# PROGRAMMING A QUANTUM COMPUTER WITH QISKIT - IBM SDK

Shubhajit Dey

DEPARTMENT OF MATHEMATICS,
INDIAN INSTITUTE OF SCIENCE EDUCATION AND RESEARCH, BHOPAL

December 7, 2022

## Abstract

The primary objective of this project is to analyse the Bernstein-Vazirani Algorithm. In order to achieve the primary objective, we first get familiarised with the standard Bra-Ket Notations and basic qubit arithmetics and algebra. Next, we proceed towards understanding the preliminary operations of some common quantum circuits and quantum gates. Finally, we arrive at our proposed primary objective.

**Pre-requisites**

Linear Algebra, Introductory Matrix Theory, Basics of programming in python, Familiarity with Quantum Mechanics.

**Keywords**

Quantum Computing, Qiskit.

## 1 Introduction

This article documents all the major theories involved in the mentioned project, whose code report is also attached or can be found at `GitHub Repo`. Quantum Computing has seen a significant boost in its popularity, and with this comes a demand for the simulation of quantum gates and quantum circuits through programming. This project proposes an elementary attempt at understanding basic quantum gate simulation and, finally, implementing the Bernstein-Vazirani Algorithm to solidify its applications.

This article is structured in a way where significant mathematical concepts involved in the project are put first through the section `preliminary mathematical concepts`, followed by the `standard definition` section, which comprises all notations and standard definitions under usage. Then `Quantum Gates` section is kept, where the mentioned topic is described in some degree of detail. Then this section is followed by the section for `Bernstein-Vazirani Algorithm`. Finally, the article is put to a halt with `Acknowledgements`.

## 2 Preliminary Mathematics

Let us go through all the preliminary mathematics required for the objective mentioned. This includes revisiting, Bra-Ket Notations, Vector Inner Products and Matrix Tensor Product.

### 2.1 Bra-Ket Notations & Inner product

Here, we will get familiar with the Bra-Ket Notations[1] used in classical Quantum computing literature. Row vectors in quantum mechanics are also referred to as **Bra vectors** and are denoted as follows :

$$\langle A| = \left(a_1, a_2, a_3, \cdots, a_n\right)$$

Likewise, column vectors are also called **Ket vectors** in quantum mechanics and are denoted as follows:

$$|B\rangle = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{pmatrix}$$

In general, if we have a column vector, that is, a ket-vector like:

$$|A\rangle = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{pmatrix}$$

its corresponding bra-vector would be:

$$\langle A| = \begin{pmatrix} a_1^*, a_2^*, a_3^*, \cdots, a_n^* \end{pmatrix}$$

where, $a_i^*$ denotes the complex conjugate of the scalar $a_i$. Precisely, if $a_i = p + i \cdot q$, a complex scalar, then $a_i^* = p - i \cdot q$. We define the **Inner Product** of the above-presented bra and ket vectors as :

$$\begin{aligned} \langle A|B\rangle &= \begin{pmatrix} a_1, a_2, a_3, \cdots, a_n \end{pmatrix} \cdot \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{pmatrix} \\ &= a_1 b_1 + a_2 b_2 + \cdots + a_n b_n. \\ &= \sum_1^n a_i b_i \end{aligned}$$

## 2.2 Matrix Tensor Product

If we have two matrices say :

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} and \begin{pmatrix} x & y \\ z & w \end{pmatrix},$$

then, we can define their **tensor product** as :

$$\begin{aligned} \begin{pmatrix} a & b \\ c & d \end{pmatrix} \otimes \begin{pmatrix} x & y \\ z & w \end{pmatrix} &= \begin{pmatrix} a\begin{pmatrix} x & y \\ z & w \end{pmatrix} & b\begin{pmatrix} x & y \\ z & w \end{pmatrix} \\ c\begin{pmatrix} x & y \\ z & w \end{pmatrix} & d\begin{pmatrix} x & y \\ z & w \end{pmatrix} \end{pmatrix} \\ &= \begin{pmatrix} ax\ ay\ bx\ by \\ az\ aw\ bz\ bw \\ cx\ cy\ dx\ dy \\ cz\ cw\ dz\ dw \end{pmatrix}. \end{aligned}$$

## 3 Standard Definitions

**Qubits :** Qubit or a Quantum Bit[2]-`3.3`, is a basic unit of quantum information. Basically, a qubit is the quantum version of the classical bit used in classical computing. For the sake of mathematical sanity, we can introduce a qubit as $|\psi\rangle$ such as :

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}, \text{ where } \sqrt{\langle\psi|\psi\rangle} = 1.$$

Now this definition trivially provides two contenders for a qubit, that is,

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ and } \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Thus, it is clear that a qubit can be physically interpreted as an electron, possessing two states, namely, spin-up and spin-down.

$$\begin{aligned} \text{spin-up} : |0\rangle &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ \text{spin-down} : |1\rangle &= \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \end{aligned}$$

Another representation of qubits can be done via the Bloch Sphere, `[section 1.2.2 of code report attached]`. Unlike the classical bit, where each bit has to be either 0 or 1, a qubit can remain in a linear superimposed state of both the basis states, i.e., $|0\rangle$ and $|1\rangle$ [2]-`3.5`.

The basis states $|0\rangle$ and $|1\rangle$ are orthonormal ket vectors which are also called the computational basis. Qubit basis states are often combined to form product basis states. A set of qubits taken together is often called a Quantum Register. One can prepare other states from these basis states as:

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

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

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

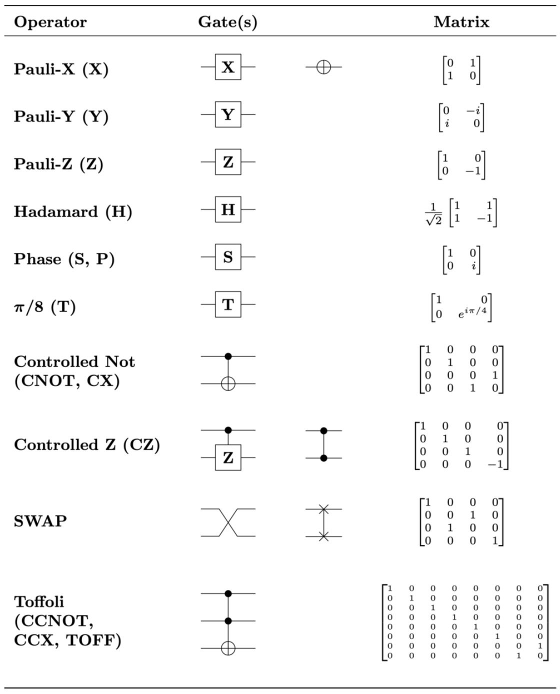
$$|11\rangle = |1\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

The above ket vectors formed can also be thought of as a representation of two basis qubits in a four-dimensional linear vector space spanned by the above product basis states.

## 4 Quantum Gates

In Quantum Computing, a Quantum Logic Gate or simply a **Quantum Gate** [2]-`3.4`, is a basic quantum circuit which operates on a small number of qubits. Like classical logic gates are the building blocks for conventional digital circuits, quantum gates are the building blocks of quantum circuits. In general, the majority of quantum gates under usage are a reversal in contrast to their classical counterparts. Practically quantum circuits can perform all operations performed by classical circuits and an existential proof for this statement can be produced by presenting the example of the reversible Toffoli gate[3]`-3.2`, which can implement all Boolean functions often at the cost of having to use ancilla bits. The Toffoli gate has a direct quantum equivalent.

From a mathematical perspective, in theory, we have that quantum gates can be represented as unitary matrices[4] relative to the appropriate basis simply because they are unitary operators. Some popular quantum logic gates are presented in figure[1].



| Operator | Gate(s) | | Matrix |
|---|---|---|---|
| **Pauli-X (X)** | $\boxed{X}$ | $\oplus$ | $\begin{bmatrix}0&1\\1&0\end{bmatrix}$ |
| **Pauli-Y (Y)** | $\boxed{Y}$ | | $\begin{bmatrix}0&-i\\i&0\end{bmatrix}$ |
| **Pauli-Z (Z)** | $\boxed{Z}$ | | $\begin{bmatrix}1&0\\0&-1\end{bmatrix}$ |
| **Hadamard (H)** | $\boxed{H}$ | | $\frac{1}{\sqrt{2}}\begin{bmatrix}1&1\\1&-1\end{bmatrix}$ |
| **Phase (S, P)** | $\boxed{S}$ | | $\begin{bmatrix}1&0\\0&i\end{bmatrix}$ |
| **$\pi/8$ (T)** | $\boxed{T}$ | | $\begin{bmatrix}1&0\\0&e^{i\pi/4}\end{bmatrix}$ |
| **Controlled Not (CNOT, CX)** | | | $\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}$ |
| **Controlled Z (CZ)** | $\boxed{Z}$ | | $\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&-1\end{bmatrix}$ |
| **SWAP** | | | $\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix}$ |
| **Toffoli (CCNOT, CCX, TOFF)** | | | $\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&0&1&0\end{bmatrix}$ |

Figure 1: Popular quantum logic gates, their circuits and respective unitary matrices.

The **tensor product**, also often called the Kronecker product, is used to combine quantum states. The combined state for a qubit register is the tensor product of the constituent qubits. The tensor product, as described earlier, is denoted by the symbol $\otimes$. The vector representation of the state of two qubits is:

$$|\psi\rangle = a_{00}|00\rangle + a_{01}|01\rangle + a_{10}|10\rangle + a_{11}|11\rangle$$

The action of the gate on a specific quantum state is found by multiplying the vector $|\psi_1\rangle$ which represents the state of the qubit, by the matrix U, which represents the gate. The result is a new quantum state $|\psi_2\rangle$:

$$U|\psi_1\rangle = |\psi_2\rangle.$$

## 5 Bernstein-Vazirani Algorithm

The Bernstein–Vazirani algorithm[5], which solves the Bernstein–Vazirani problem, is a quantum algorithm invented by Ethan Bernstein and Umesh Vazirani in 1992. It is a restricted version of the Deutsch–Jozsa algorithm where instead of distinguishing between two different classes of functions, it tries to learn a string encoded in a function[6].

### Problem Statement

Given an oracle (basically a black box) that implements a function $f : \{0,1\}^n \to \{0,1\}$, in which $f(x)$ is promised to be the dot product of the input $x$ and a hidden string $s \in \{0,1\}^n$ modulo 2,

$$f(x) = x \cdot s = x_1s_1 \oplus x_2s_2 \oplus \cdots \oplus x_ns_n,$$

find $s$.

### Algorithm

The quantum Bernstein-Vazirani Oracle:

1. Initialise the inputs qubits to the $|0\rangle^{\otimes n}$ state, and output qubit to $|-\rangle$.
2. Apply Hadamard gates to the input register.
3. Query the oracle.
4. Apply Hadamard gates to the input register.
5. Measure.

## Acknowledgements

The author acknowledges the support and guidance of Dr. Farhad Abdi, which were crucial for the progress and successful completion of this project.

## References

[1] Michael Dickson. Non-relativistic quantum mechanics. In Jeremy Butterfield and John Earman, editors, *Philosophy of Physics*, Handbook of the Philosophy of Science, pages 275–415. North-Holland, Amsterdam, 2007.

[2] Ivan S. Oliveira, Tito J. Bonagamba, Roberto S. Sarthour, Jair C.C. Freitas, and Eduardo R. deAzevedo. 3 - fundamentals of quantum computation and quantum information. In Ivan S. Oliveira, Tito J. Bonagamba, Roberto S. Sarthour, Jair C.C. Freitas, and Eduardo R. deAzevedo, editors, *NMR Quantum Information Processing*, pages 93–136. Elsevier Science B.V., Amsterdam, 2007.

[3] S.G. Roy and A. Chakrabarti. Chapter 11 - a novel graph clustering algorithm based on discrete-time quantum random walk. In Siddhartha Bhattacharyya, Ujjwal Maulik, and Paramartha Dutta, editors, *Quantum Inspired Computational Intelligence*, pages 361–389. Morgan Kaufmann, Boston, 2017.

[4] Physicspages. Unitary matrices. `http://physicspages.com/pdf/Quantum%20mechanics/Unitary%20matrices%20-%20some%20examples.pdf`, May 2013.

[5] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1997.

[6] Colin J Trout, Muyuan Li, Mauricio Gutiérrez, Yukai Wu, Sheng-Tao Wang, Luming Duan, and Kenneth R Brown. Simulating the performance of a distance-3 surface code in a linear ion trap. *New Journal of Physics*, 20(4):043038, apr 2018.

Reach us by -
Shubhajit Dey · · · shubhajit19@iiserb.ac.in