangle = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}_{\{\alpha\beta\}} \quad ; \quad |\alpha\beta\rangle = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}_{\{\alpha\beta\}} \quad ; \quad |\beta\alpha\rangle = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}_{\{\alpha\beta\}} \quad ; \quad |\beta\beta\rangle = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}_{\{\alpha\beta\}} \tag{14.7}$$

We see that they are the same because each element in the column form represents how much the corresponding basis vector it has.

Therefore, $\left|\Psi^-\right\rangle_{\{\alpha\beta\}}$ is just

$$\left|\Psi^-\right\rangle_{\{\alpha\beta\}} = \frac{1}{\sqrt{2}}(|\alpha\beta\rangle - |\beta\alpha\rangle) = \begin{pmatrix}0\\1\\-1\\0\end{pmatrix}_{\{\alpha\beta\}} \tag{14.8}$$

This is similar to Eq. (14.1). Now I want to know if this entangled $\left|\Psi^-\right\rangle_{\{\alpha\beta\}}$ state is still entangled when it is represented in the {01} basis. From Eq. (14.2) and Eq. (14.3),

$$\begin{aligned}
&\left|\Psi^-\right\rangle_{\{\alpha\beta\}} \\
&= \frac{1}{\sqrt{2}}(|\alpha\beta\rangle - |\beta\alpha\rangle) \\
&= \frac{1}{\sqrt{2}}(|\alpha\rangle \otimes |\beta\rangle - |\beta\rangle \otimes |\alpha\rangle) \\
&= \frac{1}{\sqrt{2}}(|\alpha\rangle\,|\beta\rangle - |\beta\rangle\,|\alpha\rangle) \\
&= \frac{1}{\sqrt{2}}((a\,|0\rangle + b\,|1\rangle)(e^{i\theta}(b^*\,|0\rangle - a^*\,|1\rangle))) \\
&\quad -(e^{i\theta}(b^*\,|0\rangle - a^*\,|1\rangle))(a\,|0\rangle + b\,|1\rangle)))
\end{aligned}$$

By using the distribution rule and making sure *not* to swap the orders of the *ket's* in the tensor products during this process, some terms are canceled and we have

$$\begin{aligned}
\left|\Psi^-\right\rangle_{\{\alpha\beta\}} &= \frac{1}{\sqrt{2}}(-(|a|^2 + |b|^2)e^{i\theta}\,|0\rangle\,|1\rangle + e^{i\theta}(|a|^2 + |b|^2)\,|1\rangle\,|0\rangle) \\
&= -e^{i\theta}(\frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)) \\
&= -e^{i\theta}\left|\Psi^-\right\rangle_{\{10\}}
\end{aligned} \tag{14.9}$$

Note that $-e^{i\theta}$ is just an overall phase factor and we can ignore it as discussed earlier. Therefore, the Bell state is still Bell state even after rotations or being represented on another basis. And, thus, it is still entangled. This answers our question.

14.3 Pure State, Mixed State, and Density Matrix

In this book, we will only deal with **pure states**. But I need to introduce **density matrix** for entanglement measurement. Therefore, I would like to talk about also **mixed states**. These are important concepts in quantum computing if you want to explore further after this book.

A pure state is just a state (i.e. vector) in a space, with which we have been dealing. For example, an electron spin of $|0\rangle$ is a pure state. But an electron spin of $\sqrt{0.7}\,|0\rangle + \sqrt{0.3}\,|1\rangle$ is *also* a pure state. A pure state is just a state in a space so it is not limited to the basis states. It can be any linear combination of the basis states. If we perform a measurement on the given pure state, there is a probability of 0.7 that we will get $|0\rangle$ and 0.3 that we will get $|1\rangle$, which are obtained by finding the square of the magnitude of the corresponding coefficient.

However, it is often that we do not have complete information about the state. For example, there might be 100 electrons and we only know that 70% of them are $|0\rangle$ and 30% of them are $|1\rangle$. So each of the electron is in a pure state (either $|0\rangle$ or $|1\rangle$), but we do not know which one is at $|0\rangle$ or $|1\rangle$. Then we say the electrons are in mixed states. If we perform a measurement, there is still a probability of 0.7 that we will get $|0\rangle$ and 0.3 that we will get $|1\rangle$ just because we know that 70% of them are $|0\rangle$ and 30% of them are $|1\rangle$. Or in other words, I know for *each electron*, there is a 70% chance to be in $|0\rangle$ and 30% chance to be in $|1\rangle$.

How to represent the mixed state mathematically? It turns out *density matrix* is a very suitable tool. Density matrix of an ensemble (either mixed or pure) is defined as

$$\boldsymbol{\rho} = \sum_{j=0}^{n-1} p_j \left|\Psi_j\right\rangle\left\langle\Psi_j\right| \tag{14.10}$$

where we assume there are n different pure states, $\left|\Psi_j\right\rangle$, each has the portion of p_j. Therefore, sum of p_j is 1 and it also represents the probability of finding an electron in state $\left|\Psi_j\right\rangle$.

Let us look at an example. For the system with pure state $|\Psi\rangle = \sqrt{0.7}\,|0\rangle + \sqrt{0.3}\,|1\rangle$, it only has one pure state ($|\Psi_0\rangle = |\Psi\rangle$) and the density matrix is

$$\begin{aligned}\boldsymbol{\rho} = |\Psi\rangle\langle\Psi| &= \begin{pmatrix}\sqrt{0.7}\\ \sqrt{0.3}\end{pmatrix}\begin{pmatrix}\sqrt{0.7} & \sqrt{0.3}\end{pmatrix}\\ &= \begin{pmatrix}0.7 & \sqrt{0.21}\\ \sqrt{0.21} & 0.3\end{pmatrix}\end{aligned} \tag{14.11}$$

For the system with the mixed state, we have

$$\begin{aligned}\boldsymbol{\rho} &= 0.7\,|0\rangle\,\langle 0| + 0.3\,|1\rangle\,\langle 1| \\ &= 0.7\begin{pmatrix}1\\0\end{pmatrix}\begin{pmatrix}1\ 0\end{pmatrix} + 0.3\begin{pmatrix}0\\1\end{pmatrix}\begin{pmatrix}0\ 1\end{pmatrix} \\ &= 0.7\begin{pmatrix}1\ 0\\0\ 0\end{pmatrix} + 0.3\begin{pmatrix}0\ 0\\0\ 1\end{pmatrix} = \begin{pmatrix}0.7\ \ 0\\0\ \ 0.3\end{pmatrix}\end{aligned} \tag{14.12}$$

where $|\Psi_0\rangle = |0\rangle$, $|\Psi_1\rangle = |1\rangle$, $p_0 = 0.7$, and $p_0 = 0.3$.

We see that although the given pure and mixed states have the same probabilistic measurement outcomes, their density matrices are different.

14.4 Measurement of Entanglement

To check if a pure state is an entangled state, we need to find its **entanglement entropy**. If it is 1, it is *maximally entangled*. If it is 0, it is a pure product state.

Entanglement entropy is defined as

$$S_A = -Tr_B(\boldsymbol{\rho_B} log_2(\boldsymbol{\rho_B})) \quad \text{where} \quad \boldsymbol{\rho_B} = Tr_A(\boldsymbol{\rho}) \tag{14.13}$$

This looks complicated. But we have learned that nothing can scare us if we try to understand each symbol one by one and carefully. Firstly, there is something called the **trace** of a matrix that we have not learned. The trace of a matrix is simply the sum of its diagonal elements. Therefore,

$$Tr(\boldsymbol{\rho}) = \sum_{i=0}^{n-1} \rho_{i,i} \tag{14.14}$$

For example, the trace of the density matrix in Eqs. (14.11) and (14.12) are both $\rho_{0,0}+\rho_{1,1} = 0.7+0.3 = 1$. But what is Tr_A or Tr_B? This is called the **partial trace** in which the sum is only performed over the index of A or B subspaces, respectively. Note that we are talking about entanglement. So the space must be at least a tensor product of two subspaces (e.g. $\mathbb{C}^4 = \mathbb{C}^2 \otimes \mathbb{C}^2$). The easiest way to understand is to do a numerical example.

Let us consider $\left|\Phi^-\right\rangle$. We know it is an entangled state. To what degree it is entangled? $\left|\Phi^-\right\rangle$ is a pure vector in the $\mathbb{C}^4$ space, which is a tensor product of two

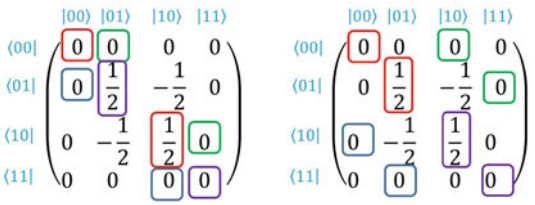


Fig. 14.2 Illustrations on how to do a partial tracing. Left: tracing only the first qubit. Right: tracing only the second qubit. Those with the same colors will be added during tracing

$\mathbb{C}^2$ spaces and let us call them space A and space B and thus $\mathbb{C}^4 = \mathbb{C}^2_A \otimes \mathbb{C}^2_B$. Firstly, let us evaluate the density matrix of $|\Phi^-\rangle$.

$$\boldsymbol{\rho} = |\Phi^-\rangle\langle\Phi^-| = \frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\-1\\0\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}0&1&-1&0\end{pmatrix}$$

$$= \frac{1}{2}\begin{pmatrix}0&0&0&0\\0&1&-1&0\\0&-1&1&0\\0&0&0&0\end{pmatrix} \tag{14.15}$$

Now let us compute $\boldsymbol{\rho_B} = Tr_A(\boldsymbol{\rho})$. If we were to find the trace, it is just $0 + 1 + 1 + 0 = 2$ by tracing over both the first and the second spaces (i.e. A and B). However, if we only want to trace over space A, which is the first space in the tensor product, we need to keep space B indices intact. Figure 14.2 shows the methodology. Like regular tracing, we only care about the elements with row index of space A equal to the column index of space A. Since we are only tracing A, there are 2 blocks and 8 elements that meet this criterion. The following shows the indices and the corresponding values

$$\frac{1}{2}\begin{pmatrix}\rho_{\{\mathbf{0}0\},\{\mathbf{0}0\}} & \rho_{\{\mathbf{0}0\},\{\mathbf{0}1\}} & \rho_{\{00\},\{10\}} & \rho_{\{00\},\{11\}}\\ \rho_{\{\mathbf{0}1\},\{\mathbf{0}0\}} & \rho_{\{\mathbf{0}1\},\{\mathbf{0}1\}} & \rho_{\{01\},\{10\}} & \rho_{\{01\},\{11\}}\\ \rho_{\{10\},\{00\}} & \rho_{\{10\},\{01\}} & \rho_{\{\mathbf{1}0\},\{\mathbf{1}0\}} & \rho_{\{\mathbf{1}0\},\{\mathbf{1}1\}}\\ \rho_{\{11\},\{00\}} & \rho_{\{11\},\{01\}} & \rho_{\{\mathbf{1}1\},\{\mathbf{1}0\}} & \rho_{\{\mathbf{1}1\},\{\mathbf{1}1\}}\end{pmatrix} = \frac{1}{2}\begin{pmatrix}0&0&&\\0&1&&\\&&1&0\\&&0&0\end{pmatrix} \tag{14.16}$$

The element indices are shown in binary form with the first index being that of space A and the second of space B. For example, $\rho_{\{10\},\{11\}}$ is the element in row 2 and column 3 (note that the row and column labels begin with 0 instead of 1). The A indices are highlighted when their row and column values are the same and only those corresponding element values are shown on the right. Since we need to keep the B index intact during the partial trace calculation, we will add the elements with the same B index and therefore, the partial trace over A is a 2×2 matrix. For

example, $\rho_{\{\mathbf{00}\},\{\mathbf{00}\}}$ is added to $\rho_{\{\mathbf{10}\},\{\mathbf{10}\}}$ because it has the same B index (i.e. 00). Therefore,

$$\boldsymbol{\rho_B} = \begin{pmatrix} 0 & 0 \\ 0 & \frac{1}{2} \end{pmatrix} + \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix} \tag{14.17}$$

where $\boldsymbol{\rho_B}$ is the **reduced density matrix** of space B because its element still depends on the B indices.

We can then calculate the entanglement entropy accordingly.

$$\begin{aligned} S_A &= -Tr_B(\boldsymbol{\rho_B} log_2(\boldsymbol{\rho_B})) \\ &= -Tr_B\left(\boldsymbol{\rho_B} log_2\left(\begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix}\right)\right) \\ &= -Tr_B\left(\boldsymbol{\rho_B}\begin{pmatrix} log_2(\frac{1}{2}) & 0 \\ 0 & log_2(\frac{1}{2}) \end{pmatrix}\right) = -Tr_B\left(\boldsymbol{\rho_B}\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}\right) \\ &= -Tr_B\left(\begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix}\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}\right) = -Tr_B\left(\begin{pmatrix} -\frac{1}{2} & 0 \\ 0 & -\frac{1}{2} \end{pmatrix}\right) = 1 \end{aligned} \tag{14.18}$$

Therefore the entanglement entropy of $\left|\Phi^-\right\rangle$ is 1, which means that *the entanglement is maximal. It is important to note that I take the logarithmic values of the diagonal elements of* $\boldsymbol{\rho_B}$ *directly because it is a diagonal matrix.* If it is not diagonal, I will need to diagonalize the matrix first, take the log, and then transform it back to do the rest of the calculations and operations. On the other hand, since the trace of a matrix does not change when the basis is changed, we can just work in the basis formed by the eigenvectors of $\boldsymbol{\rho_B}$ (i.e. work directly with the diagonalized matrix of $\boldsymbol{\rho_B}$) without converting back to the original basis to save time.

14.5 Summary

We have used a more advanced entanglement example to review most of the fundamental concepts and skills we have learned so far. I hope in this process you have discovered that you have been equipped with many useful mathematical tools and you are able to appreciate the critical concepts much better. We see that the Bell state is always fully entangled no matter how you rotate the coordinate (or on which basis you represent it). We also learned the meaning of pure and mixed states and how to calculate the density matrix of pure states and mixed states. Finally, we learned how to calculate the entanglement entropy, which is useful when we want to know how much entanglement a pure state has. In the next "step," we will start our first quantum algorithm.

Problems

14.1 Entanglement Entropy of a Product State

Find the density matrix of $\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$. Then find its entanglement entropy. (Hint: It should be zero because it is a product state.) Can you also factorize it into the product of two vectors in the $\mathbb{C}^2$ space? Note that its density matrix is not diagonal. Try to diagonalize it first (See Eq. (9.13)) before taking the logarithmic values.

14.2 Partial Tracing

Find the entanglement entropy of $\left|\Phi^{-}\right\rangle$ but trace the B space. You should get the same answer. The equation now becomes

$$S_B = -Tr_A(\boldsymbol{\rho_A} log_2(\boldsymbol{\rho_A})) \quad \text{where} \quad \boldsymbol{\rho_A} = Tr_B(\boldsymbol{\rho})$$

Chapter 15
Quantum Gate Introduction: NOT and CNOT Gates

15.1 Learning Outcomes

Understand how a quantum gate is related to the physics underneath; understand how a quantum gate is different from a classical gate; able to describe the matrix and properties of NOT and C-NOT gates; and understand the relationship between a quantum gate and its corresponding classical gate.

15.2 Basic Quantum Gate Properties

What is quantum computing? In short, it is just *a series of rotations of a high-dimensional vector in a high-dimensional space*. This high-dimensional space is the tensor product of many lower dimension spaces. For what we are interested in here, the basic low-dimensional space is the $\mathbb{C}^2$ Hilbert space where the qubit vectors reside. For example, in a 4-qubit quantum algorithm, the vector of interest is a 16-dimensional vector in the $\mathbb{C}^{16}$ space, which may be constructed by *grouping* four electron spins together. At the end of the rotations, we then perform a measurement to obtain the answer we are looking for (Fig. 15.1).

Of course, we need to perform meaningful rotations, so that we obtain the result we want. Besides that, the rotation needs to be *unitary* as we have discussed before. A unitary matrix, $\boldsymbol{U}$, obeys $\boldsymbol{U^{-1}} = \boldsymbol{U}^{\dagger}$ (Eq. (9.14)). The most important property of a unitary matrix is that it preserves the inner product of vectors and this is required in the laws of physics. Since it is unitary, it means its *inverse* exists. This is another requirement of the quantum gate. This is because quantum mechanics requires that every evolution (rotation) of the wavefunction is **reversible**. Again, this is natural if we think about the operations as the rotations in a hyperspace. For a rotation, it is always possible to rotate it backwards if we are given the result (the output) and the nature of the rotation (the type of the gate). Every step in quantum

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_15

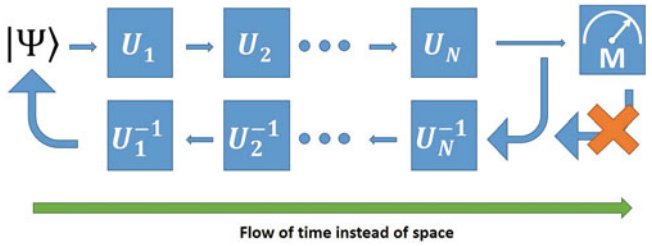


Fig. 15.1 Overview of quantum computing. $\boldsymbol{U_1}$ and $\boldsymbol{U_1^{-1}}$, etc. are quantum gates, which are unitary

computing is unitary and reversible. However, at the measurement, this is no longer true (Fig. 15.1).

A **quantum gate** is a gate that performs a unitary operation. In a classical computer, in general, the electrical signals flow through some physical gates (e.g. a NAND gate formed by 4 transistors) in the space. In quantum computing, usually, the qubit and its "signal" do not flow in the space. Instead, we apply operations such as microwave or laser pulses to *operate/manipulate/change* the qubit. Therefore, the flow of a quantum algorithm usually represents the flow of time instead of a physical layout in the space (Fig. 15.1). The qubits are usually stationary (except that in a trapped-ion quantum computer, the trapped ions are moved around but this still does not completely resemble the flow of signals in a classical computer).

In principle, we can stop here and continue to introduce individual quantum gates. But we can dive a little bit deeper to understand how a quantum gate is related to physics.

When the external pulses or fields are applied to a physical qubit, this is just physics and we know that it creates a system in which the qubit state will evolve accordingly. In physics, how the state changes depends on the total energy landscape of the system. The total energy has a more fancy term and it is called the **Hamiltonian**. You do not need to understand this, but it is good to know this term when you talk to the physicist. Basically, it is just the total energy. The law that describes how the state in the system evolves is called the **Schrödinger equation**, which is equivalent to Newton's Laws in classical mechanics. Schrödinger equation is written as

$$i\hbar \frac{\partial |\Psi\rangle}{\partial t} = \boldsymbol{H} |\Psi\rangle \tag{15.1}$$

where $\hbar$ is the reduced Planck's constant, $\boldsymbol{H}$ is the Hamiltonian, and $|\Psi\rangle$ is the state (vector) of the system. The left-hand side of the equation is the rate of change of the state. The right-hand size tells us that the rate of the change of the state can be obtained by multiplying the Hamiltonian on the state. This is as simple as Newton's $F = ma$! However, we need to be careful in understanding the meaning of each symbol. $|\Psi\rangle$ is a vector and can be expressed in a column form. $\boldsymbol{H}$ is an operator

and, thus, is a matrix. Therefore, it is not difficult to understand the Schrödinger equation, except that we need to be very careful when using it by asking the question I have been asking since the beginning, i.e., which are scalars, which are vectors, and which are matrices?

What is the solution to Eq. (15.1)? I will just give you and then we will check if it is correct. The solution is

$$|\Psi(t)\rangle = e^{\frac{-i\boldsymbol{H}t}{\hbar}} |\Psi(0)\rangle \tag{15.2}$$

Therefore, the wavefunction (or the state vector) at time t, $|\Psi(t)\rangle$, is just equal to $e^{\frac{-i\boldsymbol{H}t}{\hbar}}$ multiplied by the initial wavefunction at time 0, $|\Psi(0)\rangle$. Here we need to ask again, which one is a matrix and which one is a vector. Definitely, $|\Psi(t)\rangle$ and $|\Psi(0)\rangle$ are vectors and, thus, $e^{\frac{-i\boldsymbol{H}t}{\hbar}}$ must be a matrix (it can also be a scalar, but it is not in this case). How do we compute $e^{\frac{-i\boldsymbol{H}t}{\hbar}}$? In other words, how do we exponentiate a matrix? We have encountered something similar in Eq. (14.18) already when we take the logarithmic value of a matrix. Similar to that situation, if the matrix is diagonal, we just need to exponentiate the diagonal elements. If not, we need to diagonalize it first. But let us do not worry about it. I only need you to believe that we can do any mathematical operations on a matrix like we do on a number if it is diagonal. Let us assume we are working in the basis formed by the eigenvectors of $\boldsymbol{H}$ so that $\boldsymbol{H}$ is diagonal. By substituting Eq. (15.2) in Eq. (15.1), we have

$$\begin{aligned}
i\hbar \frac{\partial e^{\frac{-i\boldsymbol{H}t}{\hbar}} |\Psi(0)\rangle}{\partial t} &= \boldsymbol{H} e^{\frac{-i\boldsymbol{H}t}{\hbar}} |\Psi(0)\rangle \\
i\hbar \frac{-i\boldsymbol{H}}{\hbar} e^{\frac{-i\boldsymbol{H}t}{\hbar}} |\Psi(0)\rangle &= \boldsymbol{H} e^{\frac{-i\boldsymbol{H}t}{\hbar}} |\Psi(0)\rangle \\
\boldsymbol{H} e^{\frac{-i\boldsymbol{H}t}{\hbar}} |\Psi(0)\rangle &= \boldsymbol{H} e^{\frac{-i\boldsymbol{H}t}{\hbar}} |\Psi(0)\rangle
\end{aligned} \tag{15.3}$$

In the first line, we take the derivative just like how we do to a scalar in $\frac{\partial e^{\alpha t}}{\partial t} = \alpha e^{\alpha t}$ because the matrix $\boldsymbol{H}$ is assumed to be diagonal and then use the fact that $-ii = 1$. This shows that Eq. (15.2) satisfies Eq. (15.1).

Therefore, all vectors evolve in the way described by Eq. (15.2) and we can say $e^{\frac{-i\boldsymbol{H}t}{\hbar}}$ is a general expression of any *quantum gate* and let us call it $\boldsymbol{U}$. Let us check if it is unitary. Before that, we need to recognize that $\boldsymbol{H}$, which is the operator for energy, is Hermitian because energy is an observable. Therefore, $\boldsymbol{H} = \boldsymbol{H}^\dagger$. We have

$$\begin{aligned}
& \boldsymbol{U}\boldsymbol{U}^\dagger \\
= \; & e^{\frac{-i\boldsymbol{H}t}{\hbar}} (e^{\frac{-i\boldsymbol{H}t}{\hbar}})^\dagger \\
= \; & e^{\frac{-i\boldsymbol{H}t}{\hbar}} e^{\frac{i\boldsymbol{H}^\dagger t}{\hbar}} \\
= \; & e^{\frac{-i\boldsymbol{H}t}{\hbar}} e^{\frac{i\boldsymbol{H}t}{\hbar}} \\
= \; & \boldsymbol{I}
\end{aligned} \tag{15.4}$$

This section is an attempt to connect physics to quantum computing. You might not fully appreciate every step. This will not affect your understanding of quantum computing algorithm. But I hope you have a stronger belief that a quantum gate is just a physical operation and it is unitary in nature.

15.3 NOT (X) Gate

15.3.1 Definition

The first quantum gate we will study is the NOT gate. It is called X gate for a reason to be clear soon. Again, since every quantum gate is a unitary gate, I will label its matrix as $\boldsymbol{U_{NOT}}$. It is called the NOT gate because it is defined in this way:

$$\begin{aligned} \boldsymbol{U_{NOT}}\,|0\rangle &= |1\rangle \\ \boldsymbol{U_{NOT}}\,|1\rangle &= |0\rangle \end{aligned} \tag{15.5}$$

15.3.2 Matrix

You see that it is exactly the same as the classical NOT gate if we just treat the basis states as the classical values. This is consistent with what we discussed in Chap. 13 that a quantum register is exactly the same as a classical register if it is only allowed to store the basis states. To achieve the function in Eq. (15.5), we found that

$$\boldsymbol{U_{NOT}} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{15.6}$$

You may treat that as given. Or since a qubit state is in the two-dimensional $\mathbb{C}^2$ space, the matrix (operator) must be 2×2 and if you go through all the possible combinations, you find that this is the only matrix that will satisfy Eq. (15.5). Let us check if this is the case.

How do we represent $|0\rangle$ in a column form? It is $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$. And let me remind you that this makes sense because $|0\rangle$ is the first basis state and the column elements

tell us that it has one component of the first basis state and zero component of the second basis state. Similarly, we represent $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. Therefore,

$$\begin{aligned} \boldsymbol{U_{NOT}} |0\rangle &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} |0\rangle \\ &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle \end{aligned} \tag{15.7}$$

and

$$\begin{aligned} \boldsymbol{U_{NOT}} |1\rangle &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} |1\rangle \\ &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle \end{aligned} \tag{15.8}$$

15.3.3 Circuit and Properties

Just like the power of a quantum register is that it can store the superpositions of the basis states, the power of the quantum gates is that they can process the superpositions of the basis states. Now let us see what happens if $\boldsymbol{U_{NOT}}$ is applied to a general state $|\Psi\rangle = \alpha |0\rangle + \beta |1\rangle$.

$$\begin{aligned} \boldsymbol{U_{NOT}} |\Psi\rangle &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} |\Psi\rangle \\ &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix} = \beta |0\rangle + \alpha |1\rangle \end{aligned} \tag{15.9}$$

Therefore, we can also understand $\boldsymbol{U_{NOT}}$ as a gate that will swap the coefficients of the basis states (i.e. α becomes β and vice versa).

Now, you probably have already figured out why the NOT gate is also called an X gate? This is because the matrix in Eq. (15.6) is just $\boldsymbol{\sigma_x}$. Figure 15.2 shows the NOT operation and its circuits.

Besides the one-qubit NOT gate, we can also construct a two-qubit or n-qubit NOT gate. We just need to perform a tensor product of two one-qubit NOT gate

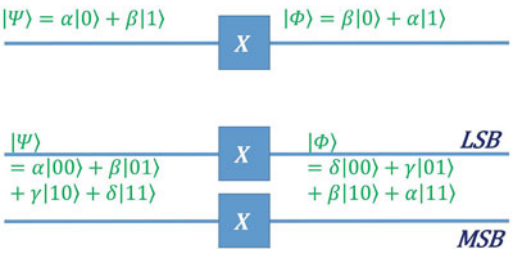


Fig. 15.2 NOT gate and its circuits. Top: 1-qubit NOT gate. Bottom: 2-qubit NOT gate. Note the bottom qubit is the MSB

to obtain a two-qubit NOT gate because a two-qubit vector resides in a $\mathbb{C}^4$ space, which is a tensor product of two $\mathbb{C}^2$ spaces. Therefore,

$$\begin{aligned} \boldsymbol{U}_{\boldsymbol{NOT}_2} = \boldsymbol{U}_{\boldsymbol{NOT}} \otimes \boldsymbol{U}_{\boldsymbol{NOT}} &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} & 1 \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\ 1 \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} & 0 \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \end{aligned} \tag{15.10}$$

Let us now apply the 2-qubit NOT gate to a general vector $|\Psi\rangle = \alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + \delta\,|11\rangle$.

$$\boldsymbol{U}_{\boldsymbol{NOT}_2}\,|\Psi\rangle = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} = \begin{pmatrix} \delta \\ \gamma \\ \beta \\ \alpha \end{pmatrix} \tag{15.11}$$

We see that it just reverses the order of the coefficients. This is the same in an n-qubit NOT gate. If you want to invent a quantum algorithm eventually, it is very important to have an insight on how each quantum gate *effectively* changes the input vector instead of just knowing how to perform matrix multiplications.

Figure 15.2 also shows the circuit of a 2-qubit NOT gate. If the qubit is not entangled, e.g. $|\Psi\rangle = (a_0\,|0\rangle + b_0\,|1\rangle) \otimes (a_1\,|0\rangle + b_1\,|1\rangle)$, I can assign the first qubit to be $a_0\,|0\rangle + b_0\,|1\rangle$ and the second qubit to be $a_1\,|0\rangle + b_1\,|1\rangle$. However, if it is at least partially entangled, I need to just write the 2-qubit state as $|\Psi\rangle = \alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + \delta\,|11\rangle$.

Also, please note that we will follow IBM-Q/qiskit's convention to assign the lower bit in the circuit diagram to be the *most significant bit* (MSB). The MSB means that that qubit is written on the left in a tensor product form. For example, the qubit $a_0\,|0\rangle + b_0\,|1\rangle$ is the MSB in $|\Psi\rangle = (a_0\,|0\rangle + b_0\,|1\rangle) \otimes (a_1\,|0\rangle + b_1\,|1\rangle)$, while $a_1\,|0\rangle + b_1\,|1\rangle$ is the least significant bit (LSB).

15.4 XOR (CNOT) Gate

15.4.1 Definition

In classical logic, the XOR gate is a 2-input gate. Similarly, an XOR quantum gate is 2-qubit. It is more commonly known as the CNOT gate, which stands for **Controlled-NOT** gate for an obvious reason to be explained soon. An XOR quantum gate is defined as

$$\boldsymbol{U_{XOR}}\,|ab\rangle = |aa \oplus b\rangle \tag{15.12}$$

I deliberately wrote it in a very confusing notation with which I expect you to be very familiar eventually. What is $|ab\rangle$? It is a vector in the $\mathbb{C}^4$ space and a and b are either 0 or 1. $|ab\rangle$ can be any of the basis states, namely, $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. This equation tells us that after the XOR gate is applied to one of the basis states, it will become another basis state $|aa \oplus b\rangle$. $|aa \oplus b\rangle$ is just $|a, a \oplus b\rangle$ or $|a\rangle \otimes |a \oplus b\rangle$. After the operation, the first number is still a, but the second number becomes $a \oplus b$, where $\oplus$ is the classical *exclusive or* (XOR) logical operation. As a reminder, $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, and $1 \oplus 1 = 0$. Let us see how it changes the individual basis state.

$$\begin{aligned}
\boldsymbol{U_{XOR}}\,|00\rangle &= |0, 0 \oplus 0\rangle = |0, 0\rangle = |00\rangle \\
\boldsymbol{U_{XOR}}\,|01\rangle &= |0, 0 \oplus 1\rangle = |0, 1\rangle = |01\rangle \\
\boldsymbol{U_{XOR}}\,|10\rangle &= |1, 1 \oplus 0\rangle = |1, 1\rangle = |11\rangle \\
\boldsymbol{U_{XOR}}\,|11\rangle &= |1, 1 \oplus 1\rangle = |1, 1\rangle = |10\rangle
\end{aligned} \tag{15.13}$$

We see that $\boldsymbol{U_{XOR}}$ swaps the third and the fourth basis and keeps the rest unchanged.

15.4.2 Matrix

The matrix of $\boldsymbol{U_{XOR}}$ is

$$\boldsymbol{U_{XOR}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{15.14}$$

We can treat this as given. Or since this is a 2-qubit gate for operating on a vector in the four-dimensional $\mathbb{C}^4$ space, the matrix (operator) must be 4×4. Moreover,

it only swaps the third and the fourth basis states. Therefore, this is an appropriate construction. Let us try a numerical example.

$$\boldsymbol{U_{XOR}}\,|10\rangle = \begin{pmatrix} 1&0&0&0\\ 0&1&0&0\\ 0&0&0&1\\ 0&0&1&0 \end{pmatrix} \begin{pmatrix} 0\\0\\1\\0 \end{pmatrix} = \begin{pmatrix} 0\\0\\0\\1 \end{pmatrix} = |11\rangle \tag{15.15}$$

This is because $|10\rangle$ is the third basis state and thus it is $\begin{pmatrix} 0\\0\\1\\0 \end{pmatrix}$ and $|11\rangle$ is $\begin{pmatrix} 0\\0\\0\\1 \end{pmatrix}$.

15.4.3 Circuit and Properties

You see that the output depends on $a \oplus b$, where a is the value of the first qubit and b is the value of the second qubit in the basis state. Based on the operations discussed, this is also equivalent to saying "if a is zero, keep b unchanged and if a is one, negate b (i.e. apply NOT to b)." Therefore, it is also called Controlled-NOT. Whether we will apply NOT to the second qubit depends on a (i.e. controlled by a). Therefore, the first qubit is called the **control qubit** and the second one is called the **target qubit**. For example, for $|10\rangle$, a is 1. Therefore it will apply *NOT* to b, which is 0. b will be negated to 1 and the state becomes $|11\rangle$. But for $|00\rangle$, a is 0. Therefore, b is unchanged and is still 0 and thus the basis state is still $|00\rangle$.

In general, for a 2-qubit vector, $|\Psi\rangle = \alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + \delta\,|11\rangle$, we have

$$\begin{aligned} \boldsymbol{U_{XOR}}\,|\Psi\rangle &= \begin{pmatrix} 1&0&0&0\\ 0&1&0&0\\ 0&0&0&1\\ 0&0&1&0 \end{pmatrix} \begin{pmatrix} \alpha\\ \beta\\ \gamma\\ \delta \end{pmatrix} = \begin{pmatrix} \alpha\\ \beta\\ \delta\\ \gamma \end{pmatrix} \\ &= \alpha\,|00\rangle + \beta\,|01\rangle + \delta\,|10\rangle + \gamma\,|11\rangle \end{aligned} \tag{15.16}$$

As expected, the coefficients of the last two basis states are swapped. Figure 15.3 shows the circuit of a CNOT operation. Note that again the bottom qubit is the MSB and that qubit acts as the *control qubit*.

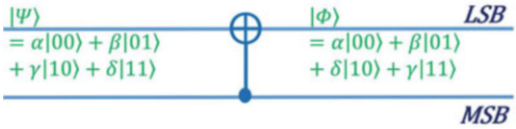


Fig. 15.3 CNOT gate and its circuits

15.5 Summary

We learned that a quantum gate is nothing but just the application of energy to a quantum state to rotate it in the hyperspace. Therefore, a quantum state evolves by following the physics law, and the quantum gate must be unitary and reversible. We introduced the NOT gate and the CNOT gate, which have their classical counterparts. Firstly, we understand them by checking how they change a basis state, based on how they are named. And then apply them to a more general superposition state. We learned their corresponding matrices and circuits. When we draw the circuit, we need to be extremely careful to annotate the location of the Most Significant Bit and the Least Significant Bit. In this book, we follow the convention in IBM-Q, with the MSB drawn at the bottom of the circuit diagram.

Problems

15.1 Unitarity of the NOT Gate
Use hand calculation and co-lab to show that a 2-qubit NOT gate is unitary.

15.2 Unitarity of the CNOT Gate
Use hand calculation and co-lab to show that the CNOT gate is unitary.

15.3 Implementation of a Quantum Circuit
Use IBM-Q, by choosing "IBM Quantum Composer" to implement a 2-qubit circuit in which a 1-qubit NOT gate is applied to the MSB and then a CNOT gate is applied to the two qubits with the MSB as the control qubit. Refer to Problem 2.3 for how to construct a circuit. Firstly, based on what you have learned, what result do you expect? Add measurement to check if you get the desired result. Hint: since the input begins with $|00\rangle$, the output is 100% $|11\rangle$. And note that in IBM-Q, the bottom qubit is the MSB.

Chapter 16
SWAP, Phase Shift, and CCNOT (Toffoli) Gates

16.1 Learning Outcomes

Able to describe the matrix and properties of SWAP, Phase Shift, and CCNOT gates.

16.2 SWAP Gate

16.2.1 Definition

A SWAP gate, as its name implies, swaps the numbers in the basis states of a 2-qubit register. If we consider only the basis states, it is equivalent to swapping the states of two electrons. The definition is

$$\boldsymbol{U_{SWAP}}\,|ab\rangle = |ba\rangle \tag{16.1}$$

where a and b are just the numbers of the first and the second qubits in the basis states, respectively. They can be either 0 or 1. The easiest way to gain a deeper understanding is just to plug in all the basis states and we get

$$\begin{aligned}
\boldsymbol{U_{SWAP}}\,|00\rangle &= |00\rangle \\
\boldsymbol{U_{SWAP}}\,|01\rangle &= |10\rangle \\
\boldsymbol{U_{SWAP}}\,|10\rangle &= |01\rangle \\
\boldsymbol{U_{SWAP}}\,|11\rangle &= |11\rangle
\end{aligned} \tag{16.2}$$

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_16

Therefore, only $|01\rangle$ and $|10\rangle$ are changed (to each other) under the SWAP operation. For $|00\rangle$ and $|11\rangle$, they are unchanged because swapping "0" and "0" or "1" and "1" is equivalent to doing nothing.

16.2.2 Matrix

The matrix of $\boldsymbol{U_{SWAP}}$ is

$$\boldsymbol{U_{SWAP}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{16.3}$$

We can understand it in this way. Since this is a 2-qubit gate for operating on a vector in the four-dimensional $\mathbb{C}^4$ space, the matrix (operator) must be 4×4. Moreover, it only swaps the second and the third basis states. Therefore, only the second and the third rows are different from the identity matrix. Let us try a numerical example.

$$\boldsymbol{U_{SWAP}}\,|10\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = |01\rangle \tag{16.4}$$

This is because $|10\rangle$ is the third basis state and thus it is $\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$ and $|01\rangle$ is $\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$.

16.2.3 Circuit and Properties

In general, for a 2-qubit vector, $|\Psi\rangle = \alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + \delta\,|11\rangle$, we have

$$\begin{aligned} \boldsymbol{U_{XOR}}\,|\Psi\rangle &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} = \begin{pmatrix} \alpha \\ \gamma \\ \beta \\ \delta \end{pmatrix} \\ &= \alpha\,|00\rangle + \gamma\,|01\rangle + \beta\,|10\rangle + \delta\,|11\rangle \end{aligned} \tag{16.5}$$

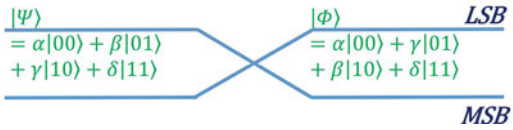


Fig. 16.1 SWAP gate and its circuit

As expected, the coefficients of the second and the third basis states are swapped. Figure 16.1 shows the circuit of a SWAP operation. Note that again the bottom qubit is the MSB. As we have been emphasizing, a quantum circuit represents a flow of time not a flow of signal. Therefore, unlike the classical swap operation, the implementation of a SWAP gate is complicated and is not as simple as how the circuit shows. To perform a SWAP operation, we need to apply some correct external pulses so that the states of two electron spins are swapped.

16.3 Phase Shift Gate

16.3.1 Definition

The phase shift gate, which is also called the **P-gate** or the phase gate, is a 1-qubit gate that shifts the relative phase between the two basis vectors. It is defined as the following:

$$\begin{aligned} \boldsymbol{U_{PS,\Phi}}\,|0\rangle &= |0\rangle \\ \boldsymbol{U_{PS,\Phi}}\,|1\rangle &= e^{i\Phi}\,|1\rangle \end{aligned} \tag{16.6}$$

where Φ is the **phase** and $e^{i\Phi}$ is the **phase factor**. When the phase shift gate is applied to the basis vector $|0\rangle$, it performs nothing. But when it is applied to the basis vector $|1\rangle$, it adds an additional phase to it. Adding an additional phase means that the vector is multiplied by $e^{i\Phi}$. Why do we only apply the additional phase to the second basis? What is the meaning of shifting the relative phase between the two basis vectors? It is easier to understand if we look at its matrix form and observe how it evolves a general vector.

16.3.2 Matrix

The matrix of $\boldsymbol{U_{PS,\Phi}}$ is

$$\boldsymbol{U_{PS,\Phi}} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\Phi} \end{pmatrix} \tag{16.7}$$

You can check it easily that Eq. (16.7) satisfies Eq. (16.6). For example,

$$\boldsymbol{U_{PS,\Phi}}\,|1\rangle = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\Phi} \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ e^{i\Phi} \end{pmatrix} = e^{i\Phi} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = e^{i\Phi}\,|1\rangle \tag{16.8}$$

16.3.3 Circuit and Properties

In general, for a 1-qubit vector, $|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, we have

$$\begin{aligned} \boldsymbol{U_{PS,\Phi}}\,|\Psi\rangle &= \begin{pmatrix} 1 & 0 \\ 0 & e^{i\Phi} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha \\ e^{i\Phi}\beta \end{pmatrix} \\ &= \alpha\,|0\rangle + e^{i\Phi}\beta\,|1\rangle \end{aligned} \tag{16.9}$$

In the original vector, α and β, which are complex numbers in general, can be represented as the product of a phase factor and their magnitude. For example, $\alpha = e^{i\theta_1}|\alpha|$ and $\beta = e^{i\theta_2}|\beta|$, where θ_1 and θ_2 are the phases of α and β, respectively. Therefore, their phase difference is $\theta_1 - \theta_2$. Due to the phase shift gate, the phase of α is unchanged but the phase of β is changed to $\Phi + \theta_2$. Therefore, their relative phase is changed by Φ. This also explains why the phase shift is only applied to the second basis state to be meaningful. If it is applied to both, there will be no change in the phase difference.

Φ is a parameter of the phase shift gate. When $\Phi = \pi$, the gate becomes $\begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ because $e^{i\pi} = \cos\pi + i\sin\pi = -1$. And this is just the $\boldsymbol{\sigma_z}$ matrix, one of the important Pauli matrices. And it is also called the **Z-gate**.

The phase shift gate shifts the relative phase of $|0\rangle$ and $|1\rangle$ by Φ. And thus the Z-gate shifts the relative phase by π. This makes the Z-gate very special and it turns out that in the $|+\rangle\,/\,|-\rangle$ basis, the Z-gate acts as a NOT gate (X-gate). This might sound confusing. Let us recall what the meaning of a NOT gate is. When the NOT gate is applied to a basis state, the basis state is converted to the other basis state. Since it is a 1-qubit gate, there are only 2 basis vectors. This means that the NOT gate converts the basis state to each other. For example, in the space formed by the basis vectors $|0\rangle$ and $|1\rangle$, the NOT gate converts $|0\rangle$ to $|1\rangle$ and vice versa. Therefore, a NOT gate in the $|+\rangle\,/\,|-\rangle$ space is expected to convert $|+\rangle$ to $|-\rangle$ and vice versa. Let us check if the Z-gate behaves as a NOT gate in the $|+\rangle\,/\,|-\rangle$ basis.

$$\begin{aligned} \boldsymbol{U_{PS,\pi}}\,|+\rangle &= \boldsymbol{U_Z}\,|+\rangle = \boldsymbol{U_Z}\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle \end{aligned} \tag{16.10}$$

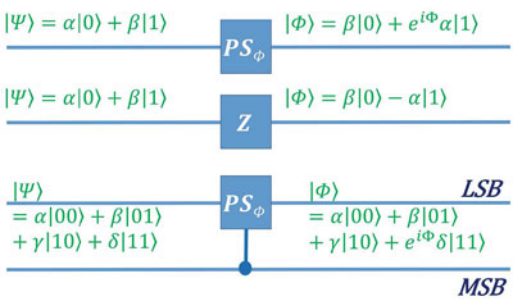


Fig. 16.2 Phase shift gate circuits. Top: 1-bit phase shift gate. Middle: Z-gate. Bottom: controlled-PS gate

We see that the Z-gate does convert $|+\rangle$ to $|-\rangle$. Here I do not use the matrix representation. I use my understanding of the Z-gate that it will add a phase shift of π to the coefficient of the second basis state (i.e. $|1\rangle$). Therefore, I just multiply -1 to $|1\rangle$. I hope you can start training yourselves on this skill because such an understanding and insight are important for you to be able to design new quantum algorithms. Matrix multiplications are just like circuit simulations that we should let the computer do it. Understanding the nature of the gate is just like how we understand an analog circuit through inspection.

Figure 16.2 shows the circuit of a phase shift gate and Z-gate.

16.4 Controlled Phase Shift Gate

16.4.1 Definition

Since Z-gate is the NOT gate in the $|+\rangle\,/\,|-\rangle$ basis, it is then natural that there may be a "controlled-Z-gate," which is a CNOT gate in the $|+\rangle\,/\,|-\rangle$ basis. And in general, there must be a controlled phase shift gate. Indeed, we can define a 2-qubit controlled phase shift gate as the following:

$$\boldsymbol{U_{CPS,\Phi}}\,|ab\rangle = e^{i(a\cdot b)\Phi}\,|ab\rangle \tag{16.11}$$

where a and b are just the numbers in the first and the second qubit in the basis states, respectively. They can be either 0 or 1. Here, "$\cdot$" means the classical logic AND operation. Therefore, $a \cdot b$ equals 1 only when both a and b are 1. Let us see how it changes the basis states.

$$\begin{aligned}
\boldsymbol{U_{CPS,\Phi}}\,|00\rangle &= e^{i(0\cdot 0)\Phi}\,|00\rangle = |00\rangle \\
\boldsymbol{U_{CPS,\Phi}}\,|01\rangle &= e^{i(0\cdot 1)\Phi}\,|01\rangle = |01\rangle \\
\boldsymbol{U_{CPS,\Phi}}\,|10\rangle &= e^{i(1\cdot 0)\Phi}\,|10\rangle = |10\rangle \\
\boldsymbol{U_{CPS,\Phi}}\,|11\rangle &= e^{i(1\cdot 1)\Phi}\,|11\rangle = e^{i\Phi}\,|11\rangle
\end{aligned} \tag{16.12}$$

Therefore, only $|11\rangle$ is changed with an extra phase. We can also understand in this way. When the first qubit is 0 (i.e. in the $|00\rangle$ and $|01\rangle$ cases), nothing is applied to the second qubit. But when the first qubit is 1, a phase shift gate is applied. Therefore, a phase shift gate is applied to the second qubit when the first qubit is "1." So, the first qubit is the **control qubit** and the second qubit is the **target qubit** in this controlled phase shift gate. Why only $|11\rangle$ is changed? Again, in a phase shift gate, the phase shift is only applied to the second basis state. Therefore, $|10\rangle$ is unaffected and only $|11\rangle$ gets the extra phase shift.

16.4.2 Matrix

The matrix of $\boldsymbol{U_{CPS,\Phi}}$ is

$$\boldsymbol{U_{CPS,\Phi}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\Phi} \end{pmatrix} \tag{16.13}$$

Again, since this is a 2-qubit gate for operating on a vector in the four-dimensional $\mathbb{C}^4$ space, the matrix (operator) must be 4×4. And as discussed, only $|11\rangle$ gains an extra phase after this gate. Therefore, it is almost like an identity matrix except the last row. Let us try a numerical example.

$$\boldsymbol{U_{CPS,\Phi}}\,|11\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\Phi} \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ e^{i\Phi} \end{pmatrix} = e^{i\Phi} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = e^{i\Phi}\,|11\rangle \tag{16.14}$$

16.4.3 Circuit and Properties

In general, for a 2-qubit vector, $|\Psi\rangle = \alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + \delta\,|11\rangle$, we have

$$\begin{aligned} \boldsymbol{U_{CPS,\Phi}}\,|\Psi\rangle &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\Phi} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ e^{i\Phi}\delta \end{pmatrix} \\ &= \alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + e^{i\Phi}\delta\,|11\rangle \end{aligned} \tag{16.15}$$

Therefore, a controlled phase shift gate only adds a phase to the $|11\rangle$ component. Figure 16.2 shows the circuit of a controlled phase shift gate operation. Note that again the bottom qubit is the MSB.

16.5 Toffoli (CCNOT) Gate

16.5.1 Definition

A Toffoli gate is a 3-qubit gate. It is also known as the CCNOT gate, which stands for **Controlled-Controlled-NOT** gate. As discussed earlier, when a CNOT gate is applied to the basis states of $\mathbb{C}^4$ (a 2-qubit space), it will apply a NOT operator to the second qubit if the first qubit (the control qubit) is 1. Therefore, we can expect that a CCNOT will apply a NOT operator to the third qubit if both the first and second qubits of a $\mathbb{C}^8$ (a 3-qubit space) basis state are 1. We will check this later.

Let us begin with the definition:

$$\boldsymbol{T}\,|abc\rangle = \boldsymbol{T}\,|a\rangle \otimes |b\rangle \otimes |c\rangle = \boldsymbol{T}\,|a, b, c\rangle = |a, b, (a \cdot b) \oplus c\rangle \qquad (16.16)$$

$\boldsymbol{T}$ is the unitary operator of the Toffoli gate. $|abc\rangle$ is the basis state of a $\mathbb{C}^8$ space, which is formed by the tensor products of three $\mathbb{C}^2$ spaces due to three qubits (See the more complex $\mathbb{C}^{16}$ example in Table 13.1). a and b are either 0 or 1. $|abc\rangle$ can be any of the 8 basis states running from $|000\rangle$, $|001\rangle$, to $|111\rangle$. I deliberately put commons to separate the a, b, and c in the *ket* for clarity, which is completely legitimate as I am allowed to use any descriptive language to name the basis states. After applying $\boldsymbol{T}$ to $|a, b, c\rangle$, the basis state becomes $|a, b, (a \cdot b) \oplus c\rangle$. This means the values of the first 2 qubits remain unchanged. But the last qubit value becomes $(a \cdot b) \oplus c$. The "$\cdot$" operator is the *AND* operator in the classical logic. Therefore, it is only one when both a and b are one. $\oplus$ is the classical XOR operation. Again, $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, and $1 \oplus 1 = 0$. As discussed earlier in the XOR (CNOT) gate, this means that a NOT operator is applied to the value of the last qubit if $a \cdot b$ is one. Therefore, this is a controlled-controlled-NOT gate with the first and the second qubits being the *control* qubits. Let us check this numerically.

$$\begin{aligned}
\boldsymbol{T}\,|0, 0, 0\rangle &= |0, 0, (0 \cdot 0) \oplus 0\rangle = |0, 0, 0\rangle \\
\boldsymbol{T}\,|0, 0, 1\rangle &= |0, 0, (0 \cdot 0) \oplus 1\rangle = |0, 0, 1\rangle \\
\boldsymbol{T}\,|0, 1, 0\rangle &= |0, 1, (0 \cdot 1) \oplus 0\rangle = |0, 1, 0\rangle \\
\boldsymbol{T}\,|0, 1, 1\rangle &= |0, 1, (0 \cdot 1) \oplus 1\rangle = |0, 1, 1\rangle \\
\boldsymbol{T}\,|1, 0, 0\rangle &= |1, 0, (1 \cdot 0) \oplus 0\rangle = |1, 0, 0\rangle \\
\boldsymbol{T}\,|1, 0, 1\rangle &= |1, 0, (1 \cdot 0) \oplus 1\rangle = |1, 0, 1\rangle
\end{aligned}$$

$$\begin{aligned} \boldsymbol{T}\,|1,1,0\rangle &= |1,1,(1\cdot 1)\oplus 0\rangle = |1,1,1\rangle \\ \boldsymbol{T}\,|1,1,1\rangle &= |1,1,(1\cdot 1)\oplus 1\rangle = |1,1,0\rangle \end{aligned} \tag{16.17}$$

We see that $\boldsymbol{T}$ keeps the basis vectors intact except swaps the last two basis vectors because only the last two basis vectors have both the first and the second qubit values being one to enable the NOT operation on the last qubit value.

16.5.2 Matrix

The matrix of T is

$$\boldsymbol{T} = \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1&0 \end{pmatrix} \tag{16.18}$$

Since the Toffoli gate is a 3-qubit gate in the eight-dimensional $\mathbb{C}^8$ space, the matrix (operator) must be 8×8. Moreover, it only swaps the last two basis states. Therefore, it is different from an identity matrix only in the last two rows, which has the NOT gate matrix embedded. Let us try a numerical example.

$$\boldsymbol{T}\,|111\rangle = \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1&0 \end{pmatrix} \begin{pmatrix} 0\\0\\0\\0\\0\\0\\0\\1 \end{pmatrix} = \begin{pmatrix} 0\\0\\0\\0\\0\\0\\1\\0 \end{pmatrix} = |110\rangle \tag{16.19}$$

16.5.3 Circuit and Properties

In general, for a 3-qubit vector, $|\Psi\rangle = \alpha\,|000\rangle + \beta\,|001\rangle + \gamma\,|010\rangle + \delta\,|011\rangle + \epsilon\,|100\rangle + \zeta\,|101\rangle + \eta\,|110\rangle + \theta\,|111\rangle$, we have

$$
\begin{aligned}
\boldsymbol{T}\,|\Psi\rangle &= \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1&0 \end{pmatrix} \begin{pmatrix} \alpha\\ \beta\\ \gamma\\ \delta\\ \epsilon\\ \zeta\\ \eta\\ \theta \end{pmatrix} = \begin{pmatrix} \alpha\\ \beta\\ \gamma\\ \delta\\ \epsilon\\ \zeta\\ \theta\\ \eta \end{pmatrix} \\
&= \alpha\,|000\rangle + \beta\,|001\rangle + \gamma\,|010\rangle + \delta\,|011\rangle + \epsilon\,|100\rangle \\
&\quad + \zeta\,|101\rangle + \theta\,|110\rangle + \eta\,|111\rangle \\
&= \alpha\,|0\rangle + \beta\,|1\rangle + \gamma\,|2\rangle + \delta\,|3\rangle + \epsilon\,|4\rangle + \zeta\,|5\rangle + \theta\,|6\rangle + \eta\,|7\rangle \qquad (16.20)
\end{aligned}
$$

As expected, the coefficients of the last two basis states are swapped. Figure 16.3 shows the circuit of a CCNOT operation. Note that again the bottom qubit is the MSB and the two most significant qubits act as the *control qubits*. Here I deliberately rewrite the *kets* in their decimal form to remind you that we should choose a basis state naming convention that is convenient.

There is a very special property of the Toffoli gate. It is the *universal gate of the classical logic*. What does it mean? Remember that, so far, we define all gates in terms of how they change the basis states in the space of interest. For example, a Toffoli gate keeps all 8-dimensional basis states unchanged except swaps $|110\rangle$ and $|111\rangle$. *Quantum computing gates, which have classical analogs (i.e. for the gates we have discussed so far), and quantum registers are exactly the same as the classical gates and classical registers, respectively, if they are only allowed to process the basis states*. Quantum computer is only special because they can also process the superposition of the basis states (and also because there are quantum gates that have no classical analogs, which we will discuss soon). Therefore, if we limit the quantum computers to only processing the basis states, there is no difference between the

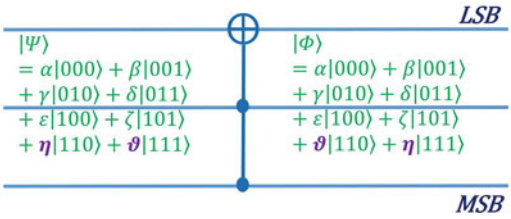


Fig. 16.3 Toffoli (CCNOT) gate circuit. The swapped coefficients are highlighted

quantum computer and the classical computer (**but this is only true for the gates we have discussed so far**).

We can make a Toffoli gate to act as a classical AND gate by setting $c = 0$. In this case,

$$\boldsymbol{T}\,|abc\rangle = \boldsymbol{T}\,|ab0\rangle = |a, b, (a \cdot b) \oplus 0\rangle = |a, b, a \cdot b\rangle \tag{16.21}$$

This is because any number XOR with 0 is unchanged. Therefore, the third qubit value equals *a AND b* and thus it can act as a classical AND gate, with the first two qubits as the inputs and the third qubit as the output.

Similarly, if we set $a = 1$ and $b = 1$, we have

$$\boldsymbol{T}\,|abc\rangle = \boldsymbol{T}\,|11c\rangle = |1, 1, (1 \cdot 1) \oplus c\rangle = |1, 1, 1 \oplus c\rangle = |1, 1, \bar{c}\rangle \tag{16.22}$$

This is because $1 \oplus c = \bar{c}$, where $\bar{c}$ is the negation of c. If we treat the third qubit as the signal to be processed, the Toffoli gate negates the value of c in the output and thus it is a *NOT* gate.

Since any classical logic can be implemented with a combination of the AND and NOT gates, thus, the Toffoli gate is a universal gate for classical logic.

I hope you appreciate deeper the connection between the quantum gates and classical gates through this exercise.

16.6 Summary

We learned a few more complex quantum gates, namely the SWAP, Phase Shift, and Toffoli gates. The Toffoli gate is a 3-qubit gate. Similar to how we understand and treat simpler gates in the previous chapter, we can always define the gates as how they rotate the basis states of the corresponding space. These gates all have classical counterparts. In particular, they are the same as their classical counterparts if they are restricted to only process the basis states. Also, it is very important to note that for these quantum gates with classical counterparts, *the basis states are just swapping among each other or stay intact* just like in the classical case. We find that we can even use the Toffoli gate to construct any classical logic. In the next chapter, we will see a gate that has no classical counterpart.

Problems

16.1 Z-Gate

Use hand calculation and co-lab to show that the Z-gate converts $|-\rangle$ to $|+\rangle$.

16.2 Controlled-Z Gate

Construct a controlled-Z gate from the controlled phase shift gate. Then show that it is a CNOT gate in the $|+\rangle$ / $|-\rangle$ basis.

16.3 Implementation of a SWAP gate

A SWAP gate can be implemented using 3 CNOT gates in sequence. We will discuss this later. But try it now to challenge yourselves. The first and the third CNOT gates have the first bit as the control bit, but the second CNOT gate has the second qubit as the control bit. Try to create the matrix of the second CNOT gate and then prove the circuit is indeed a SWAP gate.

Chapter 17
Walsh–Hadamard Gate and Its Properties

17.1 Learning Outcomes

Understand that the Walsh–Hadamard gate has no classical counterpart; remember the special properties of the Walsh–Hadamard gate; be familiar with the mathematical skills for deriving the Walsh–Hadamard gate properties.

17.2 Walsh–Hadamard Gate

17.2.1 Definition

The **Walsh–Hadamard** gate is commonly called the **Hadamard** gate. The symbol of a Hadamard gate is $\boldsymbol{H}$. *Hadamard gate is so special because it has no classical counterpart*. The Hadamard gate is a 1-qubit gate, but, of course, we can also form a multi-qubit Hadamard gate through the tensor products of 1-qubit Hadamard gates. We will discuss this later. Firstly, let us look at the definition of a 1-qubit Hadamard gate.

$$\begin{aligned} \boldsymbol{H}\,|0\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle \\ \boldsymbol{H}\,|1\rangle &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle \end{aligned} \tag{17.1}$$

We see that when the Hadamard gate is applied to the basis vectors $|0\rangle$ and $|1\rangle$, they are rotated to a *superposition* of the basis vectors. Note that for the gates we have studied so far, when the quantum gates are applied to the basis vectors (e.g. NOT gate to $|0\rangle$), the basis vectors are only transformed (rotated) to other

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_17

basis vectors (e.g. $\boldsymbol{U_{NOT}}\,|0\rangle = |1\rangle$). Since the basis vectors have a one-to-one correspondence to classical logic binary values, those gates have their classical counterpart (e.g. $NOT(0) = 1$). However, there is no classical counterpart for the Hadamard gate because it is impossible to find a classical gate that gives "$\frac{1}{\sqrt{2}}(0+1)$" as "$0 + 1$" is meaningless in the classical Boolean logic.

We note that the Hadamard gate also equivalently transforms $|0\rangle$ and $|1\rangle$ to $|+\rangle$ and $|-\rangle$, respectively. As a review, let us remark that $|0\rangle$ and $|1\rangle$ are the eigenvectors of $\boldsymbol{\sigma_z}$ and $|+\rangle$ and $|-\rangle$ are the eigenvectors of $\boldsymbol{\sigma_x}$ and they can be represented as the superpositions of the others. We used a fake real-2D plane to illustrate this idea in Fig. 5.1 and repeat this in the inset of Fig. 17.1. While the fake 2D plane illustration is incomplete, the equations derived turned out to be correct. For example, using vector addition, we can easily show that $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.

17.2.2 Matrix

The matrix of $\boldsymbol{H}$ is

$$\boldsymbol{H} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{17.2}$$

We can understand it in this way. Since this is a 1-qubit gate for operating on a vector in the two-dimensional $\mathbb{C}^2$ space, the matrix (operator) must be 2×2. And we can easily prove that this is correct by doing the numerical substitutions.

$$\begin{aligned} \boldsymbol{H}\,|0\rangle &= \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ \boldsymbol{H}\,|1\rangle &= \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{aligned} \tag{17.3}$$

Again let us recall that we are working on the basis formed by the eigenvectors of $\boldsymbol{\sigma_z}$ when we write the operator and the vectors in their matrix and column form, respectively. Therefore, $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

17.2.3 Circuit

In general, for a 1-qubit vector, $|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, we have

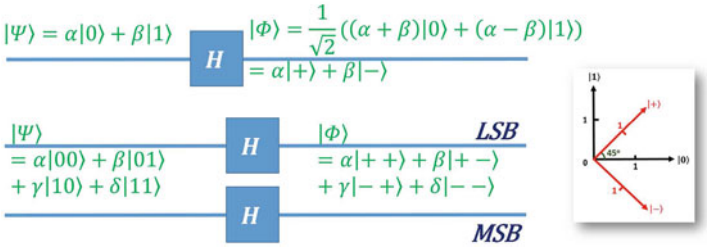


Fig. 17.1 Hadamard gate circuits. The inset shows the relationship between the $|+\rangle$ / $|-\rangle$ basis and the $|0\rangle$ / $|1\rangle$ basis on a fake real 2D plane

$$\begin{aligned} \boldsymbol{H}\,|\Psi\rangle &= \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}\alpha+\beta\\ \alpha-\beta\end{pmatrix} \\ &= \alpha\,|+\rangle + \beta\,|-\rangle \end{aligned} \tag{17.4}$$

I will leave it as an exercise for you to show the last step. You can do this by expressing the $|+\rangle$ and $|-\rangle$ as the superpositions of $|0\rangle$ and $|1\rangle$. However, I can also obtain the final expression easily by recalling how $\boldsymbol{H}$ rotates the basis states. That is, it will transform $|0\rangle$ to $|+\rangle$ and $|1\rangle$ to $|-\rangle$, respectively, based on Eq. (17.1). This is the insight you need to have in order to design a quantum algorithm.

Figure 17.1 shows the circuits of the Hadamard gate. Note that again the bottom qubit is the MSB in the multiple-qubit case.

17.3 Properties of the Hadamard Gate

17.3.1 Inverse of Hadamard Gate

The inverse of the Hadamard gate ($\boldsymbol{H^{-1}}$) equals the Hadamard gate itself, i.e.

$$\boldsymbol{H} = \boldsymbol{H^{-1}} \tag{17.5}$$

This is a special property that we will use in some quantum computing algorithms. Since for any matrix, when it is multiplied by its inverse, it becomes an identity matrix. Therefore, we have

$$\begin{aligned} \boldsymbol{H}\boldsymbol{H^{-1}} &= \boldsymbol{I} \\ \boldsymbol{H}\boldsymbol{H} &= \boldsymbol{I} \end{aligned} \tag{17.6}$$

This means that if you apply the Hadamard gate twice, the input state vector is unchanged!

The construction of the inverse of a general matrix is fair cumbersome and we will not discuss it here. But we will try to do so for a 2×2 matrix as it is relatively easy. The inverse of a 2×2 matrix, $\boldsymbol{U} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is

$$\boldsymbol{U^{-1}} = \frac{1}{ad - cb} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} \tag{17.7}$$

For a Hadamard gate, $a = b = c = \frac{1}{\sqrt{2}}$ and $d = -\frac{1}{\sqrt{2}}$, therefore,

$$\begin{aligned} \boldsymbol{H^{-1}} &= \frac{1}{\frac{1}{\sqrt{2}}\frac{-1}{\sqrt{2}} - \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}} \begin{pmatrix} \frac{-1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \\ &= \frac{1}{-\frac{1}{2} - \frac{1}{2}} \begin{pmatrix} \frac{-1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \boldsymbol{H} \end{aligned} \tag{17.8}$$

which is the same as Eq. (17.5).

17.3.2 Multiple-Qubit Hadamard Gate

Multiple-qubit Hadamard gate is just a simple tensor product of 1-qubit Hadamard gates. Therefore, for an n-qubit Hadamard gate, denoted by $\boldsymbol{H^{\otimes n}}$, it is found by:

$$\begin{aligned} \boldsymbol{H^{\otimes n}} &= \boldsymbol{H} \otimes \boldsymbol{H} \otimes \cdots \otimes \boldsymbol{H} \\ &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \cdots \otimes \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \end{aligned} \tag{17.9}$$

Note that in $\boldsymbol{H^{\otimes n}}$, n means there are n $\boldsymbol{H}'s$ instead of n $\otimes' s$. For example, when $n = 2$, a 2-qubit Hadamard gate is

$$\begin{aligned} \boldsymbol{H^{\otimes 2}} &= \boldsymbol{H} \otimes \boldsymbol{H} \\ &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} & 1 \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\ 1 \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} & -1 \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \end{pmatrix} \end{aligned}$$

$$= \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \tag{17.10}$$

When it is applied to $|00\rangle$, i.e. the first basis vector of the $\mathbb{C}^4$ space formed by two qubits, we get

$$\begin{aligned} \boldsymbol{H}^{\otimes 2}|00\rangle &= \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \\ &= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \end{aligned} \tag{17.11}$$

We see that it creates an *equal* superposition of all the basis vectors in the $\mathbb{C}^4$ space like in the 1-qubit case for the $\mathbb{C}^2$ space.

We want to gain more insight into how it operates on a general state, $|\Psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$. This is complicated as it will create superpositions for each of the basis states. On the other hand, we can use the fact that a Hadamard gate transforms $|0\rangle$ to $|+\rangle$ and $|1\rangle$ to $|-\rangle$ to help us gain more insight.

$$\begin{aligned} \boldsymbol{H}^{\otimes 2}|\Psi\rangle &= \boldsymbol{H} \otimes \boldsymbol{H}(\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle) \\ &= \boldsymbol{H} \otimes \boldsymbol{H}(\alpha|0\rangle \otimes |0\rangle + \beta|0\rangle \otimes |1\rangle + \gamma|1\rangle \otimes |0\rangle + \delta|1\rangle \otimes |1\rangle) \\ &= \alpha(\boldsymbol{H}|0\rangle) \otimes (\boldsymbol{H}|0\rangle) + \beta(\boldsymbol{H}|0\rangle) \otimes (\boldsymbol{H}|1\rangle) + \gamma(\boldsymbol{H}|1\rangle) \otimes (\boldsymbol{H}|0\rangle) \\ &\quad + \delta(\boldsymbol{H}|1\rangle) \otimes (\boldsymbol{H}|1\rangle) \\ &= \alpha|+\rangle \otimes |+\rangle + \beta|+\rangle \otimes |-\rangle + \gamma|-\rangle \otimes |+\rangle + \delta|-\rangle \otimes |-\rangle) \\ &= \alpha|++\rangle + \beta|+-\rangle + \gamma|-+\rangle + \delta|--\rangle \end{aligned} \tag{17.12}$$

We have used the fact that each 1-qubit Hadamard gate is only applied to the vector in the corresponding space. It shows that, by applying the Hadamard gate to any state, we only need to replace $|0\rangle$ by $|+\rangle$ and $|1\rangle$ by $|-\rangle$. Figure 17.1 shows the 2-qubit Hadamard gate circuit.

17.3.3 Properties of n-Qubit Hadamard Gate

The first property I would like to spend time to discuss is that when an n-qubit Hadamard gate ($\boldsymbol{H}^{\otimes n}$) is applied to $|0\rangle_n$, it will create an equal superposition of all basis vectors in the 2^n-dimensional space (the $\mathbb{C}^{2^n}$ space). $|0\rangle_n$ is the first basis

vector in the 2^n-D space. It is formed by the tensor product of n $|0\rangle's$ in the subspaces. This is a special basis vector because this is usually what we obtain after the *initialization of a quantum register*.

$$|0\rangle_n = |00\cdots0\rangle = |0\rangle \otimes |0\rangle \otimes \cdots \otimes |0\rangle \tag{17.13}$$

Applying an n-qubit Hadamard gate to $|0\rangle_n$, we have

$$\begin{aligned} \boldsymbol{H}^{\otimes n}|0\rangle_n &= \boldsymbol{H}^{\otimes n}|0\rangle \otimes |0\rangle \otimes \cdots \otimes |0\rangle \\ &= (\boldsymbol{H} \otimes \boldsymbol{H} \otimes \cdots \otimes \boldsymbol{H})|0\rangle \otimes |0\rangle \otimes \cdots \otimes |0\rangle \\ &= (\boldsymbol{H}|0\rangle) \otimes (\boldsymbol{H}|0\rangle) \otimes \cdots \otimes (\boldsymbol{H}|0\rangle) \end{aligned} \tag{17.14}$$

Here I am using the fact that each $\boldsymbol{H}$ is applied to the corresponding space. For example, applying a Hadamard gate to an electron spin might correspond to a certain sequence of external pulses. An n-qubit Hadamard gate can be just n copies of such pulses applying to n electrons in parallel. So it is natural that each $\boldsymbol{H}$ only operates on the corresponding subspace (belongs to the spin of the corresponding electron).

Based on Eq. (17.3), Eq. (17.14) becomes

$$\begin{aligned} \boldsymbol{H}^{\otimes n}|0\rangle_n &= (\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)) \otimes (\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)) \otimes \cdots \otimes (\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)) \\ &= \frac{1}{2^{\frac{n}{2}}}(|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes \cdots \otimes (|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \end{aligned} \tag{17.15}$$

Recalling the distribution law of the tensor products, the above equation can be expanded in the same way as how we distribute multiplication into parenthesis of addition. So this is nothing but just performing the permutations of 0's and 1's. Let us firstly expand the rightmost two terms. We have $(|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) = |0\rangle|0\rangle + |0\rangle|1\rangle + |1\rangle|0\rangle + |1\rangle|1\rangle = |00\rangle + |01\rangle + |10\rangle + |11\rangle$. We realize that it contains the binary representations of 0, 1, 2, and 3. If I include the third rightmost term, this term will distribute its $|0\rangle$ and $|1\rangle$ to $|00\rangle + |01\rangle + |10\rangle + |11\rangle$ from the left and it becomes $|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + |111\rangle$ by prepending "0" and "1" in front of the tensor product of the rightmost two terms. And it contains the binary representations of 0, 1, 2, 3, 4, 5, 6, and 7. We see that every time we multiply $|0\rangle + |1\rangle$ from the left, it doubles the number of the terms by prepending "0" and "1." Therefore, after all terms are expanded, Eq. (17.15) becomes

$$\begin{aligned} \boldsymbol{H}^{\otimes n}|0\rangle_n = \frac{1}{2^{\frac{n}{2}}}(&|00\cdots00\rangle + |00\cdots01\rangle + |00\cdots10\rangle + |00\cdots11\rangle + \cdots + \\ &|11\cdots10\rangle + |11\cdots11\rangle \end{aligned}$$

$$= \frac{1}{2^{\frac{n}{2}}}(|0\rangle + |1\rangle + |2\rangle + |3\rangle + \cdots + |2^n - 2\rangle + |2^n - 1\rangle)$$
$$= \frac{1}{2^{\frac{n}{2}}} \sum_{x=0}^{2^n-1} |x\rangle \tag{17.16}$$

Therefore, we can use the n-qubit Hadamard gate to create an equal superposition of all basis vectors from $|0\rangle_n$. Note that at the end, I use decimal representation in the *ket*'s for compactness.

Naturally, we will then ask what happens if the $\boldsymbol{H}^{\otimes n}$ is applied to an arbitrary basis vector $|y\rangle$. Note that we are talking about a basis vector but not a general 2^n-D vector, which is a superposition of the basis vectors. Here $|y\rangle$ can be represented in a binary (e.g. $|00101\rangle$) or decimal (e.g. $|5\rangle$) form. It really does not matter how it is represented *as long as* we know how many qubits are being used (e.g. $|5\rangle = |101\rangle$ for 3 qubits and $|5\rangle = |00101\rangle$ for 5 qubits). Again, *ket* is just a notation and we can write anything as long as there is no ambiguity. So the question we are investigating can be represented mathematically as $\boldsymbol{H}^{\otimes n}|y\rangle$. To derive the result, I will first represent y in its binary form.

$$|y\rangle = |y_{n-1}y_{n-2}\cdots y_1 y_0\rangle \tag{17.17}$$

Here, y_i, for $i = 0$ to $n-1$, are the binary digits and can only take "0" or "1." Recalling Eq. (13.1), $y_{n-1}y_{n-2}\cdots y_1 y_0$ is a string of numbers representing the value $y_{n-1} \times 2^{n-1} + y_{n-2} \times 2^{n-2} + \cdots + y_0 \times 2^0$. For example, in a 3-qubit case, $|4\rangle = |100\rangle$. So, $n = 3$, $y_2 = 1$, $y_1 = 0$, and $y_0 = 0$ and thus it represents $1 \times 2^{3-1} + 0 \times 2^{3-2} + 0 \times 2^0 = 4$. Based on what we discussed earlier, each $\boldsymbol{H}$ only applies to the corresponding qubit. However, the qubit may be $|0\rangle$ or $|1\rangle$. And it becomes $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, respectively, after applying the Hadamard gate (Eq. (17.3)). And, this is equivalent to

$$\boldsymbol{H}|y_i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + (-1)^{y_i}|1\rangle) \tag{17.18}$$

because $(-1)^{y_i} = 1$ and $(-1)^{y_i} = -1$ for $y_i = 0$ and $y_i = 1$, respectively. Therefore,

$$\begin{aligned}
\boldsymbol{H}^{\otimes n}|y\rangle &= \boldsymbol{H}^{\otimes n}|y_{n-1}y_{n-2}\cdots y_0\rangle \\
&= (\boldsymbol{H}|y_{n-1}\rangle) \otimes (\boldsymbol{H}|y_{n-2}\rangle) \otimes \cdots \otimes (\boldsymbol{H}|y_0\rangle) \\
&= (\frac{1}{\sqrt{2}}(|0\rangle + (-1)^{y_{n-1}}|1\rangle)) \otimes (\frac{1}{\sqrt{2}}(|0\rangle + (-1)^{y_{n-2}}|1\rangle)) \otimes \cdots \\
&\quad \otimes (\frac{1}{\sqrt{2}}(|0\rangle + (-1)^{y_0}|1\rangle))
\end{aligned}$$

$$= \frac{1}{2^{\frac{n}{2}}}(|0\rangle + (-1)^{y_{n-1}}|1\rangle) \otimes (|0\rangle + (-1)^{y_{n-2}}|1\rangle)$$
$$\otimes \cdots \otimes (|0\rangle + (-1)^{y_0}|1\rangle) \tag{17.19}$$

Now we are in a similar situation as in Eq. (17.15). If we expand the terms (using the distribution law of tensor product into the parentheses), we will again get the permutation of the 0's and 1's and all basis vectors $|x\rangle$ (from $|0\rangle$, $|1\rangle$, $|2\rangle$, to $\left|2^{n-1}\right\rangle$) will appear. Therefore, just like $\boldsymbol{H}^{\otimes n}|0\rangle_n$, we also obtain a superposition of all basis vectors in $\boldsymbol{H}^{\otimes n}|y\rangle$. However, the coefficients will not be constantly 1 (after neglecting the factor $\frac{1}{2^{\frac{n}{2}}}$). It will be either +1 or −1. And we see that only when there is $(-1)^{y_i}$ in the multiplication of the coefficients would it have a *chance* to get −1.

Let $|x\rangle = |x_{n-1}x_{n-2}\cdots x_1x_0\rangle$. We see that $(-1)^{y_i}$ will appear in the coefficient if $x_i = 1$. But $(-1)^{y_i}$ may be 1 or −1 if y_i is 0 and 1, respectively. Therefore, in order to get a −1 as the coefficient for $|x\rangle$ due to x_i, we need both $x_i = 1$ and $y_i = 1$, or (x_i AND y_i) =1 in classical logic. Finally, the final coefficient of $|x\rangle$ is 1 if there are odd number of (x_i AND y_i)=1. And this can be found by applying the classical XOR operation to all operations. We define an operation

$$x \cdot y = (x_{n-1}AND y_{n-1}) \oplus (x_{n-2}AND y_{n-2}) \oplus \cdots \oplus (x_0AND y_0) \tag{17.20}$$

Note that this is *not* a dot product in the regular sense. But it has a similar property of a dot product, namely that it performs AND operation elementwise (which is just the element-wise multiplication as x_i and y_i only take either 0 or 1) followed by XOR operation (which is just an addition modulus 2 operation). The addition modulus 2 operation means to take the remainder after dividing the addition by 2. For example, 1+1+1=3 and 3/2=1 remains 1. Therefore, 1+1+1 = 1. So in a more general sense, $x \cdot y$ indeed is a dot product if you treat x and y as the vectors and x_i and y_i as the corresponding elements/components. Therefore, we have

$$\boldsymbol{H}^{\otimes n}|y\rangle = \frac{1}{2^{\frac{n}{2}}}\sum_{x=0}^{2^n-1}(-1)^{x\cdot y}|x\rangle \tag{17.21}$$

To enhance the understanding, let us look at an example.

Example 17.1 Find $\boldsymbol{H}^{\otimes 2}|3\rangle$ using Eq. (17.21).

We see that $n = 2$. We may just plug in the numbers:

$$\boldsymbol{H}^{\otimes 2}|3\rangle = \frac{1}{2^{\frac{2}{2}}}\sum_{x=0}^{2^2-1}(-1)^{x\cdot 3}|x\rangle = \frac{1}{2}\sum_{x=0}^{3}(-1)^{x\cdot 3}|x\rangle$$
$$= \frac{1}{2}((-1)^{0\cdot 3}|0\rangle + (-1)^{1\cdot 3}|1\rangle + (-1)^{2\cdot 3}|2\rangle + (-1)^{3\cdot 3}|3\rangle)$$

For $n = 2$, we have $0_{10} = 00_2$, $1_{10} = 01_2$, $2_{10} = 10_2$, and $3_{10} = 11_2$. Therefore, based on Eq. (17.20)

$$\begin{aligned}
0 \cdot 3 &= (0AND1) \oplus (0AND1) = 0 \oplus 0 = 0 \\
1 \cdot 3 &= (0AND1) \oplus (1AND1) = 0 \oplus 1 = 1 \\
2 \cdot 3 &= (1AND1) \oplus (0AND1) = 1 \oplus 0 = 1 \\
3 \cdot 3 &= (1AND1) \oplus (1AND1) = 1 \oplus 1 = 0
\end{aligned}$$

Therefore,

$$\boldsymbol{H}^{\otimes 2} |3\rangle = \frac{1}{2}(|0\rangle - |1\rangle - |2\rangle + |3\rangle)$$

Example 17.2 Now use Eq. (17.19) to calculate $\boldsymbol{H}^{\otimes 2} |3\rangle$.

This can help us appreciate deeper how Eq. (17.21) is derived.

$$\begin{aligned}
\boldsymbol{H}^{\otimes 2} |3\rangle &= \boldsymbol{H}^{\otimes 2} |11\rangle \\
&= (\boldsymbol{H} |1\rangle) \otimes (\boldsymbol{H} |1\rangle) \\
&= \frac{1}{2^{\frac{2}{2}}}(|0\rangle + (-1)^1 |1\rangle) \otimes (|0\rangle + (-1)^1 |1\rangle) \\
&= \frac{1}{2}(1 |0\rangle + (-1) |1\rangle) \otimes (1 |0\rangle + (-1) |1\rangle) \\
&= \frac{1}{2}(1 \times 1 |0\rangle |0\rangle + 1 \times (-1) |0\rangle |1\rangle + (-1) \times 1 |1\rangle |0\rangle \\
&\quad + (-1) \times (-1) |1\rangle |1\rangle) \\
&= \frac{1}{2}(|00\rangle - |01\rangle - |10\rangle + |11\rangle)
\end{aligned}$$

For $|00\rangle$, x_1=0 and x_0=0. Its coefficients come from that of $|0\rangle$ after applying $\boldsymbol{H}$, which is $+1$. Therefore, regardless of the y values, it must have a final coefficient of $+1$, which is the products of two $+1$. For $|11\rangle$, x_1=1 and x_0=1. The coefficients contributing to the final coefficient of $|11\rangle$ come from those of $|1\rangle$ after applying $\boldsymbol{H}$. Since $y = 3$, these coefficients are -1. Here we see that we need x_i and y_i to be both 1 in order to get a factor of -1. So the final coefficient of $|11\rangle$ is $+1$, which is a product of two -1. For $|01\rangle$, $x_1 = 0$ and $x_0 = 1$. Its final coefficients is the product of the coefficients of $|0\rangle$ and $|1\rangle$ after applying $\boldsymbol{H}$, which are $+1$ and -1, respectively. Therefore, the final coefficient is -1.

17.4 Summary

We have spent a whole chapter just learning the Hadamard gate. This is because it is one of the most important gates in quantum computing. It does *not* have classical counterparts. It creates superposition after applying to a basis vector. Indeed, it is used to create superpositions in many quantum algorithms by acting on the initialized quantum register at $|0\rangle_n$. *Using the superposition state, we can then perform the parallel computation to speed up many operations or use it for constructive or destructive interferences to get the answers we look for.* We also learned some very important mathematical skills in deriving the Hadamard gate properties, which will be used again in more sophisticated algorithms. We also learned that the inverse of the Hadamard gate is itself.

Problems

17.1 1-qubit Hadamard Gate
Show Eq. (17.4) by substituting $|+\rangle$ and $|-\rangle$ in terms of $|0\rangle$ and $|1\rangle$.

17.2 *n*-qubit Hadamard Gate Numerical Substitution 1
Use Eq. (17.21) to compute $\boldsymbol{H}^{\otimes 2}\,|2\rangle$. What is n in this case?

17.3 *n*-qubit Hadamard Gate Numerical Substitution 2
Use expansion method in Eq. (17.19) to compute $\boldsymbol{H}^{\otimes 2}\,|2\rangle$. Try to identify the fact that only when x_i and y_i are both 1 would it get a contribution of -1 to the final coefficient.

17.4 *n*-qubit Hadamard Gate Numerical Substitutions 3
Compute $\boldsymbol{H}^{\otimes 2}\,|2\rangle$ but use matrix multiplications.

Chapter 18
Two Quantum Circuit Examples

18.1 Learning Outcomes

Able to describe the meaning of little-endian convention; be aware of the existence of big-endian convention; contrast the flow direction of the vector in an equation and a circuit; able to construct a quantum circuit based on a given simple equation; and able to understand a quantum circuit through both matrix method and intuition.

18.2 Quantum Circuit for Rotating Basis

Let us consider a circuit that can be used to transform the standard $\mathbb{C}^4$ basis to the Bell states introduced in Chap. 13 and discussed further in Chap. 14. The standard basis vectors are $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. And they can be written in the column form as

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}; \quad |01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}; \quad |10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}; \quad |11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{18.1}$$

As a reminder, this column representation is trivial because the value in each row represents the amount of the corresponding basis vector the vector in question has. For example, $|00\rangle$ is 100% of the zeroth basis vector, i.e. $|00\rangle$. It has no component

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_18

of other basis vectors and thus $|00\rangle = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}$. The Bells states are given in Eq. (13.13) and repeated here for convenience:

$$\begin{aligned}
\left|\Phi^{+}\right\rangle &= \frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\0\\1\end{pmatrix}; \quad \left|\Phi^{-}\right\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\0\\-1\end{pmatrix} \\
\left|\Psi^{+}\right\rangle &= \frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\1\\0\end{pmatrix}; \quad \left|\Psi^{-}\right\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\-1\\0\end{pmatrix}
\end{aligned} \tag{18.2}$$

Here I assume we already know which circuit can perform the transformation. It is given as $\boldsymbol{U}_{XOR}(\boldsymbol{H}\otimes\boldsymbol{I})$. $\boldsymbol{U}_{XOR}$ is a 2-qubit quantum gate (i.e. the $CNOT$ gate in Eq. (15.12)). $\boldsymbol{H}$ is a 1-qubit Hadamard gate and $\boldsymbol{I}$ is a 1-qubit identity gate (i.e. doing nothing). Therefore, $\boldsymbol{H}\otimes\boldsymbol{I}$ forms a 2-qubit gate to rotate the input vector, which is then further rotated by the $\boldsymbol{U}_{XOR}$ gate. Let us perform some numerical substitutions to check if this is correct.

Example 18.1 Find the matrix representation of $\boldsymbol{U}_{XOR}(\boldsymbol{H}\otimes\boldsymbol{I})$.

From Eqs. (15.14) and (17.2), we have

$$\begin{aligned}
\boldsymbol{U}_{XOR}\boldsymbol{H}\otimes\boldsymbol{I} &= \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1&1\\1&-1\end{pmatrix}\otimes\begin{pmatrix}1&0\\0&1\end{pmatrix} \\
&= \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1\begin{pmatrix}1&0\\0&1\end{pmatrix} & 1\begin{pmatrix}1&0\\0&1\end{pmatrix}\\ 1\begin{pmatrix}1&0\\0&1\end{pmatrix} & -1\begin{pmatrix}1&0\\0&1\end{pmatrix}\end{pmatrix} \\
&= \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1&0&1&0\\0&1&0&1\\1&0&-1&0\\0&1&0&-1\end{pmatrix} \\
&= \frac{1}{\sqrt{2}}\begin{pmatrix}1&0&1&0\\0&1&0&1\\0&1&0&-1\\1&0&-1&0\end{pmatrix}
\end{aligned} \tag{18.3}$$

Example 18.2 Show $\boldsymbol{U}_{XOR}(\boldsymbol{H}\otimes\boldsymbol{I})\,|01\rangle = |\Psi^+\rangle$.

$$\boldsymbol{U}_{XOR}(\boldsymbol{H}\otimes\boldsymbol{I})\,|01\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1&0&1&0\\0&1&0&1\\0&1&0&-1\\1&0&-1&0\end{pmatrix}\begin{pmatrix}0\\1\\0\\0\end{pmatrix}$$

$$= \frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\1\\0\end{pmatrix} = |\Psi^+\rangle \tag{18.4}$$

We are very familiar with the matrix operations now. How to construct a quantum circuit for this? We should note that *in matrix operations, the states evolve from the right to the left in the equation*, but *in a quantum circuit, the states evolve from the left to the right in the circuit diagram*. Therefore, we should put the $\boldsymbol{H}\otimes\boldsymbol{I}$ gate on the left and the $\boldsymbol{U}_{XOR}$ on the right in the circuit diagram. Figure 18.1 shows the corresponding quantum circuit. *There are three things we want to pay attention to*.

Firstly, besides drawing the circuit in the conventional way we have been using (i.e. the MSB at the bottom), I also draw the one commonly used in other textbooks. They are exactly the same if you label the MSB correctly and associate the rightmost qubit in the bra–ket notation with the LSB. Both approaches are the so-called little-endian convention. In this example, the $\boldsymbol{I}$ operator (for the rightmost qubit) is always associated with the LSB. If one associates the rightmost qubit to the MSB, then it is called the **big-endian** convention. We need to be careful in converting these two cases to each other.

Secondly, as emphasized, the state evolves from left to right in the circuit.

Thirdly, I give a general expression of the circuit and you can see that it rotates each regular basis vector to the corresponding Bell state. And we can prove it as the following.

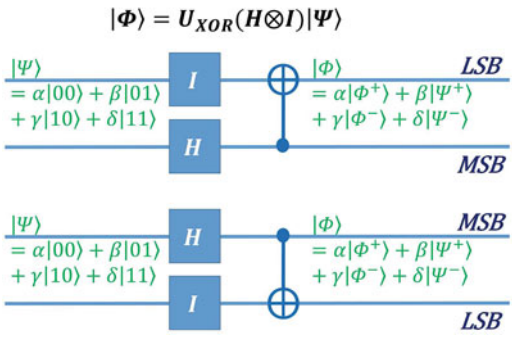


Fig. 18.1 The quantum circuits for rotating regular $\mathbb{C}^4$ basis vectors to Bell states. The top one is the convention we have been using (IBM-Q) with MSB at the bottom and the bottom one is the other version often seen in other books. Both are using little-endian convention

Example 18.3 Show that $\boldsymbol{U}_{XOR}(\boldsymbol{H} \otimes \boldsymbol{I})(\alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + \delta\,|11\rangle) = \alpha\left|\Phi^{+}\right\rangle + \beta\left|\Psi^{+}\right\rangle + \gamma\left|\Phi^{-}\right\rangle + \delta\left|\Psi^{-}\right\rangle$.

For the left-hand side, we have

$$\begin{aligned}
&\boldsymbol{U}_{XOR}(\boldsymbol{H} \otimes \boldsymbol{I})(\alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + \delta\,|11\rangle) \\
&= \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} \\
&= \frac{1}{\sqrt{2}}\begin{pmatrix} \alpha+\gamma \\ \beta+\delta \\ \beta-\delta \\ \alpha-\gamma \end{pmatrix}
\end{aligned} \tag{18.5}$$

For the right hand side, we have

$$\begin{aligned}
&\alpha\left|\Phi^{+}\right\rangle + \beta\left|\Psi^{+}\right\rangle + \gamma\left|\Phi^{-}\right\rangle + \delta\left|\Psi^{-}\right\rangle \\
&= \frac{\alpha}{\sqrt{2}}\begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} + \frac{\beta}{\sqrt{2}}\begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} + \frac{\gamma}{\sqrt{2}}\begin{pmatrix} 1 \\ 0 \\ 0 \\ -1 \end{pmatrix} + \frac{\delta}{\sqrt{2}}\begin{pmatrix} 0 \\ 1 \\ -1 \\ 0 \end{pmatrix} \\
&= \frac{1}{\sqrt{2}}\begin{pmatrix} \alpha+\gamma \\ \beta+\delta \\ \beta-\delta \\ \alpha-\gamma \end{pmatrix}
\end{aligned} \tag{18.6}$$

Therefore, they are the same and the equation is proved.

18.2.1 Run on IBM-Q

Let us build the circuit on IBM-Q. Figure 18.2 shows the circuit built on IBM-Q. $q0$ and $q1$ refer to the right- and left-most qubits, respectively, in a bra–ket notation. Note that it uses the *little-endian* convention and therefore, the bottom line is the MSB (i.e. the leftmost qubit $q1$ is associated with the MSB). It also has 2 classical bits to store the measurement results and it also uses the little-endian convention. In the setup, we assign the measurement results of $q0$ and $q1$ to the classical register bit 0 and 1, respectively. Therefore, if we get $(01)_2$ in the classical bits (this is a binary number and I add the subscript 2 to remind you), it means $q0$ is measured to be 1 and $q1$ is measured to be 0. These are the convention we have been using in

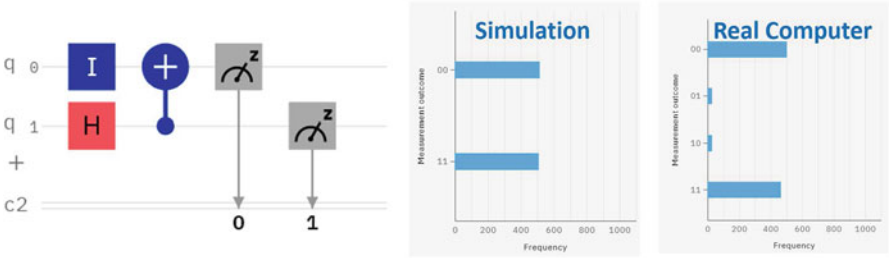


Fig. 18.2 The quantum circuit in Fig. 18.1 built in IBM-Q. The simulation result is shown in the middle and the real quantum computer execution result is shown on the right

this book so far. You may recreate this circuit by dragging the icons to the places as shown. For the $\boldsymbol{U}_{XOR}$ (which is CNOT), you need to double click on the icon to set it to have the MSB ($q1$) controlling the LSB ($q0$).

By default, the input state to any quantum circuit is $|0\rangle_n$ for an n-qubit circuit. In this case, it is $|00\rangle$. Therefore, we expect the output to be $\left|\Phi^+\right\rangle$ as discussed. And we know that $\left|\Phi^+\right\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$; therefore, we should only get $|00\rangle$ and $|11\rangle$ in the measurements. Indeed, if the circuit is submitted to a simulator, only $|00\rangle$ and $|11\rangle$ are found (non-zero and equal values in the histogram).

However, if this circuit is submitted to a real quantum computer (e.g. *ibmq_casablanca* that might have retired when you read this book), it also has non-zero values for $|01\rangle$ and $|10\rangle$. This is due to the error in the quantum computer, which effectively rotated the state to a state with $|01\rangle$ and $|10\rangle$ components. This is a simple quantum circuit. We can imagine that for any useful algorithm, the error will accumulate and eventually make the quantum computer useless. Therefore, it is very important to have *error corrections*, which is a more advanced topic and will not be covered in this book.

18.3 Quantum Circuit for Implementing a SWAP Gate

The definition of a SWAP gate is given in Eq. (16.1) and its circuit is shown in Fig. 16.1. The equation is repeated here for convenience, $\boldsymbol{U_{SWAP}}\,|ab\rangle = |ba\rangle$. As we mentioned, the SWAP gate seems to be very simple. And one might think it is as simple as just crossing over two wires in a classical circuit. So maybe it is as easy as just swapping two electrons physically. However, the meaning is deeper than that. What it means is that I want electron 1 to acquire the state of electron 2 and electron 2 to acquire the state of electron 1 after swapping. Sometimes we may get confused and argue that the electrons are indistinguishable, so there is no difference between these two approaches. But let us do not make this too complicated and just assume the situation that the electrons are in two different atoms or in two different quantum

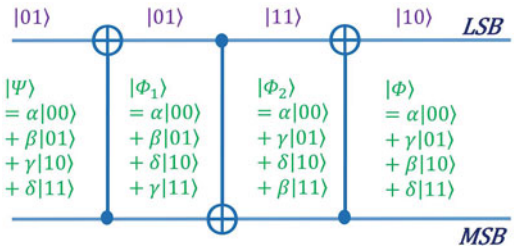


Fig. 18.3 Implementation of a SWAP gate using CNOT gates. The purple ket's on top shows an example of how a single basis vector evolves in the circuit

wells, which is usually the case in a real quantum computer. In this case, they are distinguishable. How do we swap the states of the electrons in two different atoms? In principle, you can also argue that we can just physically swap them. But this is not easy technically, especially if it is a larger system with many other electrons involved. Therefore, we need to implement a quantum circuit to perform the SWAP operation.

Here again, I will give you the circuit. Our goal is just to understand it, appreciate it, and implement it. The circuit is given in Fig. 18.3.

We are pretty good at understanding the quantum gate now. Let us try to understand the circuit intuitively. The circuit is composed of 3 CNOT ($\boldsymbol{U}_{XOR}$) gates. The first one on the left has the bottom (MSB) as the control qubit and the top (LSB) as the target qubit. This means, for the basis vectors, if the control qubit is 0, it will do nothing to them. If the control qubit is 1, it will flip the LSB. Therefore, $|00\rangle$ and $|01\rangle$ will not be changed, but $|10\rangle$ will become $|11\rangle$ and $|11\rangle$ will become $|10\rangle$. This is equivalent to swapping the coefficients of $|10\rangle$ and $|11\rangle$ in a general vector, which we have discussed when we introduced the CNOT gate. Therefore, for a general input, $|\Psi\rangle = \alpha\,|00\rangle + \beta\,|01\rangle + \gamma\,|10\rangle + \delta\,|11\rangle$, and after the first CNOT gate, it becomes $|\Phi_1\rangle = \alpha\,|00\rangle + \beta\,|01\rangle + \delta\,|10\rangle + \gamma\,|11\rangle$ as δ and γ are swapped.

For the second CNOT gate, the control qubit is LSB. Therefore, it means $|00\rangle$ and $|10\rangle$ will not be changed because the LSB is 0. $|01\rangle$ will become $|11\rangle$ and $|11\rangle$ will become $|01\rangle$ because the LSB is 1 and it will flip the MSB. Therefore, for a general vector, it will swap the coefficients of $|01\rangle$ and $|11\rangle$. So the β and γ in $|\Phi_1\rangle$ are swapped and it becomes $|\Phi_2\rangle = \alpha\,|00\rangle + \gamma\,|01\rangle + \delta\,|10\rangle + \beta\,|11\rangle$.

Finally, the rightmost CNOT gate swaps the coefficients of $|10\rangle$ and $|11\rangle$ just as the leftmost CNOT gate. So the β and δ in $|\Phi_2\rangle$ are swapped and it becomes $|\Phi\rangle = \alpha\,|00\rangle + \gamma\,|01\rangle + \beta\,|10\rangle + \delta\,|11\rangle$. This is the result of a SWAP gate as shown in Fig. 16.1. And let us recall the function of a SWAP gate again. It is to swap the values of the LSB and MSB in the basis states. Therefore, only $|10\rangle$ and $|01\rangle$ will change and in a general vector, their coefficients are swapped. And this is exactly what this circuit does to the input vector $|\Psi\rangle$ coming from the far left.

Based on what we have discussed, we might be able to guess how people came up with this circuit for a SWAP gate. The SWAP gate effectively swaps the coefficients of $|10\rangle$ and $|01\rangle$. We know that a CNOT gate swaps the coefficients of $|10\rangle$ and $|11\rangle$ if the MSB is the control qubit and it swaps the coefficients of $|01\rangle$ and $|11\rangle$ if the LSB is the control qubit. Both of them involve $|11\rangle$. Therefore, we can treat $|11\rangle$ as a

"middleman" or a "buffer" to complete the swapping using only CNOT gates. This is similar to how we swap the contents of two memories through copy operations by using a buffer in a classical computer. We need this type of insight if we want to invent any new quantum circuits.

Although we already have the intuition on how the SWAP circuit works, it is instructional to understand it from the matrix perspective. Again, in an equation, the input vector evolves from the right to the left. Therefore, the matrix of the leftmost CNOT in the circuit appears at the rightmost position of the equation, i.e. $\boldsymbol{U}_{XOR,right}\boldsymbol{U}_{XOR,center}\boldsymbol{U}_{XOR,left}$, where "right," "center," and "left" indicate the locations of the CNOT gate in the *circuit* in Fig. 18.3.

Example 18.4 Find the matrix representation of the circuit in Fig. 18.3.

$$\begin{aligned}\boldsymbol{U}_{XOR,right}\boldsymbol{U}_{XOR,center}\boldsymbol{U}_{XOR,left} &= \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\begin{pmatrix}1&0&0&0\\0&0&0&1\\0&0&1&0\\0&1&0&0\end{pmatrix}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\\ &= \begin{pmatrix}1&0&0&0\\0&0&0&1\\0&1&0&0\\0&0&1&0\end{pmatrix}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix} = \begin{pmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{pmatrix}\\ &= \boldsymbol{U}_{SWAP}\end{aligned} \tag{18.7}$$

which is the same as Eq. (16.3).

How do we obtain the matrix for $\boldsymbol{U}_{XOR,center}$, which is a CNOT with the LSB as the control qubit? We obtain this by the fact that it will swap the coefficients of $|01\rangle$ (the second basis vector) and $|11\rangle$ (the fourth basis vector) and thus it is different from the identity matrix by swapping the second and the fourth rows. Try to follow how we derived the matrix of CNOT with MSB as the control qubit in Chap. 15 to see if you can get $\boldsymbol{U}_{XOR,center}$.

Finally, let us try to appreciate the circuit from another perspective.

Example 18.5 Assume the input is $|01\rangle$ (Fig. 18.3), how does it evolve?

Note that the MSB is 0, and therefore, the first CNOT gate has no effect and it is still $|01\rangle$ after the first CNOT gate. However, the LSB is 1 and thus it will flip the value of the MSB from 0 to 1 after the second CNOT gate, which has the LSB as the control qubit, and thus it becomes $|11\rangle$. Finally, since now the MSB is 1, it will flip the LSB to 0 after the last CNOT gate and thus the state becomes $|10\rangle$. It is thus working as a SWAP gate.

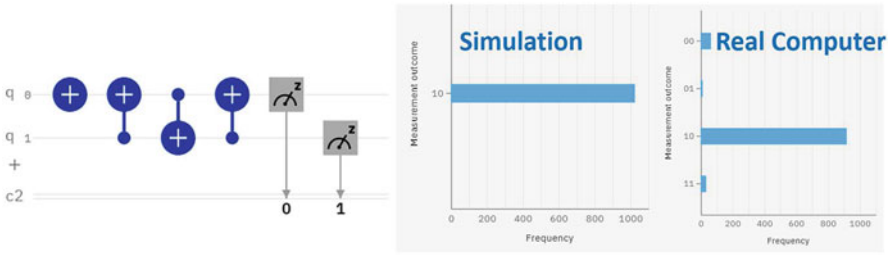


Fig. 18.4 The quantum circuit in Fig. 18.3 built in IBM-Q. The simulation result is shown in the middle and the real quantum computer execution result is shown on the right

18.3.1 Run on IBM-Q

We implement the circuit in Fig. 18.3 using IBM-Q and the results are shown in Fig. 18.4. An additional NOT gate is added to the LSB on the left of the circuit so that the input to the SWAP gate circuit is $|01\rangle$ (as a reminder, the initial vector going into the whole circuit is always initialized to $|0\rangle_n$ for an n-qubit circuit). Although I only added a NOT gate to the LSB, effectively, I also included an identity gate to the MSB. So the gate I inserted to this 2-qubit system is actually $\boldsymbol{I} \otimes \boldsymbol{X}$. We expect to obtain $|10\rangle$ as output after the SWAP circuit. Indeed, when this circuit is executed using a simulator, all the measurement results are "10," which means that the output vector only has the $|10\rangle$ component. However, when it is executed in a real quantum computer, it also has components of $|00\rangle$, $|01\rangle$, and $|11\rangle$ due to errors.

18.4 Summary

In this "step," we have constructed two simple quantum circuits. We learned that the input vectors evolve from the right to the left in an equation but evolve from the left to the right in a circuit diagram. We also need to remind ourselves that a quantum circuit, *unlike* a classical circuit, is not a layout of real space. It describes how a state evolves as a function of time from the left to the right. Therefore, a SWAP gate cannot be simply implemented by crossing over two wires. We tried to understand the circuits based on intuition, matrix multiplications, and numerical substitutions. We also implemented them in IBM-Q and found that the errors are not negligible even for such simple circuits. In this chapter, we also emphasized again the importance of tracking the MSB and the LSB in the circuit diagram.

Problems

18.1 SWAP Gate Implementation

We may change the implementation of the SWAP gate using LSB-controlling CNOT gate followed by MSB-controlling CNOT gate and then followed by LSB-controlling CNOT gate. Please repeat what we have done in the main text and prove that using the intuition method and matrix multiplication.

18.2 LSB-Controlling CNOT Gate

Repeat the definition and derivation of the CNOT gate matrix in Chap. 15 to obtain the matrix of LSB-controlling CNOT gate. That is to derive $\boldsymbol{U}_{XOR,center}$ in Eq. (18.7).

18.3 Quantum Computing Exercise 1

Run the basis transformation circuit in Fig. 18.2 on IBM-Q using real hardware and simulation. Do necessary modification to change the input to $|11\rangle$ and check the output.

18.4 Quantum Computing Exercise 2

Run the SWAP circuit in Fig. 18.3 on IBM-Q using real hardware and simulation. Do necessary modification to change the input to $|10\rangle$ and check the output.

18.5 Quantum Computing Exercise 3

Run the SWAP circuit in Fig. 18.3 on IBM-Q using real hardware and simulation but use the SWAP gate directly instead of using CNOT gates. Compare the outputs to Fig. 18.4.

Chapter 19
No-Cloning Theorem and Quantum Teleportation I

19.1 Learning Outcomes

Able to prove No-Cloning Theorem and describe which type of quantum states cannot be cloned; Understand quantum teleportation and compare against "real" teleportation; Understand the limitation of the simplified version of quantum teleportation; Know how to measure a qubit in its $|+\rangle$ / $|-\rangle$ basis using the apparatus in the $|0\rangle$ / $|1\rangle$ basis.

19.2 No-Cloning Theorem

We have finished the discussion of quantum gates. We will start discussing quantum algorithms. One of the first algorithms we will discuss is the **Quantum Teleportation**, although not everyone will agree that it is a type of algorithm. But an algorithm is a process to perform certain computations, however simple, complex, local, and non-local it is. So I will still call this our first quantum algorithm.

Before that, we need to study a very important property of quantum mechanics, namely the **No-Cloning Theorem**. It says that not *all* quantum states can be cloned or copied. This is different from our classical experience. We can copy the contents of any classical register or memory to another classical register or memory. We do this every day when we copy a document from our cell phone to our laptop or when we submit the assignment electronically. But quantum mechanics tells us that we cannot copy any arbitrary quantum state from a system to another identical system. This looks absurd but, philosophically, it makes sense. In the old time, when we were not able to submit an assignment electronically, we put the paper into our instructor's office. So once I submitted it, I lose the copy of my paper. Of course, I can make a photocopy of my assignment before submission. But this is not a real copy (clone) as it is on a different paper and thus the arrangement of the molecules

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_19

and electrons is not exactly the same. So we are familiar with the no-cloning theorem since ancient times. In the electronic age, while we can copy the information, we are not copying the matters. The memory in my computer is not exactly the same as the memory in my cell phone, at least at the molecular level. So, I am not really cloning a system when I copy a file from my cell phone to my computer. I only copy the information.

At the microscopic level, where quantum mechanics dominates, we know that particles are identical. That electron in your body is the same as this electron in my body. They are both electrons mathematically. However, they might have different states (e.g. $|0\rangle$, $|1\rangle$, or any linear combinations of $|0\rangle$ and $|1\rangle$). If I am able to copy the state of the electron in your body to my body, although I might be just thinking of copying the information like how I copy a picture from my phone to my laptop, since the electrons are identical, it is not different from cloning an electron. The effect is just as cloning my assignment, not just with the same solutions but also on the same matter (same arrangement of molecules). So this is consistent with our daily experience since the old time that cloning is impossible. This is not rigorous logic. I just want to point out that non-cloning is not a very absurd phenomenon.

You might not agree with my philosophy (or my sentiment to be more appropriate). But I hope you agree that copying a state in the microscopic scale has more meaning than just copying information on the macroscopic scale.

Now let me show you the proof of the no-cloning theorem by finding what can be cloned. Assume there are two electrons, A and B. Their initial states are $|\Psi\rangle_A$ and $|0\rangle_B$. I do not know what $|\Psi\rangle_A$ is but I know it must be a linear combination of $|0\rangle$ and $|1\rangle$. That is

$$|\Psi\rangle_A = \alpha\,|0\rangle_A + \beta\,|1\rangle_A \tag{19.1}$$

Note that subscripts A and B are used to make it clear that we have two subspaces due to electron A and electron B. The state of the whole system (i.e. electrons A and B) is a tensor product of both, which is $|\Psi\rangle_A \otimes |0\rangle_B = |\Psi\rangle_A\,|0\rangle_B$.

The question is can I clone the state (wavefunction) from electron A to electron B? If so, it means that after cloning, the state of the whole system becomes $|\Psi\rangle_A\,|\Psi\rangle_B$. In a classical computer, we need to read before copying. But in a quantum computer, we cannot clone by reading the state first because the state will collapse to either $|0\rangle$ or $|1\rangle$, after which we would have destroyed the original state, and thus it is not a cloning operation. *Cloning must be done through an operation (not a measurement)* and we can write it as an operator (matrix) like other quantum gates, and it must be unitary (if it exists). It must also be a rotation of the vector in the corresponding hyperspace. Let the cloning operator be $\boldsymbol{L}$, which is an operator in the $\mathbb{C}^4$ space. We have

$$\boldsymbol{L}(|\Psi\rangle_A\,|0\rangle_B) = |\Psi\rangle_A\,|\Psi\rangle_B \tag{19.2}$$

We can expand the final state as

$$\begin{aligned}\boldsymbol{L}(|\Psi\rangle_A|0\rangle_B) &= |\Psi\rangle_A|\Psi\rangle_B = (\alpha|0\rangle_A + \beta|1\rangle_A) \otimes (\alpha|0\rangle_B + \beta|1\rangle_B) \\ &= (\alpha|0\rangle_A \alpha|0\rangle_B + \alpha|0\rangle_A \beta|1\rangle_B + \beta|1\rangle_A \alpha|0\rangle_B + \beta|1\rangle_A \beta|1\rangle_B) \\ &= \alpha^2|0\rangle_A|0\rangle_B + \alpha\beta|0\rangle_A|1\rangle_B + \beta\alpha|1\rangle_A|0\rangle_B + \beta^2|1\rangle_A|1\rangle_B \\ &= \alpha^2|00\rangle + \alpha\beta|01\rangle + \beta\alpha|10\rangle + \beta^2|11\rangle \end{aligned} \tag{19.3}$$

In the last line, I omit the subscript again because we know that the first electron is labeled as A and the second one is B.

On the other hand, the initial state before cloning can also be written as

$$\begin{aligned}|\Psi\rangle_A|0\rangle_B &= (\alpha|0\rangle_A + \beta|1\rangle_A)|0\rangle_B = \alpha|0\rangle_A|0\rangle_B + \beta|1\rangle_A|0\rangle_B \\ &= \alpha|00\rangle + \beta|10\rangle \end{aligned} \tag{19.4}$$

Since the cloning operator is just a regular operator, it obeys the rules of matrix-vector multiplication including the linearity and distribution law. Therefore, after cloning on the state in Eq. (19.4), we have the following:

$$\begin{aligned}\boldsymbol{L}(\alpha|00\rangle + \beta|10\rangle) &= \boldsymbol{L}\alpha|00\rangle + \boldsymbol{L}\beta|10\rangle \\ &= \alpha|00\rangle + \beta|11\rangle \\ &= \alpha|00\rangle + 0|01\rangle + 0|10\rangle + \beta|11\rangle \end{aligned} \tag{19.5}$$

where we use the distribution rule in the first line and use the definition of cloning in the second line (i.e. $\beta|1\rangle_A|0\rangle_B$ becomes $\beta|1\rangle_A|1\rangle_B$). But Eqs. (19.3) and (19.5) are expected to be the same because they have the same initial state and the same cloning operator. Comparing the coefficients of $|01\rangle$, it means $\alpha\beta = 0$, which further means either $\alpha = 0$ or $\beta = 0$. This means that *we can only clone the basis vectors* $|1\rangle$ ($\alpha = 0$) *or* $|0\rangle$ ($\beta = 0$) but *not* any general vector. And this is consistent with our expectation. If electron A is in a basis state ($|0\rangle$ or $|1\rangle$), we can read it without destroying its state (because the basis state will stay the same after any measurement) and then just use any appropriate method to set electron B to that basis state to complete the cloning process.

Therefore, *it is possible to clone the basis states but NOT other superposition states*.

The no-cloning theorem has many important applications in quantum computing and quantum communication. For example, an eavesdropper will not be able to copy your message without destroying your original information. We can thus use this to avoid eavesdropping in quantum communication. The no-cloning theorem also prevents the design of any quantum algorithms that will allow useful information to propagate faster than the speed of the light. Indeed, the discovery of the no-cloning theorem was due to a paper in which an algorithm was invented to allow information

to propagate faster than the speed of the light by assuming it is possible to clone a quantum state.

19.3 Quantum Teleportation

What is *teleportation*? It is the transfer of matters and energies without traversing the space in between. This also leads us to associate the transfer of matters and energies faster than the speed of light. We know this is contradicting the Theory of Relativity. In this section, we want to explore the relationship between **quantum teleportation** and hypothetical teleportation. We will understand the algorithm and find out if quantum teleportation will allow information to be transferred at a speed faster than light.

19.3.1 A Simplified Version

To make it easier to understand, we will start with a simplified version that performs quantum teleportation by assuming the existence of an ideal CNOT gate that can act over two qubits separated by a very long distance. After that, we will reuse a part of this algorithm in a useful quantum teleportation circuit in the next chapter.

Assume Alice has an electron with the state $|\Psi\rangle_A = \alpha|0\rangle_A + \beta|1\rangle_A$. This is a general state that no one knows what α and β are. The purpose of quantum teleportation is to transfer the state to Bob's electron, which might have an initial state of $|0\rangle_B$. Here I use subscripts A and B to make it clear that the states belong to the electrons of Alice and Bob, respectively. We want to transfer the state of Alice's electron to Bob in no time regardless of the distance between them! To ensure you will continue to read instead of thinking that I am teaching you pseudoscience, let me explain the essence first and then show you why what I just said does not contradict the Laws of Physics.

First of all, *why it is called teleportation*? In my opinion, since electrons are identical, if Alice can transfer the state of her electron to Bob's electron, the *final* result is not different from moving her electron to Bob's hand. Therefore, the final result is just like the hypothetical teleportation. Assume I am made up of 1000 particles. Each of the particles has a certain quantum state. Therefore, the 1000 particles with their respective states define me. If I can transfer the states of these 1000 particles to another 1000 particles on Mars, I believe the 1000 particles will act like me (same shape, same mentality, same emotion, etc.). The net result is that I am teleported to Mars!

However, do not forget the **no-cloning theorem**. When the state of Alice's electron is transferred to Bob's electron, the original state in Alice's electron will be changed (i.e. no longer $|\Psi\rangle$). Otherwise, we violate the no-cloning theorem. Similarly, if I am quantum teleported to Mars, the 1000 particles on the Earth will

lose their original states and I am no longer who I am on the Earth. Therefore, quantum teleportation indeed has the sense of teleportation in which cloning does not happen.

In the quantum teleportation process, it seems that the state information will be teleported in no time. However, the information can only be confirmed (i.e. completion of the information transfer process) through two classical communications. Since classical communication cannot be faster than the speed of light, the whole quantum teleportation process is only completed at a speed slower than light. Therefore, it does not violate the Theory of Relativity. But we still call it teleportation because the action of copying the quantum state is instantaneous regardless of the distance. *In the simplified version, only one classical communication is needed.*

Now let us start describing the algorithm and please try to associate it to what I have just mentioned.

Step 1: Entanglement Between $|\Psi\rangle_A$ *and* $|0\rangle_B$

The first step is to create entanglement between Alice's and Bob's electrons (Fig. 19.1). We can use a CNOT gate ($\boldsymbol{U}_{XOR}$) to achieve this purpose. Before the CNOT gate, Alice's electron and Bob's electron form a $\mathbb{C}^4$ space as there are two qubits and their state is $|\Psi\rangle_A \otimes |0\rangle_B = (\alpha|0\rangle_A + \beta|1\rangle_A) \otimes |0\rangle_B = \alpha|0\rangle_A|0\rangle_B + \beta|1\rangle_A|0\rangle_B$. If I keep Alice's qubit as the MSB and Bob's qubit as the LSB, without ambiguity, I can also just write it as $\alpha|00\rangle + \beta|10\rangle$. We are using Alice's electron as the control qubit (Fig. 19.1). Based on the definition of a CNOT gate, we expect the state will become $\alpha|00\rangle + \beta|11\rangle$ because the control qubit is 1 in the $|10\rangle$ term and thus it flips the LSB to 1 and the basis state becomes $|11\rangle$. This is an entangled state because whenever Alice's measurement is 0 (or 1), Bob's is also 0 (or 1). We can also show the entanglement process using matrix multiplication.

Example 19.1 Use matrix multiplication to show that the CNOT creates entanglement between Alice's and Bob's qubits.

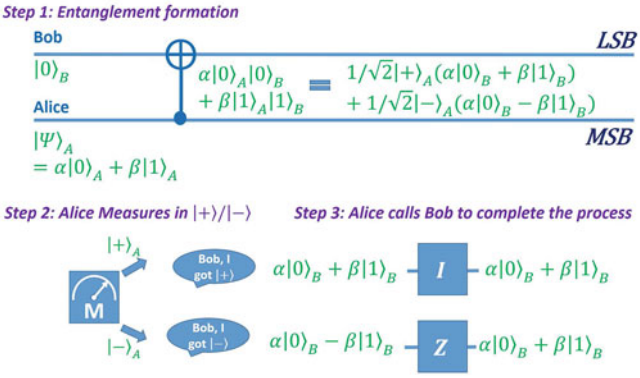


Fig. 19.1 Quantum teleportation algorithm for transferring a quantum state from Alice's lab to Bob's lab

The initial state is $\alpha\,|00\rangle+\beta\,|10\rangle$ and it is $\begin{pmatrix}\alpha\\0\\\beta\\0\end{pmatrix}$ in column representation because it has α component of $|00\rangle$ and β component of $|10\rangle$. Therefore,

$$\boldsymbol{U}_{XOR}(\alpha\,|00\rangle+\beta\,|10\rangle)=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\begin{pmatrix}\alpha\\0\\\beta\\0\end{pmatrix}$$

$$=\begin{pmatrix}\alpha\\0\\0\\\beta\end{pmatrix}=\alpha\,|00\rangle+\beta\,|11\rangle \tag{19.6}$$

You can see that to achieve this entanglement operation, we need to have a CNOT gate that is able to span a very large distance between Alice's lab (maybe in San José, California) and Bob's lab (maybe in San José, Costa Rica). It is almost impossible to build a quantum gate with high fidelity in this way. We will solve this problem later.

Step 2: Alice Measures Her Qubits in the $|+\rangle\,/\,|-\rangle$ Basis
After Alice has entangled her qubit with Bob's, Alice will try to measure her qubit in the $|+\rangle\,/\,|-\rangle$ basis. In order to calculate the result, we need to perform a suitable substitution. Since $|0\rangle=\frac{1}{\sqrt{2}}(|+\rangle+|-\rangle)$ and $|1\rangle=\frac{1}{\sqrt{2}}(|+\rangle-|-\rangle)$, we have

$$\begin{aligned}\alpha\,|00\rangle+\beta\,|11\rangle&=\alpha\,|0\rangle_A\,|0\rangle_B+\beta\,|1\rangle_A\,|1\rangle_B\\&=\alpha(\frac{1}{\sqrt{2}}(|+\rangle_A+|-\rangle_A))\,|0\rangle_B+\beta(\frac{1}{\sqrt{2}}(|+\rangle_A-|-\rangle_A))\,|1\rangle_B\\&=|+\rangle_A\,\frac{1}{\sqrt{2}}(\alpha\,|0\rangle_B+\beta\,|1\rangle_B)+|-\rangle_A\,\frac{1}{\sqrt{2}}(\alpha\,|0\rangle_B-\beta\,|1\rangle_B)\end{aligned} \tag{19.7}$$

Here we just collect the terms so that it is partitioned into two parts and each of them contains an orthogonal state in the $|+\rangle\,/\,|-\rangle$ basis for Alice's qubit. By doing so, we see that Bob's qubit starts getting some information related to Alice's qubit's original state, i.e. it has α and β as the coefficients. Now if Alice performs a measurement on her qubit in the $|+\rangle\,/\,|-\rangle$ basis, she will obtain either $|+\rangle_A$ or $|-\rangle_A$ and Bob's qubit will become either $\alpha\,|0\rangle_B+\beta\,|1\rangle_B$ or $\alpha\,|0\rangle_B-\beta\,|1\rangle_B$. This is a result of the collapse of the wavefunction and it is instantaneous. *Why the factor $\frac{1}{\sqrt{2}}$ disappears after measurement?* This is because once the wavefunction is collapsed, we need to renormalize the vector. When we set $|\Psi\rangle_A=\alpha\,|0\rangle_A+\beta\,|1\rangle_A$, we assume the vector is normalized already and therefore, $|\alpha|^2+|\beta|^2=1$. Therefore, after

renormalization, the $\frac{1}{\sqrt{2}}$ is gone. And if Alice obtained $|+\rangle$, her state is teleported to Bob faster than the speed of light. However, there is no transfer of useful information yet because Bob does not know Alice has performed the measurement and also does not know if his qubit has a state of $\frac{1}{\sqrt{2}}(\alpha\,|0\rangle_B + \beta\,|1\rangle_B)$ or $\frac{1}{\sqrt{2}}(\alpha\,|0\rangle_B - \beta\,|1\rangle_B)$ even Alice has measured.

Step 3: Classical Communication to Complete the Teleportation
Alice will then call Bob and let him know her outcome through a classical channel, which is slower than or the same as the speed of the light. If Alice obtains $|+\rangle$, Bob knows that his electron is in the state of $|\Psi\rangle$, and the quantum teleportation process is completed. Or equivalently, he will apply an identity gate to his qubit.

However, if Alice obtains $|-\rangle$, Bob knows his electron is in the state of $\frac{1}{\sqrt{2}}(\alpha\,|0\rangle_B - \beta\,|1\rangle_B)$, which is *not* $|\Psi\rangle$. He will apply a Z-gate ($\boldsymbol{U}_Z$, Eq. (16.10)), which is a phase shift gate with angle being π to convert it to $|\Psi\rangle$. A Z-gate negates the coefficient of the basis vector $|1\rangle$ but not $|0\rangle$. Therefore, $\alpha\,|0\rangle_B - \beta\,|1\rangle_B$ becomes $\alpha\,|0\rangle_B + \beta\,|1\rangle_B$, which is $|\Psi\rangle$. Now, the quantum teleportation process is also completed.

Of course, Alice's qubit's original state has also been destroyed due to the measurement.

19.3.2 Measurement in the $|+\rangle\,/\,|-\rangle$ Basis

In the quantum teleportation algorithm, Alice needs to measure her qubit in the $|+\rangle\,/\,|-\rangle$ basis in Step 2. If the apparatus can only measure in the $|0\rangle\,/\,|1\rangle$ basis, how do we achieve this? This is important because, in many quantum computers, we are only allowed to program to measure in the $|0\rangle\,/\,|1\rangle$ basis throughout the circuit. The trick is to apply the Hadamard gate before the measurement. Figure 19.2 shows the simplified version of quantum teleportation built in IBM-Q. In Step 2, instead of just performing a measurement in the $|+\rangle\,/\,|-\rangle$ basis, a Hadamard gate is applied followed by a measurement in the $|0\rangle\,/\,|1\rangle$ basis.

Example 19.2 Prove that for an arbitrary wavefunction $|\phi\rangle = a\,|0\rangle + b\,|1\rangle$, the probability distribution in the $|+\rangle\,/\,|-\rangle$ basis measurement is equivalent to that in the $|0\rangle\,/\,|1\rangle$ basis measurement after it is transformed by a Hadamard gate.

We first represent $|\phi\rangle$ in the $|+\rangle\,/\,|-\rangle$ basis to find out the probabilities of getting $|+\rangle$ and $|-\rangle$. Since, $|0\rangle = \frac{1}{\sqrt{2}}(|+\rangle + |-\rangle)$ and $|1\rangle = \frac{1}{\sqrt{2}}(|+\rangle - |-\rangle)$, we have

$$\begin{aligned} |\phi\rangle &= a\,|0\rangle + b\,|1\rangle \\ &= a\left(\frac{1}{\sqrt{2}}(|+\rangle + |-\rangle)\right) + b\left(\frac{1}{\sqrt{2}}(|+\rangle - |-\rangle)\right) \\ &= \frac{1}{\sqrt{2}}((a+b)\,|+\rangle + (a-b)\,|-\rangle) \end{aligned} \tag{19.8}$$

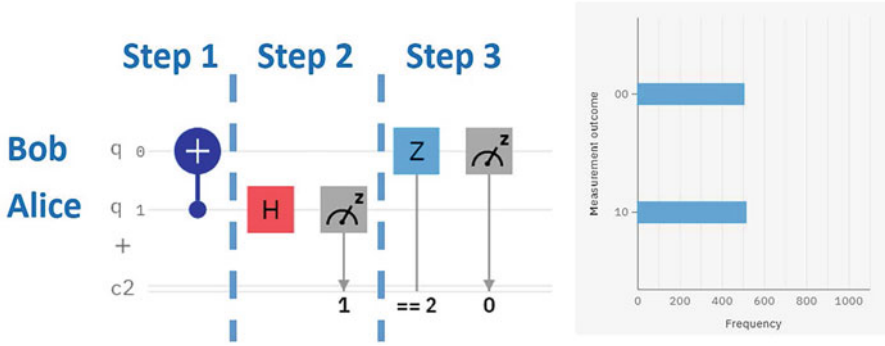


Fig. 19.2 Quantum teleportation circuit corresponds to Fig. 19.1 with $|\Psi\rangle = |0\rangle$

If we perform a measurement on the $|+\rangle / |-\rangle$ basis, the probability of getting $|+\rangle$ is $|a+b|^2/2$ and that for $|-\rangle$ is $|a-b|^2/2$.

What if we apply a Hadamard gate to $|\phi\rangle$ and then perform the measurement in the $|0\rangle / |1\rangle$ basis? Eq. (17.4) shows that $\boldsymbol{H}|\phi\rangle = \boldsymbol{H}(a|0\rangle + b|1\rangle) = a|+\rangle + b|-\rangle$ (i.e. a Hadamard gate transforms $|0\rangle$ and $|1\rangle$ to $|+\rangle$ and $|-\rangle$, respectively). Since, $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, we have

$$\begin{aligned} \boldsymbol{H}|\phi\rangle &= a|+\rangle + b|-\rangle \\ &= a\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right) + b\left(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right) \\ &= \frac{1}{\sqrt{2}}((a+b)|0\rangle + (a-b)|1\rangle) \end{aligned} \tag{19.9}$$

If we perform a measurement on the $|0\rangle / |1\rangle$ basis, the probability of getting $|0\rangle$ is $|a+b|^2/2$ and that for $|1\rangle$ is $|a-b|^2/2$. This is the same as the distribution of performing a $|\phi\rangle$ measurement on the $|+\rangle / |-\rangle$ basis.

Above we only show that the measurement probability distribution is the same. How about its effect on the state of the system? They are still equivalent. We can understand this by considering the basis states. Since this is a 2-qubit system, the basis states are either in the form of $|1X\rangle$ or $|0X\rangle$, where X can be 0 or 1. Since $|1\rangle = \frac{1}{\sqrt{2}}(|+\rangle - |-\rangle)$, if it is $|1X\rangle$, we can replace the first qubit (Alice' qubit) accordingly and get $\frac{1}{\sqrt{2}}(|+\rangle - |-\rangle) \otimes |X\rangle$. If we perform a measurement using the $|+\rangle / |-\rangle$ basis, it will collapse to either $|+\rangle|X\rangle$ or $-|-\rangle|X\rangle$. From Bob's perspective, his qubit becomes $|X\rangle$ or $-|X\rangle$, respectively, after Alice's measurement.

On the other hand, if we apply a Hadamard gate to Alice's qubit first, $|1X\rangle$ becomes $|+\rangle|X\rangle$, which is $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \otimes |X\rangle$. When Alice performs a measurement using the $|0\rangle / |1\rangle$ basis, it will collapse to either $|0\rangle|X\rangle$ or $-|1\rangle|X\rangle$. From Bob's

perspective, his qubit again becomes $|X\rangle$ or $-|X\rangle$, respectively. Therefore, the effect on the LSB is the same. So, performing a $|+\rangle / |-\rangle$ measurement is equivalent to performing a $|0\rangle / |1\rangle$ measurement after the application of the Hadamard gate.

Similarly, this is the same for the $|0X\rangle$ state. This is also true when there is superposition or more qubits are involved.

19.3.3 Run on IBM-Q

Figure 19.2 shows the implementation of the circuit. $q0$ is the LSB and represents Bob's qubit while $q1$ is the MSB representing Alice's qubit. Since both $q0$ and $q1$ are initialized to $|0\rangle$, this circuit represents the case that Alice's qubit is $|\Psi\rangle = |0\rangle$. After it is teleported to Bob, Bob's should be $|0\rangle$.

Step 1 performs the entanglement and Step 2 effectively performs a measurement on Alice's qubit in the $|+\rangle / |-\rangle$ basis. The measurement result is stored in the MSB in the classical register because "1" is chosen in the measurement in Step 2. In Step 3, the Z-gate is only performed if the previous measurement equals 2. That means if the previous measurement is $(10)_2$ in the binary form, the Z-gate will be applied to Bob's qubit. Note that the LSB has not been measured yet and thus is 0 by default. Therefore, a measurement of 1 on Alice's qubit in the second step is equivalent to having the classical register storing a value of $(10)_2$, i.e. 2. As discussed, the measurement of $|1\rangle$ after the Hadamard gate is equivalent to the measurement of $|-\rangle$. Therefore, the Z-gate is applied only when Alice gets a $|-\rangle$ when measuring her qubit in the $|+\rangle / |-\rangle$ basis. Finally, in Step 3, the LSB is measured just to check the results. As the simulation showed, regardless of the MSB value, the LSB is always 0. Therefore, Bob has acquired the state of $|\Psi\rangle = |0\rangle$. This is a trivial case. In the Problems, we will try a more complex one.

19.4 Summary

In this chapter, we learned our first quantum algorithm, namely, the *quantum teleportation*. We also learned that we cannot clone an arbitrary unknown state due to the *no-cloning theorem*. They are related in the sense that if the state of a qubit is quantum teleported to another qubit, the original state will be destroyed so that there is no cloning. We also learned that quantum teleportation indeed has certain aspects similar to hypothetical teleportation. However, it cannot propagate useful information faster than the speed of the light because it involves classical communications to finalize the teleportation. We also learned an important trick to measure a state in the $|+\rangle / |-\rangle$ basis when only $|0\rangle / |1\rangle$ basis measurement is available. That is to apply a Hadamard gate before the measurement. The simplified version of quantum teleportation is not complete because it is almost impossible to

create an XOR (CNOT) gate over a larger distance in Step 1 of the algorithm. We will solve this problem in the next chapter.

Problems

19.1 No-Cloning Theorem

We may also prove the No-Cloning Theorem using another method (i.e. by contradiction). We know $\boldsymbol{L}(|a\rangle_A |0\rangle_B) = |a\rangle_A |a\rangle_B$, $\boldsymbol{L}(|b\rangle_A |0\rangle_B) = |b\rangle_A |b\rangle_B$ and also $\boldsymbol{L}(\frac{1}{\sqrt{2}}(|a\rangle_A + |b\rangle_A)|0\rangle_B) = \frac{1}{\sqrt{2}}(|a\rangle_A + |b\rangle_A)\frac{1}{\sqrt{2}}(|a\rangle_B + |b\rangle_B)$ by definition, where $|a\rangle$ and $|b\rangle$ are arbitrary normalized states. Expand the last equation and evaluate $\boldsymbol{L}(|a\rangle_A |0\rangle_B) + |b\rangle_A |0\rangle_B)$ to show that there is contradiction (that $|a\rangle$ and $|b\rangle$ cannot be generalized states) and thus cloning of a general state is impossible.

19.2 Quantum Teleportation of a Superposition State

(a) Figure 19.3 shows the simplified version of quantum teleportation. The state to be teleported is $|+\rangle$. Explain why this is the state to be teleported?
(b) Construct the matrix (4×4) from the left to Alice's first measurement.
(c) Explain the simulation results and why quantum teleportation is successful.
(d) Note that the MSB contains not just 0 but also 1 in the final output. Why the two Hadamard gates do not cancel each other? Hints: Look at the matrix constructed.

19.3 No-Cloning Theorem and Heisenberg Uncertainty Principle

If the state is a basis state, it is possible to clone. For any arbitrary state, it must be a basis state of another basis. Can I clone it? How is it related to the Heisenberg Uncertainty Principle? Please discuss.

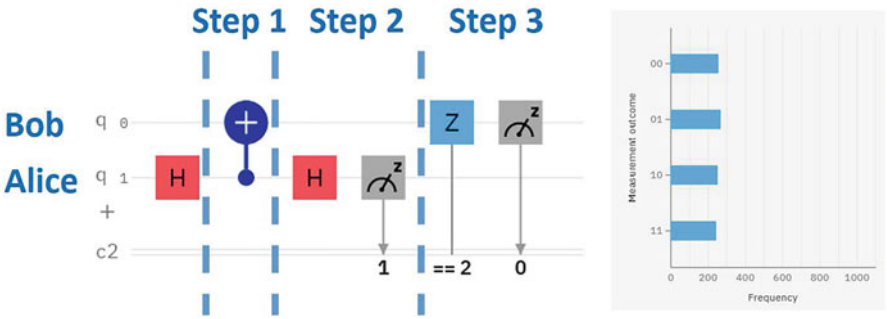


Fig. 19.3 Quantum teleportation circuit corresponds to Fig. 19.1 with $|\Psi\rangle = |+\rangle$

Chapter 20
Quantum Teleportation II and Entanglement Swapping

20.1 Learning Outcomes

Describe the role of the ancillary bit in the complete version of quantum teleportation; Construct the quantum teleportation circuit on IBM-Q; Apply quantum teleportation circuit to achieve entanglement swapping; Understand deeper how to use the classical register in a conditional quantum gate.

20.2 Quantum Teleportation: The Full Version

In the full version of quantum teleportation, we will solve the difficulty in creating an ideal CNOT gate that needs to operate on two qubits separated by a huge distance. We will involve one more qubit (an **ancillary** qubit) and also need to physically transport that qubit from Bob's lab to Alice's lab before performing any quantum teleportation. This means that we cannot quantum teleport a quantum state to an arbitrary lab. We must prepare for teleportation ahead by transporting an ancillary qubit to the target lab.

An *ancillary* qubit appears often in quantum computing algorithms. It is an "extra" qubit but is critical for the circuit to achieve the goal. Therefore, although it is extra, it is not redundant. For example, in the quantum teleportation circuit, Alice's qubit is the source and Bob's qubit is the target. Therefore, these are the only two qubits we need for quantum teleportation. However, if we want to obviate the use of the ideal CNOT gate, we need to add an extra ancillary bit.

In the complete algorithm, we will replace Step 1 with Step 1a and Step 1b. The purpose is to create the entanglement in Step 1 of the simplified version.

Step 1a: Qubit Entanglement and Ancillary Qubit Transportation

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_20

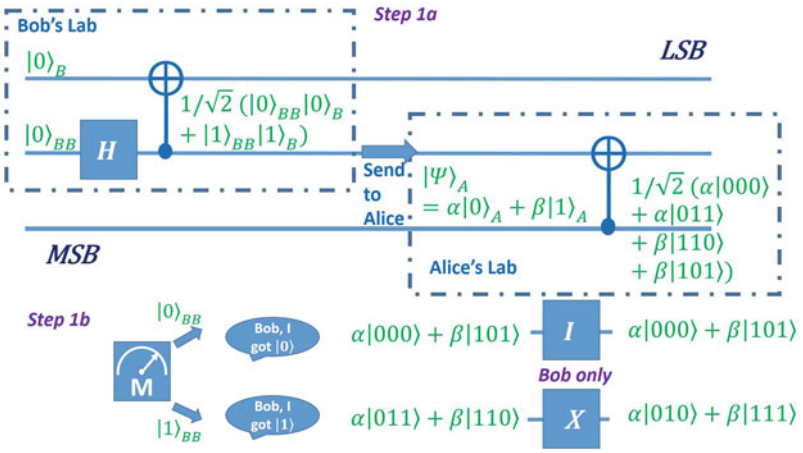


Fig. 20.1 Algorithm to replace Step 1 in the simplified quantum teleportation circuit in Fig. 19.1. After this, Steps 2 and 3 in Fig. 19.1 need to be executed to complete the quantum teleportation process

Figure 20.1 shows the algorithm. Bob will entangle the qubit, $|0\rangle_B$, he wants to write ahead with another ancillary qubit initialized as $|0\rangle_{BB}$. Again, we need to make sure which one is the LSB. In this case, the qubit he wants to write is the LSB. Therefore, the state of Bob's qubits is $|0\rangle_{BB} \otimes |0\rangle_B$. He then applies $\boldsymbol{H} \otimes \boldsymbol{I}$ to it. We know that $\boldsymbol{H}$, the Hadamard gate, will create a superposition of the basis states. Therefore, it becomes $\frac{1}{\sqrt{2}}(|0\rangle_{BB} + |1\rangle_{BB}) \otimes |0\rangle_B$. Now let me drop the subscript to make it more succinct as we know that $|0\rangle_B$ is the LSB already. So it can be written as $\frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)$. Then it goes through the CNOT gate. Note that both qubits are in close proximity and they are both in Bob's lab. So there is no problem in creating a CNOT gate with high fidelity. Again, CNOT will change the basis vector if the control qubit is 1 by flipping the value of the target qubit. Therefore, $|10\rangle$ becomes $|11\rangle$ and the state of Bob's qubits is now $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, which is entangled. We have learned this state before and it is one of the famous *Bell states*, $\left|\Phi^+\right\rangle$ in Eq. (13.13).

Now Bob will send the BB qubit to Alice and it will stay there and still entangled with the B qubit in Bob's lab until they want to do quantum teleportation.

One day, Alice decides to quantum teleport her qubit information to Bob. That qubit has a state $|\Psi\rangle_A = \alpha\,|0\rangle_A + \beta\,|1\rangle_A$ like in the simplified version. She will then perform a CNOT operation, by using her qubit A as the control qubit, on qubit BB which was sent by Bob earlier and is still entangled with Bob's qubit B. Let us see how the whole system will evolve when the CNOT gate is applied.

Example 20.1 Describe how the state of the 3-qubit system evolves after Alice decided to quantum teleport the information to Bob.

Before the CNOT gate by Alice, the state of the 3-qubit system is

$$
\begin{aligned}
|\Psi\rangle_A \otimes \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) &= (\alpha\,|0\rangle_A + \beta\,|1\rangle_A) \otimes \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\
&= (\alpha\,|0\rangle + \beta\,|1\rangle)\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\
&= \frac{1}{\sqrt{2}}(\alpha\,|000\rangle + \beta\,|100\rangle + \alpha\,|011\rangle + \beta\,|111\rangle) \qquad (20.1)
\end{aligned}
$$

Here, Alice's original qubit (A) to be teleported is the MSB followed by the qubit sent by Bob (BB) and Bob's qubit that still resides in Bob's lab is the LSB. I gradually removed the subscripts and the $\otimes$ as we are familiar with the meaning of tensor product already. After that, the CNOT gate is applied. The control qubit is the MSB (A) and the second qubit (BB) is the target qubit. When passing through the CNOT gate, whenever the basis MSB is 1, the target qubit will be flipped. Therefore, $|100\rangle$ becomes $|110\rangle$ and $|111\rangle$ becomes $|101\rangle$. As a result, the state of the 3-qubit system becomes,

$$
\begin{aligned}
&\frac{1}{\sqrt{2}}(\alpha\,|000\rangle + \beta\,|110\rangle + \alpha\,|011\rangle + \beta\,|101\rangle) \\
&\quad = \frac{1}{\sqrt{2}}(\alpha\,|000\rangle + \beta\,|101\rangle + \alpha\,|011\rangle + \beta\,|110\rangle) \qquad (20.2)
\end{aligned}
$$

Step 1b: Measurement of the Ancillary Qubit
Now Alice will perform a measurement on the second qubit (BB) on the $|0\rangle$ / $|1\rangle$ basis. If she obtains $|0\rangle$, it means the whole system has collapsed to a state containing only $|000\rangle$ and $|101\rangle$ in Eq. (20.2), which is $\alpha\,|000\rangle + \beta\,|101\rangle$ (after normalization). If we only consider the MSB (i.e. A in Alice's lab) and LSB (i.e. B in Bob's lab), they are entangled! They are $\alpha\,|0\rangle_A\,|0\rangle_B + \beta\,|1\rangle_A\,|1\rangle_B$. This is the output of *Step 1* in the simplified version in Fig. 19.1. Therefore, Alice will just call Bob telling him that she got $|0\rangle$ and their qubits have been entangled and they can continue to perform Step 2 and Step 3 in Fig. 19.1 to complete the quantum teleportation.

If Alice, on the other hand, obtains $|1\rangle$, it means the whole system has collapsed to a state containing only $|011\rangle$ and $|110\rangle$ in Eq. (20.2), which is $\alpha\,|011\rangle + \beta\,|110\rangle$ (after normalization). If only consider the MSB and LSB, they are $\alpha\,|01\rangle + \beta\,|10\rangle$. They are still entangled because when the MSB is 1 (or 0), the LSB is always 0 (or 1). But this is not exactly what we want. However, Alice can just call Bob and tell him that she got $|1\rangle$. Bob just needs to apply a NOT gate to his qubit to flip the bit value in the basis states. Therefore, the state becomes $\alpha\,|00\rangle + \beta\,|11\rangle$, which again is the output of *Step 1* in the simplified version in Fig. 19.1. They will then continue to perform Step 2 and Step 3 in Fig. 19.1 to complete the quantum teleportation.

I would like to emphasize *why we are allowed to only consider the MSB and the LSB* in the last part of the reasoning (i.e. why we can ignore the BB qubit in the final discussion). This is because the second qubit is fully disentangled with the MSB and the LSB after the measurement. Therefore, we can factorize it and thus *ignore* it. For example, after Alice obtains $|0\rangle$ in the measurement, we obtained $\alpha\,|011\rangle + \beta\,|110\rangle$. This can be rewritten as $\alpha\,|0\rangle_A\,|1\rangle_{BB}\,|1\rangle_B + \beta\,|1\rangle_A\,|1\rangle_{BB}\,|0\rangle_B$. We can always rearrange the electrons if we want because counting them in a different order does not change Physics. We can write it as $\alpha\,|0\rangle_A\,|1\rangle_B\,|1\rangle_{BB} + \beta\,|1\rangle_A\,|0\rangle_B\,|1\rangle_{BB} = (\alpha\,|0\rangle_A\,|1\rangle_B + \beta\,|1\rangle_A\,|0\rangle_B)\,|1\rangle_{BB}$. There is nothing wrong to do so. We just need to make sure not to shuffle them in the *equations*. Now the ancillary bit is factored out, and it is just an electron sitting somewhere and has no effect on Alice's and Bob's electrons. We do not need to consider it and thus it is the ancillary qubit that is critical to achieving our goal but is useless in the end.

20.2.1 Run on IBM-Q

Figure 20.2 shows the quantum circuit built-on IBM-Q. It has four steps. *Step 1a* and *Step 1b* correspond to Fig. 20.2 to perform entanglement between Alice's qubit (A) and Bob's qubit (B) with the help of the ancillary qubit (BB) to obviate the need of an ideal CNOT gate. *Step 2* and *Step 3* implement the *Step 2* and *Step 3* in Fig. 19.1. It is worth mentioning that the ancillary qubit goes through only identity gates and has no contribution to the algorithm anymore after *Step 1b*.

This circuit has three qubits, namely $q2$ for Alice's source qubit, $q1$ for the ancillary qubit, and $q0$ for Bob's target qubit. $q2$ is the MSB and $q0$ is the LSB. There are also, by default, three classical bits ($c3$). Therefore, upon each measurement, the $c3$ (the 3-bit classical register to store the measurement output) may bear any values between $(000)_2$ to $(111)_2$. Note that little-endian convention is being used. So counting from the right, the bits are numbered as the 0th, 1st, and 2nd bit.

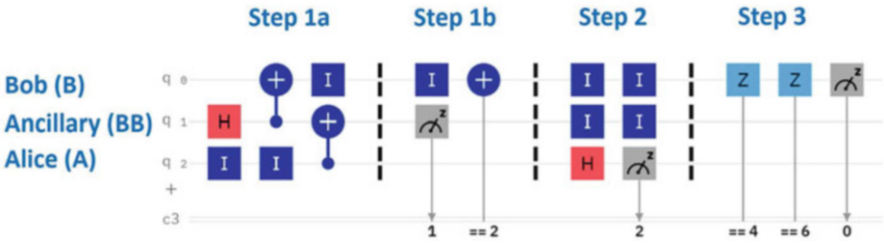


Fig. 20.2 Implementation of the full quantum teleportation algorithm by combining Fig. 20.1 and Steps 2 and 3 in Fig. 19.1

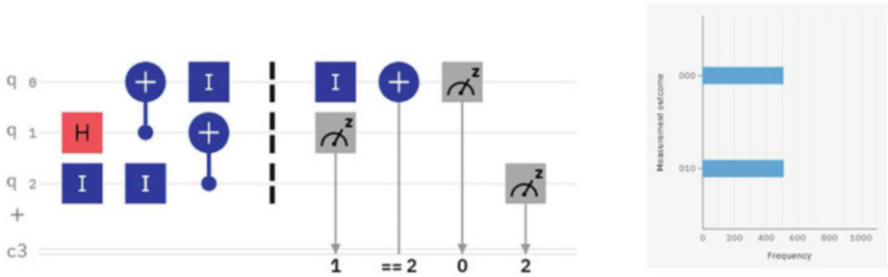


Fig. 20.3 Measurement results of Step 1a and Step 1b in Fig. 20.2

Identity gates, $\boldsymbol{I}$ are used to fill the space for better alignment. They are not necessary and can be replaced by wires.

In *Step 1b*, Alice measures $q1$ using the $|0\rangle / |1\rangle$ basis. As the setup shows, it is stored in 1st classical bit. Therefore, if she measures $|1\rangle$, $c3$ becomes $(010)_2$. If she measures $|0\rangle$, $c3$ is still $(000)_2$. Then Bob will apply a NOT-gate if the value in $c3$ is 2 (in decimal), which is $(010)_2$. This is to simulate *Step 1b* in Fig. 20.1, where Bob will apply a NOT gate to its qubit if Alice measures $|1\rangle$ and doing nothing otherwise.

For complex circuits, we often want to gauge the intermediate results for debugging or to enhance our understanding. To do so, I keep only *Step 1a* and *Step 1b* and measure all qubit bits to check the probability distribution in Fig. 20.3. Since $q3$ is $|0\rangle$ by default, this circuit is performing a trivial teleportation of $|\Psi\rangle_A = |0\rangle$. Therefore, after *Step 1b*, it is expected to be either $|000\rangle$ (if Alice measures $|0\rangle$) or $|010\rangle$ (if Alice measures $|1\rangle$) as the β in Fig. 20.1 equals 0. Figure 20.3 shows the circuit and the measurement results. And indeed it only has $|000\rangle$ and $|010\rangle$ components.

In *Step 2*, Alice performs a measurement on the source qubit (A) using the $|+\rangle / |-\rangle$ basis. We discussed this in the previous chapter already. In summary, the Hadamard gate is used so that we can effectively perform a $|+\rangle / |-\rangle$ basis measurement while we are actually performing a $|0\rangle / |1\rangle$ basis measurement. The measurement result is stored in the 2nd bit in the classical register, i.e. the leftmost/MSB bit. Note again the rightmost is the LSB or the 0th bit. If she measures $|1\rangle$ (corresponds to $|+\rangle$ when there is no Hadamard gate), $c3$ becomes $(100)_2$ or $(110)_2$ depending on the value she obtained previously in the BB measurement. If she measures $|0\rangle$ (corresponds to $|-\rangle$), $c3$ becomes $(000)_2$ or $(010)_2$. Bob will apply Z-gate if Alice measures $|1\rangle$ in *Step 2*. This means $c3$ can be either $(100)_2$ or $(110)_2$, which are 4 or 6 in decimal, respectively. Therefore, in *Step 3*, two conditional Z-gates are used and they are only applied when either $c3 = 4$ or $c3 = 6$. Note that for every single experiment, $c3$ cannot be 4 and 6 at the same time. Therefore, the Z-gate will never be applied twice.

Figure 20.4 shows the simulation result. Again, by default, the qubits are initialized as $|0\rangle$. Therefore, the default circuit represents the situation that Alice

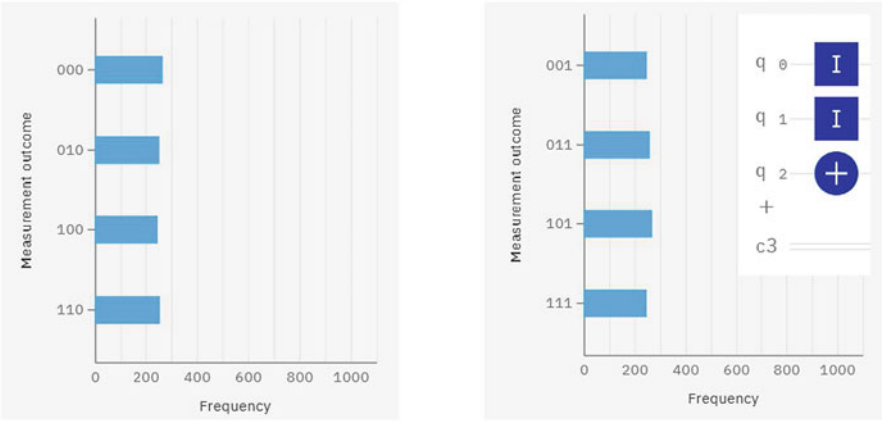


Fig. 20.4 Simulation results of the circuit in Fig. 20.2 with $|\Psi\rangle = |0\rangle$ (left) and $|\Psi\rangle = |1\rangle$ (right). The circuit to create $|\Psi\rangle = |1\rangle$ is shown as the inset

tries to teleport $|\Psi\rangle = |0\rangle$ to Bob. The measurement shows that the LSB is always 0. Therefore, Bob's qubit is $|0\rangle$ as expected due to quantum teleportation.

We may also test with the case that Alice teleports $|\Psi\rangle = |1\rangle$ to Bob. We achieve this by applying a NOT-gate to Alice's qubit before quantum teleportation. In this case, we expect the LSB in the measurement to be always 1. Indeed, the measurement shows that Bob's qubit has acquired $|1\rangle$ because all the outcomes have 1 as the LSB.

20.3 Entanglement Swapping

Let us look at an application of quantum teleportation, namely the **Entanglement Swapping**. I spent a lot of time discussing my opinion on the meaning of quantum teleportation. I even said that quantum teleportation does have the meaning of hypothetical teleportation on the microscopic scale because microscopic particles are indistinguishable. Therefore, teleporting the state from a particle at a place to another identical particle at another place achieves a similar purpose of teleporting that particle between those two places. I also claim that if the states of the particles in my body are teleported to some particles on Mars, I believe I would have been effectively teleported to Mars, where I will continue to live from the last state I was on Earth.

If that is true, is my wife still my wife after I am teleported to Mars? Do I still love her or does she still love me as before teleportation? I believe so as long as quantum teleportation is successful and faithful, although she will start complaining I do not take her on this trip to Mars.

Assume Alice's electron was entangled with another electron, owned by Carlos (our new character in this play). If Alice teleports the state of her electron to Bob,

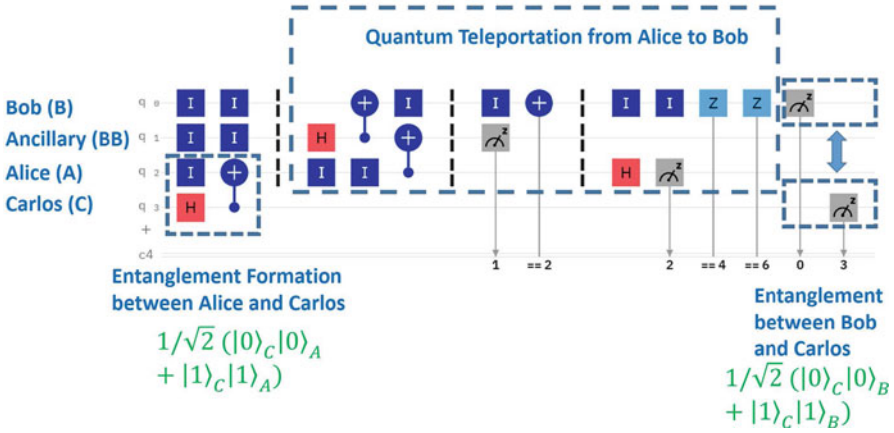


Fig. 20.5 Entanglement swapping circuit built in IBM-Q

I will expect that Bob's electron will then entangle with Carlos' electron. This is a natural consequence if we agree quantum teleportation of a state in the microscopic world is very similar to hypothetic teleportation. Or, if you agree that my wife would still love the "me" on Mars formed by quantum teleportation.

In quantum teleportation, there are 3-qubits. Now we also have Carlos' electron and this becomes a 4-qubit system. Figure 20.5 shows the corresponding entanglement swapping circuit. Note that Carlo's qubit (C) is the MSB. While this is a big circuit, it is not difficult to understand. Most of the part belongs to the quantum teleportation circuit and is the same as Fig. 20.2. Its sole function is to teleport Alice's qubit state to Bob's qubit. To entangle Carlos' and Alice's qubits, a Hadamard gate followed by an XOR (CNOT) gate is used. We discussed this earlier and let us take this opportunity to do it using matrix multiplication.

Example 20.2 Use matrix multiplication to show that Alice's and Carlos' are entangled right before the quantum teleportation block in Fig. 20.5.

The input state to the circuit is $|0\rangle_C \otimes |0\rangle_A \otimes |0\rangle_{BB} \otimes |0\rangle_B$. The part of the circuit performing the operation is $\boldsymbol{H}_C \otimes \boldsymbol{I}_A \otimes \boldsymbol{I}_{BB} \otimes \boldsymbol{I}_B$ followed by $\boldsymbol{U}_{XOR,C,A} \otimes \boldsymbol{I}_{BB} \otimes \boldsymbol{I}_B$. We are very familiar with the notations already and we can rewrite the input state as $|0000\rangle$ and the operators as $(\boldsymbol{U}_{XOR} \otimes \boldsymbol{I}^{\otimes 2})(\boldsymbol{H} \otimes \boldsymbol{I}^{\otimes 3})$. I hope you understand why (again vector evolves from right to left in the equation but from left to right in the circuit diagram). This is a 4-qubit system and the matrix will be 16×16, which is too clumsy to write down. But we see that only identity gates are applied to the last two qubits (the ancillary bit BB and Bob's B), therefore,

$$\begin{aligned}
&(\boldsymbol{U}_{XOR} \otimes \boldsymbol{I}^{\otimes 2})(\boldsymbol{H} \otimes \boldsymbol{I}^{\otimes 3})\,|0000\rangle \\
&\quad = ((\boldsymbol{U}_{XOR})(\boldsymbol{H} \otimes \boldsymbol{I})\,|00\rangle) \otimes (\boldsymbol{I}^{\otimes 2}\boldsymbol{I}^{\otimes 2}\,|00\rangle) \\
&\quad = ((\boldsymbol{U}_{XOR})(\boldsymbol{H} \otimes \boldsymbol{I})\,|00\rangle) \otimes (|00\rangle)
\end{aligned} \tag{20.3}$$

So we only need to consider the leftmost 2 qubits, i.e. the qubits that belong to Carlos and Alice.

$$
\begin{aligned}
(\boldsymbol{U}_{XOR})(\boldsymbol{H}\otimes\boldsymbol{I})\,|00\rangle &= \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1&1\\1&-1\end{pmatrix}\otimes\begin{pmatrix}1&0\\0&1\end{pmatrix}\begin{pmatrix}1\\0\\0\\0\end{pmatrix}\\
&= \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1\begin{pmatrix}1&0\\0&1\end{pmatrix}&1\begin{pmatrix}1&0\\0&1\end{pmatrix}\\1\begin{pmatrix}1&0\\0&1\end{pmatrix}&-1\begin{pmatrix}1&0\\0&1\end{pmatrix}\end{pmatrix}\begin{pmatrix}1\\0\\0\\0\end{pmatrix}\\
&= \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1&0&1&0\\0&1&0&1\\1&0&-1&0\\0&1&0&-1\end{pmatrix}\begin{pmatrix}1\\0\\0\\0\end{pmatrix}\\
&= \frac{1}{\sqrt{2}}\begin{pmatrix}1&0&1&0\\0&1&0&1\\0&1&0&-1\\1&0&-1&0\end{pmatrix}\begin{pmatrix}1\\0\\0\\0\end{pmatrix}\\
&= \frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\0\\1\end{pmatrix}\\
&= \frac{1}{\sqrt{2}}(|00\rangle+|11\rangle) \qquad (20.4)
\end{aligned}
$$

Therefore, Carlos' and Alice's qubits are entangled as whenever Carlos measures $|0\rangle$ (or $|1\rangle$), Alice's qubit will collapse to $|0\rangle$ (or $|1\rangle$).

Now including the BB and C qubits, the state is

$$\frac{1}{\sqrt{2}}(|0000\rangle+|1100\rangle) \qquad (20.5)$$

After quantum teleportation, Alice's qubit state will be teleported to Bob's qubit. We will not go through the matrix multiplications and measurements. But we can understand in this way. Since the quantum mechanics that we are using is linear, it obeys the superposition law. We can consider each basis state in Eq. (20.5) individually and then combine them. For the $|0000\rangle$ term, it is actually $|0\rangle_C\,|0\rangle_A\,|0\rangle_{BB}\,|0\rangle_B$. We can ignore the MSB $|0\rangle_C$ for now as it is not a part of the quantum teleportation circuit and the 3 right qubits will become $|X\rangle_A\,|Y\rangle_{BB}\,|0\rangle_B$ after quantum teleportation because the qubit state of A is copied to B while A and

BB can be any values depending on the measurement results (see Fig. 20.4 left). Here, X and Y can be either 0 or 1. Therefore, the final state will be $|0XY0\rangle$ for the whole system. For the $|1100\rangle$ term, similarly, we need to consider the rightmost 3 qubits only and they become $|W\rangle_A |V\rangle_{BB} |1\rangle_B$ after quantum teleportation because the qubit state of A is copied to B (see Fig. 20.4 right). The state of the 4-qubit becomes $|1WV1\rangle$, where W and V can be either 0 or 1. Therefore, after quantum teleportation, the state in Eq. (20.5) becomes a linear combination of $|0XY0\rangle$ and $|1WV1\rangle$ due to superposition. However, for every term, the LSB (Bob's qubit) and the MSB (Carlo's qubit) are the same. Therefore, Bob's qubit and Carlos' qubit are entangled now.

One implication of entanglement swapping is that we can entangle two qubits that are far apart from each other. For example, Alice's and Carlos' qubits may be both on the Earth. We can entangle them easily in the same lab. With entanglement swapping, we can then entangle Carlo's with Bob's (which might be on Mars) by teleporting Alice's qubit state to Bob.

Based on this, if marriage is based on entanglement, if I am teleported to Mars, my wife is still my wife.

20.3.1 Run on IBM-Q

Figure 20.6 shows the simulation results. We can see that all the states have their MSB equal to LSB and X, Y, W, and V can be either 0 or 1.

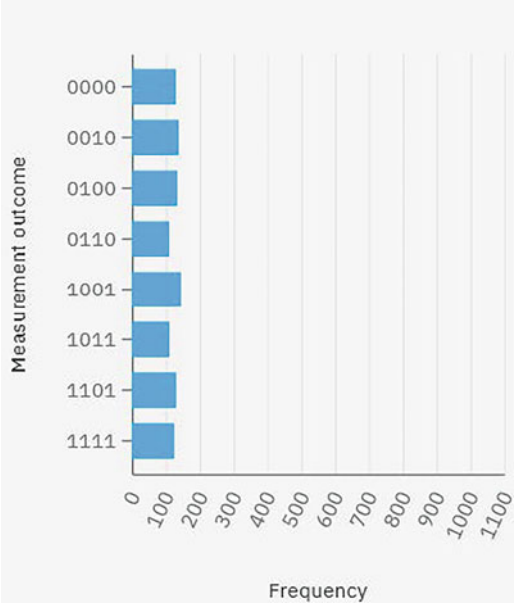


Fig. 20.6 Simulation result of the entanglement swapping circuit in Fig. 20.5

20.4 Summary

In this chapter, we completed quantum teleportation by introducing an ancillary qubit to obviate the need in creating a CNOT gate that needs to interact with two qubits spanning a large distance. We see that two classical communications and one physical transport of qubit are required to achieve quantum teleportation. Therefore, it does not violate the Laws of Physics. We also see that it further resembles some of the features of hypothetical teleportation. If we teleport Alice's qubit to Bob while Alice's qubit was entangled with Carlos' qubit, Bob's qubit will then entangle with Carlos' qubit after quantum teleportation.

Problems

20.1 Entanglement

Derive Eq. (20.2) using matrix.

20.2 Classical Register and Conditional Gate

(a) Reproduce the result in Figs. 20.2 and 20.4
(b) Store the Step 1b measurement result in the 0th classical qubit and store Step 3 measurement result in the 1st classical qubit. How should we modify the conditional CNOT and Z-gates in Step 1b and Step 3 so it will produce the same result? Hints: The classical bit we want to inspect now is the 1st qubit.

20.3 Quantum Teleportation vs. Hypothetical Teleportation

Go through the quantum teleportation one more time and answer the following questions: (1) Why the state of the original electron is destroyed? How is it related to the no-cloning theorem? (2) If I want to quantum teleport myself to Moon or Mars and back to the Earth, how should I prepare for that? (3) If Alice's state was $|0\rangle$, do we need quantum teleportation to transfer the state to Bob? Do we violate the no-cloning theorem?

Chapter 21
Deutsch Algorithm

21.1 Learning Outcomes

Understand the Deutsch–Jozsa and Deutsch problems; Appreciate how quantum parallelism is used to speed up the solution-finding process in the Deutsch algorithm; Understand while parallel computations are performed in certain quantum algorithms, the information we can extract is limited. Understand the origin of the quantum oracle; Able to derive the equations in the Deutsch algorithm and implement them in IBM-Q.

21.2 Deutsch Algorithm

The **Deutsch Algorithm** is a sub-problem of a more general **Deutsch–Jozsa Algorithm**. We will only discuss the Deutsch Algorithm and inspect the algorithm in detail. The derivation of the Deutsch–Jozsa Algorithm can be found elsewhere and can be understood easily if you understand the details I am going to explain in the Deutsch Algorithm. In the derivation, you will see that the Deutsch Algorithm provides a two-time speedup in the computation. In the more general Deutsch–Jozsa Algorithm, it provides an **exponential speedup** over the classical one. This means that for an n-qubit problem, classically we need $2^{n-1}+1$ computations but only 1 computation using the Deutsch–Jozsa Algorithm. Therefore, the Deutsch Algorithm also provides an exponential speedup, but it just happens that $n = 1$ and thus it only has two times of speedup.

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_21

21.2.1 The Problem

In the **Deutsch–Jozsa** problem, we are given an **oracle** (a black box into which we cannot peep), which maps $f : \{0, 1\}^n \mapsto \{0, 1\}$. This means that it takes an *n bi*nary digi*t*s (n-bits) as the input. For example, if $n = 3$, the inputs can be any permutation of 3 digits of 0 or 1 (i.e. from the set $\{0, 1\}^3$). For example, the input may be "000", "001", $\cdots$, "111". Its output is either 0 or 1 (i.e. one single-bit output). So an oracle is just a *function* and we do not know which function it implements. But we are told that it is either *balanced* or *constant* in the Deutsch–Jozsa problem (this is given). A balanced function means that half of its outputs are "0" and the other half are "1." A constant function means that, regardless of the input values, it always outputs the same value (a constant value). Of course, for some constant functions, the output is constantly "1" while for other constant functions, the output is constantly "0." *The **Deutsch–Jozsa Algorithm** is to figure out if the oracle is balanced or constant. The problem is completely classical.*

Example 21.1 Give an example of an oracle that is constant for the $n = 3$ case.

There are eight possible inputs in the $n = 3$ case. Let the input bits be x_2, x_1, and x_0. If an oracle implements the following function:

$$f(x_2x_1x_0) = x_2 \cdot x_1 \cdot x_0 \cdot 0 \tag{21.1}$$

where "$\cdot$" is the classical Boolean AND operation, then

$$\begin{aligned}
f(000) &= 0 \cdot 0 \cdot 0 \cdot 0 = 0 \\
f(001) &= 0 \cdot 0 \cdot 1 \cdot 0 = 0 \\
f(010) &= 0 \cdot 1 \cdot 0 \cdot 0 = 0 \\
f(011) &= 0 \cdot 1 \cdot 1 \cdot 0 = 0 \\
f(100) &= 1 \cdot 0 \cdot 0 \cdot 0 = 0 \\
f(101) &= 1 \cdot 0 \cdot 1 \cdot 0 = 0 \\
f(110) &= 1 \cdot 1 \cdot 0 \cdot 0 = 0 \\
f(111) &= 1 \cdot 1 \cdot 1 \cdot 0 = 0
\end{aligned}$$

And this is a *constant* oracle, which always gives "0" as the output.

Example 21.2 Give an example of an oracle that is balanced for the $n = 3$ case.

Again, there are eight possible inputs in the $n = 3$ case. Let the input bits be x_2, x_1, and x_0. If an oracle implements the following function:

$$f(x_2x_1x_0) = x_2 \oplus x_1 \oplus x_0 \tag{21.2}$$

where "$\oplus$" is the classical XOR operation, then

$$
\begin{aligned}
f(000) &= 0 \oplus 0 \oplus 0 = 0 \\
f(001) &= 0 \oplus 0 \oplus 1 = 1 \\
f(010) &= 0 \oplus 1 \oplus 0 = 1 \\
f(011) &= 0 \oplus 1 \oplus 1 = 0 \\
f(100) &= 1 \oplus 0 \oplus 0 = 1 \\
f(101) &= 1 \oplus 0 \oplus 1 = 0 \\
f(110) &= 1 \oplus 1 \oplus 0 = 0 \\
f(111) &= 1 \oplus 1 \oplus 1 = 1
\end{aligned}
$$

This is a *balanced* oracle. A half (4) of the inputs results in "0" and a half (4) of the inputs results in "1". How did I come up with this function? Eq. (21.2) is just counting whether the number of "1" in the input bits is odd or even. If it is even, it gives "0" and if it is odd, it gives "1." We have encountered something similar earlier when we tried to find the expression when an n-qubit Hadamard gate is applied to an arbitrary basis state $|x\rangle_n$ in Eq. (17.20).

Here I want to remind you again that we do NOT know what function the oracle implements. But we are told that it can be either balanced or constant. I show you the examples is just to let you know two of the possible functions.

Yes, I am supposed to only discuss the **Deutsch** problem instead of the **Deutsch–Jozsa** problem, but I want to give you a deeper understanding of the background so you can explore more in the future.

Now, let us turn to the **Deutsch** problem. It is a special case of the **Deutsch–Jozsa** problem with $n = 1$. Therefore, the oracle maps $f : \{0, 1\} \mapsto \{0, 1\}$. It only has 1 bit of input and thus it only has two possible inputs (either "0" or "1"). As a result, it only has two possible outputs ($f(0)$ and $f(1)$). *Our goal is still to find out if the oracle is "balanced" or "constant"* ($f(0) = f(1)$). Since it only has two possible outputs, the "balanced" case is the same as a "non-constant" case as there is only one non-constant case, i.e. $f(0) \neq f(1)$. While in the Deutsch–Jozsa problem with $n > 1$, non-constant is not necessarily balanced.

21.2.2 Classical Algorithm

To find out if the solution is constant, in a classical way, we need to provide the input one by one and measure the outputs. If both outputs are the same, then it is a constant oracle. Otherwise, it is balanced (or non-constant). Therefore, we need to use the oracle to perform **two** computations to know the answer. If I only have one computer and each of the computations will take 50 years, I will not know the

answer before I die. Why do I bother? Well, maybe that oracle will perform a very sophisticated computation to tell me if the girl likes me.

Example 21.3 In the Deutsch problem, what are the possible input and output combinations?

Since there are two inputs and two outputs, there are only 4 types of functions.

$$\begin{aligned} f_A(0) &= 0 \quad \text{and} \quad f_A(1) = 0 \\ f_B(0) &= 0 \quad \text{and} \quad f_B(1) = 1 \\ f_C(0) &= 1 \quad \text{and} \quad f_C(1) = 0 \\ f_D(0) &= 1 \quad \text{and} \quad f_D(1) = 1 \end{aligned} \tag{21.3}$$

We see that f_A and f_D are constant functions while f_B and f_C are balanced functions (or non-constant functions). And there is no way that we can determine if the function is constant or not without performing two calculations (i.e. computing $f(0)$ and $f(1)$).

21.2.3 Quantum Computing Solution

Luckily, we have quantum computers. By using the Deutsch Algorithm, we only need to perform one computation by using **quantum parallelism** and I will know the answer within 50 years!

***Step 0**: Create a Quantum Oracle*

We will discuss the **quantum oracle** in detail in the next chapter. Now we will only learn the essence of quantum oracle so we can understand the Deutsch algorithm. Figure 21.1 shows a quantum oracle embedded in a quantum circuit of 2 qubits. *A quantum oracle is just a quantum gate*! It needs to be **unitary**. The quantum oracle encodes the problem we are trying to solve. Therefore, a quantum oracle is so special

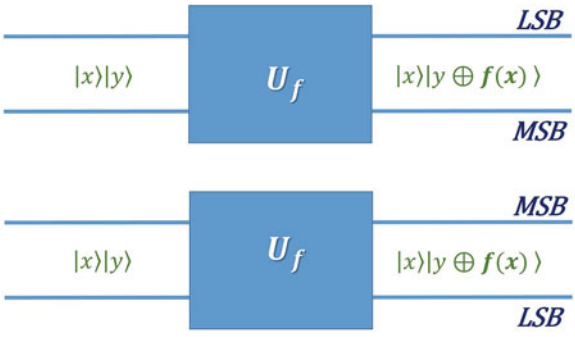


Fig. 21.1 The quantum oracle of a 2 qubit circuit used in the Deutsch algorithm. The top is the convention we have been using (IBM-Q) with MSB at the bottom and the bottom one is the other version often seen in other books. Both are using the little-endian convention. Note that $|x\rangle$ $|y\rangle$ are basis vectors

is just because we might not know how to construct it using fundamental quantum gates as it is given by the nature. Often, if we can construct the quantum oracle in a short time using simple quantum gates, then we do not need a quantum computer. For example, if I can construct the quantum oracle corresponding to the $f(x)$ in the Deutsch problem, then it means I already know what it is and I probably know if it is constant. Of course, it is also possible that I can construct it in a short time but I am not able to evaluate it fast. Then, we still need a quantum computer. Anyway, we will just treat it as a *black box* and we assume some physical systems in nature can be used to create this oracle in the quantum computer if we do not know how to create it from simple quantum gates.

Since a quantum oracle is just a quantum gate, the definition of a quantum oracle is still defined by *how it transforms the basis states* just like how we defined a simple quantum gate. Figure 21.1 shows the definition. The symbol of the oracle is $\boldsymbol{U}_f$, signifying that this is a unitary gate corresponding to the oracle function, $f(x)$. For any **basis vector**, $|x\rangle\,|y\rangle$, the output is $|x\rangle\,|y \oplus f(x)\rangle$, where $\oplus$ is the classical XOR and $f(x)$ is the function we are evaluating. That is

$$\boldsymbol{U}_f\,|x\rangle\,|y\rangle = |x\rangle\,|y \oplus f(x)\rangle \tag{21.4}$$

This means that when the input is a basis vector with x being the MSB and y being the LSB, the output will be another basis vector with the MSB unchanged (still x) but the LSB becomes $y \oplus f(x)$, which depends on the $f(x)$ of the oracle.

Example 21.4 If $f(x)$ is a constant function that always gives "0" (i.e. $f_A(x)$ in Eq. (21.3)), how does the oracle transform the basis vectors?

There are only four basis vectors in a 2-qubit system, namely $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. By tracing the change of x and y carefully, we have

$$\begin{aligned}
\boldsymbol{U}_f\,|0\rangle\,|0\rangle &= |0\rangle\,|0 \oplus f(0)\rangle = |0\rangle\,|0 \oplus 0\rangle = |0\rangle\,|0\rangle \\
\boldsymbol{U}_f\,|0\rangle\,|1\rangle &= |0\rangle\,|1 \oplus f(0)\rangle = |0\rangle\,|1 \oplus 0\rangle = |0\rangle\,|1\rangle \\
\boldsymbol{U}_f\,|1\rangle\,|0\rangle &= |1\rangle\,|0 \oplus f(1)\rangle = |1\rangle\,|0 \oplus 0\rangle = |1\rangle\,|0\rangle \\
\boldsymbol{U}_f\,|1\rangle\,|1\rangle &= |1\rangle\,|1 \oplus f(1)\rangle = |1\rangle\,|1 \oplus 0\rangle = |1\rangle\,|1\rangle
\end{aligned} \tag{21.5}$$

We see that we can implement this oracle by an identity gate. *But we still do not know what is $f(x)$ in the oracle. We only know how it behaves.*

***Step 1**: Create a Superposition Input*

We will create a special superposition state as the input to the oracle. It is not surprising that a superposition state is needed because we want to use *quantum parallelism* to evaluate the function with different basis vectors at the same time.

The superposition state to be created is $|+\rangle\,|-\rangle$. Let us represent it in the original $|0\rangle$ / $|1\rangle$ basis.

$$\begin{aligned}|+\rangle\,|-\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ &= \frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)\end{aligned} \tag{21.6}$$

We can see that it has equal weight for every basis vector, but the phases are not the same.

***Step 2**: Evaluate Using the Quantum Oracle*
Now let us apply $\boldsymbol{U}_f$ to $|+\rangle\,|-\rangle$.

$$\begin{aligned}\boldsymbol{U}_f\,|+\rangle\,|-\rangle &= \boldsymbol{U}_f\left(\frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)\right) \\ &= \frac{1}{2}(\boldsymbol{U}_f\,|00\rangle - \boldsymbol{U}_f\,|01\rangle + \boldsymbol{U}_f\,|10\rangle - \boldsymbol{U}_f\,|11\rangle) \\ &= \frac{1}{2}(|0, 0 \oplus f(0)\rangle - |0, 1 \oplus f(0)\rangle + |1, 0 \oplus f(1)\rangle - |1, 1 \oplus f(1)\rangle)\end{aligned} \tag{21.7}$$

Here, we firstly used the linearity of quantum mechanics that I can apply $\boldsymbol{U}_f$ to individual basis vector first before performing the superposition. Then, we applied the definition of the quantum oracle in Eq. (21.4). Since we are very familiar with the bra–ket notation already, I write, for example, $|0\rangle\,|0 \oplus f(x)\rangle$ as $|0, 0 \oplus f(x)\rangle$ without ambiguity.

We know anything XOR with "0" is itself and anything XOR with "1" is its negation. Therefore, Eq. (21.7) can be further simplified as

$$\boldsymbol{U}_f\,|+\rangle\,|-\rangle = \frac{1}{2}\left(|0, f(0)\rangle - \left|0, \overline{f(0)}\right\rangle + |1, f(1)\rangle - \left|1, \overline{f(1)}\right\rangle\right) \tag{21.8}$$

where $\overline{f(x)}$ is the negation of $f(x)$. Note that $f(x)$ is either 0 or 1 in the Deutsch problem. For example, if $f(x) = 1$, the $\overline{f(x)} = 0$.

***Step 3**: Measure on the Right Basis to Get the Answer*
By inspecting Eq. (21.8) carefully, we start seeing some clues of how it works. If $f(x)$ is a constant function, then $f(0) = f(1)$ and $\overline{f(0)} = \overline{f(1)}$. I can then perform a factorization on the tensor product on Eq. (21.8),

$$\begin{aligned}\boldsymbol{U}_f\,|+\rangle\,|-\rangle &= \frac{1}{2}\left(|0, f(0)\rangle - \left|0, \overline{f(0)}\right\rangle + |1, f(1)\rangle - \left|1, \overline{f(1)}\right\rangle\right) \\ &= \frac{1}{2}\left(|0, f(0)\rangle - \left|0, \overline{f(0)}\right\rangle + |1, f(0)\rangle - \left|1, \overline{f(0)}\right\rangle\right)\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{2}\left(|0\rangle\,|f(0)\rangle - |0\rangle\left|\overline{f(0)}\right\rangle + |1\rangle\,|f(0)\rangle - |1\rangle\left|\overline{f(0)}\right\rangle\right) \\
&= \frac{1}{2}\left((|0\rangle + |1\rangle)\,|f(0)\rangle - (|0\rangle + |1\rangle)\left|\overline{f(0)}\right\rangle\right) \\
&= \frac{1}{\sqrt{2}}\left(|+\rangle\,|f(0)\rangle - |+\rangle\left|\overline{f(0)}\right\rangle\right) \\
&= |+\rangle\,\frac{1}{\sqrt{2}}\left(|f(0)\rangle - \left|\overline{f(0)}\right\rangle\right) \qquad (21.9)
\end{aligned}
$$

Here, I have used the fact that $f(0) = \overline{f(1)}$ and I write both as $f(0)$ for the factorization and similarly for the $\overline{f(0)} = f(1)$ case. We see the result is a tensor product of $|+\rangle$ (the MSB) and $\frac{1}{\sqrt{2}}(|f(0)\rangle - \left|\overline{f(0)}\right\rangle)$ (the LSB), which is just a linear superposition of the basis vectors. Therefore, if $f(x)$ is a constant function, the MSB is $|+\rangle$. Here, we also see that the oracle is only *evaluated once* but both $f(0)$ and $f(1)$ are evaluated *at the same time* to achieve quantum parallelism! We also see that the oracle keeps the MSB unchanged if it is constant (i.e. both the input and output are $|+\rangle$).

If $f(x)$ if balanced (non-constant), then $f(0) = \overline{f(1)}$ and $\overline{f(0)} = f(1)$. For example, if $f(0) = 1$ and $f(1) = 0$, then $f(0) = \overline{f(1)} = \overline{0} = 1$. I will perform a factorization on the tensor product on Eq. (21.8) in another way.

$$
\begin{aligned}
\boldsymbol{U}_f\,|+\rangle\,|-\rangle &= \frac{1}{2}\left(|0, f(0)\rangle - \left|0, \overline{f(0)}\right\rangle + |1, f(1)\rangle - \left|1, \overline{f(1)}\right\rangle\right) \\
&= \frac{1}{2}\left((|0\rangle - |1\rangle)\,|f(0)\rangle - (|0\rangle - |1\rangle)\left|\overline{f(0)}\right\rangle\right) \\
&= \frac{1}{\sqrt{2}}\left(|-\rangle\,|f(0)\rangle - |-\rangle\left|\overline{f(0)}\right\rangle\right) \\
&= |-\rangle\,\frac{1}{\sqrt{2}}\left(|f(0)\rangle - \left|\overline{f(0)}\right\rangle\right) \qquad (21.10)
\end{aligned}
$$

Here, I have used the fact that $f(0) = \overline{f(1)}$ and I write both as $f(0)$ for factorization and similarly for the $\overline{f(0)} = f(1)$ case, I use $\overline{f(0)}$ for both. Therefore, when the oracle is non-constant, the MSB is flipped from $|+\rangle$ to $|-\rangle$.

21.2.4 The Quantum Circuit

In summary, the quantum oracle in the Deutsch algorithm takes $|+-\rangle$ as the input and keeps the MSB as $|+\rangle$ if $f(x)$ is constant and flips it to $|-\rangle$ if it is balanced (non-constant). This is simple but we need to do two things to complete the circuit.

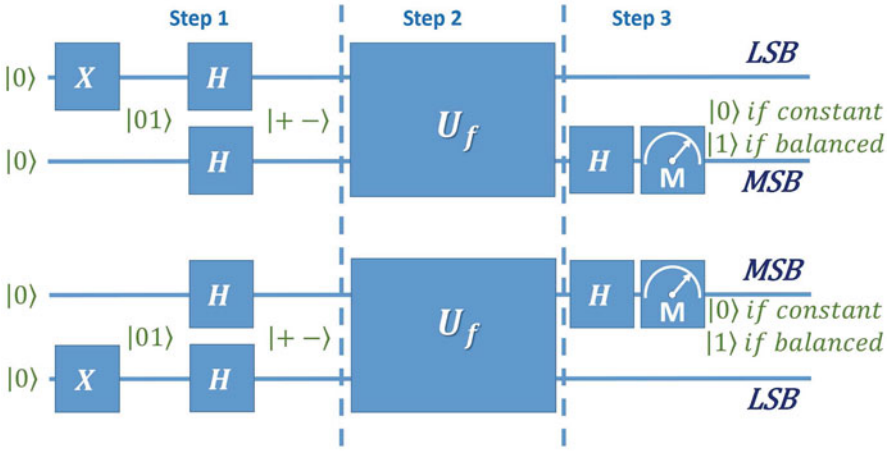


Fig. 21.2 The Deutsch algorithm circuit. The top is the convention we have been using (IBM-Q) with MSB at the bottom and the bottom one is the other version often seen in other books. Both are using little-endian convention

Firstly, we need to create $|+-\rangle$ and secondly, we need to measure in the $|0\rangle / |1\rangle$ basis to determine if the MSB is $|+\rangle$ or $|-\rangle$.

As mentioned before, we prefer to start the quantum circuits by initializing the input qubits to $|0\rangle_n$ with $n = 2$ in this case, i.e. $|0\rangle\,|0\rangle$. In order to prepare the $|+\rangle\,|-\rangle$ state, we will use the Hadamard gates. We know that $\boldsymbol{H}\,|0\rangle = |+\rangle$ and $\boldsymbol{H}\,|1\rangle = |-\rangle$ (Eq. (17.1)). So we need to create $|1\rangle$ first, and this can be done easily using the NOT-gate ($\boldsymbol{X}$). Step 1 in Fig. 21.2 shows the corresponding circuit. We can also represent Step 1 of the circuit using the following equation:

$$(\boldsymbol{H}\boldsymbol{H})(\boldsymbol{I}\boldsymbol{X})\,|0\rangle\,|0\rangle = (\boldsymbol{H}\boldsymbol{H})\,|0\rangle\,|1\rangle = |+\rangle\,|-\rangle \tag{21.11}$$

In this equation, we carefully apply each operator to its corresponding qubit. We can do this because the 2-qubit operators used can be factorized to be tensor products of two 1-qubit operators. For example, the first operator is represented as $\boldsymbol{I} \otimes \boldsymbol{X}$ and second operator, which is $\boldsymbol{H}^{\otimes 2}$, is represented as $\boldsymbol{H} \otimes \boldsymbol{H}$.

How do we determine if the MSB is $|+\rangle$ or $|-\rangle$ while we can only measure in the $|0\rangle / |1\rangle$ basis? The trick is again to use the Hadamard gate. We discussed this in the quantum teleportation algorithm already and it is shown in Eqs. (19.8) and (19.9) that *measuring a state in the* $|+\rangle / |-\rangle$ *basis is the same as measuring in the* $|0\rangle / |1\rangle$ *basis after applying a Hadamard gate*. Therefore, in Fig. 21.2, we apply a Hadamard gate after the quantum oracle and then perform a measurement on the MSB. If the result is $|0\rangle$ (corresponds to measuring $|+\rangle$ before the Hadamard gate), we know the function is constant. If the result is $|1\rangle$ (corresponds to measuring $|-\rangle$ before the Hadamard gate), we know the function is balanced.

Example 21.5 Show that in Step 3 in Fig. 21.2, if $f(x)$ is constant, measuring the MSB will always result in $|0\rangle$.

After the quantum oracle, the state becomes $\frac{1}{2}(|0,\, f(0)\rangle - \left|0,\, \overline{f(0)}\right\rangle + |1,\, f(1)\rangle - \left|1,\, \overline{f(1)}\right\rangle)$ (Eq. (21.8)). Applying a Hadamard gate to the MSB (which means we also need to apply an identity gate to the LSB), we have

$$
\begin{aligned}
&\boldsymbol{HI}\left(\frac{1}{2}\left(|0,\, f(0)\rangle - \left|0,\, \overline{f(0)}\right\rangle + |1,\, f(1)\rangle - \left|1,\, \overline{f(1)}\right\rangle\right)\right) \\
&= \frac{1}{2}\left(|+,\, f(0)\rangle - \left|+,\, \overline{f(0)}\right\rangle + |-,\, f(1)\rangle - \left|-,\, \overline{f(1)}\right\rangle\right) \\
&= \frac{1}{2}\left(|+\rangle\left(|f(0)\rangle - \left|\overline{f(0)}\right\rangle\right) + |-\rangle\left(|f(1)\rangle - \left|\overline{f(1)}\right\rangle\right)\right) \\
&= \frac{1}{2}\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\left(|f(0)\rangle - \left|\overline{f(0)}\right\rangle\right)\right. \\
&\quad \left. + \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\left(|f(1)\rangle - \left|\overline{f(1)}\right\rangle\right)\right) \\
&= \frac{1}{2^{3/2}}\left((|0\rangle + |1\rangle)\left(|f(0)\rangle - \left|\overline{f(0)}\right\rangle\right)\right. \\
&\quad \left. + (|0\rangle - |1\rangle)\left(|f(1)\rangle - \left|\overline{f(1)}\right\rangle\right)\right)
\end{aligned}
\tag{21.12}
$$

If $f(x)$ is constant, then $|f(0)\rangle - \left|\overline{f(0)}\right\rangle = |f(1)\rangle - \left|\overline{f(1)}\right\rangle$ as $f(0) = f(1)$. Therefore, by factorizing $|f(0)\rangle - \left|\overline{f(0)}\right\rangle$, Eq. (21.12) becomes

$$
\begin{aligned}
&\frac{1}{2^{3/2}}\left((|0\rangle + |1\rangle + |0\rangle - |1\rangle)\left(|f(0)\rangle - \left|\overline{f(0)}\right\rangle\right)\right) \\
&= \frac{1}{2^{3/2}}(2\,|0\rangle\left(|f(0)\rangle - \left|\overline{f(0)}\right\rangle\right)) \\
&= \frac{1}{\sqrt{2}}\left(|0\rangle\left(|f(0)\rangle - \left|\overline{f(0)}\right\rangle\right)\right)
\end{aligned}
\tag{21.13}
$$

Therefore, we always measure $|0\rangle$ for the MSB if the function is constant!

21.2.5 Run on IBM-Q

A quantum circuit corresponding to an oracle with $f(x) = 0$ (i.e. f_A in Eq. (21.3)) is implemented on IBM-Q (Fig. 21.3). Again, in IBM-Q, the bottom qubit is the MSB (corresponding to the top circuit of Fig. 21.2). As we mentioned, we usually do not

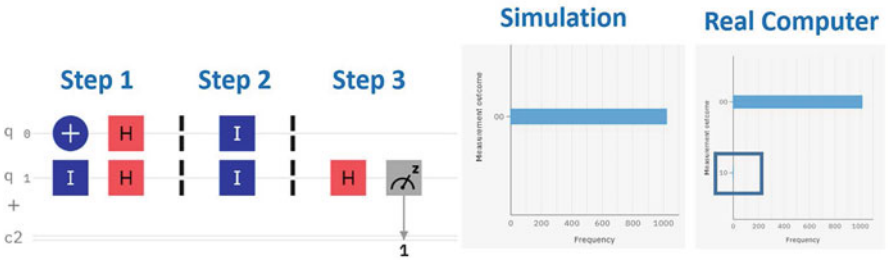


Fig. 21.3 Deutsch algorithm implemented in IBM-Q with a constant quantum oracle $f(x) = 0$. Simulation and hardware results are shown on the right. MSB is at the bottom

know the quantum oracle. Here, we just make a quantum oracle using basic gates to demonstrate the idea. As mentioned after Eq. (21.5), f_A is equivalent to an identity gate. Therefore, the quantum oracle is constructed simply using an identity gate. The measurement result is always 0 for the MSB, which confirms that the oracle is constant. Note again, since we are not measuring the LSB, so by default, it is set to 0 in IBM-Q. Therefore, we only get $|00\rangle$. Similar to what we saw in other circuits, due to errors, the hardware sometimes measures $|10\rangle$, which is wrong.

21.3 Summary

We understand that the Deutsch–Jozsa problem is for checking if a function is constant or balanced. The Deutsch problem is a special case of it with $n = 1$. However, we need 2 qubits to implement. It will calculate $f(0)$ and $f(1)$ in parallel by taking a superposition of the basis states as an input. If $f(x)$ is constant (or balanced), we always measure $|0\rangle$ (or $|1\rangle$) in the MSB. Although this is a simple speed up, we can imagine that with more qubits (e.g. 30 qubits), we can perform many computations at the same time (e.g. $2^{30} \approx 10^9$) for some more complex problems. We have first encountered a quantum oracle. It is nothing but usually a black box. It is also just a unitary quantum gate. To demonstrate the algorithm, we may construct an equivalent quantum gate to represent the quantum oracle. But usually, we rely on a real physical system to be the quantum oracle in a quantum computer.

Finally, I want to emphasize one very important observation. *While parallel computation is performed in the Deutsch algorithm, we do NOT know the values of* $f(0)$ *and* $f(1)$. *We only know if* $f(x)$ *is constant or balanced.* **This is the nature of many quantum computing algorithms from which we usually can only extract a limited amount of information.**

Problems

21.1 Other forms of *f(x)*
Repeat what we did in Eq. (21.5) for f_B, f_C, and f_D.

21.2 Creation of superposition
Repeat Eq. (21.11) using matrix.

21.3 Hadamard gate applied to 2-qubit basis state
Creating the superposition in Step 1 in Fig. 21.2 after the NOT-gate is equivalent to applying a 2-qubit Hadamard gate to state $|01\rangle$. Show that you can get the result in Eq. (21.6) by using Eq. (17.21).

21.4 Non-Constant Function in the Deutsch algorithm
Repeat Example 21.5 for a balanced function.

Chapter 22
Quantum Oracles and Construction of Quantum Gate Matrices

22.1 Learning Outcomes

Able to distinguish the two types of quantum oracles, namely the XOR and phase quantum oracles; Able to explain why a quantum oracle needs to be unitary and reversible; Able to construct the matrix of any quantum gate when the definition is given.

22.2 Quantum Oracle

We introduced the concept of **quantum oracle** in the previous chapter. Since it is an oracle, very often we do not know how it is implemented. We only know that its behavior has some correlations to the problem that we try to solve. The problem we are solving usually is related to a function $f(x)$ (such as the $f(x)$, which we want to know if it is constant or balanced in the Deutsch algorithm). It can be as simple as a p-qubit identity gate ($\boldsymbol{I}^{\otimes p}$) (e.g. in Fig. 21.3, $p = 2$). But usually, it is complex and we can use some physical systems to implement it. We will create an equivalent oracle when we need to test the circuit. But most of the time, we can just treat it as a black box.

There are two types of quantum oracles. One is called the **XOR quantum oracle** while the other is called the **phase quantum oracle**. A quantum oracle is nothing but just a *quantum gate*. Therefore, it must be **unitary** and **reversible**. Thus, a quantum oracle is represented as $\boldsymbol{U}_f$, which is a unitary operator corresponding to the function $f(x)$.

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_22

22.2.1 XOR Quantum Oracle

For any function $f(x)$, it maps n bits of inputs to m bits of outputs. Mathematically, it is written as $f : \{0,1\}^n \mapsto \{0,1\}^m$. For example, in the Deutsch problem, we have $n = m = 1$. That is, both input and output have 1-qubit. In general, $n \neq m$. Classically, we can build a classical gate to map the n-bit inputs to the m-bit outputs. But we cannot do this in quantum computing. Why? This is because every quantum gate needs to be reversible. And this is only possible if $n = m$. For example, for $n > m$, many different inputs can produce the same output. Then for a given output, we cannot find the corresponding input in the reverse direction (see Fig. 2.2). Or we may say that an n-qubit input m-qubit output matrix is not square (it is $m \times n$) and thus it cannot be unitary.

This issue can be solved by implementing an *XOR quantum oracle*. It has $(n+m)$-qubit input and output (Fig. 22.1). Like for other quantum gates, we define the XOR quantum oracle by describing how it transforms a basis vector to other basis vectors. Note that now the XOR quantum oracle is $(n+m) \times (n+m)$. If $f(x)$ has n-bit inputs and m-bit outputs, we denote the input basis vector as $|x\rangle_n |y\rangle_m$ (i.e. the n MSB are denoted as $|x\rangle$ and the m LSB are denoted as $|y\rangle_m$). Here $|x\rangle_n$ is the **query**, which represents the input to $f(x)$ and $|y\rangle_m$ are the **auxiliary qubits**. It is natural to add the m auxiliary bits because we need to store the result $f(x)$ somehow. An XOR quantum oracle is defined as that it will keep the n MSB unchanged but the m LSB will be changed to $y \oplus f(x)$. So the output becomes $|x\rangle_n |y \oplus f(x)\rangle_m$. Note that $y \oplus f(x)$ is a classical bit-wise XOR operator. Similar to Eq. (21.4), we can represent its behavior as

$$\boldsymbol{U}_f |x\rangle_n |y\rangle_m = |x\rangle_n |y \oplus f(x)\rangle_m \tag{22.1}$$

where $|x\rangle_n |y\rangle_m$ is a basis vector in the $\mathbb{C}^{2^{(n+m)}}$ space. In Eq. (21.4), $n = m = 1$.

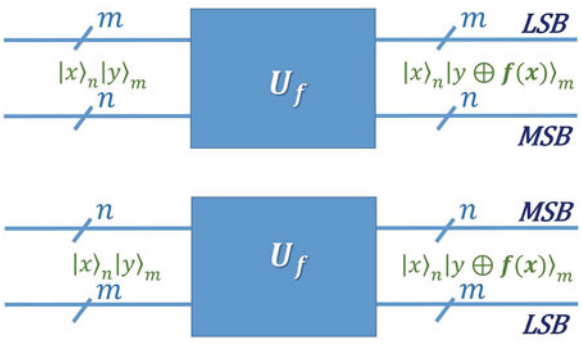


Fig. 22.1 A general XOR quantum oracle. The top one is the convention we have been using (IBM-Q) with MSB at the bottom and the bottom one is the other version often seen in other books. Both are using the little-endian convention. Note that $|x\rangle_n |y\rangle_m$ are basis vectors

Example 22.1 For $n = 3$, $m = 2$, and let $x = 3$ and $f(3) = 3$, find the corresponding input and output basis vectors of the XOR quantum oracle corresponding to $f(x)$.

The m LSB part of the input is not given in the question, so we assume it is $|0\rangle_m$. Therefore, the input basis vector is $|x\rangle_n |y\rangle_m = |3\rangle_3 |0\rangle_2$ where we substituted the decimal values of x, n, and m. This can be written as $|011\rangle |00\rangle = |01100\rangle$, which is a 5-qubit ($m + n = 5$) input.

The output vector is also a basis vector and can be found as

$$\begin{aligned} \boldsymbol{U}_f |3\rangle_3 |0\rangle_2 &= |3\rangle_3 |0 \oplus f(3)\rangle_2 = |3\rangle_3 |0 \oplus 3\rangle_2 \\ &= |011\rangle |(00) \oplus (11)\rangle \\ &= |011\rangle |11\rangle \\ &= |01111\rangle \end{aligned} \tag{22.2}$$

where $(00) \oplus (11) = (0 \oplus 1)(0 \oplus 1) = (11)$ is a bit-wise XOR operation. Therefore, this XOR quantum oracle transforms $|01100\rangle$ to $|01111\rangle$ in this example.

Now, let us prove that the XOR quantum oracle is indeed reversible. We first apply $\boldsymbol{U}_f$ twice to the input.

$$\begin{aligned} \boldsymbol{U}_f \boldsymbol{U}_f |x\rangle_n |y\rangle_m &= \boldsymbol{U}_f |x\rangle_n |y \oplus f(x)\rangle_m \\ &= |x\rangle_n |y \oplus f(x) \oplus f(x)\rangle_m \\ &= |x\rangle_n |y \oplus 0\rangle_m \\ &= |x\rangle_n |y\rangle_m \\ &= \boldsymbol{I}^{\otimes(n+m)} |x\rangle_n |y\rangle_m \end{aligned} \tag{22.3}$$

where we use the fact that $f(x) \oplus f(x) = 0$ and $y \oplus 0 = y$ because anything XOR with itself results in 0 and anything XOR with 0 is still itself. Therefore, $\boldsymbol{U}_f \boldsymbol{U}_f = \boldsymbol{I}^{\otimes(n+m)}$ (i.e. an $(n+m)$-dimensional identity matrix). But we know that $\boldsymbol{U}_f \boldsymbol{U}_f^{-1} = \boldsymbol{I}^{\otimes(n+m)}$, where $\boldsymbol{U}_f^{-1}$ is the inverse of $\boldsymbol{U}_f$. Therefore, $\boldsymbol{U}_f = \boldsymbol{U}_f^{-1}$. This also means that the inverse of $\boldsymbol{U}_f$ exits. Therefore, $\boldsymbol{U}_f$ is reversible.

We have used too much math in the last paragraph. Actually, Eq. (22.3) shows that we can get back $|x\rangle_n |y\rangle_m$ when the input is $|x\rangle_n |y \oplus f(x)\rangle_m$ (i.e. $\boldsymbol{U}_f |x\rangle_n |y \oplus f(x)\rangle_m = |x\rangle_n |y\rangle_m$). So the inverse exists and it just happens that in this case, the inverse is the quantum oracle itself. It means that *the quantum oracle looks the same whether you look from the left or the right!*

Based on the derivation, you can also see that the XOR quantum oracle is reversible *by construction*. The XOR quantum oracle is also unitary by construction. We will try to prove this in the Problems.

22.2.2 Phase Quantum Oracle

In some applications, we encode $f(x)$ so that it changes the phase of the input basis vector based on $f(x)$ and uses it as an output. In this case, we only need n qubits for both the input and the output. The phase quantum oracle matrix has a dimension of $n \times n$. Again, the phase quantum oracle is just a quantum gate and it is defined by how it transforms the basis vectors.

$$\boldsymbol{U}_f \, |x\rangle_n = (-1)^{f(x)} \, |x\rangle_n \tag{22.4}$$

where $|x\rangle_n$ is an n-qubit basis vector.

In the phase quantum oracle, we see that if $f(x)$ is even, the output is the same as the input. And if $f(x)$ is odd, the output is the same as the input with a phase shift of π. The phase shift is π because $e^{i\pi} = \cos\pi + i \sin\pi = -1$. Figure 22.2 shows the phase quantum oracle.

At this point, I hope you can appreciate that *a quantum oracle can be used to "express" the effect of* $f(x)$ *(i.e. the answer to our query) in any form as long as it can help us achieve the final goal.* For an XOR quantum oracle, $f(x)$ is stored (or encoded) in the m-qubits and it is just $f(x)$ if $y = 0$. But in a phase quantum oracle, $f(x)$ only affects the phase of the output (therefore, less information is retained).

Similar to the XOR quantum oracle, a phase quantum oracle needs to be reversible and unitary. To show that it is reversible, we again apply it twice to the input.

$$\begin{aligned}
\boldsymbol{U}_f \boldsymbol{U}_f \, |x\rangle_n &= \boldsymbol{U}_f (-1)^{f(x)} \, |x\rangle_n \\
&= (-1)^{f(x)} \boldsymbol{U}_f \, |x\rangle_n \\
&= (-1)^{f(x)} (-1)^{f(x)} \, |x\rangle_n \\
&= (-1)^{2f(x)} \, |x\rangle_n \\
&= |x\rangle_n \\
&= \boldsymbol{I}^{\otimes n} \, |x\rangle_n
\end{aligned} \tag{22.5}$$

Therefore, it is reversible because its inverse exists (and equals to the oracle itself) by using the reasoning we used in the XOR quantum oracle case. It is also unitary. The proof is straightforward. We will prove by showing that the set of the

Fig. 22.2 A general phase quantum oracle. Note that $|x\rangle_n$ is a basis vector

basis vectors is still orthonormal after being transformed by the phase quantum oracle and thus the phase quantum oracle is unitary (see Problem 9.4).

For any two basis vectors, $|a\rangle$ and $|b\rangle$, their inner product is $\langle a|b\rangle$. If they are transformed by a phase quantum oracle, they become $(-1)^{f(a)}|a\rangle$ and $(-1)^{f(b)}|b\rangle$, respectively. The inner product of the transformed basis vectors is $(-1)^{(f(a)^*+f(b))}\langle a|b\rangle$. But note, $f(a)$ and $f(b)$ are real numbers, because they are the outputs of the mapping $f : \{0,1\}^n \mapsto \{0,1\}^m$ and are just real binary numbers. If $a = b$, then $(-1)^{f(a)^*+f(b)} = (-1)^{f(a)+f(b)} = (-1)^{2f(a)} = 1$ and the inner product is still the same (i.e. $(-1)^{f(a)^*+f(b)}\langle a|b\rangle = 1 \times \langle a|a\rangle = 1$). If $a \neq b$, then the inner product is still 0 as $\langle a|b\rangle = 0$. Therefore, the phase quantum oracle keeps the inner products of the basis vectors unchanged (so still orthonormal) and, thus, it is unitary.

22.3 Construction of Quantum Gates and Oracles

I have not told you how to systematically construct the matrix for a quantum gate (also for a quantum oracle that is just a quantum gate). It is easy as each quantum gate is defined by how it transforms the basis vectors. Since every basis vector has only 1 non-zero entry in its column form, how it is transformed defines the corresponding row of the matrix. For example, the 2nd basis vector (we count from 0th) in a 4-dimensional space has the form $\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$. When it is transformed by a matrix of a quantum gate, it only affects the output through the 2nd row (again we count from 0th row) of the matrix due to the way it is multiplied. Based on this understanding, we can guess the equation for a quantum gate construction. For a quantum gate, $\boldsymbol{U}$, with a dimension of $N \times N$,

$$\boldsymbol{U} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \langle j|\, \boldsymbol{U}\, |i\rangle\, |j\rangle\, \langle i| \tag{22.6}$$

where $|i\rangle$ and $|j\rangle$ are the basis vectors. $|j\rangle\langle i|$ is the outer produce to define the location of the element in the matrix ($j-th$ row $i-th$ column). $\langle j|\,\boldsymbol{U}\,|i\rangle$ tells us how much of $|i\rangle$ will contribute to the formation of $|j\rangle$ after the transformation.

But this is just Eq. (12.20) except I swapped i and j that is immaterial! Yes, I am just telling you how to construct a matrix from another angle, but they are the same.

Let us look at some examples and it will be clearer.

Example 22.2 Construct the matrix for a 1-bit NOT-gate.

We repeat the definition of NOT gate here (Eq. (15.5)),

$$\begin{aligned} \boldsymbol{U_{NOT}} |0\rangle &= |1\rangle \\ \boldsymbol{U_{NOT}} |1\rangle &= |0\rangle \end{aligned} \tag{22.7}$$

We see that $|0\rangle$ is transformed wholly to $|1\rangle$ and $|1\rangle$ is transformed wholly to $|0\rangle$. Now let us apply Eq. (22.6) with $N = 2$ as this is a 1-qubit gate with only 2 basis vectors.

$$\begin{aligned} \boldsymbol{U_{NOT}} &= \sum_{i=0}^{2-1} \sum_{j=0}^{2-1} \langle j| \boldsymbol{U_{NOT}} |i\rangle |j\rangle \langle i| \\ &= \langle 0| \boldsymbol{U_{NOT}} |0\rangle |0\rangle \langle 0| + \langle 1| \boldsymbol{U_{NOT}} |0\rangle |1\rangle \langle 0| \\ &\quad + \langle 0| \boldsymbol{U_{NOT}} |1\rangle |0\rangle \langle 1| + \langle 1| \boldsymbol{U_{NOT}} |1\rangle |1\rangle \langle 1| \\ &= \langle 0|1\rangle |0\rangle \langle 0| + \langle 1|1\rangle |1\rangle \langle 0| \\ &\quad + \langle 0|0\rangle |0\rangle \langle 1| + \langle 1|0\rangle |1\rangle \langle 1| \\ &= |1\rangle \langle 0| + |0\rangle \langle 1| \qquad (22.8) \\ &= \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad (22.9) \end{aligned}$$

Here we have used Eq. (22.7) from the 2nd line to the 3rd line. Then we use the orthonormal properties of the basis vectors from the 3rd line to the 4th line.

Example 22.3 Construct the matrix for a 1-bit Hadamard gate. We repeat the definition of the Hadamard gate here (Eq. (17.1)),

$$\begin{aligned} \boldsymbol{H} |0\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle \\ \boldsymbol{H} |1\rangle &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle \end{aligned} \tag{22.10}$$

Now let us apply Eq. (22.6) with $N = 2$ as this is a 1-qubit gate with only 2 basis vectors.

$$
\begin{aligned}
\boldsymbol{H} &= \sum_{i=0}^{2-1}\sum_{j=0}^{2-1} \langle j|\,\boldsymbol{H}\,|i\rangle\,|j\rangle\,\langle i| \\
&= \langle 0|\,\boldsymbol{H}\,|0\rangle\,|0\rangle\,\langle 0| + \langle 1|\,\boldsymbol{H}\,|0\rangle\,|1\rangle\,\langle 0| \\
&\quad + \langle 0|\,\boldsymbol{H}\,|1\rangle\,|0\rangle\,\langle 1| + \langle 1|\,\boldsymbol{H}\,|1\rangle\,|1\rangle\,\langle 1| \\
&= \langle 0|\,(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle))\,|0\rangle\,\langle 0| + \langle 1|\,(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle))\,|1\rangle\,\langle 0| \\
&\quad + \langle 0|\,(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle))\,|0\rangle\,\langle 1| + \langle 1|\,(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle))\,|1\rangle\,\langle 1| \\
&= \frac{1}{\sqrt{2}}(|0\rangle\,\langle 0| + |1\rangle\,\langle 0| + |0\rangle\,\langle 1| - |1\rangle\,\langle 1|) \\
&= \frac{1}{\sqrt{2}}(\begin{pmatrix}1\\0\end{pmatrix}\begin{pmatrix}1&0\end{pmatrix} + \begin{pmatrix}0\\1\end{pmatrix}\begin{pmatrix}1&0\end{pmatrix} + \begin{pmatrix}1\\0\end{pmatrix}\begin{pmatrix}0&1\end{pmatrix} - \begin{pmatrix}0\\1\end{pmatrix}\begin{pmatrix}0&1\end{pmatrix}) \\
&= \frac{1}{\sqrt{2}}(\begin{pmatrix}1&0\\0&0\end{pmatrix} + \begin{pmatrix}0&0\\1&0\end{pmatrix} + \begin{pmatrix}0&1\\0&0\end{pmatrix} - \begin{pmatrix}0&0\\0&1\end{pmatrix}) \\
&= \frac{1}{\sqrt{2}}\begin{pmatrix}1&1\\1&-1\end{pmatrix}
\end{aligned}
\tag{22.11}
$$

Here we have used Eq. (22.10) and the orthonormal properties of the basis vectors.

Now let us try to construct the matrix for a quantum oracle. We already did that in the previous chapter when $f(x)$ outputs constantly 0 (Eq. (21.5) and Fig. 21.3). The oracle can be easily constructed by using an identity gate. If we use Eq. (22.6), we will get the same answer. We will try a more complex example.

Example 22.4 Find the matrix of a XOR quantum oracle in the Deutsch algorithm corresponding to f_B in Eq. (21.3). Then construct the oracle using fundamental quantum gate(s) and simulate on IBM-Q.

Equation (21.3) is repeated for convenience.

$$
f_B(0) = 0 \quad \text{and} \quad f_B(1) = 1 \tag{22.12}
$$

Therefore, the quantum oracle transforms the basis as

$$
\begin{aligned}
\boldsymbol{U}_f\,|0\rangle\,|0\rangle &= |0\rangle\,|0 \oplus f(0)\rangle = |0\rangle\,|0 \oplus 0\rangle = |0\rangle\,|0\rangle \\
\boldsymbol{U}_f\,|0\rangle\,|1\rangle &= |0\rangle\,|1 \oplus f(0)\rangle = |0\rangle\,|1 \oplus 0\rangle = |0\rangle\,|1\rangle \\
\boldsymbol{U}_f\,|1\rangle\,|0\rangle &= |1\rangle\,|0 \oplus f(1)\rangle = |1\rangle\,|0 \oplus 1\rangle = |1\rangle\,|1\rangle \\
\boldsymbol{U}_f\,|1\rangle\,|1\rangle &= |1\rangle\,|1 \oplus f(1)\rangle = |1\rangle\,|1 \oplus 1\rangle = |1\rangle\,|0\rangle
\end{aligned}
\tag{22.13}
$$

where we substituted $f(0) = 0$ and $f(1) = 1$ based on Eq. (22.12). This is the *definition* of this oracle.

In this case, $N = 4$ with 4 basis vectors ($|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$) and there are $4 \times 4 = 16$ elements in the matrix. This is too cumbersome to write down. But the definition of this oracle is simple. The basis vectors are transformed to themselves or the other as a whole (instead of a linear combination of basis vectors like in the Hadamard gate). So there are only 4 non-zero elements. Based on the definition of this oracle (i.e. Eq. (22.13)), only these 4 terms are non-zero:

$$\begin{aligned}
\langle 0|\langle 0|(\boldsymbol{U}_f|0\rangle|0\rangle) &= \langle 0|\langle 0||0\rangle|0\rangle = \langle 00||00\rangle = 1 \\
\langle 0|\langle 1|(\boldsymbol{U}_f|0\rangle|1\rangle) &= \langle 0|\langle 1||0\rangle|1\rangle = \langle 01||01\rangle = 1 \\
\langle 1|\langle 1|(\boldsymbol{U}_f|1\rangle|0\rangle) &= \langle 1|\langle 1||1\rangle|1\rangle = \langle 11||11\rangle = 1 \\
\langle 1|\langle 0|(\boldsymbol{U}_f|1\rangle|1\rangle) &= \langle 1|\langle 0||1\rangle|0\rangle = \langle 10||10\rangle = 1
\end{aligned} \tag{22.14}$$

For example, in the 3rd line, this term is non-zero because we know that $|1\rangle|0\rangle$ is transformed to $|1\rangle|1\rangle$. Therefore, the oracle is (using Eq. (22.6) but omitting the zero terms)

$$\begin{aligned}
\boldsymbol{U}_f &= \langle 0|\langle 0|(\boldsymbol{U}_f|0\rangle|0\rangle)|00\rangle\langle 00| + \langle 0|\langle 1|(\boldsymbol{U}_f|0\rangle|1\rangle)|01\rangle\langle 01| \\
&\quad + \langle 1|\langle 1|(\boldsymbol{U}_f|1\rangle|0\rangle)|11\rangle\langle 10| + \langle 1|\langle 0|(\boldsymbol{U}_f|1\rangle|1\rangle)|10\rangle\langle 11| \\
&= |00\rangle\langle 00| + |01\rangle\langle 01| + |11\rangle\langle 10| + |10\rangle\langle 11| \\
&= \begin{pmatrix}1\\0\\0\\0\end{pmatrix}\begin{pmatrix}1&0&0&0\end{pmatrix} + \begin{pmatrix}0\\1\\0\\0\end{pmatrix}\begin{pmatrix}0&1&0&0\end{pmatrix} + \begin{pmatrix}0\\0\\0\\1\end{pmatrix}\begin{pmatrix}0&0&1&0\end{pmatrix} + \begin{pmatrix}0\\0\\1\\0\end{pmatrix}\begin{pmatrix}0&0&0&1\end{pmatrix} \\
&= \begin{pmatrix}1&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{pmatrix} + \begin{pmatrix}0&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&0&0\end{pmatrix} + \begin{pmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&1&0\end{pmatrix} + \begin{pmatrix}0&0&0&0\\0&0&0&0\\0&0&0&1\\0&0&0&0\end{pmatrix} \\
&= \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}
\end{aligned} \tag{22.15}$$

This is just a CNOT gate. So we can implement the Deutsch algorithm for this type of function by using the CNOT gate as the oracle. Figure 22.3 shows the Deutsch algorithm for the corresponding $f(x)$ implemented in IBM-Q. Note that the MSB is at the bottom, and it is always measured to be "1." This is consistent with the fact that the quantum oracle corresponds to a balanced function. When it is run on a real computer, occasionally it measures "0" due to errors. If you compare

Fig. 22.3 Deutsch algorithm implemented in IBM-Q with a balanced quantum oracle $f(0) = 0$ and $f(1) = 1$. Simulation and hardware results are shown on the right. MSB is at the bottom

the error in this circuit to that in Fig. 21.3, it is much larger. This is because the CNOT gate introduces additional errors.

22.4 Summary

We learned the definition of XOR and phase quantum oracles. We understand that they are unitary and reversible by construction. This is expected because quantum oracles are just quantum gates. We also learned and practiced how to create the matrix for any given quantum oracle or gate based on its definition (i.e. how it transforms the basis states).

Problems

22.1 The General Form of XOR Quantum Oracle

Assume $x = 5$, $f(5) = 3$, $n = 3$, and $m = 3$, find the corresponding input and output basis vectors of the XOR quantum oracle corresponding to $f(x)$.

22.2 Unitarity of XOR Quantum Oracle

Prove that the XOR quantum oracle is unitary. Prove it using the same methodology we used for the phase quantum oracle.

22.3 Construction of SWAP-gate

Construct the matrix for SWAP gate using Eq. (22.6).

22.4 Deutsch Algorithm

Find the quantum oracle for f_C and f_D in Eq. (21.3). Implement them on IBM-Q.

Chapter 23
Grover's Algorithm: I

23.1 Learning Outcomes

Able to perform simple computational complexity analysis; Understand the meaning of exponential and quadratic speedups; Understand the concept of basis encoding; Able to describe the meanings and roles of the three important vectors and the two important matrices in Grover's algorithm; Able to explain Grover's algorithm pictorially.

23.2 Grover's Algorithm

23.2.1 Computational Complexity

Grover's algorithm is a quantum search algorithm. In the big data era, it is not difficult to appreciate the importance of an efficient search algorithm. Grover's algorithm provides a **quadratic speedup** over the classical one. Let us spend some time understanding the terminologies of computational complexity.

A computation problem usually takes more time as its size increases. For example, assume we do addition by adding each pair of single digits in a step. To do a 1-digit addition (e.g. $3 + 4$), it might take you 1 second because you only need to add 1 pair of single digits. For a 2-digit addition (e.g. $13 + 24$), it will take you about 2 seconds because you need to add two pairs of single digits and it takes two steps (let us ignore the time due to carry over). If your computation efficiency does not decrease as the number of digits increases, it takes you about N seconds to compute an N-bit addition. We say the computational complexity of this problem is $\mathcal{O}(N)$. This means that the computation time goes up linearly with the size of the problem (N). If I find a method in which the computational complexity becomes $\mathcal{O}(\log N)$, then the new algorithm achieves an exponential speedup over

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_23

the original one. This is called so because $10^{\log N} = N$. The original complexity is exponentially more than the new algorithm's. For example, to compute a 1-trillion-digit addition, in the old algorithm, I need to spend 1 trillion seconds. With the new algorithm, I only need to spend $\log 10^{12} = 12$ seconds! As discussed earlier, the **Deutsch–Jozsa Algorithm** provides an exponential speedup over the classical algorithm. And its computational complexity is not just $\mathcal{O}(\log N)$ but $\mathcal{O}(1)$, which means it is independent of the size of the problem! This is the most attractive part of quantum computing.

However, not all quantum algorithms can provide an exponential speedup. Assume you invented another algorithm. For a 1-digit addition, you need $\sqrt{1}$ second. For a 2-digit addition, you need $\sqrt{2}$ seconds. For an N-digit addition, you need $\sqrt{N}$ seconds. Then this has a complexity of $\mathcal{O}(\sqrt{N})$. We say that this algorithm has a quadratic speedup over the original one because $(\sqrt{N})^2 = N$. Therefore, for a 1-trillion-digit addition, I only need to spend $\sqrt{10^{12}} = 10^6$ seconds. *Grover's algorithm is such an algorithm that provides a quadratic speedup over the classical one*.

I also want to emphasize that, in reality, just comparing the algorithm speedup is incomplete. We also need to compare the amount of hardware required. If I can get hardware for free to do parallel classical computations, then quantum computing is not so attractive. This is not a trivial topic but I would like you to keep this in mind.

23.2.2 The Problem

Assume we have an unstructured database of N entries. An unstructured database means that the data are not organized in a certain way. We have no additional knowledge on how the data are stored besides we know that the data we are looking for is somewhere in the database. A structured one may be a sorted database (e.g. the names are sorted) and we can utilize certain classical algorithms (e.g. binary search) to speed up the searching process. Since it is unstructured, I need to search from the beginning to the end. If I am lucky, the first entry might be the one I am looking for. Then it will take me only 1 unit of time. If I am unlucky (or equally lucky), the target entry might be at the end of the database and it will take me N units of time. On average, it will take me $\frac{N}{2}$ units of time to find a target entry. Therefore, the complexity in a classical search is $\mathcal{O}(\frac{N}{2}) \sim \mathcal{O}(N)$. Note that in complexity calculation, the constant coefficients are usually not important. Indeed, there is no difference between whether the boy needs to spend 1 million years or 2 million years to find out if the girl likes him. They are both equally too long for him.

What is the meaning of search? It is a series of computations on each entry until we find what we want. When the computation meets certain criteria, we say that we have found the entry. For example, I need to find the grade of a student. I will look for the student's ID in each entry. And this is a computation. The computation might be doing subtraction between the ID of each entry and the ID I am looking

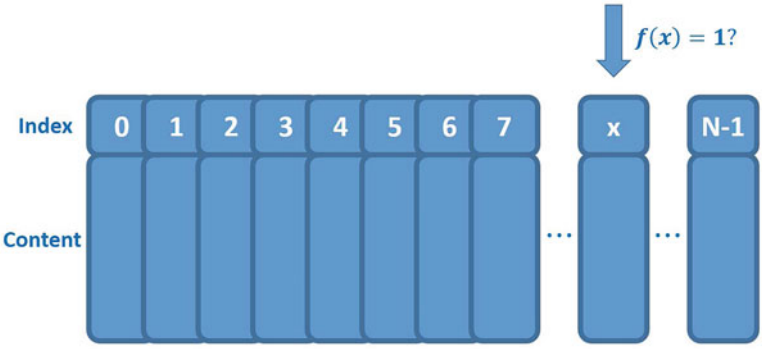


Fig. 23.1 Illustration of a searching process in a database with N entries. A function $f(x)$ is used to determine if the target is located at entry x

for. If the result is 0, then I know I have found the ID. This is a simple computation. Other computations can be more complex and take more time. For example, it can be finding the image or movie of a student using an inference neural network. No matter how complex it is, any search is related to the computation of a function $f(x)$, where x is the index of an entry in the database. Note that even the function depends only explicitly on x, it will take the entry content (e.g. the student ID or an image in the entry x) for computation. Of course, the content of each entry is a function of the index. Therefore, $f(x)$ completely describes the complex computation process. Our goal is to search for the entry that matches certain criteria. Let us set the criteria as $f(x) = 1$ if x is the index of the entry we are searching for. Assume entry with index a contains what we are searching for, then $f(a) = 1$. And $f(x) = 0$ if $x \neq a$. Figure 23.1 illustrates the searching process.

23.2.3 An Overview of Grover's Algorithm

We will first discuss the general idea of Grover's algorithm. And then we will construct the individual parts. In the next chapter, we will perform numerical substitutions and implement them in IBM-Q. We will use the notations in Mermin's book, "Quantum Computer Science: An Introduction". Assume what we are searching resides in entry with index $x = a$. For simplicity, we assume $N = 2^n$. We can always make up redundant entries so that $N = 2^n$. Therefore, x is between 0 and $2^n - 1$.

Firstly, we need to encode x for quantum computation. We will encode it as the basis vectors. This is called **basis encoding**. Therefore, we only need n-qubit for the encoding. In this process, the index of the target entry, i.e. a, is encoded as the basis vector $|a\rangle$. $|a\rangle$ **is the first important vector in Grover's algorithm**.

Example 23.1 If there are 13 entries to search and the target is at entry 9, how should we encode the problem for quantum computing?

Since there are 13 entries, we can use 4 qubits that have basis vectors from $|0000\rangle$ to $|1111\rangle$. The target is encoded in the basis vector $|1001\rangle$.

The second important vector in Grover's algorithm is the $|a_\perp\rangle$. This is a vector formed by the equal linear superposition of all basis vectors except $|a\rangle$. Therefore, it is orthogonal (perpendicular) to $|a\rangle$ and it has a zero inner product with $|a\rangle$.

$$\langle a_\perp | a \rangle = 0 \tag{23.1}$$

It is expressed as

$$|a_\perp\rangle = \frac{1}{\sqrt{2^n - 1}} \sum_{x=0(x\neq a)}^{2^n-1} |x\rangle \tag{23.2}$$

Note that the coefficient is $\frac{1}{\sqrt{2^n-1}}$ (instead of $\frac{1}{\sqrt{2^n}} = \frac{1}{2^{n/2}}$) because there are only $2^n - 1$ terms after excluding $|a\rangle$.

Figure 23.2 illustrates the Grover's algorithm. Note that $|a\rangle$ is drawn perpendicular to $|a_\perp\rangle$ in the 2D plane.

The third vector that is important to Grover's algorithm is the $|\phi\rangle$ (Fig. 23.2). This vector is the superposition of all basis vectors. Therefore, for an n-qubit problem, it is represented as

$$|\phi\rangle = \frac{1}{2^{\frac{n}{2}}} \sum_{x=0}^{2^n-1} |x\rangle \tag{23.3}$$

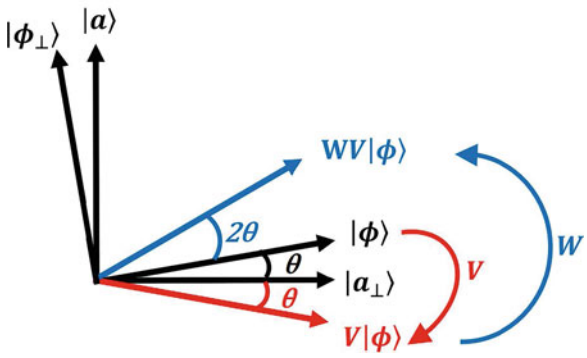


Fig. 23.2 Illustration of the Grover's algorithm. The first two steps are shown. Applying $\boldsymbol{V}$ to $|\phi\rangle$ rotates $|\phi\rangle$ to its mirror image about $|a_\perp\rangle$. Applying $\boldsymbol{W}$ to the new vector rotates it to its mirror image about $|\phi\rangle$. This effectively rotates $|\phi\rangle$ by 2θ closer to the solution $|a\rangle$. This is repeated until it aligns with the solution $|a\rangle$

Again, each x is the decimal form of the basis vector from 0 to $2^n - 1$ and the corresponding basis vectors in binary form are from $|00\cdots00\rangle$ to $|11\cdots11\rangle$ with n qubits.

Note that $|a_\perp\rangle$, $|a\rangle$, and $|\phi\rangle$ are drawn *on the same plane* as illustrated in Fig. 23.2 because $|\phi\rangle$ is a linear combination of $|a_\perp\rangle$ and $|a\rangle$. Therefore, $|\phi\rangle$ lies on the plane formed by $|a_\perp\rangle$ and $|a\rangle$.

The idea in the Grover's algorithm is to reflect $|\phi\rangle$ about $|a_\perp\rangle$ (Fig. 23.2). If the initial angle between $|\phi\rangle$ and $|a_\perp\rangle$ is θ, this is equivalent to rotating $|\phi\rangle$ clockwise by 2θ. We assume this can be accomplished by a matrix $\boldsymbol{V}$. Then we further reflect the new vector ($\boldsymbol{V}|\phi\rangle$) about $|\phi\rangle$. We assume we can find a matrix $\boldsymbol{W}$ to perform this operation. The final vector becomes $\boldsymbol{WV}|\phi\rangle$, which is equivalent to rotating $|\phi\rangle$ by 2θ closer to $|a\rangle$. This process is repeated many times. Each time after $\boldsymbol{WV}$ is applied, the vector will be rotated by 2θ closer to $|a\rangle$ and eventually, we hope that it will align with $|a\rangle$. In this case, the wavefunction contains only $|a\rangle$ and we have found the entry. It is also instructive to say that as the vector is rotated closer and closer to $|a\rangle$, it has a larger and larger component of $|a\rangle$ as the projection to the y-axis is increasing while it has smaller components to the basis vectors that are orthogonal to $|a\rangle$. Therefore, the idea of Grover's algorithm is to begin with a wavefunction in an equal superposition of all basis vectors and then transform it gradually by suppressing the non-solution. Eventually, we have 100% certainty to get our solution when we perform a measurement as the final wavefunction is composed of only $|a\rangle$.

It is obvious that the function $f(x)$ needs to be a part of this algorithm. As you might have guessed, it will be embedded in a quantum oracle. Moreover, the fact (or requirement) that $f(a) = 1$ must be embedded in $\boldsymbol{V}$ because $|a_\perp\rangle$ (which is strongly related to $|a\rangle$) is the vector about which $\boldsymbol{V}$ reflects the vectors. Let us now try to realize $|\phi\rangle$, $\boldsymbol{W}$, and $\boldsymbol{V}$ and understand the meaning of θ.

23.2.4 Implementation of Grover's Algorithm

Creation of $|\phi\rangle$ How do we create an equal superposition of all basis states? Yes, we can use the Hadamard gate. This was derived in Eq. (17.16) and repeated here for convenience.

$$\boldsymbol{H}^{\otimes n}|0\rangle_n = \frac{1}{2^{\frac{n}{2}}}\sum_{x=0}^{2^n-1}|x\rangle = |\phi\rangle \tag{23.4}$$

Therefore, all we need to do is just to initialize the n-qubit system at the ground state $|0\rangle_n$ and then apply an n-qubit Hadamard gate to get $|\phi\rangle$.

Properties of θ and Algorithm Complexity If the problem we care about has many entries (i.e. N is large), then we expect θ to be small. This is because θ is the angle between $|a_\perp\rangle$ (which is a linear superposition of all basis vectors except $|a\rangle$)

and $|\phi\rangle$ (which is a linear superposition of all basis vectors). Their inner product must be close to 1 when the dimension is high as they only have a difference in 1 basis vector among N. This can be shown in this way. We note that the inner product of two *normalized* vectors on a 2D plane is $\cos\theta$ (like Eq. (3.12) but with vector magnitudes being 1). Therefore,

$$\begin{aligned}\cos\theta = \langle\phi|a_\perp\rangle &= \left(\frac{1}{2^{\frac{n}{2}}}\sum_{x=0}^{2^n-1}\langle x|\right)\left(\frac{1}{\sqrt{2^n-1}}\sum_{y=0(y\neq a)}^{2^n-1}|y\rangle\right)\\ &= \frac{1}{2^{\frac{n}{2}}}\frac{1}{\sqrt{2^n-1}}(2^n-1)\\ &= \frac{\sqrt{2^n-1}}{2^{\frac{n}{2}}}\end{aligned} \tag{23.5}$$

Here, I have substituted Eqs. (23.3) and (23.2). I also change the dummy variable from x to y when I use Eq. (23.2). Then, using the orthonormality of the basis vectors, there are only $2^n - 1$ terms left after expanding the first line.

Using the identity, $\sin^2\theta + \cos^2\theta = 1$, we find that

$$\sin\theta = \frac{1}{2^{\frac{n}{2}}} \tag{23.6}$$

When $N = 2^n$ is large, this is a small number. We know that $\sin\theta \approx \theta$ when θ is small. Therefore, for a large N,

$$\theta \approx \frac{1}{2^{\frac{n}{2}}} \tag{23.7}$$

Since θ is small, the angle between $|\phi\rangle$ and $|a\rangle$ is $\pi/2 - \theta \approx \pi/2$. Each $\boldsymbol{WV}$ operation rotates $|\phi\rangle$ by 2θ towards $|a\rangle$. Therefore, the number of $\boldsymbol{WV}$ operation required to rotate $|\phi\rangle$ to $|a\rangle$ is

$$\frac{\frac{\pi}{2}}{2\theta} = \frac{\pi\sqrt{2^n}}{4} = \frac{\pi}{4}\sqrt{N} \tag{23.8}$$

This explains why the complexity of Grover's algorithm is $\mathcal{O}(\sqrt{N})$, where we assume each $\boldsymbol{WV}$ is a significant computation in this problem like the single-digit addition in the addition problem mentioned earlier.

Implementation of $\boldsymbol{V}$ As shown in Fig. 23.2, the $\boldsymbol{V}$ operation reflects $|\phi\rangle$ about $|a_\perp\rangle$. $\boldsymbol{V}$ must be a quantum gate or a set of quantum gates and, thus, it is defined by telling how it transforms basis vectors, $|x\rangle$. The following is the *given* definition:

$$\boldsymbol{V}\,|x\rangle_n = (-1)^{f(x)}\,|x\rangle_n \tag{23.9}$$

This is given. We are not good enough to invent it yet but we are good enough to understand why it works. It is obvious that $\boldsymbol{V}$ keeps all basis vector unchanged except when $x = a$, where $f(a) = 1$, it adds a phase shift of π to $|a\rangle$ (i.e. the coefficient is multiplied by $-1 = e^{i\pi}$). Let us apply $\boldsymbol{V}$ to an arbitrary vector $|\beta\rangle$, which can be $|\phi\rangle$ or the rotated $|\phi\rangle$. $|\beta\rangle$ can be expressed as a linear combination of the basis vectors.

$$|\beta\rangle = \sum_{i=0}^{2^n-1} \beta_i |i\rangle \tag{23.10}$$

where β_i's are the complex coefficients and $|i\rangle$'s are the basis vectors. Then

$$\begin{aligned} \boldsymbol{V}|\beta\rangle &= \boldsymbol{V} \sum_{i=0}^{2^n-1} \beta_i |i\rangle \\ &= \sum_{i=0}^{2^n-1} \beta_i \boldsymbol{V}|i\rangle \\ &= \sum_{i=0}^{2^n-1} (-1)^{f(i)} \beta_i |i\rangle \\ &= (\sum_{i=0, i\neq a}^{2^n-1} \beta_i |i\rangle) - \beta_a |a\rangle \end{aligned} \tag{23.11}$$

Here we used the fact that $f(a) = 1$ and $f(x \neq a) = 0$. *Therefore, the* $\boldsymbol{V}$ *operator keeps all components perpendicular to* $|a\rangle$ *intact but flips the sign of* $|a\rangle$. *This is equivalent to reflecting about* $a_\perp$.

How do we implement $\boldsymbol{V}$ in the form of Eq. (23.9)? You might have found that it is just a phase oracle we discussed before (Eq. (22.4)). Indeed, we will use the quantum oracle to implement $\boldsymbol{V}$. Therefore, $\boldsymbol{V} = \boldsymbol{U}_f$, where $\boldsymbol{U}_f$ is the quantum phase oracle of the problem and we assume it is given and available. *The quantum oracle is assumed to contain the information of* $f(x)$ *and can compute* $f(x)$. *Therefore, it knows the value of* a *that gives* $f(a) = 1$. *Our goal is to query the quantum oracle so that it reveals the value of* a, *i.e. the index of the entry that contains the content we are looking for.*

Implementation of $\boldsymbol{W}$ and the Quantum Oracle The action of $\boldsymbol{W}$ is to reflect any vector on the plane in Fig. 23.2 about $|\phi\rangle$. A vector, $|\gamma\rangle$, on that plane can always be expressed as a linear combination of the two orthonormal vectors, $|\phi\rangle$ and $|\phi_\perp\rangle$. For example,

$$|\gamma\rangle = \gamma_0 |\phi\rangle + \gamma_1 |\phi_\perp\rangle \tag{23.12}$$

where γ_0 and γ_1 are complex coefficients. We expect that after applying $\boldsymbol{W}$, it will become $\gamma_0 |\phi\rangle - \gamma_1 |\phi_\perp\rangle$ because reflection about $|\phi\rangle$ means that the $|\phi\rangle$ coefficient should be unchanged and the $|\phi_\perp\rangle$ coefficient should be negated (gain a phase shift of π). To achieve this, we can set

$$\boldsymbol{W} = 2 |\phi\rangle \langle\phi| - \boldsymbol{I} \tag{23.13}$$

Example 23.2 Shows that $\boldsymbol{W}$ given in Eq. (23.13) reflects $|\gamma\rangle$ about $|\phi\rangle$.

$$\begin{aligned}
\boldsymbol{W} |\gamma\rangle &= (2 |\phi\rangle \langle\phi| - \boldsymbol{I}) |\gamma\rangle \\
&= 2 |\phi\rangle \langle\phi| |\gamma\rangle - \boldsymbol{I} |\gamma\rangle \\
&= 2 |\phi\rangle \langle\phi| (\gamma_0 |\phi\rangle + \gamma_1 |\phi_\perp\rangle) - (\gamma_0 |\phi\rangle + \gamma_1 |\phi_\perp\rangle) \\
&= 2 |\phi\rangle \gamma_0 - (\gamma_0 |\phi\rangle + \gamma_1 |\phi_\perp\rangle) \\
&= \gamma_0 |\phi\rangle - \gamma_1 |\phi_\perp\rangle
\end{aligned} \tag{23.14}$$

We have used the orthonormality that $\langle\phi|\phi_\perp\rangle = 0$ and $\langle\phi|\phi\rangle = 1$.

How do we implement $\boldsymbol{W}$ as quantum gates? Here I will further expand Eq. (23.13) as the following, which will help us construct $\boldsymbol{W}$ using more fundamental gates and we will discuss this in the next chapter.

$$\boldsymbol{W} = \boldsymbol{H}^{\otimes n} (2 |0\rangle_n \langle 0|_n - \boldsymbol{I}) \boldsymbol{H}^{\otimes n} \tag{23.15}$$

This is called the **Grover diffusion operator**.

Example 23.3 Derive Eq. (23.15).

Since $|\phi\rangle = \boldsymbol{H}^{\otimes n} |0\rangle_n$ (Eq. (23.4)), substitute into Eq. (23.13), we have

$$\begin{aligned}
\boldsymbol{W} &= 2 |\phi\rangle \langle\phi| - \boldsymbol{I} \\
&= 2(\boldsymbol{H}^{\otimes n} |0\rangle_n)(\langle 0|_n (\boldsymbol{H}^{\otimes n})^\dagger) - \boldsymbol{I} \\
&= 2(\boldsymbol{H}^{\otimes n} |0\rangle_n)(\langle 0|_n \boldsymbol{H}^{\otimes n}) - \boldsymbol{I}\boldsymbol{I} \\
&= 2(\boldsymbol{H}^{\otimes n} |0\rangle_n)(\langle 0|_n \boldsymbol{H}^{\otimes n}) - \boldsymbol{I}\boldsymbol{H}^{\otimes n}\boldsymbol{H}^{\otimes n} \\
&= 2(\boldsymbol{H}^{\otimes n} |0\rangle_n)(\langle 0|_n \boldsymbol{H}^{\otimes n}) - \boldsymbol{H}^{\otimes n}\boldsymbol{I}\boldsymbol{H}^{\otimes n} \\
&= \boldsymbol{H}^{\otimes n} (2 |0\rangle_n \langle 0|_n - \boldsymbol{I}) \boldsymbol{H}^{\otimes n}
\end{aligned} \tag{23.16}$$

Here we have used many important linear algebra properties we have learned in this book. The second line uses the property that the *bra* version of $\boldsymbol{H}^{\otimes n} |0\rangle_n$ is $\langle 0|_n (\boldsymbol{H}^{\otimes n})^\dagger$. The third line uses the property that $\boldsymbol{H}$ is Hermitian, so $\boldsymbol{H} = \boldsymbol{H}^\dagger$. The fourth line uses the property of the Hadamard gate that $\boldsymbol{H}\boldsymbol{H} = \boldsymbol{I}$.

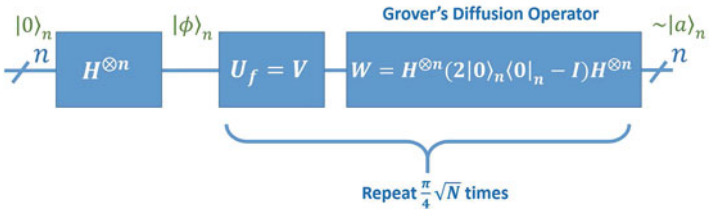


Fig. 23.3 Circuit to implement the Grover's algorithm when a quantum phase oracle, $\boldsymbol{U}_f$, is available

Again we express the *Grover diffusion operator* in the form of Eq. (23.15) is just to prepare ourselves to implement it using more fundamental quantum gates in the next chapter.

23.2.5 Circuit for Grover's Algorithm

By combining what we have learned, Fig. 23.3 shows the circuit for Grover's algorithm when a quantum phase oracle $\boldsymbol{U}_f$ is available. Although it looks sophisticated that we need to implement $\boldsymbol{WV}$ many times, this is actually not a problem for a quantum circuit. As discussed earlier, unlike in a classical circuit, the horizontal direction in a quantum circuit represents the flow of time instead of the space and a quantum gate is nothing but just a sequence of pulses (e.g. microwave or laser). The repetition just refers to the application of pulses multiple times to the same atoms or electrons. Therefore, even $N = 10^{12}$, we just need to apply the sequence of pulses corresponding to $\boldsymbol{WV}$ for 10^6 times instead of having 10^6 gates in the circuit.

23.3 Summary

In this chapter, we have learned Grover's algorithm. But more importantly, through this process, we have learned that we need to encode a problem in a quantum computing algorithm. Here we use the *basis encoding* so that each basis vector represents a value of a variable (i.e. entry index x). We also understand that Grover's algorithm can only provide *quadratic speedup* over the classical one but can be very useful already when the database is large. We find that there are three important vectors ($|\phi\rangle$, $|a\rangle$, and $|a\perp\rangle$) and two matrices ($\boldsymbol{W}$ and $\boldsymbol{V}$) that play important roles in the algorithm. We already know how to implement all of them. In the next chapter, we will try using numerical examples and run on IBM-Q.

Problems

23.1 Rotation of $|\phi\rangle$
Make a table to show the angles between the rotated $|\phi\rangle$ and $|a\rangle$ and between the rotated $|\phi\rangle$ and $|a_\perp\rangle$ after applying $\boldsymbol{V}$, $\boldsymbol{WV}$, $\boldsymbol{VWV}$, $\boldsymbol{WVWV}$ ...

23.2 Another form of $\boldsymbol{V}$
Show that $\boldsymbol{V} = \boldsymbol{I} - 2\,|a\rangle\,\langle a|$. Compare this to $\boldsymbol{W}$ in Eq. (23.13).

23.3 Numerical Substitution I
If $n = 2$ and $a = 3$, find $|\phi\rangle$, $|a\rangle$, and $|a_\perp\rangle$.

23.4 Numerical Substitution II
Continue from Problem 23.3, express $\boldsymbol{V}\,|\phi\rangle$ as a linear combination of $|\phi\rangle$ and $|\phi_\perp\rangle$.

23.5 Computational Complexity
If algorithm A has a complexity of $\mathcal{O}(e^N)$ and algorithm B has a complexity of $\mathcal{O}(2N)$. What type of speedup does algorithm B have over algorithm A?

Chapter 24
Grover's Algorithm: II

24.1 Learning Outcomes

Gain a deeper understanding of Grover's algorithm; Able to perform numerical substitutions; Able to construct the algorithm on IBM-Q and analyze the data; Able to contrast the difference between a phase quantum oracle and an XOR quantum oracle in Grover's algorithm and how they affect the quantum circuit construction.

24.2 Numerical Example for Grover's Algorithm

Assume there is a database with four entries. Therefore, it has indices, 0, 1, 2, and 3. It is unstructured and we do not know where what we are looking for is stored. Assume it is stored in entry 1, i.e. index $a = 1$. I assume there is a phase quantum oracle, $\boldsymbol{U_f}$, corresponding to $f(x = 1) = 1$ and $f(x \neq 1) = 0$ available. Let us construct Grover's algorithm using numerical substitution.

Firstly, $N = 4$ in this problem. Therefore, $n = 2$ as $2^2 = 4$. So we only need 2 qubits. As $a = 1$, therefore,

$$|a\rangle = |01\rangle \tag{24.1}$$

For $|a_\perp\rangle$, which is a superposition of all basis vectors but excluding $|a\rangle = |01\rangle$, based on Eq. (23.2), it is

$$\begin{aligned} |a_\perp\rangle &= \frac{1}{\sqrt{2^n - 1}} \sum_{x=0(x\neq a)}^{2^n-1} |x\rangle \\ &= \frac{1}{\sqrt{2^2 - 1}} (|00\rangle + |10\rangle + |11\rangle) \end{aligned}$$

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_24

$$= \frac{1}{\sqrt{3}}(|00\rangle + |10\rangle + |11\rangle) \tag{24.2}$$

For $|\phi\rangle$, which is a superposition of all basis vectors, based on Eq. (23.3), it is

$$\begin{aligned} |\phi\rangle &= \frac{1}{2^{\frac{n}{2}}} \sum_{x=0}^{2^n-1} |x\rangle \\ &= \frac{1}{2^{\frac{2}{2}}}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \\ &= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \end{aligned} \tag{24.3}$$

We actually do not need to construct these to emulate Grover's algorithm. But these help us gain a deeper understanding and we will also use them to check against the matrix multiplication results.

24.2.1 Construction of Quantum Oracle

In reality, we do not know the structure of the quantum oracle, $\boldsymbol{U}_f$. It will be realized through some physical systems. But it should behave as expressed in Eq. (23.9). We are supposed to have no knowledge of the value of a. But to construct the oracle for calculation and simulation, we need to use the knowledge in this example that $a = 1$. Based on Eq. (23.9), we have

$$\boldsymbol{U}_f = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{24.4}$$

This is because $|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$, $|01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$, $|10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$, and $|11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$ and particularly, $\boldsymbol{U}_f |01\rangle = (-1)^{f(1)} |01\rangle = (-1)^1 |01\rangle = -|01\rangle$.

I can also construct this matrix using Eq. (22.6) by substituting Eq. (23.9). I will leave this as an exercise for you.

And of course, $\boldsymbol{V} = \boldsymbol{U}_f$. I want to further decompose $\boldsymbol{U}_f$ to more elementary gates. $\boldsymbol{U}_f$ looks like a controlled-Z-gate, $\boldsymbol{CZ}$, (Problem 16.2, which is also a controlled phase shift gate with phase shift being π, $\boldsymbol{U}_{\boldsymbol{CPS},\boldsymbol{\pi}}$) except that it gains a negative sign when the MSB is 0 and the LSB is 1 instead of when both are 1. In this

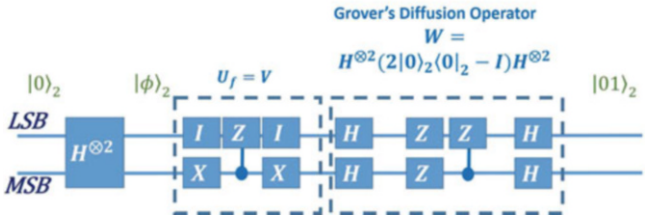


Fig. 24.1 Circuit to implement the Grover's algorithm for the given example

case, I can perform a NOT operation on the MSB first and then apply a controlled-Z-gate. After that, I will apply a NOT gate again to bring the basis vectors back to themselves (Fig. 24.1). Therefore,

$$\boldsymbol{U_f = (U_{NOT} \otimes I) U_{CPS,\pi} (U_{NOT} \otimes I)} \tag{24.5}$$

Example 24.1 Prove Eq. (24.5)

$$\begin{aligned}
&(\boldsymbol{U_{NOT} \otimes I}) \boldsymbol{U_{CPS,\pi}} (\boldsymbol{U_{NOT} \otimes I}) \\
&= \left(\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right) \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \left(\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right) \\
&= \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \\
&= \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \boldsymbol{U_f}
\end{aligned} \tag{24.6}$$

You can see that the rightmost NOT gate in the equation (which will be the leftmost NOT gate in the circuit) shuffles the basis vectors so that $|01\rangle$ can get the negative sign from the $\boldsymbol{CZ}$ gate in line 3 (therefore, in line 3, I performed the

multiplication of the rightmost 2 matrices first). The last NOT gate then shuffles the basis vector components back to the original position (lines 4 and 5).

24.2.2 Construction of the Grover Diffusion Operator

In Eqs. (23.15) and (23.16), we show that the Grover diffusion operator can be expressed as

$$\boldsymbol{W} = \boldsymbol{H}^{\otimes n}(2\,|0\rangle_n\,\langle 0|_n - \boldsymbol{I})\boldsymbol{H}^{\otimes n} = \boldsymbol{H}^{\otimes 2}(2\,|0\rangle_2\,\langle 0|_2 - \boldsymbol{I})\boldsymbol{H}^{\otimes 2} \tag{24.7}$$

It is easy to implement the leftmost and rightmost Hadamard gates, which are just

$$\begin{aligned}\boldsymbol{H}^{\otimes 2} &= \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \otimes \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \\ &= \frac{1}{2}\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & -1 & 1 & -1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\end{pmatrix}\end{aligned} \tag{24.8}$$

Example 24.2 Find the matrix for $2\,|0\rangle_2\,\langle 0|_2 - \boldsymbol{I}$.

$$\begin{aligned}2\,|0\rangle_2\,\langle 0|_2 - \boldsymbol{I} &= 2\begin{pmatrix}1\\0\\0\\0\end{pmatrix}\begin{pmatrix}1 & 0 & 0 & 0\end{pmatrix} - \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix} \\ &= \begin{pmatrix}2 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{pmatrix} - \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix} \\ &= \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & -1 & 0 & 0\\ 0 & 0 & -1 & 0\\ 0 & 0 & 0 & -1\end{pmatrix}\end{aligned} \tag{24.9}$$

This is enough for numerical substitution. But to implement it using more elementary quantum gates, we need to further decompose it. To get three -1's and one 1 in the diagonal, one possibility is to use the tensor product of two $\boldsymbol{Z}$ gates (which will give us two -1's) and then apply $\boldsymbol{CZ} = \boldsymbol{U}_{\boldsymbol{CPS},\boldsymbol{\pi}}$ gate again to get another -1. Therefore, $2\,|0\rangle_2\,\langle 0|_2 - \boldsymbol{I} = \boldsymbol{U}_{\boldsymbol{CPS},\boldsymbol{\pi}}(\boldsymbol{Z} \otimes \boldsymbol{Z})$ (Fig. 24.1).

Example 24.3 Show that $2\,|0\rangle_2\,\langle 0|_2 - \boldsymbol{I} = \boldsymbol{U_{CPS,\pi}}(\boldsymbol{Z}\otimes\boldsymbol{Z})$.

$$\begin{aligned}\boldsymbol{U_{CPS,\pi}}(\boldsymbol{Z}\otimes\boldsymbol{Z}) &= \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&-1\end{pmatrix}\left(\begin{pmatrix}1&0\\0&-1\end{pmatrix}\otimes\begin{pmatrix}1&0\\0&-1\end{pmatrix}\right)\\ &= \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&-1\end{pmatrix}\begin{pmatrix}1&0&0&0\\0&-1&0&0\\0&0&-1&0\\0&0&0&1\end{pmatrix}\\ &= \begin{pmatrix}1&0&0&0\\0&-1&0&0\\0&0&-1&0\\0&0&0&-1\end{pmatrix} = 2\,|0\rangle_2\,\langle 0|_2 - \boldsymbol{I}\end{aligned} \tag{24.10}$$

24.2.3 Evolution of the Wavefunction

Let us now use matrix multiplications to see how the wavefunction evolves in Fig. 24.1.

The wavefunction is first initialized to $|0\rangle_2 = |00\rangle = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}$ at the far left. After passing through the Hadamard gate, it becomes

$$\begin{aligned}\boldsymbol{H}^{\otimes 2}\,|00\rangle &= \frac{1}{2}\begin{pmatrix}1&1&1&1\\1&-1&1&-1\\1&1&-1&-1\\1&-1&-1&1\end{pmatrix}\begin{pmatrix}1\\0\\0\\0\end{pmatrix}\\ &= \frac{1}{2}\begin{pmatrix}1\\1\\1\\1\end{pmatrix} = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)\end{aligned} \tag{24.11}$$

This is just the $|\phi\rangle$ in Eq. (24.3). Then it goes through the quantum oracle (i.e. $\boldsymbol{V}$ in Eq. (24.6)) and it becomes

$$\begin{aligned}\boldsymbol{V}\,|\phi\rangle &= \boldsymbol{U}_f\left(\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)\right)\\ &= \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & -1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}\frac{1}{2}\begin{pmatrix}1\\1\\1\\1\end{pmatrix}\\ &= \frac{1}{2}\begin{pmatrix}1\\-1\\1\\1\end{pmatrix}\end{aligned} \tag{24.12}$$

It is clear that it flips the sign of $|a\rangle = |01\rangle$ while keeping other basis vectors intact.

We then apply the Grover diffusion operator using Eqs. (24.8) and (24.10),

$$\begin{aligned}&\boldsymbol{H}^{\otimes 2}(2\,|0\rangle_2\,\langle 0|_2 - \boldsymbol{I})\boldsymbol{H}^{\otimes 2}(\boldsymbol{V}\,|\phi\rangle)\\ &= \frac{1}{2}\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & -1 & 1 & -1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & -1 & 0 & 0\\ 0 & 0 & -1 & 0\\ 0 & 0 & 0 & -1\end{pmatrix}\frac{1}{2}\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & -1 & 1 & -1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\end{pmatrix}\frac{1}{2}\begin{pmatrix}1\\-1\\1\\1\end{pmatrix}\\ &= \frac{1}{4}\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & -1 & 1 & -1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\end{pmatrix}\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & -1 & 0 & 0\\ 0 & 0 & -1 & 0\\ 0 & 0 & 0 & -1\end{pmatrix}\begin{pmatrix}1\\1\\-1\\1\end{pmatrix}\\ &= \frac{1}{4}\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & -1 & 1 & -1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\end{pmatrix}\begin{pmatrix}1\\-1\\1\\-1\end{pmatrix}\\ &= \begin{pmatrix}0\\1\\0\\0\end{pmatrix} = |01\rangle = |a\rangle\end{aligned} \tag{24.13}$$

Interestingly, we have achieved the solution by applying $\boldsymbol{W}\boldsymbol{V}$ only *one time* instead of the approximately 2 times predicted by Eq. (23.8). This is because N is small here, and it is no longer valid to ignore θ in the calculation of the number of $\boldsymbol{W}\boldsymbol{V}$ operations required. That is, $\pi/2 - \theta \approx \pi/2$ is no longer valid.

Let us find θ using Eqs. (24.2) and (24.3) and the orthonormal property of the basis vectors.

$$\begin{aligned}\cos\theta &= \langle a_\perp|\phi\rangle \\ &= \frac{1}{\sqrt{3}}(\langle 00| + \langle 10| + \langle 11|)\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \\ &= \frac{\sqrt{3}}{2}\end{aligned} \tag{24.14}$$

Therefore, $\theta = \pi/6$. One $\boldsymbol{WV}$ operation will rotate it by $2\theta = \pi/3$ and it will be aligned with $|a\rangle$ because $\pi/6 + \pi/3 = \pi/2$ (see Fig. 23.2).

24.3 Simulation on IBM-Q

The implementation of the circuit in Fig. 24.1 in IBM-Q is straightforward as I have already made the bottom qubit the MSB. The difficult part is to implement the $\boldsymbol{CZ}$ gate because IBM-Q does not have the gate handy (in IBM Quantum Composer). But as discussed before, $\boldsymbol{CZ}$ is just a CNOT-gate, $\boldsymbol{U_{XOR}}$, in the $|+\rangle$ / $|-\rangle$ basis (see Chap. 16). Therefore, we only need to apply $\boldsymbol{H}$ before and after a $\boldsymbol{U_{XOR}}$ gate to emulate a $\boldsymbol{CZ}$ in the $|0\rangle$ / $|1\rangle$ basis. We have,

$$\boldsymbol{CZ} = (\boldsymbol{I} \otimes \boldsymbol{H})\boldsymbol{U_{XOR}}(\boldsymbol{I} \otimes \boldsymbol{H}) \tag{24.15}$$

We will prove this in the Problems. Figure 24.2 shows the implementation and hardware execution results on IBM-Q. We can see that most of the time it gives $|01\rangle$, which is the desired solution. However, due to errors, it also has a substantial probability of showing $|00\rangle$ and other values. If you submit it to a simulator, it will give only $|01\rangle$, which confirms our derivation in Eq. (24.13).

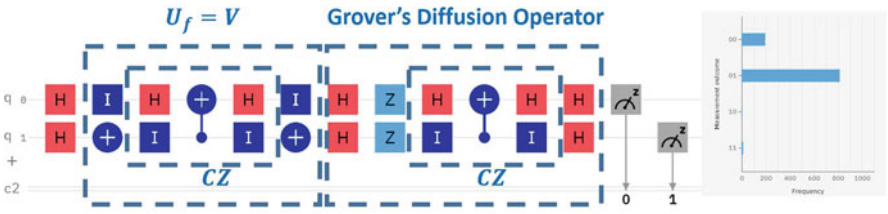


Fig. 24.2 Circuit corresponds to Fig. 24.1 implemented in IBM-Q. The hardware result is shown on the right

24.4 Implementation Using XOR Quantum Oracle

So far, we have been assuming that the quantum oracle is given as a phase oracle. It is also possible to implement Grover's algorithm using XOR quantum oracle. Since it is given by a physical system, we do not need to worry about it (except when we want to simulate it, then we need to create it ourselves like what we did for the phase quantum oracle). However, we need to think of how to incorporate the XOR quantum oracle in $\boldsymbol{V}$. This is because we are relying on using the oracle to implement $\boldsymbol{V}$ to make Grover's algorithm works.

Firstly, the XOR quantum oracle is a quantum gate of $n+m$ qubits, where n is the number of qubits we need to encode the index of the database (i.e. $2^n = N$) and m is the number of the auxiliary qubit to encode the output $f(x)$. Since $f(x)$ is either 0 or 1, we only need 1-qubit and thus $m = 1$. For convenience, the definition of an XOR quantum oracle in Eq. (22.1) is repeated here

$$\boldsymbol{U}_f \left|x\right\rangle_n \left|y\right\rangle = \left|x\right\rangle_n \left|y \oplus f(x)\right\rangle \tag{24.16}$$

where we have used $m = 1$. Therefore, the oracle has $n+1$ qubits. In order to use the oracle in $\boldsymbol{V}$, we need to apply $|-\rangle$ to the input of the $m = 1$ auxiliary qubit. To obtain $|-\rangle$, we can apply a $\boldsymbol{U_{NOT}}$ gate followed by a $\boldsymbol{H}$ gate to the initial $|0\rangle$ state of the auxiliary qubit (Fig. 24.3). The following proves why it needs to be done in this way.

For any basis vector $|x\rangle_n$, it combines (performs tensor product with) the auxiliary qubit as the input to the oracle. Therefore,

$$\begin{aligned}
\boldsymbol{U}_f(\left|x\right\rangle_n \left|-\right\rangle) &= \boldsymbol{U}_f(\left|x\right\rangle_n \frac{1}{\sqrt{2}}(\left|0\right\rangle - \left|1\right\rangle)) \\
&= \frac{1}{\sqrt{2}}(\boldsymbol{U}_f(\left|x\right\rangle_n \left|0\right\rangle) - \boldsymbol{U}_f(\left|x\right\rangle_n \left|1\right\rangle)) \\
&= \frac{1}{\sqrt{2}}(\left|x\right\rangle_n \left|f(x) \oplus 0\right\rangle - \left|x\right\rangle_n \left|f(x) \oplus 1\right\rangle) \\
&= \frac{1}{\sqrt{2}}(\left|x\right\rangle_n \left|f(x)\right\rangle - \left|x\right\rangle_n \left|f(x) \oplus 1\right\rangle)
\end{aligned} \tag{24.17}$$

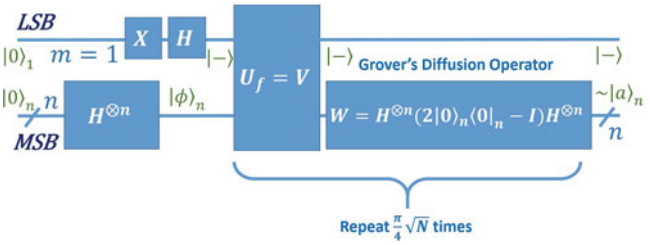


Fig. 24.3 Circuit to implement the Grover's algorithm with a XOR quantum oracle, $\boldsymbol{U}_f$ (instead of using the phase oracle in Fig. 23.3)

Here we have used the distribution and linear properties of operators and the definition of an XOR quantum oracle.

If $f(x) = 0$, then $f(x) \oplus 1 = 1$ and $(-1)^{f(x)} = 1$. Therefore, continuing from Eq. (24.17)

$$\begin{aligned} \boldsymbol{U}_f(|x\rangle_n |-\rangle) &= \frac{1}{\sqrt{2}}(|x\rangle_n |0\rangle - |x\rangle_n |1\rangle) \\ &= |x\rangle_n \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ &= |x\rangle_n |-\rangle \\ &= (-1)^{f(x)} |x\rangle_n |-\rangle \end{aligned} \tag{24.18}$$

If $f(x) = 1$, then $f(x) \oplus 1 = 0$ and $(-1)^{f(x)} = -1$. Therefore, continuing from Eq. (24.17)

$$\begin{aligned} \boldsymbol{U}_f(|x\rangle_n |-\rangle) &= \frac{1}{\sqrt{2}}(|x\rangle_n |1\rangle - |x\rangle_n |0\rangle) \\ &= -|x\rangle_n \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ &= -|x\rangle_n |-\rangle \\ &= (-1)^{f(x)} |x\rangle_n |-\rangle \end{aligned} \tag{24.19}$$

Combining Eqs. (24.18) and (24.19), regardless of the value of $f(x)$, we have

$$\begin{aligned} \boldsymbol{U}_f(|x\rangle_n |-\rangle) &= (-1)^{f(x)} |x\rangle_n |-\rangle \\ &= (\boldsymbol{V} |x\rangle_n) \otimes (\boldsymbol{I} |-\rangle) \\ &= \boldsymbol{V} \otimes \boldsymbol{I} |x\rangle_n |-\rangle \end{aligned} \tag{24.20}$$

This means that, by adding the auxiliary bit with $|-\rangle$ as the input value, the XOR quantum oracle behaves exactly as a phase quantum oracle on the n query qubits. Therefore, Fig. 24.3 behaves exactly the same as Fig. 23.3 for the n MSB, which we care about. The output of the auxiliary qubit is unchanged (still $|-\rangle$). Therefore, we can repeat the $\boldsymbol{W}\boldsymbol{V}$ as before by about $\sqrt{N}$ times to find the solution.

24.5 Summary

In this chapter, we have performed numerical substitution in an example for Grover's algorithm. We also implemented the circuit in IBM-Q. By tracing the wavefunction (state vector) evolution throughout the quantum circuit, we developed

a deeper insight into how Grover's algorithm works. We also learned how to modify the circuit when an XOR quantum oracle is used instead of a phase quantum oracle. I encourage you to perform numerical substitution as we have done here when you encounter a new algorithm. We also gained a better understanding of the meaning of oracle and its role. Literally, a quantum computing algorithm is to query the oracle in a smart way to extract the desired information. In this example, it evolves the equal superposition wavefunction to only $|a\rangle$ to extract the index a, which has the property $f(a) = 1$ known by the oracle.

Problems

24.1 Construction of Oracle
Derive Eq. (24.4) using Eq. (22.6) by substituting Eq. (23.9).

24.2 Rotation of $\boldsymbol{VW}\,|\phi\rangle$
Perform another $\boldsymbol{WV}$ operation to the wavefunction obtained in Eq. (24.13). Is it over-rotated?

24.3 Numerical Substitution of Grover's Algorithm
Repeat what we done in the numerical substitution with $a = 3$.

24.4 Construction of $\boldsymbol{CZ}$ Gate
Using matrix multiplication, prove that the circuit in the $\boldsymbol{CZ}$ block in Fig. 24.2 indeed works as a $\boldsymbol{CZ}$ gate.

24.5 Write your own Grover's Algorithm Simulator
In the numerical substitution part, we have derived the matrices for all gates. Please use Google Colab to write a simulator to simulate the quantum circuit. The minimal function of your simulator should perform the matrix multiplications and give the correct output wavefunction. You may also add measurement using a random variable to see the statistics (although if you do it right, it always gives $|01\rangle$).

24.6 Implement Grover's Algorithm with XOR Gate
Implement the circuit in Fig. 24.3 on IBM-Q. You need to first construct the corresponding XOR oracle. Set $a = 1$. Note that this is a 3-qubit circuit due to the addition of the auxiliary bit.

Chapter 25
Quantum Fourier Transform I

25.1 Learning Outcomes

Able to describe some important identities of the N-th root of unity; can describe the differences and similarities between Discrete Fourier Transform and Quantum Fourier Transform; able to describe inverse Quantum Fourier Transform; understand why Quantum Fourier Transform gate is unitary and symmetric.

25.2 The N-th Root of Unity

The purpose of this section is to review some of the important properties and identities of the complex numbers, particularly those related to the **N-th root of unity**. This is a very important warm-up to understand the **Quantum Fourier Transform (QFT)**.

A complex number, c, has two components, namely the real and the imaginary parts. For example, $c = a + bi = 1.5 + 1.4i$, in which a and b are the real numbers and $a = 1.5$, $b = 1.4$, and $i = \sqrt{-1}$. This is equivalent to a 2D real space vector, and we can draw it on a 2D plane with the imaginary (real) part as the vertical (horizontal) axis (Fig. 25.1). Each point on the plane represents a complex number (or a vector pointing from the origin to that point). But I need to remind you that *this is NOT a quantum state vector*, and it is still just a number (although it is a complex number and it is equivalent to a 2D real space vector). It is used as the coefficients for the state vectors. Remembering that we even use Pauli matrices as the coefficients of a Pauli vector in Eq. (7.15), it is not surprising that we can use a 2D real space vector (equivalent to a complex number) as the coefficients for the quantum state vector.

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_25

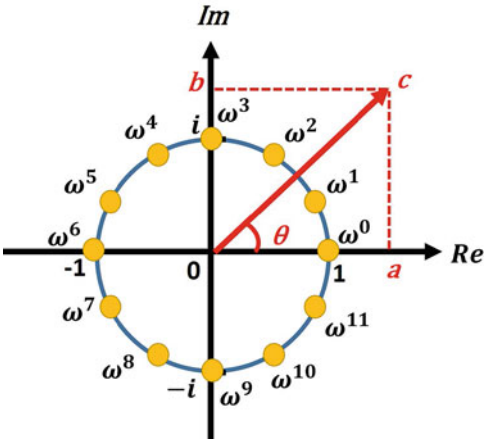


Fig. 25.1 Illustration of the complex plane and the N-th root of unity with $N = 12$

We may also represent a complex number using the length of the vector, $|c|$, and its angle to the positive real axis, θ. That is

$$c = |c|e^{i\theta} = |c|(\cos\theta + i\sin\theta) \tag{25.1}$$

There is a special set of complex numbers with lengths equal to 1, and they lie on the unit circle in Fig. 25.1. And they can be expressed simply as $e^{i\theta} = \cos\theta + i\sin\theta$ because $|c| = 1$. Among them, some of them (yellow dots), z, are even more special because they are the N-th roots of unity, which means

$$z^N = 1 \tag{25.2}$$

We can easily check that $z = e^{i2\pi m/N}$, with m being an integer, satisfies Eq. (25.2) because

$$z^N = (e^{i2\pi m/N})^N = e^{i2\pi m} = \cos 2\pi m + i\sin 2\pi m = 1 \tag{25.3}$$

The N-th roots of unity reside and spread evenly on the unit circle. We define $\omega = e^{i2\pi/N}$, and all can be expressed as ω^m, where m runs from 0 to $N-1$. For example, when $m = 0$, $\omega^0 = 1$. Figure 25.1 shows the distribution of the N-th roots of unity on the unit circle when $N = 12$.

There are three properties we want to review. First, as m increases, ω^m moves counter-clockwise, and eventually, it repeats after one cycle (corresponds to increasing the exponent by N). This can be seen from the picture or proved using

$$\omega^{m+N} = e^{i2\pi(m+N)/N} = e^{i2\pi m/N}e^{i2\pi N/N} = e^{i2\pi m/N} = \omega^m \tag{25.4}$$

The second property is that the sum of all N-th roots of unity for a given N is zero. This can be seen from the picture again. Since they are all evenly and

symmetrically distributed, their corresponding vectors sum to null. This can also be proved as

$$\sum_{m=0}^{N-1} \omega^m = \omega^0 + \omega^1 + \cdots + \omega^{N-1}$$
$$= \frac{\omega^0 - \omega^N}{\omega^0 - \omega} = \frac{1-1}{1-\omega} = 0 \tag{25.5}$$

Here we have used the summation equation of a geometric series with the first term being ω^0, the ratio being ω, and the number of terms being N. We also used Eq. (25.4) that $\omega^N = \omega^{0+N} = \omega^0$.

Similarly, for any integer $q \neq 0$,

$$\sum_{m=0}^{N-1} \omega^{mq} = \omega^{0q} + \omega^{1q} + \cdots + \omega^{(N-1)q}$$
$$= \frac{\omega^{0q} - (\omega^q)^N}{\omega^{0q} - \omega^q} = \frac{1-1}{1-\omega^q} = 0 \tag{25.6}$$

For this geometric series, the first term is ω^{0q}, the ratio is ω^q, and the number of terms is N. $(\omega^q)^N = 1$ is just because $(\omega^q)^N = (\omega^N)^q = 1^q = 1$. Equations (25.5) and (25.6) represent the **destructive interference** we see in many quantum algorithms.

However, if $q = 0$,

$$\sum_{m=0}^{N-1} \omega^{mq} = \sum_{m=0}^{N-1} \omega^0 = \sum_{m=0}^{N-1} 1 = N \tag{25.7}$$

because there are N 1's. This represents the **constructive interference** we see in many quantum algorithms.

The final property is that $\omega^{N-m} = \omega^{-m}$. This can be seen pictorially that going counter-clockwise by $N - m$ steps from the positive real axis is the same as going clockwise by m steps. It can be proved as

$$\omega^{N-m} = e^{i2\pi(N-m)/N} = e^{i2\pi N/N} e^{i2\pi(-m/N)} = e^{i2\pi(-m/N)} = \omega^{-m} \tag{25.8}$$

25.3 Discrete Fourier Transform

Discrete Fourier Transform (DFT) is the transformation (or mapping) of a discrete data series (*perceived* as in one domain e.g. time) into another data series (*perceived* as in another domain e.g. frequency). For example, if you sample the voltages

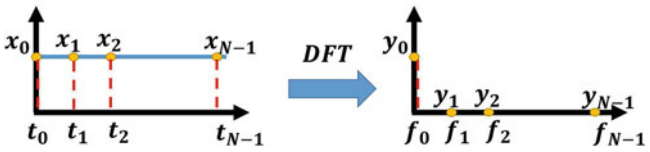


Fig. 25.2 Illustration of Discrete Fourier Transform

of an electrical signal throughout the time, you will get a series of data (a series of voltages). By performing a DFT on the voltage series, you will get another series of data corresponding to the strength of the signal due to various frequency components. Both data series contain the same amount of information.

Figure 25.2 shows an example. The signal is constant (so it is a DC voltage). It is sampled at different times, from t_0 to t_{N-1}, and the sampled values form a series of numbers, $\{x_0, x_1, \ldots, x_{N-1}\}$. After DFT, this series is transformed into another series of numbers $\{y_0, y_1, \ldots, y_{N-1}\}$. In this particular case, each y_i represents the strength of various frequency components from f_0 to f_{N-1}. Since it is a DC signal, only $f_0 = 0\,\text{Hz}$ has a non-zero value as expected (meaning that it has no contribution due to any AC components). We can say that the t_i's and the f_i's form two sets of basis vectors with x_i and y_i being their "magnitudes," respectively. They are just like the $\hat{x}$, $\hat{y}$, and $\hat{z}$ in the 3D real space or the "1-egg" or "1-dozen egg" basis vectors we mentioned before. In this application, they are perceived to be two different sets of basis (i.e. time vs. frequency). But mathematically, they can be the same set of basis vectors because, as you will see below, they (the basis vectors) do not appear in any equations except they mandate the locations of x_i and y_i. This is expected because, as we have been emphasized many times, the exact content of the basis vector is not important to the operations after we have chosen the basis.

Therefore, we can also make the series to be a vector and let them share the same basis. We then have $\vec{X} = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{pmatrix}$ and $\vec{Y} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{pmatrix}$. The DFT is defined as

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \omega^{-kj} x_j = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{-i2\pi kj/N} x_j \tag{25.9}$$

where $\omega = e^{i2\pi/N}$ is the N-th root of unity. Note that the dimensions of the series/vectors are N. We see that the new series is just a *linear combination* of the old series weighted by the N-th roots of unity.

This can also be represented in a matrix form as

$$\vec{Y} = DFT\{\vec{X}\}$$
$$\vec{Y} = \boldsymbol{\Omega}\vec{X}$$
$$\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{pmatrix} = \frac{1}{\sqrt{N}} \begin{pmatrix} \omega^{-0\cdot 0} & \omega^{-0\cdot 1} & \cdots & \omega^{-0\cdot(N-1)} \\ \omega^{-1\cdot 0} & \omega^{-1\cdot 1} & \cdots & \omega^{-1\cdot(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \omega^{-(N-1)\cdot 0} & \omega^{-(N-1)\cdot 1} & \cdots & \omega^{-(N-1)\cdot(N-1)} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{pmatrix} \tag{25.10}$$

We see that each column and row in the DFT matrix are composed of a power series of the N-th roots of unity. In particular, the component of the series after DFT (i.e. y_k) is the sum of a power series of the N-th roots of unity (i.e. ω^{-kj}) weighted by the components of the input series (i.e. x_j). If x_j is constant, then it is just purely a sum of a power series of the N-th roots of unity scaled by the x_j.

Example 25.1 Find $\vec{Y}$ if $\vec{X}$ is a normalized constant vector.

First, $\vec{X} = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$ if it is a normalized constant vector. Each component has a magnitude of $\frac{1}{\sqrt{N}}$; therefore, its total length is $\sqrt{N(\frac{1}{\sqrt{N}})^2} = 1$. This means that $x_j = \frac{1}{\sqrt{N}}$.

We can find y_k using Eq. (25.9), which is repeated here for clarity.

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \omega^{-kj} x_j = \frac{1}{N} \sum_{j=0}^{N-1} \omega^{-kj} \tag{25.11}$$

But we already know the answer. For $k = 0$, we can apply Eq. (25.7) to the summation and $y_0 = \frac{N}{N} = 1$. This is due to **constructive interference**. For $k \neq 0$, we apply Eq. (25.6) and it becomes 0. This is due to **destructive interference**.

Therefore, $\vec{Y} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$, which is also a *normalized vector*.

From this example, we appreciate the power of DFT that *it is constructed in a way to perform constructive and destructive interferences by arranging the power series of the N-th roots of unity in the rows and columns*.

25.4 Quantum Fourier Transform

I spent a lot of time discussing the N-th root of unity and DFT. This is essential because the Quantum Fourier Transform (QFT) becomes very trivial after that. In DFT, we emphasized that the basis vectors of the input and output vectors (or series) can be the same. In engineering, we usually perceive them in different domains (e.g. time vs. frequency), but this is not necessary. If they are treated the same, DFT is just a rotation of the vector on a given basis.

If we decide to use quantum basis states ($|j\rangle$) *as the basis vectors, we can use DFT as usual. And this is called QFT.*

Of course, QFT is nothing but a quantum gate, and, thus, it needs to obey the properties of a quantum gate, namely, being unitary.

In an N-dimensional space, and for convenient, we can assume $N = 2^n$ where n is the number of qubits, and the matrix of a QFT gate, $\boldsymbol{U_{QFT}}$, is defined as (same as the $\boldsymbol{\Omega}$ in Eq. (25.10))

$$\boldsymbol{U_{QFT}} = \frac{1}{\sqrt{N}} \begin{pmatrix} \omega^{-0\cdot 0} & \omega^{-0\cdot 1} & \cdots & \omega^{-0\cdot(N-1)} \\ \omega^{-1\cdot 0} & \omega^{-1\cdot 1} & \cdots & \omega^{-1\cdot(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \omega^{-(N-1)\cdot 0} & \omega^{-(N-1)\cdot 1} & \cdots & \omega^{-(N-1)\cdot(N-1)} \end{pmatrix} \tag{25.12}$$

Note that $\boldsymbol{U_{QFT}}$ is a **symmetric matrix**. This is because $\boldsymbol{U_{QFT,ij}} = \boldsymbol{U_{QFT,ji}}$. For example, $\boldsymbol{U_{QFT,01}} = \omega^{-0\cdot 1} = \omega^{-1\cdot 0} = \boldsymbol{U_{QFT,10}}$.

Another equivalent way to define $\boldsymbol{U_{QFT}}$ is to define it by showing how it transforms the basis vectors like how we have been doing to other quantum gates. Based on Eq. (25.12), for basis vectors $|j\rangle$ and $|k\rangle$,

$$\boldsymbol{U_{QFT}}\,|j\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega^{-kj}\,|k\rangle \tag{25.13}$$

This can be checked easily using the fact that the k-th row and j-th column of $\boldsymbol{U_{QFT}}$ are given by $\langle k|\,\boldsymbol{U_{QFT}}\,|j\rangle$ (Eq. (12.17)),

$$\boldsymbol{U_{QFT,kj}} = \langle k|\,\boldsymbol{U_{QFT}}\,|j\rangle = \langle k|\,\frac{1}{\sqrt{N}} \sum_{k'=0}^{N-1} \omega^{-k'j}\,\left|k'\right\rangle = \frac{1}{\sqrt{N}}\omega^{-kj} \tag{25.14}$$

which is the expression of the elements in Eq. (25.12). Here, the dummy variable in the summation is changed into k' to avoid confusion, and the orthonormal property of the basis vectors is used.

We can also check that Eqs. (25.12) and (25.13) are equivalent using numerical substitutions.

Example 25.2 Show that, for example, when $j = 1$, Eqs. (25.12) and (25.13) are equivalent.

Set $|j\rangle = |1\rangle$, from Eq. (25.12),

$$\begin{aligned}
&\boldsymbol{U_{QFT}}\,|j\rangle = \boldsymbol{U_{QFT}}\,|1\rangle \\
&= \frac{1}{\sqrt{N}} \begin{pmatrix} \omega^{-0\cdot 0} & \omega^{-0\cdot 1} & \cdots & \omega^{-0\cdot(N-1)} \\ \omega^{-1\cdot 0} & \omega^{-1\cdot 1} & \cdots & \omega^{-1\cdot(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \omega^{-(N-1)\cdot 0} & \omega^{-(N-1)\cdot 1} & \cdots & \omega^{-(N-1)\cdot(N-1)} \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} \\
&= \frac{1}{\sqrt{N}} \begin{pmatrix} \omega^{-0\cdot 1} \\ \omega^{-1\cdot 1} \\ \vdots \\ \omega^{-(N-1)\cdot 1} \end{pmatrix} \\
&= \frac{1}{\sqrt{N}} (\omega^{-0\cdot 1}\,|0\rangle + \omega^{-1\cdot 1}\,|1\rangle + \cdots + \omega^{-(N-1)\cdot 1}\,|N-1\rangle \qquad (25.15)
\end{aligned}$$

The last line is just the $\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega^{-kj}\,|k\rangle$ in Eq. (25.13) when $j = 1$.

Let us now study the property of a 1-qubit QFT.

Example 25.3 For $N = 2$, show how $\boldsymbol{U_{QFT}}$ transforms $|0\rangle$ and $|1\rangle$.

Using Eq. (25.13),

$$\begin{aligned}
\boldsymbol{U_{QFT}}\,|0\rangle &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega^{-k\cdot 0}\,|k\rangle \\
&= \frac{1}{\sqrt{2}} (\omega^{-0\cdot 0}\,|0\rangle + \omega^{-1\cdot 0}\,|1\rangle) \\
&= \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) = |+\rangle \qquad (25.16)
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{U_{QFT}}\,|1\rangle &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega^{-k\cdot 1}\,|k\rangle \\
&= \frac{1}{\sqrt{2}} (\omega^{-0\cdot 1}\,|0\rangle + \omega^{-1\cdot 1}\,|1\rangle) \\
&= \frac{1}{\sqrt{2}} (\omega^{-0\cdot 1}\,|0\rangle + e^{-i2\pi/2}\,|1\rangle) \\
&= \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) = |-\rangle \qquad (25.17)
\end{aligned}$$

This is the same as the definition of the Hadamard gate in Eq. (17.3)! Therefore, a 1-qubit $\boldsymbol{U_{QFT}}$ is just a Hadamard gate.

This is the second time we see a quantum gate transforms a basis vector into a superposition of basis vectors after the Hadamard gate. Therefore, there is no counterpart in classical gates.

Let us now prove that $\boldsymbol{U_{QFT}}$ is **unitary**. To prove that it is unitary, I will try to show that its column vectors are orthonormal (Eq. (9.23)).

Based on the matrix in Eq. (25.12), the m-th column and the n-th column of $\boldsymbol{U_{QFT}}$ are

$$|m\rangle = \frac{1}{\sqrt{N}} \begin{pmatrix} \omega^{-0\cdot m} \\ \omega^{-1\cdot m} \\ \vdots \\ \omega^{-(N-1)\cdot m} \end{pmatrix} \quad \text{and} \quad |n\rangle = \frac{1}{\sqrt{N}} \begin{pmatrix} \omega^{-0\cdot n} \\ \omega^{-1\cdot n} \\ \vdots \\ \omega^{-(N-1)\cdot n} \end{pmatrix} \tag{25.18}$$

Therefore, the inner product of the two vectors is

$$\begin{aligned} \langle m|n\rangle &= \frac{1}{N}\sum_{l=0}^{N-1} (\omega^{-l\cdot m})^* \omega^{-l\cdot n} \\ &= \frac{1}{N}\sum_{l=0}^{N-1} (e^{-i2\pi l\cdot m/N})^* e^{-i2\pi l\cdot n/N} \\ &= \frac{1}{N}\sum_{l=0}^{N-1} (e^{-i2\pi l\cdot(-m)/N}) e^{-i2\pi l\cdot n/N} \\ &= \frac{1}{N}\sum_{l=0}^{N-1} (\omega^{-l\cdot -m})\omega^{-l\cdot n} \\ &= \frac{1}{N}\sum_{l=0}^{N-1} (\omega^{-l\cdot(n-m)}) \end{aligned} \tag{25.19}$$

Note that when we perform the inner product, we need to take the complex conjugate of the coefficients of the *bra* version of the vector. Referring to Eqs. (25.6) and (25.7), again, if $m = n$, this is evaluated to 1 and otherwise, 0. Therefore, the column vectors are orthonormal and $\boldsymbol{U_{QFT}}$ is unitary.

Now, let us do some numerical substitutions.

Example 25.4 Find the matrix representation of $\boldsymbol{U_{QFT}}$ when $N = 2$.

This is a 1-qubit $\boldsymbol{U_{QFT}}$. Using Eq. (25.12), we have

$$\begin{aligned}\boldsymbol{U_{QFT}} &= \frac{1}{\sqrt{2}}\begin{pmatrix}\omega^{-0\cdot 0} & \omega^{-0\cdot 1}\\ \omega^{-(2-1)\cdot 0} & \omega^{-(2-1)\cdot 1}\end{pmatrix}\\ &= \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & \omega^{-1}\end{pmatrix}\\ &= \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & e^{-i2\pi/2}\end{pmatrix}\\ &= \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\end{aligned} \tag{25.20}$$

This is the matrix of the Hadamard gate and is consistent with what we found in Example 25.3.

We see that the first row and first column of any $\boldsymbol{U_{QFT}}$ are evaluated to be 1 because either the row or the column index is 0, and thus it is always $\omega^0 = 1$. For column j, the elements increase as a geometric series with ratio ω^{-j} (or decreases by ω^j). Therefore, we can rewrite Eq. (25.12) as

$$\boldsymbol{U_{QFT}} = \frac{1}{\sqrt{N}}\begin{pmatrix}1 & 1 & 1 & \cdots & 1\\ 1 & \omega^{-1} & \omega^{-2} & \cdots & \omega^{-(N-1)}\\ 1 & \omega^{-2} & \omega^{-4} & \cdots & \omega^{-2(N-1)}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 1 & \omega^{-(N-1)} & \omega^{-2(N-1)} & \cdots & \omega^{-(N-1)(N-1)}\end{pmatrix} \tag{25.21}$$

With this insight, we can construct $\boldsymbol{U_{QFT}}$ of any dimension easily.

Example 25.5 Find the matrix representation of $\boldsymbol{U_{QFT}}$ when $N = 4$.

This is a 2-qubit $\boldsymbol{U_{QFT}}$ gate. First, $\omega = e^{i2\pi/4} = \cos\frac{\pi}{2} + i\sin\frac{\pi}{2} = i$. So the element of the j-th column increases by $\omega^{-j} = i^{-j}$. Therefore,

$$\begin{aligned}\boldsymbol{U_{QFT}} &= \frac{1}{\sqrt{4}}\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & i^{-1} & i^{-2} & i^{-3}\\ 1 & i^{-2} & i^{-4} & i^{-6}\\ 1 & i^{-3} & i^{-6} & i^{-9}\end{pmatrix}\\ &= \frac{1}{2}\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & -i & -1 & i\\ 1 & -1 & 1 & -1\\ 1 & i & -1 & -i\end{pmatrix}\end{aligned} \tag{25.22}$$

where elements in columns 0, 1, 2, and 3 increase by $i^{-0} = 1, i^{-1} = -i, i^{-2} = -1$, and $i^{-3} = i$, respectively, from row to row.

25.5 Inverse Quantum Fourier Transform

The inverse Quantum Fourier Transform (IQFT) is just the inverse matrix of QFT. It is easy to remember as it happens that it can be constructed by removing the negative sign in the exponents in the QFT matrix. Therefore,

$$\begin{aligned} \boldsymbol{U_{IQFT}} &= \frac{1}{\sqrt{N}} \begin{pmatrix} \omega^{0\cdot 0} & \omega^{0\cdot 1} & \cdots & \omega^{0\cdot(N-1)} \\ \omega^{1\cdot 0} & \omega^{1\cdot 1} & \cdots & \omega^{1\cdot(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \omega^{(N-1)\cdot 0} & \omega^{(N-1)\cdot 1} & \cdots & \omega^{(N-1)\cdot(N-1)} \end{pmatrix} \\ &= \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^1 & \omega^2 & \cdots & \omega^{(N-1)} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(N-1)} \\ \vdots & \vdots & \ddots & \vdots & \\ 1 & \omega^{(N-1)} & \omega^{2(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{pmatrix} \end{aligned} \tag{25.23}$$

To prove it, we only need to show $\boldsymbol{U_{IQFT}U_{QFT}} = \boldsymbol{U_{QFT}U_{IQFT}} = \boldsymbol{I}$. This is not difficult if you can see that the i-th row vector of $\boldsymbol{U_{IQFT}}$ is the complex conjugate of the i-th column vector of $\boldsymbol{U_{QFT}}$. By utilizing the definition of matrix multiplication and the unitary property like what we did for Eq. (25.19), we can prove it. That also means that $\boldsymbol{U_{IQFT}}$ *and* $\boldsymbol{U_{QFT}}$ *are the inverse matrices of each other.*

25.6 Summary

In this chapter, we reviewed the important properties of complex numbers, particularly those of the N-th roots of unity that result in constructive and destructive interferences in quantum algorithms. We learned that QFT is the same as DFT when DFT is applied to a quantum state vector. The QFT matrix is constructed such that it expresses the constructive and destructive interference properties of the N-th root of unity when an appropriate input state is presented. We learned how to construct the matrix of QFT and IQFT. We also learned that QFT and IQFT are inverse matrices to the other.

Problems

25.1 Discrete Fourier Transform Definition
Use matrix multiplication to show that Eq. (25.10) is equivalent to Eq. (25.9).

25.2 Hermiticity of Quantum Fourier Transform
Is $\boldsymbol{U_{QFT}}$ Hermitian? Does it need to be Hermitian?

25.3 Unitarity of Quantum Fourier Transform
Prove that $\boldsymbol{U_{QFT}}$ is unitary by considering the orthonormal property of the row vectors.

25.4 Construction of $\boldsymbol{U_{QFT}}$
Construct a 3-qubit $\boldsymbol{U_{QFT}}$.

25.5 Inverse Quantum Fourier Transform 1
Show $\boldsymbol{U_{IQFT}U_{QFT} = U_{QFT}U_{IQFT} = I}$.

25.6 Inverse Quantum Fourier Transform 2
Construct the matrix of a 2-qubit $\boldsymbol{U_{IQFT}}$. Check your answer by multiplying it with the 2-qubit $\boldsymbol{U_{QFT}}$ in Eq. (25.22).

25.7 Programming of QFT
Use Google Colab to create the matrix of a $\boldsymbol{U_{QFT}}$ gate of arbitrary N.

Chapter 26
Quantum Fourier Transform II

26.1 Learning Outcomes

Have a deeper appreciation of the difference between basis transformation and vector transformation; be aware of the existence of the two definitions of QFT that are inverse of the other; understand how to construct an n-qubit SWAP gate; able to construct a 3-qubit QFT circuit and run on IBM-Q.

26.2 Another Definition of QFT and IQFT

There is another definition of $\boldsymbol{U_{QFT}}$ and $\boldsymbol{U_{IQFT}}$. That new QFT, $\boldsymbol{U_{QFT,new}}$, uses the definition of our $\boldsymbol{U_{IQFT}}$, and the new IQFT, $\boldsymbol{U_{IQFT,new}}$, uses the definition of our $\boldsymbol{U_{QFT}}$. In other words, for the same matrix, it is called differently. We need to be careful when reading the algorithms that use QFT and IQFT to make sure which definition they are using. Other than that everything else is the same as what we have learned.

But why there are two definitions. For our definition in the previous chapter, the QFT matrix is made exactly the same as the DFT matrix. This is a natural way to define the QFT matrix. Let us take a closer look at how $\boldsymbol{U_{QFT}}$ transforms a vector. Assume the vector is $|X\rangle = x_0|0\rangle + x_1|1\rangle + \cdots + x_{N-1}|N-1\rangle$. This is a quantum state vector on the basis formed by the basis vectors $|0\rangle$ to $|N-1\rangle$. And assume after transformation, it becomes $|Y\rangle = y_0|0\rangle + y_1|1\rangle + \cdots + y_{N-1}|N-1\rangle$. Then

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_26

the transformation shares exactly the same equation as Eq. (25.10) because we are using the same matrix for DFT and QFT.

$$|Y\rangle = \boldsymbol{U_{QFT}}\,|X\rangle$$

$$\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{pmatrix} = \frac{1}{\sqrt{N}} \begin{pmatrix} \omega^{-0\cdot 0} & \omega^{-0\cdot 1} & \cdots & \omega^{-0\cdot(N-1)} \\ \omega^{-1\cdot 0} & \omega^{-1\cdot 1} & \cdots & \omega^{-1\cdot(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \omega^{-(N-1)\cdot 0} & \omega^{-(N-1)\cdot 1} & \cdots & \omega^{-(N-1)\cdot(N-1)} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{pmatrix} \tag{26.1}$$

Therefore, each component of the transformed vector, $|Y\rangle$, still has the same equation as in the first part of Eq. (25.9). That is

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \omega^{-kj} x_j \tag{26.2}$$

Therefore, the matrix we use is for transforming a quantum vector. It tells us what we will get (i.e. $|Y\rangle$) when we rotate a given $|X\rangle$ using this matrix. And *we call it QFT, instead of IQFT, because the matrix for transforming/rotating a quantum vector is the same as the DFT*. Figure 26.1 shows an example of transforming a vector $|V\rangle$ by rotating it counter-clockwise by θ. The basis and basis vectors are not changed, and only the vector is rotated/transformed into $|V_{new}\rangle$ in the original $|0\rangle$ / $|1\rangle$ basis. However, we can also perform a transformation of the basis in the reverse direction to achieve the same purpose. In Fig. 26.1, if we rotate the basis vector clockwise by θ, we achieve the same result. We see that to achieve the same result, we need to rotate the basis using an inverse matrix of the one for rotating the vector. In the QFT case, applying QFT, $\boldsymbol{U_{QFT}}$, to a vector is the same as applying IQFT, $\boldsymbol{U_{IQFT}}$, to the basis.

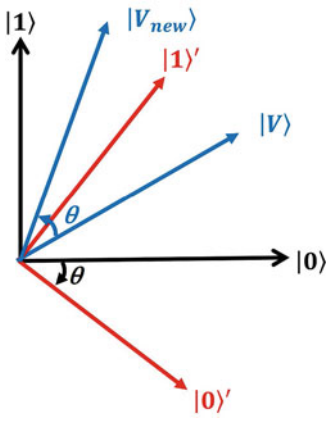


Fig. 26.1 Illustration of two equivalent transformations, transformation of basis (rotating basis vectors clockwise by θ) and transformation of vector (rotating vector by θ counter-clockwise)

We define quantum gate on how it rotates the basis vectors. Therefore, it is also natural to define the QFT matrix as the matrix for rotating the basis vectors (instead of the matrix of rotating the vector as we have been doing). It is the reason why in some other sources, our $\boldsymbol{U_{IQFT}}$ is called the QFT matrix, $\boldsymbol{U_{QFT,new}}$.

In summary, $\boldsymbol{U_{QFT}}$ (with negative exponents) is always used for transforming the vectors. It is called QFT in this book but may be called IQFT in other sources.

26.3 Many-Qubit SWAP Gate

The many-qubit SWAP gate is an important part of the QFT circuit. We discussed the 2-qubit SWAP gate in Chap. 16. The definition of a 2-qubit swap gate in Eq. (16.1) is repeated here for convenience.

$$\boldsymbol{U_{SWAP}}\,|ab\rangle = |ba\rangle \tag{26.3}$$

An n-qubit SWAP gate is defined as

$$\boldsymbol{U_{SWAP,n}}\,|x_0x_1x_2\cdots x_{n-3}x_{n-2}x_{n-1}\rangle = |x_{n-1}x_{n-2}x_{n-3}\cdots x_2x_1x_0\rangle \tag{26.4}$$

where $|x_0x_1x_2\cdots x_{n-3}x_{n-2}x_{n-1}\rangle$ is the basis vector of the 2^n-dimensional space, $\mathbb{C}^{2^n}$. As a reminder of the meaning of the basis vectors of a 2^n-dimensional space, we can also write $|x_0x_1x_2\cdots x_{n-3}x_{n-2}x_{n-1}\rangle = |x_0\rangle\,|x_1\rangle\,|x_2\rangle\cdots|x_{n-3}\rangle\,|x_{n-2}\rangle\,|x_{n-1}\rangle$. We see that we can implement the n-qubit SWAP gate using a series of 2-qubit swap gate. Let $\boldsymbol{U_{SWAP,i,j}}$ be a 2-qubit SWAP gate swapping the i-th and j-th qubits. We have

$$\begin{aligned}
&\boldsymbol{U_{SWAP,n}}\,|x_0x_1x_2\cdots x_{n-3}x_{n-2}x_{n-1}\rangle\\
&= \boldsymbol{U_{SWAP,n/2-1,n/2}}\cdots\boldsymbol{U_{SWAP,2,n-3}}\boldsymbol{U_{SWAP,1,n-2}}\boldsymbol{U_{SWAP,0,n-1}}\,|x_0x_1x_2\cdots x_{n-3}x_{n-2}x_{n-1}\rangle\\
&= \boldsymbol{U_{SWAP,n/2-1,n/2}}\cdots\boldsymbol{U_{SWAP,2,n-3}}\boldsymbol{U_{SWAP,1,n-2}}\,|x_{n-1}x_1x_2\cdots x_{n-3}x_{n-2}x_0\rangle\\
&= \boldsymbol{U_{SWAP,n/2-1,n/2}}\cdots\boldsymbol{U_{SWAP,2,n-3}}\,|x_{n-1}x_{n-2}x_2\cdots x_{n-3}x_1x_0\rangle\\
&\;\;\vdots\\
&= |x_{n-1}x_{n-2}x_{n-3}\cdots x_2x_1x_0\rangle
\end{aligned} \tag{26.5}$$

Here we assume n is even. We first swap the 0-th and $(n-1)$-th qubits and then swap the 1-st and $(n-2)$-th qubits and continue until the $(n/2-1)$-th and $n/2$-th qubits. If n is odd, the middle qubit need not to be swapped.

Example 26.1 Using both Eqs. (26.4) and (26.5), demonstrate how a 5-qubit basis vector, $|10100\rangle$, is transformed/rotated. Simulate using IBM-Q.

From Eq. (26.4), we have

$$\boldsymbol{U}_{\boldsymbol{SWAP,5}}\,|10100\rangle = |00101\rangle \tag{26.6}$$

So $|10100\rangle$ is rotated to $|00101\rangle$.
Using Eq. (26.5), we have

$$\begin{aligned} &\boldsymbol{U}_{\boldsymbol{SWAP,5}}\,|10100\rangle \\ =\ &\boldsymbol{U}_{\boldsymbol{SWAP,1,3}}\boldsymbol{U}_{\boldsymbol{SWAP,0,4}}\,|10100\rangle \\ =\ &\boldsymbol{U}_{\boldsymbol{SWAP,1,3}}\,|00101\rangle \\ =\ &|00101\rangle \end{aligned} \tag{26.7}$$

where we keep the middle qubit, the second qubit, unchanged as N is odd. Or you can treat it as swapping with itself. We can thus build a SWAP gate as shown in Fig. 26.2. The result shows that the 5-qubit SWAP gate can be decomposed into two 2-qubit SWAP gates and transforms $|10100\rangle$ into $|00101\rangle$.

Figure 26.3 shows the implementation on IBM-Q. Note that the symbol of a 2-qubit SWAP gate is different from ours. It is used to test the $|10100\rangle$ basis vector. Therefore, two NOT gates are added to the 1st ($q4$) and the 3rd ($q2$) most significant qubits to generate $|10100\rangle$ from the initial state $|00000\rangle$. There are 5 classical bits to store the measured qubits. The MSB is at the bottom corresponding to bit 4.

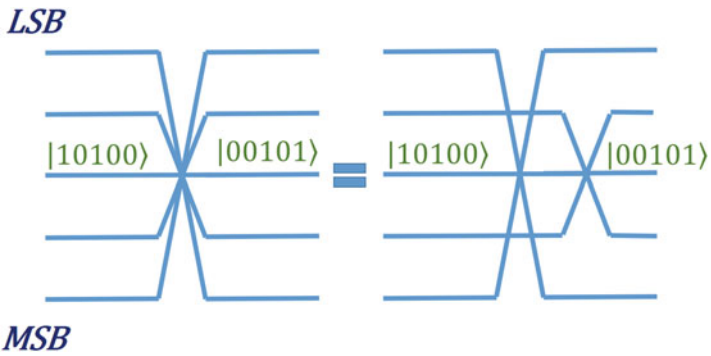


Fig. 26.2 Quantum circuit for a 5-qubit SWAP gate (left). It can be implemented using two 2-qubit SWAP gates (right)

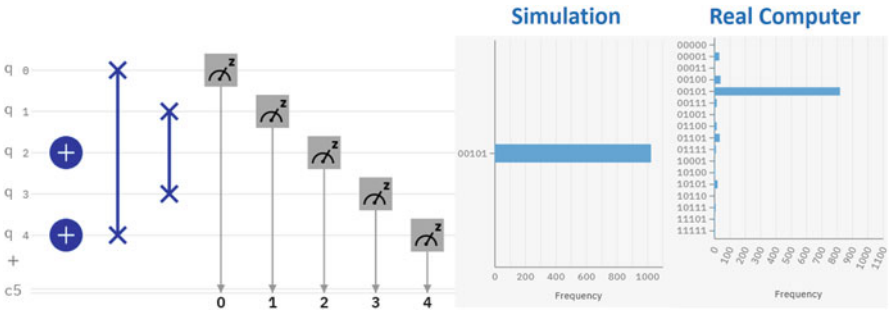


Fig. 26.3 Quantum circuit constructed to test the transformation of $|10100\rangle$ under a 5-qubit SWAP gate based on Fig. 26.2 in IBM-Q. The simulation shows that the output is $|00101\rangle$ as expected. In hardware, there are errors due to noise. Note that the MSB is at the bottom

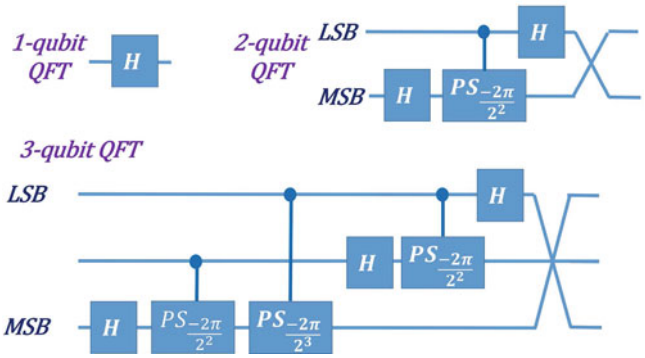


Fig. 26.4 Quantum circuits of 1-, 2-, and 3-qubit QFT

26.4 QFT Circuit

Here we will describe how to construct a QFT circuit. We will not prove it, but we will show that it is correct for the 1- and 2-qubit cases.

A 1-qubit QFT circuit is just a Hadamard gate as we have proved in Eq. (25.20) (Fig. 26.4).

A 2-qubit QFT circuit is also shown in Fig. 26.4. We need a 2-qubit SWAP gate, two Hadamard gates, and also a controlled phase shift gate. The matrix for a controlled phase shift gate is given in Eq. (16.13), and it is repeated here for convenience.

$$\boldsymbol{U_{CPS,\Phi}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\Phi} \end{pmatrix} \tag{26.8}$$

From Fig. 26.4, we see that the phase needs to be shifted is $\phi = \frac{-2\pi}{2^2} = \frac{-\pi}{2}$. Therefore, the matrix is

$$\boldsymbol{U}_{\boldsymbol{CPS},\frac{-2\pi}{2^2}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{-i\pi/2} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -i \end{pmatrix} \tag{26.9}$$

Therefore, the matrix of the 2-qubit QFT circuit is given by (note again that the vector goes from left to right in the circuit, but in matrix multiplication, the vector goes from right to left)

$$\begin{aligned} &\boldsymbol{U}_{\boldsymbol{SWAP}}(\boldsymbol{I} \otimes \boldsymbol{H})\boldsymbol{U}_{\boldsymbol{CPS},\frac{-2\pi}{2^2}}(\boldsymbol{H} \otimes \boldsymbol{I}) \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -i \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} \\ &= \frac{1}{2} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & -i & 0 & i \end{pmatrix} \\ &= \frac{1}{2} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \\ 1 & i & -1 & -i \end{pmatrix} \\ &= \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{pmatrix} \end{aligned} \tag{26.10}$$

which is the same as Eq. (25.22).

There Are Two Important Notes First, the controlled phase shift gate has the LSB as the control qubit and the MSB as the target qubit in the QFT circuit. When we introduced the $\boldsymbol{U}_{\boldsymbol{CPS},\boldsymbol{\phi}}$ in Fig. 16.2 and its matrix in Eq. (16.13), the control qubit was the MSB. However, they have the same matrix because $e^{i\phi}$ appears only when the basis vector is $|11\rangle$ in both cases (as the control qubit needs to be 1 and it is applied to the target qubit only when it is 1). Therefore, it does not matter which is the MSB. Thus, we use Eq. (16.13) directly in Eq. (26.10). Second, we defined QFT as the matrix that rotates the vectors (Eq. (26.1)); therefore, the phase shift is

negative. If you will use the definition from some other sources (i.e. define the QFT as the matrix rotates the basis vectors), then it will not have the negative sign.

26.4.1 Implementation of a 3-Qubit QFT Circuit

Figure 26.4 also shows the circuit for a 3-qubit QFT. We see that when we add one more qubit so that the total number of qubits is n, that qubit needs to go through a $\boldsymbol{H}$ gate, and then $(n-1)$ $\boldsymbol{U_{CPS,\phi}}$ gates, with ϕ going from $\frac{-2\pi}{2^2}$ to $\frac{-2\pi}{2^n}$. For the 3-qubit QFT, we need a $\boldsymbol{U_{CPS,\frac{-2\pi}{2^3}}}$ gate, which has a matrix of

$$\boldsymbol{U_{CPS,\phi}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\frac{-2\pi}{2^3}} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\frac{-\pi}{4}} \end{pmatrix} \tag{26.11}$$

Let us implement this circuit on IBM-Q. Figure 26.5 shows the implementation. Everything is a straightforward mapping to the circuit in Fig. 26.4. The only part that needs special attention is the $\boldsymbol{U_{CPS,\phi}}$ gate. To create the $\boldsymbol{U_{CPS,\phi}}$, one needs to choose the "Phase gate" icon first and then drop the "Control gate modifier" (the icon with a black dot) to it. We then choose which qubit to be the control qubit by clicking on the dots to appear. One can then double-click the phase gate to set the phase angle. The phase angles used from left to right are $-\pi/2$, $-\pi/4$, and $-\pi/2$ in this circuit. Since the input is $|000\rangle$ by default (the initialized quantum state), we expect the output to be an equal linear superposition of all basis vectors, and indeed the hardware execution shows the correction result besides with noise.

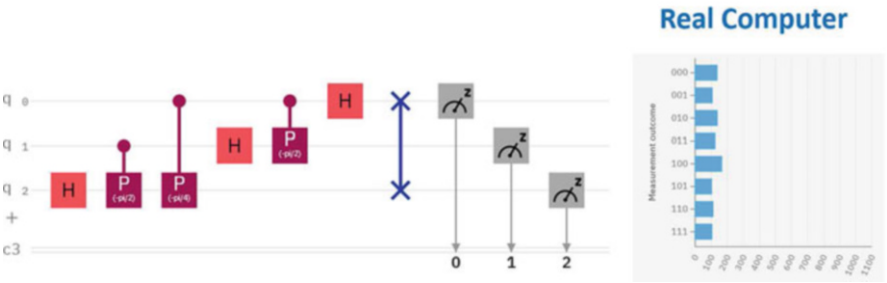


Fig. 26.5 Quantum circuit of a 3-qubit QFT implemented on IBM-Q. The phase angles used from left to right are $-\pi/2$, $-\pi/4$, and $-\pi/2$. The hardware result is shown on the right

26.5 Implementation of IQFT

By definition, IQFT is the inverse of QFT. Therefore, to construct the circuit for IQFT, we only need to reverse the arrangement of the gates in the QFT circuit and replace them with their inverse counterparts. Since the Hadamard gate and the SWAP gate are their own inverse, we can just leave them as they are. For the controlled phase shift gate, all we need to do is to change the angle to positive to get its inverse counterpart. Therefore, we can just reverse the order of the QFT circuit and negate the phase angles to construct an IQFT circuit. This is shown at the top of Fig. 26.6.

Another way to implement the circuit for IQFT is by realizing that IQFT and QFT have the same matrix form except the exponents have different signs (See Eqs. (25.12) and (25.23)). In our previous examples, we see that phase shift term $e^{i\phi} = e^{-i2\pi/2^n}$ is just the ω^{-1} in the QFT for the 2^n dimension, and this is the only source for generating the ω in the final QFT or IQFT matrices (see e.g. Eq. (26.10)). Therefore, we can just use the QFT gates as they are but negate all phases, i.e. change ϕ to $-\phi$. This is shown at the bottom of Fig. 26.6.

If you implement the circuits in Fig. 26.6, you will get $|011\rangle$. This is because the input is converted into $|011\rangle$ from $|000\rangle$ through two NOT gates, and then it goes through QFT and IQFT. Since $\boldsymbol{U_{QFT}U_{IQFT}} = \boldsymbol{I}$, it will restore back to $|011\rangle$.

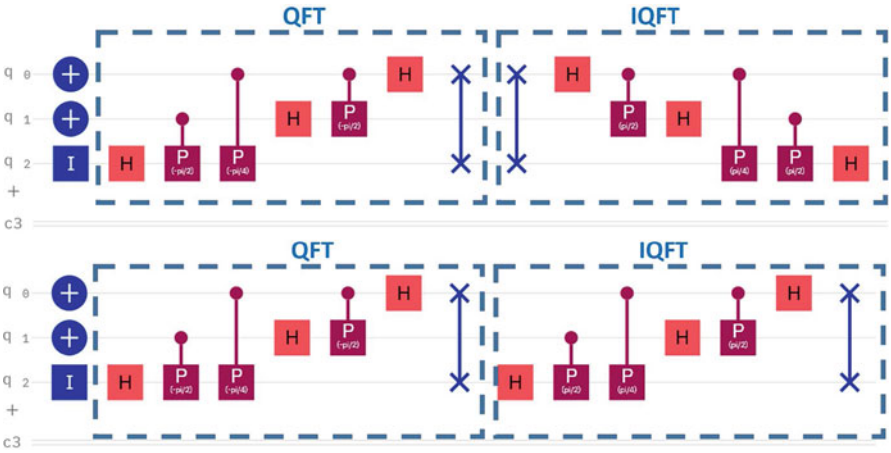


Fig. 26.6 Implementation of IQFT after QFT. Top: IQFT is implemented by reversing the gates in QFT with ϕ changed to $-\phi$. Bottom: IQFT is implemented by using QFT circuits with ϕ changed to $-\phi$

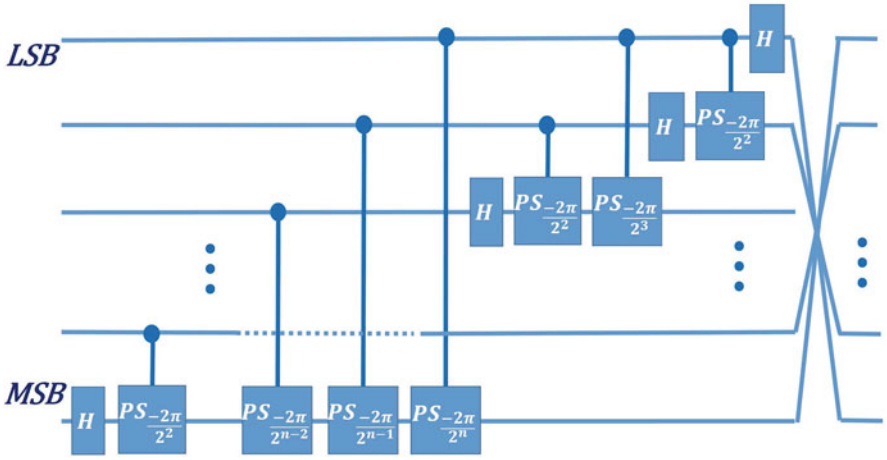


Fig. 26.7 An n-qubit QFT circuit

26.6 General Circuit of QFT

The general circuit for an n-qubit QFT is shown in Fig. 26.7. As mentioned earlier, when we add one more qubit, that qubit needs to go through a $\boldsymbol{H}$ gate and then $n-1$ $\boldsymbol{U_{CPS,\phi}}$ gates, with ϕ going from $\frac{-2\pi}{2^2}$ to $\frac{-2\pi}{2^n}$. Therefore, for the MSB, it has $n-1$ controlled phase shift gates after the Hadamard gate.

To construct IQFT, as we have shown in the 3-qubit case, we can use the same circuit but use positive phases, or we can reverse the circuit and use positive phases. That is, changing $\boldsymbol{U_{CPS,\frac{-2\pi}{2^j}}}$ to $\boldsymbol{U_{CPS,\frac{2\pi}{2^j}}}$, for $j = 2$ to n.

26.7 Summary

In this chapter, we have learned that there is another definition of QFT. But this is just a naming issue. The most important is to realize that the matrix that rotates the vector has negative exponents in the elements. We need to make sure of the exact meaning of QFT when reading other sources. We learned how to implement an n-qubit SWAP gate by decomposing it into 2-qubit SWAP gates. We then learned how to construct QFT circuits using Hadamard gates, controlled phase shift gates, and SWAP gates. We implemented a 3-qubit QFT and IQFT circuits on IBM-Q after learning how to create a controlled phase shift gate.

Problems

26.1 Quantum Fourier Transform of a Basis State
Construct the 3-qubit QFT matrix and show that when the input is $|000\rangle$, the output is an equal linear superposition of all basis vectors as shown in Fig. 26.5.

26.2 Inverse of Controlled Phase Shift Gate
Show that $\boldsymbol{U}_{CPS,\phi}\boldsymbol{U}_{CPS,-\phi} = \boldsymbol{I}$.

26.3 Inverse of SWAP Gate
Show that a 2-qubit SWAP gate is an inverse to itself.

26.4 Construction of IQFT on IBM-Q
Construct the circuits in Fig. 26.6 on IBM-Q and verify the output. Try different combinations of NOT gates to get different basis vectors as the input to further verify the circuits.

26.5 n-Qubit QFT Circuit
Based on Fig. 26.7, derive the number of controlled phase shift gates required in an n-qubit QFT circuits. If all gates have the same execution time, how long does it take to finish the computation? (Hints: Do not forget the Hadamard gates and SWAP gates.)

26.6 3-Qubit QFT
Derive the matrix of a 3-qubit QFT gate.

26.7 7-Qubit SWAP Gate
How would $|1001100\rangle$ be transformed in a 7-qubit SWAP gate?

Chapter 27
Bloch Sphere and Single-Qubit Arbitrary Unitary Gate

27.1 Learning Outcomes

Able to describe how to map a qubit state to the surface of the Bloch sphere; able to perform rotation on the Bloch sphere for a given set of Euler angles; be aware of the correct and incorrect relationship between the qubit space and the real 3D space; able to construct arbitrary unitary rotation using the $\boldsymbol{U_{\theta,\phi,\lambda}}$ gate.

27.2 Bloch Sphere

Very often the **Bloch Sphere** is taught at the beginning of a quantum computing class. However, it is not necessary to understand the Bloch Sphere first to do quantum computing. But Bloch Sphere is a very useful tool (but can be confusing) for us to understand the operation of quantum gates. That is why we introduce until almost the end. If we use it correctly, we can use it to help us construct quantum gates and understand the underlying physics.

We are all familiar with the concept that a single qubit resides in the $\mathbb{C}^2$ space. This is a two-dimensional complex space. It means that it has two basis vectors, $|0\rangle$ and $|1\rangle$, which are *orthonormal* to each other. Any qubit state is a vector in this space formed by a linear combination of $|0\rangle$ and $|1\rangle$ with *complex coefficients*. We have been using a real 2D space to illustrate the properties of a qubit state vector, but we emphasize that this is just an illustration and **the $\mathbb{C}^2$ space is NOT the real 2D space that we can feel** (e.g. Fig. 5.1).

How many **degrees of freedom (DOF)** does a qubit state have? It means how many real numbers do we need to specify in order to fix a qubit. We know that any single-qubit state $|\Psi\rangle$ can be expressed as $|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, where α and β are complex numbers. Each of the α and β is determined by two real numbers.

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_27

For example, $\alpha = |\alpha| e^{i\delta_\alpha}$ and $\beta = |\beta| e^{i\delta_\beta}$, where $|\alpha|$, δ_α, $|\beta|$, and δ_β are the real numbers. Therefore, it *looks like* it has 4 DOFs.

However, for any physical qubit, it has to be normalized. Therefore, $|\alpha|^2 + |\beta|^2 = 1$. This reduces the DOF by 1 because if I specify $|\alpha|$, $|\beta|$ is specified at the same time due to this normalization equation. This also means that I can determine $|\alpha|$ and $|\beta|$ using 1 single real number parameter as long as I make the vector normalized. For example, I can set $|\alpha| = \cos\frac{\theta}{2}$ and $|\beta| = \sin\frac{\theta}{2}$ as $\cos\frac{\theta}{2}^2 + \sin\frac{\theta}{2}^2 = 1$ will help satisfy the normalization criteria. So, now, the amplitudes of α and β can be described by a single real parameter θ, and the DOF of the qubit state is 3.

There is also another thing that can help us further reduce the DOF. The global phase of a qubit does not have any physical meaning (this was discussed after Eq. (6.12)). For example,

$$\begin{aligned}
|\Psi\rangle &= \alpha\,|0\rangle + \beta\,|1\rangle \\
&= |\alpha| e^{i\delta_\alpha}\,|0\rangle + |\beta| e^{i\delta_\beta}\,|1\rangle \\
&= \cos\frac{\theta}{2} e^{i\delta_\alpha}\,|0\rangle + \sin\frac{\theta}{2} e^{i\delta_\beta}\,|1\rangle \\
&= e^{i(\delta_\alpha+\delta_\beta)/2}\left(\cos\frac{\theta}{2} e^{i(\delta_\alpha-\delta_\beta)/2}\,|0\rangle + \sin\frac{\theta}{2} e^{i(-\delta_\alpha+\delta_\beta)/2}\,|1\rangle\right) \\
&= e^{i(\delta_\alpha+\delta_\beta)/2}\left(\cos\frac{\theta}{2} e^{-i\phi/2}\,|0\rangle + \sin\frac{\theta}{2} e^{i\phi/2}\,|1\rangle\right) \\
&= e^{i\gamma}\left(\cos\frac{\theta}{2} e^{-i\phi/2}\,|0\rangle + \sin\frac{\theta}{2} e^{i\phi/2}\,|1\rangle\right)
\end{aligned} \tag{27.1}$$

Here, we factorize a **global phase** $(\delta_\alpha + \delta_\beta)/2$ and call it γ. And we also define a new parameter $\phi = \delta_\beta - \delta_\alpha$. So we still have 3 DOFs characterized by θ, ϕ, and γ. The global phase means the phase shared by both $|0\rangle$ and $|1\rangle$. It has no physical significant because the physical properties for this single qubit are all computed by involving the *bra* and *ket* versions of the vector. For example, the expectation value of a matrix, $\boldsymbol{M}$, is $\langle\Psi|\,\boldsymbol{M}\,|\Psi\rangle$. We can take $e^{i\gamma}$ out from $|\Psi\rangle$ to be $e^{i\gamma}$ and take $e^{i\gamma}$ out from $\langle\Psi|$ to be $e^{-i\gamma}$ (note that we need to take the complex conjugate of the coefficient when it is a *bra* version, see Eq. (5.7)). $e^{i\gamma}$ and $e^{-i\gamma}$ multiply together to be 1. Therefore, the global phase factor does not affect the physical results. We can ignore γ when we describe a single qubit. As a result, we only need two real parameters, θ and ϕ, to describe a qubit. *Without losing its physics, a single qubit can then be completely described as*

$$|\Psi\rangle = \cos\frac{\theta}{2} e^{-i\phi/2}\,|0\rangle + \sin\frac{\theta}{2} e^{i\phi/2}\,|1\rangle \tag{27.2}$$

Since it has only two parameters, we can describe it on a real 2D plane. But the 2D plane is too much for it (it can describe also vectors with non-unit lengths) and

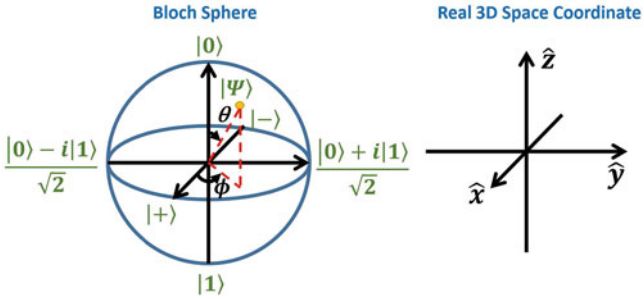


Fig. 27.1 The Bloch sphere (left) and its relationship to the 3-D real space (right)

also does not give a lot of insights. We can also describe it on the *surface* of a unit sphere (sphere with unit length), and this is called the **Bloch Sphere**. Figure 27.1 shows the Bloch sphere and its relationship to the real 3D space we live in. The *surface* of the Bloch sphere can be mapped to the space of a qubit in Eq. (27.2). We may say that we **embed/map** the qubit space in our real 3D space. It is very important to understand that *we only embed the qubit space to our real 3D space, and it does not mean that the qubit state is in our real 3D space*. This is just like we draw a map on 2D paper. First, San Francisco is not on your paper although you see it on the map. Second, the Earth's surface is the surface of a sphere, but you map it to the 2D plane on the paper. However, there are many advantages in describing the qubit using the Bloch sphere. First, it allows us to "visualize" how a qubit state evolves. Second, some of the physical properties of the qubit are linked to the Bloch sphere orientation in the real 3D space.

In Fig. 27.1, it shows the location of state $|\Psi\rangle$ on the Bloch sphere when it has the parameters θ and ϕ. We see that if we put the Bloch sphere in the 3D orientation shown in Fig. 27.1, the θ and ϕ are just the **polar angle** and **azimuthal angle** of a spherical coordinate, respectively. Note again, we can *embed* the surface in our 3D space in any way, but embedding in this way gives us the most intuition.

Let us try to understand how some of the important qubit states are mapped to the Bloch sphere.

Example 27.1 To which qubit states do the extrema on the Bloch sphere correspond?

We will only discuss 3 of them here. You will try the rest in the problems. Let us consider the "North Pole." It has $\theta = 0$ and $\phi = 0$. Therefore,

$$\begin{aligned} \left|\Psi_{\theta=0,\phi=0}\right\rangle &= \cos\frac{\theta}{2}e^{-i\phi/2}\,|0\rangle + \sin\frac{\theta}{2}e^{i\phi/2}\,|1\rangle \\ &= \cos\frac{0}{2}e^{-i0/2}\,|0\rangle + \sin\frac{0}{2}e^{i0/2}\,|1\rangle \\ &= |0\rangle \end{aligned} \tag{27.3}$$

What if we choose ϕ to be non-zero? It is still the "North Pole" as long as $\theta = 0$. Then we will get $e^{-i\phi/2}|0\rangle$. However, since $e^{-i\phi/2}$ is a global phase (as the coefficient for $|1\rangle$ is 0), this does not matter to the physics. Remember that we ignore the global phase of the qubit state when we construct the Bloch sphere; therefore, the Bloch sphere has lost the global phase information. *For the same point on the Bloch sphere, we can add any global phase to it without affecting its physical properties*. And a point on the Bloch sphere is not unique, but it corresponds to qubits that differ in only a global phase (e.g. $e^{-i\phi/2}|0\rangle$ and $|0\rangle$ in this case).

Let us consider the point corresponds to $x = -1$ and $y = z = 0$ in the 3D real space. It has $\theta = \pi/2$ and $\phi = \pi$. Therefore,

$$\begin{aligned}
\left|\Psi_{\theta=\pi/2,\phi=\pi}\right\rangle &= \cos\frac{\theta}{2}e^{-i\phi/2}|0\rangle + \sin\frac{\theta}{2}e^{i\phi/2}|1\rangle \\
&= \cos\frac{\pi}{4}e^{-i\pi/2}|0\rangle + \sin\frac{\pi}{4}e^{i\pi/2}|1\rangle \\
&= \frac{1}{\sqrt{2}}(-i)|0\rangle + \frac{1}{\sqrt{2}}i|1\rangle \\
&= -i\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\
&= -i|-\rangle
\end{aligned} \tag{27.4}$$

Again, the global phase $-i = e^{-i\pi/2}$ can be discarded. Therefore, this point corresponds to $|-\rangle$.

Let us now consider the point corresponds to $y = 1$ and $x = z = 0$ in the 3D real space. It has $\theta = \pi/2$ and $\phi = \pi/2$. Therefore,

$$\begin{aligned}
\left|\Psi_{\theta=\pi/2,\phi=\pi/2}\right\rangle &= \cos\frac{\theta}{2}e^{-i\phi/2}|0\rangle + \sin\frac{\theta}{2}e^{i\phi/2}|1\rangle \\
&= \cos\frac{\pi}{4}e^{-i\pi/4}|0\rangle + \sin\frac{\pi}{4}e^{i\pi/4}|1\rangle \\
&= \frac{1}{\sqrt{2}}e^{-i\pi/4}(|0\rangle + e^{i\pi/2}|1\rangle) \\
&= e^{-i\pi/4}\frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)
\end{aligned} \tag{27.5}$$

Again, the global phase $e^{-i\pi/4}$ can be discarded. Therefore, this point corresponds to $\frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$, which is one of the eigenvalues of $\boldsymbol{\sigma_y}$.

From the examples, we see that the extrema of the Bloch sphere in the x, y, and z directions correspond to the eigenvalues of the $\boldsymbol{\sigma_x}$, $\boldsymbol{\sigma_y}$, and $\boldsymbol{\sigma_z}$ matrices, respectively (e.g. Eq. (6.11)). This is the first place where we see it is related to our real 3D space. This is because, for example, if it is a spin qubit under an external magnetic field (we will not discuss here), the **Pauli matrices** are related to the directions of

the magnetic field. So now we see the link between the extrema in a direction (e.g. z direction) on the Bloch sphere to the external magnetic direction (e.g. z direction). But again, *having* $|0\rangle$ *at the "North pole" and* $|1\rangle$ *at the "South pole" does not mean that the states are on the opposite direction in the real 3D space!* We are just embedding the space in our real 3D space.

27.3 Expectation Values of Pauli Matrices

Although we will not cover it in detail, I already told you that the Pauli matrices are related to the directions of the external magnetic field in the spin qubit case. Moreover, the expectation values of the Pauli matrices of any state are related to the energy it gains under the external magnetic field. Therefore, the calculation of the expectation value of the Pauli matrices is very important.

Example 27.2 Find the expectation value of $\boldsymbol{\sigma}_z$ of $|\Psi\rangle$.

First, we can express $|\Psi\rangle$ in its column form. That is $|\Psi\rangle = \cos\frac{\theta}{2}e^{-i\phi/2}|0\rangle + \sin\frac{\theta}{2}e^{i\phi/2}|1\rangle = \begin{pmatrix}\cos\frac{\theta}{2}e^{-i\phi/2}\\ \sin\frac{\theta}{2}e^{i\phi/2}\end{pmatrix}$. Therefore,

$$\begin{aligned}
\langle\Psi|\,\boldsymbol{\sigma}_z\,|\Psi\rangle &= \left(\cos\tfrac{\theta}{2}e^{i\phi/2}\ \sin\tfrac{\theta}{2}e^{-i\phi/2}\right)\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}\begin{pmatrix}\cos\frac{\theta}{2}e^{-i\phi/2}\\ \sin\frac{\theta}{2}e^{i\phi/2}\end{pmatrix}\\
&= \left(\cos\tfrac{\theta}{2}e^{i\phi/2}\ \sin\tfrac{\theta}{2}e^{-i\phi/2}\right)\begin{pmatrix}\cos\frac{\theta}{2}e^{-i\phi/2}\\ -\sin\frac{\theta}{2}e^{i\phi/2}\end{pmatrix}\\
&= \cos\frac{\theta}{2}e^{i\phi/2}\cos\frac{\theta}{2}e^{-i\phi/2} + \sin\frac{\theta}{2}e^{-i\phi/2}\left(-\sin\frac{\theta}{2}e^{i\phi/2}\right)\\
&= \cos\frac{\theta}{2}\cos\frac{\theta}{2} - \sin\frac{\theta}{2}\sin\frac{\theta}{2}\\
&= \cos\theta
\end{aligned} \tag{27.6}$$

Here we have taken the complex conjugates of the coefficients when we write the $\langle\Psi|$. At the end, we use the identity $\cos^2 x - \sin^2 x = \cos 2x$.

What is the geometric meaning of $\cos\theta$ on the Bloch sphere in Fig. 27.1? Since it is a unit sphere, this is just *the projection of the state* $|\Psi\rangle$ *on the z-axis*! This is the second place where we see how the Bloch sphere is related to our real 3D space. Although the states on the Bloch sphere have no direct relationship to the real 3D space, their Paul matrices' expectation values turn out to be the projections on the axes from the states on the Bloch sphere. Therefore, although $|0\rangle$ and $|1\rangle$ do not lie in our real 3D space on the opposite sides of the z-axis, their $\boldsymbol{\sigma}_z$ expectation values do have opposite values. These expectation values are related to the energy splitting and magnetic moments of the qubits, which we will not discuss in this book.

We can also calculate the expectation values of $\boldsymbol{\sigma_x}$ and $\boldsymbol{\sigma_y}$, and we will find that they are just the projections of the state on the x- and y-axis, respectively. If so, without any derivation, based on the geometry, we know that

$$\langle \Psi | \boldsymbol{\sigma_x} | \Psi \rangle = \sin\theta \cos\phi \tag{27.7}$$

$$\langle \Psi | \boldsymbol{\sigma_y} | \Psi \rangle = \sin\theta \sin\phi \tag{27.8}$$

We will prove this in the problems.

27.4 Single-Qubit Arbitrary Unitary Rotation

We have been emphasizing that quantum computing is nothing but just the rotation of the quantum states in the hyperdimensional space that we cannot see. For example, applying a NOT gate to the $|0\rangle$ state will rotate it to the $|1\rangle$ state. Now, since we have mapped the qubit space to the Bloch sphere embedded in the real 3D space, we will be able to "see" how a vector rotates in the 3D space. This is another great benefit of using the Bloch sphere. But I need to emphasize again, the rotation on the Bloch sphere is NOT the rotation of the state in the 3D space. The qubit space is not something we can feel. This is just a convenient visualization.

How do we describe this rotation? This rotation is a rotation in the hyperspace where the qubit state resides, and it must be a unitary rotation and it must be a 2×2 matrix just like any other 1-qubit quantum gate. In general, it is described by a **Single-Qubit Arbitrary Unitary Gate**, $\boldsymbol{U_{\theta,\phi,\lambda}}$.

$$\boldsymbol{U_{\theta,\phi,\lambda}} = \begin{pmatrix} \cos\frac{\theta}{2} & -e^{i\lambda}\sin\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2} & e^{i(\lambda+\phi)}\cos\frac{\theta}{2} \end{pmatrix} \tag{27.9}$$

We will not derive this equation, but there is a lot to appreciate in this equation. First, the parameters, θ, ϕ, and λ, are angles, and they are the **Euler angles** in the **Euler rotation**. We will not discuss the Euler rotation in detail, but it is a 3D real space rotation of a rigid body. Euler rotation is a sequence of three rotations. It can be described in two ways. One is the *intrinsic rotation* in which it rotates about the axes embedded in/moving with the body. We will NOT use this one. Another equivalent way is the *extrinsic rotation* that it rotates about fixed 3D coordinate axes (Fig. 27.2). In the extrinsic rotation, the body will first rotate about the z-axis by λ (also called γ in some literature) and then rotate about the y-axis by θ (also called β in some literature) and finally rotate about the z-axis by ϕ (also called α in some literature). If you are studying Euler rotation for comparison, make sure that we are using the so-called $z - y - z$ basis. Also be careful that in the $\boldsymbol{U_{\theta,\phi,\lambda}}$ notation, we do not put the rotation angles in the order of rotation (i.e. θ, ϕ, λ instead of ϕ, θ, λ or λ, θ, ϕ). These are the confusions you might have if you try to compare to Euler rotation.

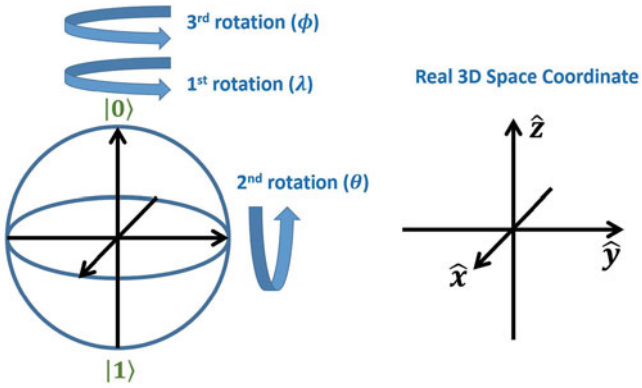


Fig. 27.2 Relationship between the Bloch sphere and Euler rotation

However, regardless of the notations, you see another benefit of using the Bloch sphere, i.e. the angles in Eq. (27.9) have the meanings corresponding to the real 3D space rotations when you embed the Bloch sphere in the 3D space. This can help us understand the transformation/rotation of the state vectors.

Example 27.3 Construct a NOT gate using $\boldsymbol{U_{\theta,\phi,\lambda}}$ by matching the matrices.

We can match the elements in Eq. (27.9) to the NOT gate matrix elements in Eq. (15.6).

$$\begin{gathered} \boldsymbol{U_{\theta,\phi,\lambda}} = \boldsymbol{U_{NOT}} \\ \begin{pmatrix} \cos\frac{\theta}{2} & -e^{i\lambda}\sin\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2} & e^{i(\lambda+\phi)}\cos\frac{\theta}{2} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{gathered} \tag{27.10}$$

I can set up four equations to match each of the elements.

$$\begin{aligned} \cos\frac{\theta}{2} &= 0 \\ -e^{i\lambda}\sin\frac{\theta}{2} &= 1 \\ e^{i\phi}\sin\frac{\theta}{2} &= 1 \\ e^{i(\lambda+\phi)}\cos\frac{\theta}{2} &= 0 \end{aligned} \tag{27.11}$$

But since $\boldsymbol{U_{NOT}}$ is simple, I can see $\theta = \pi$ is required to make the diagonal element zero as $\cos\pi/2 = 0$. Then $\sin\theta/2 = \sin\pi/2 = 1$. And I can put $\lambda = \pi$ so that $-e^{i\lambda} = 1$ and $\phi = 0$ so that $e^{i\phi} = 1$. So the solution is $(\theta, \phi, \lambda) = (\pi, 0, \pi)$. Figure 27.3 shows the path of the rotation of the $|0\rangle$ state under this quantum gate. When it is at the "North" or "South" poles, the first and third rotations have no

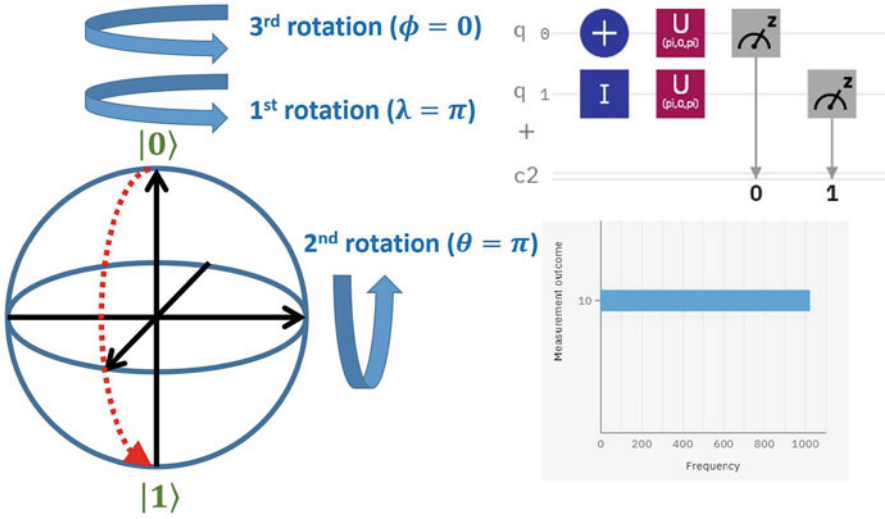


Fig. 27.3 Implementation of U_{NOT} using $U_{\theta,\phi,\lambda}$. Left shows how $|0\rangle$ is rotated on the Bloch sphere under a NOT operation. Right shows the implementation on IBM-Q with $|01\rangle$ as the input. MSB is at the bottom of the circuit

effects. We can see the second rotation (about y-axis by $\theta = \pi$) brings it to $|1\rangle$. The figure also shows the implementation on IBM-Q. Two unentangled qubits are shown. The MSB (bottom) has an input of $|0\rangle$, and the LSB (top) has an input of $|1\rangle$ (after a NOT gate) to the $U_{\theta,\phi,\lambda} = U_{\pi,0,\pi}$. Therefore, the input is $|01\rangle$, and we can see the output is $|10\rangle$ as expected and, thus, $U_{\pi,0,\pi}$ behaves as a U_{NOT}.

Now we have the Bloch sphere to help the visualization. Can we construct matrices by identifying the initial and final states and the path only?

Example 27.4 Construct the matrix corresponding to the rotation of $|0\rangle$ to $|1\rangle$ on the Bloch sphere.

Figure 27.3 clearly shows that the initial state $|0\rangle$ is at the "North" pole and the final state is at the "South" pole. To go from the "North" pole to the "South" pole, the most straightforward way is to rotate about the y-axis by π in the second rotation and do nothing in the first and third rotations. This corresponds to $U_{\theta,\phi,\lambda} = U_{\pi,0,0}$. Then we obtain

$$\begin{aligned} U_{\pi,0,0} &= \begin{pmatrix} \cos\frac{\theta}{2} & -e^{i\lambda}\sin\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2} & e^{i(\lambda+\phi)}\cos\frac{\theta}{2} \end{pmatrix} \\ &= \begin{pmatrix} \cos\frac{\pi}{2} & -e^{i0}\sin\frac{\pi}{2} \\ e^{i0}\sin\frac{\pi}{2} & e^{i(0+0)}\cos\frac{\pi}{2} \end{pmatrix} \\ &= \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \end{aligned} \tag{27.12}$$

Unfortunately, this is *not* a NOT gate. But this matrix correctly describes the transformation from $|0\rangle$ to $|1\rangle$ because

$$\boldsymbol{U}_{\boldsymbol{\pi},\mathbf{0},\mathbf{0}}\,|0\rangle = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle \tag{27.13}$$

The reason is that many gates can bring $|0\rangle$ to $|1\rangle$, but they are not necessarily the NOT gate. We need to use the rotation of a general vector on the Block sphere to derive the NOT gate if we want to construct a quantum gate using $\boldsymbol{U}_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\lambda}}$. We need to match the elements carefully like in Eqs. (27.10) and (27.11). Be aware not to just pick one or two rotation examples of a special state (like $|0\rangle$) and construct the gate directly using the θ, ϕ, and λ seen in the Bloch sphere (like in Eqs. (27.12) and (27.13)). On the other hand, once we have the correct $\boldsymbol{U}_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\lambda}}$ for a certain quantum gate, we can see the "path" of how a qubit is transformed on a Bloch sphere.

Example 27.5 Describe how $|+\rangle$ rotates on the Bloch sphere when the NOT gate is applied.

First, we know that $\boldsymbol{U}_{\boldsymbol{NOT}}\,|+\rangle = \boldsymbol{U}_{\boldsymbol{NOT}}\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{\sqrt{2}}(|1\rangle + |0\rangle) = |+\rangle$. We used the definition of $\boldsymbol{U}_{\boldsymbol{NOT}}$ that it changes $|0\rangle$ and $|1\rangle$ to each other. So, $|+\rangle$ is transformed/rotated to itself.

The NOT gate is $\boldsymbol{U}_{\boldsymbol{\pi},\mathbf{0},\boldsymbol{\pi}}$ as proved in Example 27.3. So, on the Bloch sphere, it will first rotate about the z-axis by π, and then about the y-axis by π, and do nothing in the third rotation as $\phi = 0$. Figure 27.4 shows that it is brought to $|-\rangle$ in the first rotation and brought back to $|+\rangle$ in the second rotation.

In the same figure, it shows that if we use $\boldsymbol{U}_{\boldsymbol{\pi},\mathbf{0},\mathbf{0}}$, we get the wrong result because $\boldsymbol{U}_{\boldsymbol{\pi},\mathbf{0},\mathbf{0}}$ is not the NOT gate.

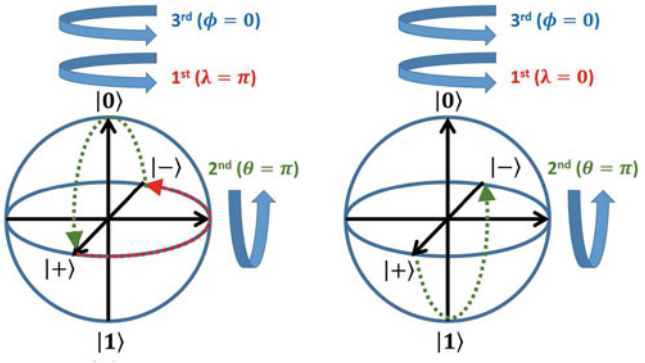


Fig. 27.4 Left: Rotation path of $|+\rangle$ under a NOT gate operation. Right: Rotation path of $|+\rangle$ under the $\boldsymbol{U}_{\boldsymbol{\pi},\mathbf{0},\mathbf{0}}$ operation

27.5 Summary

In this chapter, we have introduced the Bloch sphere that is not necessary to understand quantum computing. However, it gives us intuition and is particularly useful if we want to link it to the physics construction and operation of qubits (which is not discussed in this book). But most importantly, we need to understand that the Bloch sphere is just a way to embed the qubit space into our real 3D space. If so, you will not ask why $|0\rangle + |1\rangle$ is not zero as it looks like the "vectors" at the "North" and "South" poles should get canceled. This shows again, they are not the states in our real 3D space. If you use it carefully, you will be able to understand the arbitrary unitary gate better. This can be used to construct other single-qubit gates. We appreciate that if we can represent the gate in the form of $\boldsymbol{U_{\theta,\phi,\lambda}}$, we will know the evolution path of the qubit, and this will help us design quantum gate hardware (not covered in this book). For now, knowing the path is good for visualization.

Problems

27.1 Effect of Global Phase on a Single Qubit
Show that the global phase of a single qubit has no effect when finding its expectation value on a $\boldsymbol{\sigma_z}$ matrix.

27.2 Extrema on the Bloch Sphere
To which qubit states do the extrema on the Bloch sphere correspond? We have proved three of them in the text. Please prove the rest.

27.3 Expectation Values of Pauli Matrices of the States on Bloch Sphere
Prove Eqs. (27.7) and (27.8).

27.4 State on Bloch Sphere
Find θ and ϕ for $|\Psi\rangle = (\frac{3}{4} - i\frac{\sqrt{3}}{4})|0\rangle + (\frac{\sqrt{3}}{4} + i\frac{1}{4})|1\rangle$. Draw it on the Bloch sphere.

27.5 $\boldsymbol{U_{\theta,\phi,\lambda}}$ as a Hadamard Gate
Construct the Hadamard gate using $\boldsymbol{U_{\theta,\phi,\lambda}}$ like in Example 27.3. Hints: Answer is $\boldsymbol{U_{\pi/2,0,\pi}}$.

27.6 Rotation on Bloch Sphere by Hadamard Gate
Show on the Bloch sphere how $|+\rangle$ and $|1\rangle$ evolve when it is applied with the Hadamard gate.

27.7 Entanglement Circuit on IBM-Q
Construct an entanglement state using only $\boldsymbol{U_{\theta,\phi,\lambda}}$ on IBM-Q. We know that we need a CNOT gate, and it can be done by constructing a controlled version of $\boldsymbol{U_{\theta,\phi,\lambda}}$. Set $\gamma = 0$, which will be discussed in the next chapter.

Chapter 28
Quantum Phase Estimation

28.1 Learning Outcomes

Understand the four parameters in the general controlled unitary gate; able to describe the meaning of the qubits in the Quantum Phase Estimation (QPE) algorithm; able to explain the gates needed to construct a QPE circuit; understand mathematically how QPE works; able to implement and inspect the hardware results of QPE.

28.2 General Controlled Unitary Gate

In the previous chapter, we introduced the single-qubit arbitrary unitary gate, $\boldsymbol{U_{\theta,\phi,\lambda}}$. It describes how a single qubit rotates on the **Bloch sphere**. We have also emphasized that in order to map the qubit to the Bloch sphere, we discarded the **global phase** information. This is fine because, for a single qubit, the global phase has no physical significance. The rotation on a Bloch sphere can be described by the $\boldsymbol{U_{\theta,\phi,\lambda}}$ defined in Eq. (27.9) and is repeated here for convenience.

$$\boldsymbol{U_{\theta,\phi,\lambda}} = \begin{pmatrix} \cos\frac{\theta}{2} & -e^{i\lambda}\sin\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2} & e^{i(\lambda+\phi)}\cos\frac{\theta}{2} \end{pmatrix} \tag{28.1}$$

This gate represents all possible one-qubit gates as long as we do not care about the global phase of the state. This is not a problem for a single qubit. But it is a problem when multiple qubits are considered. When more qubits are involved, while the global phase of the whole system does not matter, the "global phases" of the individual qubits matter. The global phases of the individual qubits determine the relative phase between different qubits and can result in constructive or destructive interferences that are important mechanisms in quantum computing (e.g. we have

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_28

seen this in QFT). Therefore, Eq. (28.1) is not enough to describe all unitary rotation *when we are not allowed to neglect the global phase of a single qubit*. We can extend the concept, so it will be able to represent any unitary rotation of a qubit by adding a phase factor, γ, as the following:

$$\boldsymbol{U_{\theta,\phi,\lambda,\gamma}} = e^{i\gamma} \begin{pmatrix} \cos\frac{\theta}{2} & -e^{i\lambda}\sin\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2} & e^{i(\lambda+\phi)}\cos\frac{\theta}{2} \end{pmatrix} \tag{28.2}$$

The θ, ϕ, and λ still have the same meaning as in $\boldsymbol{U_{\theta,\phi,\lambda}}$. You can see that when we only have $\boldsymbol{U_{\theta,\phi,\lambda}}$, for some matrices, we are missing the $e^{i\gamma}$. This causes an error when it is applied to a vector by omitting $e^{i\gamma}$. But since for a single-qubit case, missing a phase does not matter, we were fine with that.

Example 28.1 Represent the gate $\begin{pmatrix} i & 0 \\ 0 & -i \end{pmatrix}$ in the form of $\boldsymbol{U_{\theta,\phi,\lambda,\gamma}}$.

$$\begin{pmatrix} i & 0 \\ 0 & -i \end{pmatrix} = i \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = e^{i\pi/2} \begin{pmatrix} \cos\frac{0}{2} & -e^{i\pi}\sin\frac{0}{2} \\ e^{i0}\sin\frac{0}{2} & e^{i(\pi+0)}\cos\frac{0}{2} \end{pmatrix} \tag{28.3}$$

Therefore, $\boldsymbol{U_{0,0,\pi,\pi/2}}$ is the correct representation. When I apply this gate to $\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$, we get

$$\boldsymbol{U_{0,0,\pi,\pi/2}} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} i & 0 \\ 0 & -i \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = i \begin{pmatrix} \alpha \\ -\beta \end{pmatrix} \tag{28.4}$$

If we use $\boldsymbol{U_{0,0,\pi}}$ instead, a phase factor of $e^{i\gamma} = e^{i\pi/2} = i$ is missing. When we apply $\boldsymbol{U_{0,0,\pi}}$ to $\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$, it becomes

$$\boldsymbol{U_{0,0,\pi}} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha \\ -\beta \end{pmatrix} \tag{28.5}$$

The rotated state vectors are the same except differ by the phase factor, i.

Now we want to construct a general version of a **controlled unitary gate**. Since two qubits are involved, we cannot ignore the global phase of each qubit anymore. Therefore, the controlled unitary gate must use all four parameters, and its symbol is $\boldsymbol{C-U_{\theta,\phi,\lambda,\gamma}}$. It has this matrix:

$$\boldsymbol{C-U_{\theta,\phi,\lambda,\gamma}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{i\gamma}\cos\frac{\theta}{2} & -e^{i(\lambda+\gamma)}\sin\frac{\theta}{2} \\ 0 & 0 & e^{i(\phi+\gamma)}\sin\frac{\theta}{2} & e^{i(\lambda+\phi+\gamma)}\cos\frac{\theta}{2} \end{pmatrix} \tag{28.6}$$

Here we assume the MSB is the control qubit and the LSB is the target qubit. We can find this matrix by using the definition. A $\boldsymbol{C-U}$ gate means that if the control qubit in the basis vector is 0, it does nothing to the target qubit, and if the control qubit is 1, it will apply the unitary gate. Therefore, the definition can be written as

$$\begin{aligned}
\boldsymbol{C-U}_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\lambda},\boldsymbol{\gamma}}\,|00\rangle &= \boldsymbol{I}\,|0\rangle \otimes \boldsymbol{I}\,|0\rangle = |00\rangle \\
\boldsymbol{C-U}_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\lambda},\boldsymbol{\gamma}}\,|01\rangle &= \boldsymbol{I}\,|0\rangle \otimes \boldsymbol{I}\,|1\rangle = |01\rangle \\
\boldsymbol{C-U}_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\lambda},\boldsymbol{\gamma}}\,|10\rangle &= \boldsymbol{I}\,|1\rangle \otimes \boldsymbol{U}_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\lambda},\boldsymbol{\gamma}}\,|0\rangle \\
&= |1\rangle \left(e^{i\gamma} \cos\frac{\theta}{2}\,|0\rangle + e^{i(\phi+\gamma)} \sin\frac{\theta}{2}\,|1\rangle \right) \\
&= e^{i\gamma} \cos\frac{\theta}{2}\,|10\rangle + e^{i(\phi+\gamma)} \sin\frac{\theta}{2}\,|11\rangle \\
\boldsymbol{C-U}_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\lambda},\boldsymbol{\gamma}}\,|11\rangle &= \boldsymbol{I}\,|1\rangle \otimes \boldsymbol{U}_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\lambda},\boldsymbol{\gamma}}\,|1\rangle \\
&= |1\rangle \left(-e^{i(\lambda+\gamma)} \sin\frac{\theta}{2}\,|0\rangle + e^{i(\lambda+\phi+\gamma)} \cos\frac{\theta}{2}\,|1\rangle \right) \\
&= -e^{i(\lambda+\gamma)} \sin\frac{\theta}{2}\,|10\rangle + e^{i(\lambda+\phi+\gamma)} \cos\frac{\theta}{2}\,|11\rangle
\end{aligned} \tag{28.7}$$

where we use the fact that e.g. $|01\rangle = |0\rangle \otimes |1\rangle$ and then apply the operators to each qubit before performing the tensor products on the right-hand side, and Eq. (28.2) is used in the last two terms with $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. We can show that Eq. (28.6) is correct based on the definitions on Eq. (28.7) by using Eq. (12.17).

Example 28.2 Find the element $(e_{2,2})$ of the second row and second column (counting from 0-th) of $\boldsymbol{C-U}_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\lambda},\boldsymbol{\gamma}}$.

The element corresponds to $\langle 10|$ (row) and $|10\rangle$ (column). Using Eq. (28.7),

$$e_{2,2} = \langle 10|\,\boldsymbol{C-U}_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\lambda},\boldsymbol{\gamma}}\,|10\rangle = e^{i\gamma} \cos\frac{\theta}{2} \tag{28.8}$$

where we apply $\langle 10|$ to line 3 in Eq. (28.7), and we use the orthonormal property of the basis vectors that $\langle 10|10\rangle = 1$ and $\langle 10|11\rangle = 0$. This is the same as in Eq. (28.6).

Finally, we may also construct the matrix in Eq. (28.6) using this equation.

$$\boldsymbol{C-U}_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\lambda},\boldsymbol{\gamma}} = |0\rangle\langle 0| \otimes \boldsymbol{I} + |1\rangle\langle 1| \otimes \boldsymbol{U}_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\lambda},\boldsymbol{\gamma}} \tag{28.9}$$

It is easy to see the result if we recognize that $|0\rangle\langle 0| = \begin{pmatrix} 1 \\ 0 \end{pmatrix}\begin{pmatrix} 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ and $|0\rangle\langle 1| = \begin{pmatrix} 0 \\ 1 \end{pmatrix}\begin{pmatrix} 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$. We then perform the tensor products accordingly.

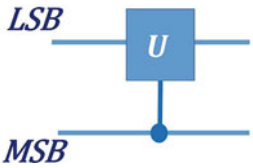


Fig. 28.1 Symbol of a $C - U_{\theta,\phi,\lambda,\gamma}$ gate with the MSB as the control qubit

This equation says that we should perform identity operation to the target qubit if the control qubit is 0 and apply the unitary gate to the target qubit if the control qubit is 1. We will do this in the problems. Figure 28.1 shows the gate symbol.

28.3 Quantum Phase Estimation

The purpose of the **Quantum Phase Estimation (QPE)** algorithm is to estimate the phase, $2\pi\psi$, of the eigenvalue of a unitary $m \times m$ matrix, $\boldsymbol{U}$. Since it is a unitary matrix, its eigenvalue must have a magnitude of one. This is because we can always transform the matrix into a basis formed by its eigenvectors (as the basis states), and its diagonal elements are just its eigenvalues (Eq. (9.13)). And since the columns of the unitary matrix must be normalized (Eq. (9.23)), then the eigenvalues must have a magnitude of one and be in the form of $e^{i2\pi\psi}$. We choose to write its phase as $2\pi\psi$ instead of a single variable is just for convenience in the derivation.

Let us look at a QPE example for a 2×2 matrix before discussing the general algorithm.

28.3.1 QPE for a 2 × 2 Matrix

Assume we are given a phase shift gate with $\phi = \pi$, $\boldsymbol{U_{PS,\pi}}$ (Eq. (16.9)), which is just a Z-gate, $\boldsymbol{Z}$.

$$\begin{aligned} \boldsymbol{U_{PS,\pi}} &= \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi} \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \\ &= \begin{pmatrix} e^{i2\pi 0} & 0 \\ 0 & e^{i2\pi\frac{1}{2}} \end{pmatrix} \end{aligned} \tag{28.10}$$

We see that it is already in its diagonal form, and therefore, the elements are its eigenvalues, 1 and -1, which have phases of 0 and π, respectively. We also express

them in the form of $e^{i2\pi\psi}$. So their ψ's are $\psi_0 = 0$ and $\psi_1 = \frac{1}{2}$, respectively. Each of the eigenvalues corresponds to an eigenvector, and they are $|e_0\rangle = \begin{pmatrix}1\\0\end{pmatrix}$ and $|e_1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$, respectively. You may refer to Chap. 9 to review how to find the eigenvectors and eigenvalues of a matrix. We can prove that they are the correct eigenvectors by showing, by definition,

$$\begin{aligned} \boldsymbol{U_{PS,\pi}}\,|e_0\rangle &= \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix} = 1\begin{pmatrix}1\\0\end{pmatrix} = 1\,|e_0\rangle \\ \boldsymbol{U_{PS,\pi}}\,|e_1\rangle &= \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix} = -1\begin{pmatrix}0\\1\end{pmatrix} = -1\,|e_1\rangle \end{aligned} \tag{28.11}$$

The goal of the QPE is to find ψ_0 and ψ_1 for the given $\boldsymbol{U_{PS,\pi}}$.

Figure 28.2 shows the quantum circuit for finding the phase of the Z-gate. *To be exact, it finds ψ instead of $2\pi\psi$.* There are two groups of inputs. The less significant qubits (called the *b-register*) take the eigenvector of the Z-gate. Since the Z-gate is a 2×2 matrix, we only need 1-qubit to represent its eigenvector. In the example, we assume the input is $|e_1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$. The other group of inputs (called the *c-register*) contains the more significant qubits. In this example, we use two qubits. These qubits will be used to store a value related to ψ, which is $2^n\psi$, where n is the number of the qubits in the c-register. In this case, it is $2^n\psi = 4\psi$. The inputs to the c-register are the ground state $|0\rangle_2$. This is a $2 + 1 = 3$-qubit circuit. Therefore, it starts with

$$|\xi_0\rangle = |0\rangle\,|0\rangle\,|e_1\rangle \tag{28.12}$$

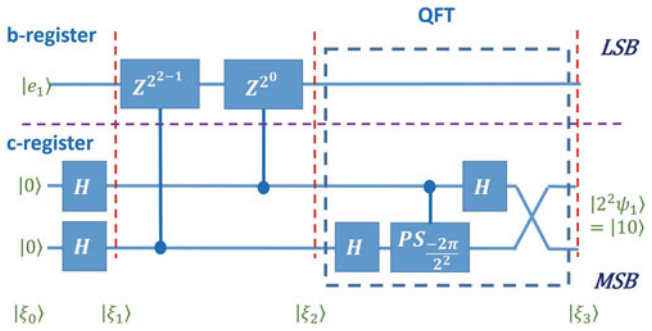


Fig. 28.2 The QPE circuit for finding the phase of the second eigenvalue of the Z-gate ($\boldsymbol{U_{PS,\pi}}$). Note that the bottom is the MSB

The circuit begins with a two-dimensional Hadamard gate ($\boldsymbol{H} \otimes \boldsymbol{H}$) to create a superposition on the c-register. That is

$$\begin{aligned}
|\xi_1\rangle &= \boldsymbol{H} \otimes \boldsymbol{H} \otimes \boldsymbol{I}(|0\rangle\,|0\rangle\,|e_1\rangle) \\
&= \boldsymbol{H}\,|0\rangle \otimes \boldsymbol{H}\,|0\rangle \otimes \boldsymbol{I}\,|e_1\rangle \\
&= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\,|e_1\rangle \\
&= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)\,|e_1\rangle \\
&= \frac{1}{2}(|00\rangle\,|e_1\rangle + |01\rangle\,|e_1\rangle + |10\rangle\,|e_1\rangle + |11\rangle\,|e_1\rangle \qquad (28.13)
\end{aligned}$$

After that, controlled gates are used. The type of controlled gate used is *related* to the Z-gate, of which we are finding the eigenvalue phase. If we were to find that of another gate/unitary matrix (e.g. the NOT gate), we need to use the controlled gate related to that new gate (e.g. CNOT gate). I say it is *related* to the Z-gate because, besides controlled-Z-gate, it also uses controlled-Z^k gates, where $k = 2^l$, with l running from 0 to $n - 1$. Again n is the number of qubits in the c-register. A controlled-Z^k gate means that, for a basis state, if the control qubit is 0, it does nothing to the target qubit. If the control qubit is 1, it will apply Z^k gate to the target qubit. The control qubit is one of the qubits in the c-register, and the target qubits are those of the b-register. Each qubit in the c-register acts as the control bit of one of the controlled gates. A Z^k gate is equivalent to applying Z-gate k times. In this case, l runs from 0 to 1 and thus k can be 1 or 2. Therefore, qubit 0 of the c-register is connected to a controlled-Z^k with $l = 0$, and qubit 1 of the c-register is connected to a controlled-Z^k with $l = 1$. In other words, qubit 0 (the less significant bit) of the c-register is the control qubit of a controlled-Z^{2^0} gate, i.e. a controlled-Z-gate. And qubit 1 (the more significant bit) of the c-register is the control qubit of a controlled-Z^{2^1} gate, i.e. a controlled-Z^2 gate.

What happens to the b-register qubit when the controlled-Z^k gates are applied? Note that the b-register has the eigenvector of Z. Therefore, the operation will give the eigenvalue of $\boldsymbol{Z}$ because of Eq. (28.11). That is, $\boldsymbol{U_{PS,\pi}}\,|e_1\rangle = \boldsymbol{Z}\,|e_1\rangle = e^{i2\pi\frac{1}{2}}\,|e_1\rangle = -1\,|e_1\rangle$. And if it is $\boldsymbol{Z}^2$, it becomes $(e^{i2\pi\frac{1}{2}})^2\,|e_1\rangle = |e_1\rangle$ as $\boldsymbol{Z}$ is applied twice. However, this only happens if the corresponding control qubit is 1. Therefore, the n controlled gates produce the following results:

$$\begin{aligned}
|\xi_2\rangle &= \frac{1}{2}(|00\rangle\,|e_1\rangle + |01\rangle\,\boldsymbol{Z}\,|e_1\rangle + |10\rangle\,\boldsymbol{Z}^2\,|e_1\rangle + |11\rangle\,\boldsymbol{Z}^2\boldsymbol{Z}\,|e_1\rangle \\
&= \frac{1}{2}(|00\rangle\,\boldsymbol{Z}^0\boldsymbol{Z}^0\,|e_1\rangle + |01\rangle\,\boldsymbol{Z}^0\boldsymbol{Z}^1\,|e_1\rangle + |10\rangle\,\boldsymbol{Z}^2\boldsymbol{Z}^0\,|e_1\rangle + |11\rangle\,\boldsymbol{Z}^2\boldsymbol{Z}^1\,|e_1\rangle \\
&= \frac{1}{2}(|00\rangle\,\boldsymbol{Z}^{0+0}\,|e_1\rangle + |01\rangle\,\boldsymbol{Z}^{0+1}\,|e_1\rangle + |10\rangle\,\boldsymbol{Z}^{2+0}\,|e_1\rangle + |11\rangle\,\boldsymbol{Z}^{2+1}\,|e_1\rangle
\end{aligned}$$

$$= \frac{1}{2}(|0\rangle \boldsymbol{Z}^0 |e_1\rangle + |1\rangle \boldsymbol{Z}^1 |e_1\rangle + |2\rangle \boldsymbol{Z}^2 |e_1\rangle + |3\rangle \boldsymbol{Z}^3 |e_1\rangle$$

$$= \frac{1}{2}\sum_{j=0}^{3} |j\rangle \boldsymbol{Z}^j |e_1\rangle \tag{28.14}$$

In the first line, $\boldsymbol{Z}$ is applied to the b-register if qubit 0 of the c-register is one, and $\boldsymbol{Z}^2$ is applied to the b-register if qubit 1 of the c-register is one. This is based on the definition of the controlled gates. In the second line, I also added $\boldsymbol{Z}^0$ that is just an identity gate when the control qubits are 0. In the third line, we group the $\boldsymbol{Z}$'s so we can see the sum of their exponents for each basis vector. And in the fourth line, I write the basis vectors in decimal. *Now we clearly see that the action of all controlled gates is equivalent to applying $\boldsymbol{Z}^j$ gate on the b-register if the c-register is $|j\rangle$*. This is a very important feature we use very often in other quantum computing algorithms.

And since $|e_1\rangle$ is an eigenvector of $\boldsymbol{Z}^j$, $\boldsymbol{Z}^j |e_1\rangle = (e^{i2\pi\frac{1}{2}})^j |e_1\rangle$. Therefore,

$$\begin{aligned} |\xi_2\rangle &= \frac{1}{2}\sum_{j=0}^{3} |j\rangle \boldsymbol{Z}^j |e_1\rangle \\ &= \frac{1}{2}\sum_{j=0}^{3} |j\rangle (e^{i2\pi\frac{1}{2}})^j |e_1\rangle \\ &= \frac{1}{2}\sum_{j=0}^{3} (e^{i2\pi\frac{1}{2}})^j |j\rangle |e_1\rangle \end{aligned} \tag{28.15}$$

I did not simplify $e^{i2\pi\frac{1}{2}}$, which is just -1 because I want to keep it in this form so you can appreciate the constructive and destructive interference in the following QFT step.

The next step is to apply QFT. Note that some other sources may use IQFT, and this is just because it uses a different definition of QFT than ours (see Chap. 26). What really important is that we need to use the matrix for rotating the vector instead of the basis and the matrix is shown in Eq. (25.12). The definition of QFT (Eq. (25.13)) is repeated here for convenience,

$$\boldsymbol{U_{QFT}} |j\rangle = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} \omega^{-kj} |k\rangle \tag{28.16}$$

We only apply it to the c-register and keep the b-register intact. Therefore, we will apply $\boldsymbol{U_{QFT}} \otimes \boldsymbol{I}$ to $|\xi_2\rangle$, and we have

$$
\begin{aligned}
|\xi_3\rangle &= \boldsymbol{U_{QFT}} \otimes \boldsymbol{I} \left(\frac{1}{2} \sum_{j=0}^{3} (e^{i2\pi \frac{1}{2}})^j \, |j\rangle \, |e_1\rangle \right) \\
&= \frac{1}{2} \sum_{j=0}^{3} (e^{i2\pi \frac{1}{2}})^j (\boldsymbol{U_{QFT}} \, |j\rangle) \otimes \boldsymbol{I} \, |e_1\rangle \\
&= \frac{1}{2} \sum_{j=0}^{3} (e^{i2\pi \frac{1}{2}})^j \left(\frac{1}{\sqrt{4}} \sum_{k=0}^{3} \omega^{-kj} \, |k\rangle \right) |e_1\rangle \\
&= \frac{1}{4} \sum_{j=0}^{3} \sum_{k=0}^{3} (e^{i2\pi \frac{1}{2}})^j \omega^{-kj} \, |k\rangle \, |e_1\rangle \\
&= \frac{1}{4} \sum_{j=0}^{3} \sum_{k=0}^{3} (e^{i \frac{2\pi}{4} 2})^j (e^{i2\pi/4})^{-kj} \, |k\rangle \, |e_1\rangle \\
&= \frac{1}{4} \sum_{j=0}^{3} \sum_{k=0}^{3} \omega^{j(2-k)} \, |k\rangle \, |e_1\rangle
\end{aligned} \tag{28.17}
$$

Although the math looks complex, it is actually simple. The first three lines just apply $\boldsymbol{U_{QFT}}$ to the c-register in $|\xi_2\rangle$, use the linear property of quantum mechanics, and then apply Eq. (28.16), the definition of QFT. The last two lines use the definition of the n-th root of unity, ω (Chap. 25). Finally, we group the terms. Note that the b-register is no longer useful now. *It is completely disentangled with the rest of the qubit, and it can be factorized out in the tensor product.*

If we perform a measurement on the c-register, we will get one of the $|k\rangle$'s. However, if we rewrite Eq. (28.17) by shuffling the summations, we have

$$
|\xi_3\rangle = \frac{1}{4} \sum_{k=0}^{3} \left(\sum_{j=0}^{3} \omega^{j(2-k)} \right) |k\rangle \, |e_1\rangle \tag{28.18}
$$

We see that each $|k\rangle$ has a coefficient of $\sum_{j=0}^{3} \omega^{j(2-k)}$, and this is a summation of the n-th roots of unity. This term is 0 unless $(2-k) = 0$ due to the **constructive interference** (Eq. (25.7)). Here j is the m in Eq. (25.7), and $2-k$ is the q in Eq. (25.7). Therefore, if we perform the measurement, we will only measure $|k\rangle$ if $2-k=0$. The basis state we will measure is *always*

$$
|k\rangle = |2\rangle_{10} = |10\rangle_2 \tag{28.19}
$$

Here we write the *ket* in both decimal and binary notations.

What is the meaning of $k = 2$? Let us inspect carefully how we obtain $\omega^{j(2-k)}$ in line 6 of Eq. (28.17). It came from $(e^{i2\pi\frac{1}{2}})^j\omega^{-kj}$ (line 4). The "2" in $\omega^{j(2-k)}$ is obtained by multiplying $\frac{1}{2}$ by 4 (line 5). And 4 is the $N = 2^2$ in the N-th root of unity. Note that $\frac{1}{2}$ is the ψ in the eigenvalue of the eigenvector $|e_1\rangle$, and 4 is $2^n = 2^2$, where n is the number of qubits in the c-register. Therefore, the $k = 2$ we obtained is due to the ψ ($\frac{1}{2}$) being multiplied by 2^n ($2^2 = 4$). From this, we can deduce that in a general QPE, if we measure $|k\rangle$ in a n-qubit c-register, and if the input to the b-register is the eigenvector corresponding to the eigenvalue, $e^{i2\pi\psi}$, the phase is given by

$$2\pi\psi = 2\pi k/2^n \tag{28.20}$$

28.3.2 Implementation on IBM-Q

Figure 28.3 shows the implementation on IBM-Q. $|e_1\rangle$ is created by using a NOT gate. You can see the controlled-Z-gate that is a vertical line with two dots. There is no controlled-Z^2 gate because $\boldsymbol{Z}^2 = \boldsymbol{I}$. The $\boldsymbol{I}$ gate in the circuit is there just to align the circuits. The QFT circuit is the one we used in Fig. 26.5 but only 2 qubits. You can see that the output is 100% $|100\rangle$. The first two qubits are for the c-register. Therefore, $|k\rangle = |10\rangle = |2\rangle$. That means $2\pi\psi = 2\pi k/2^n = 2\pi 2/2^2 = \pi$, which is the phase of the corresponding eigenvalue, -1. Note that we did not measure the b-register (the LSB); therefore, it is 0 by default in the classical register.

28.3.3 General QPE Circuit

Figure 28.4 shows the general QPE circuit. It is just a simple extension of the circuit we have discussed. We note that the b-register can be m-qubit that is required if we are finding the eigenvalue phase in a $m \times m$ matrix case. In that case, the controlled

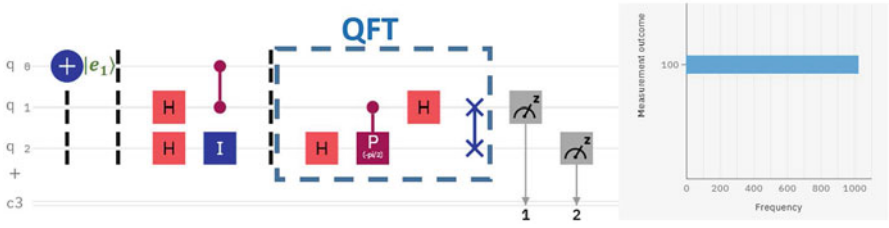


Fig. 28.3 Implementation of the QPE circuit in Fig. 28.2. Note that the bottom is the MSB

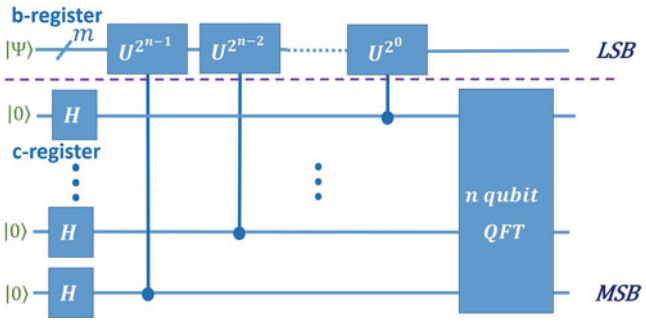


Fig. 28.4 General implementation of the QPE circuit

gate is a $m+1$-qubit gate (including the control bit). The c-register can be n qubits. n determines the accuracy of the phase it can estimate. Referring to Eq. (28.20), larger n gives a higher resolution. The resolution is $1/2^n$, when $k=1$. Usually, the ψ is not a fraction nor an integer as in the case we have discussed. Therefore, we cannot get the exact value, and thus it is an "estimation." In summary, the QPE has $m+n$ qubits.

We spent a lot of time discussing the controlled arbitrary unitary gate at the beginning of the chapter. It is not used so far. But it is useful if the matrix you want to study is not a simple gate. You need to use them if you want to implement the QPE for an arbitrary matrix.

The QPE seems not to be useful as we need to know the eigenvector (as input to the b-register) to find its phase. However, every vector is a linear superposition of the eigenvectors. Due to the linearity of quantum mechanics, their phases can be estimated at the same time. If we use it correctly, we can use it to solve other more important problems such as in the HHL algorithm for solving systems of linear equations.

Finally, we can derive the equation for the general QPE like what we did for the 2-qubit case. Please try so by following the steps earlier. I did not do simplification and kept ψ and n or 2^n in the derivation, so it should be relatively straightforward to generalize from 2-qubit to n-qubit. Of course, the phase estimated is given in Eq. (28.20).

28.4 Summary

In this chapter, we discussed how to construct a general controlled unitary gate. We note that it has four parameters in order not to lose the phase information of the rotation. If one wants to implement the QPE for an arbitrary matrix, we need to use it. QPE is used to estimate the phase of the eigenvalue of a matrix. We need to use its eigenvector as the input to the b-register, and the c-register will give a

number related to its phase. This is achieved by using constructive and destructive interferences through the QFT.

Problems

28.1 Unitary Gate Construction

Represent the gate $\frac{1}{\sqrt{2}}\begin{pmatrix} -i & -i \\ -i & i \end{pmatrix}$ in the form of $\boldsymbol{U}_{\theta,\phi,\lambda,\gamma}$.

28.2 General Controlled Unitary Gate

Using the method in Example 28.2, show that other non-zero and non-unity elements in Eq. (28.6) are correct.

28.3 General Controlled Unitary Gate Matrix

Use matrix multiplication to prove Eq. (28.9).

28.4 QPE Calculation

In the text, we used $|e_1\rangle$ as the input to the b-register. Now, derive the QPE result for $|e_0\rangle$ of the Z-gate.

28.5 QPE on IBM-Q

Repeat what we did in the text, but use 3 qubits ($n = 3$) for the c-register. Hints: You need to create a $Z^{2^{3-1}}$ gate and its controlled version and simplify it first. The answer should be $|1000\rangle$.

28.6 Derive the General QPE Algorithm

Show that if c-register has n qubits, the answer is that in Eq. (28.20).

Chapter 29
Shor's Algorithm

29.1 Learning Outcomes

Understand the role of Shor's algorithm in the prime integer factorization process and its relationship to encryption; able to derive and explain the equations in the Shor's algorithm; appreciate that Shor's algorithm is the type of quantum algorithm that does not give deterministic results and needs trials and errors, but the results can be verified easily.

29.2 Background

29.2.1 Encryption

Nowadays, data encryption is an indispensable part of our daily life. In the encryption process, we usually generate a very big integer number, N, using two large prime numbers, P and Q, such that $N = P \times Q$. N is shared with the public, and only the persons who have a key can find P and Q quickly. Otherwise, to break the encryption (e.g. in RSA), one needs to be able to perform **prime integer factorization** of N, i.e. to find the values of P and Q. This is computationally intensive. There is no known method to do it quickly. Internet security relies almost entirely on the fact/expectation that no one can perform the prime integer factorization efficiently.

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_29

29.2.2 Period Finding

It turns out that the prime integer factorization of N is related to the **period funding** of a certain function. More precisely, if I want to factorize the number N, mathematically, it is equivalent to finding the period, r, of the following function (we will not prove this):

$$f_{a,N}(x) \equiv a^x \bmod N \tag{29.1}$$

where a is a parameter and x is an independent variable. Therefore, $f_{a,N}(x)$ is a function depending on the parameters, a and N, and the variable x. The variable x is also an integer. This is a function using the **modulo operator**, mod, which returns the *remainder* of a division. For example, $7 \mod 2 = 1$ because $7 = 3 \times 2 + 1$. When 7 is divided by 2, the remainder is 1. Therefore, the function returns the remainder of a^x divided by N.

Example 29.1 Assume we want to factorize $N = 21$. Let us pick $a = 11$ (we will discuss how to pick a later). Find the period, r, of the function $f_{a,N}(x) = f_{11,21}(x)$.

We will find it by substituting x beginning from 0.

$$\begin{aligned}
f_{11,21}(0) &\equiv 11^0 \bmod 21 = 1 \quad as \quad 11^0 = 0 \times 21 + 1 \\
f_{11,21}(1) &\equiv 11^1 \bmod 21 = 11 \quad as \quad 11^1 = 0 \times 21 + 11 \\
f_{11,21}(2) &\equiv 11^2 \bmod 21 = 16 \quad as \quad 11^2 = 5 \times 21 + 16 \\
f_{11,21}(3) &\equiv 11^3 \bmod 21 = 8 \quad as \quad 11^3 = 63 \times 21 + 8 \\
f_{11,21}(4) &\equiv 11^4 \bmod 21 = 4 \quad as \quad 11^4 = 697 \times 21 + 4 \\
f_{11,21}(5) &\equiv 11^5 \bmod 21 = 2 \quad as \quad 11^5 = 7669 \times 21 + 2 \\
f_{11,21}(6) &\equiv 11^6 \bmod 21 = 1 \quad as \quad 11^6 = 84360 \times 21 + 1
\end{aligned} \tag{29.2}$$

We see that the answer repeats with a period of 6 ($f_{11,21}(0) = f_{11,21}(0 + 6)$). Therefore, $r = 6$.

Note that finding the period is computationally intensive.

29.2.3 Prime Integer Factorization

Once we find the period r of $f_{a,N}(x)$, we can perform factorization in a quick way. Let us consider the following derivation. We do not assume you have a number theory background. So please just trust the derivation. It is not difficult to follow if you treat the modulo operations as if they were not there and just use the rules in the regular algebra.

Due to the periodicity, r,

$$f_{a,N}(0) = a^0 \bmod N = 1 \bmod N = 1$$
$$f_{a,N}(r) = f_{a,N}(r+0) = a^r \bmod N = f_{a,N}(0) = 1 \tag{29.3}$$

where we used the fact that $1 \bmod N = 1$ (i.e. the remainder of 1 divided by N is 1 as long as $N > 1$). We can use the rules in algebra and rearrange the second equation, and we have

$$a^r - 1 \bmod N = 0$$
$$(a^{\frac{1}{2}r-1})(a^{\frac{1}{2}r+1}) \bmod N = 0 \tag{29.4}$$

where we use the rule like in the algebra that $(a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1) = a^r - 1$.

Then we will use a theorem that *if r is even and $a^{\frac{r}{2}} + 1 \bmod N \neq 0$, then the prime factors are given by*

$$P = gcd(a^{\frac{r}{2}} + 1, N)$$
$$Q = gcd(a^{\frac{r}{2}} - 1, N) \tag{29.5}$$

where $gcd(x, y)$ refers to the greatest common divisor of x and y. For example, $gcd(10, 20) = 10$ because 10 is the greatest divisor of 10 and 20. Another example is $gcd(21, 14) = 7$ because 7 is the largest number and can divide (without remainder) 21 and 14.

If r is not even or $a^{\frac{r}{2}} + 1 \bmod N = 0$, we need to try another a. There is a theory that says that 50% of the a will satisfy this requirement. Therefore, we have a high chance to be able to use Eq. (29.5) to do factorization.

Therefore, once the period is found, the problem is converted into a problem of finding the gcd of the numbers that can be done effectively e.g. using the **Euclidean algorithm**.

Example 29.2 Let us continue on the previous example. We know that $r = 6$ now. Find P and Q.

$r = 6$ is even. We still need to check if $a^{\frac{r}{2}} + 1 \bmod N \neq 0$.

$$a^{\frac{r}{2}} + 1 \bmod N = 11^{\frac{6}{2}} + 1 \bmod 21 = 1332 \bmod 21 = 9 \neq 0 \tag{29.6}$$

Therefore, we can go ahead to find the gcd. We have

$$gcd(a^{\frac{r}{2}} + 1, N) = gcd(11^{\frac{6}{2}} + 1, 21) = gcd(1332, 21) = 3$$
$$gcd(a^{\frac{r}{2}} - 1, N) = gcd(11^{\frac{6}{2}} - 1, 21) = gcd(1330, 21) = 7 \tag{29.7}$$

Therefore, $21 = 3 \times 7$.

29.3 Shor's Algorithm

In the previous section, we show that to break the classical encryptions, we need to perform prime integer factorization of a big number ($N = P \times Q$). We can do this by first finding the period, r, of the function $f_{a,N}(x) \equiv a^x \mod N$. Once the period is found to be an even number and $a^{\frac{r}{2}} + 1 \mod N \neq 0$, we can find P and Q easily by finding the gcd of some numbers. We need to guess the parameter a so we can use the theory, but the chance of picking the right a is pretty high (50%).

The most time-consuming part of this procedure is in the **period finding** process. The quantum computing part of the **Shor's algorithm** is used to speed up this process. Classically, the period finding process complexity is $\mathcal{O}(r)$ (refer to Sect. 23.2.1 for the meaning of computational complexity). With Shor's algorithm, it becomes $\mathcal{O}(log(r))$ and thus it is an exponential speedup.

Figure 29.1 shows the quantum circuit of Shor's algorithm. There are 4 parts in Shor's algorithm. For a function $f(x) = f(x + r)$:

1. Create a superposition of exponentially many arguments.
2. Evaluate the function on a superposition of exponentially many arguments.
3. Compute a parallel Fourier transform on the superposition using QFT.
4. Sample the Fourier power spectrum to obtain the function's period.

For any N, we can always find n such that $2^n \geq N$. For example, if $N = 21$, we need $n = 5$ so that $2^n = 32 \geq 21$. In Shor's algorithm, we need $3n$ qubits. They are grouped into two registers. The register with the more significant qubits has $2n$ qubits, and the other one has n qubits as shown in Fig. 29.1.

Therefore, the quantum state starts with

$$|\xi_0\rangle = |0\rangle^{\otimes 2n} |0\rangle^{\otimes n} \tag{29.8}$$

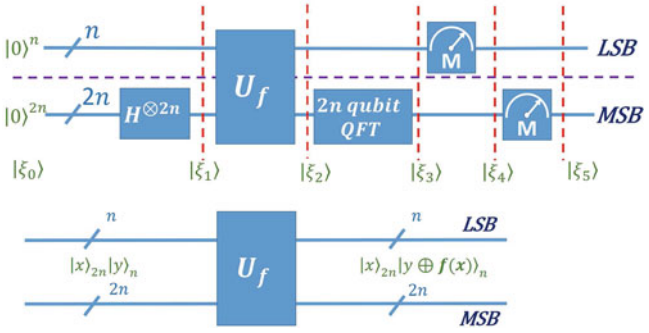


Fig. 29.1 Top: quantum circuit for Shor's algorithm. Bottom: XOR quantum oracle circuit displayed for convenience. This is the same as the top figure in Fig. 22.1 by setting $m \to n$ and $n \to 2n$

Step 1

After that, a $2n$-qubit Hadamard gate, $\boldsymbol{H}^{\otimes 2n}$, is applied to the first register, while the second register is kept intact.

$$\begin{aligned}
|\xi_1\rangle &= (\boldsymbol{H}^{\otimes 2n}\boldsymbol{I}^{\otimes n})(|0\rangle^{\otimes 2n}\,|0\rangle^{\otimes n}) \\
&= (\boldsymbol{H}^{\otimes 2n}\,|0\rangle^{\otimes 2n}) \otimes (\boldsymbol{I}^{\otimes n}\,|0\rangle^{\otimes n}) \\
&= (\frac{1}{2^{\frac{2n}{2}}}\sum_{x=0}^{2^{2n}-1}|x\rangle_{2n}) \otimes |0\rangle^{\otimes n} \\
&= \frac{1}{2^n}\sum_{x=0}^{2^{2n}-1}|x\rangle_{2n}\,|0\rangle_n
\end{aligned} \tag{29.9}$$

A n-qubit identity gate, $\boldsymbol{I}^{\otimes n}$, is also applied at the same time to the last n-qubit register. I added parentheses to separate the operators and states in the first line. Then parentheses are used to separate the two registers in the second line. In the third line, Eq. (17.16) is used. Remember that each 1-qubit Hadamard gate transforms $|0\rangle$ into $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, and the products of the n transformed qubits become the summation in the third line. Note that the x in each $|x\rangle$ is written in decimal form, and it has $2n$ qubits as indicated in the subscript. *This step creates a superposition of exponentially many basis states* (2^{2n}) *to be evaluated in parallel in the next step*.

Step 2

Since the Shor's algorithm involves a function $f_{a,N}(x) \equiv a^x \bmod N$, we naturally expect we need a quantum oracle that has the property of $f_{a,N}(x)$. The quantum oracle has $3n$ qubits. As shown in Fig. 29.1, it transforms any basis vector of the $3n$ qubits by

$$\boldsymbol{U}_f\,|x\rangle_{2n}\,|y\rangle_n = |x\rangle_{2n}\,|y \oplus f(x)\rangle_n \tag{29.10}$$

which is just the regular behavior of an XOR quantum oracle. This is the same as the top figure in Fig. 22.1 by setting $m \to n$ and $n \to 2n$. The most significant $2n$ qubits are the *query qubits*, and the least significant n qubits are the *auxiliary qubits*. The $f(x)$ here is the $f_{a,N}(x)$ function of which the period is to be found. Let us apply the quantum oracle to $|\xi_1\rangle$. We will use Eq. (29.10) with $y = 0$.

$$\begin{aligned}
|\xi_2\rangle &= \boldsymbol{U}_f\,|\xi_1\rangle \\
&= \boldsymbol{U}_f(\frac{1}{2^n}\sum_{x=0}^{2^{2n}-1}|x\rangle_{2n})\,|0\rangle_n \\
&= \frac{1}{2^n}\sum_{x=0}^{2^{2n}-1}|x\rangle_{2n}\,|0 \oplus f(x)\rangle_n \\
&= \frac{1}{2^n}\sum_{x=0}^{2^{2n}-1}|x\rangle_{2n}\,|f(x)\rangle_n
\end{aligned} \tag{29.11}$$

Note that the classical XOR operation of the *label* (i.e. $0 \oplus f(x)$) of the least significant n qubits is relatively straightforward because $y = 0$, and thus the least significant n qubits become just the corresponding values of $f(x)$. For example, if $n = 3$ and $f(x) = 5$, then $|0 \oplus f(x)\rangle_n = |5\rangle_3 = |101\rangle$. This step *evaluates the exponentially many basis vectors with label x all at one time in parallel*. That is, it evaluates $f(0)$ up to $f(2^{2n} - 1)$ in one execution. *This is where the quantum computing provides the exponential speedup.*

Step 3

However, the evaluated results are stored as the basis vectors in a superposition wavefunction, and we *cannot* extract them directly. The next step is to perform QFT to extract the results we want. We apply a QFT operation defined in Eq. (25.13), which is repeated here for convenience but with the dummy variable, j changed to x and $N = 2^{2n}$ as this is a $2n$-qubit QFT.

$$\boldsymbol{U_{QFT}} |x\rangle = \frac{1}{\sqrt{2^{2n}}} \sum_{k=0}^{2^{2n}-1} \omega^{-kx} |k\rangle \tag{29.12}$$

The QFT is only applied to the first register. We also need to apply an n-qubit identity gate to the second register. Therefore,

$$\begin{aligned}
|\xi_3\rangle &= (\boldsymbol{U_{QFT}} \otimes \boldsymbol{I}^{\otimes n}) |\xi_2\rangle \\
&= (\boldsymbol{U_{QFT}} \otimes \boldsymbol{I}^{\otimes n})(\frac{1}{2^n} \sum_{x=0}^{2^{2n}-1} |x\rangle_{2n} |f(x)\rangle_n \\
&= \frac{1}{2^n} \sum_{x=0}^{2^{2n}-1} (\boldsymbol{U_{QFT}} |x\rangle_{2n})(\boldsymbol{I}^{\otimes n} |f(x)\rangle_n) \\
&= \frac{1}{2^n} \sum_{x=0}^{2^{2n}-1} (\boldsymbol{U_{QFT}} |x\rangle_{2n}) |f(x)\rangle_n \\
&= \frac{1}{2^n} \sum_{x=0}^{2^{2n}-1} (\frac{1}{\sqrt{2^{2n}}} \sum_{k=0}^{2^{2n}-1} \omega^{-kx} |k\rangle_{2n}) |f(x)\rangle_n \\
&= \frac{1}{2^{2n}} \sum_{x=0}^{2^{2n}-1} \sum_{k=0}^{2^{2n}-1} \omega^{-kx} |k\rangle_{2n} |f(x)\rangle_n
\end{aligned} \tag{29.13}$$

Here we again use parentheses to separate the operators and vectors in the first two lines. We then substitute Eq. (29.12) for $\boldsymbol{U_{QFT}}$ in the second last line followed by grouping terms. We note that $|k\rangle$ has $2n$ qubits and $\omega = e^{i2\pi/2^{2n}}$ (see definition of N-th root unity when $N = 2^{2n}$ in Chap. 25).

Step 4

We will now perform a measurement on the least significant n-qubit (the second register). The wavefunction will collapse to one of the $|f(x)\rangle_n$. Referring to Eq. (29.2), for example, if the function is $f_{11,21}(x)$, there are only 6 possible $|f(x)\rangle_n$, which are $|1\rangle$, $|11\rangle$, $|16\rangle$, $|8\rangle$, $|4\rangle$, and $|2\rangle$. However, there are many x that will give the same $f(x)$. For example, $f(x = 0) = f(x = 6) = f(x = 12) = \cdots = 1$ and $f(x = 2) = f(x = 8) = f(x = 14) = \cdots = 16$. And this is because of the periodicity of the function. Therefore, when the wavefunction collapses, for example becomes $|f(x_0)\rangle_n$, it actually collapses by keeping $|f(x_0)\rangle_n$, $|f(x_0 + r)\rangle_n$, $|f(x_0 + 2r)\rangle_n$, $|f(x_0 + 3r)\rangle_n$, etc., where r is the periodicity. That means when the wavefunction collapses to a certain state after measurement, it will keep all x separated by the period r. Therefore, if after the measurement, it collapses to $|f(x_0)\rangle_n$, the wavefunction becomes

$$
\begin{aligned}
|\xi_4\rangle &\approx \sum_{x=x_0+yr} \sum_{k=0}^{2^{2n}-1} \omega^{-kx} |k\rangle_{2n} |f(x)\rangle_n \\
&= \sum_{x=x_0+yr} \sum_{k=0}^{2^{2n}-1} \omega^{-kx} |k\rangle_{2n} |f(x_0)\rangle_n \\
&= \sum_{k=0}^{2^{2n}-1} \sum_{x=x_0+yr} \omega^{-kx} |k\rangle_{2n} |f(x_0)\rangle_n \\
&= \sum_{k=0}^{2^{2n}-1} \sum_{y} \omega^{-k(x_0+yr)} |k\rangle_{2n} |f(x_0)\rangle_n \\
&= \sum_{k=0}^{2^{2n}-1} \sum_{y} \omega^{-kx_0} \omega^{-kyr} |k\rangle_{2n} |f(x_0)\rangle_n
\end{aligned}
\tag{29.14}
$$

In the first line, I removed all x not separated by multiples of r from x_0. Therefore, the summation $\sum_{x=0}^{2^{2n}-1}$ becomes $\sum_{x=x_0+yr}$, which means we only have a periodic series of x that are x_0, $x_0 + r$, $x_0 + 2r$, etc. and y are integers. (If there were no periodicity, we even would not have this summation.) The maximum y should be such that $x_0 + yr < 2^{2n} - 1$. *Note that I also skipped the constants in front of the summations*. This is because we need to perform normalization after the measurement and the equation is too messy to put. In the second line, I just replace $|f(x)\rangle_n$ by $|f(x_0)\rangle_n$ because all x in $x = x_0 + yr$ give $|f(x_0)\rangle_n$. In the fourth line, we then convert the summation of x into a summation of y and rewrite $x = x_0 + yr$ in the exponents.

Step 5

We will now perform a measurement on the first register that has the most significant $2n$ qubits. Assume it will collapse to $|k_0\rangle$. Then

$$|\xi_5\rangle \approx \sum_y \omega^{-k_0 x_0} \omega^{-k_0 y r} |k_0\rangle_{2n} |f(x_0)\rangle_n$$
$$= \omega^{-k_0 x_0} (\sum_y \omega^{-k_0 y r}) |k_0\rangle_{2n} |f(x_0)\rangle_n \tag{29.15}$$

where again we do not perform normalization and skip the constant coefficients. We see that the coefficient of the measured state is proportional to $\sum_y \omega^{-k_0 y r}$, and I put in a pair of parentheses to emphasize this. Let us substitute $\omega = e^{i2\pi/2^{2n}}$.

$$\sum_y \omega^{-k_0 y r} = \sum_y e^{-i2\pi k_0 y r/2^{2n}} \tag{29.16}$$

We will not evaluate the sum but will try to understand it graphically. Figure 29.2 shows two types of possible distributions of $e^{-i2\pi k_0 y r/2^{2n}}$. Starting with $y = 0$, the terms go around the unit circle in the complex plane anti-clockwise. After going through many rounds, the numbers distribute fairly evenly on the unit circuit. Their summation is thus very small as they cancel the others (**destructive interference**). This happens for some k, and thus the probability of measuring these $|k\rangle$ is very small and near zero. On the other hand, certain values of k will give a large enough and just enough step to go around a full circle when y is incremented by 1. In this

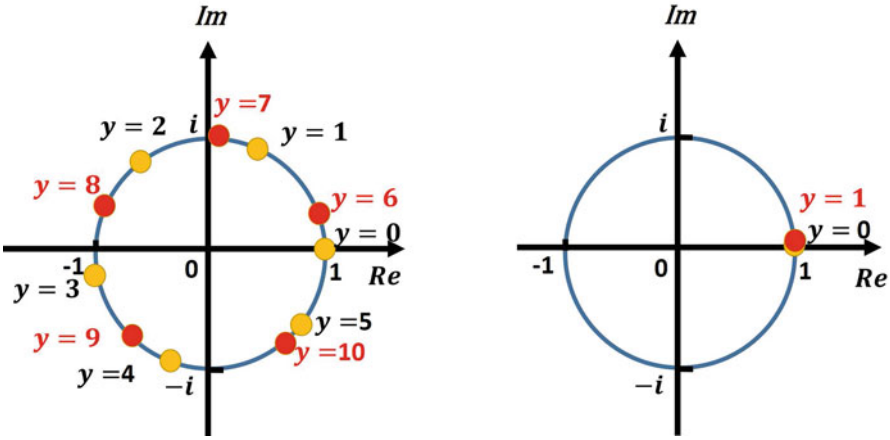


Fig. 29.2 Plotting of $e^{-i2\pi k_0 y r/2^{2n}}$ on the complex plane. Yellow dots refer to the first round, and red dots refer to the second round. Left: Distribution of the terms resulting in destructive interference. Right: Distribution of the terms resulting in constructive interference

case, all the terms will be at the same point or very close to the others. This means the phase of $e^{-i2\pi k_0 yr/2^{2n}}$ needs to increase by 2π or multiples of 2π when y is increased by 1 so to result in a **constructive interference**. That is

$$\begin{aligned} 2\pi k_0 r/2^{2n} &= 2\pi d \\ r &= 2^{2n} d/k_0 \end{aligned} \tag{29.17}$$

where d is an integer indicating how many circles it goes around when y increases by 1. Here, it means that if we measure $|k_0\rangle$, then the period, r, is $2^{2n}d/k_0$. Since d is an integer that we do not know, we still do not know the value of r. However, Shor's algorithm already helps to narrow down the possible values of r. After this, the classical **continued fraction expansion** will be used to further narrow down the possible value of r, which will not be discussed here. The result can be checked easily, and if it is wrong, we will redo Shor's algorithm until the correct answer is found.

29.4 Summary

In this chapter, we have presented Shor's algorithm. We understand that Shor's algorithm is a period finding algorithm. The period finding is important but a slow part of prime integer factorization. Shor's algorithm provides an exponential speedup because it computes a function for exponentially many arguments through quantum computing. The answer is obtained through constructive interference by QFT and measurements. Shor's algorithm only works when certain periodic settings are correct. The result given by Shor's algorithm is also not necessarily the correct answer. However, the answer can be verified easily using multiplication in classical computers. Due to statistics, we expect to find the correct answer after some trials and errors.

Problems

29.1 Modulo Operation
Find 177 mod 25.

29.2 Greatest Common Divisor
Find the gcd of 1242 and 242.

29.3 Prime Number Factorization
Repeat what was done in the text and perform the prime number factorization for 63. You need to pick an a for this problem. Try to do this in Google Colab and use the numpy.gcd function. Try different values of a.

29.4 Shor's Algorithm

Let $n = 1$, show how the 3-qubit vector evolves after each stage (partitioned by the red lines in Fig. 29.1), and stop before measurement. Keep $f(x)$. This is a problem to practice substitution. Is it possible to have an $n = 1$ Shor's algorithm circuit?

Chapter 30
The Last But Not the Least

30.1 Learning Outcomes

Understand how to practice quantum programming; understand what you do not know.

30.2 End of the Beginning

As the saying goes, *This is not the end. This is not even the beginning of the end. This is the end of the beginning*. I hope you have learned the basic and necessary linear algebra, quantum computing gates, and quantum algorithms. It does not matter if you can memorize them because you can search or google easily (maybe using Grover's algorithm someday). The most important is that you have understood the languages and concepts and you are able to follow the derivations and reasoning, so you know what to look for. If so, you are ready to move forward, and I can tell you that you have enough knowledge to take more advanced quantum computing classes and do research in quantum computing as long as you understand this is just the *beginning*. If you think this is the end for you as you do not plan to continue to pursue the technical aspects, it is also fine because you are able to talk to the engineers and physicists in quantum computing already. In this last chapter of the book, I want to talk about what can be the next steps for those who want to continue to learn quantum computing.

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0_30

30.3 Quantum Programming

I hope you are not disappointed that it seems that the so-called quantum programming in this book is just a graphical interface to build a circuit (like what an electrical engineer does instead of what a programmer does). There are a few reasons I have only shown you the graphical interface so far. Quantum programming has no difference from classical programming. The only difference is how you understand the meaning of the operations. Therefore, it is not necessary to discuss quantum programming separately. This will allow readers who are not familiar with a certain programming language or platform to learn the essence of quantum programming first. And please remember, *the horizontal axis of the quantum circuit signifies the flow of time instead of a physical space*. This is just like programming.

Of course, there is a limitation on using the graphical interface when the program becomes large. In this case, we want to use the programming language and the library (e.g. calling a 6-qubit QFT function instead of drawing yourself when you want to build a large QPE circuit). Let us look at an example of how to do this in IBM-Q. In Chap. 28, we studied QPE. In Fig. 28.3, we created a QPE with a 2-qubit c-register to estimate the phase of the eigenvalue of the eigenvector $|e_1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ of the Z-gate. The result is $|100\rangle$. We can also construct the circuit with a 3-qubit c-register to increase the accuracy of the phase estimation (although 2 qubits are enough for the case of Z-gate). We expect the result to be $|1000\rangle$ (Exercise 28.5). I can create a circuit using IBM-Q's "IBM Quantum Composer," which you have been using as shown in Fig. 30.1.

We can convert this into a program by clicking "View" and then "Code Editor." It shows the code in the "OpenQASM" language. The following is what it was converted into with my comments. It should be easy for you to understand also. Please read it carefully. *Please try to map to the circuit*.

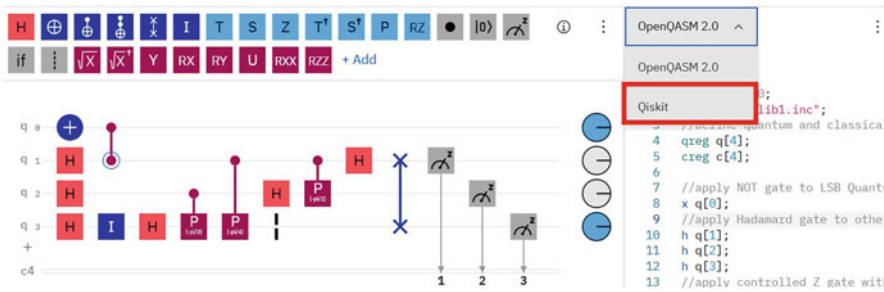


Fig. 30.1 QPE circuit to find the phase of the Z-gate's eigenvalue using a 3-qubit c-register

```
OPENQASM 2.0;
include "qelib1.inc";
//Define quantum and classical registers
qreg q[4];
creg c[4];

//apply NOT gate to LSB Quantum qubit
x q[0];
//apply Hadamard gate to other Quantum qubits
h q[1];
h q[2];
h q[3];
//apply controlled Z gate with q[1] as the control qubit
// and q[0] as the target qubit
cz q[1],q[0];
//apply identity gate to q[3]
id q[3];
//apply Hadamard gate to q[3]
h q[3];
//apply controlled Phase Shift gate (phase = -pi/2) with
// q[2] as the control qubit and q[3] as the target qubit
cp(-pi/2) q[2],q[3];
//apply controlled Phase Shift gate (phase = -pi/4) with
//q[1] as the control qubit and q[3] as the target qubit
cp(-pi/4) q[1],q[3];
//apply Hadamard gate to q[2]
h q[2];
//add barrier for alignment (just an example)
barrier q[3];
//apply controlled Phase Shift gate (phase = -pi/2) with
//q[1] as the control qubit and q[2] as the target qubit
cp(-pi/2) q[1],q[2];
//apply Hadamard gate to q[1]
h q[1];
//apply SWAP gate to swap q[1] and q[2]
swap q[1],q[3];
//measure q[1] and store in the classical bit c[1]
measure q[1] -> c[1];
//measure q[2] and store in the classical bit c[2]
measure q[2] -> c[2];
//measure q[3] and store in the classical bit c[3]
measure q[3] -> c[3];
```

However, there are many different languages that can achieve the same goal, and they are different in different computing platforms. Fortunately, most of them are based on Python. One of them is the Qiskit framework that is compatible with IBM-Q. In IBM-Q, it will convert the circuit into Qiskit automatically (See Fig. 30.1 top right.). It will open the code in "IBM Quantum Lab" that is in the form of Jupyter Lab. The following is the converted circuit code in Qiskit. However, it is not ready to run. I added the commands to run the simulation and print the circuit with comments. They are easier to read than the QASM language. Figure 30.2 shows the outputs.

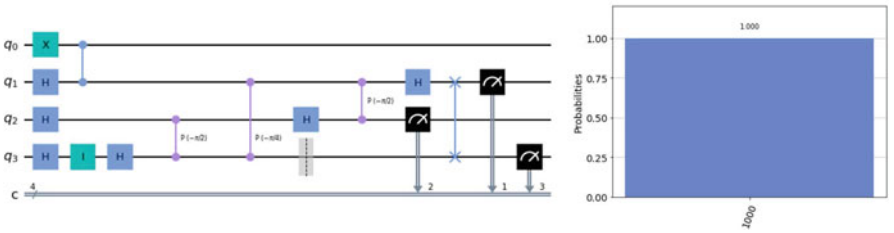


Fig. 30.2 Output from Qiskit for the QPE circuit to find the phase of the Z-gate's eigenvalue using a 3-qubits c-register

```
from ibm_quantum_widgets import CircuitComposer
# import BasicAer to access python simulator through
# BasicAer provider
# import execute to execute a list of circuits on the
#backend
from qiskit import QuantumRegister, ClassicalRegister
from qiskit import QuantumCircuit, BasicAer, execute
from numpy import pi
from qiskit.visualization import plot_histogram

qreg_q = QuantumRegister(4, 'q')
creg_c = ClassicalRegister(4, 'c')
circuit = QuantumCircuit(qreg_q, creg_c)

circuit.x(qreg_q[0])
circuit.h(qreg_q[1])
circuit.h(qreg_q[2])
circuit.h(qreg_q[3])
circuit.cz(qreg_q[1], qreg_q[0])
circuit.id(qreg_q[3])
circuit.h(qreg_q[3])
circuit.cp(-pi/2, qreg_q[2], qreg_q[3])
circuit.cp(-pi/4, qreg_q[1], qreg_q[3])
circuit.h(qreg_q[2])
circuit.barrier(qreg_q[3])
circuit.cp(-pi/2, qreg_q[1], qreg_q[2])
circuit.h(qreg_q[1])
circuit.swap(qreg_q[1], qreg_q[3])
circuit.measure(qreg_q[1], creg_c[1])
circuit.measure(qreg_q[2], creg_c[2])
circuit.measure(qreg_q[3], creg_c[3])

# I will remove this as this is to embed the circuit editor
# in the notebook.
# editor = CircuitComposer(circuit=circuit)
# editor

# Execute the circuit using the simulator. Run for 65536 times
simulator = BasicAer.get_backend('qasm_simulator')
```

```
job = execute(circuit, backend=simulator, shots=65536)

#Get the result of the execution
result = job.result()

# Get the counts, the frequency of each answer
counts = result.get_counts(circuit)

# Display the results
plot_histogram(counts)

# Print the circuit. Put in a new cell
circuit.draw('mpl',scale=1)
```

If you are already a skillful programmer, you can use Jupyter Lab (or start with "IBM Quantum Lab") directly to create more sophisticated quantum circuits. If not, you just need to learn Python as a classical programmer and get used to this environment.

30.4 Next Steps

What we have learned is the so-called *gate models*. We use the quantum gates to perform computations. This has the potential to solve universal problems. However, as you have been seeing, the error in the quantum gate accumulates, and even our simple quantum circuits give wrong results very often. Therefore, error correction is needed. But we cannot use the classical error correction schemes. In classical error corrections, we rely on replication and measurement (to check error). But quantum computing does not allow us to clone (no-cloning theorem) and measure to check error (states will collapse and lose their information). We need special schemes, and, as a result, a logical qubit needs to be implemented by tens to hundreds of physical qubits in order to achieve a low enough error rate for large-scale computations. I urge you to study error correction as the next step.

At the moment that we still do not have robust qubits (as of 2022), people propose to use only the noisy intermediate-scale quantum (NISQ) algorithms in quantum computing.

Besides *gate models*, *quantum annealing*, which is not covered in this book, is also widely used as a means for quantum computing. It uses quantum computers to find the optimized solution to a problem by minimizing the energy of a system. Not all problems can be mapped to an optimization problem. But many computationally difficult problems find it useful. This type of problem is also less sensitive to noise than the gate models.

As the next step, I challenge you to learn the HHL algorithm that is a famous quantum computing algorithm to solve systems of linear equations. You can find step-by-step guidance in https://arxiv.org/abs/2108.09004 (Hector Morrell

and Hiu Yung Wong, “Step-by-Step HHL Algorithm Walkthrough to Enhance the Understanding of Critical Quantum Computing Concepts”, arXiv preprint arXiv:2108.09004). In this document, you will also find the Matlab code and Qiskit code to run the example HHL algorithm.

Index

H. Y. Wong, *Introduction to Quantum Computing*,
https://doi.org/10.1007/978-3-030-98339-0