

## II Quantum computation

### 4 Quantum circuits

*The theory of computation has traditionally been studied almost entirely in the abstract, as a topic in pure mathematics. This is to miss the point of it. Computers are physical objects, and computations are physical processes. What computers can or cannot compute is determined by the laws of physics alone, and not by pure mathematics.*

– David Deutsch

*Like mathematics, computer science will be somewhat different from the other sciences, in that it deals with artificial laws that can be proved, instead of natural laws that are never known with certainty.*

– Donald Knuth

*The opposite of a profound truth may well be another profound truth.*

– Niels Bohr

This chapter begins Part II of the book, in which we explore quantum computation in detail. The chapter develops the fundamental principles of quantum computation, and establishes the basic building blocks for quantum circuits, a universal language for describing sophisticated quantum computations. The two fundamental quantum algorithms known to date are constructed from these circuits in the following two chapters. Chapter 5 presents the quantum Fourier transform and its applications to phase estimation, order-finding and factoring. Chapter 6 describes the quantum search algorithm, and its applications to database search, counting and speedup of solutions to NP-complete problems. Chapter 7 concludes Part II with a discussion of how quantum computation may one day be experimentally realized. Two other topics of great interest for quantum computation, quantum noise and quantum error-correction, are deferred until Part III of the book, in view of their wide interest also *outside* quantum computation.

There are two main ideas introduced in this chapter. First, we explain in detail the fundamental model of quantum computation, the quantum circuit model. Second, we demonstrate that there exists a small set of gates which are *universal*, that is, any quantum computation whatsoever can be expressed in terms of those gates. Along the way we also have occasion to describe many other basic results of quantum computation. Section 4.1 begins the chapter with an overview of quantum algorithms, focusing on what algorithms are known, and the unifying techniques underlying their construction. Section 4.2 is a detailed study of single qubit operations. Despite their simplicity, single qubit operations offer a rich playground for the construction of examples and techniques, and it is essential to understand them in detail. Section 4.3 shows how to perform multi-qubit *controlled unitary* operations, and Section 4.4 discusses the description of measurement in the quantum circuits model. These elements are then brought together in Section 4.5 for the statement and proof of the universality theorem. We summarize all the basic elements

of quantum computation in Section 4.6, and discuss possible variants of the model, and the important question of the relationship in computational power between classical and quantum computers. Section 4.7 concludes the chapter with an important and instructive application of quantum computation to the *simulation* of real quantum systems.

This chapter is perhaps the most reader-intensive of all the chapters in the book, with a high density of exercises for you to complete, and it is worth explaining the reason for this intensity. Obtaining facility with the basic elements of the quantum circuit model of computation is quite easy, but requires assimilating a large number of simple results and techniques that must become second nature if one is to progress to the more difficult problem of designing quantum algorithms. For this reason we take an example-oriented approach in this chapter, and ask you to fill in many of the details, in order to acquire such a facility. A less intensive, but somewhat superficial overview of the basic elements of quantum computation may be obtained by skipping to Section 4.6.

## 4.1 Quantum algorithms

What is a quantum computer good for? We're all familiar with the frustration of needing more computer resources to solve a computational problem. Practically speaking, many interesting problems are impossible to solve on a classical computer, not because they are in principle insoluble, but because of the astronomical resources required to solve realistic cases of the problem.

The spectacular promise of quantum computers is to enable new algorithms which render feasible problems requiring exorbitant resources for their solution on a classical computer. At the time of writing, two broad classes of quantum algorithms are known which fulfill this promise. The first class of algorithms is based upon Shor's *quantum Fourier transform*, and includes remarkable algorithms for solving the factoring and discrete logarithm problems, providing a striking *exponential* speedup over the best known classical algorithms. The second class of algorithms is based upon Grover's algorithm for performing *quantum searching*. These provide a less striking but still remarkable *quadratic* speedup over the best possible classical algorithms. The quantum searching algorithm derives its importance from the widespread use of search-based techniques in classical algorithms, which in many instances allows a straightforward adaptation of the classical algorithm to give a faster quantum algorithm.

Figure 4.1 sketches the state of knowledge about quantum algorithms at the time of writing, including some sample applications of those algorithms. Naturally, at the core of the diagram are the quantum Fourier transform and the quantum searching algorithm. Of particular interest in the figure is the quantum counting algorithm. This algorithm is a clever combination of the quantum searching and Fourier transform algorithms, which can be used to estimate the number of solutions to a search problem more quickly than is possible on a classical computer.

The quantum searching algorithm has many potential applications, of which but a few are illustrated. It can be used to extract statistics, such as the minimal element, from an unordered data set, more quickly than is possible on a classical computer. It can be *used to speed up algorithms for some problems in NP* – specifically, those problems for which a straightforward search for a solution is the best algorithm known. Finally, it can be used to speed up the search for keys to cryptosystems such as the widely used Data Encryption Standard (DES). These and other applications are explained in Chapter 6.

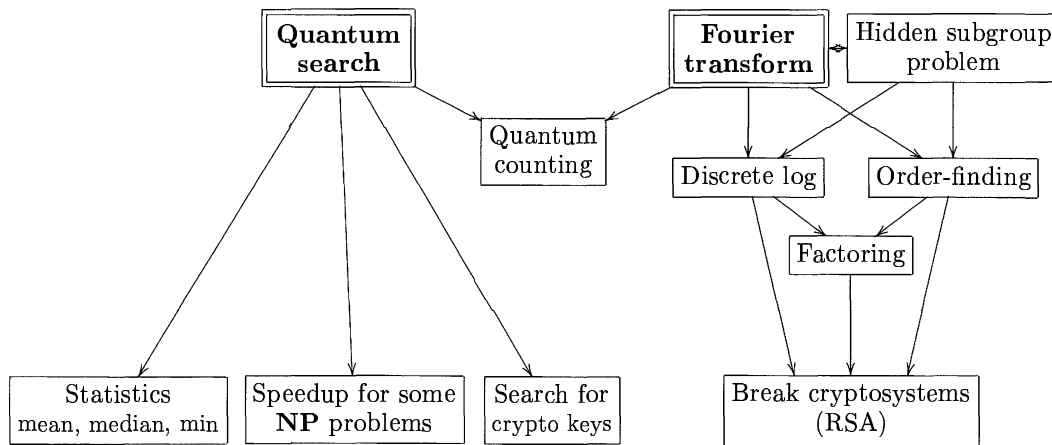


Figure 4.1. The main quantum algorithms and their relationships, including some notable applications.

The quantum Fourier transform also has many interesting applications. It can be used to solve the discrete logarithm and factoring problems. These results, in turn, enable a quantum computer to break many of the most popular cryptosystems now in use, including the RSA cryptosystem. The Fourier transform also turns out to be closely related to an important problem in mathematics, finding a hidden subgroup (a generalization of finding the period of a periodic function). The quantum Fourier transform and several of its applications, including fast quantum algorithms for factoring and discrete logarithm, are explained in Chapter 5.

Why are there so few quantum algorithms known which are better than their classical counterparts? The answer is that coming up with good quantum algorithms seems to be a difficult problem. There are at least two reasons for this. First, algorithm design, be it classical or quantum, is not an easy business! The history of algorithms shows us that considerable ingenuity is often required to come up with near optimal algorithms, even for apparently very simple problems, like the multiplication of two numbers. Finding good quantum algorithms is made doubly difficult because of the additional constraint that we want our quantum algorithms to be *better* than the best known classical algorithms. A second reason for the difficulty of finding good quantum algorithms is that our intuitions are much better adapted to the classical world than they are to the quantum world. If we think about problems using our native intuition, then the algorithms which we come up with are going to be classical algorithms. It takes special insights and special tricks to come up with good quantum algorithms.

Further study of quantum algorithms will be postponed until the next chapter. In this chapter we provide an efficient and powerful language for describing quantum algorithms, the language of quantum circuits – assemblies of discrete sets of components which describe computational procedures. This construction will enable us to quantify the cost of an algorithm in terms of things like the total number of gates required, or the circuit depth. The circuit language also comes with a toolbox of tricks that simplifies algorithm design and provides ready conceptual understanding.

## 4.2 Single qubit operations

The development of our quantum computational toolkit begins with operations on the simplest quantum system of all – a single qubit. Single qubit gates were introduced in Section 1.3.1. Let us quickly summarize what we learned there; you may find it useful to refer to the notes on notation on page xxiii as we go along.

A single qubit is a vector  $|\psi\rangle = a|0\rangle + b|1\rangle$  parameterized by two complex numbers satisfying  $|a|^2 + |b|^2 = 1$ . Operations on a qubit must preserve this norm, and thus are described by  $2 \times 2$  unitary matrices. Of these, some of the most important are the Pauli matrices; it is useful to list them again here:

$$X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}; \quad Y \equiv \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}; \quad Z \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \quad (4.1)$$

Three other quantum gates will play a large part in what follows, the Hadamard gate (denoted  $H$ ), phase gate (denoted  $S$ ), and  $\pi/8$  gate (denoted  $T$ ):

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}; \quad S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}; \quad T = \begin{bmatrix} 1 & 0 \\ 0 & \exp(i\pi/4) \end{bmatrix}. \quad (4.2)$$

A couple of useful algebraic facts to keep in mind are that  $H = (X + Z)/\sqrt{2}$  and  $S = T^2$ . You might wonder why the  $T$  gate is called the  $\pi/8$  gate when it is  $\pi/4$  that appears in the definition. The reason is that the gate has historically often been referred to as the  $\pi/8$  gate, simply because up to an unimportant global phase  $T$  is equal to a gate which has  $\exp(\pm i\pi/8)$  appearing on its diagonals.

$$T = \exp(i\pi/8) \begin{bmatrix} \exp(-i\pi/8) & 0 \\ 0 & \exp(i\pi/8) \end{bmatrix}. \quad (4.3)$$

Nevertheless, the nomenclature is in some respects rather unfortunate, and we often refer to this gate as the  $T$  gate.

Recall also that a single qubit in the state  $a|0\rangle + b|1\rangle$  can be visualized as a point  $(\theta, \varphi)$  on the unit sphere, where  $a = \cos(\theta/2)$ ,  $b = e^{i\varphi} \sin(\theta/2)$ , and  $a$  can be taken to be real because the overall phase of the state is unobservable. This is called the Bloch sphere representation, and the vector  $(\cos \varphi \sin \theta, \sin \varphi \sin \theta, \cos \theta)$  is called the Bloch vector. We shall return to this picture often as an aid to intuition.

**Exercise 4.1:** In Exercise 2.11, which you should do now if you haven't already done it, you computed the eigenvectors of the Pauli matrices. Find the points on the Bloch sphere which correspond to the normalized eigenvectors of the different Pauli matrices.

The Pauli matrices give rise to three useful classes of unitary matrices when they are exponentiated, the *rotation operators* about the  $\hat{x}$ ,  $\hat{y}$ , and  $\hat{z}$  axes, defined by the equations:

$$R_x(\theta) \equiv e^{-i\theta X/2} = \cos \frac{\theta}{2} I - i \sin \frac{\theta}{2} X = \begin{bmatrix} \cos \frac{\theta}{2} & -i \sin \frac{\theta}{2} \\ -i \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{bmatrix} \quad (4.4)$$

$$R_y(\theta) \equiv e^{-i\theta Y/2} = \cos \frac{\theta}{2} I - i \sin \frac{\theta}{2} Y = \begin{bmatrix} \cos \frac{\theta}{2} & -\sin \frac{\theta}{2} \\ \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{bmatrix} \quad (4.5)$$

$$R_z(\theta) \equiv e^{-i\theta Z/2} = \cos \frac{\theta}{2} I - i \sin \frac{\theta}{2} Z = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}. \quad (4.6)$$

**Exercise 4.2:** Let  $x$  be a real number and  $A$  a matrix such that  $A^2 = I$ . Show that

$$\exp(iAx) = \cos(x)I + i \sin(x)A. \quad (4.7)$$

Use this result to verify Equations (4.4) through (4.6).

**Exercise 4.3:** Show that, up to a global phase, the  $\pi/8$  gate satisfies  $T = R_z(\pi/4)$ .

**Exercise 4.4:** Express the Hadamard gate  $H$  as a product of  $R_x$  and  $R_z$  rotations and  $e^{i\varphi}$  for some  $\varphi$ .

If  $\hat{n} = (n_x, n_y, n_z)$  is a real unit vector in three dimensions then we generalize the previous definitions by defining a rotation by  $\theta$  about the  $\hat{n}$  axis by the equation

$$R_{\hat{n}}(\theta) \equiv \exp(-i\theta \hat{n} \cdot \vec{\sigma}/2) = \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) (n_x X + n_y Y + n_z Z), \quad (4.8)$$

where  $\vec{\sigma}$  denotes the three component vector  $(X, Y, Z)$  of Pauli matrices.

**Exercise 4.5:** Prove that  $(\hat{n} \cdot \vec{\sigma})^2 = I$ , and use this to verify Equation (4.8).

**Exercise 4.6: (Bloch sphere interpretation of rotations)** One reason why the  $R_{\hat{n}}(\theta)$  operators are referred to as rotation operators is the following fact, which you are to prove. Suppose a single qubit has a state represented by the Bloch vector  $\vec{\lambda}$ . Then the effect of the rotation  $R_{\hat{n}}(\theta)$  on the state is to rotate it by an angle  $\theta$  about the  $\hat{n}$  axis of the Bloch sphere. This fact explains the rather mysterious looking factor of two in the definition of the rotation matrices.

**Exercise 4.7:** Show that  $XYX = -Y$  and use this to prove that  $XR_y(\theta)X = R_y(-\theta)$ .

**Exercise 4.8:** An arbitrary single qubit unitary operator can be written in the form

$$U = \exp(i\alpha)R_{\hat{n}}(\theta) \quad (4.9)$$

for some real numbers  $\alpha$  and  $\theta$ , and a real three-dimensional unit vector  $\hat{n}$ .

1. Prove this fact.
2. Find values for  $\alpha$ ,  $\theta$ , and  $\hat{n}$  giving the Hadamard gate  $H$ .
3. Find values for  $\alpha$ ,  $\theta$ , and  $\hat{n}$  giving the phase gate

$$S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}. \quad (4.10)$$

An arbitrary unitary operator on a single qubit can be written in many ways as a combination of rotations, together with global phase shifts on the qubit. The following theorem provides a means of expressing an arbitrary single qubit rotation that will be particularly useful in later applications to controlled operations.

**Theorem 4.1: (Z-Y decomposition for a single qubit)** Suppose  $U$  is a unitary operation on a single qubit. Then there exist real numbers  $\alpha, \beta, \gamma$  and  $\delta$  such that

$$U = e^{i\alpha} R_z(\beta) R_y(\gamma) R_z(\delta). \quad (4.11)$$

*Proof*

Since  $U$  is unitary, the rows and columns of  $U$  are orthonormal, from which it follows that there exist real numbers  $\alpha, \beta, \gamma$ , and  $\delta$  such that

$$U = \begin{bmatrix} e^{i(\alpha-\beta/2-\delta/2)} \cos \frac{\gamma}{2} & -e^{i(\alpha-\beta/2+\delta/2)} \sin \frac{\gamma}{2} \\ e^{i(\alpha+\beta/2-\delta/2)} \sin \frac{\gamma}{2} & e^{i(\alpha+\beta/2+\delta/2)} \cos \frac{\gamma}{2} \end{bmatrix}. \quad (4.12)$$

Equation (4.11) now follows immediately from the definition of the rotation matrices and matrix multiplication.  $\square$

**Exercise 4.9:** Explain why any single qubit unitary operator may be written in the form (4.12).

**Exercise 4.10: ( $X$ - $Y$  decomposition of rotations)** Give a decomposition analogous to Theorem 4.1 but using  $R_x$  instead of  $R_z$ .

**Exercise 4.11:** Suppose  $\hat{m}$  and  $\hat{n}$  are non-parallel real unit vectors in three dimensions. Show that an arbitrary single qubit unitary  $U$  may be written as

$$U = e^{i\alpha} R_{\hat{n}}(\beta_1) R_{\hat{m}}(\gamma_1) R_{\hat{n}}(\beta_2) R_{\hat{m}}(\gamma_2) \dots, \quad (4.13)$$

for appropriate choices of  $\alpha$  and  $\beta_k, \gamma_k$ .

The utility of Theorem 4.1 lies in the following mysterious looking corollary, which is the key to the construction of controlled multi-qubit unitary operations, as explained in the next section.

*Corollary 4.2:* Suppose  $U$  is a unitary gate on a single qubit. Then there exist unitary operators  $A, B, C$  on a single qubit such that  $ABC = I$  and  $U = e^{i\alpha} AXBXC$ , where  $\alpha$  is some overall phase factor.

*Proof*

In the notation of Theorem 4.1, set  $A \equiv R_z(\beta)R_y(\gamma/2)$ ,  $B \equiv R_y(-\gamma/2)R_z(-(\delta+\beta)/2)$  and  $C \equiv R_z((\delta-\beta)/2)$ . Note that

$$ABC = R_z(\beta)R_y\left(\frac{\gamma}{2}\right)R_y\left(-\frac{\gamma}{2}\right)R_z\left(-\frac{\delta+\beta}{2}\right)R_z\left(\frac{\delta-\beta}{2}\right) = I. \quad (4.14)$$

Since  $X^2 = I$ , and using Exercise 4.7, we see that

$$XBX = XR_y\left(-\frac{\gamma}{2}\right)XXR_z\left(-\frac{\delta+\beta}{2}\right)X = R_y\left(\frac{\gamma}{2}\right)R_z\left(\frac{\delta+\beta}{2}\right). \quad (4.15)$$

Thus

$$AXBXC = R_z(\beta)R_y\left(\frac{\gamma}{2}\right)R_y\left(\frac{\gamma}{2}\right)R_z\left(\frac{\delta+\beta}{2}\right)R_z\left(\frac{\delta-\beta}{2}\right) \quad (4.16)$$

$$= R_z(\beta)R_y(\gamma)R_z(\delta). \quad (4.17)$$

Thus  $U = e^{i\alpha} AXBXC$  and  $ABC = I$ , as required.  $\square$

**Exercise 4.12:** Give  $A, B, C$ , and  $\alpha$  for the Hadamard gate.

**Exercise 4.13: (Circuit identities)** It is useful to be able to simplify circuits by inspection, using well-known identities. Prove the following three identities:

$$H X H = Z; \quad H Y H = -Y; \quad H Z H = X. \quad (4.18)$$

**Exercise 4.14:** Use the previous exercise to show that  $HTH = R_x(\pi/4)$ , up to a global phase.

**Exercise 4.15: (Composition of single qubit operations)** The Bloch representation gives a nice way to visualize the effect of composing two rotations.

- (1) Prove that if a rotation through an angle  $\beta_1$  about the axis  $\hat{n}_1$  is followed by a rotation through an angle  $\beta_2$  about an axis  $\hat{n}_2$ , then the overall rotation is through an angle  $\beta_{12}$  about an axis  $\hat{n}_{12}$  given by

$$c_{12} = c_1 c_2 - s_1 s_2 \hat{n}_1 \cdot \hat{n}_2 \quad (4.19)$$

$$s_{12} \hat{n}_{12} = s_1 c_2 \hat{n}_1 + c_1 s_2 \hat{n}_2 - s_1 s_2 \hat{n}_2 \times \hat{n}_1, \quad (4.20)$$

where  $c_i = \cos(\beta_i/2)$ ,  $s_i = \sin(\beta_i/2)$ ,  $c_{12} = \cos(\beta_{12}/2)$ , and  $s_{12} = \sin(\beta_{12}/2)$ .

- (2) Show that if  $\beta_1 = \beta_2$  and  $\hat{n}_1 = \hat{z}$  these equations simplify to

$$c_{12} = c^2 - s^2 \hat{z} \cdot \hat{n}_2 \quad (4.21)$$

$$s_{12} \hat{n}_{12} = s c (\hat{z} + \hat{n}_2) - s^2 \hat{n}_2 \times \hat{z}, \quad (4.22)$$

where  $c = c_1$  and  $s = s_1$ .

Symbols for the common single qubit gates are shown in Figure 4.2. Recall the basic properties of quantum circuits: time proceeds from left to right; wires represent qubits, and a ‘/’ may be used to indicate a bundle of qubits.






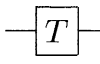
Hadamard		$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Pauli-X		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Phase		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$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

Figure 4.2. Names, symbols, and unitary matrices for the common single qubit gates.

### 4.3 Controlled operations

‘If  $A$  is true, then do  $B$ ’. This type of *controlled operation* is one of the most useful in computing, both classical and quantum. In this section we explain how complex controlled operations may be implemented using quantum circuits built from elementary operations.

The prototypical controlled operation is the controlled-NOT, which we met in Section 1.2.1. Recall that this gate, which we'll often refer to as CNOT, is a quantum gate with two input qubits, known as the *control qubit* and *target qubit*, respectively. It is drawn as shown in Figure 4.3. In terms of the computational basis, the action of the CNOT is given by  $|c\rangle|t\rangle \rightarrow |c\rangle|t \oplus c\rangle$ ; that is, if the control qubit is set to  $|1\rangle$  then the target qubit is flipped, otherwise the target qubit is left alone. Thus, in the computational basis  $|control, target\rangle$  the matrix representation of CNOT is

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \quad (4.23)$$

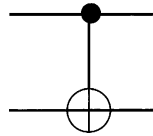


Figure 4.3. Circuit representation for the controlled-NOT gate. The top line represents the control qubit, the bottom line the target qubit.

More generally, suppose  $U$  is an arbitrary single qubit unitary operation. A *controlled- $U$*  operation is a two qubit operation, again with a control and a target qubit. If the control qubit is set then  $U$  is applied to the target qubit, otherwise the target qubit is left alone; that is,  $|c\rangle|t\rangle \rightarrow |c\rangle U^c|t\rangle$ . The controlled- $U$  operation is represented by the circuit shown in Figure 4.4.

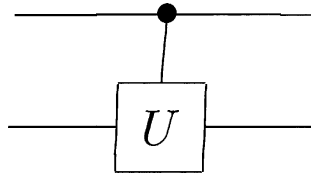


Figure 4.4. Controlled- $U$  operation. The top line is the control qubit, and the bottom line is the target qubit. If the control qubit is set then  $U$  is applied to the target, otherwise it is left alone.

**Exercise 4.16: (Matrix representation of multi-qubit gates)** What is the  $4 \times 4$  unitary matrix for the circuit

$$x_2 - \boxed{H} -$$

$$x_1 \text{ —————}$$

in the computational basis? What is the unitary matrix for the circuit

$$x_2 \text{ —————}$$

$$x_1 - \boxed{H} -$$



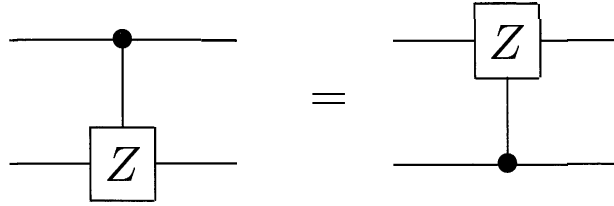
in the computational basis?

**Exercise 4.17: (Building CNOT from controlled- $Z$  gates)** Construct a CNOT gate from one controlled- $Z$  gate, that is, the gate whose action in the computational basis is specified by the unitary matrix

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

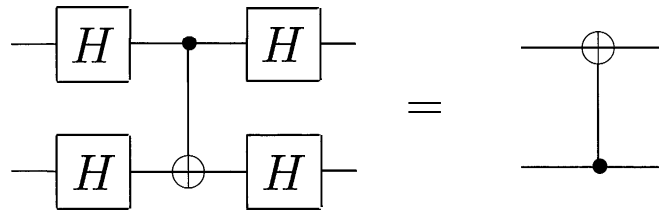
and two Hadamard gates, specifying the control and target qubits.

**Exercise 4.18:** Show that



**Exercise 4.19: (CNOT action on density matrices)** The CNOT gate is a simple permutation whose action on a density matrix  $\rho$  is to rearrange the elements in the matrix. Write out this action explicitly in the computational basis.

**Exercise 4.20: (CNOT basis transformations)** Unlike ideal classical gates, ideal quantum gates do not have (as electrical engineers say) ‘high-impedance’ inputs. In fact, the role of ‘control’ and ‘target’ are arbitrary – they depend on what basis you think of a device as operating in. We have described how the CNOT behaves with respect to the computational basis, and in this description the state of the control qubit is not changed. However, if we work in a different basis then the control qubit *does* change: we will show that its phase is flipped depending on the state of the ‘target’ qubit! Show that



Introducing basis states  $|\pm\rangle \equiv (|0\rangle \pm |1\rangle)/\sqrt{2}$ , use this circuit identity to show that the effect of a CNOT with the first qubit as control and the second qubit as target is as follows:

$$|+\rangle|+\rangle \rightarrow |+\rangle|+\rangle \quad (4.24)$$

$$|-\rangle|+\rangle \rightarrow |-\rangle|+\rangle \quad (4.25)$$

$$|+\rangle|-\rangle \rightarrow |-\rangle|-\rangle \quad (4.26)$$

$$|-\rangle|-\rangle \rightarrow |+\rangle|-\rangle. \quad (4.27)$$

Thus, with respect to this new basis, the state of the target qubit is not changed, while the state of the control qubit is flipped if the target starts as  $|-\rangle$ , otherwise

it is left alone. That is, in this basis, the target and control have essentially interchanged roles!

Our immediate goal is to understand how to implement the controlled- $U$  operation for arbitrary single qubit  $U$ , using only single qubit operations and the CNOT gate. Our strategy is a two-part procedure based upon the decomposition  $U = e^{i\alpha}AXBXC$  given in Corollary 4.2 on page 176.

Our first step will be to apply the phase shift  $\exp(i\alpha)$  on the target qubit, controlled by the control qubit. That is, if the control qubit is  $|0\rangle$ , then the target qubit is left alone, while if the control qubit is  $|1\rangle$ , a phase shift  $\exp(i\alpha)$  is applied to the target. A circuit implementing this operation using just a single qubit unitary gate is depicted on the right hand side of Figure 4.5. To verify that this circuit works correctly, note that the effect of the circuit on the right hand side is

$$|00\rangle \rightarrow |00\rangle, \quad |01\rangle \rightarrow |01\rangle, \quad |10\rangle \rightarrow e^{i\alpha}|10\rangle, \quad |11\rangle \rightarrow e^{i\alpha}|11\rangle, \quad (4.28)$$

which is exactly what is required for the controlled operation on the left hand side.

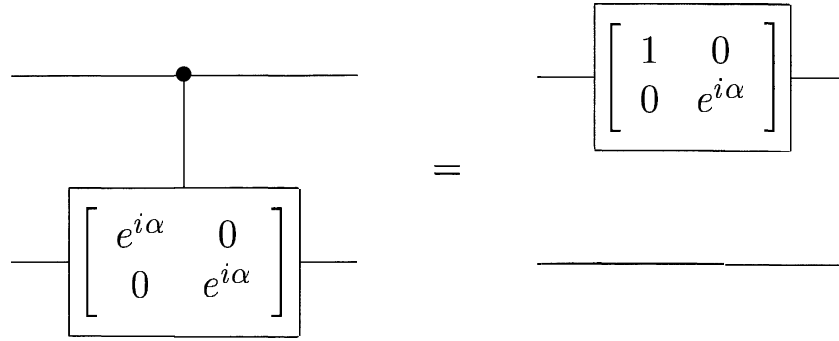


Figure 4.5. Controlled phase shift gate and an equivalent circuit for two qubits.

We may now complete the construction of the controlled- $U$  operation, as shown in Figure 4.6. To understand why this circuit works, recall from Corollary 4.2 that  $U$  may be written in the form  $U = e^{i\alpha}AXBXC$ , where  $A$ ,  $B$  and  $C$  are single qubit operations such that  $ABC = I$ . Suppose that the control qubit is set. Then the operation  $e^{i\alpha}AXBXC = U$  is applied to the second qubit. If, on the other hand, the control qubit is not set, then the operation  $ABC = I$  is applied to the second qubit; that is, no change is made. That is, this circuit implements the controlled- $U$  operation.

Now that we know how to condition on a single qubit being set, what about conditioning on multiple qubits? We've already met one example of multiple qubit conditioning, the Toffoli gate, which flips the third qubit, the target qubit, conditioned on the first two qubits, the control qubits, being set to one. More generally, suppose we have  $n + k$  qubits, and  $U$  is a  $k$  qubit unitary operator. Then we define the controlled operation  $C^n(U)$  by the equation

$$C^n(U)|x_1x_2 \dots x_n\rangle|\psi\rangle = |x_1x_2 \dots x_n\rangle U^{x_1x_2 \dots x_n}|\psi\rangle, \quad (4.29)$$

where  $x_1x_2 \dots x_n$  in the exponent of  $U$  means the *product* of the bits  $x_1, x_2, \dots, x_n$ . That is, the operator  $U$  is applied to the last  $k$  qubits if the first  $n$  qubits are all equal to one, and otherwise, nothing is done. Such conditional operations are so useful that we

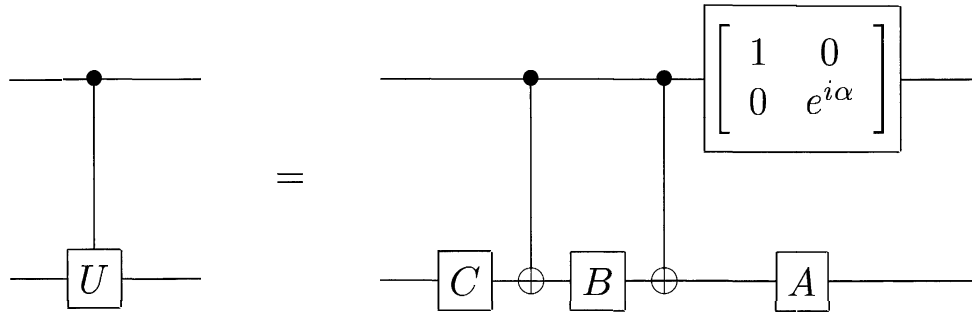


Figure 4.6. Circuit implementing the controlled- $U$  operation for single qubit  $U$ .  $\alpha$ ,  $A$ ,  $B$  and  $C$  satisfy  $U = \exp(i\alpha)AXBXC$ ,  $ABC = I$ .

introduce a special circuit notation for them, illustrated in Figure 4.7. For the following we assume that  $k = 1$ , for simplicity. Larger  $k$  can be dealt with using essentially the same methods, however for  $k \geq 2$  there is the added complication that we don't (yet) know how to perform arbitrary operations on  $k$  qubits.

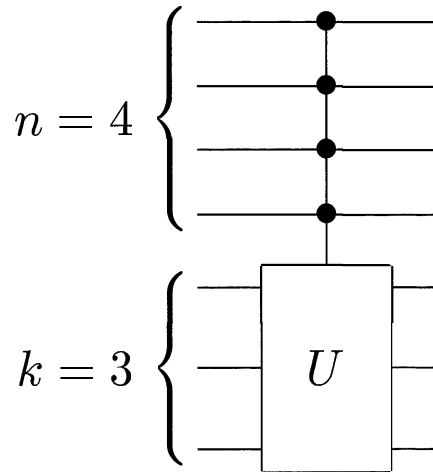


Figure 4.7. Sample circuit representation for the  $C^n(U)$  operation, where  $U$  is a unitary operator on  $k$  qubits, for  $n = 4$  and  $k = 3$ .

Suppose  $U$  is a single qubit unitary operator, and  $V$  is a unitary operator chosen so that  $V^2 = U$ . Then the operation  $C^2(U)$  may be implemented using the circuit shown in Figure 4.8.

**Exercise 4.21:** Verify that Figure 4.8 implements the  $C^2(U)$  operation.

**Exercise 4.22:** Prove that a  $C^2(U)$  gate (for any single qubit unitary  $U$ ) can be constructed using at most eight one-qubit gates, and six controlled-NOTs.

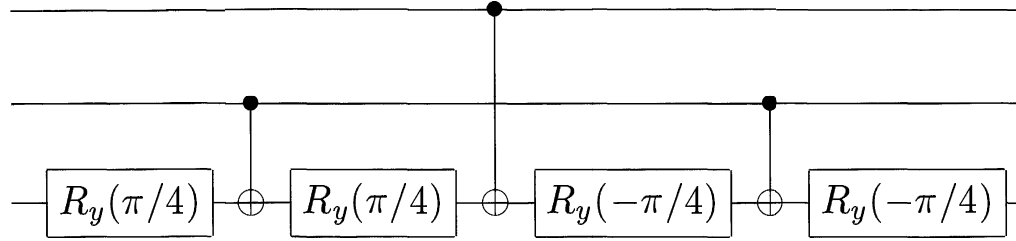
**Exercise 4.23:** Construct a  $C^1(U)$  gate for  $U = R_x(\theta)$  and  $U = R_y(\theta)$ , using only CNOT and single qubit gates. Can you reduce the number of single qubit gates needed in the construction from three to two?

The familiar Toffoli gate is an especially important special case of the  $C^2(U)$  operation,



- (1) Give a quantum circuit which uses three Toffoli gates to construct the Fredkin gate (*Hint*: think of the swap gate construction – you can control each gate, one at a time).
- (2) Show that the first and last Toffoli gates can be replaced by CNOT gates.
- (3) Now replace the middle Toffoli gate with the circuit in Figure 4.8 to obtain a Fredkin gate construction using only six two-qubit gates.
- (4) Can you come up with an even simpler construction, with only five two-qubit gates?

**Exercise 4.26:** Show that the circuit:



differs from a Toffoli gate only by relative phases. That is, the circuit takes  $|c_1, c_2, t\rangle$  to  $e^{i\theta(c_1, c_2, t)}|c_1, c_2, t \oplus c_1 \cdot c_2\rangle$ , where  $e^{i\theta(c_1, c_2, t)}$  is some relative phase factor. Such gates can sometimes be useful in experimental implementations, where it may be much easier to implement a gate that is the same as the Toffoli up to relative phases than it is to do the Toffoli directly.

**Exercise 4.27:** Using just CNOTs and Toffoli gates, construct a quantum circuit to perform the transformation

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \quad (4.31)$$

This kind of partial cyclic permutation operation will be useful later, in Chapter 7.

How may we implement  $C^n(U)$  gates using our existing repertoire of gates, where  $U$  is an arbitrary single qubit unitary operation? A particularly simple circuit for achieving this task is illustrated in Figure 4.10. The circuit divides up into three stages, and makes use of a small number  $(n - 1)$  of working qubits, which all start and end in the state  $|0\rangle$ . Suppose the control qubits are in the computational basis state  $|c_1, c_2, \dots, c_n\rangle$ . The first stage of the circuit is to reversibly AND all the control bits  $c_1, \dots, c_n$  together to produce the product  $c_1 \cdot c_2 \dots c_n$ . To do this, the first gate in the circuit ANDs  $c_1$  and  $c_2$  together, using a Toffoli gate, changing the state of the first work qubit to  $|c_1 \cdot c_2\rangle$ . The next Toffoli gate ANDs  $c_3$  with the product  $c_1 \cdot c_2$ , changing the state of the second work qubit to  $|c_1 \cdot c_2 \cdot c_3\rangle$ . We continue applying Toffoli gates in this fashion, until the final work qubit is in the state  $|c_1 \cdot c_2 \dots c_n\rangle$ . Next, a  $U$  operation on the target qubit is

performed, conditional on the final work qubit being set to one. That is,  $U$  is applied if and only if all of  $c_1$  through  $c_n$  are set. Finally, the last part of the circuit just reverses the steps of the first stage, returning all the work qubits to their initial state,  $|0\rangle$ . The combined result, therefore, is to apply the unitary operator  $U$  to the target qubit, if and only if all the control bits  $c_1$  through  $c_n$  are set, as desired.

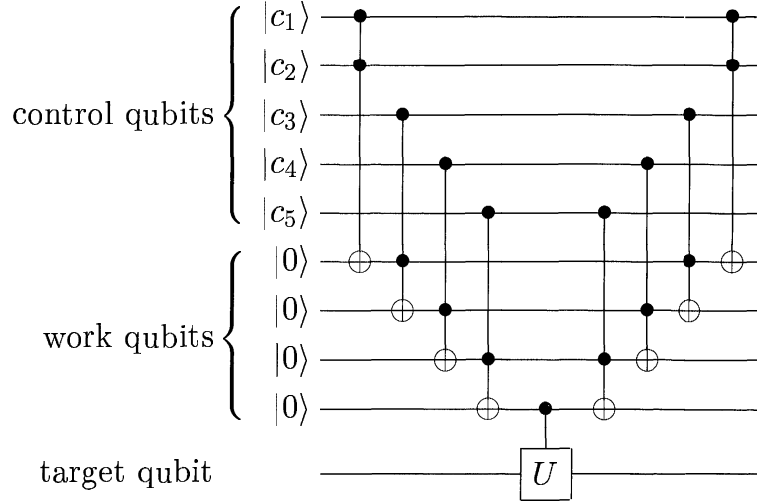


Figure 4.10. Network implementing the  $C^n(U)$  operation, for the case  $n = 5$ .

**Exercise 4.28:** For  $U = V^2$  with  $V$  unitary, construct a  $C^5(U)$  gate analogous to that in Figure 4.10, but using no work qubits. You may use controlled- $V$  and controlled- $V^\dagger$  gates.

**Exercise 4.29:** Find a circuit containing  $O(n^2)$  Toffoli, CNOT and single qubit gates which implements a  $C^n(X)$  gate (for  $n > 3$ ), using no work qubits.

**Exercise 4.30:** Suppose  $U$  is a single qubit unitary operation. Find a circuit containing  $O(n^2)$  Toffoli, CNOT and single qubit gates which implements a  $C^n(U)$  gate (for  $n > 3$ ), using no work qubits.

In the controlled gates we have been considering, conditional dynamics on the target qubit occurs if the control bits are set to *one*. Of course, there is nothing special about one, and it is often useful to consider dynamics which occur conditional on the control bit being set to zero. For instance, suppose we wish to implement a two qubit gate in which the second ('target') qubit is flipped, conditional on the first ('control') qubit being set to zero. In Figure 4.11 we introduce a circuit notation for this gate, together with an equivalent circuit in terms of the gates we have already introduced. Generically we shall use the open circle notation to indicate conditioning on the qubit being set to zero, while a closed circle indicates conditioning on the qubit being set to one.

A more elaborate example of this convention, involving three control qubits, is illustrated in Figure 4.12. The operation  $U$  is applied to the target qubit if the first and third qubits are set to zero, and the second qubit is set to one. It is easy to verify by inspection that the circuit on the right hand side of the figure implements the desired operation. More generally, it is easy to move between circuits which condition on qubits being set

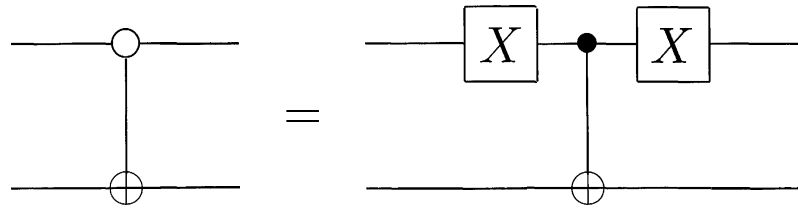


Figure 4.11. Controlled operation with a NOT gate being performed on the second qubit, conditional on the first qubit being set to zero.

to one and circuits which condition on qubits being set to zero, by insertion of  $X$  gates in appropriate locations, as illustrated in Figure 4.12.

Another convention which is sometimes useful is to allow controlled-NOT gates to have multiple targets, as shown in Figure 4.13. This natural notation means that when the control qubit is 1, then all the qubits marked with a  $\oplus$  are flipped, and otherwise nothing happens. It is convenient to use, for example, in constructing classical functions such as permutations, or in encoders and decoders for quantum error-correction circuits, as we shall see in Chapter 10.

**Exercise 4.31: (More circuit identities)** Let subscripts denote which qubit an operator acts on, and let  $C$  be a CNOT with qubit 1 the control qubit and qubit 2 the target qubit. Prove the following identities:

$$CX_1C = X_1X_2 \quad (4.32)$$

$$CY_1C = Y_1X_2 \quad (4.33)$$

$$CZ_1C = Z_1 \quad (4.34)$$

$$CX_2C = X_2 \quad (4.35)$$

$$CY_2C = Z_1Y_2 \quad (4.36)$$

$$CZ_2C = Z_1Z_2 \quad (4.37)$$

$$R_{z,1}(\theta)C = CR_{z,1}(\theta) \quad (4.38)$$

$$R_{x,2}(\theta)C = CR_{x,2}(\theta). \quad (4.39)$$

## 4.4 Measurement

A final element used in quantum circuits, almost implicitly sometimes, is measurement. In our circuits, we shall denote a projective measurement in the computational basis (Section 2.2.5) using a ‘meter’ symbol, illustrated in Figure 4.14. In the theory of quantum circuits it is conventional to not use any special symbols to denote more general measurements, because, as explained in Chapter 2, they can always be represented by unitary transforms with ancilla qubits followed by projective measurements.

There are two important principles that it is worth bearing in mind about quantum circuits. Both principles are rather obvious; however, they are of such great utility that they are worth emphasizing early. The first principle is that classically conditioned operations can be replaced by quantum conditioned operations:

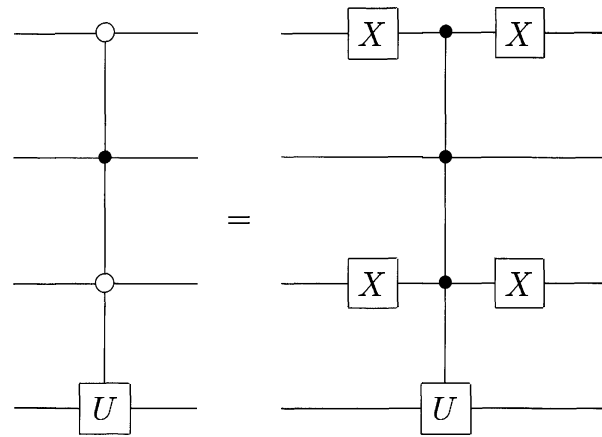


Figure 4.12. Controlled- $U$  operation and its equivalent in terms of circuit elements we already know how to implement. The fourth qubit has  $U$  applied if the first and third qubits are set to zero, and the second qubit is set to one.



Figure 4.13. Controlled-NOT gate with multiple targets.

**Principle of deferred measurement:** Measurements can always be moved from an intermediate stage of a quantum circuit to the end of the circuit; if the measurement results are used at any stage of the circuit then the classically controlled operations can be replaced by conditional quantum operations.

Often, quantum measurements are performed as an intermediate step in a quantum circuit, and the measurement results are used to conditionally control subsequent quantum gates. This is the case, for example, in the teleportation circuit of Figure 1.13 on page 27. However, such measurements can *always* be moved to the end of the circuit. Figure 4.15 illustrates how this may be done by replacing all the classical conditional operations by corresponding quantum conditional operations. (Of course, some of the interpretation of this circuit as performing ‘teleportation’ is lost, because no classical information is transmitted from Alice to Bob, but it is clear that the overall action of the two quantum circuits is the same, which is the key point.)

The second principle is even more obvious – and surprisingly useful!

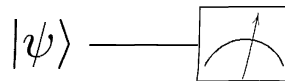


Figure 4.14. Symbol for projective measurement on a single qubit. In this circuit nothing further is done with the measurement result, but in more general quantum circuits it is possible to change later parts of the quantum circuit, *conditional* on measurement outcomes in earlier parts of the circuit. Such a usage of classical information is depicted using wires drawn with double lines (not shown here).



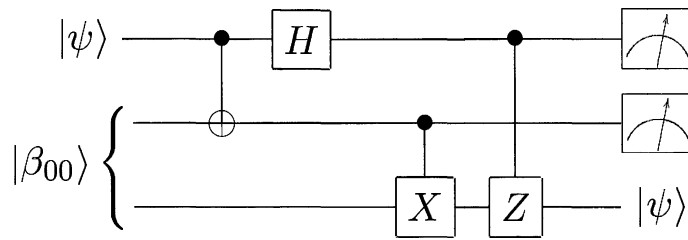


Figure 4.15. Quantum teleportation circuit in which measurements are done at the end, instead of in the middle of the circuit. As in Figure 1.13, the top two qubits belong to Alice, and the bottom one to Bob.

**Principle of implicit measurement:** Without loss of generality, any unterminated quantum wires (qubits which are not measured) at the end of a quantum circuit may be assumed to be measured.

To understand why this is true, imagine you have a quantum circuit containing just two qubits, and only the first qubit is measured at the end of the circuit. Then the measurement statistics observed at this time are completely determined by the reduced density matrix of the first qubit. However, if a measurement had also been performed on the second qubit, then it would be highly surprising if that measurement could change the statistics of measurement on the first qubit. You'll prove this in Exercise 4.32 by showing that the reduced density matrix of the first qubit is not affected by performing a measurement on the second.

As you consider the role of measurements in quantum circuits, it is important to keep in mind that in its role as an interface between the quantum and classical worlds, measurement is generally considered to be an irreversible operation, destroying quantum information and replacing it with classical information. In certain carefully designed cases, however, this need not be true, as is vividly illustrated by teleportation and quantum error-correction (Chapter 10). What teleportation and quantum error-correction have in common is that in neither instance does the measurement result reveal any information about the identity of the quantum state being measured. Indeed, we will see in Chapter 10 that this is a more general feature of measurement – in order for a measurement to be reversible, it must reveal no information about the quantum state being measured!

**Exercise 4.32:** Suppose  $\rho$  is the density matrix describing a two qubit system.

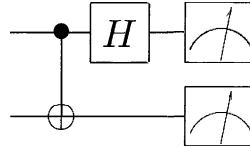
Suppose we perform a projective measurement in the computational basis of the second qubit. Let  $P_0 = |0\rangle\langle 0|$  and  $P_1 = |1\rangle\langle 1|$  be the projectors onto the  $|0\rangle$  and  $|1\rangle$  states of the second qubit, respectively. Let  $\rho'$  be the density matrix which would be assigned to the system after the measurement by an observer who did not learn the measurement result. Show that

$$\rho' = P_0 \rho P_0 + P_1 \rho P_1. \quad (4.40)$$

Also show that the reduced density matrix for the first qubit is not affected by the measurement, that is,  $\text{tr}_2(\rho) = \text{tr}_2(\rho')$ .

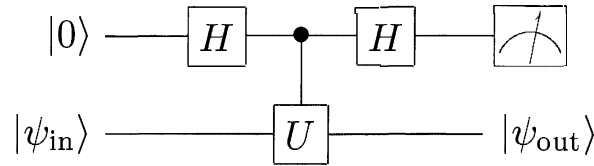
**Exercise 4.33: (Measurement in the Bell basis)** The measurement model we have specified for the quantum circuit model is that measurements are performed only

in the computational basis. However, often we want to perform a measurement in some other basis, defined by a complete set of orthonormal states. To perform this measurement, simply unitarily transform from the basis we wish to perform the measurement in to the computational basis, then measure. For example, show that the circuit

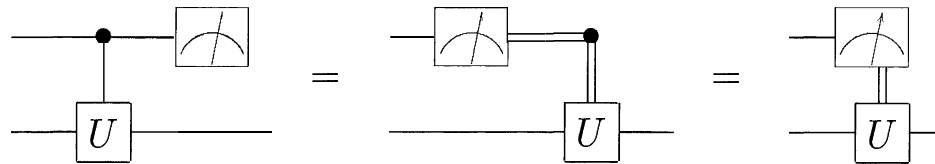


performs a measurement in the basis of the Bell states. More precisely, show that this circuit results in a measurement being performed with corresponding POVM elements the four projectors onto the Bell states. What are the corresponding measurement operators?

**Exercise 4.34: (Measuring an operator)** Suppose we have a single qubit operator  $U$  with eigenvalues  $\pm 1$ , so that  $U$  is both Hermitian and unitary, so it can be regarded both as an observable and a quantum gate. Suppose we wish to measure the observable  $U$ . That is, we desire to obtain a measurement result indicating one of the two eigenvalues, and leaving a post-measurement state which is the corresponding eigenvector. How can this be implemented by a quantum circuit? Show that the following circuit implements a measurement of  $U$ :



**Exercise 4.35: (Measurement commutes with controls)** A consequence of the principle of deferred measurement is that measurements commute with quantum gates when the qubit being measured is a control qubit, that is:



(Recall that the double lines represent classical bits in this diagram.) Prove the first equality. The rightmost circuit is simply a convenient notation to depict the use of a measurement result to classically control a quantum gate.

## 4.5 Universal quantum gates

A small set of gates (e.g. AND, OR, NOT) can be used to compute an arbitrary classical function, as we saw in Section 3.1.2. We say that such a set of gates is *universal* for classical computation. In fact, since the Toffoli gate is universal for classical computation, quantum circuits subsume classical circuits. A similar universality result is true for quantum computation, where a set of gates is said to be *universal for quantum computation* if any unitary operation may be approximated to arbitrary accuracy by a quantum circuit

involving only those gates. We now describe three universality constructions for quantum computation. These constructions build upon each other, and culminate in a proof that any unitary operation can be approximated to arbitrary accuracy using Hadamard, phase, CNOT, and  $\pi/8$  gates. You may wonder why the phase gate appears in this list, since it can be constructed from two  $\pi/8$  gates; it is included because of its natural role in the fault-tolerant constructions described in Chapter 10.

The first construction shows that an arbitrary unitary operator may be expressed *exactly* as a product of unitary operators that each acts non-trivially only on a subspace spanned by two computational basis states. The second construction combines the first construction with the results of the previous section to show that an arbitrary unitary operator may be expressed *exactly* using single qubit and CNOT gates. The third construction combines the second construction with a proof that single qubit operation may be approximated to arbitrary accuracy using the Hadamard, phase, and  $\pi/8$  gates. This in turn implies that any unitary operation can be approximated to arbitrary accuracy using Hadamard, phase, CNOT, and  $\pi/8$  gates.

Our constructions say little about efficiency – how many (polynomially or exponentially many) gates must be composed in order to create a given unitary transform. In Section 4.5.4 we show that there *exist* unitary transforms which require exponentially many gates to approximate. Of course, the goal of quantum computation is to find interesting families of unitary transformations that *can* be performed efficiently.

**Exercise 4.36:** Construct a quantum circuit to add two two-bit numbers  $x$  and  $y$  modulo 4. That is, the circuit should perform the transformation  $|x, y\rangle \rightarrow |x, x + y \bmod 4\rangle$ .

#### 4.5.1 Two-level unitary gates are universal

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

$$U = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & j \end{bmatrix}. \quad (4.41)$$

We will find two-level unitary matrices  $U_1, \dots, U_3$  such that

$$U_3 U_2 U_1 U = I. \quad (4.42)$$

It follows that

$$U = U_1^\dagger U_2^\dagger U_3^\dagger. \quad (4.43)$$

$U_1, U_2$  and  $U_3$  are all two-level unitary matrices, and it is easy to see that their inverses,  $U_1^\dagger, U_2^\dagger$  and  $U_3^\dagger$  are also two-level unitary matrices. Thus, if we can demonstrate (4.42), then we will have shown how to break  $U$  up into a product of two-level unitary matrices.

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

$$U_1 \equiv \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (4.44)$$

If  $b \neq 0$  then set

$$U_1 \equiv \begin{bmatrix} \frac{a^*}{\sqrt{|a|^2+|b|^2}} & \frac{b^*}{\sqrt{|a|^2+|b|^2}} & 0 \\ \frac{b}{\sqrt{|a|^2+|b|^2}} & \frac{-a}{\sqrt{|a|^2+|b|^2}} & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (4.45)$$

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

$$U_1 U = \begin{bmatrix} a' & d' & g' \\ 0 & e' & h' \\ c' & f' & j' \end{bmatrix}. \quad (4.46)$$

The key point to note is that the middle entry in the left hand column is zero. We denote the other entries in the matrix with a generic prime  $'$ ; their actual values do not matter.

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

$$U_2 \equiv \begin{bmatrix} a'^* & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad (4.47)$$

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

$$U_2 \equiv \begin{bmatrix} \frac{a'^*}{\sqrt{|a'|^2+|c'|^2}} & 0 & \frac{c'^*}{\sqrt{|a'|^2+|c'|^2}} \\ 0 & 1 & 0 \\ \frac{c'}{\sqrt{|a'|^2+|c'|^2}} & 0 & \frac{-a'}{\sqrt{|a'|^2+|c'|^2}} \end{bmatrix}. \quad (4.48)$$

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

$$U_2 U_1 U = \begin{bmatrix} 1 & d'' & g'' \\ 0 & e'' & h'' \\ 0 & f'' & j'' \end{bmatrix}. \quad (4.49)$$

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

$$U_3 \equiv \begin{bmatrix} 1 & 0 & 0 \\ 0 & e''^* & f''^* \\ 0 & h''^* & j''^* \end{bmatrix}. \quad (4.50)$$

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

More generally, suppose  $U$  acts on a  $d$ -dimensional space. Then, in a similar fashion to the  $3 \times 3$  case, we can find two-level unitary matrices  $U_1, \dots, U_{d-1}$  such that the matrix

$U_{d-1}U_{d-2}\dots U_1U$  has a one in the top left hand corner, and all zeroes elsewhere in the first row and column. We then repeat this procedure for the  $d-1$  by  $d-1$  unitary submatrix in the lower right hand corner of  $U_{d-1}U_{d-2}\dots U_1U$ , and so on, with the end result that an arbitrary  $d\times d$  unitary matrix may be written

$$U = V_1 \dots V_k, \quad (4.51)$$

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

**Exercise 4.37:** Provide a decomposition of the transform

$$\frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix} \quad (4.52)$$

into a product of two-level unitaries. This is a special case of the quantum Fourier transform, which we study in more detail in the next chapter.

A corollary of the above result is that an arbitrary unitary matrix on an  $n$  qubit system may be written as a product of at most  $2^{n-1}(2^n - 1)$  two-level unitary matrices. For specific unitary matrices, it may be possible to find much more efficient decompositions, but as you will now show there exist matrices which *cannot* be decomposed as a product of fewer than  $d-1$  two-level unitary matrices!

**Exercise 4.38:** Prove that there exists a  $d\times d$  unitary matrix  $U$  which cannot be decomposed as a product of fewer than  $d-1$  two-level unitary matrices.

### 4.5.2 Single qubit and CNOT gates are universal

We have just shown that an arbitrary unitary matrix on a  $d$ -dimensional Hilbert space may be written as a product of two-level unitary matrices. Now we show that single qubit and CNOT gates together can be used to implement an arbitrary two-level unitary operation on the state space of  $n$  qubits. Combining these results we see that single qubit and CNOT gates can be used to implement an arbitrary unitary operation on  $n$  qubits, and therefore are universal for quantum computation.

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

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

code

$$\begin{array}{cccccc}
 1 & 0 & 1 & 0 & 0 & 1 \\
 1 & 0 & 1 & 0 & 1 & 1 \\
 1 & 0 & 0 & 0 & 1 & 1 \\
 1 & 1 & 0 & 0 & 1 & 1
 \end{array} \tag{4.53}$$

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

The basic idea of the quantum circuit implementing  $U$  is to perform a sequence of gates effecting the state changes  $|g_1\rangle \rightarrow |g_2\rangle \rightarrow \dots \rightarrow |g_{m-1}\rangle$ , then to perform a controlled- $\tilde{U}$  operation, with the target qubit located at the single bit where  $g_{m-1}$  and  $g_m$  differ, and then to undo the first stage, transforming  $|g_{m-1}\rangle \rightarrow |g_{m-2}\rangle \rightarrow \dots \rightarrow |g_1\rangle$ . Each of these steps can be easily implemented using operations developed earlier in this chapter, and the final result is an implementation of  $U$ .

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

$$|g_1\rangle \rightarrow |g_{m-1}\rangle \tag{4.54}$$

$$|g_2\rangle \rightarrow |g_1\rangle \tag{4.55}$$

$$|g_3\rangle \rightarrow |g_2\rangle \tag{4.56}$$

.....

$$|g_{m-1}\rangle \rightarrow |g_{m-2}\rangle. \tag{4.57}$$

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

A simple example illuminates the procedure further. Suppose we wish to implement the two-level unitary transformation

$$U = \begin{bmatrix} a & 0 & 0 & 0 & 0 & 0 & 0 & c \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ b & 0 & 0 & 0 & 0 & 0 & 0 & d \end{bmatrix}. \tag{4.58}$$

Here,  $a, b, c$  and  $d$  are any complex numbers such that  $\tilde{U} \equiv \begin{bmatrix} a & c \\ b & d \end{bmatrix}$  is a unitary matrix.

Notice that  $U$  acts non-trivially only on the states  $|000\rangle$  and  $|111\rangle$ . We write a Gray code connecting 000 and 111:

$$\begin{array}{ccc}
 A & B & C \\
 0 & 0 & 0 \\
 0 & 0 & 1 \\
 0 & 1 & 1 \\
 1 & 1 & 1
 \end{array} \quad (4.59)$$

From this we read off the required circuit, shown in Figure 4.16. The first two gates shuffle the states so that  $|000\rangle$  gets swapped with  $|011\rangle$ . Next, the operation  $\tilde{U}$  is applied to the first qubit of the states  $|011\rangle$  and  $|111\rangle$ , conditional on the second and third qubits being in the state  $|11\rangle$ . Finally, we unshuffle the states, ensuring that  $|011\rangle$  gets swapped back with the state  $|000\rangle$ .

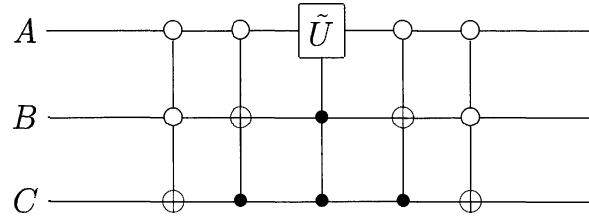


Figure 4.16. Circuit implementing the two-level unitary operation defined by (4.58).

Returning to the general case, we see that implementing the two-level unitary operation  $U$  requires at most  $2(n-1)$  controlled operations to swap  $|g_1\rangle$  with  $|g_{m-1}\rangle$  and then back again. Each of these controlled operations can be realized using  $O(n)$  single qubit and CNOT gates; the controlled- $\tilde{U}$  operation also requires  $O(n)$  gates. Thus, implementing  $U$  requires  $O(n^2)$  single qubit and CNOT gates. We saw in the previous section that an arbitrary unitary matrix on the  $2^n$ -dimensional state space of  $n$  qubits may be written as a product of  $O(2^{2n}) = O(4^n)$  two-level unitary operations. Combining these results, we see that an arbitrary unitary operation on  $n$  qubits can be implemented using a circuit containing  $O(n^2 4^n)$  single qubit and CNOT gates. Obviously, this construction does not provide terribly efficient quantum circuits! However, we show in Section 4.5.4 that the construction is close to optimal in the sense that there are unitary operations that require an exponential number of gates to implement. Thus, to find fast quantum algorithms we will clearly need a different approach than is taken in the universality construction.

**Exercise 4.39:** Find a quantum circuit using single qubit operations and CNOTs to implement the transformation

$$\begin{bmatrix}
 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & a & 0 & 0 & 0 & 0 & c \\
 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
 0 & 0 & b & 0 & 0 & 0 & 0 & d
 \end{bmatrix}, \quad (4.60)$$

where  $\tilde{U} = \begin{bmatrix} a & c \\ b & d \end{bmatrix}$  is an arbitrary  $2 \times 2$  unitary matrix.

### 4.5.3 A discrete set of universal operations

In the previous section we proved that the CNOT and single qubit unitaries together form a universal set for quantum computation. Unfortunately, no straightforward method is known to implement all these gates in a fashion which is resistant to errors. Fortunately, in this section we'll find a discrete set of gates which can be used to perform universal quantum computation, and in Chapter 10 we'll show how to perform these gates in an error-resistant fashion, using quantum error-correcting codes.

#### *Approximating unitary operators*

Obviously, a discrete set of gates can't be used to implement an arbitrary unitary operation *exactly*, since the set of unitary operations is continuous. Rather, it turns out that a discrete set can be used to *approximate* any unitary operation. To understand how this works, we first need to study what it means to approximate a unitary operation. Suppose  $U$  and  $V$  are two unitary operators on the same state space.  $U$  is the target unitary operator that we wish to implement, and  $V$  is the unitary operator that is actually implemented in practice. We define the *error* when  $V$  is implemented instead of  $U$  by

$$E(U, V) \equiv \max_{|\psi\rangle} \|(U - V)|\psi\rangle\|, \quad (4.61)$$

where the maximum is over all normalized quantum states  $|\psi\rangle$  in the state space. In Box 4.1 on page 195 we show that this measure of error has the interpretation that if  $E(U, V)$  is small, then any measurement performed on the state  $V|\psi\rangle$  will give approximately the same measurement statistics as a measurement of  $U|\psi\rangle$ , for any initial state  $|\psi\rangle$ . More precisely, we show that if  $M$  is a POVM element in an arbitrary POVM, and  $P_U$  (or  $P_V$ ) is the probability of obtaining this outcome if  $U$  (or  $V$ ) were performed with a starting state  $|\psi\rangle$ , then

$$|P_U - P_V| \leq 2E(U, V). \quad (4.62)$$

Thus, if  $E(U, V)$  is small, then measurement outcomes occur with similar probabilities, regardless of whether  $U$  or  $V$  were performed. Also shown in Box 4.1 is that if we perform a sequence of gates  $V_1, \dots, V_m$  intended to approximate some other sequence of gates  $U_1, \dots, U_m$ , then the errors add at most linearly,

$$E(U_m U_{m-1} \dots U_1, V_m V_{m-1} \dots V_1) \leq \sum_{j=1}^m E(U_j, V_j). \quad (4.63)$$

The approximation results (4.62) and (4.63) are extremely useful. Suppose we wish to perform a quantum circuit containing  $m$  gates,  $U_1$  through  $U_m$ . Unfortunately, we are only able to approximate the gate  $U_j$  by the gate  $V_j$ . In order that the probabilities of different measurement outcomes obtained from the approximate circuit be within a tolerance  $\Delta > 0$  of the correct probabilities, it suffices that  $E(U_j, V_j) \leq \Delta/(2m)$ , by the results (4.62) and (4.63).

#### *Universality of Hadamard + phase + CNOT + $\pi/8$ gates*

We're now in a good position to study the approximation of arbitrary unitary operations by discrete sets of gates. We're going to consider two different discrete sets of gates, both



**Box 4.1: Approximating quantum circuits**

Suppose a quantum system starts in the state  $|\psi\rangle$ , and we perform either the unitary operation  $U$ , or the unitary operation  $V$ . Following this, we perform a measurement. Let  $M$  be a POVM element associated with the measurement, and let  $P_U$  (or  $P_V$ ) be the probability of obtaining the corresponding measurement outcome if the operation  $U$  (or  $V$ ) was performed. Then

$$|P_U - P_V| = |\langle\psi|U^\dagger MU|\psi\rangle - \langle\psi|V^\dagger MV|\psi\rangle|. \quad (4.64)$$

Let  $|\Delta\rangle \equiv (U - V)|\psi\rangle$ . Simple algebra and the Cauchy–Schwarz inequality show that

$$|P_U - P_V| = |\langle\psi|U^\dagger M|\Delta\rangle + \langle\Delta|MV|\psi\rangle|. \quad (4.65)$$

$$\leq |\langle\psi|U^\dagger M|\Delta\rangle| + |\langle\Delta|MV|\psi\rangle| \quad (4.66)$$

$$\leq \|\Delta\| + \|\Delta\| \quad (4.67)$$

$$\leq 2E(U, V). \quad (4.68)$$

The inequality  $|P_U - P_V| \leq 2E(U, V)$  gives quantitative expression to the idea that when the error  $E(U, V)$  is small, the difference in probabilities between measurement outcomes is also small.

Suppose we perform a sequence  $V_1, V_2, \dots, V_m$  of gates intended to approximate some other sequence of gates,  $U_1, U_2, \dots, U_m$ . Then it turns out that the error caused by the entire sequence of imperfect gates is at most the sum of the errors in the individual gates,

$$E(U_m U_{m-1} \dots U_1, V_m V_{m-1} \dots V_1) \leq \sum_{j=1}^m E(U_j, V_j). \quad (4.69)$$

To prove this we start with the case  $m = 2$ . Note that for some state  $|\psi\rangle$  we have

$$E(U_2 U_1, V_2 V_1) = \|(U_2 U_1 - V_2 V_1)|\psi\rangle\| \quad (4.70)$$

$$= \|(U_2 U_1 - V_2 U_1)|\psi\rangle + (V_2 U_1 - V_2 V_1)|\psi\rangle\|. \quad (4.71)$$

Using the triangle inequality  $\| |a\rangle + |b\rangle \| \leq \| |a\rangle \| + \| |b\rangle \|$ , we obtain

$$E(U_2 U_1, V_2 V_1) \leq \|(U_2 - V_2)U_1|\psi\rangle\| + \|V_2(U_1 - V_1)|\psi\rangle\| \quad (4.72)$$

$$\leq E(U_2, V_2) + E(U_1, V_1), \quad (4.73)$$

which was the desired result. The result for general  $m$  follows by induction.

of which are universal. The first set, the *standard set* of universal gates, consists of the Hadamard, phase, controlled-NOT and  $\pi/8$  gates. We provide fault-tolerant constructions for these gates in Chapter 10; they also provide an exceptionally simple universality construction. The second set of gates we consider consists of the Hadamard gate, phase gate, the controlled-NOT gate, and the Toffoli gate. These gates can also all be done fault-tolerantly; however, the universality proof and fault-tolerance construction for these gates is a little less appealing.

We begin the universality proof by showing that the Hadamard and  $\pi/8$  gates can be

used to approximate any single qubit unitary operation to arbitrary accuracy. Consider the gates  $T$  and  $HTH$ .  $T$  is, up to an unimportant global phase, a rotation by  $\pi/4$  radians around the  $\hat{z}$  axis on the Bloch sphere, while  $HTH$  is a rotation by  $\pi/4$  radians around the  $\hat{x}$  axis on the Bloch sphere (Exercise 4.14). Composing these two operations gives, up to a global phase,

$$\begin{aligned} \exp\left(-i\frac{\pi}{8}Z\right)\exp\left(-i\frac{\pi}{8}X\right) &= \left[\cos\frac{\pi}{8}I - i\sin\frac{\pi}{8}Z\right]\left[\cos\frac{\pi}{8}I - i\sin\frac{\pi}{8}X\right] \quad (4.74) \\ &= \cos^2\frac{\pi}{8}I - i\left[\cos\frac{\pi}{8}(X+Z) + \sin\frac{\pi}{8}Y\right]\sin\frac{\pi}{8}. \end{aligned} \quad (4.75)$$

This is a rotation of the Bloch sphere about an axis along  $\vec{n} = (\cos\frac{\pi}{8}, \sin\frac{\pi}{8}, \cos\frac{\pi}{8})$  with corresponding unit vector  $\hat{n}$ , and through an angle  $\theta$  defined by  $\cos(\theta/2) \equiv \cos^2\frac{\pi}{8}$ . That is, using only the Hadamard and  $\pi/8$  gates we can construct  $R_{\hat{n}}(\theta)$ . Moreover, this  $\theta$  can be shown to be an irrational multiple of  $2\pi$ . Proving this latter fact is a little beyond our scope; see the end of chapter ‘History and further reading’.

Next, we show that repeated iteration of  $R_{\hat{n}}(\theta)$  can be used to approximate to arbitrary accuracy any rotation  $R_{\hat{n}}(\alpha)$ . To see this, let  $\delta > 0$  be the desired accuracy, and let  $N$  be an integer larger than  $2\pi/\delta$ . Define  $\theta_k$  so that  $\theta_k \in [0, 2\pi)$  and  $\theta_k = (k\theta) \bmod 2\pi$ . Then the pigeonhole principle implies that there are distinct  $j$  and  $k$  in the range  $1, \dots, N$  such that  $|\theta_k - \theta_j| \leq 2\pi/N < \delta$ . Without loss of generality assume that  $k > j$ , so we have  $|\theta_{k-j}| < \delta$ . Since  $j \neq k$  and  $\theta$  is an irrational multiple of  $2\pi$  we must have  $\theta_{k-j} \neq 0$ . It follows that the sequence  $\theta_{l(k-j)}$  fills up the interval  $[0, 2\pi)$  as  $l$  is varied, so that adjacent members of the sequence are no more than  $\delta$  apart. It follows that for any  $\epsilon > 0$  there exists an  $n$  such that

$$E(R_{\hat{n}}(\alpha), R_{\hat{n}}(\theta)^n) < \frac{\epsilon}{3}. \quad (4.76)$$

**Exercise 4.40:** For arbitrary  $\alpha$  and  $\beta$  show that

$$E(R_{\hat{n}}(\alpha), R_{\hat{n}}(\alpha + \beta)) = |1 - \exp(i\beta/2)|, \quad (4.77)$$

and use this to justify (4.76).

We are now in position to verify that any single qubit operation can be approximated to arbitrary accuracy using the Hadamard and  $\pi/8$  gates. Simple algebra implies that for any  $\alpha$

$$HR_{\hat{n}}(\alpha)H = R_{\hat{m}}(\alpha), \quad (4.78)$$

where  $\hat{m}$  is a unit vector in the direction  $(\cos\frac{\pi}{8}, -\sin\frac{\pi}{8}, \cos\frac{\pi}{8})$ , from which it follows that

$$E(R_{\hat{m}}(\alpha), R_{\hat{m}}(\theta)^n) < \frac{\epsilon}{3}. \quad (4.79)$$

But by Exercise 4.11 an arbitrary unitary  $U$  on a single qubit may be written as

$$U = R_{\hat{n}}(\beta)R_{\hat{m}}(\gamma)R_{\hat{n}}(\delta), \quad (4.80)$$

up to an unimportant global phase shift. The results (4.76) and (4.79), together with the

chaining inequality (4.63) therefore imply that for suitable positive integers  $n_1, n_2, n_3$ ,

$$E(U, R_{\hat{n}}(\theta)^{n_1} H R_{\hat{n}}(\theta)^{n_2} H R_{\hat{n}}(\theta)^{n_3}) < \epsilon. \quad (4.81)$$

That is, given any single qubit unitary operator  $U$  and any  $\epsilon > 0$  it is possible to approximate  $U$  to within  $\epsilon$  using a circuit composed of Hadamard gates and  $\pi/8$  gates alone.

Since the  $\pi/8$  and Hadamard gates allow us to approximate any single qubit unitary operator, it follows from the arguments of Section 4.5.2 that we can approximate any  $m$  gate quantum circuit, as follows. Given a quantum circuit containing  $m$  gates, either CNOTs or single qubit unitary gates, we may approximate it using Hadamard, controlled-NOT and  $\pi/8$  gates (later, we will find that phase gates make it possible to do the approximation fault-tolerantly, but for the present universality argument they are not strictly necessary). If we desire an accuracy of  $\epsilon$  for the entire circuit, then this may be achieved by approximating each single qubit unitary using the above procedure to within  $\epsilon/m$  and applying the chaining inequality (4.63) to obtain an accuracy of  $\epsilon$  for the entire circuit.

How efficient is this procedure for approximating quantum circuits using a discrete set of gates? This is an important question. Suppose, for example, that approximating an arbitrary single qubit unitary to within a distance  $\epsilon$  were to require  $\Omega(2^{1/\epsilon})$  gates from the discrete set. Then to approximate the  $m$  gate quantum circuit considered in the previous paragraph would require  $\Omega(m2^{m/\epsilon})$  gates, an exponential increase over the original circuit size! Fortunately, the rate of convergence is much better than this. Intuitively, it is plausible that the sequence of angles  $\theta_k$  ‘fills in’ the interval  $[0, 2\pi)$  in a more or less uniform fashion, so that to approximate an arbitrary single qubit gate ought to take roughly  $\Theta(1/\epsilon)$  gates from the discrete set. If we use this estimate for the number of gates required to approximate an arbitrary single qubit gate, then the number required to approximate an  $m$  gate circuit to accuracy  $\epsilon$  becomes  $\Theta(m^2/\epsilon)$ . This is a quadratic increase over the original size of the circuit,  $m$ , which for many applications may be sufficient.

Rather remarkably, however, a much faster rate of convergence can be proved. The *Solovay–Kitaev theorem*, proved in Appendix 3, implies that an arbitrary single qubit gate may be approximated to an accuracy  $\epsilon$  using  $O(\log^c(1/\epsilon))$  gates from our discrete set, where  $c$  is a constant approximately equal to 2. The Solovay–Kitaev theorem therefore implies that to approximate a circuit containing  $m$  CNOTs and single qubit unitaries to an accuracy  $\epsilon$  requires  $O(m \log^c(m/\epsilon))$  gates from the discrete set, a polylogarithmic increase over the size of the original circuit, which is likely to be acceptable for virtually all applications.

To sum up, we have shown that the Hadamard, phase, controlled-NOT and  $\pi/8$  gates are universal for quantum computation in the sense that given a circuit containing CNOTs and arbitrary single qubit unitaries it is possible to simulate this circuit to good accuracy using only this discrete set of gates. Moreover, the simulation can be performed efficiently, in the sense that the overhead required to perform the simulation is polynomial in  $\log(m/\epsilon)$ , where  $m$  is the number of gates in the original circuit, and  $\epsilon$  is the desired accuracy of the simulation.

**Exercise 4.41:** This and the next two exercises develop a construction showing that the Hadamard, phase, controlled-NOT and Toffoli gates are universal. Show that

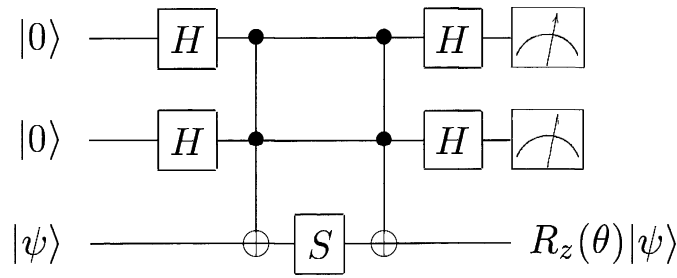


Figure 4.17. Provided both measurement outcomes are 0 this circuit applies  $R_z(\theta)$  to the target, where  $\cos \theta = 3/5$ . If some other measurement outcome occurs then the circuit applies  $Z$  to the target.

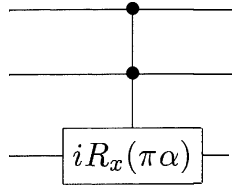
the circuit in Figure 4.17 applies the operation  $R_z(\theta)$  to the third (target) qubit if the measurement outcomes are both 0, where  $\cos \theta = 3/5$ , and otherwise applies  $Z$  to the target qubit. Show that the probability of both measurement outcomes being 0 is  $5/8$ , and explain how repeated use of this circuit and  $Z = S^2$  gates may be used to apply a  $R_z(\theta)$  gate with probability approaching 1.

**Exercise 4.42: (Irrationality of  $\theta$ )** Suppose  $\cos \theta = 3/5$ . We give a proof by contradiction that  $\theta$  is an irrational multiple of  $2\pi$ .

- (1) Using the fact that  $e^{i\theta} = (3 + 4i)/5$ , show that if  $\theta$  is rational, then there must exist a positive integer  $m$  such that  $(3 + 4i)^m = 5^m$ .
- (2) Show that  $(3 + 4i)^m \equiv 3 + 4i \pmod{5}$  for all  $m > 0$ , and conclude that no  $m$  such that  $(3 + 4i)^m = 5^m$  can exist.

**Exercise 4.43:** Use the results of the previous two exercises to show that the Hadamard, phase, controlled-NOT and Toffoli gates are universal for quantum computation.

**Exercise 4.44:** Show that the three qubit gate  $G$  defined by the circuit:



is universal for quantum computation whenever  $\alpha$  is irrational.

**Exercise 4.45:** Suppose  $U$  is a unitary transform implemented by an  $n$  qubit quantum circuit constructed from  $H$ ,  $S$ , CNOT and Toffoli gates. Show that  $U$  is of the form  $2^{-k/2}M$ , for some integer  $k$ , where  $M$  is a  $2^n \times 2^n$  matrix with only complex integer entries. Repeat this exercise with the Toffoli gate replaced by the  $\pi/8$  gate.

#### 4.5.4 Approximating arbitrary unitary gates is generically hard

We've seen that any unitary transformation on  $n$  qubits can be built up out of a small set of elementary gates. Is it always possible to do this efficiently? That is, given a unitary transformation  $U$  on  $n$  qubits does there always exist a circuit of size polynomial in  $n$  approximating  $U$ ? The answer to this question turns out to be a resounding no: in fact, *most unitary transformations can only be implemented very inefficiently*. One way to see

this is to consider the question: how many gates does it take to generate an arbitrary state of  $n$  qubits? A simple counting argument shows that this requires exponentially many operations, in general; it immediately follows that there are unitary operations requiring exponentially many operations. To see this, suppose we have  $g$  different types of gates available, and each gate works on at most  $f$  input qubits. These numbers,  $f$  and  $g$ , are fixed by the computing hardware we have available, and may be considered to be constants. Suppose we have a quantum circuit containing  $m$  gates, starting from the computational basis state  $|0\rangle^{\otimes n}$ . For any particular gate in the circuit there are therefore at most  $\left[ \begin{smallmatrix} n \\ f \end{smallmatrix} \right]^g = O(n^{fg})$  possible choices. It follows that at most  $O(n^{fgm})$  different states may be computed using  $m$  gates.

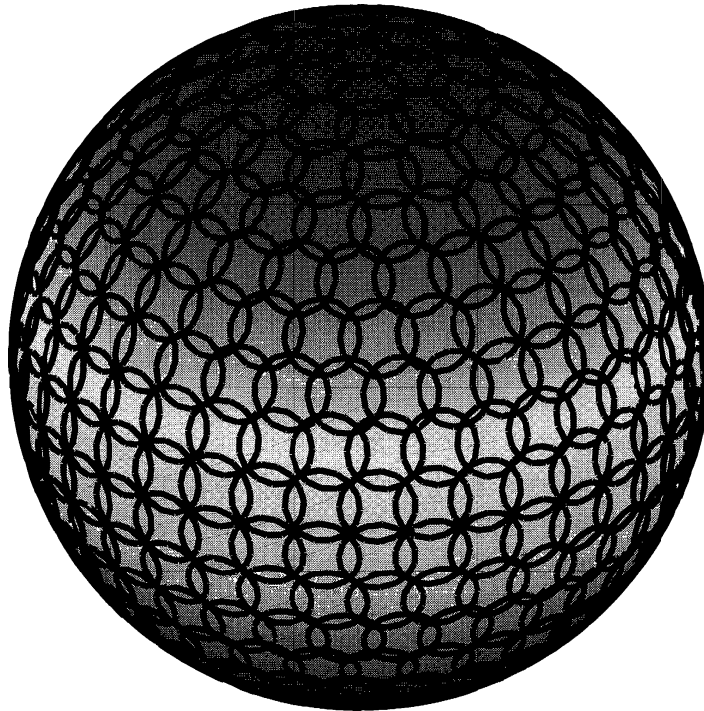


Figure 4.18. Visualization of covering the set of possible states with patches of constant radius.

Suppose we wish to approximate a particular state,  $|\psi\rangle$ , to within a distance  $\epsilon$ . The idea of the proof is to cover the set of all possible states with a collection of ‘patches,’ each of radius  $\epsilon$  (Figure 4.18), and then to show that the number of patches required rises doubly exponentially in  $n$ ; comparing with the exponential number of different states that may be computed using  $m$  gates will imply the result. The first observation we need is that the space of state vectors of  $n$  qubits can be regarded as just the unit  $(2^{n+1} - 1)$ -sphere. To see this, suppose the  $n$  qubit state has amplitudes  $\psi_j = X_j + iY_j$ , where  $X_j$  and  $Y_j$  are the real and imaginary parts, respectively, of the  $j$ th amplitude. The normalization condition for quantum states can be written  $\sum_j (X_j^2 + Y_j^2) = 1$ , which is just the condition for a point to be on the unit sphere in  $2^{n+1}$  real dimensions, that is, the unit  $(2^{n+1} - 1)$ -sphere. Similarly, the surface area of radius  $\epsilon$  near  $|\psi\rangle$  is approximately the same as the volume of a  $(2^{n+1} - 2)$ -sphere of radius  $\epsilon$ . Using the formula  $S_k(r) = 2\pi^{(k+1)/2} r^k / \Gamma((k+1)/2)$  for the surface area of a  $k$ -sphere of radius  $r$ , and  $V_k(r) = 2\pi^{(k+1)/2} r^{k+1} / [(k+1)\Gamma((k+1)/2)]$  for the volume of a  $k$ -sphere of radius  $r$ , we see that the number of patches needed to

cover the state space goes like

$$\frac{S_{2^{n+1}-1}(1)}{V_{2^{n+1}-2}(\epsilon)} = \frac{\sqrt{\pi}\Gamma(2^n - \frac{1}{2})(2^{n+1} - 1)}{\Gamma(2^n)\epsilon^{2^{n+1}-1}}, \quad (4.82)$$

where  $\Gamma$  is the usual generalization of the factorial function. But  $\Gamma(2^n - 1/2) \geq \Gamma(2^n)/2^n$ , so the number of patches required to cover the space is at least

$$\Omega\left(\frac{1}{\epsilon^{2^{n+1}-1}}\right). \quad (4.83)$$

Recall that the number of patches which can be reached in  $m$  gates is  $O(n^{f^{gm}})$ , so in order to reach all the  $\epsilon$ -patches we must have

$$O(n^{f^{gm}}) \geq \Omega\left(\frac{1}{\epsilon^{2^{n+1}-1}}\right) \quad (4.84)$$

which gives us

$$m = \Omega\left(\frac{2^n \log(1/\epsilon)}{\log(n)}\right). \quad (4.85)$$

That is, there are states of  $n$  qubits which take  $\Omega(2^n \log(1/\epsilon)/\log(n))$  operations to approximate to within a distance  $\epsilon$ . This is exponential in  $n$ , and thus is ‘difficult’, in the sense of computational complexity introduced in Chapter 3. Furthermore, this immediately implies that there are unitary transformations  $U$  on  $n$  qubits which take  $\Omega(2^n \log(1/\epsilon)/\log(n))$  operations to approximate by a quantum circuit implementing an operation  $V$  such that  $E(U, V) \leq \epsilon$ . By contrast, using our universality constructions and the Solovay–Kitaev theorem it follows that an arbitrary unitary operation  $U$  on  $n$  qubits may be approximated to within a distance  $\epsilon$  using  $O(n^2 4^n \log^c(n^2 4^n/\epsilon))$  gates. Thus, to within a polynomial factor the construction for universality we have given is optimal; unfortunately, it does not address the problem of determining which families of unitary operations can be computed efficiently in the quantum circuits model.

#### 4.5.5 Quantum computational complexity

In Chapter 3 we described a theory of ‘computational complexity’ for classical computers that classified the resource requirements to solve computational problems on classical computers. Not surprisingly there is considerable interest in developing a theory of quantum computational complexity, and relating it to classical computational complexity theory. Although only first steps have been taken in this direction, it will doubtless be an enormously fruitful direction for future researchers. We content ourselves with presenting one result about quantum complexity classes, relating the quantum complexity class **BQP** to the classical complexity class **PSPACE**. Our discussion of this result is rather informal; for more details you are referred to the paper of Bernstein and Vazirani referenced in the end of chapter ‘History and further reading’.

Recall that **PSPACE** was defined in Chapter 3 as the class of decision problems which can be solved on a Turing machine using space polynomial in the problem size and an arbitrary amount of time. **BQP** is an essentially quantum complexity class consisting of those decision problems that can be solved with bounded probability of error using a polynomial size quantum circuit. Slightly more formally, we say a language  $L$  is in **BQP** if there is a family of polynomial size quantum circuits which decides the language,

accepting strings in the language with probability at least  $3/4$ , and rejecting strings which aren't in the language with probability at least  $3/4$ . In practice, what this means is that the quantum circuit takes as input binary strings, and tries to determine whether they are elements of the language or not. At the conclusion of the circuit one qubit is measured, with 0 indicating that the string has been accepted, and 1 indicating rejection. By testing the string a few times to determine whether it is in  $L$ , we can determine with very high probability whether a given string is in  $L$ .

Of course, a quantum circuit is a fixed entity, and any given quantum circuit can only decide whether strings up to some finite length are in  $L$ . For this reason, we use an entire family of circuits in the definition of **BQP**; for every possible input length there is a different circuit in the family. We place two restrictions on the circuit in addition to the acceptance / rejection criterion already described. First, the size of the circuits should only grow polynomially with the size of the input string  $x$  for which we are trying to determine whether  $x \in L$ . Second, we require that the circuits be *uniformly generated*, in a sense similar to that described in Section 3.1.2. This uniformity requirement arises because, in practice, given a string  $x$  of some length  $n$ , somebody will have to build a quantum circuit capable of deciding whether  $x$  is in  $L$ . To do so, they will need to have a clear set of instructions – an algorithm – for building the circuit. For this reason, we require that our quantum circuits be uniformly generated, that is, there is a Turing machine capable of efficiently outputting a description of the quantum circuit. This restriction may seem rather technical, and in practice is nearly always satisfied trivially, but it does save us from pathological examples such as that described in Section 3.1.2. (You might also wonder if it matters whether the Turing machine used in the uniformity requirement is a quantum or classical Turing machine; it turns out that it doesn't matter – see 'History and further reading'.)

One of the most significant results in quantum computational complexity is that **BQP**  $\subseteq$  **PSPACE**. It is clear that **BPP**  $\subseteq$  **BQP**, where **BPP** is the classical complexity class of decision problems which can be solved with bounded probability of error using polynomial time on a classical Turing machine. Thus we have the chain of inclusions **BPP**  $\subseteq$  **BQP**  $\subseteq$  **PSPACE**. Proving that **BQP**  $\neq$  **BPP** – intuitively the statement that quantum computers are more powerful than classical computers – will therefore imply that **BPP**  $\neq$  **PSPACE**. However, it is not presently known whether **BPP**  $\neq$  **PSPACE**, and proving this would represent a major breakthrough in classical computer science! So proving that quantum computers are more powerful than classical computers would have some very interesting implications for classical computational complexity! Unfortunately, it also means that providing such a proof may be quite difficult.

Why is it that **BQP**  $\subseteq$  **PSPACE**? Here is an intuitive outline of the proof (a rigorous proof is left to the references in 'History and further reading'). Suppose we have an  $n$  qubit quantum computer, and do a computation involving a sequence of  $p(n)$  gates, where  $p(n)$  is some polynomial in  $n$ . Supposing the quantum circuit starts in the state  $|0\rangle$  we will explain how to evaluate in polynomial space on a classical computer the probability that it ends up in the state  $|y\rangle$ . Suppose the gates that are executed on the quantum computer are, in order,  $U_1, U_2, \dots, U_{p(n)}$ . Then the probability of ending up in the state  $|y\rangle$  is the modulus squared of

$$\langle y | U_{p(n)} \cdots U_2 U_1 | 0 \rangle. \quad (4.86)$$

This quantity may be estimated in polynomial space on a classical computer. The basic

idea is to insert the completeness relation  $\sum_x |x\rangle\langle x| = I$  between each term in (4.86), obtaining

$$\langle y|U_{p(n)} \cdots U_2 U_1|0\rangle = \sum_{x_1, \dots, x_{p(n)-1}} \langle y|U_{p(n)}|x_{p(n)-1}\rangle \langle x_{p(n)-1}|U_{p(n)-1} \cdots U_2|x_1\rangle \langle x_1|U_1|0\rangle. \quad (4.87)$$

Given that the individual unitary gates appearing in this sum are operations such as the Hadamard gate, CNOT, and so on, it is clear that each term in the sum can be calculated to high accuracy using only polynomial space on a classical computer, and thus the sum as a whole can be calculated using polynomial space, since individual terms in the sum can be erased after being added to the running total. Of course, this algorithm is rather slow, since there are exponentially many terms in the sum which need to be calculated and added to the total; however, only polynomially much space is consumed, and thus  $\mathbf{BQP} \subseteq \mathbf{PSPACE}$ , as we set out to show.

A similar procedure can be used to simulate an *arbitrary* quantum computation on a classical computer, no matter the length of the quantum computation. Therefore, the class of problems solvable on a quantum computer with unlimited time and space resources is no larger than the class of problems solvable on a classical computer. Stated another way, this means that quantum computers do not violate the Church–Turing thesis; any algorithmic process can be simulated using a Turing machine. Of course, quantum computers may be much more *efficient* than their classical counterparts, thereby challenging the *strong* Church–Turing thesis that any algorithmic process can be simulated *efficiently* using a probabilistic Turing machine.

## 4.6 Summary of the quantum circuit model of computation

In this book the term ‘quantum computer’ is synonymous with the quantum circuit model of computation. This chapter has provided a detailed look at quantum circuits, their basic elements, universal families of gates, and some applications. Before we move on to more sophisticated applications, let us summarize the key elements of the quantum circuit model of computation:

- (1) **Classical resources:** A quantum computer consists of two parts, a classical part and a quantum part. In principle, there is no need for the classical part of the computer, but in practice certain tasks may be made much easier if parts of the computation can be done classically. For example, many schemes for quantum error-correction (Chapter 10) are likely to involve classical computations in order to maximize efficiency. While classical computations can always be done, in principle, on a quantum computer, it may be more convenient to perform the calculations on a classical computer.
- (2) **A suitable state space:** A quantum circuit operates on some number,  $n$ , of qubits. The state space is thus a  $2^n$ -dimensional complex Hilbert space. Product states of the form  $|x_1, \dots, x_n\rangle$ , where  $x_i = 0, 1$ , are known as *computational basis states* of the computer.  $|x\rangle$  denotes a computational basis state, where  $x$  is the number whose binary representation is  $x_1 \dots x_n$ .
- (3) **Ability to prepare states in the computational basis:** It is assumed that any computational basis state  $|x_1, \dots, x_n\rangle$  can be prepared in at most  $n$  steps.



- (4) **Ability to perform quantum gates:** Gates can be applied to any subset of qubits as desired, and a universal family of gates can be implemented. For example, it should be possible to apply the CNOT gate to any pair of qubits in the quantum computer. The Hadamard, phase, CNOT and  $\pi/8$  gates form a family of gates from which any unitary operation can be approximated, and thus is a universal set of gates. Other universal families exist.
- (5) **Ability to perform measurements in the computational basis:** Measurements may be performed in the computational basis of one or more of the qubits in the computer.

The quantum circuit model of quantum computation is equivalent to many other models of computation which have been proposed, in the sense that other models result in essentially the same resource requirements for the same problems. As a simple example which illustrates the basic idea, one might wonder whether moving to a design based on three-level quantum systems, rather than the two-level qubits, would confer any computational advantage. Of course, although there may be some slight advantage in using three-level quantum systems (*qutrits*) over two-level systems, any difference will be essentially negligible from the theoretical point of view. At a less trivial level, the ‘quantum Turing machine’ model of computation, a quantum generalization of the classical Turing machine model, has been shown to be equivalent to the model based upon quantum circuits. We do not consider that model of computation in this book, but the reader interested in learning more about quantum Turing machines may consult the references given in the end of chapter ‘History and further reading’.

Despite the simplicity and attraction of the quantum circuit model, it is useful to keep in mind possible criticisms, modifications, and extensions. For example, it is by no means clear that the basic assumptions underlying the state space and starting conditions in the quantum circuit model are justified. Everything is phrased in terms of finite dimensional state spaces. Might there be anything to be gained by using systems whose state space is infinite dimensional? Assuming that the starting state of the computer is a computational basis state is also not necessary; we know that many systems in Nature ‘prefer’ to sit in highly entangled states of many systems; might it be possible to exploit this preference to obtain extra computational power? It might be that having access to certain states allows particular computations to be done much more easily than if we are constrained to start in the computational basis. Likewise, the ability to efficiently perform entangling measurements in multi-qubit bases might be as useful as being able to perform just entangling unitary operations. Indeed, it may be possible to harness such measurements to perform tasks intractable within the quantum circuit model.

A detailed examination and attempted justification of the physics underlying the quantum circuit model is outside the scope of the present discussion, and, indeed, outside the scope of present knowledge! By raising these issues we wish to introduce the question of the completeness of the quantum circuit model, and re-emphasize the fundamental point that information is physical. In our attempts to formulate models for information processing we should always attempt to go back to fundamental physical laws. For the purposes of this book, we shall stay within the quantum circuit model of computation. It offers a rich and powerful model of computation that exploits the properties of quantum mechanics to perform amazing feats of information processing, without classical prece-

dent. Whether physically reasonable models of computation exist which go beyond the quantum circuit model is a fascinating question which we leave open for you.

## 4.7 Simulation of quantum systems

*Perhaps [...] we need a mathematical theory of quantum automata. [...] the quantum state space has far greater capacity than the classical one: for a classical system with  $N$  states, its quantum version allowing superposition accommodates  $c^N$  states. When we join two classical systems, their number of states  $N_1$  and  $N_2$  are multiplied, and in the quantum case we get the exponential growth  $c^{N_1 N_2}$ . [...] These crude estimates show that the quantum behavior of the system might be much more complex than its classical simulation.*

– Yu Manin (1980)<sup>[Man80]</sup>, as translated in [Man99]

*The quantum-mechanical computation of one molecule of methane requires  $10^{42}$  grid points. Assuming that at each point we have to perform only 10 elementary operations, and that the computation is performed at the extremely low temperature  $T = 3 \times 10^{-3} K$ , we would still have to use all the energy produced on Earth during the last century.*

– R. P. Poplavskii (1975)<sup>[Pop75]</sup>, as quoted by Manin

*Can physics be simulated by a universal computer? [...] the physical world is quantum mechanical, and therefore the proper problem is the simulation of quantum physics [...] the full description of quantum mechanics for a large system with  $R$  particles [...] has too many variables, it cannot be simulated with a normal computer with a number of elements proportional to  $R$  [ ... but it can be simulated with ] quantum computer elements. [...] Can a quantum system be probabilistically simulated by a classical (probabilistic, I'd assume) universal computer? [...] If you take the computer to be the classical kind I've described so far [...] the answer is certainly, No!*

– Richard P. Feynman (1982)<sup>[Fey82]</sup>

Let us close out this chapter by providing an interesting and useful application of the quantum circuit model. One of the most important practical applications of computation is the simulation of physical systems. For example, in the engineering design of a new building, finite element analysis and modeling is used to ensure safety while minimizing cost. Cars are made lightweight, structurally sound, attractive, and inexpensive, by using computer aided design. Modern aeronautical engineering depends heavily on computational fluid dynamics simulations for aircraft designs. Nuclear weapons are no longer exploded (for the most part), but rather, tested by exhaustive computational modeling. Examples abound, because of the tremendous practical applications of predictive simulations. We begin by describing some instances of the simulation problem, then we present a quantum algorithm for simulation and an illustrative example, concluding with some perspective on this application.

### 4.7.1 Simulation in action

The heart of simulation is the solution of differential equations which capture the physical laws governing the dynamical behavior of a system. Some examples include Newton's

law,

$$\frac{d}{dx} \left( m \frac{dx}{dt} \right) = F, \quad (4.88)$$

Poisson's equation,

$$-\vec{\nabla} \cdot (k \vec{\nabla} u) = \vec{Q}, \quad (4.89)$$

the electromagnetic vector wave equation,

$$\vec{\nabla} \cdot \vec{\nabla} \vec{E} = \epsilon_0 \mu_0 \frac{\partial^2 \vec{E}}{\partial t^2}, \quad (4.90)$$

and the diffusion equation,

$$\vec{\nabla}^2 \psi = \frac{1}{a^2} \frac{\partial \psi}{\partial t}, \quad (4.91)$$

just to name a very few. The goal is generally: given an initial state of the system, what is the state at some other time and/or position? Solutions are usually obtained by *approximating* the state with a digital representation, then *discretizing* the differential equation in space and time such that an iterative application of a procedure carries the state from the initial to the final conditions. Importantly, the error in this procedure is bounded, and known not to grow faster than some small power of the number of iterations. Furthermore, *not* all dynamical systems can be simulated *efficiently*: generally, only those systems which can be described efficiently can be simulated efficiently.

Simulation of quantum systems by classical computers is possible, but generally only very inefficiently. The dynamical behavior of many simple quantum systems is governed by Schrödinger's equation,

$$i\hbar \frac{d}{dt} |\psi\rangle = H |\psi\rangle. \quad (4.92)$$

We will find it convenient to absorb  $\hbar$  into  $H$ , and use this convention for the rest of this section. For a typical Hamiltonian of interest to physicists dealing with real particles in space (rather than abstract systems such as qubits, which we have been dealing with!), this reduces to

$$i \frac{\partial}{\partial t} \psi(x) = \left[ -\frac{1}{2m} \frac{\partial^2}{\partial x^2} + V(x) \right] \psi(x), \quad (4.93)$$

using a convention known as the position representation  $\langle x | \psi \rangle = \psi(x)$ . This is an elliptical equation very much like Equation (4.91). So just simulating Schrödinger's equation is not the especial difficulty faced in simulating quantum systems. What is the difficulty?

The key challenge in simulating quantum systems is the *exponential* number of differential equations which must be solved. For one qubit evolving according to the Schrödinger equation, a system of two differential equations must be solved; for two qubits, four equations; and for  $n$  qubits,  $2^n$  equations. Sometimes, insightful approximations can be made which reduce the effective number of equations involved, thus making classical simulation of the quantum system feasible. However, there are many physically interesting quantum systems for which no such approximations are known.

**Exercise 4.46: (Exponential complexity growth of quantum systems)** Let  $\rho$  be a density matrix describing the state of  $n$  qubits. Show that describing  $\rho$  requires  $4^n - 1$  independent real numbers.

The reader with a physics background may appreciate that there are many important quantum systems for which classical simulation is intractable. These include the Hubbard model, a model of interacting fermionic particles with the Hamiltonian

$$H = \sum_{k=1}^n V_0 n_{k\uparrow} n_{k\downarrow} + \sum_{k,j \text{ neighbors}, \sigma} t_0 c_{k\sigma}^* c_{j\sigma} , \quad (4.94)$$

which is useful in the study of superconductivity and magnetism, the Ising model,

$$H = \sum_{k=1}^n \vec{\sigma}_k \cdot \vec{\sigma}_{k+1} , \quad (4.95)$$

and many others. Solutions to such models give many physical properties such as the dielectric constant, conductivity, and magnetic susceptibility of materials. More sophisticated models such as quantum electrodynamics (QED) and quantum chromodynamics (QCD) can be used to compute constants such as the mass of the proton.

Quantum computers can efficiently simulate quantum systems for which there is no known efficient classical simulation. Intuitively, this is possible for much the same reason any quantum circuit can be constructed from a universal set of quantum gates. Moreover, just as there exist unitary operations which cannot be *efficiently* approximated, it is possible in principle to imagine quantum systems with Hamiltonians which cannot be efficiently simulated on a quantum computer. Of course, we believe that such systems aren't actually realized in Nature, otherwise we'd be able to exploit them to do information processing beyond the quantum circuit model.

#### 4.7.2 The quantum simulation algorithm

Classical simulation begins with the realization that in solving a simple differential equation such as  $dy/dt = f(y)$ , to first order, it is known that  $y(t + \Delta t) \approx y(t) + f(y)\Delta t$ . Similarly, the quantum case is concerned with the solution of  $id|\psi\rangle/dt = H|\psi\rangle$ , which, for a time-independent  $H$ , is just

$$|\psi(t)\rangle = e^{-iHt}|\psi(0)\rangle . \quad (4.96)$$

Since  $H$  is usually extremely difficult to exponentiate (it may be sparse, but it is also exponentially large), a good beginning is the first order solution  $|\psi(t + \Delta t)\rangle \approx (I - iH\Delta t)|\psi(t)\rangle$ . This is tractable, because for many Hamiltonians  $H$  it is straightforward to compose quantum gates to efficiently approximate  $I - iH\Delta t$ . However, such first order solutions are generally not very satisfactory.

*Efficient approximation of the solution to Equation (4.96), to high order, is possible for many classes of Hamiltonian. For example, in most physical systems, the Hamiltonian can be written as a sum over many local interactions. Specifically, for a system of  $n$  particles,*

$$H = \sum_{k=1}^L H_k , \quad (4.97)$$

where each  $H_k$  acts on at most a constant  $c$  number of systems, and  $L$  is a polynomial in  $n$ . For example, the terms  $H_k$  are often just two-body interactions such as  $X_i X_j$  and one-body Hamiltonians such as  $X_i$ . Both the Hubbard and Ising models have Hamiltonians of this form. Such locality is quite physically reasonable, and originates in many systems

from the fact that most interactions fall off with increasing distance or difference in energy. There are sometimes additional global symmetry constraints such as particle statistics; we shall come to those shortly. The important point is that although  $e^{-iHt}$  is difficult to compute,  $e^{-iH_k t}$  acts on a much smaller subsystem, and is straightforward to approximate using quantum circuits. But because  $[H_j, H_k] \neq 0$  in general,  $e^{-iHt} \neq \prod_k e^{-iH_k t}$ . How, then, can  $e^{-iH_k t}$  be useful in constructing  $e^{-iHt}$ ?

**Exercise 4.47:** For  $H = \sum_k^L H_k$ , prove that  $e^{-iHt} = e^{-iH_1 t} e^{-iH_2 t} \dots e^{-iH_L t}$  for all  $t$  if  $[H_j, H_k] = 0$ , for all  $j, k$ .

**Exercise 4.48:** Show that the restriction of  $H_k$  to involve at most  $c$  particles *implies* that in the sum (4.97),  $L$  is upper bounded by a polynomial in  $n$ .

The heart of quantum simulation algorithms is the following asymptotic approximation theorem:

**Theorem 4.3: (Trotter formula)** Let  $A$  and  $B$  be Hermitian operators. Then for any real  $t$ ,

$$\lim_{n \rightarrow \infty} (e^{iAt/n} e^{iBt/n})^n = e^{i(A+B)t}. \quad (4.98)$$

Note that (4.98) is true even if  $A$  and  $B$  do not commute. Even more interestingly, perhaps, it can be generalized to hold for  $A$  and  $B$  which are generators of certain kinds of semigroups, which correspond to general quantum operations; we shall describe such generators (the ‘Lindblad form’) in Section 8.4.1 of Chapter 8. For now, we only consider the case of  $A$  and  $B$  being Hermitian matrices.

*Proof*

By definition,

$$e^{iAt/n} = I + \frac{1}{n} iAt + O\left(\frac{1}{n^2}\right), \quad (4.99)$$

and thus

$$e^{iAt/n} e^{iBt/n} = I + \frac{1}{n} i(A+B)t + O\left(\frac{1}{n^2}\right). \quad (4.100)$$

Taking products of these gives us

$$(e^{iAt/n} e^{iBt/n})^n = I + \sum_{k=1}^n \binom{n}{k} \frac{1}{n^k} [i(A+B)t]^k + O\left(\frac{1}{n}\right), \quad (4.101)$$

and since  $\binom{n}{k} \frac{1}{n^k} = \left(1 + O\left(\frac{1}{n}\right)\right) / k!$ , this gives

$$\lim_{n \rightarrow \infty} (e^{iAt/n} e^{iBt/n})^n = \lim_{n \rightarrow \infty} \sum_{k=0}^n \frac{(i(A+B)t)^k}{k!} \left(1 + O\left(\frac{1}{n}\right)\right) + O\left(\frac{1}{n}\right) = e^{i(A+B)t}. \quad (4.102)$$

□

Modifications of the Trotter formula provide the methods by which higher order

approximations can be derived for performing quantum simulations. For example, using similar reasoning to the proof above, it can be shown that

$$e^{i(A+B)\Delta t} = e^{iA\Delta t} e^{iB\Delta t} + O(\Delta t^2). \quad (4.103)$$

Similarly,

$$e^{i(A+B)\Delta t} = e^{iA\Delta t/2} e^{iB\Delta t} e^{iA\Delta t/2} + O(\Delta t^3). \quad (4.104)$$

An overview of the quantum simulation algorithm is given below, and an explicit example of simulating the one-dimensional non-relativistic Schrödinger equation is shown in Box 4.2.

### Algorithm: Quantum simulation

**Inputs:** (1) A Hamiltonian  $H = \sum_k H_k$  acting on an  $N$ -dimensional system, where each  $H_k$  acts on a small subsystem of size independent of  $N$ , (2) an initial state  $|\psi_0\rangle$ , of the system at  $t = 0$ , (3) a positive, non-zero accuracy  $\delta$ , and (3) a time  $t_f$  at which the evolved state is desired.

**Outputs:** A state  $|\tilde{\psi}(t_f)\rangle$  such that  $|\langle\tilde{\psi}(t_f)|e^{-iHt_f}|\psi_0\rangle|^2 \geq 1 - \delta$ .

**Runtime:**  $O(\text{poly}(1/\delta))$  operations.

**Procedure:** Choose a representation such that the state  $|\tilde{\psi}\rangle$  of  $n = \text{poly}(\log N)$  qubits approximates the system and the operators  $e^{-iH_k\Delta t}$  have efficient quantum circuit approximations. Select an approximation method (see for example Equations (4.103)–(4.105)) and  $\Delta t$  such that the expected error is acceptable (and  $j\Delta t = t_f$  for an integer  $j$ ), construct the corresponding quantum circuit  $U_{\Delta t}$  for the iterative step, and do:

- |    |  |                  |
|----|--|------------------|
| 1. | $ \tilde{\psi}_0\rangle \leftarrow  \psi_0\rangle ; j = 0$                     | initialize state |
| 2. | $\rightarrow  \tilde{\psi}_{j+1}\rangle = U_{\Delta t}  \tilde{\psi}_j\rangle$ | iterative update |
| 3. | $\rightarrow j = j + 1 ; \text{ goto 2 until } j\Delta t \geq t_f$             | loop             |
| 4. | $\rightarrow  \tilde{\psi}(t_f)\rangle =  \tilde{\psi}_j\rangle$               | final result     |

**Exercise 4.49: (Baker–Campbell–Hausdorff formula)** Prove that

$$e^{(A+B)\Delta t} = e^{A\Delta t} e^{B\Delta t} e^{-\frac{1}{2}[A,B]\Delta t^2} + O(\Delta t^3), \quad (4.105)$$

and also prove Equations (4.103) and (4.104).

**Exercise 4.50:** Let  $H = \sum_k^L H_k$ , and define

$$U_{\Delta t} = \left[ e^{-iH_1\Delta t} e^{-iH_2\Delta t} \dots e^{-iH_L\Delta t} \right] \left[ e^{-iH_L\Delta t} e^{-iH_{L-1}\Delta t} \dots e^{-iH_1\Delta t} \right]. \quad (4.106)$$

(a) Prove that  $U_{\Delta t} = e^{-2iH\Delta t} + O(\Delta t^3)$ .

(b) Use the results in Box 4.1 to prove that for a positive integer  $m$ ,

$$E(U_{\Delta t}^m, e^{-2miH\Delta t}) \leq m\alpha\Delta t^3, \quad (4.107)$$

for some constant  $\alpha$ .

### Box 4.2: Quantum simulation of Schrödinger's equation

The methods and limitations of quantum simulation may be illustrated by the following example, drawn from the conventional models studied by physicists, rather than the abstract qubit model. Consider a single particle living on a line, in a one-dimensional potential  $V(x)$ , governed by the Hamiltonian

$$H = \frac{p^2}{2m} + V(x), \quad (4.108)$$

where  $p$  is the momentum operator and  $x$  is the position operator. The eigenvalues of  $x$  are continuous, and the system state  $|\psi\rangle$  resides in an infinite dimensional Hilbert space; in the  $x$  basis, it can be written as

$$|\psi\rangle = \int_{-\infty}^{\infty} |x\rangle \langle x|\psi\rangle dx. \quad (4.109)$$

In practice, only some finite region is of interest, which we may take to be the range  $-d \leq x \leq d$ . Furthermore, it is possible to choose a differential step size  $\Delta x$  sufficiently small compared to the shortest wavelength in the system such that

$$|\tilde{\psi}\rangle = \sum_{k=-d/\Delta x}^{d/\Delta x} a_k |k\Delta x\rangle \quad (4.110)$$

provides a good physical approximation of  $|\psi\rangle$ . This state can be represented using  $n = \lceil \log(2d/\Delta x + 1) \rceil$  qubits; we simply replace the basis  $|k\Delta x\rangle$  (an eigenstate of the  $x$  operator) with  $|k\rangle$ , a computational basis state of  $n$  qubits. Note that only  $n$  qubits are required for this simulation, whereas classically  $2^n$  complex numbers would have to be kept track of, thus leading to an exponential resource saving when performing the simulation on a quantum computer.

Computation of  $|\tilde{\psi}(t)\rangle = e^{-iHt}|\tilde{\psi}(0)\rangle$  must utilize one of the approximations of Equations (4.103)–(4.105) because in general  $H_1 = V(x)$  does not commute with  $H_0 = p^2/2m$ . Thus, we must be able to compute  $e^{-iH_1\Delta t}$  and  $e^{-iH_0\Delta t}$ . Because  $|\tilde{\psi}\rangle$  is expressed in the eigenbasis of  $H_1$ ,  $e^{-iH_1\Delta t}$  is a diagonal transformation of the form

$$|k\rangle \rightarrow e^{-iV(k\Delta x)\Delta t} |k\rangle. \quad (4.111)$$

It is straightforward to compute this, since we can compute  $V(k\Delta x)\Delta t$ . (See also Problem 4.1.) The second term is also simple, because  $x$  and  $p$  are conjugate variables related by a quantum Fourier transform  $U_{\text{FFT}} x U_{\text{FFT}}^\dagger = p$ , and thus  $e^{-iH_0\Delta t} = U_{\text{FFT}} e^{-ix^2\Delta t/2m} U_{\text{FFT}}^\dagger$ ; to compute  $e^{-iH_0\Delta t}$ , do

$$|k\rangle \rightarrow U_{\text{FFT}} e^{-ix^2/2m} U_{\text{FFT}}^\dagger |k\rangle. \quad (4.112)$$

The construction of  $U_{\text{FFT}}$  is discussed in Chapter 5.

### 4.7.3 An illustrative example

The procedure we have described for quantum simulations has concentrated on simulating Hamiltonians which are sums of local interactions. However, this is not a fundamental

requirement! As the following example illustrates, efficient quantum simulations are possible even for Hamiltonians which act non-trivially on all or nearly all parts of a large system.

Suppose we have the Hamiltonian

$$H = Z_1 \otimes Z_2 \otimes \cdots \otimes Z_n, \quad (4.113)$$

which acts on an  $n$  qubit system. Despite this being an interaction involving all of the system, indeed, it can be simulated efficiently. What we desire is a simple quantum circuit which implements  $e^{-iH\Delta t}$ , for arbitrary values of  $\Delta t$ . A circuit doing precisely this, for  $n = 3$ , is shown in Figure 4.19. The main insight is that although the Hamiltonian involves all the qubits in the system, it does so in a *classical* manner: the phase shift applied to the system is  $e^{-i\Delta t}$  if the *parity* of the  $n$  qubits in the computational basis is even; otherwise, the phase shift should be  $e^{i\Delta t}$ . Thus, simple simulation of  $H$  is possible by first classically computing the parity (storing the result in an ancilla qubit), then applying the appropriate phase shift conditioned on the parity, then uncomputing the parity (to erase the ancilla). This strategy clearly works not only for  $n = 3$ , but also for arbitrary values of  $n$ .

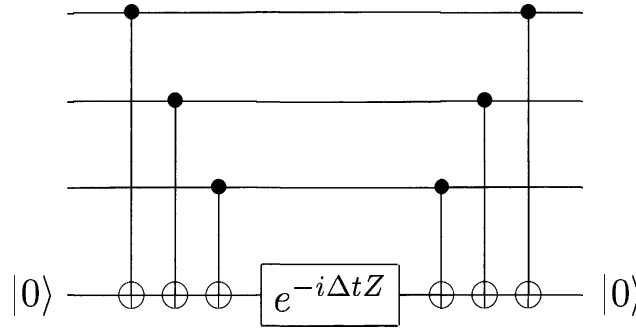


Figure 4.19. Quantum circuit for simulating the Hamiltonian  $H = Z_1 \otimes Z_2 \otimes Z_3$  for time  $\Delta t$ .

Furthermore, extending the same procedure allows us to simulate more complicated extended Hamiltonians. Specifically, we can efficiently simulate any Hamiltonian of the form

$$H = \bigotimes_{k=1}^n \sigma_{c(k)}^k, \quad (4.114)$$

where  $\sigma_{c(k)}^k$  is a Pauli matrix (or the identity) acting on the  $k$ th qubit, with  $c(k) \in \{0, 1, 2, 3\}$  specifying one of  $\{I, X, Y, Z\}$ . The qubits upon which the identity operation is performed can be disregarded, and  $X$  or  $Y$  terms can be transformed by single qubit gates to  $Z$  operations. This leaves us with a Hamiltonian of the form of (4.113), which is simulated as described above.

**Exercise 4.51:** Construct a quantum circuit to simulate the Hamiltonian

$$H = X_1 \otimes Y_2 \otimes Z_3, \quad (4.115)$$

performing the unitary transform  $e^{-i\Delta t H}$  for any  $\Delta t$ .

Using this procedure allows us to simulate a wide class of Hamiltonians containing terms which are not local. In particular, it is possible to simulate a Hamiltonian of the form



$H = \sum_{k=1}^L H_k$  where the only restriction is that the individual  $H_k$  have a tensor product structure, and that  $L$  is polynomial in the total number of particles  $n$ . More generally, all that is required is that there be an efficient circuit to simulate each  $H_k$  separately. As an example, the Hamiltonian  $H = \sum_{k=1}^n X_k + Z^{\otimes n}$  can easily be simulated using the above techniques. Such Hamiltonians typically do not arise in Nature. However, they provide a new and possibly valuable vista on the world of quantum algorithms.

#### 4.7.4 Perspectives on quantum simulation

The quantum simulation algorithm is very similar to classical methods, but also differs in a fundamental way. Each iteration of the quantum algorithm must completely replace the old state with the new one; there is no way to obtain (non-trivial) information from an intermediate step without significantly changing the algorithm, because the state is a quantum one. Furthermore, the final measurement must be chosen cleverly to provide the desired result, because it disturbs the quantum state. Of course, the quantum simulation can be repeated to obtain statistics, but it is desirable to repeat the algorithm only at most a polynomial number of times. It may be that even though the simulation can be performed efficiently, there is no way to efficiently perform a desired measurement.

Also, there are Hamiltonians which simply can't be simulated efficiently. In Section 4.5.4, we saw that there exist unitary transformations which quantum computers cannot efficiently approximate. As a corollary, not all Hamiltonian evolutions can be efficiently simulated on a quantum computer, for if this were possible, then all unitary transformations could be efficiently approximated!

Another difficult problem – one which is very interesting – is the simulation of equilibration processes. A system with Hamiltonian  $H$  in contact with an environment at temperature  $T$  will generally come to thermal equilibrium in a state known as the *Gibbs* state,  $\rho_{\text{therm}} = e^{-H/k_B T} / \mathcal{Z}$ , where  $k_B$  is Boltzmann's constant, and  $\mathcal{Z} = \text{tr } e^{-H/k_B T}$  is the usual partition function normalization, which ensures that  $\text{tr}(\rho) = 1$ . The process by which this equilibration occurs is not very well understood, although certain requirements are known: the environment must be large, it must have non-zero population in states with energies matching the eigenstates of  $H$ , and its coupling with the system should be weak. Obtaining  $\rho_{\text{therm}}$  for arbitrary  $H$  and  $T$  is generally an exponentially difficult problem for a classical computer. Might a quantum computer be able to solve this efficiently? We do not yet know.

On the other hand, as we discussed above many interesting quantum problems can indeed be simulated efficiently with a quantum computer, even when they have extra constraints beyond the simple algorithms presented here. A particular class of these involve global symmetries originating from particle statistics. In the everyday world, we are used to being able to identify different particles; tennis balls can be followed around a tennis court, keeping track of which is which. This ability to keep track of which object is which is a general feature of classical objects – by continuously measuring the position of a classical particle it can be tracked at all times, and thus uniquely distinguished from other particles. However, this breaks down in quantum mechanics, which prevents us from following the motion of individual particles exactly. If the two particles are inherently different, say a proton and an electron, then we can distinguish them by measuring the sign of the charge to tell which particle is which. But in the case of identical particles, like two electrons, it is found that they are truly indistinguishable.

Indistinguishability of particles places a constraint on the state vector of a system which

manifests itself in two ways. Experimentally, particles in Nature are found to come in two distinct flavors, known as bosons and fermions. The state vector of a system of bosons remains unchanged under permutation of any two constituents, reflecting their fundamental indistinguishability. Systems of fermions, in contrast, experience a sign change in their state vector under interchange of any two constituents. Both kinds of systems can be simulated efficiently on a quantum computer. The detailed description of how this is done is outside the scope of this book; suffice it to say the procedure is fairly straightforward. Given an initial state of the wrong symmetry, it can be properly symmetrized before the simulation begins. And the operators used in the simulation can be constructed to respect the desired symmetry, even allowing for the effects of higher order error terms. The reader who is interested in pursuing this and other topics further will find pointers to the literature in ‘History and further reading,’ at the end of the chapter.

**Problem 4.1: (Computable phase shifts)** Let  $m$  and  $n$  be positive integers.

Suppose  $f : \{0, \dots, 2^m - 1\} \rightarrow \{0, \dots, 2^n - 1\}$  is a classical function from  $m$  to  $n$  bits which may be computed reversibly using  $T$  Toffoli gates, as described in Section 3.2.5. That is, the function  $(x, y) \rightarrow (x, y \oplus f(x))$  may be implemented using  $T$  Toffoli gates. Give a quantum circuit using  $2T + n$  (or fewer) one, two, and three qubit gates to implement the unitary operation defined by

$$|x\rangle \rightarrow \exp\left(\frac{-2i\pi f(x)}{2^n}\right) |x\rangle. \quad (4.116)$$

**Problem 4.2:** Find a depth  $O(\log n)$  construction for the  $C^n(X)$  gate. (*Comment:*

The depth of a circuit is the number of distinct timesteps at which gates are applied; the point of this problem is that it is possible to parallelize the  $C^n(X)$  construction by applying many gates in parallel during the same timestep.)

**Problem 4.3: (Alternate universality construction)** Suppose  $U$  is a unitary matrix on  $n$  qubits. Define  $H \equiv i \ln(U)$ . Show that

- (1)  $H$  is Hermitian, with eigenvalues in the range 0 to  $2\pi$ .
- (2)  $H$  can be written

$$H = \sum_g h_g g, \quad (4.117)$$

where  $h_g$  are real numbers and the sum is over all  $n$ -fold tensor products  $g$  of the Pauli matrices  $\{I, X, Y, Z\}$ .

- (3) Let  $\Delta = 1/k$ , for some positive integer  $k$ . Explain how the unitary operation  $\exp(-ih_g g \Delta)$  may be implemented using  $O(n)$  one and two qubit operations.
- (4) Show that

$$\exp(-iH\Delta) = \prod_g \exp(-ih_g g \Delta) + O(4^n \Delta^2), \quad (4.118)$$

where the product is taken with respect to any fixed ordering of the  $n$ -fold tensor products of Pauli matrices,  $g$ .

(5) Show that

$$U = \left[ \prod_g \exp(-ih_g g \Delta) \right]^k + O(4^n \Delta). \quad (4.119)$$

(6) Explain how to approximate  $U$  to within a distance  $\epsilon > 0$  using  $O(n16^n/\epsilon)$  one and two qubit unitary operations.

**Problem 4.4: (Minimal Toffoli construction) (Research)**

- (1) What is the smallest number of two qubit gates that can be used to implement the Toffoli gate?
- (2) What is the smallest number of one qubit gates and CNOT gates that can be used to implement the Toffoli gate?
- (3) What is the smallest number of one qubit gates and controlled- $Z$  gates that can be used to implement the Toffoli gate?

**Problem 4.5: (Research)** Construct a family of Hamiltonians,  $\{H_n\}$ , on  $n$  qubits, such that simulating  $H_n$  requires a number of operations super-polynomial in  $n$ . (*Comment:* This problem seems to be quite difficult.)

**Problem 4.6: (Universality with prior entanglement)** Controlled-NOT gates and single qubit gates form a universal set of quantum logic gates. Show that an alternative universal set of resources is comprised of single qubit unitaries, the ability to perform measurements of pairs of qubits in the Bell basis, and the ability to prepare arbitrary four qubit entangled states.

### Summary of Chapter 4: Quantum circuits

- **Universality:** Any unitary operation on  $n$  qubits may be implemented exactly by composing single qubit and controlled-NOT gates.
- **Universality with a discrete set:** The Hadamard gate, phase gate, controlled-NOT gate, and  $\pi/8$  gate are *universal* for quantum computation, in the sense that an arbitrary unitary operation on  $n$  qubits can be approximated to an arbitrary accuracy  $\epsilon > 0$  using a circuit composed of only these gates. Replacing the  $\pi/8$  gate in this list with the Toffoli gate also gives a universal family.
- **Not all unitary operations can be efficiently implemented:** There are unitary operations on  $n$  qubits which require  $\Omega(2^n \log(1/\epsilon)/\log(n))$  gates to approximate to within a distance  $\epsilon$  using any finite set of gates.
- **Simulation:** For a Hamiltonian  $H = \sum_k H_k$  which is a sum of polynomially many terms  $H_k$  such that efficient quantum circuits for  $H_k$  can be constructed, a quantum computer can efficiently simulate the evolution  $e^{-iHt}$  and approximate  $|\psi(t)\rangle = e^{-iHt}|\psi(0)\rangle$ , given  $|\psi(0)\rangle$ .

## History and further reading

The gate constructions in this chapter are drawn from a wide variety of sources. The paper by Barenco, Bennett, Cleve, DiVincenzo, Margolus, Shor, Sleator, Smolin, and Weinfurter<sup>[BBC<sup>+</sup>95]</sup> was the source of many of the circuit constructions in this chapter, and for the universality proof for single qubit and controlled-NOT gates. Another useful source of insights about quantum circuits is the paper by Beckman, Chari, Devabhaktuni, and Preskill<sup>[BCDP96]</sup>. A gentle and accessible introduction has been provided by DiVincenzo<sup>[DiV98]</sup>. The fact that measurements commute with control qubit terminals was pointed out by Griffiths and Niu<sup>[GN96]</sup>.

The universality proof for two-level unitaries is due to Reck, Zeilinger, Bernstein, and Bertani<sup>[RZBB94]</sup>. The universality of the controlled-NOT and single qubit gates was proved by DiVincenzo<sup>[DiV95b]</sup>. The universal gate  $G$  in Exercise 4.44 is sometimes known as the Deutsch gate<sup>[Deu89]</sup>. Deutsch, Barenco, and Ekert<sup>[DBE95]</sup> and Lloyd<sup>[Llo95]</sup> independently proved that almost any two qubit quantum logic gate is universal. That errors caused by sequences of gates is at most the sum of the errors of the individual gates was proven by Bernstein and Vazirani<sup>[BV97]</sup>. The specific universal set of gates we have focused on – the Hadamard, phase, controlled-NOT and  $\pi/8$  gates, was proved universal in Boykin, Mor, Pulver, Roychowdhury, and Vatan<sup>[BMP<sup>+</sup>99]</sup>, which also contains a proof that  $\theta$  defined by  $\cos(\theta/2) \equiv \cos^2(\pi/8)$  is an irrational multiple of  $\pi$ . The bound in Section 4.5.4 is based on a paper by Knill<sup>[Kni95]</sup>, which does a much more detailed investigation of the hardness of approximating arbitrary unitary operations using quantum circuits. In particular, Knill obtains tighter and more general bounds than we do, and his analysis applies also to cases where the universal set is a continuum of gates, not just a finite set, as we have considered.

The quantum circuit model of computation is due to Deutsch<sup>[Deu89]</sup>, and was further developed by Yao<sup>[Yao93]</sup>. The latter paper showed that the quantum circuit model of computation is equivalent to the quantum Turing machine model. Quantum Turing machines were introduced in 1980 by Benioff<sup>[Ben80]</sup>, further developed by Deutsch<sup>[Deu85]</sup> and Yao<sup>[Yao93]</sup>, and their modern definition given by Bernstein and Vazirani<sup>[BV97]</sup>. The latter two papers also take first steps towards setting up a theory of quantum computational complexity, analogous to classical computational complexity theory. In particular, the inclusion  $\mathbf{BQP} \subseteq \mathbf{PSPACE}$  and some slightly stronger results was proved by Bernstein and Vazirani. Knill and Laflamme<sup>[KL99]</sup> develop some fascinating connections between quantum and classical computational complexity. Other interesting work on quantum computational complexity includes the paper by Adleman, Demarrais and Huang<sup>[ADH97]</sup>, and the paper by Watrous<sup>[Wat99]</sup>. The latter paper gives intriguing evidence to suggest that quantum computers are more powerful than classical computers in the setting of ‘interactive proof systems’.

That quantum computers might simulate quantum systems more efficiently than classical computers was intimated by Manin<sup>[Man80]</sup> in 1980, and independently developed in more detail by Feynman<sup>[Fey82]</sup> in 1982. Much more detailed investigations were subsequently carried out by Abrams and Lloyd<sup>[AL97]</sup>, Boghosian and Taylor<sup>[BT97]</sup>, Sornborger and Stewart<sup>[SS99]</sup>, Wiesner<sup>[Wie96]</sup>, and Zalka<sup>[Zal98]</sup>. The Trotter formula is attributed to Trotter<sup>[Tro59]</sup>, and was also proven by Chernoff<sup>[Che68]</sup>, although the simpler form for unitary operators is much older, and goes back to the time of Sophus Lie. The third order version of the Baker–Campbell–Hausdorff formula, Equation (4.104), was given by Sornborger and Stewart<sup>[SS99]</sup>. Abrams and Lloyd<sup>[AL97]</sup> give a procedure for simulating

many-body Fermi systems on a quantum computer. Terhal and DiVincenzo address the problem of simulating the equilibration of quantum systems to the Gibbs state<sup>[TD98]</sup>. The method used to simulate the Schrödinger equation in Box 4.2 is due to Zalka<sup>[Zal98]</sup> and Wiesner<sup>[Wie96]</sup>.

Exercise 4.25 is due to Vandersypen, and is related to work by Chau and Wilczek<sup>[CW95]</sup>. Exercise 4.45 is due to Boykin, Mor, Pulver, Roychowdhury, and Vatan<sup>[BMP<sup>+</sup>99]</sup>. Problem 4.2 is due to Gottesman. Problem 4.6 is due to Gottesman and Chuang<sup>[GC99]</sup>.