Introduction to Quantum Gates: A Comprehensive Study on Quantum Information

Debrup Roy Chowdhury[1], Asoke Nath[2]
[1,2]*Department of Computer Science, St. Xavier's Autonomous College, Kolkata*

***Abstract*—Quantum computers are the emerging future and special gates, different from those used in classical computers, are required to operate them. Decades of research has produced many such gates which are being used in the existing quantum computers, although they are in their infancy. To learn quantum computing, it is paramount that a decent understanding of many of these basic gates and their workings is made before starting to develop any quantum algorithm. In the present paper the authors have given an exhaustive study on various issues and developments of Quantum Gates.**

***Index Terms*—Quantum Computers, Quantum Information, Quantum Gates, Qiskit, Quantum Gate Calculations.**

I. INTRODUCTION

Modern equivalents of the ENIAC machines, Quantum computers are highly-complex & highly-sensitive machines and are perceived as black boxes by most, but they are anything but that and they have been physically realised by only a handful of institutions in the past twenty-five years.

A library of specially designed quantum logic gates[1] are used in these quantum computers to manipulate the quantum bits (also called **qubits**) to ensure higher probability of a desired combination of bits (here, qubits) at a much faster rate than most modern classical computing device.

This review paper aims at explaining the most popular qubit gates in use & thus provide a ground for the intuition as to how quantum computers essentially work. Using the **Bloch Sphere** & **Circuit Diagrams** (for multi-qubit gates) generated in **qiskit** (IBM's open-source software development kit build to work with their proprietary quantum processors), the effect of each quantum gate on the qubit(s) are visualized.

II. PRELIMINARY CONCEPTS

Quantum Information is a vast field and there are a few concepts of Quantum Mechanics that need to be studied and understood before the gates can be reviewed. These concepts are used to create algorithms suited to operate any quantum computer, also called Quantum Algorithms.

A. Unitary Matrix

In linear algebra, a complex square matrix U is a **Unitary Matrix** if its conjugate transpose U^* is also its inverse.

$$\mathbf{UU^{-1} = U^*U = UU^* = I}$$

All quantum gates are unitary square matrices of dimension 2^i (i = number of qubits the gate is acting upon).

B. Bra-Ket Notation

Also called **Dirac Notation**, this set of notations are used everywhere to describe a quantum state. If x is a quantum state, then a **bra** is denoted using $\langle x|$ and a **ket** is denoted by $|x\rangle$.

Mathematically, $|x\rangle$ denotes a vector x in a complex vector space **V** and it is a physical representation of a quantum state. Every quantum state in this article is denoted using this *ket* notation.

$\langle f|$ denotes a linear function f that linearly maps each vector in the complex vector space **V** to a scalar in complex plane **C** (i.e., mathematically, f: **V**→**C**)

In quantum mechanics, this is represented by

$$\langle f | x \rangle \in \mathbf{C}$$

C. Qubit

A Qubit or **Quantum Bit**, is the most fundamental unit of quantum information. It is a two-state quantum mechanical system. Examples of qubits are the *spin of an electron* or the *polarity of a photon*. The peculiar edge that a qubit gives over a classical bit is its property of being in a logical **Superposition** of both states simultaneously which serves as an essential requirement in all aspects of quantum mechanics and information.

A qubit system is a linear superposition of two orthonormal basis states $|0\rangle$ and $|1\rangle$. These 2 states span the 2D Hilbert space of a qubit and are together called the **computational basis**. More than 1 qubit can be combined to form a quantum register and can represented by a *superpositioned product state*

vector in 2^n dimensional Hilbert Space (n being the number of qubits in a quantum register). For example, for a quantum register with 2 qubits, there are total of 4 product basis states:

$$|00\rangle = \begin{bmatrix}1\\0\\0\\0\end{bmatrix} \quad |01\rangle = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$$

$$|10\rangle = \begin{bmatrix}0\\0\\1\\0\end{bmatrix} \quad |11\rangle = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

III. SINGLE QUBIT GATES

Qubits cannot be manipulated using logic gates implemented in classical computers. They need specialized gates created using the theoretical concepts of linear algebra & complex numbers, which are uniquely suited to manipulate the subatomic particle in use. There are several quantum gates that act upon only one qubit (called **Single Qubit Gates**). A single qubit gate is multiplied to the basis state vector (i.e., the vector stating the probability of the qubit of being read as quantum states $|0\rangle$ or $|1\rangle$) of a qubit to generate a new set of basis states for it.

In the present study the authors will discuss the following quantum logic gates :

A. **I** (or Identity Gate).
B. **Pauli X, Y** & **Z** (or Pauli Gates).
C. **H** (or Hadamard Gate).
D. **S, T** & **P** (or Phase Shift Gates).
E. $\mathbf{R_x}$, $\mathbf{R_y}$, $\mathbf{R_z}$ (or Rotation Gates).

All single qubit gates can be represented using a unitary square matrix of size 2 For each gate it's unitary matrix, code snippet to add it to a quantum circuit in **Qiskit**, it's action on the quantum basis state of **Initial Qubit** $|0\rangle$ shown in *Fig. 1* (except for the **Rotation** Gates) and a table (except for the **I** Gate) elaborating it's action on 6 different axial. points (some basic basis and superpositioned quantum states), which are

1. $|0\rangle \left(=\begin{bmatrix}1\\0\end{bmatrix}\right)$ 2. $|1\rangle \left(=\begin{bmatrix}0\\1\end{bmatrix}\right)$
3. $|+\rangle \left(=\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)$ 4. $|-\rangle \left(=\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$
5. $|i\rangle \left(=\frac{0\rangle + i1\rangle}{\sqrt{2}}\right)$ 6. $|\text{-}i\rangle \left(=\frac{0\rangle - i1\rangle}{\sqrt{2}}\right)$

In most results, the answer are given as another axial point or a linear combination of more than one axial points. But in some special cases, where the result cannot be given as a linear combination of axial points, the resultant qubit vector has been given.

A. Identity Gate (I)

Multiplication with identity Matrix in basic linear algebra generates the exact matrix it gets multiplied with. Identity Gate or **I** Gate is made out of the same identity matrix. It is denoted using the unitary matrix

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

and does not modify the state vector of a qubit. The only uses of this gate is to mathematically describe the results of many gate operations or when discussing circuits made out of multiple qubits.

From here forth, every single qubit gate will act upon the qubit represented by the **Bloch Sphere** in *Fig. 1*. By using the code format from qiskit below, I gate can be added on to a qubit in position *Qubit_position* in a quantum circuit *Quantum_Circuit*:

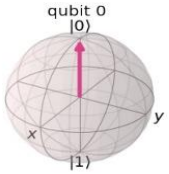

Figure 1: Initial Qubit Position

Quantum_Circuit.i(Qubit_position)

B. Pauli Gates (X, Y, Z)

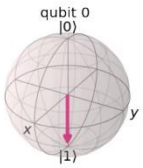

Figure 2: Qubit Position after Pauli-X

The next 3 gates are Pauli X, Y & Z gates (collectively called the **Pauli Gates**). They get their name from the celebrated Austrian theoretical physicist **Wolfgang Pauli**, who came up with the concept of Pauli Matrices, the unitary matrices used for the Pauli Gates. These matrices have anti-commute (i.e., -XZ = ZX = iY), involutory (i.e., multiplying any one of the Pauli Gate with itself gives the **I** Gate). These 3 gates rotate the target qubit in the **Bloch Sphere** by the corresponding axis in the sphere by an angle of π radians.

The first among these three is the **Pauli X** or $\boldsymbol{\sigma_x}$ Gate. On applying this gate, the initial qubit rotates along the x-axis of the Bloch sphere by π radians. In *Fig. 2*, it can be seen that Pauli X Gate transforms the initial qubit to the axial point $|1\rangle$. Also note that the *Pauli X gate is equivalent to the NOT gate in classical computing*. The unitary matrix of Pauli X Gate is

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

This gate is implemented in qiskit using the following code format:

Quantum_Circuit.x(Qubit_position)

A slight modification of the Pauli-X Gate is $\sqrt{X}$ or **V** gate, which can be represented using the unitary matrix

$$\frac{1}{2}\begin{bmatrix}1+i & 1-i\\ 1-i & 1+i\end{bmatrix}.$$

A special circuit library in qiskit offers various functions to implement such special modified gates and one of them helps in putting **V Gate** onto the circuit:

Quantum_Circuit.sx(Qubit_position)

Table 1 shows output when the 6 different axial points are given as input to both X & $\sqrt{X}$ gates. *The common expression in any tabular outputs is the phase shift above the qubit change that the initial qubit goes through on the application of the gate*.

The second is **Pauli Y** or $\boldsymbol{\sigma_y}$ Gate. When this gate is applied on an initial qubit, it rotates along the y-axis by π radians of the **Bloch Sphere**. This gate is represented in linear algebra using the unitary matrix

$$\begin{bmatrix}0 & -i\\ i & 0\end{bmatrix}$$

In qiskit, the following code format is used to implement **Pauli Y** Gate :

Quantum_Circuit.y(Qubit_position)

In *Fig. 3*, it can be seen how Pauli-Y Gate changes the initial qubit, but the output is same as the Pauli-X Gate. *Table 2* lists the output for the 6 axial points on applying Pauli-Y gate.

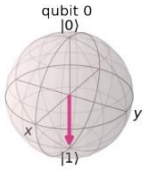

Figure 3: Qubit Position after Pauli-Y

The last of the Pauli Gates is **Pauli Z** or $\boldsymbol{\sigma_z}$ Gate & a qubit rotates by π radians along the z-axis when this gate is applied on it. In linear algebra, it is represented by the matrix

$$\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$$

An example of this is in *Fig. 4* when the gate is application is when it is applied to the initial qubit, which is same as applying the **Pauli Y** Gate.

But changes are seen when it is applied to a qubit with initial position of $|1\rangle$ in *Table 3*. So in summary, it keeps input of $|0\rangle$ unchanged but flips the phase of any other input, due to which the Pauli Z Gate is also sometimes called as the **Phase Flip Gate**. **Pauli Z** is also an example of **Phase Shift Gate**. In qiskit, the following code format is used to implement Pauli Z Gate:

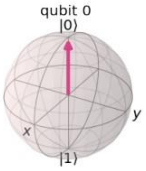

Figure 4: Qubit Position after Pauli-Z

Quantum_Circuit.z(Qubit_position)

C. Hadamard Gate (H)

The next gate is the **Hadamard** (or **H**) gate, which puts a qubit into superposition state & when measured collapses the superposition state of the qubit giving either a $|0\rangle$ or $|1\rangle$ basis state everytime the circuit is run with a 50% chance of being either of the basis states. Putting an **H** gate to the initial qubit, the result is as shown in *Fig. 5*. As seen, the qubit is now neither in basis state $|0\rangle$ or $|1\rangle$. It is in a superpositioned state. The unitary matrix of the Hadamard gate is

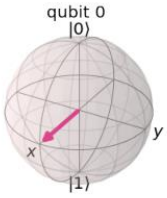

Figure 5: Qubit after Hadamard Gate

$$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$

In qiskit, the following code format is used to implement the **Hadamard Gate** onto a qubit:

Quantum_Circuit.h(Qubit_position)

Table 4 on previous page shows the result when this gate is applied on a few initial qubit positions.

Table 1: Pauli X & $\sqrt{X}$ on several initial qubit positions

Initial State	Final State (X)	Final State ($\sqrt{X}$)
$\|0\rangle$	$\|1\rangle$	$\frac{1}{\sqrt{2}}(\|+\rangle + \|-\rangle)$
$\|1\rangle$	$\|0\rangle$	$\frac{1}{\sqrt{2}}(\|+\rangle + i\,\|-\rangle)$
$\|+\rangle$	$\|+\rangle$	$\|+\rangle$
$\|-\rangle$	$-\|-\rangle$*	$i\,\|-\rangle$

$\|i\rangle$	$i\,\|-i\rangle$	$(\frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}})\,\|0\rangle$
$\|-i\rangle$	$-i\,\|i\rangle$	$(\frac{1}{\sqrt{2}} - \frac{i}{\sqrt{2}})\,\|1\rangle$

Table 2: Pauli Y Gate on initial qubit positions

Initial State	Final State
$\|0\rangle$	$i\,\|1\rangle$
$\|1\rangle$	$-i\,\|0\rangle$
$\|+\rangle$	$-i\,\|-\rangle$
$\|-\rangle$	$i\,\|+\rangle$
$\|i\rangle$	$\|i\rangle$
$\|-i\rangle$	$-\,\|-i\rangle$

Table 3: Pauli Z Gate on initial qubit positions

Initial State	Final State
$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$-\,\|1\rangle$
$\|+\rangle$	$\|-\rangle$
$\|-\rangle$	$\|+\rangle$
$\|i\rangle$	$\|-i\rangle$
$\|-i\rangle$	$\|i\rangle$

Table 4: Hadamard Gate on initial qubit positions

Initial State	Final State
$\|0\rangle$	$\|+\rangle$
$\|1\rangle$	$\|-\rangle$
$\|+\rangle$	$\|0\rangle$
$\|-\rangle$	$\|1\rangle$
$\|i\rangle$	$\|+\rangle - i\|-\rangle$
$\|-i\rangle$	$\|+\rangle + i\|-\rangle$

***Note:** $-\,|-\rangle = -(\frac{|0\rangle - |1\rangle}{\sqrt{2}}) = -\frac{1}{\sqrt{2}}(\begin{bmatrix}1\\0\end{bmatrix} - \begin{bmatrix}0\\1\end{bmatrix}) = -\frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix} = \begin{bmatrix}-\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{bmatrix}$

D. Phase Shift Gates (S, T, P)

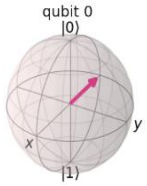

Figure 6: Qubit after Hadamard & Z Gate

Phase Shift Gates are a family of single-qubit gates that keeps the basis state $|0\rangle$ unchanged but changes basis state $|1\rangle$ to $e^{i\phi}|1\rangle$, where ϕ is the **angle of rotation** of the qubit along z-axis. Although this class of gates doesn't change the probability of either $|0\rangle$ or $|1\rangle$, they do change the phase of the quantum state. The phase shift gate is denoted by the matrix:

$$P(\phi) = \begin{bmatrix}1 & 0\\ 0 & e^{i\phi}\end{bmatrix}$$

Identity gate & **Pauli Z** gate are the most basic **Phase Shift Gates** where ϕ(or angle of rotation) is equal to 0 & π respectively. Other phase shift gates with their angles of rotation are S(or P(π/2)), T(or P(π/4)) which can be represented as $\sqrt{Z}$ & $\sqrt[4]{Z}$ respectively.

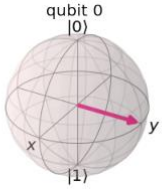

Figure 7: Qubit Position after Hadamard & S Gate

It should be noted that phase shift gates are **Hermitian** (i.e., a matrix whose transpose is equal to its complex conjugate) only when ϕ lies in the range $[0, \pi]$. To show it's working, all the 3 common phase shift gates Z, S & T are applied onto the output of **Hadamard Gate** in *Fig. 6*, *Fig. 7* & *Fig. 8* respectively.

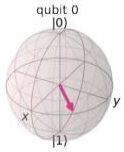

Figure 8: Qubit Position after Hadamard & T Gate

To implement gates S & T, following code snippets written in qiskit are used:

Quantum_Circuit.s(Qubit_position)
Quantum_Circuit.t(Qubit_position)

A general phase shift gate is also available which is implemented as follows:

Quantum_Circuit.p(Angle_of_rotation, Qubit_position)

Table 5 on next page shows the result of application of S & T gates on the 6 axial points.

E. Rotation Gates (R_x, R_y, R_z)

Rotation Gates are analog rotation matrices which allow rotation about all the 3 Cartesian Axes of the Euclidean Space (The axes collectively are also known as **SO(3)**). There is one for each of the axes - $R_x(\theta)$, $R_y(\theta)$, $R_z(\theta)$ - where θ is the angle of rotation. For every θ & every $b(\in \{x, y, z\})$, $R_b(-\theta) = R_b(\theta)^\dagger$ & $R_b(0) = I$. For $R_x(\theta)$, the unitary matrix is

$$\begin{bmatrix} cos(\theta/2) & -i.sin(\theta/2) \\ -i.sin(\theta/2) & cos(\theta/2) \end{bmatrix}$$

for $R_y(\theta)$, the unitary matrix is

$$\begin{bmatrix} cos(\theta/2) & -sin(\theta/2) \\ sin(\theta/2) & cos(\theta/2) \end{bmatrix}$$

& for $R_z(\theta)$, the unitary matrix is

$$e^{(-i.\theta/2)} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The matrix representations for every rotational gate for θs: (π/6), (π/4), (π/3), (π/2) are shown are shown in *Tables 6, 7 & 8*. These gates can be represented in

Table 9: Result of CNOT Gate on all Control-Target qubit combinations

Before		After	
Control	Target	Control	Target
|0⟩	|0⟩	|0⟩	|0⟩
|0⟩	|1⟩	|0⟩	|1⟩
|1⟩	|0⟩	|1⟩	|1⟩
|1⟩	|1⟩	|1⟩	|0⟩

Qiskit using the following code formats:

Quantum_Circuit.rx (Angle_Of_Rotation_In_Radians, Qubit_pos)
Quantum_Circuit.ry (Angle_Of_Rotation_In_Radians, Qubit_pos)
Quantum_Circuit.rz (Angle_Of_Rotation_In_Radians, Qubit_pos)

IV. MULTI-QUBIT GATES

A qubit's mathematical structure can be generalized to higher dimensions of quantum systems. **Any quantum state is a normalized vector in a complex vector space which ensures that on measurement, the total probability of all outcomes is one**. Currently, most quantum computers contain multiple qubits, making the knowledge of construction of the combined qubit system state given the states of the individual qubits, which is described using the tensor product operation (denoted by ⊗).

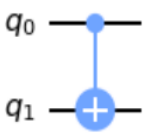

Figure 9: CNOT Gate

These tensor product operations give rise to **Multi-Qubit Gates**. The simplest of them is **CNOT**(also called **Controlled NOT**) Gate. It is a 2-qubit gate which can be used to entangle 2 qubits or disentangle already entangled qubits (whose maximally entangled states are called **Bell States**). Next is the Toffoli (also called **CCNOT**) Gate, which uses 3 qubits for its execution.

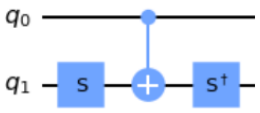

Figure 10: Quantum Circuit To Control Pauli-Y

A. Controlled NOT Gate

Controlled NOT (or **CNOT** or **CX**) Gate is a 2 qubit Gate where one qubit is used as **control** (denoted using •) & the other is used as **target** (denoted using ⊕). The target qubit gets flipped if and only if the control qubit is of basis state |1⟩, otherwise it is left unchanged (simply put, the target qubit acts like the classical XOR Gate). The circuit of CNOT Gate is given in *Fig. 9* & *Table 9* on the next page shows the result on both qubits when the gate is applied.

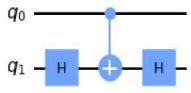

Figure 11: Quantum Circuit To Control Pauli-Z

Other names given to this gate are CX, Controlled Pauli-X & Feynman (after Richard Feynman, who developed the early notations of quantum gate

diagrams in 1986) Gates. The matrix to represent this gate is

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

and it is Hermitian in nature. These gates are essential in many quantum algorithms including Deutsch–Jozsa & Bernstein-Vazirani algorithms. To implement the CNOT Gate, following qiskit code snippet is used:

Quantum_Circuit.cnot(Control_qubit_pos, Target_qubit_pos)

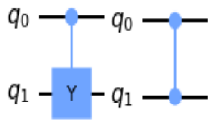

Figure 12: Controlled-Y & Controlled-Z Gates

Any single qubit gate can also be controlled using the CNOT Gate. A series of CNOT & other single qubit gate operations are needed to be performed to control a single qubit gate. To control **Pauli-Y** Gate, the operations shown in *Fig. 10* are performed, i.e., phase shift of $\pi/2$ on target qubit, a CNOT Gate & another phase shift of $-\pi/2$ on target qubit. This gate is also called **Controlled-Y** Gate. It is represented using the following matrix

$$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&-i\\0&0&i&0\end{bmatrix}$$

The gates given in the diagram can be used to create the circuit or this can also be done using only one instruction in qiskit:

Quantum_Circuit.cy(Control_qubit_pos, Target_qubit_pos)

Similarly for **Pauli-Z** Gate, a few logic gates can be used to control it. *Fig. 11* shows how; a Hadamard Gate on target qubit, followed by a CNOT Gate & terminated by another Hadamard Gate. Fig. 12 in the previous page shows the 2-qubit gates used to construct Controlled-Y & Controlled-Z respectively. Controlled Z Gate is represented using the following matrix:

$$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&-1\end{bmatrix}$$

Following qiskit instruction can also be used instead of creating the Controlled-Z circuit shown:

Quantum_Circuit.cz(Control_qubit_pos, Target_qubit_pos)

B. SWAP Gate

As it is intuitive, CNOT Gates are crucial in creating multi-qubit systems to perform complex tasks. They can be used for many other tasks also. Another crucial use of CNOT Gates is the **SWAP** Gate, which just swaps the quantum information of 2 qubits.

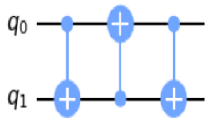

Figure 13: Quantum Circuit To Swap 2 Qubits

Table 5: S & T Gates on several initial qubit position

Initial State	Final State (S)	Final State (T)
$\|0\rangle$	$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$i\|1\rangle$	$\begin{bmatrix}0\\\frac{1}{\sqrt{2}}(1+i)\end{bmatrix}$
$\|+\rangle$	$\|+\rangle$	$\frac{1}{\sqrt{2}}\begin{bmatrix}0\\\frac{1}{\sqrt{2}}(1+i)\end{bmatrix}$
$\|-\rangle$	$\|-i\rangle$	$\frac{1}{\sqrt{2}}\begin{bmatrix}0\\\frac{-1}{\sqrt{2}}(1+i)\end{bmatrix}$
$\|i\rangle$	$\|-\rangle$	$\frac{1}{\sqrt{2}}\begin{bmatrix}0\\\frac{1}{\sqrt{2}}(i-1)\end{bmatrix}$
$\|-i\rangle$	$\|+\rangle$	$\frac{1}{\sqrt{2}}\begin{bmatrix}0\\\frac{-1}{\sqrt{2}}(i+1)\end{bmatrix}$

Table 6: Matrix of R_x Gate on different Angles Of Rotation

$R_x(\pi/6)$	$\frac{1}{4}\begin{bmatrix}(\sqrt{6}+\sqrt{2}) & -i(\sqrt{6}-\sqrt{2})\\-i(\sqrt{6}-\sqrt{2}) & (\sqrt{6}+\sqrt{2})\end{bmatrix}$	$R_x(\pi/4)$	$\frac{1}{2}\begin{bmatrix}\sqrt{\sqrt{2}+2} & -\sqrt{\sqrt{2}-2}\\-\sqrt{\sqrt{2}-2} & \sqrt{\sqrt{2}+2}\end{bmatrix}$

$R_x(\pi/3)$	$\frac{1}{2}\begin{bmatrix}\sqrt{3} & -i\\ -i & \sqrt{3}\end{bmatrix}$	$R_x(\pi/2)$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & -i\\ -i & 1\end{bmatrix}$

Table 7: Matrix of R_y Gate on different Angles Of Rotation

$R_y(\pi/6)$	$\frac{1}{4}\begin{bmatrix}(\sqrt{6}+\sqrt{2}) & (\sqrt{6}-\sqrt{2})\\ -(\sqrt{6}-\sqrt{2}) & (\sqrt{6}+\sqrt{2})\end{bmatrix}$	$R_y(\pi/4)$	$\frac{1}{2}\begin{bmatrix}\sqrt{2+\sqrt{2}} & -\sqrt{2-\sqrt{2}}\\ \sqrt{2-\sqrt{2}} & \sqrt{2+\sqrt{2}}\end{bmatrix}$
$R_y(\pi/3)$	$\frac{1}{2}\begin{bmatrix}\sqrt{3} & -1\\ 1 & \sqrt{3}\end{bmatrix}$	$R_y(\pi/2)$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & -1\\ 1 & 1\end{bmatrix}$

Table 8: Matrix of R_z Gate on different Angles Of Rotation

$R_z(\pi/6)$	$\frac{(\sqrt{6}+\sqrt{2})-i(\sqrt{6}-\sqrt{2})}{4}\cdot\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$	$R_z(\pi/4)$	$\frac{(\sqrt{2+\sqrt{2}})-(\sqrt{2-\sqrt{2}})}{2}\cdot\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$
$R_z(\pi/3)$	$\frac{\sqrt{3}-i}{2}\cdot\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$	$R_z(\pi/2)$	$\frac{1-i}{\sqrt{2}}\cdot\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$

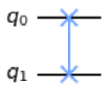

Figure 14: SWAP Gate

It is created by putting a reversed CNOT Gate in between 2 CNOT Gates. *Fig. 13* shows 1 possible way of doing it. The other way is only to invert each CNOT gate (i.e., to make each control bit the target bit and vice versa).

The matrix used to represent the SWAP Gate is

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

and in circuit diagram notation, it is represented using the symbol in *Fig. 14*. This gate can be represented in qiskit using the following code snippet:

Quantum_Circuit.swap(Qubit1_pos, Qubit2_pos)

C. Toffoli Gate

The **Toffoli Gate** or **CCNOT (Controlled-Controlled-NOT)** Gate (named after Tommaso Toffoli who invented this gate), is a 3-qubit universal reversible quantum logic gate (i.e., any classical reversible logic circuit can be constructed using this gate. It contains 2 control & 1 target qubit and performs Pauli-X operating on the target qubit iff (if & only if) both the control states are in basis state |1⟩. So, the target qubit's final state gets equal to AND of the control qubits (When the target qubit is initially at basis state |0⟩, otherwise NAND of the basis states of the control qubit occurs). This can be done by a comparatively bigger quantum circuit given in *Fig. 15* but can also be done using a single gate shown in *Fig. 16*, both of which are given on the next page.

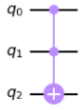

Figure 15: Toffoli Gate

A lot of things that work for CNOT Gate, work for the Toffoli Gate as well. The gate can be represented using the following 8x8 matrix:

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

The following instruction in qiskit is used to put a Toffoli Gate on a quantum circuit:

Quantum_Circuit.toffoli(Control1_qubit, Control2_qubit, Target_qubit)

We can also use the concept of Toffoli Gates for other gates, such as the **Fredkin** (or **Controlled-SWAP** or **CSWAP**) **Gate**, a 3-qubit gate with 1 control & 2 target qubits, which swaps the 2 target qubits iff the control qubit is in basis state |1⟩. The gate is represented using the following matrix:

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

The following instruction in qiskit is used to put a CSWAP Gate on a quantum circuit:

Quantum_Circuit.cswap(Control_qubit, Target1_qubit, Target2_qubit)

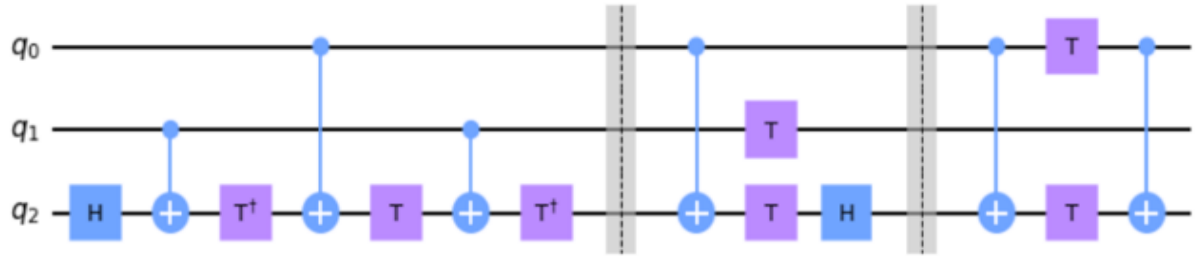

Figure 16: Extensive Quantum Circuit To Create Toffoli Gate

V. CUSTOMIZATION

Below given is a random unitary matrix generated using *python* which can act on 2 qubits (a qubit vector of 2 qubits can be considered as

$$\begin{bmatrix}\alpha\\ \beta\\ \gamma\\ \delta\end{bmatrix}$$

α being the probability of qubit state |00⟩ being read on measuring, β being the probability of |01⟩, γ being the probability of |10⟩ & δ being the probability of |11⟩) & would return a new qubit vector for the 4 possible combinations of those qubits.

Coupling it with the fact that matrix-multiplication of multiple unitary matrices also produces a unitary matrix as a result, it can be conclude that **even the largest of quantum circuits can also be represented using a unitary matrix**. A unitary matrix **U** can be represented as a gate in qiskit using the following code snippet:

```
U_gate = UnitaryGate(Unitary_Matrix)
Quantum_Circuit.append(U_gate,
[List_Of_Qubit_Positions])
```

$$\begin{bmatrix}(-0.523053-0.456867i) & (0.334272+0.046708i) & (-0.054205-0.010454i) & (-0.179382+0.607078i)\\ (-0.349395-0.509972i) & (-0.198202+0.016963i) & (0.053563-0.459960i) & (0.381165-0.467505i)\\ (-0.150407-0.261038i) & (-0.586501+0.041197i) & (-0.240314+0.702790i) & (0.107837+0.016244i)\\ (-0.198604-0.073030i) & (-0.032696-0.706937i) & (0.450475+0.167119i) & (-0.411814-0.232267i)\end{bmatrix}$$

VI.CONCLUSION

Building block of all quantum algorithms, quantum gates are fundamental at every step of quantum computation and information. With such a wide variety of gates provided by circuit libraries like the one in ***qiskit*** helps quantum computing enthusiasts tweak with IBM's quantum processors from any part of the world.

Researchers are now focusing on more complex problems as technology is advancing, quantum computing is also helping in solving complex problems like protein modelling by using physics of quantum mechanics.

There are several key challenges right now. First being unclear on selecting appropriate approach for a quantum project implementation, as there are different approaches. As it costs high to create quantum circuits, even few trials and errors of different approaches would cost heavily in both time and finance. Secondly, for a better stability and control of qubits, organizations like IBM keep the temperature of their quantum processors at very low levels (15 milliKelvin) and that results in zero ambient noise or heat that would excite the superconducted qubits and such factors require huge investments in state-of-the-art technologies.

Despite the setbacks and limitations in quantum computing and information, the field is gradually increasing with more talent pooling in to contribute to quantum information research. This naturally started bringing more innovation in better quantum gates and algorithms which resulted in refined results, freer of errors.

REFERENCE

[1] Travis S. Humble, Himanshu Thapliyal, Edgard Mu˜noz-Coreas, Fahd A. Mohiyaddin, Ryan S. Bennink, "Quantum Computing Circuits and Devices", arXiv:1804.10648v1, [quant-ph] 27 Apr 2018

[2] Jean-Luc Brylinski, Ranee Brylinski, "Universal Quantum Gates", arXiv:quant-ph/0108062v1 13 Aug 2001
[3] A. M. Krol, A. Sarkar, I. Ashraf, Z. Al-Ars, K. Bertels, "Efficient decomposition of unitary matrices in quantum circuit compilers", arXiv:2101.02993v1 [quant-ph] 8 Jan 2021
[4] J. H. Plantenberg, P. C. de Groot, C. J. P. M. Harmans, J. E. Mooij, "Demonstration of controlled-NOT quantum gates on a pair of superconducting quantum bits", Vol 447|14 June 2007| doi:10.1038/nature05896
[5] David P. DiVincenzo, "Two-bit gates are universal for quantum computation", arXiv:cond-mat/9407022v1 5 Jul 1994
[6] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John Smolin, Harald Weinfurter, "Elementary gates for quantum computation", arXiv:quant-ph/9503016v1 23 Mar 1995
[7] Ropa Roy, Asoke Nath, "Introduction to Quantum Gates : Implementation of Single and Multiple Qubit Gates", https://doi.org/10.32628/CSEIT217697
[8] Abhijith J.∗, Adetokunbo Adedoyin, John Ambrosiano, Petr Anisimov, William Casper, Gopinath Chennupati, Carleton Coffrin, Hristo Djidjev, David Gunter, Satish Karra, Nathan Lemons, Shizeng Lin, Alexander Malyzhenkov, David Mascarenas, Susan Mniszewski, Balu Nadiga, Daniel O'Malley, Diane Oyen, Scott Pakin, Lakshman Prasad, Randy Roberts, Phillip Romero, Nandakishore Santhi, Nikolai Sinitsyn, Pieter J. Swart, James G. Wendelberger, Boram Yoon, Richard Zamora, Wei Zhu, Stephan Eidenbenz, Andreas Bärtschi, Patrick J. Coles, Marc Vuffray, Andrey Y. Lokhov, "Quantum Algorithm Implementations for Beginners", arXiv:1804.03719v3 [cs.ET] 27 Jun 2022
[9] Pankaj Agrawal, Arun Pati, "Perfect Teleportation and Superdense Coding With W-States", http://arxiv.org/abs/quant-ph/0610001v1
[10] Travis S. Humble, Himanshu Thapliyal, Edgard Mu˜noz-Coreas, Fahd A. Mohiyaddin, Ryan S. Bennink, "Quantum Computing Circuits and Devices", arXiv:1804.10648v1 [quant-ph] 27 Apr 2018
[11] "Future of Quantum Computing in 2022: In-Depth Guide", https://research.aimultiple.com/future-of-quantum-computing/