Lecture 17: Quantum Computing Basics

(Extrapolated from Chapter 13 of Quantum Optics by Fox)

Phys 522 Quantum Optics and Photonics

Rohith Chandrasekar

In this lecture, we study the basics of Quantum Computing, starting from the motivation and need for such computational methods, to the basic formulation and logic, to applications. We discuss each topic briefly below.

13.1. Introduction

As we near the limits of present-day fabrication and push the boundaries of Moore's law, there has been a monumental effort in finding alternative computation methods to keep up with our ever-increasingly complex problems. Computing tasks are categorized into two categories based on how they scale – P problems scale as a polynomial of N, while NP scale even faster as a non-polynomial, e.g. an exponential function of N. These relationships are shown in Fig 13.2 below. Current computing technologies can easily solve P complex problems, however for NP problems, just a small increase in the size of the problem requires a sizeable increase in computation power. So in order to bridge this gap, one main effort has been in realizing quantum computation, a concept initially proposed by Richard Feynman in 1982; the concept was that quantum hardware was required so that the computer's computation power would also scale at the same rate as the complexity of the problem. The next step was taken by David Deutsch, who suggested that for a quantum hardware system, the information stored would have to be encoded as quantum states, or non-classical states. It was later shown that by exploiting a quantum hardware system, NP problems could be scaled down to P problems, and therefore efficiently solvable.

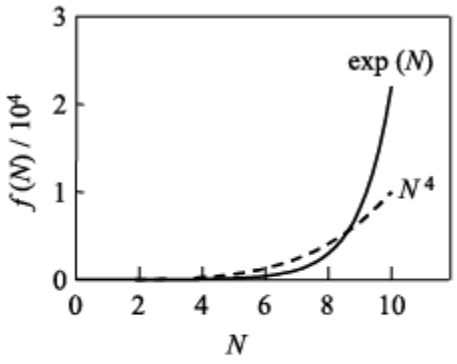


Fig. 13.2 Comparison of the size scaling of a polynomial function (N^4) with a non-polynomial function, namely $\exp(N)$.

13.2. Quantum bits (qubits)

Similar to classical computers, which store information as physical values on transistors or ferromagnetic materials, a quantum computer would store information in quantum bits, or qubits. These are quantum states of individual particles, such as atoms, photons, or nuclei. While classical states can only hold a |0> or |1> value, qubits are not limited by this requirement can take a |0> and |1> state at the same time. This follows standard quantum mechanical treatment where |0> and |1> form an orthonormal basis and a quantum state $|\varphi\rangle$ has the following equation

$$|\varphi\rangle = c_0|0\rangle + c_1|1\rangle.$$

Examples of systems considered for quantum computation are shown below in Table 13.1.

Table 13.1 Some physical realizations of qubits. In the case of the superconducting loop, the direction of the magnetic flux quantum is determined by the direction of the persistent current.

Quantum system	Physical property	$\|0\rangle$	$\|1\rangle$
Photon	Linear polarization	Horizontal	Vertical
Photon	Circular polarization	Left	Right
Nucleus	Spin	Up	Down
Electron	Spin	Up	Down
Two-level atom	Excitation state	Ground state	Excited state
Josephson junction	Electric charge	N Cooper pairs	$N+1$ Cooper pairs
Superconducting loop	Magnetic flux	Up	Down

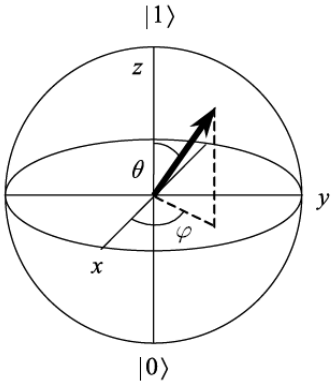


Fig. 13.3 The Bloch sphere representation of qubits. Qubit states correspond to points on the surface of the sphere, with $|0\rangle$ at the South pole, $|1\rangle$ at the North pole, and superposition states everywhere else.

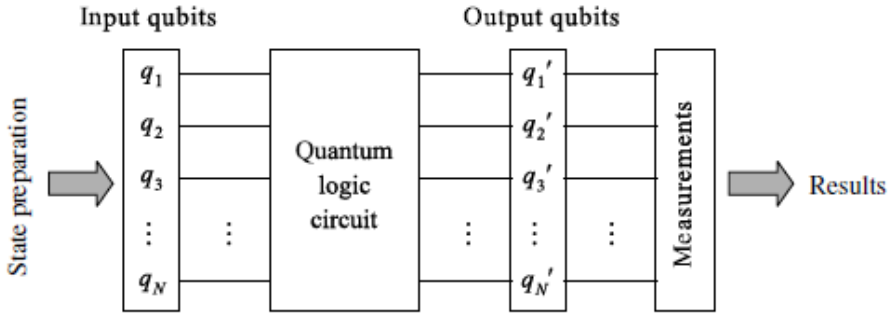
Bloch vector representation of single qubits

Since the quantum states form an orthonormal set, each state can be represented by a vector, or a Bloch vector. The vector is depicted in a Bloch sphere, as shown in Fig. 13.3, with its polar angles (θ,φ).

13.3. Quantum Logic and Gates

Single-qubit gates

The basest form of a processing unit is the logic gate, performing operations on one or two bits at a time. So as a foundation for quantum computation, it is necessary to formulate quantum logic gates using quantum state inputs, as shown in Fig. 13.4.



For a single-qubit logic gate, the input state $|\varphi\rangle = c_0|0\rangle + c_1|1\rangle$ is transformed to $|\varphi'\rangle = c_0'|0\rangle + c_1'|1\rangle$. So a gate can be represented by a 2x2 matrix having the following form

$$\begin{pmatrix} c_0' \\ c_1' \end{pmatrix} = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix}$$

where matrix M must be unitary, i.e. **MM†=I**. Three examples of single-qubit gates have been outlined in Table 13.4 below. The NOT (X) gate switches the amplitudes of states |0> and |1>, while the state Z flips the sign of state |1>. The Hadamard (H) gate turns basic states into a superposition of two states. These operations can also be represented as a rotation of the Bloch vector, as shown in Fig. 13.6. The X operator is equivalent to a rotation of π radians about the x-axis, as shown in Fig 13.6a.

Table 13.4 Single-qubit gates. Note that the X and Z gate matrices are identical to their respective Pauli spin matrices, which is one of the reasons why the gates are labelled 'X' and 'Z' in the first place. The other reason relates to their geometric interpretation as rotation operators about the x- and z-axes, respectively.

Quantum gate	Matrix representation
NOT (X)	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
Z	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
Hadamard (H)	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

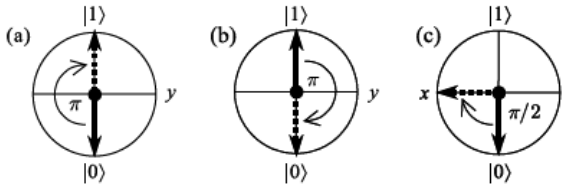


Fig. 13.6 Geometric interpretations of single-qubit operators in the Bloch sphere representation. (a) X operator on the $|0\rangle$ state observed in the y-z plane. (b) X operator on the $|1\rangle$ state observed in the y-z plane. (c) H operator on the $|0\rangle$ state observed in the x-z plane. In (c) the first rotation about the z-axis has no effect in this particular example.

Two-qubit gates

If a single-qubit gate can be represented by a 2x2 matrix, then a two-qubit gate can be represented by a 4x4 matrix. A specifically useful two-qubit gate is a controlled unitary operate gate (C-U gate). The gate has two inputs, a control qubit and a target qubit. The gate has no effect on the control qubit, but based on its value, a different operation is conducted on the target qubit. The simplest C-U gate is the controlled-NOT or C-NOT gate, as detailed below in Table 13.5 and Figure 13.7. If the control qubit is |0>, no operation is conducted on the target qubit. However, if the control qubit is |1>, then the NOT operation is conducted on the target qubit, or the operation switches the amplitudes for the |10> and |11> states.

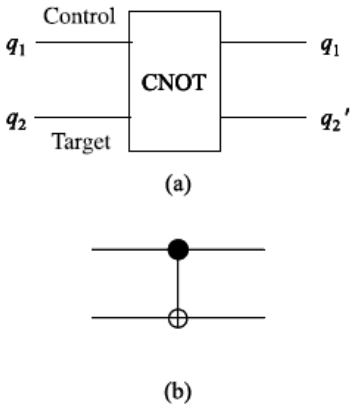


Fig. 13.7 The controlled-NOT (C-NOT) gate. (a) The gate changes the target qubit q_2 depending on the state of the control qubit q_1. (b) Symbol used to represent C-NOT gates in quantum circuits.

Table 13.5 Truth table for the controlled-NOT (C-NOT) operation.

Input qubits		Output qubits	
Control	Target	Control	Target
$\|0\rangle$	$\|0\rangle$	$\|0\rangle$	$\|0\rangle$
$\|0\rangle$	$\|1\rangle$	$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|0\rangle$	$\|1\rangle$	$\|1\rangle$
$\|1\rangle$	$\|1\rangle$	$\|1\rangle$	$\|0\rangle$

Practical Implementations

Now that we have a basic understanding of quantum logic gates, the question is how can we physically form such gates? As shown in Table 13.1, quantum states can be encoded in atoms, photons or nuclei. So implantation of a single-qubit logic gate simply requires us to irradiate atoms with short light pulses at their transition frequency with the correct pulse energy and phase to produce the required Bloch vector rotation. The rotation angle is equal to the pulse area, which is given by

$$\Theta = \left| \frac{u_{01}}{\hbar} \int_{-\infty}^{+\infty} \varepsilon_0(t) dt \right|$$

where μ_{01} is the dipole moment for the |0> to |1> transition and $\varepsilon_0(t)$ is the time-dependent electric field amplitude.

For the C-NOT gate, we have two interacting qubits, q_A and q_B, with resonant angular frequencies ω_A and ω_B. Since the two qubits interact with each other, when we excite both qubits to the |1> state, the angular frequency of the system is not $(\omega_A + \omega_B)$, but is $\omega_{AB} = \omega_A + \omega_B + \Delta$, where $\hbar\Delta$ is the interaction energy and Δ can be either positive or negative depending on whether the interaction is attractive or repulsive. If $q_A = |0\rangle$, then we can perform the NOT operation on q_B by applying a π-pulse at ω_B. However if $q_A = |1\rangle$, then the π-pulse must be shifted to $\omega_B + \Delta$. The same applies when q_B is used as the control bit. This scheme is graphically portrayed in Fig. 13.10 below.

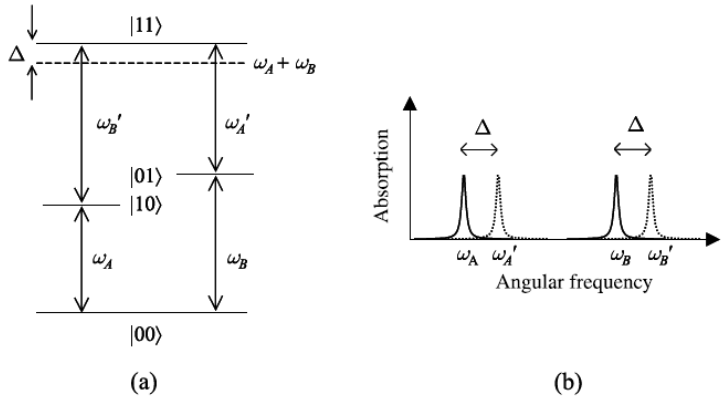


Fig. 13.10 (a) A possible level scheme for the implementation of the C-NOT gate using two qubits q_A and q_B with angular frequencies ω_A and ω_B, respectively. $\hbar\Delta$ is the interaction energy between the two qubits. (b) Absorption spectrum corresponding to the level scheme in part (a). The system only responds at angular frequency ω'_B $(\equiv \omega_B + \Delta)$ if $q_A = |1\rangle$. Similarly, the resonant frequency of q_A shifts to ω'_A if $q_B = |1\rangle$.

These gates can be implemented in real systems. For example in an **NMR system**, the qubits correspond to spin states of specified nuclei within a molecule or crystal and operations are carried out by applying RF pulses. In **ion traps**, the qubits are excitation states of a row of single ions in an ion trap. All ions are identical and hence have the same transition frequency, but can be excited separately since they are physically separated, and they interact via repulsive forces. In **cavity QED systems**, photons with opposite circular polarization states interacting with a single atom inside a resonant cavity are used as qubits. Photons have a mutual interaction with the atom, which is enhanced by the resonant cavity.

13.5. Applications of Quantum Computers

Using the logic gate theory, we can start to think how to formulate algorithms that would exploit the quantum nature of the system to efficiently compute complex problems. Here, we outline 3 algorithms, specifically Deutsch's algorithm, Grover's search algorithm, and Shor's algorithm.

Deutsch's algorithm

In order to identify a function to be balanced or constant, a classical computer requires two calls of the function, however a quantum computer can do it with just one. This is because the quantum state is a superposition of both orthonormal basis vectors, so we need only call the function once and measure only one of the output qubits.

Grover's search algorithm

A standard search problem is to find the name and address of the person with phone number 123 4567. For a classical machine, this search would require $N_{data}/_2$ operations, while a quantum machine would only require $\sqrt{N_{data}}$. So for example, to search for a number in a database with 1000000 entries, a quantum computer need only take 500x less operations than a classical one.

Shor's algorithm

Shor's algorithm concerns with Fourier transform operations. A classical computer requires $N2^N$ operations to take the Fourier transform of 2^N operations. However a quantum computer would only require N^2, hence taking an NP problem to a P problem. However, the output state would have 2^N amplitudes, which cannot be read out since we get only 1 output for each of the N qubits. So a quantum computer cannot outperform a classical computer for a basic Fourier transform, however it can identify periodicity of a set very efficiently, which is the landmark set by Shor's algorithm.

References:

[1] Fox, M. *Quantum Optics: An Introduction*. Oxford University Press: 2006.