Study Notes:

Quantum Computing in Slow Pace

量子計算慢慢來

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

January 1, 2022

Contents

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.

Part I

Quantum Gates

Chapter 1

Basics from Physics to Math

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

- 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

- 【俗說量子】神秘的自旋到底是什麼？何為泡利不相容原理？（第5期）Spin — Linvo說宇宙
- Electrons DO NOT Spin, 2021
- How do we know if electrons are really spinning? 2018

[1] Compton Effect

- *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 Discord 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 Linear Algebra

Linear Algebra for Dummies by Sterling 2009[2], *Introduction to Linear Algebra*, 5ed, Strang 2016

Products of Vector/Matrices

- Inner & outer products — Lecture 5 — Matrix Algebra for Engineers
- If the inner product of vectors $\mathbf{u}$ and $\mathbf{v}$ is equal to 0, i.e. $\mathbf{u}^T\mathbf{v} = \mathbf{0}$, then the vectors are perpendicular (orthogonal). (Chap 2)
- Tensor Prodcut (Direct Product 直積：直接用分配率乘積)

Properties (Chap 3)

- Identity matrix $\mathbf{I}$ (1s along the diagnal), Triangular matrix
- A square matrix is singular if it has a multiplicative inverse; a matrix is non-singular if it does not have a multiplicative inverse.
- Matrix addition is commutative $\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$, and associative $\mathbf{A} + (\mathbf{B} + \mathbf{C}) = (\mathbf{A} + \mathbf{B}) + \mathbf{C}$.
- Matrix multiplication is *not* commutative but is *associative*.
- Distributive: $\mathbf{A} * (\mathbf{B} + \mathbf{C}) = \mathbf{A} * \mathbf{B} + \mathbf{A} * \mathbf{C}$.
- Transposition 轉置
- If $\mathbf{B} * \mathbf{B}^{-1} = \mathbf{I}$, $\mathbf{B}^{-1}$ is the inverse matrix of $\mathbf{B}$.
  Not all matrices are invertible. For example, a matrix with a row of 0s or a column of 0s isn't invertible — it has no inverse; nor are matrices with columns or rows not-independant to each other.
  (*singular*, page 51 bottom, page 224 top, the determinant of which is 0)

Linear Tranformations

- Linear Combinations of Vectors (Chap 5)
  - Linear Combination: $\mathbf{y} = \sum c_i \mathbf{v}_i$. What you want to determine, though, is whether a particular vector is the result of some particular linear combination of a given set of vectors.
  - The set of all linear combinations of this set is called its span.
    That set of linear combinations of the vectors is spanned by the original set of vectors.
  - Vectors in parallel do not span R^n.
- $\mathbf{Ax} = \mathbf{0}$ is a Homogeneous System. (Chap 7)
- Linear Transformations (Chap 8)
  - $T(\sum \mathbf{v}_i) = \sum T(\mathbf{v}_i),\ T(c\mathbf{v}) = cT(\mathbf{v})$
  - If T transforms a given set of vectors to another set of vectors, the original vectors, the set of $\mathbf{v}_i$, for which $T(\mathbf{v}_i) = \mathbf{0}$, are called the *kernel* or *null space* of the set of vectors.
    (T is not the additive identity vector, T_0)

Determinant 行列式 (Chap 10, 11, 12)

- permutation = ordering.
  The permutation of a set of numbers is said to have an *inversion* if a larger digit precedes a smaller one.

[2] Some inconsistency in typeface, particularly with cofactor, may leed to confusion.

For example, consider the permutation 5423. The 5 precedes 4, the 5 precedes 2, the 5 precedes 3, the 4 precedes 2, and the 4 precedes 3.

The *inverse* of a permutation: 25413 $\rightarrow$ 41532, 數字對應的索引序號依序排出. $\mathbf{B} * \mathbf{B}^{-1} = \mathbf{I}$. (Chap 9)

- $det(\mathbf{A}) = |\mathbf{A}| = \sum(-1)^{i+j} a_{ij} |\mathbf{C}_{ij}|$;

  $|\mathbf{C}_{ij}|$, the minor of a determinant [3] $|\mathbf{A}|$ is a smaller determinant created by eliminating a specific row i and column j of the original determinant $|\mathbf{A}|$. [4]

  Cofactors $C_{ij} = (-1)^{i+j} |\mathbf{C}_{ij}|$ are determinants of size $n - l$. (Strang 5.2)
- Square matrices are the only candidates for having a determinant.
- $det(\mathbf{A}) = det(\mathbf{A}^T)$, $det(\mathbf{A}) = \frac{1}{det(\mathbf{A}^{-1})}$

  Interchanging two rows (or columns) of a matrix results in a determinant of the opposite sign.

  Adding the multiple of a row or column to another row or column, the determinant remains the same.

  $det(\mathbf{A} * \mathbf{B}) = det(\mathbf{A}) * det(\mathbf{B})$.
- $adj(\mathbf{A}) = \mathbf{\Gamma}^T = \begin{bmatrix} C_{11} & C_{21} & . & . & . \\ C_{12} & . & & & \\ . & & . & & \\ . & & & . & \\ . & & & & . \end{bmatrix} = det(\mathbf{A}) * \mathbf{A}^{-1}$, if $det(\mathbf{A}) \neq 0$.

  $\mathbf{\Gamma}$ is the matrix of the cofactors of $\mathbf{A}$.

Eigenvalues and Eigenvectors (Chap 16, Strang 6.1)

$\mathbf{Ax} = \lambda \mathbf{x}$,

For $\mathbf{A}$, $\mathbf{x}$: eigenvector, λ: eigenvalue. Multiplying matrix $\mathbf{A}$ by the vector $\mathbf{x}$ gives you the same result that you'd get by just multiplying vector $\mathbf{x}$ by the scalar λ.

- For examples, *any vector representing a non-origin point on the coordinate plane* is an *eigenvector*
  - for the matrix that performs 180-degree rotations with an *eigenvalue* of -1;
  - for matrices that performs the dilations (放大) and contractions (縮小), the respective ratios are the corresponding *eigenvalues.*
- All square matrices have eigenvectors and their related eigenvalues.
- But not all square matrices will have eigenvectors that are real (some involve imaginary numbers).
- When real eigenvalues and eigenvectors exist, the effect of multiplying the eigenvector by the matrix is the same as multiplying that same vector by a scalar.
- An n x n matrix will have n eigenvalues (specifically bases of the spans), which spans of bases, though, may be repeated (for triangular matrices, degenerated).
- If $\mathbf{A}$ is symmetric ($\mathbf{A}^T = \mathbf{A}$),
  - eigenvalues are real numbers,
  - and eigenvectors are perpendicular to each other?

  If $\mathbf{A}$ is antisymmetric ($\mathbf{A}^T = -\mathbf{A}$, Strang Page 122), eigenvalues are pure imaginary, and mixed inbetween.

[3] $\mathbf{C}_{ij}$ is italicized in the book which shouldn't be.

[4] $||V|| = \sqrt{\sum v_i^2}$ is the magnitude of **vector** V. (Chap 2)

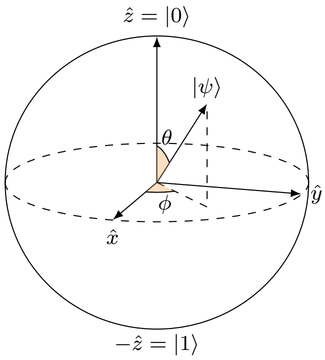
- $\sum \lambda = \sum_i a_{ii}$, the sum of eigenvalues = the sum down the diagnal of the matrix. (the trace)
  $\prod \lambda = det(\mathbf{A})$?
- Adding $c\mathbf{I}$ to the matrix $\rightarrow$ adding c to the eigenvalues with eigenvalues unchanged: $(\mathbf{A}+c\mathbf{I})\mathbf{x} = (\lambda+c)\mathbf{x}$.
  $\mathbf{A}^k\mathbf{x} = \lambda^k\mathbf{x}$?
- For rotations, no vector comes out parallel to itself.

Eigenvectors and eigenvalues — Chapter 14, Essence of linear algebra (3Blue1Brown)

- The (linear) transformation of $\mathbf{A}$ just stretches of squishes $\mathbf{x}$ by λ. ($\mathbf{x}$ remains in it's own span, parallel.)
- $\mathbf{x}$ is the axis of 3-D rotation of $\mathbf{A}$ with $\lambda = 1$.

1.2 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.

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

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

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

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

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

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

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

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

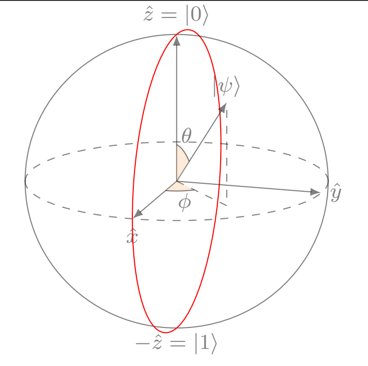
右掌大拇指：軸，四指：$|\psi\rangle$，掌面：旋轉方向

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

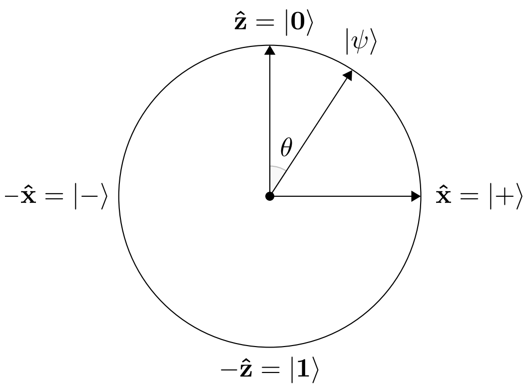
$$Z|1\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \end{pmatrix} = -|1\rangle,\ \theta = -180°.$$

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

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

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

For the sake of simplicity, if we maintain $\phi = 0°$, [5]

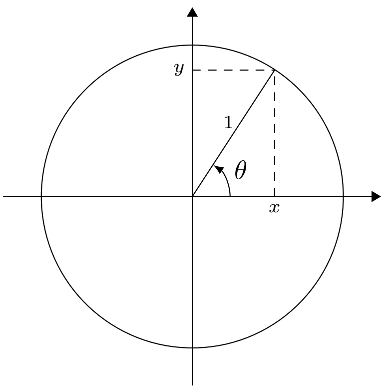


$$\begin{aligned}\lvert\psi\rangle &= \cos(\frac{\theta}{2})\lvert 0\rangle + sin(\frac{\theta}{2})\lvert 1\rangle \\ &= \alpha\lvert 0\rangle + \beta\lvert 1\rangle\end{aligned}$$

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

§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)

$$P(x) = \lvert\, \langle x\vert\psi\rangle\, \rvert^2$$

- Page 121 Clifford + T

[5] Okay up till the Grover's

1.3 More Hardware

Towards Scaling Trapped-ion Quantum Computing

Quantum Annealing

- Webinar: Quantum Computing by a Quantum Annealer
- Quantum annealing with more than one hundred qubits
- D-Wave

Chapter 2

Group I

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)

2.2 CNOT

2.2.1 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,

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

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

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

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

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

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

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

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

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

2.2.2 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,

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

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

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

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

$$q_1q_0 = |\beta\alpha\rangle = |\beta\rangle \otimes |\alpha\rangle = \begin{pmatrix}\frac{3}{5}\times\begin{pmatrix}\frac{\sqrt{3}}{2}\\ \frac{1}{2}\end{pmatrix}\\ \frac{4}{5}\times\begin{pmatrix}\frac{\sqrt{3}}{2}\\ \frac{1}{2}\end{pmatrix}\end{pmatrix} = \begin{pmatrix}\frac{3\sqrt{3}}{10}\\ \frac{3}{10}\\ \frac{4\sqrt{3}}{10}\\ \frac{4}{10}\end{pmatrix}, \text{(note that the squares of all elements sum up to 1)}$$

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

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 Discord 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

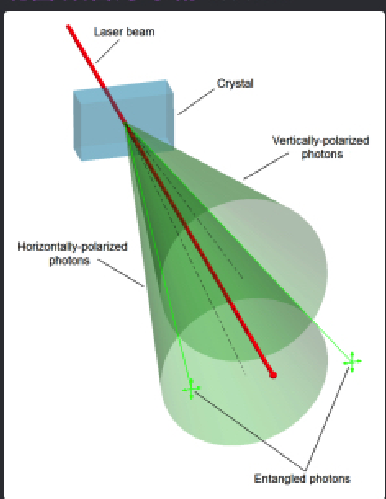
另外不同平台使用的當然很有可能不同，但我個人覺得因為光子交換很 robust，所以 photon bus 還是比較常聽到的做法（純個人看法） (已编辑)

👍 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 entangement 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)

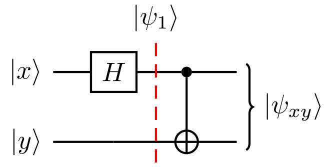
北醫 葉智海 學術 昨天15:32



👍 1

這是製造兩個糾纏光子的方法

2.3 Bell States



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

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

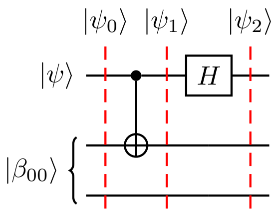
$$|\psi_1\rangle = (H|x\rangle) \otimes |y\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} (x_0+x_1)y_0 \\ (x_0+x_1)y_1 \\ (x_0-x_1)y_0 \\ (x_0-x_1)y_1 \end{pmatrix}$$

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

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

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

2.4 Teleportation



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

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

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

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

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

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

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

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

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

Chapter 3

Group II

3.1 SWAP

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

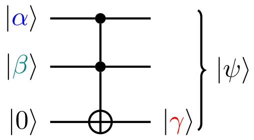
SWAP$|\alpha\beta\rangle = |\beta\alpha\rangle$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_0b_0 \\ a_0b_1 \\ a_1b_0 \\ a_1b_1 \end{pmatrix} = \begin{pmatrix} a_0b_0 \\ a_1b_0 \\ a_0b_1 \\ a_1b_1 \end{pmatrix} = \begin{pmatrix} b_0a_0 \\ b_0a_1 \\ b_1a_0 \\ b_1a_1 \end{pmatrix}$$

3.2 Toffoli (AND/OR/XOR)

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

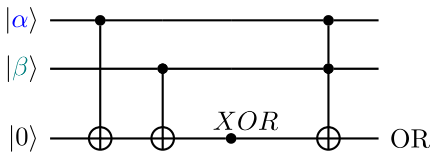
3.2.1 AND



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

$\|\psi\rangle$	$\|\beta\rangle = \|0\rangle$ $\beta_0 = 1, \beta_1 = 0$	$\|\beta\rangle = \|1\rangle$ $\beta_0 = 0, \beta_1 = 1$
$\|\alpha\rangle = \|0\rangle$ $\alpha_0 = 1$ $\alpha_1 = 0$	$\|000\rangle$	$\|010\rangle$
$\|\alpha\rangle = \|1\rangle$ $\alpha_0 = 0$ $\alpha_1 = 1$	$\|100\rangle$	$\|111\rangle$

3.2.2 OR/XOR



Part II

Quantum Algorithms, Exercises and Advanced Topics

Chapter 4

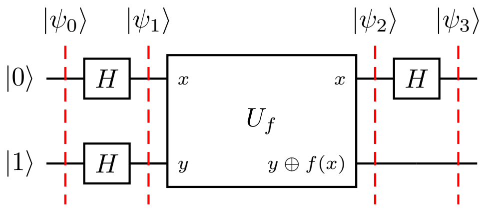
Deutsch-Jozsa

4.1 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 $U_f\,|x, y\rangle = |x, y \otimes f(x)\rangle$.

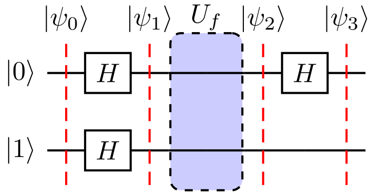
   It can be one of the four circuits below.

2. By looking at the state of 1st qubit of output of the circuit $\underline{(\boldsymbol{H} \otimes \boldsymbol{I})}U_f\underline{(\boldsymbol{H} \otimes \boldsymbol{H})(|\mathbf{0}\rangle \otimes |\mathbf{1}\rangle)}$

   (i.e. $|\psi_3\rangle$ in the diagram above), can determine whether f(x) is constant or balanced. [1]

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

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

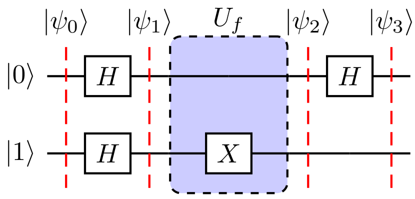


$|\psi_2\rangle = |\psi_1\rangle$

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

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

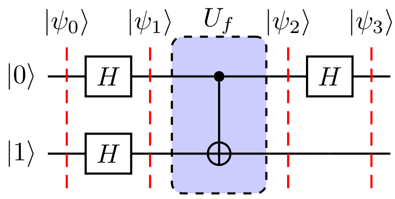
For $f(x) = |1\rangle$,



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

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

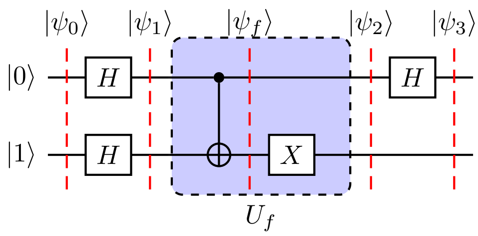
For $f(x) = x$,



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

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

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



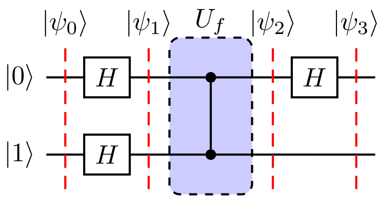
[2] $|q\rangle = \cos\frac{\theta}{2}\,|0\rangle + e^{i\phi}\sin\frac{\theta}{2}\,|1\rangle$: with $-\,|q\rangle$, $\frac{\theta}{2}$ turns 180°, i.e. θ turns 360°.

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

$$|\psi_3\rangle = |1\rangle \otimes (-\,|-\rangle)$$

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



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

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

... 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

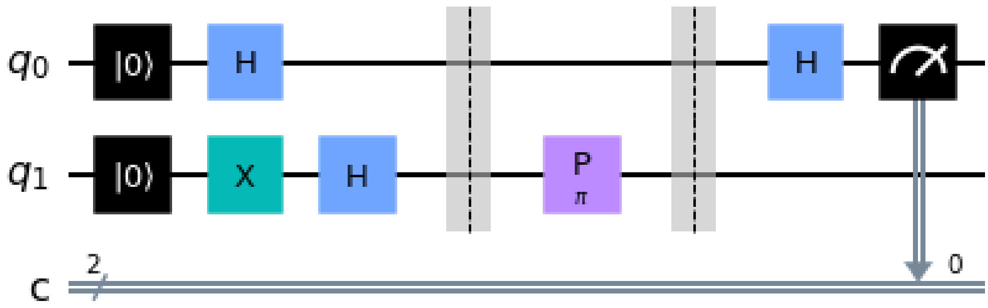
[2]:
```
# 定義 1 qubit 的 4 種 Uf (oracles)

def uf1c00 (qc): # constant
    pass

def uf1c01 (qc): # constant
```

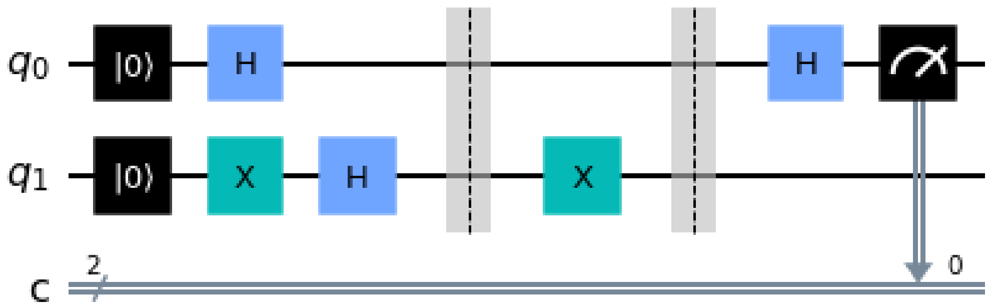
```
    qc.p (pi, 1)

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)
```

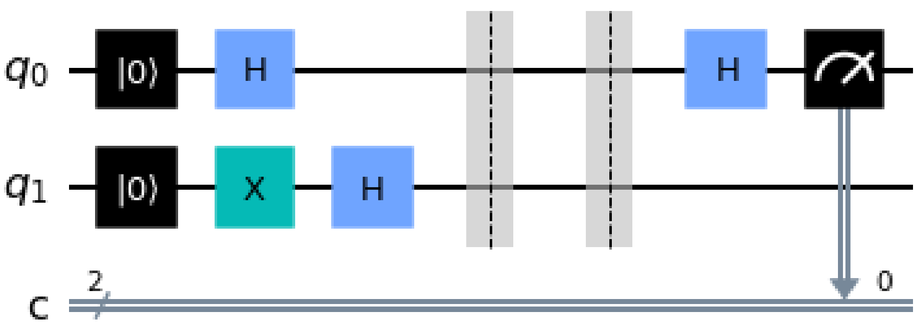
[4]:
```
from random import shuffle

# 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))
```

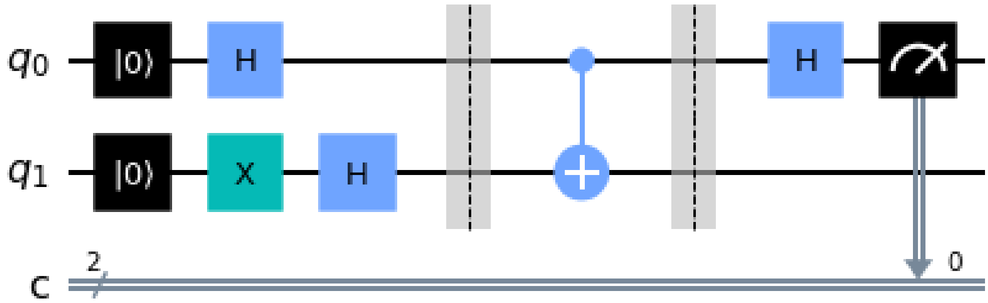
```
The best backend is ibmq_belem 5 qubit(s)
Job Status: job has successfully run
uf1c01: constant

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)
```

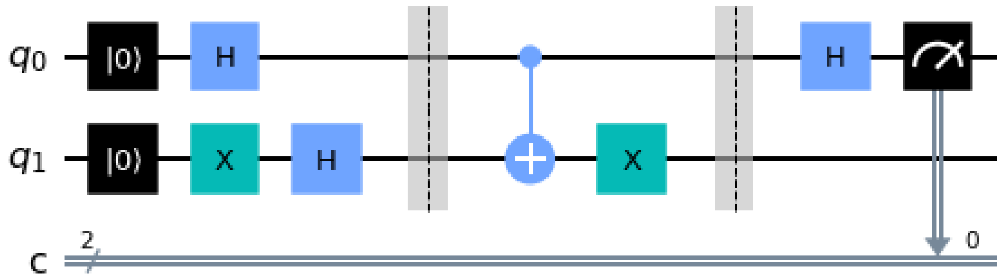
```
Job Status: job has successfully run
uf1b10: balanced
```











0.0.1 Study CZ Orcale

[2]:
```
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]:

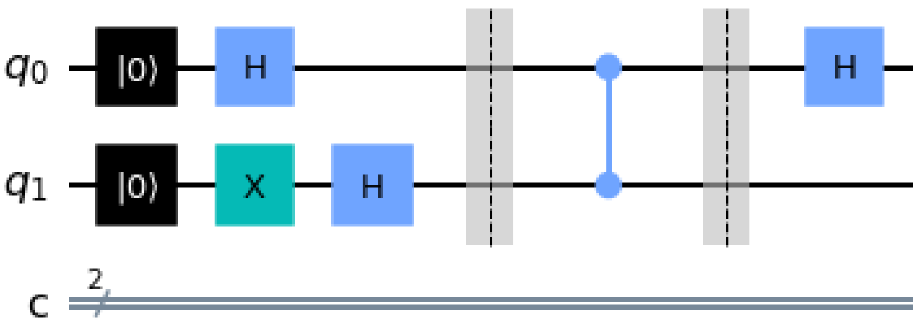
$$|\psi_2\rangle = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

[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");
```



4.2 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

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

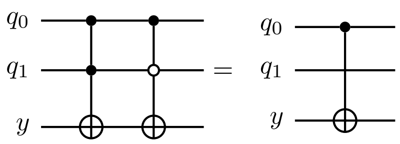
3. apply $H^{\otimes 2}$ to the data qubits

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

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)

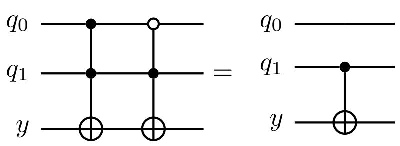
2. apply oracle

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

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

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

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



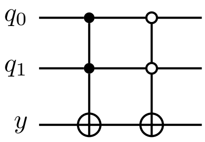
2. apply oracle

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

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

2021 (CC BY-NC-SA 4.0) Elton Huang

[1]:
```python
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]:
```python
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]:
```python
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]:
```python
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

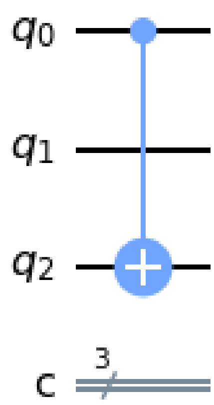
[1]:
```
from importlib import reload
import numpy as np
from IPython.display import display, Math
import qmtx as qm
import ezqc
reload (qm)
reload (ezqc)
```

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

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

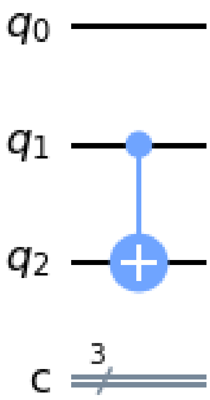
0.1.1 Gate 1

[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)

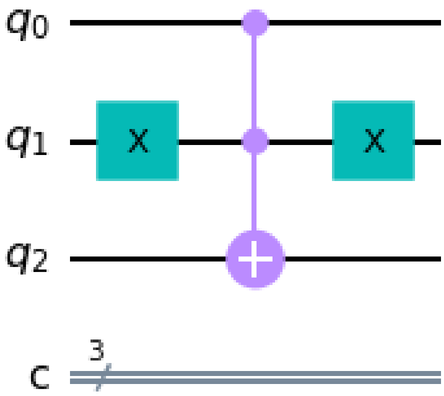
So the matrix for cix is ...

[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])
```

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

[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])
```

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

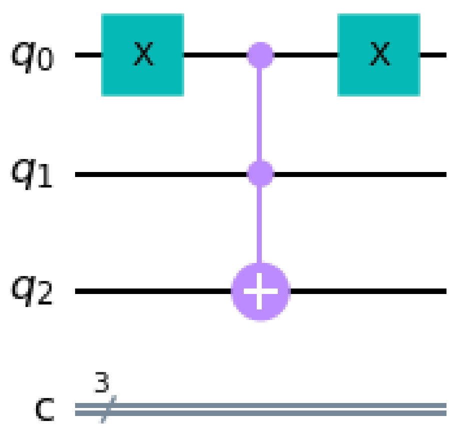
compare ccx

[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");
```

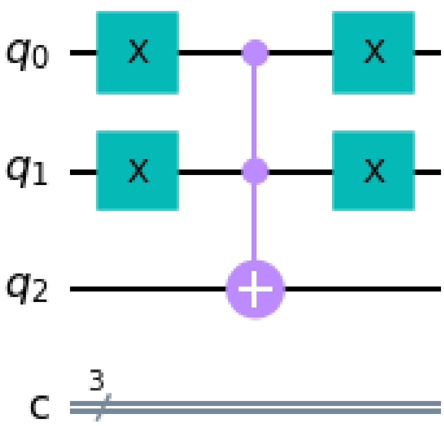


[13]:
```
xiim = qm.kron3 (qm.x1, qm.i1, qm.i1)
front2m = np.dot (xiim, qm.ccx)
qm.dumpgates ([np.dot (front2m, xiim)])
```

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

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)

Qeustion:

Why so many shots with IBM Quantum, in many cases end up spending more time than the classical computers?

Answer:

I believe it's mainly because the qubits may be in superpositions and you need to find out the ratio between ground and excited states. (*Simulating Physics with Computers* by Feynman, 1981, *3. imulating Probability*)

Qeustion:

If the qubits are not in superpositions, can we run with just one shot?

Answer:

I believe so but here comes the secondary merit: the system is noisy so you need more shots to get the proper result as illustrated in the exercise below.

dj2qskt

October 10, 2021

Qiskit 實作：Deutsch-Josza
2021 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
```

```
    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]:

```
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
```

Chapter 5

More Oracle-Based

5.1 Berstein-Vazirani

5.2 Simon

Chapter 6

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

Chapter 7

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."

$$e^{-i(\sum_k H_k)t} = lim_{N\to\infty}(\prod_k e^{-iH_k t/N})^N \text{ ... ???}$$
$$e^{-iHt} = (\prod_{k=0}^{d-1} e^{-iHkt/r})^r + 0(some\ polynomial\ factors) \text{ ... ???}$$

Chapter 8

Exercise: Superposition

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

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

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

Chapter 9

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

Part III

Applications[1]

IBM Quantum Challenge Africa 2021

IBM Quantum Challenge 2021

[1]... to appreciate the values

Chapter 10

Optimization

10.1 Quadratic[1] Problems with Qiskit

10.1.1 IBM Quantum Challenge Africa 2021 Lab 1

- Qiskit documentation for Quadratic Program
- VQE
  - *A variational eigenvalue solver on a photonic quantum processor*, Peruzzo 2014
    - Findng the eigenvalue of certain operators:

      The quantum phase estimation algorithm requires fully coherent evolution.
    - a new approach to state preparation based on
      - ansätze and
      - classical optimization.
  - *The theory of variational* ***hybrid quantum-classical*** *algorithms*, McClean 2016

... modified and simplified problem starting next page ...

[1] Why are quadratic equations called quadratics?

game

December 27, 2021

1 Qiskit Optimizer 範例: 二次函數求最佳解

Adapted from IBM Quantum Challenge Africa 2021 Lab 1
中文敘述部分：2021 (CC BY-NC-SA 4.0) Elton Huang
敘述為强調流程的重點，因而簡化了許多，建議大家還是要一起看原文檔

這個練習示範高階的量子計算使用模式：如何使用`Qiskit Optimization` 應用模組對二次函數問題求最佳解，包括

1. 如何以Qiskit 定義傳統與數學問題,
2. 如何定義量子算法,
3. 如何針對所定義的問題執行量子算法解題,
4. 如何以模擬和雲端實機的方式執行算法解題.

[1]:
```
# 先匯入需用的模組

# Import auxiliary libraries
import numpy as np

# Import Qiskit
from qiskit import IBMQ, Aer
from qiskit.algorithms import QAOA, VQE, NumPyMinimumEigensolver
from qiskit.algorithms.optimizers import COBYLA
from qiskit.utils import QuantumInstance, algorithm_globals
from qiskit.providers.aer.noise.noise_model import NoiseModel

from qiskit_optimization import QuadraticProgram
from qiskit_optimization.algorithms import MinimumEigenOptimizer
from qiskit_optimization.converters import QuadraticProgramToQubo

import qiskit.test.mock as Fake
```

1.1 問題敘述與建模

有一款單機遊戲，對手是電腦，得分完全取決與你組合中角色(包括主教W、巫師S、騎士M、士兵P) 的配置：

1. 主教W 得2 分
2. 巫師S 得1 分

3. 騎士M 得4 分
4. 一位主教W 搭配一位巫師S 得2.4 分
5. 一位主教W 搭配一位騎士M 得4 分
6. 一位主教W 搭配一位士兵P 得4 分
7. 一位巫師S 搭配一位騎士M 得2 分
8. 一位巫師S 搭配一位士兵P 得1 分
9. 一位騎士M 搭配一位士兵P 得5 分

你可以安排的組合有以下限制

1. 總共不能超過3 個人
2. 每種角色不能超過1 位

對於這類的問題，我們可以用一個二次函數的模型來描述，之後就可以操作程式的工具的解題。如果得分是f:

$$f = 2w + s + 4m + 2.4ws + 4wm + 4wp + 2sm + sp + 5mp$$

$$w + s + m + p \leq 3$$
$$0 \leq w \leq 1$$
$$0 \leq s \leq 1$$
$$0 \leq m \leq 1$$
$$0 \leq p \leq 1$$

如果用Qikit Optimization 套件的工具來描述，則如下: (程式中變數名稱請取第一個字母來對應題敘)

[2]:
```
def cropyield_quadratic_program():
    cropyield = QuadraticProgram(name="Crop Yield") # 先給這個問題個名稱
    ##############################
    # Put your implementation here
    cropyield.integer_var(name="Wheat", lowerbound=0, upperbound=1)
    cropyield.integer_var(name="Soybeans", lowerbound=0, upperbound=1)
    cropyield.integer_var(name="Maize", lowerbound=0, upperbound=1)
    cropyield.integer_var(name="PushPull", lowerbound=0, upperbound=1)
    cropyield.maximize(
        linear={"Wheat": 2, "Soybeans": 1, "Maize": 4},
        quadratic={("Wheat", "Soybeans"): 2.4, ("Wheat", "Maize"): 4, ("Wheat",␣
↪"PushPull"): 4, ("Soybeans", "Maize"): 2, ("Soybeans", "PushPull"): 1,␣
↪("Maize", "PushPull"): 5},
    )
    cropyield.linear_constraint(linear={"Wheat": 1, "Soybeans": 1, "Maize": 1,␣
↪"PushPull": 1}, sense="<=", rhs=3)
    #
    ##############################
    return cropyield

cropyield = cropyield_quadratic_program()
```

下面的操作將我們用Qiskit 建立的模型整理出來，可以用來確認模型沒有錯誤

[3]:
```
print(cropyield.export_as_lp_string())
```

```
\ This file has been generated by DOcplex
\ ENCODING=ISO-8859-1
\Problem name: Crop Yield

Maximize
 obj: 2 Wheat + Soybeans + 4 Maize + [ 4.800000000000 Wheat*Soybeans
      + 8 Wheat*Maize + 8 Wheat*PushPull + 4 Soybeans*Maize
      + 2 Soybeans*PushPull + 10 Maize*PushPull ]/2
Subject To
 c0: Wheat + Soybeans + Maize + PushPull <= 3

Bounds
       Wheat <= 1
       Soybeans <= 1
       Maize <= 1
       PushPull <= 1

Generals
 Wheat Soybeans Maize PushPull
End
```

下面的操作估計我們需要幾個qubits 來求解

[4]:
```
# Estimate the number of qubits required
ising_operations, _ = (
    QuadraticProgramToQubo()
    .convert(
        cropyield,
    )
    .to_ising()
)
print(f"Number of qubits required is {ising_operations.num_qubits}")
```

```
Number of qubits required is 6
```

最後再做一個轉換將變數轉成二元變數(細節可以之後再了解)

[5]:
```
QuadraticProgramToQubo().convert(cropyield)
```

[5]:
```
\ This file has been generated by DOcplex
\ ENCODING=ISO-8859-1
\Problem name: Crop Yield

Minimize
 obj: - 160.400000000000 Wheat@0 - 159.400000000000 Soybeans@0
      - 162.400000000000 Maize@0 - 158.400000000000 PushPull@0
```

```
        - 158.400000000000 c0@int_slack@0 - 316.800000000000 c0@int_slack@1 + [
        52.800000000000 Wheat@0^2 + 100.800000000000 Wheat@0*Soybeans@0
        + 97.600000000000 Wheat@0*Maize@0 + 97.600000000000 Wheat@0*PushPull@0
        + 105.600000000000 Wheat@0*c0@int_slack@0
        + 211.200000000000 Wheat@0*c0@int_slack@1 + 52.800000000000 Soybeans@0^2
        + 101.600000000000 Soybeans@0*Maize@0
        + 103.600000000000 Soybeans@0*PushPull@0
        + 105.600000000000 Soybeans@0*c0@int_slack@0
        + 211.200000000000 Soybeans@0*c0@int_slack@1 + 52.800000000000 Maize@0^2
        + 95.600000000000 Maize@0*PushPull@0
        + 105.600000000000 Maize@0*c0@int_slack@0
        + 211.200000000000 Maize@0*c0@int_slack@1 + 52.800000000000 PushPull@0^2
        + 105.600000000000 PushPull@0*c0@int_slack@0
        + 211.200000000000 PushPull@0*c0@int_slack@1
        + 52.800000000000 c0@int_slack@0^2
        + 211.200000000000 c0@int_slack@0*c0@int_slack@1
        + 211.200000000000 c0@int_slack@1^2 ]/2 + 237.600000000000
  Subject To

  Bounds
   0 <= Wheat@0 <= 1
   0 <= Soybeans@0 <= 1
   0 <= Maize@0 <= 1
   0 <= PushPull@0 <= 1
   0 <= c0@int_slack@0 <= 1
   0 <= c0@int_slack@1 <= 1

  Binaries
   Wheat@0 Soybeans@0 Maize@0 PushPull@0 c0@int_slack@0 c0@int_slack@1
  End
```

1.2 本機模擬解題

模型建好後，有以下三種方式用Qikit 程式工具解題：

1. 使用本機模擬器
2. 使用IBM 雲端模擬器
3. 使用IBM Quantum 量子電腦

這三種都稱為backend (後端)，都可以用來解題

1.2.1 傳統的解法

不過使用量子計算解題之前，我們先看看用傳統的計算模式解題的結果

[6]:
```python
def get_classical_solution_for(quadprog: QuadraticProgram):
    # Create solver
    solver = NumPyMinimumEigensolver()
```

```
    # Create optimizer for solver
    optimizer = MinimumEigenOptimizer(solver)

    # Return result from optimizer
    return optimizer.solve(quadprog)

# Get classical result
classical_result = get_classical_solution_for(cropyield)

# Format and print result
print("Solution found using the classical method:\n")
print(f"Maximum crop-yield is {classical_result.fval} tons")
print(f"Crops used are: ")

_crops = [v.name for v in cropyield.variables]
for cropIndex, cropHectares in enumerate(classical_result.x):
    print(f"\t{cropHectares} ha of {_crops[cropIndex]}")
```

```
Solution found using the classical method:

Maximum crop-yield is 19.0 tons
Crops used are:
        1.0 ha of Wheat
        0.0 ha of Soybeans
        1.0 ha of Maize
        1.0 ha of PushPull
```

1.2.2 使用本機模擬器解題

接著用量子計算來解題。第一種方式：使用本機模擬器。

[7]:
```
# We will use the Aer provided QASM simulator
backend = Aer.get_backend("qasm_simulator")

# Given we are using a simulator, we will fix the algorithm seed to ensure our␣
↪results are reproducible
algorithm_globals.random_seed = 271828
```

量子計算**QAOA** 解題**(**本機模擬**)** 接下來有分幾種模式，先用QAOA

[8]:
```
def get_QAOA_solution_for(
    quadprog: QuadraticProgram, quantumInstance: QuantumInstance, optimizer=None,
):
    _eval_count = 0

    def callback(eval_count, parameters, mean, std):
        nonlocal _eval_count
```

```
        _eval_count = eval_count

    # Create solver
    solver = QAOA(
        optimizer=optimizer, quantum_instance=quantumInstance, callback=callback,
    )

    # Create optimizer for solver
    optimizer = MinimumEigenOptimizer(solver)

    # Get result from optimizer
    result = optimizer.solve(quadprog)
    return result, _eval_count

# Create a QuantumInstance
simulator_instance = QuantumInstance(
    backend=backend,
    seed_simulator=algorithm_globals.random_seed,
    seed_transpiler=algorithm_globals.random_seed,
)

# Get QAOA result
qaoa_result, qaoa_eval_count = get_QAOA_solution_for(cropyield,␣
 ↪simulator_instance)

# Format and print result
print("Solution found using the QAOA method:\n")
print(f"Maximum crop-yield is {qaoa_result.fval} tons")
print(f"Crops used are: ")
for cropHectares, cropName in zip(qaoa_result.x, qaoa_result.variable_names):
    print(f"\t{cropHectares} ha of {cropName}")

print(f"\nThe solution was found within {qaoa_eval_count} evaluations of QAOA.")
```

```
Solution found using the QAOA method:

Maximum crop-yield is 19.0 tons
Crops used are:
        1.0 ha of Wheat
        0.0 ha of Soybeans
        1.0 ha of Maize
        1.0 ha of PushPull

The solution was found within 3 evaluations of QAOA.
```

結果和我們用傳統模式的方法所得到的是一樣的

量子計算**VQE 解題(本機模擬)** 量子計算中第二種模式VQE:

[11]:
```python
def get_VQE_solution_for(
    quadprog: QuadraticProgram, quantumInstance: QuantumInstance, optimizer=None,
):
    _eval_count = 0

    def callback(eval_count, parameters, mean, std):
        nonlocal _eval_count
        _eval_count = eval_count

    # Create solver and optimizer
    solver = VQE(
        optimizer=optimizer, quantum_instance=quantumInstance, callback=callback
    )

    # Create optimizer for solver
    optimizer = MinimumEigenOptimizer(solver)

    # Get result from optimizer
    result = optimizer.solve(quadprog)
    return result, _eval_count
```

[ ]:
```python
# Create a QuantumInstance
simulator_instance = QuantumInstance(
    backend=backend,
    seed_simulator=algorithm_globals.random_seed,
    seed_transpiler=algorithm_globals.random_seed,
)

# Get VQE result
vqe_result, vqe_eval_count = get_VQE_solution_for(cropyield, simulator_instance)

# Format and print result
print("Solution found using the VQE method:\n")
print(f"Maximum crop-yield is {vqe_result.fval} tons")
print(f"Crops used are: ")
for cropHectares, cropName in zip(vqe_result.x, vqe_result.variable_names):
    print(f"\t{cropHectares} ha of {cropName}")

print(f"\nThe solution was found within {vqe_eval_count} evaluations of VQE")
```

結果和之前兩個方法所得到的是一樣的

(… 這裡略過模擬真實量子電腦…)

1.3 雲端執行

最上面設定問題時我們說每個角色最多1 人，全部角色最多3 人，現在我們把限制擴大到每個角色最多10 人，全部角色最多也是10 人，這樣需要幾個qubits 呢？

[9]:
```
# Function to estimate the number of qubits required
def estimate_number_of_qubits_required_for(max_hectares_per_crop,␣
↪hectares_available):
    return int(
        4 * np.ceil(np.log2(max_hectares_per_crop + 1))
        + np.ceil(np.log2(hectares_available + 1))
    )


# Our new problem parameters
hectares_available = 10
max_hectares_per_crop = 10

# Retrieving the number of qubits required
number_of_qubits_required = estimate_number_of_qubits_required_for(
    max_hectares_per_crop=max_hectares_per_crop,␣
↪hectares_available=hectares_available
)

print(
    f"Optimizing a {hectares_available} ha farm with each crop taking up to␣
↪{max_hectares_per_crop} ha each,",
    f"the computation is estimated to require {number_of_qubits_required} qubits.
↪",
)
```

```
Optimizing a 10 ha farm with each crop taking up to 10 ha each, the computation
is estimated to require 20 qubits.
```

The number of qubits required is related to the constraints in the quadratic program and how the integer variables are converted to binary variables. In fact, the scaling of the number of qubits, as a function of the hectares available, is logarithmic in nature; owing to this conversion.

接著我們看看如何透過雲端試著用實機來執行
(編按：我看得到雲端實機最多7 quibts 只有一台ibmq_perth，最多是5 quibits，解題加上誤差，連最初的限制都不太夠用)

To use the IBM Quantum platform is easy. First you need to load the account you enabled in the week 0 content. If you didn't complete this, follow this quick guide on connecting your IBM Quantum account with Qiskit in python and Jupyter.

[2]:
```
IBMQ.load_account()
```

[2]:
```
<AccountProvider for IBMQ(hub='ibm-q', group='open', project='main')>
```

IBM Quantum backends are accessed through a provider, which manages the devices to which you have access. For this challenge, you have access to the new `ibm_perth` quantum computer! Typically, you would find your provider details under your IBM Quantum account details. Under your account you can see the different hubs, groups, and projects you are a part of. Qiskit allows us to retrieve a provider using just the hub, group, and project as follows:

```
provider = IBMQ.get_provider(hub="ibm-q", group="open", project="main")
```

However, because we have given you special access for this challenge, we are going to retrive the provider using a different method. Execute the code cell below to retrieve the correct provider.

[3]:
```
provider = None
for prov in IBMQ.providers():
    if (
        "iqc-africa-21" in prov.credentials.hub
        and "q-challenge" in prov.credentials.group
        and "ex1" in prov.credentials.project
    ):
        # Correct provider found
        provider = prov

if provider == None:
    print("ERROR: The expected provider was not found!")
else:
    print("Yay! The expected provider was found!")
```

```
Yay! The expected provider was found!
```

If the above code cell returned an error, you may not yet have access to the real quantum computer. The list of participants is updated daily, so you may have to wait some time before the correct provider appears. If you need assistance, send a message to the challenge Slack channel #challenge-africa-2021 and make sure to tag the admin team with [@africa_admin](#).

To retrieve a backend from the provider, one needs only request it by name. For example, we can request `ibm_perth` as follows.

[4]:
```
backend_real = provider.get_backend("ibm_perth")
```

(… 省略關於雲端實機與模擬選擇的敘述…)

We can create a new `QuantumInstance` object to contain our real quantum computer backend, similar to how we created one to manage our simulator. For real devices, there is an extra parameter we can set: `shots`. The output of a quantum computing algorithm is probabilistic. Therefore, we must execute the quantum computation multiple times, sampling the outputs to estimate their probabilities. The number of `shots` is the number of executions of the quantum computation. Here we set out `QuantumInstance` to use the least busy backend with 2048 shots.

[5]:
```
quantum_instance_real = QuantumInstance(backend_real, shots=2048)
```

The VQE algorithm and QAOA are iterative, meaning that they incorporate a classical-quantum loop which repeats certain computations, *hopefully* converging to a valid solution. In each iteration, or evaluation, the quantum backend will execute the quantum operations 2048 times. Each shot is quite fast, so we do not have to worry about a significant increase in processing time by using more shots.

由於雲端實機免費讓我們練習的quibits 都很少，我們將問題限制縮小到只需要4 qubits 來做示範:

組合中角色只包括主教W 和騎士M：

1. 主教W 得3 分
2. 騎士M 得3 分
3. 一位主教W 搭配一位騎士得1 分

但是如果派相同角色的人上場會扣分
4. 一位主教W 和一位主教W -2 分5. 一位騎士M 和一位騎士M -2 分

用一個二次函數的模型來描述，得分是f:

$f = 3w + 3m + 1wm - 2w^2 - 2m^2$

$0 \leq w \leq 2$
$0 \leq m \leq 2$
$w + m \leq 4$ (上面2 個條件就限制了這條，所以這條毋須另外設定)

如果用Qikit Optimization 套件的工具來描述，則如下: (程式中變數名稱請取第一個字母來對應題敘)

[6]:
```
# Create a small crop-yield example quadratic program
cropyield_small = QuadraticProgram(name="Small Crop-Yield")

# Add two variables, indicating whether we grow 0, 1, or 2 hectares for two␣
↪different crops
cropyield_small.integer_var(lowerbound=0, upperbound=2, name="Wheat")
cropyield_small.integer_var(lowerbound=0, upperbound=2, name="Maize")

# Add the objective function defining the yield in tonnes
cropyield_small.maximize(
    linear={"Wheat": 3, "Maize": 3},
    quadratic={("Maize", "Wheat"): 1, ("Maize", "Maize"): -2, ("Wheat", "Wheat"):
↪ -2},
)

# This linear constraint is not used as the model never reaches this. This is␣
↪because the
# sum of the upperbounds on both variables is 4 already. If this constraint is␣
↪applied, the
# model would require 6 qubits instead of 4.
# cropyield_small.linear_constraint(linear={"Wheat": 1, "Maize": 1}, sense="<=",␣
↪rhs=4)

print(cropyield_small)
```

```
\ This file has been generated by DOcplex
\ ENCODING=ISO-8859-1
\Problem name: Small Crop-Yield
```

```
Maximize
 obj: 3 Wheat + 3 Maize + [ - 4 Wheat^2 + 2 Wheat*Maize - 4 Maize^2 ]/2
Subject To

Bounds
       Wheat <= 2
       Maize <= 2

Generals
 Wheat Maize
End
```

確定一下確實只需要4 quibits ...

[7]:
```python
# Estimate the number of qubits required
ising_operations_small, _ = (
    QuadraticProgramToQubo()
    .convert(
        cropyield_small,
    )
    .to_ising()
)
print(f"Number of qubits required is {ising_operations_small.num_qubits}")
```

```
Number of qubits required is 4
```

接著就可以把問題丟到雲端量子電腦實機解題

[15]:
```python
# Create our optimizer
optimizer = COBYLA(maxiter=1)

## Get result from real device with VQE
vqe_result_real, vqe_eval_count_real = get_VQE_solution_for(
    cropyield_small, quantum_instance_real, optimizer=optimizer
)
```

... (小編補充接續的流程) ...

[17]:
```python
# Format and print result
print("Solution found using the VQE method:\n")
print(f"Maximum crop-yield is {vqe_result_real.fval} tons")
print(f"Crops used are: ")
for cropHectares, cropName in zip(vqe_result_real.x, vqe_result_real.
 ↪variable_names):
    print(f"\t{cropHectares} ha of {cropName}")

print(f"\nThe solution was found within {vqe_eval_count_real} evaluations of VQE.
 ↪")
```

```
job_real = backend_real.jobs()[0]
print ("執行了", job_real.result().results[0].shots, "次用了", \
       round (job_real.result().time_taken), "秒 (每", \
       job_real.result().results[0].shots/256, "次用", \
       "{:.2g}".format(job_real.result().time_taken/256), "秒)")
```

```
Solution found using the VQE method:

Maximum crop-yield is 3.0 tons
Crops used are:
        1.0 ha of Wheat
        1.0 ha of Maize

The solution was found within 1 evaluations of VQE.
執行了 2048 次用了 5 秒 (每 8.0 次用 0.018 秒)
```

和傳統解法結果比較一下

[19]:
```
import time
# Get classical result
start_time = time.time()
classical_result = get_classical_solution_for(cropyield_small)
dt = time.time() - start_time

# Format and print result
print("Solution found using the classical method:\n")
print(f"Maximum crop-yield is {classical_result.fval} tons")
print(f"Crops used are: ")

_crops = [v.name for v in cropyield_small.variables]
for cropIndex, cropHectares in enumerate(classical_result.x):
    print(f"\t{cropHectares} ha of {_crops[cropIndex]}")

print ("執行時間：", "{:.2g}".format(dt), "秒")
```

```
Solution found using the classical method:

Maximum crop-yield is 3.0 tons
Crops used are:
        1.0 ha of Wheat
        1.0 ha of Maize
執行時間： 0.04 秒
```

1.4 下面我們來探究一下：量子電腦解題是不是真的比較快?

因為量子電腦計算的結果需要執行很多次從中統計量測時qubits 量子態塌縮後取得結果。

我們先來算一下用量子電腦用和這台電腦差不多的時間解同樣的問題，最多能夠執行幾次？

[21]:
```
import math
job_real = backend_real.jobs()[0]
# print (job_real.result().time_taken / dt)
mp = math.ceil (math.log (job_real.result().time_taken / dt, 2))
shots = round (job_real.result().results[0].shots / (2**mp))
print (shots)
```

```
16
```

1.4.1 然後以這個次數再執行一次，看結果對不對？

[24]:
```
quantum_instance_real = QuantumInstance(backend_real, shots=shots)
vqe_result_real, vqe_eval_count_real = get_VQE_solution_for(
    cropyield_small, quantum_instance_real, optimizer=optimizer
)
# Format and print result
print("Solution found using the VQE method:\n")
print(f"Maximum crop-yield is {vqe_result_real.fval} tons")
print(f"Crops used are: ")
for cropHectares, cropName in zip(vqe_result_real.x, vqe_result_real.
 ↪variable_names):
    print(f"\t{cropHectares} ha of {cropName}")

print(f"\nThe solution was found within {vqe_eval_count_real} evaluations of VQE.
 ↪")
```

```
Solution found using the VQE method:

Maximum crop-yield is 3.0 tons
Crops used are:
        1.0 ha of Wheat
        1.0 ha of Maize

The solution was found within 1 evaluations of VQE.
```

[26]:
```
print ("工作結果參數：")
job_real = backend_real.jobs()[0]
print ("執行了", job_real.result().results[0].shots, "次用了", \
       round (job_real.result().time_taken), "秒")
```

```
工作結果參數：
執行了 16 次用了 4 秒
```

如果只跑一半的次數...

[ ]:
```
quantum_instance_real = QuantumInstance(backend_real, shots=rounrd (shots/2))
vqe_result_real, vqe_eval_count_real = get_VQE_solution_for(
    cropyield_small, quantum_instance_real, optimizer=optimizer
)
```

```
# Format and print result
print("Solution found using the VQE method:\n")
print(f"Maximum crop-yield is {vqe_result_real.fval} tons")
print(f"Crops used are: ")
for cropHectares, cropName in zip(vqe_result_real.x, vqe_result_real.
 ↪variable_names):
    print(f"\t{cropHectares} ha of {cropName}")

print(f"\nThe solution was found within {vqe_eval_count_real} evaluations of VQE.
 ↪")

print ("工作結果參數：")
job_real = backend_real.jobs()[0]
print ("執行了", job_real.result().results[0].shots, "次用了", \
       round (job_real.result().time_taken), "秒 (每", \
       job_real.result().results[0].shots/256, "次用", \
       "{:.2g}".format(job_real.result().time_taken/256), "秒)")
```

編按：現在我們能試的機器和問題都太小，無法做有意義的效能比較。希望在不久的將來，可以做安兔兔之類的跑分來看看量子計算是不是真的快很多。

1.5 References

[1] A. A. Nel, 'Crop rotation in the summer rainfall area of South Africa', South African Journal of Plant and Soil, vol. 22, no. 4, pp. 274–278, Jan. 2005, doi: 10.1080/02571862.2005.10634721.

[2] H. Ritchie and M. Roser, 'Crop yields', Our World in Data, 2013, [Online]. Available: https://ourworldindata.org/crop-yields.

[3] G. Brion, 'Controlling Pests with Plants: The power of intercropping', UVM Food Feed, Jan. 09, 2014. https://learn.uvm.edu/foodsystemsblog/2014/01/09/controlling-pests-with-plants-the-power-of-intercropping/ (accessed Feb. 15, 2021).

[4] N. O. Ogot, J. O. Pittchar, C. A. O. Midega, and Z. R. Khan, 'Attributes of push-pull technology in enhancing food and nutrition security', African Journal of Agriculture and Food Security, vol. 6, pp. 229–242, Mar. 2018.

[20]:
```
import qiskit.tools.jupyter

%qiskit_version_table
%qiskit_copyright
```

```
/Volumes/extra/opt/anaconda3/lib/python3.8/site-
packages/qiskit/aqua/__init__.py:86: DeprecationWarning: The package qiskit.aqua
is deprecated. It was moved/refactored to qiskit-terra For more information see
<https://github.com/Qiskit/qiskit-aqua/blob/main/README.md#migration-guide>
  warn_package('aqua', 'qiskit-terra')
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

10.2 Knapsack with Ocean

Chapter 11

Monte Carlo Method

Monte Carlo Method

Chapter 12

Machine Learning

Quantum's advantage solves black box bit riddle

Demonstration of quantum advantage in machine learning

12.1 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

12.2 TensorFlow Quantum

Chapter 13

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

13.1 Molecular Dynamics with Qiskit

13.2 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**

13.2.1 Quantum Annealing

***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**

13.3 OpenFermion

Part IV

Explore and Exercise

Chapter 14

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)

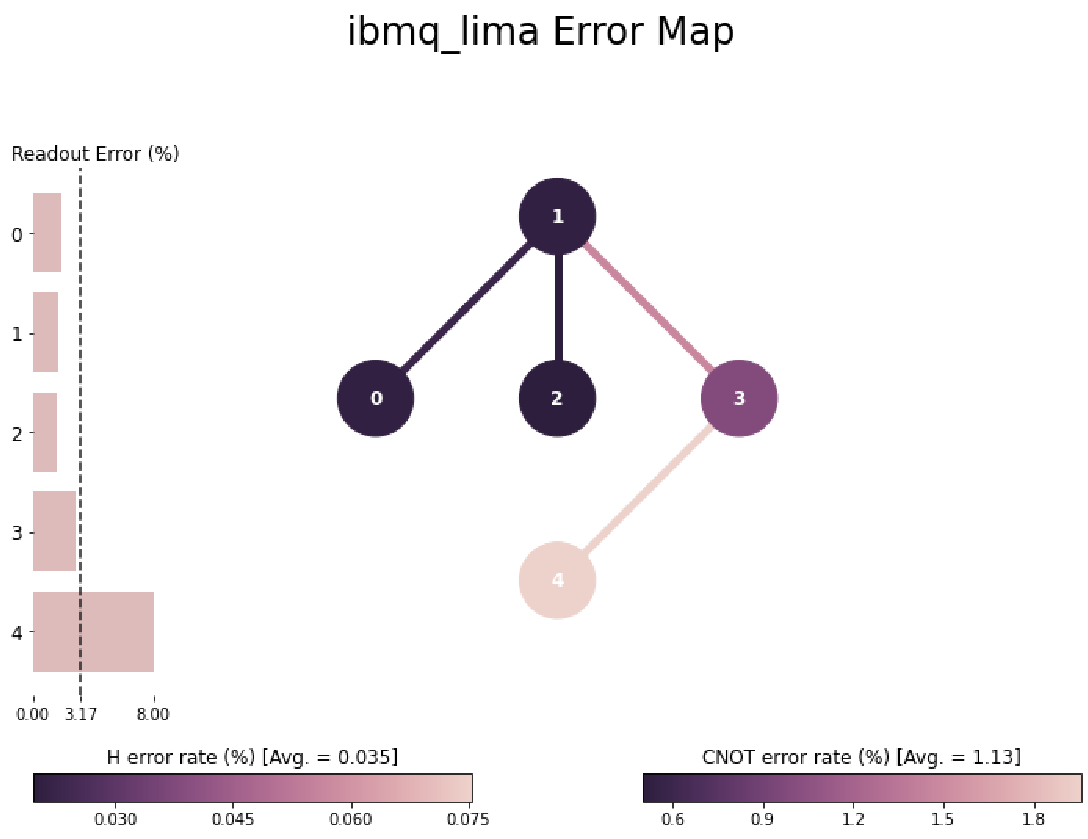
errors

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]:



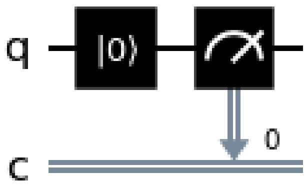
qiskit.compiler.transpile: initial_layout
qiskit.result.Result.get_counts: measured states of the vector of qubits

```
[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)
```

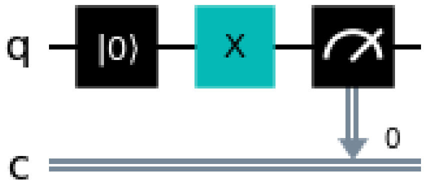
```
[3]: <qiskit.circuit.instructionset.InstructionSet at 0x7f1760070d00>
```

```
[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
```

Chapter 15

Virtual Quantum Optics Laboratory

The Virtual Quantum Optics Laboratory

Appendices

Appendix A

ezqc.py

```
# 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)

```

```
    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？