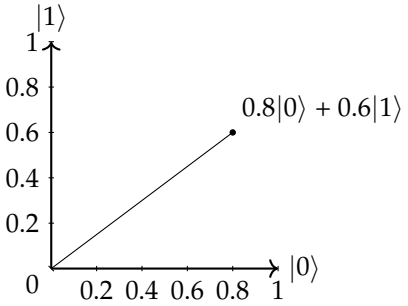
PCHPC Comprehensive Notes

GWDG

1 Basics

Quantum bits, or rather **qubits**, are the quantum version of the classical bits 0 and 1 that we are accustomed to. Briefly put, qubits are unitary vectors in $\mathbb{C}^2$-space known as the **state space** i.e. $[z_1, z_2]^T$, with $z_1, z_2 \in \mathbb{C}$ such that $z_1^2 + z_2^2 = 1$. Due to their origins in quantum physics, z_1 and z_2 are known as **amplitudes**. The vector $[1, 0]^T$ is known as a **(1-qubit state) computational basis**. This vector can also be represented as $|0\rangle$, in what is known as **ket** notation. Similarly, $[0, 1]^T$ is also a computational basis and represented by $|1\rangle$. These computational bases serve as the equivalent to the classical bits i.e. $|0\rangle \sim$ classical 0 and $|1\rangle \sim$ classical 1.

If for a moment we only consider real values for the amplitudes z_1, z_2, we can plot a qubit on a 2D plane with axes $|0\rangle$ and $|1\rangle$ as below. Notice how the below qubit is unitary, as it should be. The state which a qubit is in will be referred to as its **state vector**.



A qubit in **superposition** simply means that the qubit can be expressed as a linear combination of $|0\rangle$ and $|1\rangle$. **Quantum circuits** represent qubits and the computations done to them. A newly initialized quantum circuit with n many qubits, will have all n qubits initialized to $|0\rangle$. Operations on qubits are called **gates** (in analogy to classical logical gates). Since qubits are vectors in $\mathbb{C}^2$, we can represent (1-qubit) gates as matrices in the 'Special Unitary' group $SU(2, \mathbb{C})$, which are invertible and have determinant 1. We need gates to be invertible, also known as **reversible**, so that we can undo any gate applied. If U is a gate, then $U^\dagger$ is its adjoint such that $U^\dagger U = I$. $\dagger$ is called the **dagger operation**.

In qiskit, we can initialise a qubit to arbitrary states, but this is not an action that a quantum computer can actually do. Instead, think of initialization as a "checkpoint" that you skip to, to avoid time-expensive "setup" operations. Note that the imaginary number i is represented in Python as j.

1.1 1-qubit gates

1.1.1 Not gate

The **NOT gate** is $X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ and represents the quantum equivalent to the classical not-gate by swapping the amplitudes. Thus $X|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle$. More generally $X(\alpha|0\rangle + \beta|1\rangle) = \beta|0\rangle + \alpha|1\rangle$.

1.1.2 Hadamard gate

The **Hadamard gate** is $H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ and is especially important towards preparing circuits into certain states for computations e.g. a state in Fourier basis (more on this later). Here,

$$H(\alpha|0\rangle + \beta|1\rangle) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} \alpha + \beta\sqrt{2} \\ \alpha - \beta\sqrt{2} \end{bmatrix} = \frac{\alpha+\beta}{\sqrt{2}}|0\rangle + \frac{\alpha-\beta}{\sqrt{2}}|1\rangle$$

1.1.3 Pauli gates

These gates have representations $X, Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$ and $Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$, and represent a series of special properties. Namely, they allow us to define a series of **orthogonal basis states**. We already know the 2 orthogonal z-basis states, which are $|0\rangle$ and $|1\rangle$. The 2 orthogonal x-basis states are $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$. The 2 orthogonal y-basis states are $|i\rangle = \frac{|0\rangle+i|1\rangle}{\sqrt{2}}$ and $|-i\rangle = \frac{|0\rangle-i|1\rangle}{\sqrt{2}}$. Notice how the H-gate then interacts as $H|0\rangle = |+\rangle$ and $H|1\rangle = |-\rangle$.

1.1.4 Phases and representation

The orthogonal basis states allow us to give a representation for a state vector known as **Bloch vectors** or **Bloch spheres** (as we will refer to). The x,y,z orthogonal basis states are at the opposite points of the sphere as their sibling x,y,z basis states and orthogonal to the others. As it is a unitary sphere we are talking about, we can represent a point on the sphere with 2 angles: θ to represent the positive rotation around the X-Z plane, and ϕ to represent the positive rotation around the X-Y plane. Then we get the following representation:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}sin\left(\frac{\theta}{2}\right)|1\rangle \quad , \text{ where } 0 \leq \theta < \pi \ , \ 0 \leq \phi < 2\pi$$

where the Euler formula gives $e^{i\phi} = cos(\phi) + i \cdot sin(\phi)$.

Notice that despite the amplitude α of $|0\rangle$ being in $\mathbb{C}$, it is restricted to $\mathbb{R}$, while the amplitude β of $|1\rangle$ can still be in $\mathbb{C}$. This is due to a convention that arises from measuring (section 1.3). Put simply, if $\gamma \in \mathbb{C}$ is such that $|\gamma| = 1$, then measuring qubit $|\psi\rangle$ and measuring $\gamma|\psi\rangle$ will give the same output. Such a γ is referred to as a **global phase**. Global phases provide no physical relevance to the output of a measurement or the operations performed on circuits, and can therefore always be neglected. That is why if $\alpha \in \mathbb{C}$, then a global phase can be applied such that actually $\alpha \in \mathbb{R}$ and is positive. In fact 2 qubits have the same Bloch sphere representation if and only if they only differ by a global phase. That is why $Y|0\rangle = i|1\rangle$ and $Z|1\rangle = -1 \cdot |1\rangle$ are both just $|1\rangle$ by the global phases $-i$ and -1, respectively.

All state vectors of a qubit achieved from gate operations can be mapped by a Bloch sphere. As we can see below, $X = HZH$.

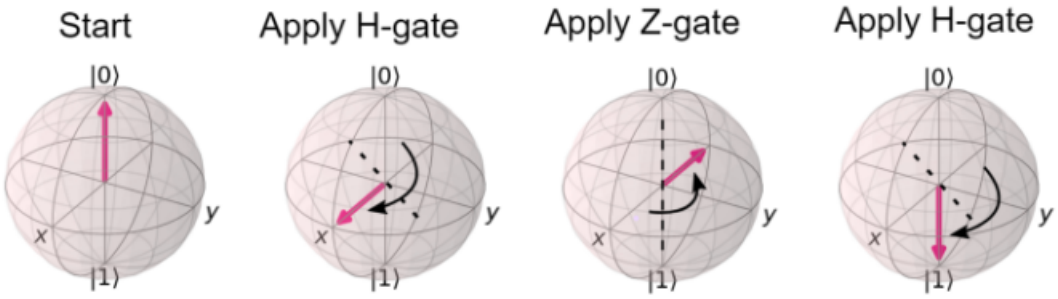


Figure 1: Example of the Bloch sphere representation of a qubit state vector.

1.1.5 Phase gate

The **Phase gate** corresponds to a rotation around the Z-axis (as visualised in a Bloch sphere) in proportion to a real parameter ϕ. It is $P(\phi) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix}$. Notice how $P(\pi) = Z$, so Z is just a special case of P. As you will see once we learn about measurements (Section 1.3), if we have a qubit $|\psi\rangle$, then measuring $|\psi\rangle$ and measuring $P(\phi)|\psi\rangle$ will give the same output. This is similar to that of global phases, but here a change in quantum operations happens (can also see from the Bloch Sphere that they represent 2 different points, rotated around the Z-axis). That is why we call such a phase a **relative phase**.

1.1.6 General

The **U gate** is the most general gate for when a very specific operation is desired. U has 3 parameters and has the form

$$U(\theta, \phi, \lambda) = \begin{bmatrix} cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}sin\left(\frac{\theta}{2}\right) & e^{i(\lambda+\phi)}cos\left(\frac{\theta}{2}\right) \end{bmatrix}$$

All above gates are special cases of the U gate. Play around and find the combinations that make them!

1.2 2-qubit gates and circuits

Since 2 classical bits can have values 00, 01, 10 and 11, it makes sense that the state vector of 2 qubits can have 4 computational bases, namely $|00\rangle, |01\rangle, |10\rangle$ and $|11\rangle$. We can then also imagine a whole mix of orthogonal basis states as well. Generalising, n many qubits will have 2^n many computational bases. The state vector of 2 qubits is the inner product of their respective state vectors i.e.

$$|\alpha\rangle \otimes |\beta\rangle = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \otimes \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} = \begin{bmatrix} a_0b_0 \\ a_0b_1 \\ a_1b_0 \\ a_1b_1 \end{bmatrix} = a_0b_0|00\rangle + a_0b_1|01\rangle + a_1b_0|10\rangle + a_1b_1|11\rangle = |\alpha\beta\rangle$$

Inner products can be applied iteratively of course. Since $|\alpha\rangle$ and $|\beta\rangle$ are each unitary, then $|\alpha\beta\rangle$ need also be unitary i.e. $|a_0b_0|^2 + |a_0b_1|^2 + |a_1b_0|^2 + |a_1b_1|^2 = 1$. 2-qubit gates are commonly designed such that one qubit acts as a **control qubit** and the other as a **target qubit**. Since the state vector of 2 qubits lies in $\mathbb{C}^4$, then a 2-qubit gate lies in $U(4, \mathbb{C})$. Theoretically, a n-qubit gate would lie in $U(2^n, \mathbb{C})$.

1.2.1 CNOT gate

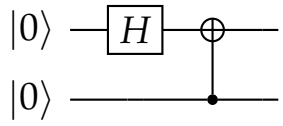
One example of a 2-qubit gate would be the **CNOT gate**, or **controlled-not gate**. The matrix form of CNOT is

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad \text{or} \quad CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

depending on whether the first qubit in $|\phi\psi\rangle$ is the control or the target, respectively. The basic idea for 2-qubit controlled gates like this is for the target qubit to perform the 1-qubit gate version of the gate, depending on if the control qubit. The $|0\rangle$ part of the control qubit says "don't you dare!", and the $|1\rangle$ part says "just do it!". However, contrary to the idea of 'control' and 'target', it is possible for the control qubit to be changed as well. This is referred to as **phase kickback**. Consider the following as an example.

$$CNOT|-+\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ -\frac{1}{2} \\ -\frac{1}{2} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \end{bmatrix} \Rightarrow \begin{matrix} a_0b_0 = \frac{1}{2} \\ a_0b_1 = -\frac{1}{2} \\ a_1b_0 = -\frac{1}{2} \\ a_1b_1 = \frac{1}{2} \end{matrix} \xrightarrow{\text{solving}} CNOT|-+\rangle = |--\rangle$$

Let's consider the following circuit:



Both qubits are initially $|0\rangle$, then an H gate is applied to qubit 1, followed by a CNOT gate with qubit 1 as the control qubit (small dot) and qubit 2 as the target qubit (oplus). The state vector thereafter is $(|00\rangle + |11\rangle)/\sqrt{2}$. Notice that $|00\rangle$ and $|11\rangle$ are equally likely, and moreover that also due to the likelihood of $|01\rangle$ and $|10\rangle$ being 0, this quantum state has some interesting deductive properties: if we see the one qubit is e.g. 1, then the other qubit must be 1 as well and vice versa. Similar and vice versa for 0 too. This phenomenon is called **entanglement**, and it is possible for more than 2 qubits to also get entangled.

1.2.2 SWAP gate

Sometimes (for some reason) we would like to swap the position of 2 qubits in a circuit. This could even be only for visual representational purposes. For this, we have the **SWAP gate**

$$SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

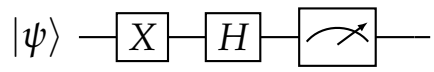
1.2.3 Control gates

Any 1-qubit gate can be used to create a controlled 2-qubit version. If we, for example, consider the Z-gate, we can get the **controlled-Z-gate**, or **CZ gate**, by $CZ = \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0} & Z \end{bmatrix}$, where I is the 2x2 identity matrix and $\mathbf{0}$ is the 2x2 zero matrix. Note that the form here is assuming the first qubit is the target and the second qubit is the control. We can continue this "stacking" indefinitely even should we want to e.g. the **controlled-controlled-Z gate** is $CCZ = \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0} & CZ \end{bmatrix}$, with I and $\mathbf{0}$ this time 3x3 matrices.

1.3 Measurement

In order to have an output after gate operations that we can actually use e.g. for analysis or to combine with classical methods, we need to **measure** the qubit(s). This reduces the state vector of the qubit (which is probably in superposition) to a computational basis $|0\rangle$ or $|1\rangle$, based on a probability density. Thus, we would only like to measure a circuit at the end of all our operations since there is no UNDO after measuring to retrieve the state vector (that is, unlike with the quantum gates, information is lost after a measurement). Recall that the computational bases are "equivalent" to the classical bit states. Therefore we can convert to and from bits and qubits after measuring.

Due to measurements happening based on a probability density, in order to have an (more) accurate output to work with, measurement is repeated many times by taking many **shots** at it, typically 1000 shots. An example 1-qubit circuit can be seen below. If, for example, $\psi = |0\rangle$ initially, then just before measurement, $\psi = |-\rangle$. Then because the state vector is "equally close" to $|0\rangle$ and $|1\rangle$ in terms of unitary weighting, we would find that there is a $50\backslash 50$ chance of measuring 0 or 1.



More precisely, the probability of measuring $|\psi\rangle$ and finding $|x\rangle$ is $p(|x\rangle) = |\langle x|\psi\rangle|^2$, where $\langle x| = [x_1^*, x_2^*]$ is called a **bra** (so bra-ket notation... get it?), $*$ is the complex conjugate operation, and $\langle x|\psi\rangle = \langle x||\psi\rangle$ (for brevity reasons). Thus

$$p_{|\psi\rangle}(|0\rangle) = |\langle 0|\psi\rangle|^2 = \left| [1,0] \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \right|^2 = \left| \frac{1}{2} \right|^2 = 0.5$$

Consider another example: Suppose we have a 3-qubit circuit with state vector

$$|\alpha\rangle \otimes |\beta\rangle \otimes |\gamma\rangle = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \otimes \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} \otimes \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = |\psi\rangle = \begin{bmatrix} z_0 & z_1 & z_2 & z_3 & z_4 & z_5 & z_6 & z_7 \end{bmatrix}^T$$

The rule still applies that $\sum_{i=0}^{7} |z_i|^2 = 1$ and the z_i are the amplitudes for the i-th computational basis e.g. z_0 is the amplitude for $|000\rangle$ and z_5 is the amplitude for $|101\rangle$. The probability of measuring $|101\rangle$ is simply $|z_5|^2$.

1.4 Design

Quantum computation is basically a complicated mess of the following 3 steps:
(1) Start in computational basis state
(2) Apply 1-qubit and 2-qubit gates
(3) Measure in computational state

Probability of any result is then the squares of absolute values of corresponding amplitudes.

More precisely, the basis for constructing a quantum model is:
BUILD a quantum circuit to solve the problem at hand.
COMPILE circuits for a specific service, such as a quantum system or classical simulator.
RUN the compiled circuits on the service.
ANALYZE the computed summary statistics and visualize the results.

In terms of the "compile" part, the compilation happens into **OpenQASM (Open Quantum Assembly Language)**. In fact, the **QASM Simulator** is the main Aer backend for *qiskit*. In terms of the "build" part, what might a problem be? One common problem is when a function exists that takes a certain amount of input and outputs some value(s), but it is unknown what the function is, or what are all the mappings for all possible inputs. Such a function is called an **oracle**. On our circuit, this would take the form of an n-qubit gate that we want to know the entries for, and from there we might want to or be able to decompose it into products of 1-qubit and 2-qubit gates.

1.5 Additional remarks

1.5.1 Phase factor

Take note of

$$\begin{bmatrix} e^{i\theta} & 0 \\ 0 & e^{i\theta} \end{bmatrix} = e^{i\theta} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = e^{i\theta} I$$

When $\theta \in \mathbb{R}$, then this is a unitary matrix. The number $e^{i\theta}$ is called the **global phase factor**. Such gates have no effect at all on the measurement probabilities. By this sense. X and -X are the same up to a global phase factor. Global phase factors only change amplitudes and nothing else.

1.5.2 Compounded Hadamard gates

An important understanding we will need later for algorithms is how multiple Hadamard gates act on a circuit. Suppose we have an n-qubit circuit. Notice that if $a \in \{0, 1\}$, then

$$H|a\rangle = H(a_0|0\rangle + a_1|1\rangle) = \frac{1}{\sqrt{2}}[(a_0 + a_1)|0\rangle + (a_0 - a_1)|1\rangle] = \frac{1}{\sqrt{2}} \sum_{x \in \{0,1\}} (-1)^{a \cdot x} |x\rangle$$

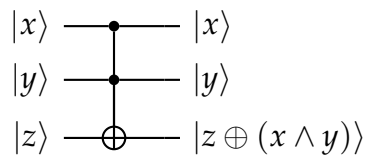
where $a \cdot x$ is the bitwise product. Fortunately, this representation can be generalised such that for $b \in \{0, 1\}^n$, we have

$$H^{\otimes n}|b\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{b \cdot x} |x\rangle$$

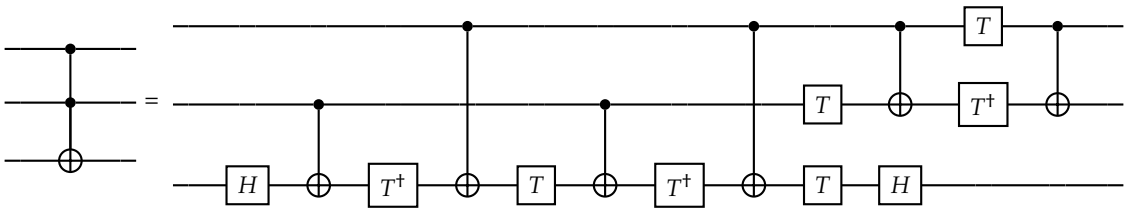
You can test this out yourself for $n = 2$ and see that it holds.

1.5.3 Toffoli gates

Quantum computers should be able to do all that computers can, and more. qNOT (or X) can easily be NOT. Whereas AND (or $x \wedge y$) needs to be represented by a 3-qubit quantum gate called the **Toffoli gate**, also known as the **controlled-controlled-NOT gate** or **CCX**:



where $\oplus$ is addition modulo 2. Toffoli gates aren't exactly part of what we consider a basic tool, since those would be 1-qubit and 2-qubit gates, but we can build it out of what else we actually have to work with:



where

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$$

2 Deutsch-Josza Algorithm

This algorithm was the first algorithm that proved a quantum algorithm can outperform any classical algorithm on a specific task.

2.0.1 Problem

Consider a function f that takes a bitstring of length n and returns a 0 or 1. We do not know how it decides whether to output a 0 or 1, but we do know that f is either constant (just outputs 0 or 1 for all bitstrings) or balanced (50% of cases output 0, 50% of cases output 1). We want to find out whether f is constant or balanced.

2.0.2 Classical solution

Input a variety of bitstrings into f in the hope of outputs differing, which would indicate non-constant i.e. balanced. Thus the best possible outcome would be testing only 2 cases and them differing, however the worst case would be needing to test (50% of cases) + 1 of the possible bitstrings, in the event that the first 50% of cases (= 2^{n-1}) many bitstrings all give the same output (0 or 1).

2.0.3 Quantum solution

The quantum solution however only ever needs to test f once in order to solve the problem (best and worst case). Assume that f has been implemented as a quantum oracle, which an n-qubit register $|x\rangle$ and 1-qubit register $|y\rangle$, is such that

$$f(|x\rangle|y\rangle) = |x\rangle|y \oplus f(x)\rangle$$

and $\oplus$ is simply addition modulo 2.

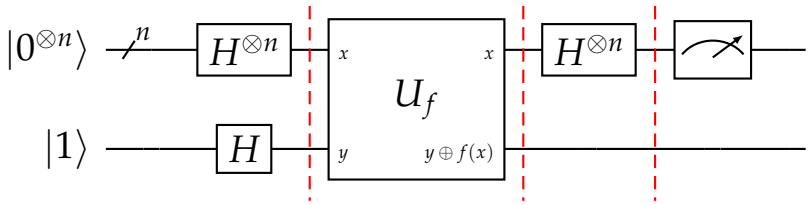
Denote the first register as X and the second register as Y. First initialize X to $|0\rangle$ and Y to $|1\rangle$ such that $|\psi_0\rangle = |0\rangle^{\otimes n}|1\rangle$. Note $|0\rangle^{\otimes n}$ indicates the n-times Kronecker product. Next we apply a Hadamard gate to all $n+1$ qubits.

$$|\psi_1\rangle = H^{\otimes(n+1)}|\psi_0\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{x \in \{0,1\}^n} |x\rangle(|0\rangle - |1\rangle)$$

We now apply our quantum oracle f to the circuit.

$$|\psi_2\rangle = f(|\psi_1\rangle) = \frac{1}{\sqrt{2^{n+1}}} \sum_{x\in\{0,1\}^n} x\rangle(|f(x)\rangle - |1 \oplus f(x)\rangle) = \frac{1}{\sqrt{2^{n+1}}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)}|x\rangle(|0\rangle - |1\rangle)$$

The second equality holds since for each $x, f(x)$ is either 0 or 1. We now ignore Y (which is $|-\rangle$) and only apply Hadamard gates once more to all qubits in X.

$$\begin{aligned}|\psi_3\rangle = (H^{\otimes n} \otimes I)|\psi_2\rangle &= \frac{1}{2^n} \sum_{x\in\{0,1\}^n} (-1)^{f(x)} \left[\sum_{y\in\{0,1\}^n} (-1)x \cdot y|y\rangle\right] |-\rangle \\ &= \frac{1}{2^n} \sum_{y\in\{0,1\}^n} \left[\sum_{x\in\{0,1\}^n} (-1)^{f(x)}(-1)^{x\cdot y}\right] |y\rangle|-\rangle\end{aligned}$$

where $x \cdot y = x_0y_0 \oplus x_1y_1 \oplus \ldots \oplus x_{n-1}y_{n-1}$ is the sum of the bitwise product. Finally, we measure X. The probability of measuring $|0\rangle^{\otimes n}$ from X is $\left|\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}\right|^2$. If $f(x)$ is constant, then this equals $|2^n/2^n|^2 = 1$ or $|-2^n/2^n|^2 = 1$, depending on if $f(x)$ is constant zero or constant one, respectively. If $f(x)$ is balanced, then this equals $|(1/2^n)(n-n)|^2 = 0$. If the measurment holds a string 0's, then the function is constant; any other string i.e. a 1 somewhere, implies the function is balanced. Thus we are done. The visual implementation of the algorithm is in the below figure.