Deutsch-Jozsa Algorithm

Indranil Ghosh
Jadavpur University, Physics Department
UG-III
Email: indranilg49@gmail.com

January 12, 2018

Abstract

This project is to compare the simulation of Deutsch-Jozsa Algorithm, between IBM's Quantum Experience's real quantum simulator and a Python program that has been designed, available in Github (https://github.com/indrag49/Quantum-SimuPy), for simulating the ideal case. Packages used: Numpy, Pandas and Mathplotlib. The quantum gates and the functions required for this simulation are available in the program and can be easily implemented. The moto of this project is to use python packages to design a quantum simulator for performing simple quantum computations.

1 Introduction

The *Deutsch-Jozsa Algorithm*, a quantum algorithm, was devised by David Deutsch and Richard Jozsa in the year 1992, which was later improved by *Cleve et al. in 1998*. This is a more general quantum algorithm derived from *Deutsch Algorithm* which was developed by David Deutsch in 1985 and was non-deterministic in nature. DJ Algorithm is rather deterministic in nature and served as the first algorithm that raised a barrier between the classical and quantum difficulty of a problem.

2 The Problem

Suppose there are two participants: a sender(**S**) and a reciever(**R**) situated at a large distance from one another. **S** tends to send a number x, between 0 and $2^n - 1$ to **R**. **R** has a quantum black box operator, the *oracle* that implements the function $f(x)$, which takes n-digit binary values as inputs and outputs either 0 or 1 for each such values, i.e, $f : [0,1]^n \rightarrow [0,1]$. Now, it is sure that, $f(x)$ is only of two kinds,

1. Constant, for all values of x, or
2. Balanced, i.e, is equal to 1 for exactly half of all the possible xs and 0 for the other half.

The algorithm determines with certainty whether$f$ used is Constant or Balanced by using the oracle U_f. In the originial non-determinstic algorithm by Deutsch, the function f had 1 bit as input and the goal was to determine whether f was constant. $f : [0, 1] \rightarrow [0, 1]$. A deterministic algorithm is a type of algorithm which always outputs the same result for a particular input.

3 The Classical counterpart

Classically, **S** needs to know from **R** $2^{n-1} + 1$ times, in the worst case scenario. This can be viewed in the following way: Let **S** gets signal everytime from **R** after **R** implements the function f using the oracle. **S** needs to recieve 2^{n-1} 0s before getting a 1 or a 0, which determines whether f is balanced or constant. so, the best classical algorithm needs $2^{n-1} + 1$ signals to be received before determining the kind of f. But using the DJ algorithm this is possible by only one function implementation. Classically, the best case occurs where the function is balanced and the first two signals are different, verifying the nature of f.

4 Action of the Oracle

The action of the *Oracle* can be depicted as,

$$U_f \left|x\right\rangle \left|q\right\rangle = \left|x\right\rangle \left|q \bigoplus f(x)\right\rangle$$

Here, $\left|x\right\rangle$ is the index register, $\bigoplus$ denotes addition modulo 2 and the oracle qubit $\left|q\right\rangle$, a single qubit which flips if $f(x) = 1$ and else remains unchanged.

5 Algorithm Procedure

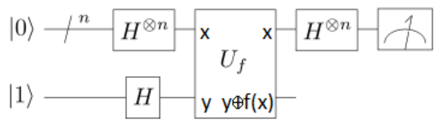

Figure 1: Deutsch-Jozsa Algorithm Circuit [Figure from Wikipedia]

Initially, to start with, the Sender, **S** has an n qubit register, **r**, to send the number x to the Receiver, **R**, and a single qubit register to store the signal **R** sends, by using quantum parallelism and evaluating f. That is, the initial state, let be depicted by $\left|P\right\rangle$, be $\left|0\right\rangle^{\otimes n} \left|1\right\rangle$. $\left|P\right\rangle = \left|0\right\rangle^{\otimes n} \left|1\right\rangle$. Applying Hadamard transform to each registers, we get the qubit state,

$$\left|P2\right\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} \left|x\right\rangle \frac{(\left|0\right\rangle - \left|1\right\rangle)}{\sqrt{2}}.$$

Next, **R** evaluates f by applying the quantum oracle matrix U_f to $\left|P\right\rangle$,

$$|P3\rangle = U_f\,|P2\rangle = \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle\,\frac{(|f(x)\rangle - |1\oplus f(x)\rangle)}{\sqrt{2}} =$$
$$\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}(-1)^{f(x}\,|x\rangle\,\frac{(|0\rangle - |1\rangle)}{\sqrt{2}}$$

For each x, $f(x)$ is either 0 or 1. **S** now possesses an array of qubits in which the evaluation of f is stored. At the next step, the last qubit, the signal register, is ignored as **S** interferes terms in the superposition using Hadamard transform on **r**. Our work now is to define the effect of Hadamard transform on a state $|x\rangle$.

$$H^{\otimes n}\,|x_0, x_1, ..., x_{n-1}\rangle = \frac{\sum_{y_0,y_1,...,y_{n-1}}(-1)^{x_0y_0\oplus x_1y_1\oplus\cdots\oplus x_{n-1}y_{n-1}}|y_0,y_1,...,y_{n-1}\rangle}{\sqrt{2^n}}$$
$$\Rightarrow H^{\otimes n}\,|x\rangle = \frac{\sum_y(-1)^{x\cdot y}|y\rangle}{\sqrt{2^n}}$$

Here, $x.y$ is the sum of the bitwise inner product of x and y modulo 2. Evaluating the final state we get,

$$\frac{1}{2^n}\sum_{x=0}^{2^n-1}\sum_{y=0}^{2^n-1}(-1)^{x.y+f(x)}\,|y\rangle\,\frac{(|0\rangle - |1\rangle)}{\sqrt{2}}$$

The probability of measuring $|0\rangle^{\otimes n}$ is ,

$$|\frac{1}{2^n}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|^2$$

So, now we can determine $f(x)$ by running the experience once.
1. Probability is 1 if $f(x)$=constant i.e., constructive interference
2. Probability is 0 if $f(x)$=balanced i.e., destructive interference [All the terms in the sum over x cancel each other]

5.1 Steps

1. Initialise $n+1$ qubits with n qubits in all zero states, $|00...0\rangle$ and the last qubit in one state, $|1\rangle$.
2. Apply Hadamard Transform to each qubits.
3. Apply the Oracle matrix U_f to the current superposition state.
4. Neglect the last qubit and apply Hadamard Transform again to the remaining qubits.
5. Perform measurement on each qubit.

6 Comparison Of The Example Circuits

We consider $n = 3$ and $f(x) = x_0 \oplus x_1x_2$. This function is balanced as flipping the bit x_0 flips the value of $f(x)$ irresepective of $x1$ and $x2$.

6.1 IBM Q Experience Results

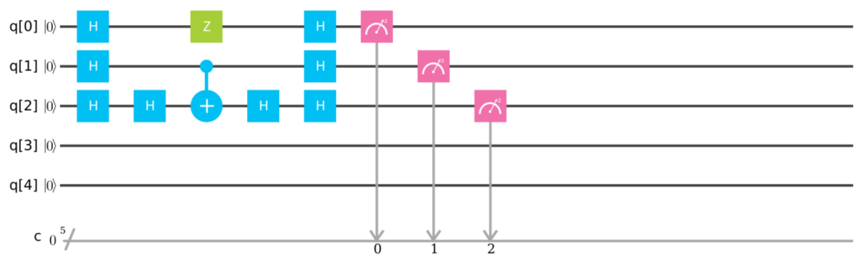

Figure 2: Algorithm Circuit[Figure from IBM Q experiences home page]

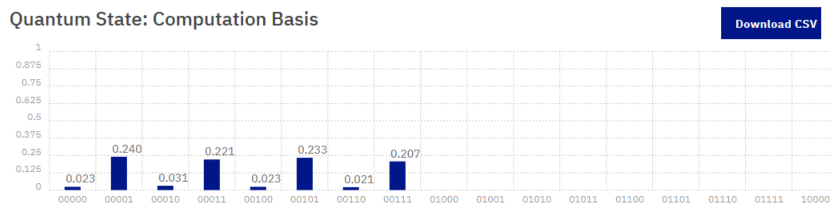

Figure 3: Probability Distribution, with noise[Figure from IBM Q experiences home page]

6.2 Python Simulation

```
k=np.kron
H=Hadamard(I2)
Z=PauliZ(I2)
```

```
cnot=CNOT(I4)
h=k(k(H, H), H)
h1=k(I2, H)
U_f=k(Z, h1.dot(cnot.dot(h1)))
m=h.dot(U_f.dot(h.dot(Q000)))
p=measure(m)
print(p)
plot_measure(p)
```

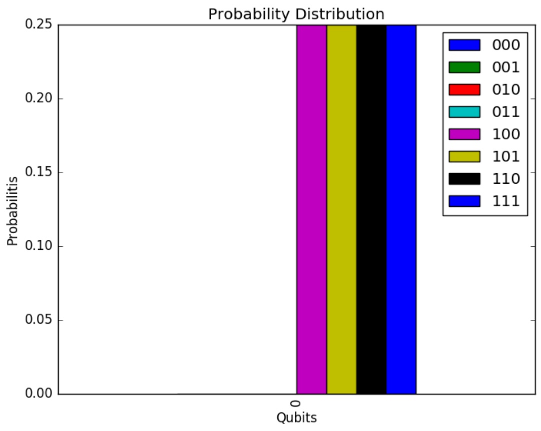

Figure 4: Probability Distribution, without noise

For constant $f(x)$,

6.3 IBM Q Experience Results

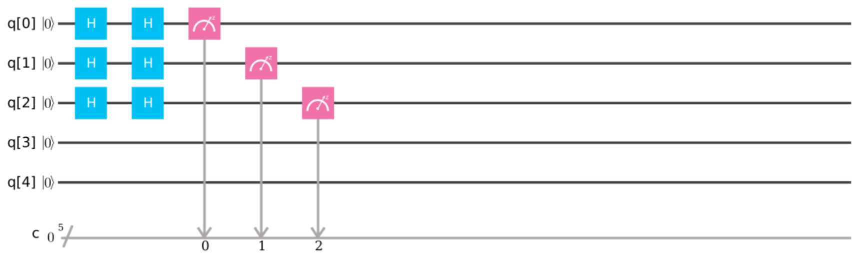

Figure 5: Algorithm Circuit[Figure from IBM Q experiences home page]

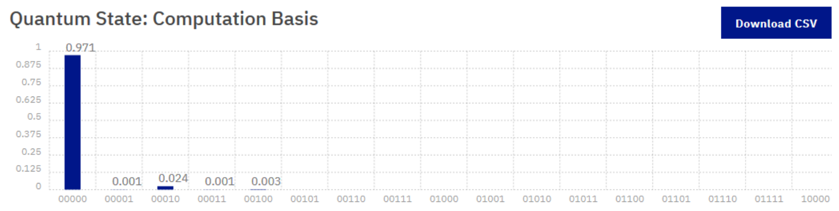

Figure 6: Probability Distribution, with noise[Figure from IBM Q experiences home page]

6.4 Python Simulation

```
k=np.kron
H=Hadamard(I2)
h=k(k(H, H), H)
m=h.dot(h.dot(Q000))
p=measure(m)
print(p)
plot_measure(p)
```

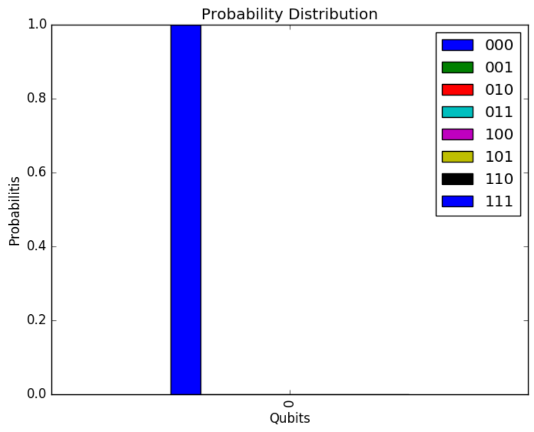

Figure 7: Probability Distribution, without noise

References

[1] Quantum Computation and Quantum Information
by Michael A. Nielsen and Isaac L. Chuang,
URL:http://www-reynal.ensea.fr/docs/iq/QC10th.pdf

[2] Wikipedia,Deutsch Jozsa Algorithm, https://en.wikipedia.org/wiki/DeutschJozsa algorithm

[3] IBM Quantum Experience Home Page, https://quantumexperience.ng.bluemix.net/qx/experience

[4] https://github.com/indrag49/Quantum-SimuPy