# Study Notes:

# Quantum Computing in Slow Pace

# 量子計算慢慢來

 Elton Huang

Draft Version

尚未校稿・僅供參考

持續更新。歡迎回饋、意見或提問

Email: eltonhuang@yahoo.com

LINE id: jesusinelton

December 23, 2021

# Contents

*What is this?* . . . 3

**1 Quantum Gates** **4**

1.1 Basics from Physics to Math . . . 4

1.1.1 Bloch Sphere . . . 7

1.1.2 Products of Matrices . . . 9

1.1.3 Quantum Annealing . . . 9

1.2 Part I . . . 10

1.2.1 Hadamard . . . 10

1.2.2 CNOT . . . 11

1.2.3 Bell States . . . 16

1.2.4 Teleportation . . . 17

1.3 Part II . . . 19

1.3.1 SWAP . . . 19

1.3.2 Toffoli (AND/OR/XOR) . . . 20

**2 Quantum Algorithms, Exercises and Advanced Topics** **21**

2.1 Deutsch-Jozsa . . . 21

2.1.1 Deutsch's Algorithm with 1 Qubit . . . 21

... Implementation with Qiskit . . . 23

2.1.2 Deutsch-Josza Algorithm with 2 Qubits . . . 29

... 3-Qubit Matrix Work . . . 34

... Implementation with Qiskit . . . 42

2.2 More Oracle-Based . . . 49

2.2.1 Berstein-Vazirani . . . 49

2.2.2 Simon . . . 49

2.3 Grover's . . . 49

2.4 Hamiltonian simulation and Trotterization . . . 50

2.5 Exercise: Superposition . . . 50

2.6 Quantum Circuit Synthesis . . . 51

**3 Applications** **52**

3.1 Optimization . . . 52

3.1.1 Quadratic Problems with Qiskit . . . . 52
3.1.2 Knapsack with Ocean . . . . 52
3.2 Monte Carlo Method . . . . 52
3.3 Machine Learning . . . . 53
3.3.1 Qiskit ML . . . . 53
3.3.2 TensorFlow Quantum . . . . 53
3.4 Chemical Simulation . . . . 54
3.4.1 Molecular Dynamics with Qiskit . . . . 54
3.4.2 Published Works . . . . 54
3.4.3 OpenFermion . . . . 54

**4 Explore and Exercise** **55**
4.1 IBM Quantum *Errors* . . . . 55
4.2 Virtual Quantum Optics Laboratory . . . . 59

**Appendices** **60**

**A ezqc.py** **61**

**B qmtx.py** **64**

**C ⟨量子電腦應用與世界級競賽實務⟩ 勘誤建議** **67**

# ***What is this?***

This document is merely a collection of notes during my course studying Quantum Computing. Most of the introductory texts in Quantum Computing do not include extensive elaborations of the matrix derivations involved in developing a concept perhaps due to the large print spans they would take. My notes started with these exercise works when I *built my intuition* to understand the subject matter.

Later it expanded to serve to structure the topics of interests as well as notes and thoughts on various fundamental and application topics, and for sharing too. Related works of Python codes are also included.

# Chapter 1

# Quantum Gates

*Simulating Physics with Computers* by Richard Feynman, 1981 is a easy read (keynote speech) and may serve as a good foundation.

## 1.1 Basics from Physics to Math

- For a layperson introduction of Quantum Theory: *The Fabric of the Cosmos: Quantum Leap* (PBS NOVA, Nov 17, 2011).
- Dr. Shankar at Yale gave a fairly clear narrative on Quantum Mechanics for a term in 2011 [1]
  - Thomas Young's Double Slit Experiment, 2013
  - Fourier Tranform
  - Euler's Equation: $e^{ix} = \cos(x) + i\sin(x)$

Wave-Particle Duality of Matter; Schrödinger Equation (MIT OCW 2017, 光電效應實驗), but as for 40:02:

- Feynman's Derivation of the Schrödinger Equation
- How did Schrödinger end up with his equation? How did he decide that his wave was going to be scalar as opposed to a vector field like E&M wave? Etc.

Schrodinger's Equation

Introduction to Quantum Mechanics: Schrodinger Equation

Nobody knows how the wave function collapses:

- Copenhagen: 由於觀測而發生塌縮 (⟨量子論縱覽⟩, 人人出版 2020, Page 79)
- Many World: decoherence of quantum state when interact with the environment which effectively "split" the universe into mutually unobservable alternate histories. (More on Wikipedia)

"Researchers observed that **carbon atoms** can tunnel. They thus overcome an energetic barrier, although they do not actually possess enough energy to do that. ... At very low temperatures under ten Kelvin, one molecule form is significantly preferred due to the energy difference."

Spin and the shape of electrons

- Electrons DO NOT Spin, 2021

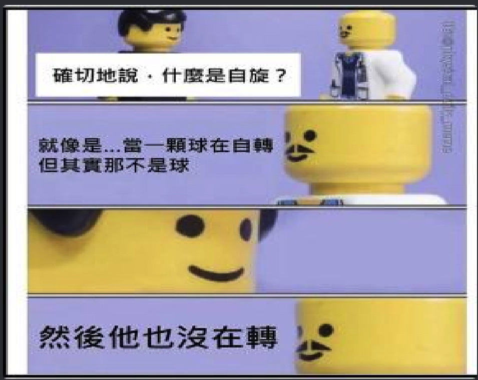
[1] Compton Effect

- How do we know if electrons are really spinning? 2018
- *Carbon displays quantum effects*, Ruhr-Universitaet-Bochum, 2017
- Electrons Are Perfectly Spherical, New Measurements Confirm, 2018
- Electron Appears Spherical, Squashing Hopes for New Physics Theories, 2013

(以下截圖自 Dicord SQCS 社群討論)

Captured with Xnip

北醫 葉智海 學術 2021/10/22




😆 10

2021年12月1日

elton 昨天00:07
是「我們確定電子沒有自己在旋轉」還是「沒有人知道電子有沒有自己在旋轉」？
是「我們確定電子的形狀不是球型」還是「還沒有人知道電子的形狀是不是球型」？ (已编辑)

北醫 葉智海 學術 昨天00:43
假設自旋是因為電子自轉
這樣電子自轉速度會超過光速
不合理
👍 1
因為波粒二象性，其實電子的形狀很難斷定
在我們沒觀測他時，他其實是波
在我們觀測他時，觀測到的半徑會隨著與電子碰撞的粒子之能量大小而有變化
👍 1

建中 賴昱錡 昨天07:49
所以算是不確定其狀態?

北醫 葉智海 學術 昨天09:17
簡言而之，不知道
在標準模型裡，目前把他視為點粒子就足夠
在 string theory 裡，電子是一維的弦，沒有體積
如果他真要有形狀，目前比較認為是圓形，但因為他有帶電，會有電矩，所以是一個很接近完美球體的橢圓形
但目前實驗還沒法確定究竟電子有沒有體積

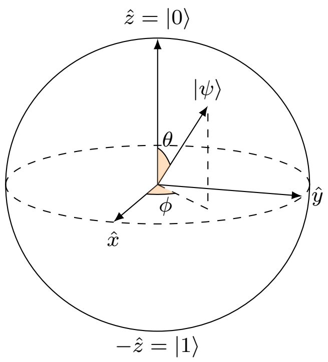
北醫 葉智海 學術 昨天09:34
目前的實驗有發現電子的偶極矩可能非常非常小，所以要嗎是真的完美球體，要麼是一個超級像球的橢圓，要嗎真的沒有體積
👍 2

建中 賴昱錡 昨天09:36
Ok，感謝解惑

Wen-Sen (Vince) 文森 昨天10:39
可能也需要釐清一下電子的體積與機率波這兩件事：機率波的詮釋（智海大說的波包的概念）應該就是一般的電子雲，說明的是"這裡有90%會看到一個帶1e電量、整顆的電子"，而不是"這裡有0.9e電量的電子"。電子雲應該是有理論可以算的（就是高中化學裡面各種球型啞鈴型的電子軌域），但詢問電子的體積我聽起來比較像是後者"這裡有90%機率看到整顆的電子，但單顆電子到底長怎樣？"這樣的問題
👍 1

### 1.1.1 Bloch Sphere

"These representations on the Bloch's circle (sphere) are combinations of the two wave functions. ... What is the physical meaning of the circle (sphere)? ... There is no physical meaning. The Bloch's circle (sphere) is a visualization; it's a way thinking about these vectors. . . . ", Cameron Akkar.

| | | | |
|---|---|---|---|
| $\frac{-\hbar^2}{2m}\frac{d^2\psi}{dr^2} + V\psi = E\psi$ |  | $\begin{aligned} \lvert\psi\rangle &= \cos(\frac{\theta}{2})\lvert 0\rangle + e^{i\phi}\sin(\frac{\theta}{2})\lvert 1\rangle \\ &\equiv e^{i\phi_\alpha}\cos(\frac{\theta}{2})\lvert 0\rangle + e^{i\phi_\beta}\sin(\frac{\theta}{2})\lvert 1\rangle\,, \\ &\quad \phi = \phi_\beta - \phi_\alpha \\ &= \alpha\lvert 0\rangle + \beta\lvert 1\rangle \end{aligned}$ | $\begin{pmatrix}\alpha \\ \beta\end{pmatrix}$ |

$|0\rangle = \begin{pmatrix}1 \\ 0\end{pmatrix}$ : $\theta = 0°$, $|1\rangle = \begin{pmatrix}0 \\ 1\end{pmatrix}$ : $\theta = 180°$.

$X|0\rangle = \begin{pmatrix}0 & 1 \\ 1 & 0\end{pmatrix}\begin{pmatrix}1 \\ 0\end{pmatrix} = \begin{pmatrix}0 \\ 1\end{pmatrix}$, $X|1\rangle = \begin{pmatrix}0 & 1 \\ 1 & 0\end{pmatrix}\begin{pmatrix}0 \\ 1\end{pmatrix} = \begin{pmatrix}1 \\ 0\end{pmatrix}$

量測 measure 時 $|\psi\rangle$ 塌縮至 $\hat{z}$ 軸

$$H|0\rangle = \tfrac{1}{\sqrt{2}}\begin{pmatrix}1 & 1 \\ 1 & -1\end{pmatrix}\begin{pmatrix}1 \\ 0\end{pmatrix} = \tfrac{1}{\sqrt{2}}\begin{pmatrix}1 \\ 1\end{pmatrix} = \tfrac{1}{\sqrt{2}}|0\rangle + \tfrac{1}{\sqrt{2}}|1\rangle = |+\rangle\,,$$

$$H|+\rangle = \tfrac{1}{2}\begin{pmatrix}1 & 1 \\ 1 & -1\end{pmatrix}\begin{pmatrix}1 \\ 1\end{pmatrix} = \tfrac{1}{2}\begin{pmatrix}2 \\ 0\end{pmatrix} = |0\rangle\,;$$

$$H|1\rangle = \tfrac{1}{\sqrt{2}}\begin{pmatrix}1 & 1 \\ 1 & -1\end{pmatrix}\begin{pmatrix}0 \\ 1\end{pmatrix} = \tfrac{1}{\sqrt{2}}\begin{pmatrix}1 \\ -1\end{pmatrix} = \tfrac{1}{\sqrt{2}}|0\rangle - \tfrac{1}{\sqrt{2}}|1\rangle = |-\rangle\,,$$

$$H|-\rangle = \tfrac{1}{2}\begin{pmatrix}1 & 1 \\ 1 & -1\end{pmatrix}\begin{pmatrix}1 \\ -1\end{pmatrix} = \tfrac{1}{2}\begin{pmatrix}0 \\ -2\end{pmatrix} = |1\rangle\,.$$

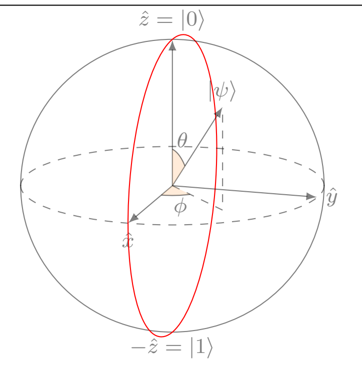
右掌大拇指：軸，四指：$|\psi\rangle$，掌面：旋轉方向

$$Z|0\rangle = \begin{pmatrix}1 & 0 \\ 0 & -1\end{pmatrix}\begin{pmatrix}1 \\ 0\end{pmatrix} = \begin{pmatrix}1 \\ 0\end{pmatrix},$$

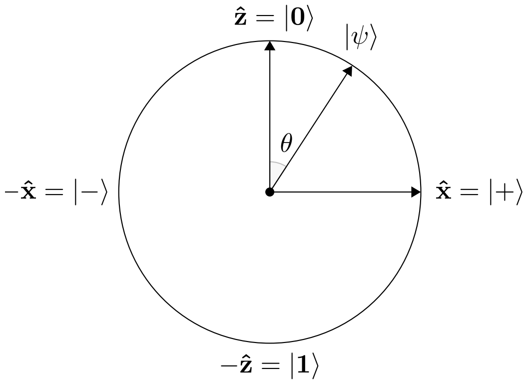
$$Z|1\rangle = \begin{pmatrix}1 & 0 \\ 0 & -1\end{pmatrix}\begin{pmatrix}0 \\ 1\end{pmatrix} = \begin{pmatrix}0 \\ -1\end{pmatrix} = -|1\rangle\,,\ \theta = -180°.$$

$$Y|0\rangle = \begin{pmatrix}0 & -i \\ i & 0\end{pmatrix}\begin{pmatrix}1 \\ 0\end{pmatrix} = \begin{pmatrix}0 \\ i\end{pmatrix} = i|1\rangle\,,$$

$$Y|1\rangle = \begin{pmatrix}0 & -i \\ i & 0\end{pmatrix}\begin{pmatrix}0 \\ 1\end{pmatrix} = \begin{pmatrix}-i \\ 0\end{pmatrix} = -i|0\rangle\,.$$

| | | | |
|---|---|---|---|
| $\frac{-\hbar^2}{2m}\frac{d^2\psi}{dr^2} + V\psi = E\psi$ |  | $\|\psi\rangle = \alpha\,\|0\rangle + \beta\,\|1\rangle$ | $\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ |

For the sake of simplicity, if we maintain $\phi = 0°$, [2]

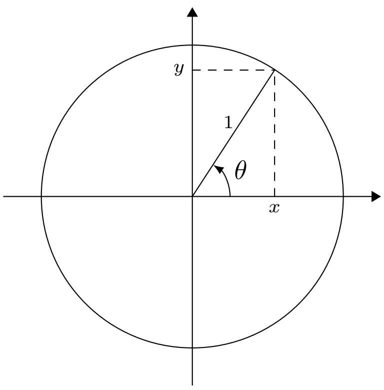



$$
\begin{aligned}
|\psi\rangle &= \cos(\frac{\theta}{2})\,|0\rangle + sin(\frac{\theta}{2})\,|1\rangle \\
&= \alpha\,|0\rangle + \beta\,|1\rangle
\end{aligned}
$$

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

§8.5 *Limitations of the quantum operations formalism* in *Quantum Computation and Quantum Information* (Nielson and Chuang 2010, Page 394 bottom)

Compare:




$$\cos(\theta) = x$$

$$\sin(\theta) = y$$

可參考張仁瑀等著 ⟨量子電腦應用與世界級競賽實務⟩ 2021，第三章：波函數和狄拉克符號之間的關係。

- Page 097 第五章：量子測量背後的數學 (Born Rule)

$$P(x) = |\,\langle x|\psi\rangle\,|^2$$

- Page 121 Clifford + T

[2] Okay up till the Grover's

### 1.1.2 Products of Matrices

Inner & outer products — Lecture 5 — Matrix Algebra for Engineers

Tensor Prodcut

### 1.1.3 Quantum Annealing

Webinar: Quantum Computing by a Quantum Annealer

Quantum annealing with more than one hundred qubits

D-Wave

## 1.2 Part I

### 1.2.1 Hadamard

$$|0\rangle + |1\rangle + |2\rangle + |3\rangle \; — \boxed{f(x) = 2x^2 - 5x + 6} — \; |6\rangle + |3\rangle + |4\rangle + |9\rangle$$

參考：鍾豪著，⟨零與一之間的威力/量子電腦的原理⟩，科學月刊 2021 (5 月號) 617 期，第 26 頁

*Quantum Computing as a High School Module* (Perry et al., 2020, Page 71 §9.2 Limitations)

注意：不能只單純理解為 qubit 進入 $|0\rangle$ 和 $|1\rangle$ 機率各半的狀態

$$|0\rangle \; — \boxed{H} — \boxed{H} — \; |0\rangle$$

$$|1\rangle \; — \boxed{H} — \boxed{H} — \; |1\rangle$$

還是要知道在 Bloch's Sphere 或是利用(複數)矩陣運算上的定義。

參考 *Quantum Computing as a High School Module* (Perry et al., 2020, Page 50 top)

### 1.2.2 CNOT

**學習單**

先完成 (A), (B), (C), (D)：(還沒學過矩陣乘法的同學可以參考連結)

如果 $q_0 = |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ 而 $q_1 = |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$,

(A) $q_1 q_0 = |10\rangle = |?\rangle \otimes |?\rangle = \begin{pmatrix} ? \times \begin{pmatrix} ? \\ ? \end{pmatrix} \\ ? \times \begin{pmatrix} ? \\ ? \end{pmatrix} \end{pmatrix} = \begin{pmatrix} ? \\ ? \\ ? \\ ? \end{pmatrix}$, 參考連結

也就是 $a_{00} = ?, a_{01} = ?, a_{10} = ?, a_{11} = ?$. 若讓 $q_1$ 以 CNOT 控制 $q_0$,

$$CNOT|10\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} a_{00} \\ a_{01} \\ a_{10} \\ a_{11} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} ? \\ ? \\ ? \\ ? \end{pmatrix} = \begin{pmatrix} ? \\ ? \\ ? \\ ? \end{pmatrix}, \text{ 也就是 } |??\rangle \text{ (如下)}$$

同樣地，

(B) $|00\rangle = |?\rangle \otimes |?\rangle = \begin{pmatrix} ? \times \begin{pmatrix} ? \\ ? \end{pmatrix} \\ ? \times \begin{pmatrix} ? \\ ? \end{pmatrix} \end{pmatrix} = \begin{pmatrix} ? \\ ? \\ ? \\ ? \end{pmatrix}$

$$CNOT|00\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} ? \\ ? \\ ? \\ ? \end{pmatrix} = \begin{pmatrix} ? \\ ? \\ ? \\ ? \end{pmatrix} = |??\rangle$$

(C) $|01\rangle = |?\rangle \otimes |?\rangle = \begin{pmatrix} ? \times \begin{pmatrix} ? \\ ? \end{pmatrix} \\ ? \times \begin{pmatrix} ? \\ ? \end{pmatrix} \end{pmatrix} = \begin{pmatrix} ? \\ ? \\ ? \\ ? \end{pmatrix}$

$$CNOT|01\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} ? \\ ? \\ ? \\ ? \end{pmatrix} = \begin{pmatrix} ? \\ ? \\ ? \\ ? \end{pmatrix} = |??\rangle$$

$$(\text{D})\ |11\rangle = |?\rangle \otimes |?\rangle = \begin{pmatrix} ? \times \begin{pmatrix} ? \\ ? \end{pmatrix} \\ ? \times \begin{pmatrix} ? \\ ? \end{pmatrix} \end{pmatrix} = \begin{pmatrix} ? \\ ? \\ ? \\ ? \end{pmatrix}$$

$$CNOT|11\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} ? \\ ? \\ ? \\ ? \end{pmatrix} = \begin{pmatrix} ? \\ ? \\ ? \\ ? \end{pmatrix} = |??\rangle$$

但是請注意：$a_{00}, a_{01}, a_{10}, a_{11}$ 可能是滿足$a_{00}^2 + a_{01}^2 + a_{10}^2 + a_{11}^2 = 1$ 的任何實數組合
各 qubit 可能處於不同的疊加態。例如，若

$$q_0 = |\alpha\rangle = \tfrac{\sqrt{3}}{2}\,|0\rangle + ?\,|1\rangle = \tfrac{\sqrt{3}}{2} \begin{pmatrix} 1 \\ 0 \end{pmatrix} + ? \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{3}}{2} \\ ? \end{pmatrix},$$

$$q_1 = |\beta\rangle = \tfrac{3}{5}\,|0\rangle + ?\,|1\rangle = \tfrac{3}{5} \begin{pmatrix} 1 \\ 0 \end{pmatrix} + ? \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} ? \\ ? \end{pmatrix},$$

$$q_1 q_0 = |\beta\alpha\rangle = |\beta\rangle \otimes |\alpha\rangle = \begin{pmatrix} ? \times \begin{pmatrix} ? \\ ? \end{pmatrix} \\ ? \times \begin{pmatrix} ? \\ ? \end{pmatrix} \end{pmatrix} = \begin{pmatrix} ? \\ ? \\ ? \\ ? \end{pmatrix}, \text{(注意這四個數的平方和為1)}$$

$$CNOT\,|\beta\alpha\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} ? \\ ? \\ ? \\ ? \end{pmatrix} = \begin{pmatrix} ? \\ ? \\ ? \\ ? \end{pmatrix} = ?\,|00\rangle + ?\,|01\rangle + ?\,|10\rangle + ?\,|11\rangle$$

**Solution**

If $q_0 = |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $q_1 = |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$,

$$q_1 q_0 = |10\rangle = |1\rangle \otimes |0\rangle = \begin{pmatrix} 0 \times \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ 1 \times \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$a_{00} = 0, a_{01} = 0, a_{10} = 1, a_{11} = 0$. Let $q_1$ control $q_0$ with CNOT,

$$CNOT|10\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} a_{00} \\ a_{01} \\ a_{10} \\ a_{11} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \text{ } i.e. \text{ } |11\rangle \text{ by below}$$

likewise,

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \times \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ 0 \times \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$CNOT|00\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = |00\rangle$$

$$|01\rangle = |0\rangle \otimes |1\rangle = \begin{pmatrix} 1 \times \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 0 \times \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

$$CNOT|01\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = |01\rangle$$

$$|11\rangle = |1\rangle \otimes |1\rangle = \begin{pmatrix} 0 \times \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 1 \times \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$CNOT|11\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = |10\rangle$$

Xref: *Quantum Computing for Everyone*, Bernhardt 2019, Chapter 4, Page 64∼67

But please note: $a_{00}, a_{01}, a_{10}, a_{11}$ may be any combination of real numbers that satisfies $a_{00}^2 + a_{01}^2 + a_{10}^2 + a_{11}^2 = 1$. (Each qubit may be in different variance of superposition of states.) For example, if

$$q_0 = |\alpha\rangle = \frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle = \frac{\sqrt{3}}{2}\begin{pmatrix}1\\0\end{pmatrix} + \frac{1}{2}\begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}\frac{\sqrt{3}}{2}\\\frac{1}{2}\end{pmatrix},$$

$$q_1 = |\beta\rangle = \frac{3}{5}|0\rangle + \frac{4}{5}|1\rangle = \frac{3}{5}\begin{pmatrix}1\\0\end{pmatrix} + \frac{4}{5}\begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}\frac{3}{5}\\\frac{4}{5}\end{pmatrix},$$

$$q_1 q_0 = |\beta\alpha\rangle = |\beta\rangle \otimes |\alpha\rangle = \begin{pmatrix}\frac{3}{5} \times \begin{pmatrix}\frac{\sqrt{3}}{2}\\\frac{1}{2}\end{pmatrix}\\ \frac{4}{5} \times \begin{pmatrix}\frac{\sqrt{3}}{2}\\\frac{1}{2}\end{pmatrix}\end{pmatrix} = \begin{pmatrix}\frac{3\sqrt{3}}{10}\\\frac{3}{10}\\\frac{4\sqrt{3}}{10}\\\frac{4}{10}\end{pmatrix}, \text{(note that the squares of all elements sum up to 1)}$$

$$CNOT|\beta\alpha\rangle = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\begin{pmatrix}\frac{3\sqrt{3}}{10}\\\frac{3}{10}\\\frac{4\sqrt{3}}{10}\\\frac{4}{10}\end{pmatrix} = \begin{pmatrix}\frac{3\sqrt{3}}{10}\\\frac{3}{10}\\\frac{4}{10}\\\frac{4\sqrt{3}}{10}\end{pmatrix} = \frac{3\sqrt{3}}{10}|00\rangle + \frac{3}{10}|01\rangle + \frac{4}{10}|10\rangle + \frac{4\sqrt{3}}{10}|11\rangle$$

The 2 qubits are entangled now, so we can't decompose them into 2 separated qubits for distinct wave probabilities for each quibt as

$$\begin{pmatrix}\frac{3\sqrt{3}}{10}\\\frac{3}{10}\\\frac{4}{10}\\\frac{4\sqrt{3}}{10}\end{pmatrix} = |\delta\gamma\rangle, \text{If } |\gamma\rangle = c_0|0\rangle + c_1|1\rangle, \text{ and } |\delta\rangle = d_0|0\rangle + d_1|1\rangle,$$

$$|\delta\gamma\rangle = |\delta\rangle \otimes |\gamma\rangle = \begin{pmatrix}d_0 \times \begin{pmatrix}c_0\\c_1\end{pmatrix}\\ d_1 \times \begin{pmatrix}c_0\\c_1\end{pmatrix}\end{pmatrix} = \begin{pmatrix}d_0c_0\\d_0c_1\\d_1c_0\\d_1c_1\end{pmatrix} = \begin{pmatrix}\frac{3\sqrt{3}}{10}\\\frac{3}{10}\\\frac{4}{10}\\\frac{4\sqrt{3}}{10}\end{pmatrix}$$

Xref: *Quantum Computing as a High School Module* (Perry et al., 2020, Page 58 §7.4 second example).

Multiple Qubits are in entanglement $\Leftrightarrow$

The state of the Multiple Qubits combined can*not* be expressed as a tensor product of each of the 2 qubits.

The No-Cloning Theorem: you can't treat the qubits in entanglement individually.

(以下截圖自 Dicord SQCS 社群討論)

Captured with Xnip

**elton** 昨天00:27

@Wen-Sen (Vince) 文森 可否請教一下：How we manifest entanglement of transmon qubits with resonance is not really quantum entanglement? Or resonance is not the only way to achieve quantum entanglement; other systems may use other mechanisms?

**北醫 葉智海 學術** 昨天00:51

參見仁禹出版書中量子硬體-超導-superconducting qubit-qubit coupling （誤

@elton @Wen-Sen (Vince) 文森 可否請教一下：How we manifest entanglement of transmon qubits with res...

**Wen-Sen (Vince) 文森** 昨天10:22

我不確定這邊 entanglement 指的是不是同一件事，但基本上cross-resonance (IBM的做法) 裡 two qubit gates 的概念很類似軌域混成的做法（順著之前 transmon 是人造原子的說法，可以想像成2 qubit/transmon system 是人造"分子"），讓兩個 qubit 各自與 resonator (bus resonator instead of readout resonator, to be precise) 混成後對齊其中一對能階，以允許兩個 qubit 交換光子 (and hence "cross"-resonance)。原則上要理解為何能達到 CNOT 需要定量寫出 two qubit Hamiltonian，但基本上這就允許了製備糾纏的兩個量子系統，所以我不是很清楚原文裡面 "entanglement" 指稱的是什麼。

細節可以參考 Rigetti 的博論 3.3 節，他開頭有定性的描述，然後是定量的推導（我沒真的推過導到CNOT那部分就是了 :p）
https://cpb-us-w2.wpmucdn.com/campuspress.yale.edu/dist/2/3627/files/2020/10/Rigetti-PhDthesis-QuantumGatesForSuperconductingQubits-Yale2009.pdf

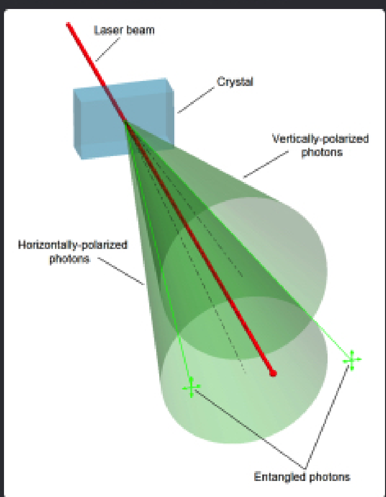
另外不同平台使用的當然很有可能不同，但我個人覺得因為光子交換很 robust，所以 photon bus 還是比較常聽到的做法（純個人看法） (已编辑)

👍 1

**elton** 昨天14:39

謝謝回覆，印象中讀過的一兩本包括 Nielson 的好像沒有特別說實際上 entanglement 如何實現，仁瑀他們那本的 35 頁最上面有這段描述，也希望能問問看那段的原作者能否提供參考資料連結，謝謝：「當兩個粒子類似且距離足夠近的時候，它們就會發生糾纏，同時量子理論表示，就算您將這兩個粒子分開也不能改變它們兩個之間的糾纏狀態」 (已编辑)

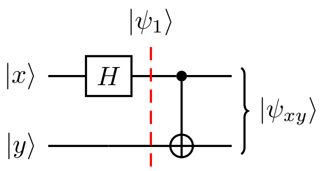
**北醫 葉智海 學術** 昨天15:32




👍 1

這是製造兩個糾纏光子的方法

### 1.2.3 Bell States




Let $|x\rangle = x_0\,|0\rangle + x_1\,|1\rangle = \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$, $|y\rangle = y_0\,|0\rangle + y_1\,|1\rangle = \begin{pmatrix} y_0 \\ y_1 \end{pmatrix}$

$$H\,|x\rangle = \tfrac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \tfrac{1}{\sqrt{2}} \begin{pmatrix} x_0 + x_1 \\ x_0 - x_1 \end{pmatrix}$$

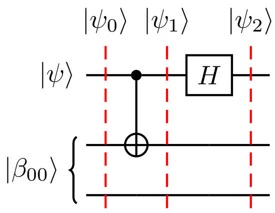
$$|\psi_1\rangle = (H\,|x\rangle) \otimes |y\rangle = \tfrac{1}{\sqrt{2}} \begin{pmatrix} (x_0+x_1)y_0 \\ (x_0+x_1)y_1 \\ (x_0-x_1)y_0 \\ (x_0-x_1)y_1 \end{pmatrix}$$

$$|\psi_{xy}\rangle = CNOT\,|\psi_1\rangle = \tfrac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} (x_0+x_1)y_0 \\ (x_0+x_1)y_1 \\ (x_0-x_1)y_0 \\ (x_0-x_1)y_1 \end{pmatrix} = \tfrac{1}{\sqrt{2}} \begin{pmatrix} (x_0+x_1)y_0 \\ (x_0+x_1)y_1 \\ (x_0-x_1)y_1 \\ (x_0-x_1)y_0 \end{pmatrix}$$

| $\lvert\psi_{xy}\rangle$ | $\lvert y\rangle = \lvert 0\rangle$ <br> $y_0 = 1, y_1 = 0$ | $\lvert y\rangle = \lvert 1\rangle$ <br> $y_0 = 0, y_1 = 1$ |
|---|---|---|
| $\lvert x\rangle = \lvert 0\rangle$ <br> $x_0 = 1$ <br> $x_1 = 0$ | $\tfrac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\0\\1\end{pmatrix} = \tfrac{1}{\sqrt{2}}(\begin{pmatrix}1\\0\\0\\0\end{pmatrix} + \begin{pmatrix}0\\0\\0\\1\end{pmatrix}) = \frac{\lvert 00\rangle + \lvert 11\rangle}{\sqrt{2}}$ | $\tfrac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\1\\0\end{pmatrix} = \tfrac{1}{\sqrt{2}}(\begin{pmatrix}0\\1\\0\\0\end{pmatrix} + \begin{pmatrix}0\\0\\1\\0\end{pmatrix}) = \frac{\lvert 01\rangle + \lvert 10\rangle}{\sqrt{2}}$ |
| $\lvert x\rangle = \lvert 1\rangle$ <br> $x_0 = 0$ <br> $x_1 = 1$ | $\tfrac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\0\\-1\end{pmatrix} = \tfrac{1}{\sqrt{2}}(\begin{pmatrix}1\\0\\0\\0\end{pmatrix} - \begin{pmatrix}0\\0\\0\\1\end{pmatrix}) = \frac{\lvert 00\rangle - \lvert 11\rangle}{\sqrt{2}}$ | $\tfrac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\-1\\0\end{pmatrix} = \tfrac{1}{\sqrt{2}}(\begin{pmatrix}0\\1\\0\\0\end{pmatrix} - \begin{pmatrix}0\\0\\1\\0\end{pmatrix}) = \frac{\lvert 01\rangle - \lvert 10\rangle}{\sqrt{2}}$ |

Xref: *Quantum Computation and Quantum Information* (Nielson and Chuang 2010, Page 26)

### 1.2.4 Teleportation




$$\textit{Let } |\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix}$$

$$|\psi_0\rangle = |\psi\rangle \otimes |\beta_{00}\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} \alpha \times \begin{pmatrix}1\\0\\0\\1\end{pmatrix} \\ \beta \times \begin{pmatrix}1\\0\\0\\1\end{pmatrix}\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}\alpha\\0\\0\\\alpha\\\beta\\0\\0\\\beta\end{pmatrix} = \frac{1}{\sqrt{2}}(\alpha(|000\rangle + |011\rangle) + \beta(|100\rangle + |111\rangle)), \text{ (exercise: verify)}$$

$$|\psi_1\rangle = (CNOT_{2qb} \otimes I_{1qb})\,|\psi_0\rangle = (\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix} \otimes \begin{pmatrix}1&0\\0&1\end{pmatrix})\,|\psi_0\rangle = \begin{pmatrix} \begin{pmatrix}1&0\\0&1\end{pmatrix} & 0 & 0 & 0 \\ 0 & \begin{pmatrix}1&0\\0&1\end{pmatrix} & 0 & 0 \\ 0 & 0 & 0 & \begin{pmatrix}1&0\\0&1\end{pmatrix} \\ 0 & 0 & \begin{pmatrix}1&0\\0&1\end{pmatrix} & 0 \end{pmatrix} |\psi_0\rangle$$

$$= \frac{1}{\sqrt{2}}\begin{pmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&1\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\end{pmatrix}\begin{pmatrix}\alpha\\0\\0\\\alpha\\\beta\\0\\0\\\beta\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}\alpha\\0\\0\\\alpha\\0\\\beta\\\beta\\0\end{pmatrix} = \frac{1}{\sqrt{2}}(\alpha(|000\rangle + |011\rangle) + \beta(|101\rangle + |110\rangle))$$

$$|\psi_2\rangle = (H_{1qb} \otimes I_{1qb} \otimes I_{1qb}) |\psi_1\rangle = (\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}) |\psi_1\rangle$$

$$= (\frac{1}{\sqrt{2}} \begin{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & -\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}) |\psi_1\rangle = (\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}) |\psi_1\rangle$$

$$= \frac{1}{2} \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} \alpha \\ 0 \\ 0 \\ \alpha \\ 0 \\ \beta \\ \beta \\ 0 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} \alpha \\ \beta \\ \beta \\ \alpha \\ \alpha \\ -\beta \\ -\beta \\ \alpha \end{pmatrix}$$

$$= \frac{1}{2}(\alpha(|000\rangle + |011\rangle + |100\rangle + |111\rangle) + \beta(|001\rangle + |010\rangle - |101\rangle - |110\rangle))$$

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

## 1.3 Part II

### 1.3.1 SWAP

Let $|\alpha\rangle = a_0 |0\rangle + a_1 |1\rangle = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix}$, $|\beta\rangle = b_0 |0\rangle + b_1 |1\rangle = \begin{pmatrix} b_0 \\ b_1 \end{pmatrix}$

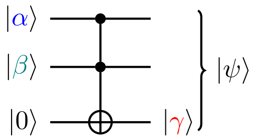
SWAP$|\alpha\beta\rangle = |\beta\alpha\rangle$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_0 b_0 \\ a_0 b_1 \\ a_1 b_0 \\ a_1 b_1 \end{pmatrix} = \begin{pmatrix} a_0 b_0 \\ a_1 b_0 \\ a_0 b_1 \\ a_1 b_1 \end{pmatrix} = \begin{pmatrix} b_0 a_0 \\ b_0 a_1 \\ b_1 a_0 \\ b_1 a_1 \end{pmatrix}$$

### 1.3.2 Toffoli (AND/OR/XOR)

Xref: *Quantum Computing for Everyone*, Bernhardt 2019, Chapter 6, Page 90~91

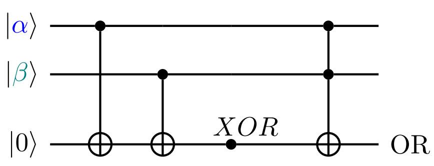
**AND**




$$|\psi\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha_0\beta_0 \\ 0 \\ \alpha_0\beta_1 \\ 0 \\ \alpha_1\beta_0 \\ 0 \\ \alpha_1\beta_1 \\ 0 \end{pmatrix} = \begin{pmatrix} \alpha_0\beta_0 \\ 0 \\ \alpha_0\beta_1 \\ 0 \\ \alpha_1\beta_0 \\ 0 \\ 0 \\ \alpha_1\beta_1 \end{pmatrix} = \alpha_0\beta_0 |000\rangle + \alpha_0\beta_1 |010\rangle + \alpha_1\beta_0 |100\rangle + \alpha_1\beta_1 |111\rangle$$

| $\lvert\psi\rangle$ | $\lvert\beta\rangle = \lvert 0\rangle$<br>$\beta_0 = 1, \beta_1 = 0$ | $\lvert\beta\rangle = \lvert 1\rangle$<br>$\beta_0 = 0, \beta_1 = 1$ |
|---|---|---|
| $\lvert\alpha\rangle = \lvert 0\rangle$<br>$\alpha_0 = 1$<br>$\alpha_1 = 0$ | $\lvert 000\rangle$ | $\lvert 010\rangle$ |
| $\lvert\alpha\rangle = \lvert 1\rangle$<br>$\alpha_0 = 0$<br>$\alpha_1 = 1$ | $\lvert 100\rangle$ | $\lvert 111\rangle$ |

**OR/XOR**



# Chapter 2

# Quantum Algorithms, Exercises and Advanced Topics

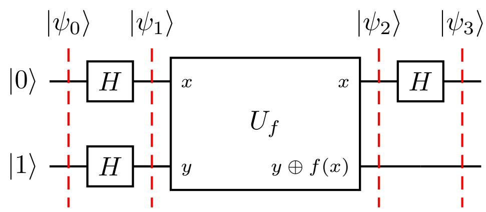
## 2.1 Deutsch-Jozsa

### 2.1.1 Deutsch's Algorithm with 1 Qubit

The function $f(x)$ is:

"constant" if regardless the input $x$, $f(x)$ always produces the same results; i.e. $f(|0\rangle) = f(|1\rangle)$.

"balanced" if for different input $x$, $f(x)$ produces different results; so $f(|0\rangle) \neq f(|1\rangle)$.



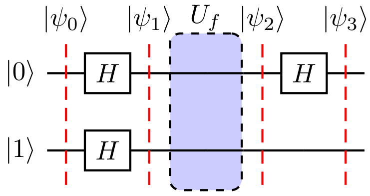

**The Deutsch's algorithm**:

1. Design the oracle $U_f$ according to $f(x)$ so that $U_f |x, y\rangle = |x, y \otimes f(x)\rangle$.
   It can be one of the four circuits below.
2. By looking at the state of 1st qubit of output of the circuit $\underline{(\boldsymbol{H} \otimes \boldsymbol{I})}U_f\underline{(\boldsymbol{H} \otimes \boldsymbol{H})(|\boldsymbol{0}\rangle \otimes |\boldsymbol{1}\rangle)}$
   (i.e. $|\psi_3\rangle$ in the diagram above), can determine whether f(x) is constant or balanced. [1]

$$|\psi_1\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle-|1\rangle}{\sqrt{2}} = \frac{|00\rangle-|01\rangle+|10\rangle-|11\rangle}{2} = \frac{1}{2}\begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix}$$

[1] $|\alpha\beta\rangle = |\alpha\rangle |\beta\rangle = |\alpha\rangle \otimes |\beta\rangle$

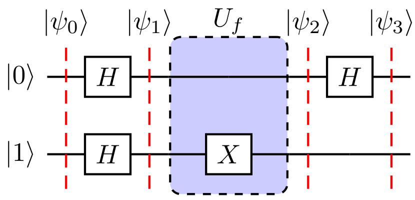
For $f(x) = |0\rangle$,




$|\psi_2\rangle = |\psi_1\rangle$

$$|\psi_3\rangle = \tfrac{1}{2\sqrt{2}} \begin{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & -\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} = \tfrac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \\ 0 \\ 0 \end{pmatrix} = \tfrac{|00\rangle - |01\rangle}{\sqrt{2}} = |0\rangle \otimes |-\rangle$$

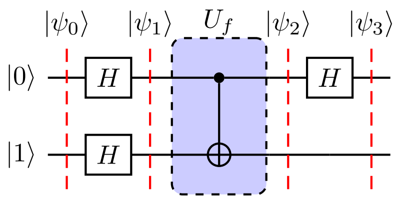
For $f(x) = |1\rangle$,




$$|\psi_2\rangle = (I \otimes X)\,|\psi_1\rangle = \tfrac{1}{2} \begin{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} & 0 \\ 0 & \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} = \tfrac{1}{2} \begin{pmatrix} -1 \\ 1 \\ -1 \\ 1 \end{pmatrix}$$

$|\psi_3\rangle = -\frac{|00\rangle - |01\rangle}{\sqrt{2}} = |0\rangle \otimes (-\,|-\rangle)$ [2]

For $f(x) = x$,

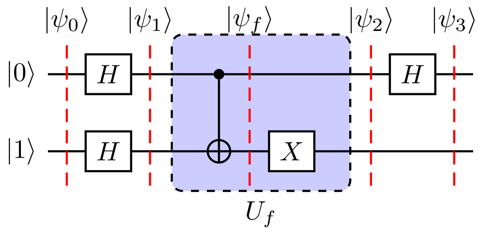



$$|\psi_2\rangle = CNOT\,|\psi_1\rangle = \tfrac{1}{2} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} = \tfrac{1}{2} \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix}$$

$$|\psi_3\rangle = \tfrac{1}{2\sqrt{2}} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix} = \tfrac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ 0 \\ 1 \\ -1 \end{pmatrix} = \tfrac{|10\rangle - |11\rangle}{\sqrt{2}} = |1\rangle \otimes |-\rangle$$

[2] $|q\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\frac{\theta}{2}\,|1\rangle$: with $-\,|q\rangle$, $\frac{\theta}{2}$ turns 180°, i.e. $\theta$ turns 360°.

For $f(x) = x \oplus |1\rangle = \overline{x}$,



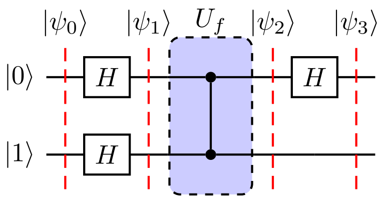

$$|\psi_2\rangle = (I \otimes X)\,|\psi_f\rangle = \tfrac{1}{2}\begin{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} & 0 \\ 0 & \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix} = \tfrac{1}{2}\begin{pmatrix} -1 \\ 1 \\ 1 \\ -1 \end{pmatrix}$$

$$|\psi_3\rangle = {\color{red}|1\rangle} \otimes (-\,|-\rangle)$$

Observation:

In order to reflect the balanced nature of the function, the oracle brings the data and target qubits into entanglement whereas the oracles for the constant functions do not, thus with the oracles for constant functions, the second Hadamard gate brings the qubit back to it's original state $|0\rangle$.

**The CZ Oracle**




$$|\psi_2\rangle = CZ\,|\psi_1\rangle = \tfrac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} = \tfrac{1}{2}\begin{pmatrix} 1 \\ -1 \\ 1 \\ 1 \end{pmatrix}$$

$$|\psi_3\rangle = \tfrac{1}{2\sqrt{2}}\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ 1 \\ 1 \end{pmatrix} = \tfrac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 0 \\ 0 \\ -1 \end{pmatrix} = \tfrac{|00\rangle - |11\rangle}{\sqrt{2}}$$

**... Implementation of 1-Qubit Deutsch's Algorithm with Qiskit (Next Page)**

# djqskt

October 10, 2021

Qiskit 實作：Deutsch-Josza


[1]:
```
from ezqc import *
from math import pi

# 定義 the Deutsch's Algorithm

def dj1 (uf):
    qc = EzQC (2)
    qc.set_qubits ([(0, 0), (1, 1)], bb = False)
    qc.h (0)
    qc.h (1)
    qc.barrier ()
    uf (qc)
    qc.barrier ()
    qc.h (0)
    qc.measure (0, 0)
    qc.draw("mpl")
    return ('constant' if list (run_real (qc, shots = 1).keys())[0][1] == '0'␣
↪else 'balanced')
    # 理論上只跑一次就夠了，會回傳: {'0x': 1} (results of lower qubit indices,␣
↪top of circuit plots, on the right)
    # q1 is of value ket-0 due to no measurement (Todo: verify)
```

uf1c00: $f(x) = |0\rangle$, constant
uf1c01: $f(x) = |0\rangle$, constant
uf1c10: $f(x) = |1\rangle$, constant
uf1b00: $f(x) = x$, balanced
uf1b10: $f(x) = x \oplus |1\rangle = \overline{x}$, balanced

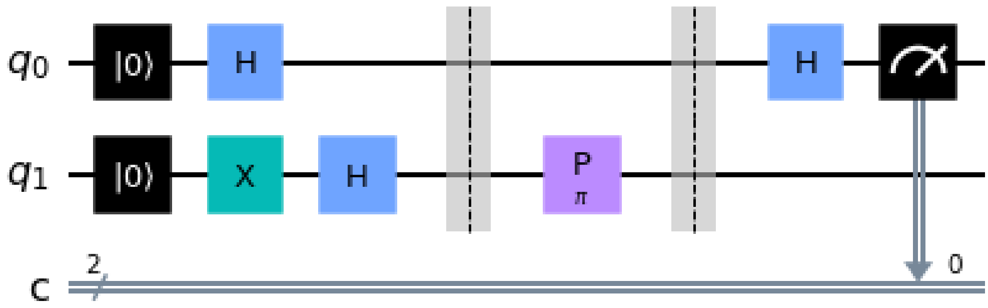
[2]:
```
# 定義 1 qubit 的 4 種 Uf (oracles)

def uf1c00 (qc): # constant
    pass

def uf1c01 (qc): # constant
```

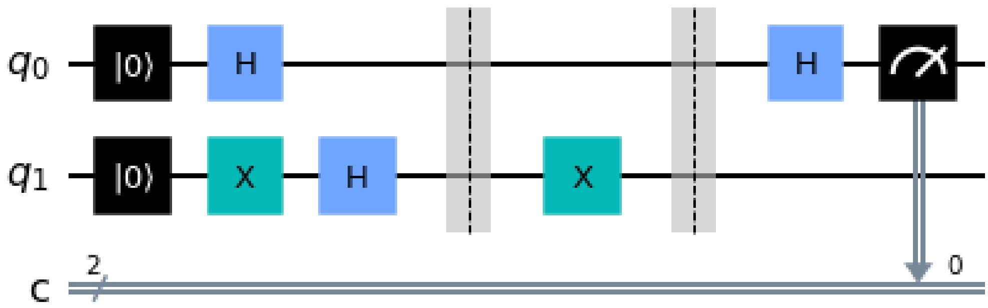
```
    qc.p (pi, 1)

def uf1c10 (qc): # constant
    qc.x (1)

def uf1b00 (qc): # balanced
    qc.cnot (0, 1)

# def uf1b01 (qc): # balanced?
#     qc.cz (0, 1)

def uf1b10 (qc): # balanced
    qc.cnot (0, 1)
    qc.x (1)
```

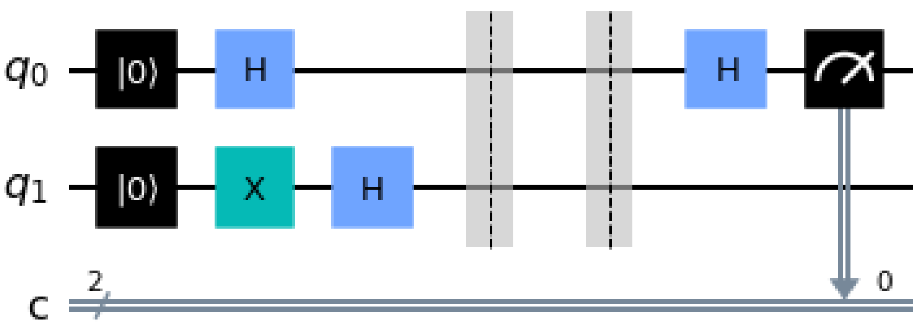
[4]:
```
from random import shuffle

# 將 4 個 Uf 存到一個 list 中，並註記名稱
uf1s = [ ('uf1c00', uf1c00), ('uf1c10', uf1c10), ('uf1b00', uf1b00), ('uf1b10',␣
 ↪uf1b10),
         ('uf1c01', uf1c01) ] #, ('uf1b01', uf1b01) ]

# 4 個 Uf 隨機排序
shuffle (uf1s)

# 看看 dj1 是否能做出正確判斷
for name, orcl in uf1s:
    print ()
    print (name + ": " + dj1 (orcl))
```

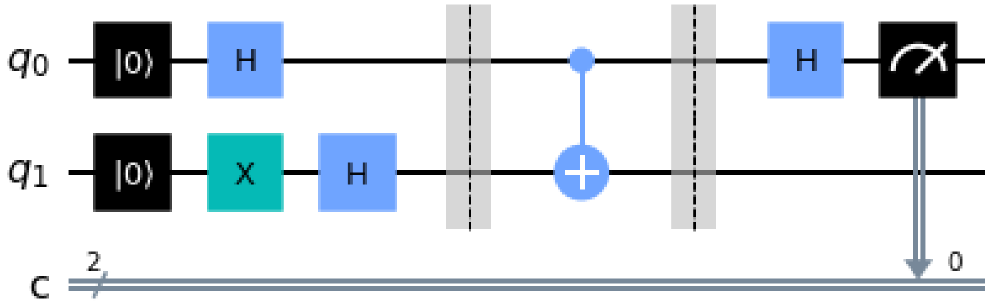
```
The best backend is ibmq_belem 5 qubit(s)
Job Status: job has successfully run
uf1c01: constant

The best backend is ibmq_belem 5 qubit(s)
Job Status: job has successfully run
uf1c10: constant

The best backend is ibmq_belem 5 qubit(s)
Job Status: job has successfully run
uf1c00: constant

The best backend is ibmq_belem 5 qubit(s)
Job Status: job has successfully run
uf1b00: balanced

The best backend is ibmq_belem 5 qubit(s)
```

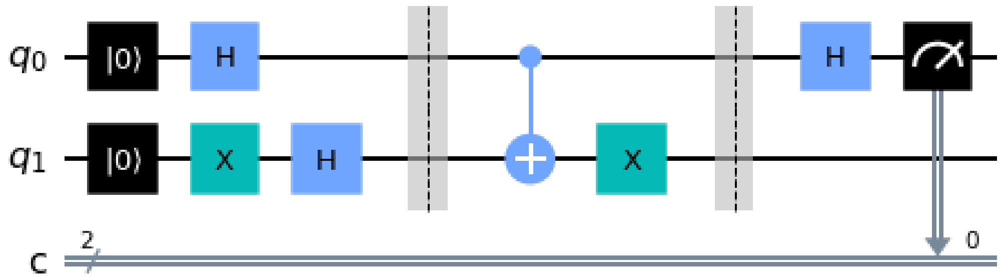
```
Job Status: job has successfully run
uf1b10: balanced
```






### 0.0.1 Study CZ Orcale

[2]:
```python
from qiskit.quantum_info import Statevector
#from qiskit.visualization import plot_bloch_multivector
#from qiskit.visualization import array_to_latex

qc = EzQC (2)
qc.set_qubits ([(0, 0), (1, 1)], bb = False)
#plot_bloch_multivector(Statevector.from_instruction(qc), title="cz",␣
↪reverse_bits=False);
#array_to_latex(Statevector.from_instruction(qc), prefix="\\text{\\psi_0 = }")
Statevector.from_instruction(qc).draw("latex", prefix="|\psi_0\\rangle = ")
```

[2]:

$$|\psi_0\rangle = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$$

$q_1 \otimes q_0$, **swap the middle to match** $q_0 \otimes q_1$

[3]:
```
qc.h (0)
qc.h (1)
qc.barrier ()
Statevector.from_instruction(qc).draw("latex", prefix="|\psi_1\\rangle = ")
```

[3]:

$$|\psi_1\rangle = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \end{bmatrix}$$

[4]:
```
qc.cz (0, 1)
qc.barrier ()
Statevector.from_instruction(qc).draw("latex", prefix="|\psi_2\\rangle = ")
```

[4]:

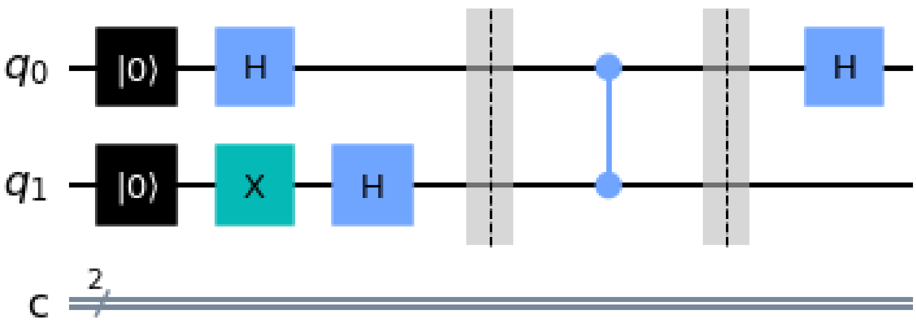
$$|\psi_2\rangle = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

[5]:
```
qc.h (0)
Statevector.from_instruction(qc).draw("latex", prefix="|\psi_3\\rangle = ")
```

[5]:

$$|\psi_3\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

[7]:
```
qc.draw("mpl");
```



### 2.1.2 Deutsch-Josza Algorithm with 2 Qubits

1. bring all qubits to superpositions

   $|\psi_1\rangle = (H \otimes H \otimes H)\,|001\rangle = \frac{1}{\sqrt{2}^3}(+\,|000\rangle - |001\rangle + |010\rangle - |011\rangle + |100\rangle - |101\rangle + |110\rangle - |111\rangle)$

**Constant**

For $f(x^{\otimes 2}) = |0\rangle$,

2. apply oracle

   $|\psi_2\rangle = |\psi_1\rangle$

3. apply $H^{\otimes 2}$ to the data qubits

   $|\psi_3\rangle = (H \otimes H \otimes I)\,|\psi_2\rangle = \frac{1}{\sqrt{2}^1}(+\,|000\rangle - |001\rangle)$

For $f(x^{\otimes 2}) = |1\rangle$,

2. apply oracle

   $|\psi_2\rangle = (I \otimes I \otimes X)\,|\psi_1\rangle = \frac{1}{\sqrt{2}^3}(-\,|000\rangle + |001\rangle - |010\rangle + |011\rangle - |100\rangle + |101\rangle - |110\rangle + |111\rangle)$

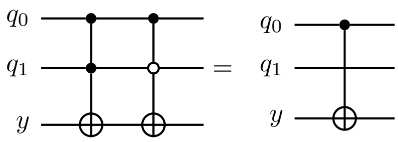
3. apply $H^{\otimes 2}$ to the data qubits

   $|\psi_3\rangle = \frac{1}{\sqrt{2}^1}(-\,|000\rangle + |001\rangle)$

**Balanced**

For $f(|11\rangle) = f(|10\rangle) = |1\rangle$

Note: $y \oplus |1\rangle = \overline{y}$, so the oracle is




(Note this is just the oracle, it needs to be plugged into the circuit with the Hadamard gates before and after according to the Deutsch-Josza algorithm)

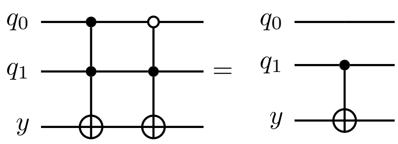
2. apply oracle

   $|\psi_2\rangle = \frac{1}{\sqrt{2}^3}(+\,|000\rangle - |001\rangle + |010\rangle - |011\rangle - |100\rangle + |101\rangle - |110\rangle + |111\rangle)$

3. apply $H^{\otimes 2}$ to the data qubits

   $|\psi_3\rangle = \frac{1}{\sqrt{2}^1}(+\,|100\rangle - |101\rangle)$

For $f(|11\rangle) = f(|01\rangle) = |1\rangle$



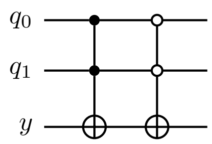

2. apply oracle

   $|\psi_2\rangle = \frac{1}{\sqrt{2}^3}(+\,|000\rangle - |001\rangle - |010\rangle + |011\rangle + |100\rangle - |101\rangle - |110\rangle + |111\rangle)$

3. apply $H^{\otimes 2}$ to the data qubits

   $|\psi_3\rangle = \frac{1}{\sqrt{2}^1}(+\,|010\rangle - |011\rangle)$

For $f(|11\rangle) = f(|00\rangle) = |1\rangle$




2. apply oracle

   $|\psi_2\rangle = \frac{1}{\sqrt{2}^3}(-|000\rangle + |001\rangle + |010\rangle - |011\rangle + |100\rangle - |101\rangle - |110\rangle + |111\rangle)$

3. apply $H^{\otimes 2}$ to the data qubits

   $|\psi_3\rangle = \frac{1}{\sqrt{2}^1}(-|110\rangle + |111\rangle)$

Next Page: Matrix Works

# dj2

October 10, 2021

## 0.1 Two-Qubit Deutsch-Josza



[1]:
```
from importlib import reload
import numpy as np
from IPython.display import display, Math
import qmtx as qm
import ezqc
reload (qm)
reload (ezqc)
```

[1]:
```
<module 'ezqc' from '/Volumes/extra0/elton4k12/spqc/ezqc.py'>
```

## 0.2 Constant

### 0.2.1 For $f(x^{\otimes 2}) = |0\rangle$,

1. bring all qubits to superpositions

[2]:
```
psi1 = qm.kron3 (np.dot (qm.h1, qm.ket0), np.dot (qm.h1, qm.ket0), np.dot (qm.
↪h1, qm.ket1))
qm.ket3 (psi1, "\\psi_1")
```

$$|\psi_1\rangle = \frac{1}{\sqrt{2}^3}(+|000\rangle - |001\rangle + |010\rangle - |011\rangle + |100\rangle - |101\rangle + |110\rangle - |111\rangle)$$

2. apply orcale

[3]:
```
psi2 = np.dot (qm.kron3 (qm.i1, qm.i1, qm.i1), psi1)
qm.ket3 (psi2, "\\psi_2")
```

$$|\psi_2\rangle = \frac{1}{\sqrt{2}^3}(+|000\rangle - |001\rangle + |010\rangle - |011\rangle + |100\rangle - |101\rangle + |110\rangle - |111\rangle)$$

3. apply $H^{\otimes 2}$ to the data qubits

[4]:
```
psi3 = np.dot (qm.kron3 (qm.h1, qm.h1, qm.i1), psi1)
qm.ket3 (psi3, "\\psi_3")
```

$$|\psi_3\rangle = \frac{1}{\sqrt{2}^1}(+|000\rangle - |001\rangle)$$

#### 0.2.2 For $f(x^{\otimes 2}) = |1\rangle$

2. apply oracle

[5]:
```
psi2 = np.dot (qm.kron3 (qm.i1, qm.i1, qm.x1), psi1)
qm.ket3 (psi2, "\\psi_2")
```

$$|\psi_2\rangle = \frac{1}{\sqrt{2}^3}(-|000\rangle + |001\rangle - |010\rangle + |011\rangle - |100\rangle + |101\rangle - |110\rangle + |111\rangle)$$

3. apply $H^{\otimes 2}$ to the data qubits

[6]:
```
psi3 = np.dot (qm.kron3 (qm.h1, qm.h1, qm.i1), psi2)
qm.ket3 (psi3, "\\psi_3")
```

$$|\psi_3\rangle = \frac{1}{\sqrt{2}^1}(-|000\rangle + |001\rangle)$$

### 0.3 Balanced

#### 0.3.1 For $f(|11\rangle) = f(|10\rangle) = |1\rangle$

note: $y \oplus |1\rangle = \overline{y}$

2. apply oracle

[7]:
```
psi2 = np.dot (qm.cox, np.dot (qm.ccx, psi1))
qm.ket3 (psi2, "\\psi_2", raw=False)
```

$$|\psi_2\rangle = \frac{1}{\sqrt{2}^3}(+|000\rangle - |001\rangle + |010\rangle - |011\rangle - |100\rangle + |101\rangle - |110\rangle + |111\rangle)$$

[8]:
```
# simplified
psi2s = np.dot (qm.cix, psi1)
qm.ket3 (psi2s, "\\psi_{2s}", raw=False)
```

$$|\psi_{2s}\rangle = \frac{1}{\sqrt{2}^3}(+|000\rangle - |001\rangle + |010\rangle - |011\rangle - |100\rangle + |101\rangle - |110\rangle + |111\rangle)$$

3. apply $H^{\otimes 2}$ to the data qubits

[9]:
```
psi3 = np.dot (qm.kron3 (qm.h1, qm.h1, qm.i1), psi2)
qm.ket3 (psi3, "\\psi_3", raw=False)
```

$$|\psi_3\rangle = \frac{1}{\sqrt{2}^1}(+|100\rangle - |101\rangle)$$

### 0.3.2 For $f(|11\rangle) = f(|01\rangle) = |1\rangle$

2. apply oracle

[10]:
```
psi2 = np.dot (qm.ocx, np.dot (qm.ccx, psi1))
qm.ket3 (psi2, "\\psi_2", raw=False)
```

$$|\psi_2\rangle = \frac{1}{\sqrt{2}^3}(+|000\rangle - |001\rangle - |010\rangle + |011\rangle + |100\rangle - |101\rangle - |110\rangle + |111\rangle)$$

[11]:
```
qm.ket3 (np.dot (qm.icx, psi1), "\\psi_{2s}", raw=False)
# simplified
```

$$|\psi_{2s}\rangle = \frac{1}{\sqrt{2}^3}(+|000\rangle - |001\rangle - |010\rangle + |011\rangle + |100\rangle - |101\rangle - |110\rangle + |111\rangle)$$

3. apply $H^{\otimes 2}$ to the data qubits

[12]:
```
psi3 = np.dot (qm.kron3 (qm.h1, qm.h1, qm.i1), psi2)
qm.ket3 (psi3, "\\psi_3", raw=False)
```

$$|\psi_3\rangle = \frac{1}{\sqrt{2}^1}(+|010\rangle - |011\rangle)$$

### 0.3.3 For $f(|11\rangle) = f(|00\rangle) = |1\rangle$

2. apply oracle

[13]:
```
psi2 = np.dot (qm.oox, np.dot (qm.ccx, psi1))
qm.ket3 (psi2, "\\psi_2", raw=False)
```

$$|\psi_2\rangle = \frac{1}{\sqrt{2}^3}(-|000\rangle + |001\rangle + |010\rangle - |011\rangle + |100\rangle - |101\rangle - |110\rangle + |111\rangle)$$

3. apply $H^{\otimes 2}$ to the data qubits

[14]:
```
psi3 = np.dot (qm.kron3 (qm.h1, qm.h1, qm.i1), psi2)
qm.ket3 (psi3, "\\psi_3", raw=False)
```

$$|\psi_3\rangle = \frac{1}{\sqrt{2}^1}(-|110\rangle + |111\rangle)$$

**... 3-Qubit Matrix Work (Next Page)**

# 3qbgates

October 10, 2021

## 0.1 3-Qubit Gate Matrix Works



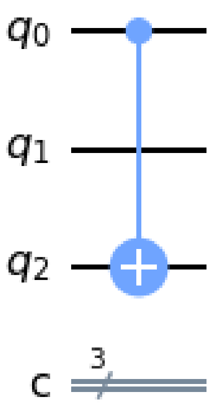
[1]:
```
from importlib import reload
import numpy as np
from IPython.display import display, Math
import qmtx as qm
import ezqc
reload (qm)
reload (ezqc)
```

[1]: `<module 'ezqc' from '/Volumes/extra0/elton4k12/spqc/ezqc.py'>`

What are the matrices for the following 3-qubit gate
($q_i = \alpha_i|0\rangle + \beta_i|1\rangle$)

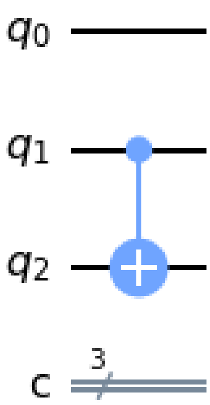
### 0.1.1 Gate 1

[2]:
```
cix = ezqc.EzQC (3)
cix.cx (0, 2)
cix.draw (output="mpl"); # semicolon to workaround draw twice bug
```



For the circuit below,

[3]:
```
icx = ezqc.EzQC (3)
icx.cx (1, 2)
icx.draw ("mpl");
```




the corresponding matrix follows ...

[4]:
```
icxm = np.kron (qm.i1, qm.cx)
qm.dumpgates ([icxm])
```

$$\begin{pmatrix} +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 \\ 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 \end{pmatrix} \begin{pmatrix} \alpha_0\alpha_1\alpha_2 \\ \alpha_0\alpha_1\beta_2 \\ \alpha_0\beta_1\alpha_2 \\ \alpha_0\beta_1\beta_2 \\ \beta_0\alpha_1\alpha_2 \\ \beta_0\alpha_1\beta_2 \\ \beta_0\beta_1\alpha_2 \\ \beta_0\beta_1\beta_2 \end{pmatrix}$$

Switching $q_0$ and $q_1$:

[5]:
```
# (this in markdown cell failed conversion to pdf)
display (Math (r"\begin{pmatrix} 000 \\ 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110␣
↪\\ 111 \end{pmatrix} \
                 \begin{pmatrix} 000 \\ 001 \\ 100 \\ 101 \\ 010 \\ 011 \\ 110␣
↪\\ 111 \end{pmatrix}"))
```

$$\begin{pmatrix} 000 \\ 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{pmatrix} \begin{pmatrix} 000 \\ 001 \\ 100 \\ 101 \\ 010 \\ 011 \\ 110 \\ 111 \end{pmatrix}$$

Effectively rows 3 and 5 switch and rows 4 and 6 switch respectively.
So the matrix for qc is qc0 with rows ~~columns~~ and 3 and 5 switch and row ~~columns~~ 4 and 6 switch respectively. (striked after verified with Qiskit)

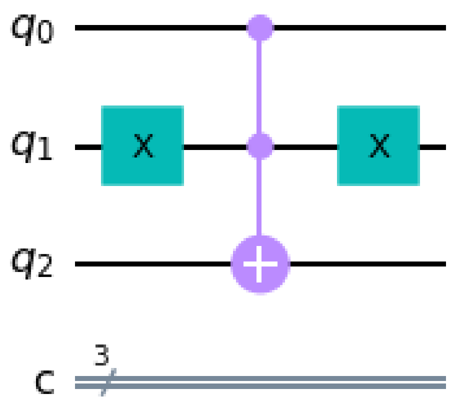
So the matrix for cix is ...

[6]:
```
cixm = icxm.copy()
cixm [[2, 4], :] = cixm [[4, 2], :]
cixm [[3, 5], :] = cixm [[5, 3], :]
qm.dumpgates ([cixm])
```

$$\begin{pmatrix} +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 \\ 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 \end{pmatrix} \begin{pmatrix} \alpha_0\alpha_1\alpha_2 \\ \alpha_0\alpha_1\beta_2 \\ \alpha_0\beta_1\alpha_2 \\ \alpha_0\beta_1\beta_2 \\ \beta_0\alpha_1\alpha_2 \\ \beta_0\alpha_1\beta_2 \\ \beta_0\beta_1\alpha_2 \\ \beta_0\beta_1\beta_2 \end{pmatrix}$$

### 0.1.2 Gate 2: cox

[7]:
```
cox = ezqc.EzQC (3)
cox.x (1)
cox.ccx (0, 1, 2)
cox.x (1)
cox.draw ("mpl");
```




[8]:
```
ixim = qm.kron3 (qm.i1, qm.x1, qm.i1)
qm.dumpgates ([ixim, qm.ccx, ixim])
```

$$\begin{pmatrix} 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 \\ +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 \\ 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 \end{pmatrix} \begin{pmatrix} +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 \\ 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & +1 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 \\ +1 & 0 & 0 & 0 & 0 \\ 0 & +1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & +1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

[9]:
```
front2m = np.dot (ixim, qm.ccx)
qm.dumpgates ([front2m, ixim])
```

$$\begin{pmatrix} 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 \\ +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 \\ 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 \\ +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 \\ 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \alpha_0\alpha_1\alpha_2 \\ \alpha_0\alpha_1\beta_2 \\ \alpha_0\beta_1\alpha_2 \\ \alpha_0\beta_1\beta_2 \\ \beta_0\alpha_1\alpha_2 \\ \beta_0\alpha_1\beta_2 \\ \beta_0\beta_1\alpha_2 \\ \beta_0\beta_1\beta_2 \end{pmatrix}$$

cox

[10]:
```
coxm = np.dot (front2m, ixim)
qm.dumpgates ([coxm])
```

$$\begin{pmatrix} +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 \\ 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 \end{pmatrix} \begin{pmatrix} \alpha_0\alpha_1\alpha_2 \\ \alpha_0\alpha_1\beta_2 \\ \alpha_0\beta_1\alpha_2 \\ \alpha_0\beta_1\beta_2 \\ \beta_0\alpha_1\alpha_2 \\ \beta_0\alpha_1\beta_2 \\ \beta_0\beta_1\alpha_2 \\ \beta_0\beta_1\beta_2 \end{pmatrix}$$

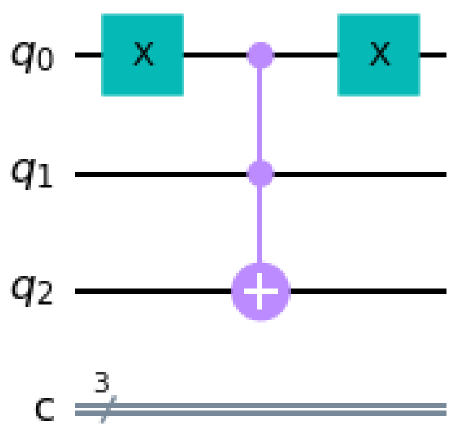
compare ccx

[11]:
```
qm.dumpgates ([qm.ccx])
```

$$\begin{pmatrix} +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 \\ 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 \end{pmatrix} \begin{pmatrix} \alpha_0\alpha_1\alpha_2 \\ \alpha_0\alpha_1\beta_2 \\ \alpha_0\beta_1\alpha_2 \\ \alpha_0\beta_1\beta_2 \\ \beta_0\alpha_1\alpha_2 \\ \beta_0\alpha_1\beta_2 \\ \beta_0\beta_1\alpha_2 \\ \beta_0\beta_1\beta_2 \end{pmatrix}$$

### 0.1.3 Gate 3: ocx

[12]:
```
cox = ezqc.EzQC (3)
cox.x (0)
cox.ccx (0, 1, 2)
cox.x (0)
cox.draw ("mpl");
```

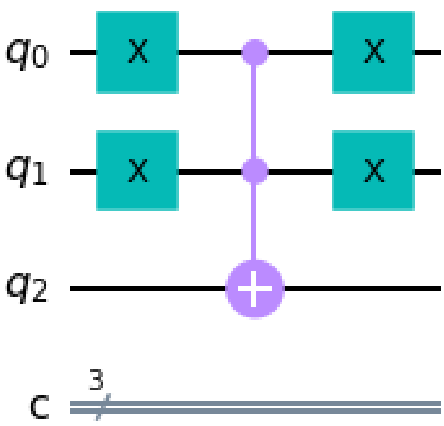


[13]:
```
xiim = qm.kron3 (qm.x1, qm.i1, qm.i1)
front2m = np.dot (xiim, qm.ccx)
qm.dumpgates ([np.dot (front2m, xiim)])
```

$$\begin{pmatrix} +1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 \end{pmatrix} \begin{pmatrix} \alpha_0\alpha_1\alpha_2 \\ \alpha_0\alpha_1\beta_2 \\ \alpha_0\beta_1\alpha_2 \\ \alpha_0\beta_1\beta_2 \\ \beta_0\alpha_1\alpha_2 \\ \beta_0\alpha_1\beta_2 \\ \beta_0\beta_1\alpha_2 \\ \beta_0\beta_1\beta_2 \end{pmatrix}$$

### 0.1.4 Gate 4: oox

[14]:
```
cox = ezqc.EzQC (3)
cox.x (0)
cox.x (1)
cox.ccx (0, 1, 2)
cox.x (0)
cox.x (1)
cox.draw ("mpl");
```




[15]:
```
xxim = qm.kron3 (qm.x1, qm.x1, qm.i1)
front2m = np.dot (xxim, qm.ccx)
qm.dumpgates ([np.dot (front2m, xxim)])
```

$$
\begin{pmatrix}
0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 \\
+1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & +1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & +1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & +1
\end{pmatrix}
\begin{pmatrix}
\alpha_0\alpha_1\alpha_2 \\
\alpha_0\alpha_1\beta_2 \\
\alpha_0\beta_1\alpha_2 \\
\alpha_0\beta_1\beta_2 \\
\beta_0\alpha_1\alpha_2 \\
\beta_0\alpha_1\beta_2 \\
\beta_0\beta_1\alpha_2 \\
\beta_0\beta_1\beta_2
\end{pmatrix}
$$

**... Implementation of 2-Qubit Deutsch-Josza with Qiskit (Next Page)**

# dj2qskt

October 10, 2021

Qiskit 實作：Deutsch-Josza
2021 (CC BY-NC-SA 4.0) Elton Huang

[1]:
```
from ezqc import *

# 定義 the Deutsch's Algorithm

def dj2b (uf, shots = 1): # returns True if balanced
    qc = EzQC (3)
    qc.set_qubits ([(0, 0), (1, 0), (2, 1)], bb = False)
    for i in range (3):
        qc.h (i)
    qc.barrier ()
    uf (qc)
    qc.barrier ()
    qc.h (0)
    qc.h (1)
    qc.measure (0, 0)
    qc.measure (1, 1)
    if shots == 1:
        runres = sorted (run_sim (qc, shots = shots).items(), key=lambda item:␣
↪item[1])
    else:
        runres = sorted (run_real (qc, shots = shots).items(), key=lambda item:␣
↪item[1])
        # 跑數次取多數決，因為會有雜訊所造成的誤差了
        # 會回傳: {'0xx': 1} (results of lower qubit indices, top of circuit␣
↪plots, on the right)
        # q2 is of value ket-0 due to no measurement (Todo: verify, done -␣
↪measure 1)
        print (runres, "\n")
    return (False if runres [-1][0] == '000' else True)
```

| name 名稱 | function 函數 | prpperty 性質 |
|---|---|---|
| uf2c0 | $f(x^{\otimes 2}) = \|0\rangle$ | constant |
| uf2c1 | $f(x^{\otimes 2}) = \|1\rangle$ | constnat |
| uf2b101 | $f(\|11\rangle) = f(\|10\rangle) = \|1\rangle$ | balanced |
| uf2b101s | $f(\|11\rangle) = f(\|10\rangle) = \|1\rangle$ | balanced |
| uf2b011 | $f(\|11\rangle) = f(\|01\rangle) = \|1\rangle$ | balanced |
| uf2b011s | $f(\|11\rangle) = f(\|01\rangle) = \|1\rangle$ | balanced |
| uf2b001 | $f(\|11\rangle) = f(\|00\rangle) = \|1\rangle$ | balanced |
| uf2b100 | $f(\|11\rangle) = f(\|10\rangle) = \|0\rangle$ | balanced |
| uf2b100s | $f(\|11\rangle) = f(\|10\rangle) = \|0\rangle$ | balanced |
| uf2b010 | $f(\|11\rangle) = f(\|01\rangle) = \|0\rangle$ | balanced |
| uf2b010s | $f(\|11\rangle) = f(\|01\rangle) = \|0\rangle$ | balanced |
| uf2b000 | $f(\|11\rangle) = f(\|00\rangle) = \|0\rangle$ | balanced |

[2]:

```
# 定義 1 qubit 的 4 種 Uf (oracles)

def uf2c0 (qc): # constant
    pass

def uf2c1 (qc): # constant
    qc.x (2)

def uf2b101 (qc): # balanced
    qc.ccx (0, 1, 2)
    qc.x (1)
    qc.ccx (0, 1, 2)
    qc.x (1)

def uf2b100 (qc): # balanced
    uf2b101 (qc)
    qc.x (2)

def uf2b011 (qc): # balanced
    qc.ccx (0, 1, 2)
    qc.x (0)
    qc.ccx (0, 1, 2)
    qc.x (0)

def uf2b010 (qc): # balanced
    uf2b011 (qc)
    qc.x (2)

def uf2b101s (qc): # balanced
    qc.cx (0, 2)

def uf2b100s (qc): # balanced
```

```python
    uf2b101 (qc)
    qc.x (2)

def uf2b011s (qc): # balanced
    qc.cx (1, 2)

def uf2b010s (qc): # balanced
    uf2b011 (qc)
    qc.x (2)

def uf2b000 (qc): # balanced
    qc.ccx (0, 1, 2)
    qc.x (0)
    qc.x (1)
    qc.ccx (0, 1, 2)
    qc.x (0)
    qc.x (1)

def uf2b001 (qc): # balanced
    uf2b000 (qc)
    qc.x (2)
```

[3]:
```python
from random import shuffle

# 將 4 個 Uf 存到一個 list 中，並註記名稱，以及預期結果 (True 為 balanced)
uf2s = [ ('uf2c0', uf2c0, False), ('uf2c1', uf2c1, False),
         ('uf2b101', uf2b101, True), ('uf2b100', uf2b100, True),
         ('uf2b011', uf2b011, True), ('uf2b010', uf2b010, True),
         ('uf2b101s', uf2b101s, True), ('uf2b100s', uf2b100s, True),
         ('uf2b011s', uf2b011s, True), ('uf2b010s', uf2b010s, True),
         ('uf2b000', uf2b000, True), ('uf2b001', uf2b001, True) ]

# 先用模擬驗證
for name, orcl, expb in uf2s:
    print (name + ": " + ("Correct" if dj2b (orcl) == expb else "INCORRECT"))
```

```
uf2c0: Correct
uf2c1: Correct
uf2b101: Correct
uf2b100: Correct
uf2b011: Correct
uf2b010: Correct
uf2b101s: Correct
uf2b100s: Correct
uf2b011s: Correct
uf2b010s: Correct
uf2b000: Correct
uf2b001: Correct
```

[4]:
```python
import sympy as sp

# 8 個 Uf 隨機排序
shuffle (uf2s)

# 看看今天跑到幾個 shots 才能全對
sh = 3
while True:
    print ("shots: " + str (sh) + "\n")
    incorrect = 0
    for name, orcl, expb in uf2s:
        print (name)
        if dj2b (orcl, shots = sh) != expb:
            incorrect += 1
            break
    # print (sh, "shots: 錯", incorrect, "個")
    if incorrect == 0:
        print (sh, "shots 完成")
        break
    else:
        sh = sp.nextprime (sh)
```

```
shots: 3

uf2b100s
The best backend is ibmq_belem 5 qubit(s)
Job Status: job has successfully run
[('000', 1), ('001', 2)]

uf2b000
The best backend is ibmq_belem 5 qubit(s)
Job Status: job has successfully run
[('011', 3)]

uf2b011s
The best backend is ibmq_belem 5 qubit(s)
Job Status: job has successfully run
[('010', 3)]

uf2b011
The best backend is ibmq_belem 5 qubit(s)
Job Status: job has successfully run
[('001', 1), ('010', 1), ('011', 1)]

uf2c1
The best backend is ibmq_belem 5 qubit(s)
Job Status: job has successfully run
[('000', 3)]
```

```
uf2b010s
The best backend is ibmq_belem 5 qubit(s)
Job Status: job has successfully run
[('000', 1), ('010', 2)]

uf2b001
The best backend is ibmq_lima 5 qubit(s)
Job Status: job has successfully run
[('011', 3)]

uf2c0
The best backend is ibmq_belem 5 qubit(s)
Job Status: job has successfully run
[('000', 3)]

uf2b100
The best backend is ibmq_belem 5 qubit(s)
Job Status: job has successfully run
[('010', 1), ('000', 2)]

shots: 5

uf2b100s
The best backend is ibmq_belem 5 qubit(s)
Job Status: job has successfully run
[('000', 1), ('001', 4)]

uf2b000
The best backend is ibmq_belem 5 qubit(s)
Job Status: job has successfully run
[('000', 1), ('001', 1), ('011', 3)]

uf2b011s
The best backend is ibmq_belem 5 qubit(s)
Job Status: job has successfully run
[('010', 5)]

uf2b011
The best backend is ibmq_belem 5 qubit(s)
Job Status: job has successfully run
[('011', 1), ('001', 2), ('010', 2)]

uf2c1
The best backend is ibmq_belem 5 qubit(s)
Job Status: job has successfully run
[('000', 5)]
```

```
uf2b010s
The best backend is ibmq_belem 5 qubit(s)
Job Status: job has successfully run
[('001', 1), ('010', 2), ('011', 2)]

uf2b001
The best backend is ibmq_belem 5 qubit(s)
Job Status: job has successfully run
[('000', 1), ('010', 2), ('011', 2)]

uf2c0
The best backend is ibmq_belem 5 qubit(s)
Job Status: job has successfully run
[('000', 5)]

uf2b100
The best backend is ibmq_belem 5 qubit(s)
Job Status: job has successfully run
[('011', 2), ('001', 3)]

uf2b101
The best backend is ibmq_belem 5 qubit(s)
Job Status: job has successfully run
[('000', 1), ('010', 1), ('011', 1), ('001', 2)]

uf2b010
The best backend is ibmq_belem 5 qubit(s)
Job Status: job has successfully run
[('010', 2), ('011', 3)]

uf2b101s
The best backend is ibmq_belem 5 qubit(s)
Job Status: job has successfully run
[('001', 5)]

5 shots 完成
```

## 2.2 More Oracle-Based

### 2.2.1 Berstein-Vazirani

### 2.2.2 Simon

## 2.3 Grover's

- Why is it phase?
- How does it work?

*Quantum Computing for Everyone* Page 179:

- flips about the mean
- CZ, the matrix reversable operations: controlling and controlled no-differently entangled

## 2.4 Hamiltonian simulation and Trotterization

"It's widely *believed* that the Hamiltonian simulation problem can be solved with an exponential number of gates on a classical computer while requiring only a polynomial number on a quantum computer."

$$e^{-i(\sum_k H_k)t} = lim_{N\to\infty}(\prod_k e^{-iH_k t/N})^N \text{ ... ???}$$

$$e^{-iHt} = (\prod_{k=0}^{d-1} e^{-iHkt/r})^r + 0(some\ polynomial\ factors) \text{ ... ???}$$

## 2.5 Exercise: Superposition

$$|1\rangle + |2\rangle + |3\rangle + |7\rangle \;-\!\!\!-\boxed{f(x) = 2x}-\!\!\!-\; |2\rangle + |4\rangle + |6\rangle + |14\rangle$$

$$\tfrac{1}{2}(|0001\rangle + |0010\rangle + |0011\rangle + |0111\rangle) \;-\!\!\!-\boxed{f(x) = 2x}-\!\!\!-\; \tfrac{1}{2}(|0010\rangle + |0100\rangle + |0110\rangle + |1110\rangle)$$

Lee et. al, *Quantum Shift Register*, 2001

## 2.6 Quantum Circuit Synthesis

Shende et. al, *Synthesis of Quantum Logic Circuits*

Page 2 bottom

Page 5 top §2.1 end: $2^3 = 8 \gg 2^1 + 2^2 = 5$

"Much interest in quantum computing is driven by this exponential scaling of the state space, and the loss of independence between different subsystems is called *quantum entanglement*."

Matteo, *Parallelizing quantum circuit synthesis*, 2015

# Chapter 3

# Applications[1]

IBM Quantum Challenge Africa 2021

IBM Quantum Challenge 2021

## 3.1 Optimization

### 3.1.1 Quadratic Problems with Qiskit

### 3.1.2 Knapsack with Ocean

## 3.2 Monte Carlo Method

Monte Carlo Method

[1]... to appreciate the values

## 3.3 Machine Learning

Quantum's advantage solves black box bit riddle

Demonstration of quantum advantage in machine learning

### 3.3.1 Qiskit ML

IBM Quantum Challenge Fall 2021 Challenge 3c

- Multi-class classifiers with binary classifiers

  One-vs-Rest: one classifier for each class: $N$ classifiers (vs. binary tree: $N-1$ classifiers)

  One-vs-One: $\frac{N(N-1)}{2}$ pairs

### 3.3.2 TensorFlow Quantum

## 3.4 Chemical Simulation

"'every finitely realizable physical system can be perfectly simulated by a universal model computingmachine operating by finite means'. Classical physics and the universal Turing machine, because the former is continuous and the latter discrete, do not obey the principle, at least in the strong form above." - D. Deutsch 1985

Exploring greener approaches to nitrogen fixation: 103 electrons in 71 orbitals
Protein crystallization

- sequence → predict shapes
- shape analysis
- hybrid: e.g. 高能物理的方式預測 interactions

### 3.4.1 Molecular Dynamics with Qiskit

### 3.4.2 Published Works

***On the construction of model Hamiltonians for adiabatic quantum computation and its application to finding low energy conformations of lattice protein models*, 2008**

***Finding low-energy conformations of lattice protein models by quantum annealing*, 2012**

***Quantum annealing versus classical machine learning applied to a simplified computational biology problem*, 2018**

***Quantum Molecular Unfolding*, Mente AI, 2021**

- shape, reactivity: shape → reactivity → energy cost
- approach assumption: pocket rigid, ligand flexible
- fixed length chemical bonds with a subset *rotatable* (torsionals) whcih split the molecule in two nonempty disjointed fragments, when virtually removed
- 3 phases in Molecular Docking: *(1) Ligand expansion,* (2) Initial Placement and (3) Shape Refinement inside the pocket

**Molecular Unfolding** find the unfolded shape of the ligand (torsion configuration) that maximizes the total sum of internal distances between pairs of atoms in the ligand (molecular volume).

**Designing Peptides on a Quantum Computer, 2019**

***Förster resonance energy transfer: Role of diffusion of fluorophore orientation and separation in observed shifts of FRET efficiency*, 2017**

***Efficient quantum simulation of photosynthetic light harvesting*, 2018**

### 3.4.3 OpenFermion

# Chapter 4

# Explore and Exercise

## 4.1 IBM Quantum *Errors*

Noise is one of the utmost problems with Quantum Computer hardware technology presently.

This exercise demonstrates how to check the map of error rates for each qubits in an IBM Qunatum computer with Qiskit and a brief example to verify it.

Further indepth exercise may involve Qiskit-Pulse, the pulse-level programming kit.

**(Next page)**

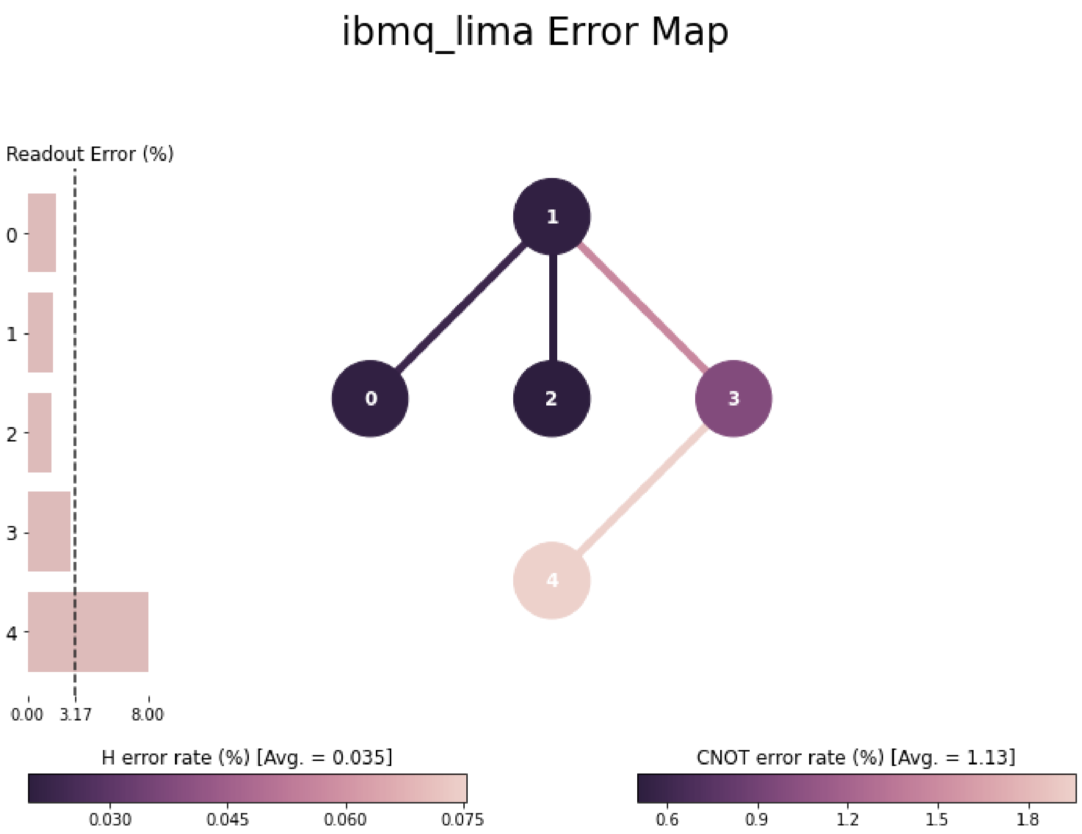
# errors

October 7, 2021

Explore and Exercise: *Errors*


[1]:
```
from qiskit import *
from qiskit.visualization import plot_error_map
IBMQ.load_account() # ezqc.SetCloud()
backend = IBMQ.get_provider('ibm-q').get_backend('ibmq_lima')
plot_error_map(backend)
```

[1]:



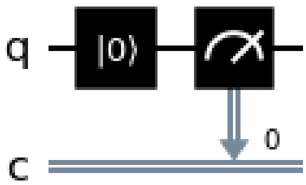

qiskit.compiler.transpile: initial_layout
qiskit.result.Result.get_counts: measured states of the vector of qubits

[2]:
```
from qiskit.tools.monitor import job_monitor

def runiton (qc, res, phqb):
    global backend
    qc_trans = transpile(qc, backend, initial_layout=[phqb],␣
 ↪optimization_level=3)
    job = backend.run(qc_trans, shots=8192)
    # print(job.job_id())
    job_monitor(job)
    output = job.result().get_counts()
    print('Run with qubit #' + str (phqb))
    print('Probability of correct answer : {:.2f}'.format(output[res]/8192))
```

[3]:
```
qc0 = QuantumCircuit(1, 1)
qc0.reset(0)
qc0.measure(0, 0)

qcx = QuantumCircuit(1, 1)
qcx.reset(0)
qcx.x (0)
qcx.measure(0, 0)
```

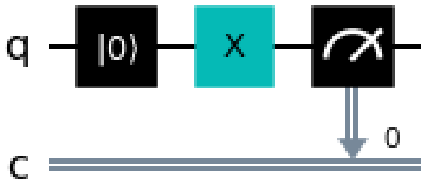
[3]: <qiskit.circuit.instructionset.InstructionSet at 0x7f1760070d00>

[4]:
```
for qc, res in [ (qc0, '0'), (qcx, '1') ]:
    display (qc.draw())
    for i in range (backend.configuration().n_qubits):
        runiton (qc, res, i)
```




```
Job Status: job has successfully run
Run with qubit #0
Probability of correct answer : 0.99
Job Status: job has successfully run
Run with qubit #1
Probability of correct answer : 1.00
```

```
Job Status: job has successfully run
Run with qubit #2
Probability of correct answer : 1.00
Job Status: job has successfully run
Run with qubit #3
Probability of correct answer : 0.99
Job Status: job has successfully run
Run with qubit #4
Probability of correct answer : 0.99
```




```
Job Status: job has successfully run
Run with qubit #0
Probability of correct answer : 0.96
Job Status: job has successfully run
Run with qubit #1
Probability of correct answer : 0.97
Job Status: job has successfully run
Run with qubit #2
Probability of correct answer : 0.97
Job Status: job has successfully run
Run with qubit #3
Probability of correct answer : 0.95
Job Status: job has successfully run
Run with qubit #4
Probability of correct answer : 0.82
```

## 4.2 Virtual Quantum Optics Laboratory

The Virtual Quantum Optics Laboratory

# Appendices

# Appendix A

# ezqc.py

```python
# EzQC Library
# 2021 (CC BY-NC-SA 4.0) Elton Huang

import numpy as np
from qiskit import * # QuantumCircuit, transpile, IBMQ
from qiskit.providers.aer import QasmSimulator
# from qiskit.visualization import plot_histogram
import matplotlib.pyplot as plt

from qiskit.providers.ibmq import least_busy
from qiskit.tools.monitor import job_monitor

bCloudYet2Set = True

def SetCloud ():
    global provider, bCloudYet2Set
    while True:
        try:
            # print ("loading IBMQ account ...")
            provider = IBMQ.load_account()
            break
        except:
            IBMQ.save_account (input ("IBMQ Token:"))
    bCloudYet2Set = False

def run_sim (circuit, shots = 1024):
    simulator = QasmSimulator()
    compiled_circuit = transpile (circuit, simulator)
    counts = simulator.run (compiled_circuit, shots = shots).result().get_counts (compiled_circuit)
    # print("\nResults:",counts)
    return (counts)

def run_real (circuit, shots = 1024): # real hw?
    global bCloudYet2Set
    if 'google.colab' in sys.modules:
        print ("You are on Google Colab. Switching to simulation ...")
        return (run_sim (circuit))
    else:
        if bCloudYet2Set:
            SetCloud ()
        devices = provider.backends (filters = lambda x: x.configuration().n_qubits > circuit.num_qubits
                                            and not x.configuration().simulator
                                            and x.name() not in ['ibmq_bogota'], simulator = False)
        # bogota: borken
        if len (devices) == 0: # not necessary, bm: sim_stab 200s, colab/min 2mins
            devices = provider.backends (filters = lambda x: x.configuration().n_qubits > circuit.num_qubits
```

```
                                                        and x.name() not in ['ibmq_bogota'], simulator = False)
        # print (devices)
        backend = least_busy (devices)
        print ("The best backend is", backend, backend.configuration().n_qubits, "qubit(s)")
        job = execute (circuit, backend = backend, shots = shots)
        job_monitor (job, interval = 5)
        result = job.result()
        counts = result.get_counts(circuit)
        # print("\nResults:",counts)
        return (counts)

def plot (counts):
    fig = plt.figure(figsize = (10, 5)) # default 6.4x4.8 inches
    ax = fig.add_axes([0,0,1,1])
    x = sorted (counts.keys(), key=lambda x: int(x, 2))
    y = list (counts [i] for i in x)
    ax.bar (x, y, width = 0.5, color = 'cornflowerblue')
    ax.set_ylabel('Frequency')
    ax.set_xlabel('Measurement Output')
    for i, v in enumerate (y):
        ax.text(i - .25, v + 3, str (v) + " / " + str (round (100 * v / sum (y))) + "%", color='blue', fontweight='bold')
    plt.show()

class EzQC (QuantumCircuit):

    def __init__ (self, qbn):
        super().__init__ (qbn, qbn)

    # fb = True to add front barrier
    def __fb (self, **kwargs):
        if 'fb' in kwargs.keys() and kwargs.get ('fb') == True:
            self.barrier ()

    # bb = False to omit back barrier
    def __bb (self, **kwargs):
        if 'bb' not in kwargs.keys() or kwargs.get ('bb') != False:
            self.barrier ()

    def set_qubits (self, qubit_ketv_pair_list, **kwargs):
        self.__fb (**kwargs)
        for qb, kv in qubit_ketv_pair_list:
            self.reset (qb)
            if kv == 1:
                self.x (qb)
        self.__bb (**kwargs)

    # reset = False for no reset to ket-0
    def ghz (self, list3qbs, **kwargs):
        self.__fb (**kwargs)
        if 'reset' not in kwargs.keys() or kwargs.get ('reset') != False:
            for i in list3qbs:
                self.reset (i)
        self.h (list3qbs [0])
        self.cx (list3qbs [0], list3qbs [1])
        self.cx (list3qbs [0], list3qbs [2])
        self.__bb (**kwargs)

    def cox (self, q0, q1, q2, **kargs):
        self.__fb (**kwargs)
        self.x (q1)
        self.ccx (q0, q1, q2)
        self.x (q1)
        self.__bb (**kwargs)

```

```python
    def measure_all (self, **kwargs):
        self.__fb (**kwargs)
        self.measure (list (range (self.num_qubits)), list (range (self.num_qubits)))
```

# Appendix B

# qmtx.py

```python
# qmtx Library
# 2021 (CC BY-NC-SA 4.0) Elton Huang

import numpy as np
# from itertools import product
from IPython.display import display, Math
import math

b3string = ['|000\\rangle', '|001\\rangle', '|010\\rangle', '|011\\rangle',
            '|100\\rangle', '|101\\rangle', '|110\\rangle', '|111\\rangle']

b3kets = ['\\ket{000}', '\\ket{001}', '\\ket{010}', '\\ket{011}',
          '\\ket{100}', '\\ket{101}', '\\ket{110}', '\\ket{111}']

opr3 = "\\begin{pmatrix} \\alpha_0\\alpha_1\\alpha_2 \\\\ \\alpha_0\\alpha_1\\beta_2 \\\\ \
                         \\alpha_0\\beta_1 \\alpha_2 \\\\ \\alpha_0\\beta_1 \\beta_2 \\\\ \
                         \\beta_0 \\alpha_1\\alpha_2 \\\\ \\beta_0 \\alpha_1\\beta_2 \\\\ \
                         \\beta_0 \\beta_1 \\alpha_2 \\\\ \\beta_0 \\beta_1 \\beta_2 \\end{pmatrix}"

ket0 = np.array ( [[1],
                   [0]] )

ket1 = np.array ( [[0],
                   [1]] )

h1 = 1/np.sqrt(2)*np.array([[1, 1],
                            [1,-1]])

i1 = np.array ([[1, 0],
                [0, 1]])

x1 = np.array ([[0, 1],
                [1, 0]])

cx = np.array ([[1, 0, 0, 0],
                [0, 1, 0, 0],
                [0, 0, 0, 1],
                [0, 0, 1, 0]])

ccx = np.array ([[1, 0, 0, 0, 0, 0, 0, 0],
                 [0, 1, 0, 0, 0, 0, 0, 0],
                 [0, 0, 1, 0, 0, 0, 0, 0],
                 [0, 0, 0, 1, 0, 0, 0, 0],
                 [0, 0, 0, 0, 1, 0, 0, 0],
                 [0, 0, 0, 0, 0, 1, 0, 0],
                 [0, 0, 0, 0, 0, 0, 0, 1],
```

```
                 [0, 0, 0, 0, 0, 0, 1, 0]])

cox = np.array ([[1, 0, 0, 0, 0, 0, 0, 0],
                 [0, 1, 0, 0, 0, 0, 0, 0],
                 [0, 0, 1, 0, 0, 0, 0, 0],
                 [0, 0, 0, 1, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0, 1, 0, 0],
                 [0, 0, 0, 0, 1, 0, 0, 0],
                 [0, 0, 0, 0, 0, 0, 1, 0],
                 [0, 0, 0, 0, 0, 0, 0, 1]])

ocx = np.array ([[1, 0, 0, 0, 0, 0, 0, 0],
                 [0, 1, 0, 0, 0, 0, 0, 0],
                 [0, 0, 0, 1, 0, 0, 0, 0],
                 [0, 0, 1, 0, 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]])

oox = np.array ([[0, 1, 0, 0, 0, 0, 0, 0],
                 [1, 0, 0, 0, 0, 0, 0, 0],
                 [0, 0, 1, 0, 0, 0, 0, 0],
                 [0, 0, 0, 1, 0, 0, 0, 0],
                 [0, 0, 0, 0, 1, 0, 0, 0],
                 [0, 0, 0, 0, 0, 1, 0, 0],
                 [0, 0, 0, 0, 0, 0, 1, 0],
                 [0, 0, 0, 0, 0, 0, 0, 1]])

icx = np.array ([[1, 0, 0, 0, 0, 0, 0, 0],
                 [0, 1, 0, 0, 0, 0, 0, 0],
                 [0, 0, 0, 1, 0, 0, 0, 0],
                 [0, 0, 1, 0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 1, 0, 0, 0],
                 [0, 0, 0, 0, 0, 1, 0, 0],
                 [0, 0, 0, 0, 0, 0, 0, 1],
                 [0, 0, 0, 0, 0, 0, 1, 0]])

cix = np.array ([[1, 0, 0, 0, 0, 0, 0, 0],
                 [0, 1, 0, 0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 1, 0, 0, 0],
                 [0, 0, 0, 0, 0, 1, 0, 0],
                 [0, 0, 0, 1, 0, 0, 0, 0],
                 [0, 0, 1, 0, 0, 0, 0, 0],
                 [0, 0, 0, 0, 0, 0, 0, 1],
                 [0, 0, 0, 0, 0, 0, 1, 0]])

def kron3 (q0, q1, q2): # 3 qubits tensor product
    return np.kron (np.kron (q0, q1), q2)

def ket3 (psi, var, raw=False): # Todo: ket-n
    lnn = 0
    out = ""
    for r in range (psi.shape[0]):
        if abs (psi [r][0]) > 1e-10: # Todo: complex numbers
            lnn += 1
            out += ('-' if psi [r][0] < 0 else '+') + (b3kets [r] if raw else b3string [r]) # Todo: non-uniform coeffecients
    if raw:
        print (("\ket{" + var.replace('\\\\', '\\') + "} = ") +
               ((""  if lnn == 1 else
                 ("\\frac{1}{\\sqrt{2}^" + str (int (math.log (lnn, 2))) + "}(")) + out +
                ("" if lnn == 1 else ")")).replace('\\\\', '\\'))
    else:
        display (Math(("|" + var + "\\rangle = ") +
```

```
                      ("" if lnn == 1 else
                       ("\\frac{1}{\\sqrt{2}^" + str (int (math.log (lnn, 2))) + "}(")) + out +
                      ("" if lnn == 1 else ")")))

def dump (mtx):
    lnn = 0
    out = "\\begin{pmatrix} "
    for i in range (mtx.shape[0]):
        for j in range (mtx.shape[1]):
            if abs (mtx [i][j]) > 1e-10: # Todo: complex numbers
                lnn += 1
                out += '-1 ' if mtx [i][j] < 0 else '+1 ' # Todo: non-uniform coeffecients
            else:
                out += ' 0 '
            if j + 1 < mtx.shape[1]:
                out += '& '
        out += "\\\\"
    out += "\\end{pmatrix}"
    display(Math(("" if lnn == 1 else
                  ("\\frac{1}{\\sqrt{2}^" + str (int (math.log (lnn, 2))) + "}")) + out))

def dumpgates (mtxs):
    sq2n = 0
    out = ""
    for mtx in mtxs:
        lnn = 0
        out += "\\begin{pmatrix} "
        for i in range (mtx.shape[0]):
            for j in range (mtx.shape[1]):
                if abs (mtx [i][j]) > 1e-10: # Todo: complex numbers
                    lnn += 1
                    out += '-1 ' if mtx [i][j] < 0 else '+1 ' # Todo: non-uniform coeffecients
                else:
                    out += ' 0 '
                if j + 1 < mtx.shape[1]:
                    out += '& '
            out += "\\\\"
        out += "\\end{pmatrix}"
        sq2n += lnn // mtx.shape[0] - 1
    display(Math(("" if sq2n == 0 else
                  ("\\frac{1}{\\sqrt{2}^" + str (sq2n) + "}")) + out + opr3))

# https://jarrodmcclean.com/basic-quantum-circuit-simulation-in-python/
```

# Appendix C

# ⟨量子電腦應用與世界級競賽實務⟩ 勘誤建議

張仁瑀等聯合編著 ⟨量子電腦應用與世界級競賽實務⟩ 應該是目前 (2021 年底) 在這個主題整理得最完整的。以下整理一些我閱讀的過程中覺得有疑問的地方。

- Page 014：最上面圖說 (b) 結尾：
  Stern-Gerlach 實驗是觀察銀原子打在屏幕上的成像，「光線」可能有點誤導？
- Page 018：⟨EPR 實驗⟩：往下第 9 行，疊加 → 糾纏？參考 Page 028 ⟨量子糾纏⟩。
- 再往下 2 行，超聚 → 超距
- Page 019：⟨計算⟩：算出來式 → 算出來是
- Page 033 應該有一兩個錯誤
- Page 107 頁首標題「貳、Ios ...」應為「貳、MacOS ...」
- Page 163 最上面的式子？