

Project Report: A Comparative Study of Quantum Logic Gates Suited For Current Algorithms

Aarav Ratra

B. Tech Engineering Physics, Indian Institute of Technology, Roorkee

Supervisor: Dr. Ajay Singh, Physics Department, IIT Roorkee

Table of Contents

ACKNOWLEDGEMENT	4
ABSTRACT.....	5
REVIEW OF BASICS: DEFINITION OF A QUBIT.....	6
REVIEW OF THE QUANTUM HARMONIC OSCILLATOR [2]	6
SIMULATION OF THE ENERGY LEVELS OF A QHO USING MATLAB	11
QUANTUM MECHANICAL TREATMENT OF AN LC OSCILLATOR – SIMILARITIES [4].....	14
ANHARMONIC OSCILLATOR (BRIEF OVERVIEW)	17
TRANSMON QUBIT [4]	17
SIMULATING 1-D POTENTIALS USING MATLAB [3]	19
BLOCH SPHERE REPRESENTATION OF THE QUBIT	22
DENSITY MATRIX REPRESENTATION OF THE STATE OF A SINGLE QUBIT [1].....	23
A SHORT REVIEW ON CLASSICAL LOGIC GATES	24
INTRODUCTION TO QUANTUM GATES AS ROTATIONS ON THE BLOCH SPHERE	27
PAULI GATES – X, Y, Z.....	28
Pauli-X Gate.....	28
Pauli-Z Gate	28
Pauli-Y Gate.....	29
OTHER IMPORTANT SINGLE-QUBIT GATES	30
The Hadamard Gate (H).....	30
Phase Shift Gates: S and T	31
SOME INTERESTING RELATIONS BETWEEN GATES	31
MULTIPLE QUBIT GATES.....	32
Controlled-NOT Gate:	32
Controlled-Z Gate:	34
Toffoli Gate.....	35
SWAP Gate.....	35
SUMMARY OF QUANTUM GATES.....	37
SIMULATION OF BASIC QUANTUM GATES VIA MATLAB.....	39
SINGLE QUBIT GATE ACTION	39
1. PAULI-X.....	39
2. PAULI-Y.....	39
3. PAULI-Z.....	39
4. HADAMARD GATE	40
5. S GATE	40
6. T GATE.....	40
VERIFICATION OF BASIC GATE IDENTITIES.....	41
ACTION OF 2-QUBIT GATES	41
CNOT-GATE.....	41
CZ GATE	42
SWAP GATE	43
3 QUBIT GATE: TOFFOLI GATE.....	43
VERIFICATION OF GATE CONSTRUCTION VIA ROTATION, PHASE AND CNOT GATES.....	44
UNIVERSAL SETS OF GATES.....	47
Family 1: $R_x(\theta)$, $R_y(\theta)$, $R_z(\theta)$, $P(\phi)$, CNOT [6].....	47
Family 2: Toffoli (CCX), H	48
Family 3: CNOT, H, S and T [6]	51
The DEUTSCH Gate [6][9][8]	51

The BARENCO Gate [6][10]	52
SOME OTHER MATLAB SIMULATIONS	53
VERIFICATION OF CLASSIC ACTION USING TOFFOLI GATES	53
AND GATE for AB = 00, 01, 10, 11	54
NAND GATE for AB = 00, 01, 10, 11	54
NOT Gate for A = 0, 1	54
OR Gate for AB = 00, 01, 10, 11	55
XOR (CNOT) Gate for AB = 00, 01, 10, 11	55
GENERATION OF TOFFOLI GATE USING CNOT, H, T AND T† GATES	55
CONCLUSION	58
REFERENCES	59

Acknowledgement

I extend my heartfelt appreciation to Prof. Ajay Singh, my supervisor, for granting me the invaluable opportunity to delve into the captivating realm of this research topic. Under his expert guidance and unwavering support, I have gained profound insights into the boundless potential this field holds. I am profoundly grateful for his indispensable mentorship, which has been instrumental in shaping the course of this project.

- **Aarav Ratra**

Abstract

With this project I aim to understand how Quantum Gates are implemented and compare them. The report starts off with the basic definition of the Qubit and a review of the basic quantum mechanical treatment of simple systems like the Quantum Harmonic Oscillator, followed by some simulations on MATLAB to obtain the energy levels and the eigenstates for the Hamiltonian. After that, I did an analysis of how such a system can be replicated using circuit electrodynamics through quantum mechanical treatment of an L-C Circuit, not diving deep into the mathematics, and focusing more on the intuition. We also introduced ourselves to the term Zero-Point-Fluctuation. The next step was to study about the Anharmonic Oscillator and how it translates to a Transmon Qubit, while going into a brief overview of the energy of the Josephson Junction Oscillator. However, instead of diving into the exact mathematical formulation which requires knowledge of the Rotating Wave Approximation and 1st Order Perturbation Theory, I decided to numerically calculate the eigenstates and energy eigenvalues and obtained suitable results.

After giving ourselves a background, I shifted my focus towards basic gates. I started off with an introduction to the Bloch Sphere and hence started with the description of simple qubit gates as Bloch Sphere rotations. In between we had a quick detour towards classical gates. We then went on to exploration of multi-Qubit gates and various gate compositions, followed by some simulations on MATLAB. This was followed by examining various families of universal sets of gates, including the DEUTSCH and BARENCO Gates.

Review of Basics: Definition of a Qubit

A qubit is the fundamental unit of quantum information. It serves as the fundamental building component for systems that use quantum processing and communication. The term "qubit" is a combination of the words "quantum" and "bit", with "bit" standing for a traditional binary unit of information.

Unlike classical bits, a qubit can exist in the superposition of two states (i.e., 0 and 1), hence representing both simultaneously. This allows it to process information differently. Properties of quantum mechanics such as superposition and entanglement followed by these qubits would allow us to develop algorithms which outperform their classical counterparts.

Qubits can be implemented using various physical systems. Ideally, we would require a two-level quantum system. While a two-level system theoretically does not exist, we can isolate any two levels in various systems. Hence, qubits have been implemented using single atoms, trapped ions, photons, superconducting qubits etc.

Review of the Quantum Harmonic Oscillator [2]

One of the simplest systems which we deal with in Quantum Mechanics is that of the Linear Harmonic Oscillator.

The Hamiltonian of the 1-D Linear Harmonic Oscillator can be written in this fashion:

$$\hat{H} = \frac{\hat{p}^2}{2m} + \frac{1}{2}m\omega^2 \hat{x}^2$$

Where the momentum operator $\hat{p} = -i\hbar \frac{\partial}{\partial x}$.

The commutator of x and p is described as:

$$[x, p] = i\hbar$$

Hence, on further simplification, we can obtain the Hamiltonian to be of the following form:

$$\hat{H} = \hbar\omega \left[\left(\frac{-i\hat{p} + m\omega\hat{x}}{\sqrt{2m\hbar\omega}} \right) \left(\frac{i\hat{p} + m\omega\hat{x}}{\sqrt{2m\hbar\omega}} \right) + \frac{1}{2} \right]$$

Which can be further written as:

$$\hat{H} = \hbar\omega \left[\hat{a}^\dagger \hat{a} + \frac{1}{2} \right]$$

Where $\hat{a}^\dagger = \frac{-i\hat{p} + m\omega\hat{x}}{\sqrt{2m\hbar\omega}}$ and $\hat{a} = \frac{i\hat{p} + m\omega\hat{x}}{\sqrt{2m\hbar\omega}}$.

The operators \hat{a}^\dagger and \hat{a} are known as ‘creation’ and ‘annihilation’ operators.

The following commutators are useful:

$$\begin{aligned}
[\hat{a}, \hat{a}^\dagger] &= 1 \\
[\hat{a}, \hat{a}^\dagger \hat{a}] &= \hat{a} \\
[\hat{a}^\dagger, \hat{a}^\dagger \hat{a}] &= -\hat{a}^\dagger \\
[\hat{a}, \hat{H}] &= \hbar\omega\hat{a} \\
[\hat{a}^\dagger, \hat{H}] &= -\hbar\omega\hat{a}^\dagger
\end{aligned}$$

Let us say we have an eigenstate $|\psi_n\rangle$ such that $\hat{H}|\psi_n\rangle = E_n|\psi_n\rangle$.

We can make the following observations:

$$\hat{H}(\hat{a}^\dagger|\psi_n\rangle) = \hbar\omega\left[\hat{a}^\dagger\hat{a} + \frac{1}{2}\right]\hat{a}^\dagger|\psi_n\rangle$$

Which on further simplification (not shown here) gives

$$\hat{H}(\hat{a}^\dagger|\psi_n\rangle) = (E_n + \hbar\omega)(\hat{a}^\dagger|\psi_n\rangle).$$

Hence, we can say that $\hat{a}^\dagger|\psi_n\rangle$ would also be an eigenstate of the operator \hat{H} .

It can also be found using similar techniques that $\hat{H}(\hat{a}|\psi_n\rangle) = (E_n - \hbar\omega)(\hat{a}|\psi_n\rangle)$

Hence, we can say that $\hat{a}|\psi_n\rangle$ would also be an eigenstate of the operator \hat{H} .

With the above inferences, operators \hat{a}^\dagger and \hat{a} are given their names as ‘creation’ and ‘annihilation’ operators and are collectively known as ladder operators.

If we let $|0\rangle$ be the ground state, by definition, applying the annihilation operator on the ground state (which has the minimum possible energy) would be 0. Hence,

$$\hat{a}|0\rangle = 0$$

Hence, on evaluating the Hamiltonian:

$$\hat{H}|0\rangle = \hbar\omega\left[\hat{a}^\dagger\hat{a} + \frac{1}{2}\right]|0\rangle = \hbar\omega\hat{a}^\dagger(\hat{a}|0\rangle) + \frac{1}{2}\hbar\omega|0\rangle = 0 + \frac{1}{2}\hbar\omega|0\rangle$$

Giving us a zero-point energy of $\frac{1}{2}\hbar\omega$.

As we have seen, the creation and annihilation operators return different eigenstates with different eigenvalues. The creation operator \hat{a}^\dagger acting on state $|n\rangle$ returns an eigenstate with an energy $E_n + \hbar\omega$, which we can call the next energy level $|n + 1\rangle$. Similarly, the annihilation operator acting on state $|n\rangle$ returns an eigenstate with an energy $E_n - \hbar\omega$ which we call the lower energy level $|n - 1\rangle$.

And hence, energy of n^{th} eigenstate can be obtained to be:

$$E_n = \left(n + \frac{1}{2}\right) \hbar\omega$$

Mathematically, we can express them as follows:

$$\begin{aligned}\hat{a}^\dagger |n\rangle &= \lambda_n |n + 1\rangle \\ \hat{a} |n\rangle &= \mu_n |n - 1\rangle \quad (0 \text{ for } n = 0)\end{aligned}$$

We can manipulate the second equation as follows:

$$\langle n - 1 | \hat{a} | n \rangle = \mu_n \Rightarrow \langle n | \hat{a}^\dagger | n - 1 \rangle = \mu_n^*$$

Now,

$$\langle n | \hat{a}^\dagger \hat{a} | n \rangle = \langle n | \hat{a}^\dagger \mu_n | n - 1 \rangle = \mu_n \langle n | \hat{a}^\dagger | n - 1 \rangle = \mu_n \mu_n^* = |\mu_n|^2$$

Now we compute $\langle n | \hat{H} | n \rangle$

$$\langle n | \hat{H} | n \rangle = \left\langle n \left| \hat{a}^\dagger \hat{a} + \frac{1}{2} \right| n \right\rangle \hbar\omega = \hbar\omega \left(\langle n | \hat{a}^\dagger \hat{a} | n \rangle + \frac{1}{2} \langle n | n \rangle \right) = \hbar\omega \left(|\mu_n|^2 + \frac{1}{2} \right)$$

Clearly this indicates that $\mu_n = \sqrt{n}$. We can also similarly obtain that $\lambda_n = \sqrt{n + 1}$. Hence, we can conclude that:

$$\begin{aligned}\hat{a}^\dagger |n\rangle &= \sqrt{n + 1} |n + 1\rangle \\ \hat{a} |n\rangle &= \sqrt{n} |n - 1\rangle \quad (0 \text{ for } n = 0)\end{aligned}$$

With this, we can also introduce the number operator $\hat{n} = \hat{a}^\dagger \hat{a}$.

$$|n\rangle = \sqrt{n} \hat{a}^\dagger |n - 1\rangle = \sqrt{n} \sqrt{n - 1 + 1} |n - 1 + 1\rangle = \sqrt{n^2} \langle n | n \rangle = n$$

Using this, we can arrive to a matrix formalism for the Quantum Harmonic Oscillator.

Let us evaluate the matrix element for the annihilation operator \hat{a} :

$$\langle i|\hat{a}|j\rangle = \langle i|\sqrt{j}|j-1\rangle = \sqrt{j}\langle i|j-1\rangle = \sqrt{j}\delta_{i,j-1}$$

Similarly, for creation operator \hat{a}^\dagger :

$$\langle i|\hat{a}^\dagger|j\rangle = \langle i|\sqrt{j+1}|j+1\rangle = \sqrt{j+1}\langle i|j+1\rangle = \sqrt{j+1}\delta_{i,j+1}$$

The matrix form, hence, comes out to be:

$$\hat{a}^\dagger = \begin{bmatrix} 0 & 0 & 0 & \\ \sqrt{1} & 0 & 0 & \dots \\ 0 & \sqrt{2} & 0 & \\ 0 & 0 & \sqrt{3} & \ddots \\ \vdots & & & \end{bmatrix}$$

$$\hat{a} = \begin{bmatrix} 0 & \sqrt{1} & 0 & 0 \\ 0 & 0 & \sqrt{2} & 0 & \dots \\ 0 & 0 & 0 & \sqrt{3} & \\ \vdots & & & & \ddots \end{bmatrix}$$

Hence, on further simplification, the matrix form for the Hamiltonian is as follows:

$$H = \begin{bmatrix} \frac{1}{2}\hbar\omega & 0 & 0 & & \\ 0 & \frac{3}{2}\hbar\omega & 0 & & \dots \\ 0 & 0 & \frac{5}{2}\hbar\omega & & \\ & \vdots & & \frac{7}{2}\hbar\omega & \\ & & & & \ddots \end{bmatrix}$$

Clearly, this is a diagonal matrix which has all the energy eigenvalues along its diagonal.

Now, we attempt to solve for the wave functions $\psi_n(x)$ for each x .

A good starting point is to use the relation:

$$\hat{a}|0\rangle = 0$$

We now express \hat{a} in terms of the position and momentum operators and write $|0\rangle$ as $\psi_0(x)$.

$$\hat{a} = \frac{i\hat{p} + m\omega\hat{x}}{\sqrt{2m\hbar\omega}} \left(\text{note that } \hat{p} = -i\hbar \frac{\partial}{\partial x} \right)$$

On further simplification we get

$$\hat{a} = \frac{1}{\beta\sqrt{2}} \frac{d}{dx} + \frac{\beta}{\sqrt{2}} \hat{x} \left(\text{where } \beta = \sqrt{\frac{m\omega}{\hbar}} \right)$$

We can similarly also say that

$$\hat{a}^\dagger = -\frac{1}{\beta\sqrt{2}} \frac{\partial}{\partial x} + \frac{\beta}{\sqrt{2}} \hat{x}$$

Substituting $\psi_0(x)$ in $\hat{a}|0\rangle = 0$, we get the following differential equation:

$$\frac{1}{\beta\sqrt{2}} \frac{d\psi_0(x)}{dx} + \frac{\beta}{\sqrt{2}} x\psi_0(x) = 0$$

This is a simple first-order differential equation. We can obtain a solution:

$$\psi_0(x) = Ae^{-\frac{1}{2}\beta^2 x^2}$$

The constant A can be obtained by normalising $\psi_0(x)$; hence we have our normalised solution as:

$$\psi_0(x) = \left(\frac{\beta}{\sqrt{\pi}} \right)^{\frac{1}{2}} e^{-\frac{1}{2}\beta^2 x^2}$$

To obtain the other states, we can make use of the creation operator \hat{a}^\dagger .

$$\psi_n(x) = \hat{a}^\dagger \psi_{n-1}(x)/\sqrt{n}$$

Hence arriving to a general relation using recursion:

$$\psi_n(x) = (\hat{a}^\dagger)^n \psi_0(x)/\sqrt{n!}$$

We can obtain a general expression in terms of Hermite Polynomials, however, for the sake of our discussion, I have chosen not to dive into those. We would be obtaining the wave functions through MATLAB which provides us with a lot of numerical tools.

Here, I have attempted to simulate the energy levels of the Linear Harmonic Oscillator using MATLAB. I attempted to obtain the Hamiltonian of the QHO via the matrix formalism.

```

5 %% Matrix Formalism
6
7 N = 10; %Number of energy levels we are restricting ourselves to for simplicity
8
9
10 %Defining creation and annihilation operators using matrices
11 a = diag(sqrt(1:N-1),1);
12 ad= diag(sqrt(1:N-1),-1);
13
14 num = ad*a; %Number operator = ad*a
15
16 hcr = 6.582119569E-16; %Reduced Planck's Const (in eV)
17 vbar = 300; %cm^-1
18 c = 2.99792458E10; %cm/s
19 w = 2*pi*vbar*c;
20 m = 6.2224e-26; %Arbitrary value chosen
21
22 Id = diag(ones(1,N)); %Identity Matrix
23
24 H = (num + Id/2)*hcr*w;
25 %X = sqrt(hcr/(2*m*w))*(ad + a);
26
27 fprintf('Given that  $\hbar\omega = 1.5f$  \n',hcr*w)
28 %eigenvectors and eigenvalues of the Energy Operator
29 fprintf('The energy eigenvalues are as follows:')
30 H_eigenvalues = eig(H)
31 fprintf('The Hamiltonian Matrix is as described:\n')
32 disp(H)
33

```

Result:

```

Given that  $\hbar\omega = 0.03720$ 
The energy eigenvalues are as follows:
H_eigenvalues =

0.0186
0.0558
0.0930
0.1302
0.1674
0.2046
0.2418
0.2790
0.3162
0.3534

The Hamiltonian Matrix is as described:

0.0186      0      0      0      0      0      0      0      0      0
      0      0.0558      0      0      0      0      0      0      0      0
      0      0      0.0930      0      0      0      0      0      0      0
      0      0      0      0.1302      0      0      0      0      0      0
      0      0      0      0      0.1674      0      0      0      0      0
      0      0      0      0      0      0.2046      0      0      0      0
      0      0      0      0      0      0      0.2418      0      0      0
      0      0      0      0      0      0      0      0.2790      0      0
      0      0      0      0      0      0      0      0      0.3162      0
      0      0      0      0      0      0      0      0      0      0.3534

```

Now, I have attempted to obtain the wave functions. Note that computation of the number operator on my system required more computer power at this stage of the project and hence led to repetitive crashes. Hence, I have only demonstrated the number operator for smaller values of $n = 0, 1, 2$.

```

34 %%
35 % On obtaining the eigenvalues of the above, we can attempt to show the
36 % energy levels and wavefunctions.
37 %We first solve for 0th eigenstate, then we use the creation operator to
38 %derive the rest.
39
40 %a|0> = 0 => we obtain  $\psi_0$  through a first order differential equation.
41 %I shall be using the Symbolic Math library of MATLAB which allows us to
42 %deal with functions and derivatives in a more convenient manner.
43 syms x;
44 beta = sqrt(m*w/hcr)
45
46 %Declaring ad_ and a_ as anonymous symbolic functions
47 ad_ = @(ps) -diff(ps)/(beta*sqrt(2)) + beta*x*ps/sqrt(2);
48 a_ = @(ps) diff(ps)/(beta*sqrt(2)) + beta*x*ps/sqrt(2);
49 n_ = @(ps) ad_(a_(ps)); %Number Operator
50
51 n_avg = @(ps) int(n_(ps)/ps,-1000,1000)/2000 ; %I have computed the average
52 %Note: This function requires more computational resources than my local
53 %system. This does not really return suitable value for  $n \geq 3$  within
54 %feasable time and ends up crashing matlab. Hence, i shall only demonstate
55 %it for  $n = 0, 1, 2$ 
56
57 psi_0 = sqrt(beta/sqrt(pi))*exp(-0.5*beta^2*x^2); %Manually calculated
58

```

```

59 %%
60 %I shall be running a loop for eigenstates we are dealing with. I would be
61 %dealing with only the first 5 eigenstates, hence, I am reassigning N = 5
62
63 N=5;
64 p = psi_0;
65
66 subplot(N+1,1,N+1)
67 xline(0)
68 yline(0)
69 fplot (psi_0,[-.2,.2])
70
71
72 for i=1:N
73     if(i<=3)
74         n = eval(n_avg(p))
75         disp('Above n evaluated using the operator')
76     else
77         n = i-1
78     end
79     E = (n +1/2);
80     p = ad_(p)/sqrt(i);
81     subplot(N+1,1,N+1-i)
82     fplot (p,[-.2,.2])
83     xlabel('x')
84     ylabel('Ψ')
85     title('n='+string(n)+' , E = '+string(E))
86 end

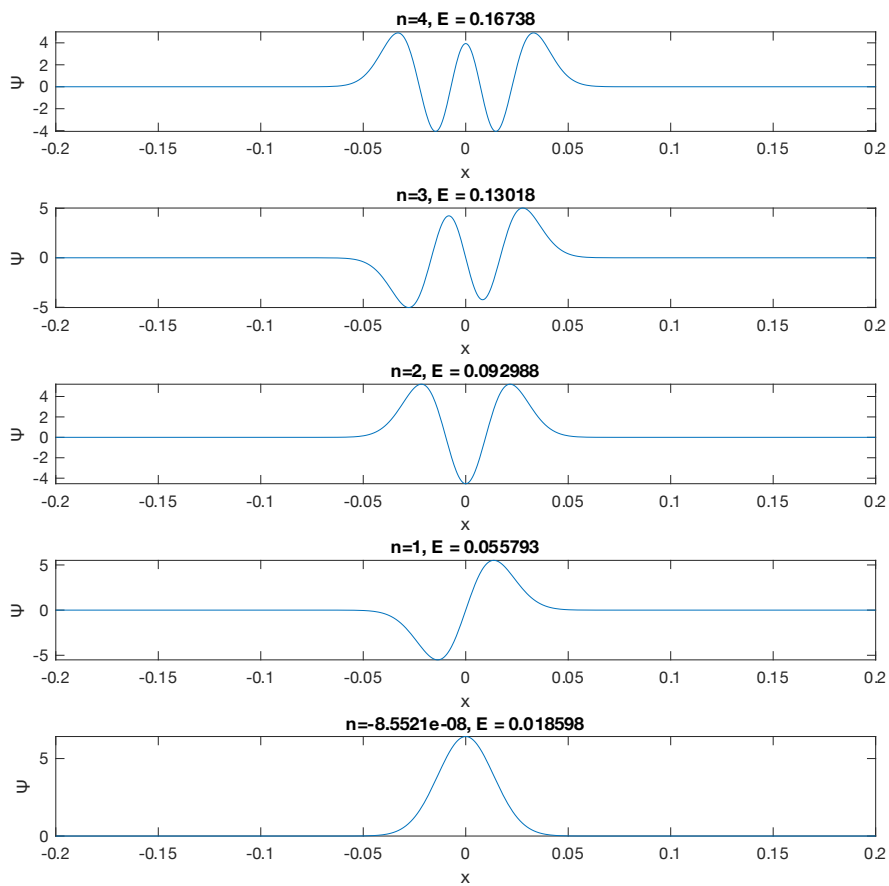
```

Result:

```

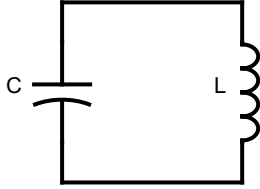
n =
-8.5521e-08
Above n evaluated using the operator
n =
1.0000
Above n evaluated using the operator
n =
2.0000
Above n evaluated using the operator
n =
3
n =
4

```



Quantum Mechanical Treatment of an LC Oscillator – Similarities [4]

It is well established that the LC Circuit is also governed by a 2nd order differential equation similar to a Classical Linear Harmonic Oscillator. However, it would make sense go dive a bit deeper to understand how we can do a quantum mechanical treatment of this system.



We start off by reviewing the very foundation of current electricity, i.e., Kirchhoff's laws. However, we will not be using them in the form which we have encountered in normal circuit electrodynamics, but will try to phrase them in terms of two quantities i.e., charge and flux.

1. Kirchhoff's Current Law: $\sum \dot{Q}_i = 0$
2. Kirchhoff's Voltage Law: $\sum \dot{\phi}_i = 0$

From the above, the following relations are apparent:

$$\begin{aligned}\dot{\phi}_c - \dot{\phi}_L &= 0 \Rightarrow \phi_c = \phi_L (= \phi) \\ \dot{Q}_c + \dot{Q}_L &= 0 \Rightarrow C\ddot{\phi} + \frac{\phi}{L} = 0\end{aligned}$$

This ultimately ends in an equation which is of familiar form:

$$\ddot{\phi} = -\frac{1}{LC}\phi = -\omega^2\phi$$

Where $\omega = 1/\sqrt{LC}$.

This bears similarity to the equation for a classical harmonic oscillator i.e., $\ddot{x} = -\omega^2 x$

Now, we attempt to write the Hamiltonian of the system:

$$H = \varepsilon_c + \varepsilon_L = \frac{Q^2}{2C} + \frac{1}{2}LI^2 = \frac{Q^2}{2C} + \frac{\phi^2}{2L}$$

Now, keeping in mind that $Q = C\dot{\phi}$, Let us compare this with the Hamiltonian of the Classical Oscillator:

$$H = \frac{p^2}{2m} + \frac{1}{2}kx^2 = \frac{p^2}{2m} + \frac{1}{2}m\omega^2 x^2$$

Hence, we can come up with the following correspondence:

$$\phi \Leftrightarrow x ; Q \Leftrightarrow p ; m \Leftrightarrow C ; \frac{1}{L} \Leftrightarrow k$$

We also define: $\omega = \frac{1}{\sqrt{LC}}$, $Z_0 = L/C$

Now, we need to study the Quantum Equivalent of the same. However, instead of proving the Hamiltonian to be of a similar form, we shall go ahead with the simple result which comes from our intuition:

$$\hat{H} = \frac{\hat{Q}^2}{2C} + \frac{\hat{\phi}^2}{2L}$$

While I will not be going deep into proving this, I will mention a result I had encountered which relates the Commutator from Quantum Mechanics and the Poisson-Bracket from Classical Mechanics.

$$\{O, O'\} = \frac{1}{i\hbar} [O, O']$$

Which works in the case of the position and momentum operators. This also explains how the term \hbar comes in the next expression.

With the similarities to the QHO, the Hamiltonian will end up taking a form very similar to what we have encountered in the past:

$$\hat{H} = \hbar\omega \left[\hat{a}^\dagger \hat{a} + \frac{1}{2} \right]$$

Where,

$$\hat{a} = \frac{iZ\hat{Q} + \hat{\phi}}{\sqrt{2\hbar Z}}$$

$$\hat{a}^\dagger = \frac{-iZ\hat{Q} + \hat{\phi}}{\sqrt{2\hbar Z}}$$

Needless to say, the energy levels generated would be similar to those of a classical harmonic oscillator.

Now, before diving further, it is important to introduce ourselves to the term ‘Zero Point Fluctuation’. This is defined as the variance in flux/charge for eigenstate $|0\rangle$.

To calculate it, we first express the operators $\hat{\phi}$ and \hat{Q} in terms of \hat{a} and \hat{a}^\dagger :

$$\hat{\phi} = \sqrt{\hbar Z/2} (\hat{a} + \hat{a}^\dagger)$$

$$\hat{Q} = -i\sqrt{\hbar/2Z} (\hat{a} - \hat{a}^\dagger)$$

The expectation values are hence calculated:

$$\langle 0 | \hat{\phi} | 0 \rangle = \hbar Z/2 (\langle 0 | \hat{a} | 0 \rangle + \langle 0 | \hat{a}^\dagger | 0 \rangle) = \hbar Z/2 (\langle 0 | (0) + (0) | 0 \rangle) = 0$$

$$\langle 0 | \hat{Q} | 0 \rangle = \hbar/2Z (\langle 0 | \hat{a} | 0 \rangle - \langle 0 | \hat{a}^\dagger | 0 \rangle) = \hbar/2Z (\langle 0 | (0) - (0) | 0 \rangle) = 0$$

This was probably expected. Since the wave-function for eigenstate $|0\rangle$ was obtained to be symmetric.

Hence, since the expectation value for both charge and flux operators turned out to be 0, the variance would be equal to the expectation of the square of the operators (i.e., the RMS).

$$\begin{aligned} (\phi_{ZPF})^2 &= \langle \hat{\phi}^2 \rangle = \hbar Z / 2 \langle (\hat{a} + \hat{a}^\dagger)^2 \rangle = \hbar Z / 2 \left(\langle \hat{a}^2 \rangle + \langle \hat{a}^{\dagger 2} \rangle + \langle \hat{a}^\dagger \hat{a} \rangle + \langle \hat{a} \hat{a}^\dagger \rangle \right) \\ &= \hbar Z / 2 \left(\langle 0 | \hat{a}^2 | 0 \rangle + \langle 0 | \hat{a}^{\dagger 2} | 0 \rangle + \langle 0 | \hat{a}^\dagger \hat{a} | 0 \rangle + \langle 0 | \hat{a} \hat{a}^\dagger | 0 \rangle \right) \\ &= \hbar Z / 2 (0 + \langle n | n + 2 \rangle + 0 + \langle 1 | 1 \rangle) = \hbar Z / 2 \end{aligned}$$

And similarly, $(Q_{ZPF})^2 = \hbar / 2 Z$

Hence, our we can rephrase operators $\hat{\phi}$ and \hat{Q} as:

$$\begin{aligned} \hat{\phi} &= \phi_{ZPF} (\hat{a} + \hat{a}^\dagger) \\ \hat{Q} &= -i Q_{ZPF} (\hat{a} - \hat{a}^\dagger) \end{aligned}$$

Till now we have seen how the Quantum Mechanical treatment of an LC Oscillator leads to energy levels like a QHO. However, it is important to understand that this system would not function as a qubit. For a qubit, we need to isolate two energy levels such that the system effectively acts like a two-level quantum system. However, here, we cannot isolate two levels as the transition frequency between subsequent levels are equal, hence none of the transitions are unique to two fixed energy levels.

To counter this, we shall now have a brief overview of the Anharmonic Oscillator and Josephson's Junctions. We shall explore these only at the surface level since a more advanced study would require the knowledge of perturbation theory and other topics, which would take too much time. Instead, we shall attempt to calculate the energy eigenvalues through numerical methods.

Anharmonic Oscillator (Brief Overview)

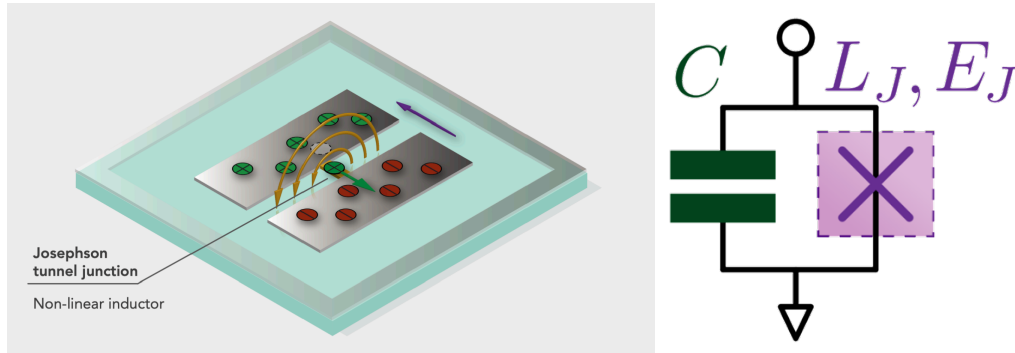
The anharmonic oscillator can be simply described as an oscillator where the potential has higher-order terms associated with it. For our study, we shall take the case where there is an additional quartic term. Hence, the resulting Hamiltonian will come out to be:

$$\hat{H} = \frac{\hat{p}^2}{2m} + \frac{1}{2}m\omega^2\hat{x}^2 - \lambda\hat{x}^4$$

Where λ is an appropriate constant.

Transmon Qubit [4]

The Transmon Qubit can be thought of as replacing the Inductor in a regular LC Oscillator with a Non-Linear Element known as the Josephson Junction.



(Image Source: Lectures 16-21 (Zlatko K. Minev), 2020 Qiskit Global Summer School on Quantum Computing and Quantum Hardware[4])

The energy for the Josephson Junction can be expressed in the following form:

$$\varepsilon_j = E_j \left(1 - \cos\left(\frac{\phi_j}{\phi_0}\right) \right) \approx E_j \left(\frac{\left(\frac{\phi_j}{\phi_0}\right)^2}{2!} - \frac{\left(\frac{\phi_j}{\phi_0}\right)^4}{4!} + H.O.T. \right)$$

Hence, if we attempt to write the Hamiltonian, we get:

$$H = \varepsilon_c + \varepsilon_j = \frac{Q^2}{2C} + E_j \left(\frac{\left(\frac{\phi_j}{\phi_0}\right)^2}{2!} - \frac{\left(\frac{\phi_j}{\phi_0}\right)^4}{4!} + \dots \right)$$

Substituting $E_j = \phi_0^2/L_j$ we get:

$$H = \frac{Q^2}{2C} + \frac{\phi_j^2}{2L_j} - \frac{\phi_j^4}{L_j 4!} + \dots$$

The term L_j will act as an effective inductance in the linear part of the Hamiltonian. The highlighted part is similar to the actual Quantum Harmonic Oscillator.

Clearly, if we ignore the higher order terms, we shall obtain a system which is similar to an anharmonic oscillator.

$$H = \frac{Q^2}{2C} + \frac{\phi_j^2}{2L_j} - \frac{\phi_j^4}{L_j 4!}$$

We shall now examine the energy levels of the anharmonic oscillator. I have used a numerical approach to approximate the energy levels of such systems.

Simulating 1-D Potentials using MATLAB [3]

Since solving the above equations mathematically can be considerably tiresome, we shall look at a general method for simulating arbitrary 1-D Potentials via method of difference. In this method, we use an approximation that the system has the boundary conditions:

$$\psi(0) = \psi(L) = 0$$

i.e., the particle is bounded from $x=0$ to $x=L$.

However, the systems we are interested in are unbounded. Hence, we assume L to be very large such that the lower states can be considered to be approximately unbounded.

For a 1-D potential, the Time Independent Schrodinger Equation is written as:

$$-\frac{\hbar^2}{2m} \frac{\partial^2 \psi}{\partial x^2} + V(x)\psi = E\psi$$

Let us use the convention $\hbar=1$, and non-dimensionalise the equation by expressing $x = L*y$,

$$-\frac{1}{2} \frac{d^2 \psi}{dy^2} + mL^2 V(y)\psi = mL^2 E\psi$$

Now, this is a second order differential equation. Note that the boundary conditions still apply, i.e., $\psi(y=0) = \psi(y=1) = 0$.

To facilitate our calculation, we divide the region from $y=0$ to $y=1$ to N equal discrete intervals Δy . We use a discrete approximation of the second derivative:

$$\frac{d^2 \psi}{dy^2} \bigg|_{y=j\Delta y} \approx \frac{\psi_{j+1} - 2\psi_j + \psi_{j-1}}{\Delta y^2}$$

Substituting this form, we get:

$$-\frac{1}{2} \left(\frac{\psi_{j+1} - 2\psi_j + \psi_{j-1}}{\Delta y^2} \right) + mL^2 V_j \psi_j = mL^2 E \psi_j$$

$$\Rightarrow -\frac{1}{2\Delta y^2} \psi_{j+1} + \left(\frac{1}{\Delta y^2} + mL^2 V_j \right) \psi_j - \frac{1}{2\Delta y^2} \psi_{j-1} = mL^2 E \psi_j \text{ (for } j = 1 \text{ to } N-1)$$

The above $N-1$ equations can be written in this matrix representation:

$$\begin{bmatrix} \frac{1}{\Delta y^2} + mL^2 V_1 & -\frac{1}{2\Delta y^2} & 0 & & & \\ -\frac{1}{2\Delta y^2} & \frac{1}{\Delta y^2} + mL^2 V_2 & -\frac{1}{2\Delta y^2} & \cdots & & 0 \\ 0 & -\frac{1}{2\Delta y^2} & \frac{1}{\Delta y^2} + mL^2 V_3 & & & \\ & \vdots & & \ddots & & \vdots \\ & 0 & & \cdots & \frac{1}{\Delta y^2} + mL^2 V_{N-1} & \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \vdots \\ \psi_{N-1} \end{bmatrix} = mL^2 E \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \vdots \\ \psi_{N-1} \end{bmatrix}$$

Hence, we can obtain the energy levels simply by obtaining the eigenvalues and eigenstates of the matrix on the LHS.

First, we shall test and obtain the energy levels for the Linear Quantum Harmonic Oscillator.

(Note that it is not correct to label H as the Hamiltonian. I have just used it as a variable name)

```
%% Part 1 : QHO
f1 = figure();
figure(f1)
N = 10000;
dy=1/N;
y = linspace(0,1,N+1);
L = 1000;
m=1;
k=1;

mL2V = m*L^2 * 0.5*k*(y-0.5).^2;

dmain = dy^(-2) + mL2V(2:N);
doff = ones(1,N-2)*-0.5/(dy^2);

mL2H = diag(dmain)+ diag(doff,1) + diag(doff,-1);
H = mL2H / (m*L^2);

E = eig(H);
V=D = eig(H);
subplot(2,2,1)
plot(y,mL2V/(m*L^2))
hold on
yline(E(1:20))
title('Energy Levels (compare to HO)')
xlabel('y')
ylabel('E')

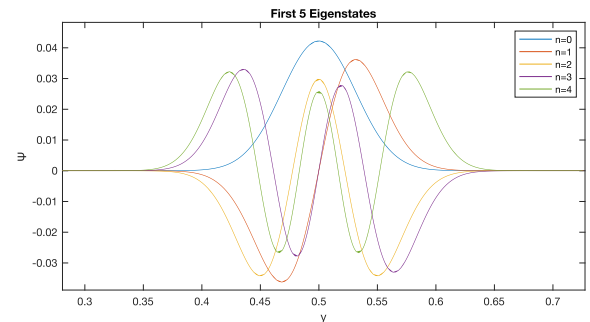
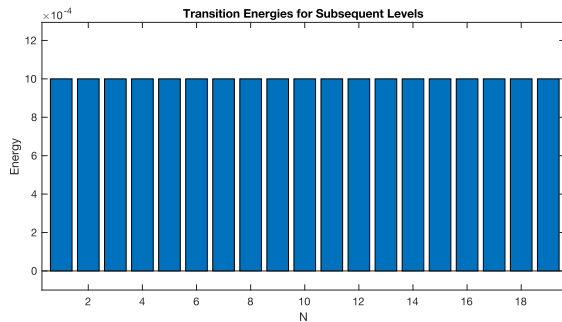
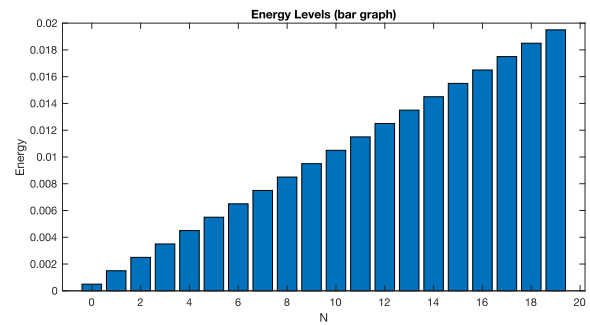
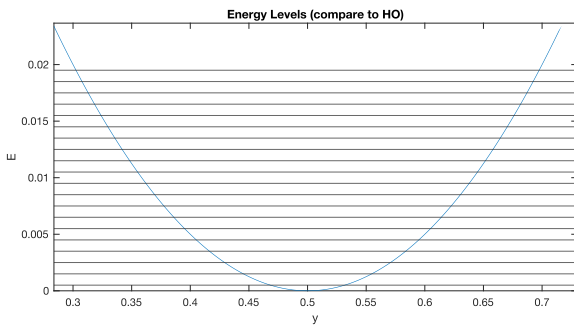
subplot(2,2,2)
bar(0:19,E(1:20))
title('Energy Levels (bar graph)')
xlabel('N')
ylabel('Energy')

Lines = [];
for i=1:19
    Lines = [Lines , E(i+1)-E(i)];
end
subplot(2,2,3)
bar(Lines)
title('Transition Energies for Subsequent Levels')
xlabel('N')
ylabel('Energy')

subplot(2,2,4)
xline(0)
yline(0)

l=[]
for num = 1:5
    plot(y(1:N-1),V(:,num))
    l = [l , 'n='+string(num-1)]
    hold on
end
title('First 5 Eigenstates')
xlabel('y')
legend(l)
ylabel('Ψ')
```

Result:



Clearly, the result is in accordance with the QHO, with energy levels being equally spaced.

Now, we shall add a quartic term to the potential and repeat the same process.

```

%% Part 2: Recreating the same for a potential with an additional anharmonic term
f2 = figure();
figure(f2)

k=1.2;

mL2V = m*L^2 * (0.5*k*(y-0.5).^2 - k*(y-0.5).^4)

dmain = dy^(-2) + mL2V(2:N);
doff = ones(1,N-2)*-0.5/(dy^2);

mL2H = diag(dmain)+ diag(doff,1) + diag(doff,-1);
H = mL2H / (m*L^2);

E = eig(H);
V;D = eig(H);
subplot(2,2,1)
plot(y,mL2V/(m*L^2))
hold on
ylines(E(1:40))
title('Energy Levels (compare to AHO)')
xlabel('y')
ylabel('E')

subplot(2,2,2)
bar((0:1:39),E(1:40))
title('Energy Levels (bar graph)')
xlabel('N')
ylabel('Energy')

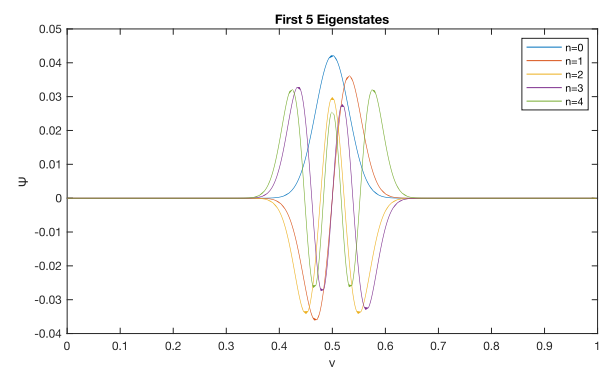
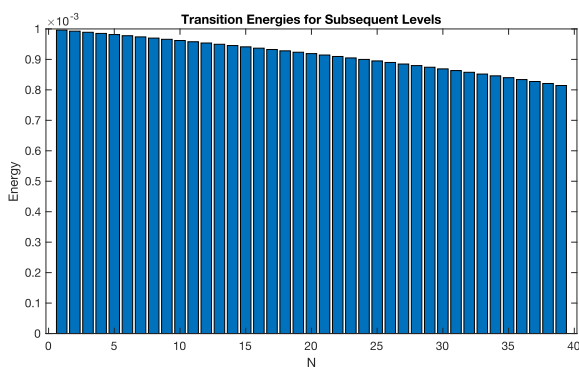
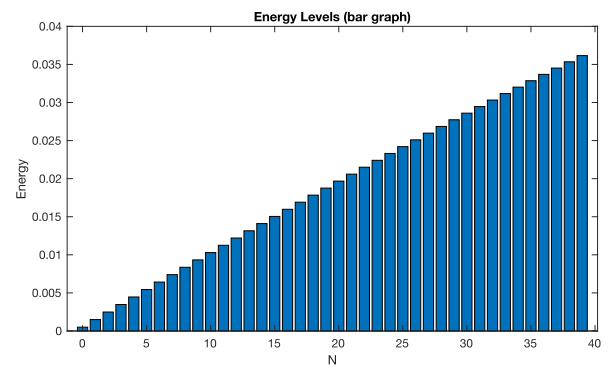
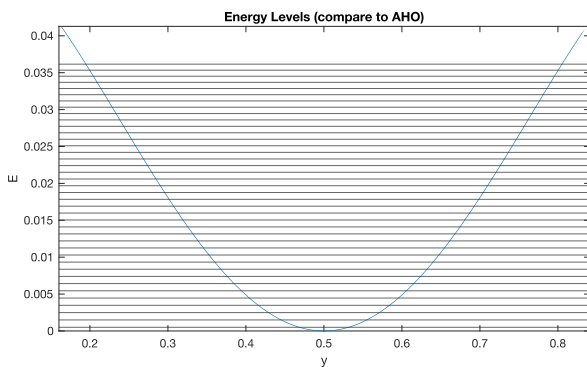
Lines = [];
for i=1:39
    Lines = [Lines , E(i+1)-E(i)];
end
subplot(2,2,3)
bar(Lines)
title('Transition Energies for Subsequent Levels')
xlabel('N')
ylabel('Energy')

subplot(2,2,4)
xline(0)
yline(0)

l=[]
for num = 1:5
    plot(y(1:N-1),V(:,num))
    l = [l , 'n='+string(num-1)]
    hold on
end
title('First 5 Eigenstates')
xlabel('y')
legend(l)
ylabel('Ψ')

```

Result:



As observable, the energy gaps between subsequent levels shows a decrease. Hence, the anharmonic oscillator potential can be used as a Qubit.

We have already learnt that the Josephson Junction Oscillator has energy levels similar to an

Anharmonic Oscillator. Hence, the Transmon Qubit is suitable for conducting Quantum Computation.

Bloch Sphere Representation of the Qubit

From this section onwards, we shall only focus on the two lower eigenstates of our quantum mechanical system, and we label them as $|0\rangle$ and $|1\rangle$. We shall shift our focus away from the physical picture and shift towards a more mathematical abstract picture.

As defined in the beginning of the report, the state of a qubit can be represented as a superposition of its two basis states.

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Where α and β are complex coefficients such that $|\alpha|^2 + |\beta|^2 = 1$ (Obvious result of normalization). $|\alpha|^2$ and $|\beta|^2$ denote the probabilities of obtaining the respective basis state on measurement.

The basis states $|0\rangle$ and $|1\rangle$ can be expressed in vector form as follows:

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

With α, β being complex coefficients (each with a magnitude and argument of its own), we would require 4 dimensions to represent the state of the qubit. However, it is important to realise that multiplying the state with any additional phase would not impact the relative phase between the basis states, and hence information about the global phase can be neglected, since it is only the local phase which impacts interaction with gates and other qubits.

So, if $\alpha = ae^{i\varphi_a}$ and $\beta = be^{i\varphi_b}$ (where a and $b \in \mathbb{R}^+$) then our state can be written as:

$$\begin{aligned} |\psi\rangle &= ae^{i\varphi_a} |0\rangle + be^{i\varphi_b} |1\rangle \\ &= e^{i\varphi_a} (a |0\rangle + be^{i(\varphi_b - \varphi_a)} |1\rangle) \end{aligned}$$

Like described above, the term $e^{i\varphi_a}$ only contains information about global phase and hence holds no significance. We rewrite $\varphi_b - \varphi_a = \varphi$ and refer to it as ‘local phase’.

$$|\psi\rangle = a |0\rangle + be^{i\varphi} |1\rangle$$

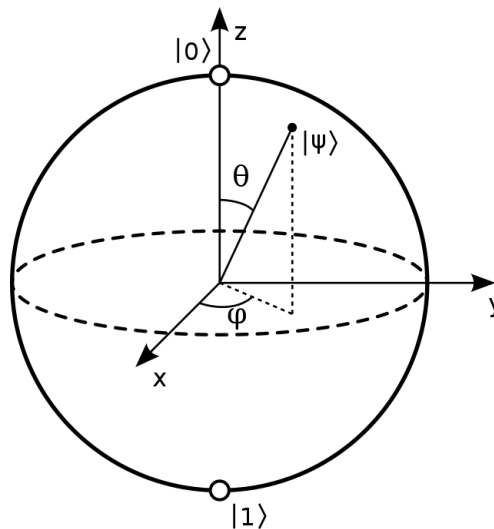
Now, we are aware that a and b are positive real numbers such that $a^2 + b^2 = 1$. An obvious thought that would arrive is substitution of $a = \cos(\theta)$ and $b = \sin(\theta)$. However, we would have to restrict ourselves to $\theta \in [0, \pi/2]$. Hence, if we try to represent θ and φ using polar coordinates, we face the following disadvantages:

1. We only get to use a hemispherical region to represent all states.
2. The state $|1\rangle$ is represented by infinite points on the circumference of the circular face of the hemisphere.

Hence, a more logical substitution would be $a = \cos(\theta/2)$ and $b = \sin(\theta/2)$, resulting in the final state representation as:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right) |0\rangle + e^{i\varphi} \sin\left(\frac{\theta}{2}\right) |1\rangle$$

We can hence represent the state of any two-level quantum system as a point/vector on a unit sphere:



(Photograph by Smite-Meister, distributed under a CC BY-SA 3.0 license[11])

As you can see, all the points on the surface of the Bloch Sphere refer to a unique single qubit state.

Density Matrix Representation of the State of a Single Qubit [1]

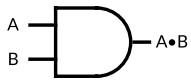
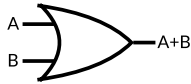
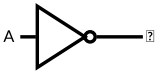
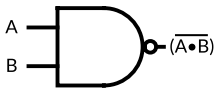
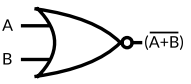
The state of a qubit can also be represented by its density matrix:


$$\rho = |\psi\rangle\langle\psi| = \begin{bmatrix} |\alpha|^2 & \alpha\beta^* \\ \alpha^*\beta & |\beta|^2 \end{bmatrix} = \begin{bmatrix} \cos^2\left(\frac{\theta}{2}\right) & e^{-i\varphi} \cos\left(\frac{\theta}{2}\right) \sin\left(\frac{\theta}{2}\right) \\ e^{i\varphi} \cos\left(\frac{\theta}{2}\right) \sin\left(\frac{\theta}{2}\right) & \sin^2\left(\frac{\theta}{2}\right) \end{bmatrix}$$

The significant advantage of using the density matrix is that we can use this to represent mixed and entangled states as well.

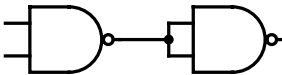
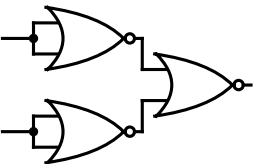
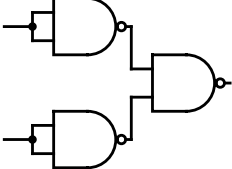
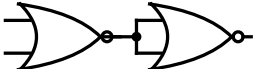
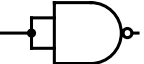

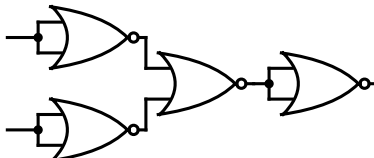
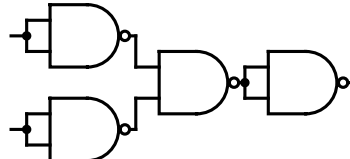
A Short Review on Classical Logic Gates

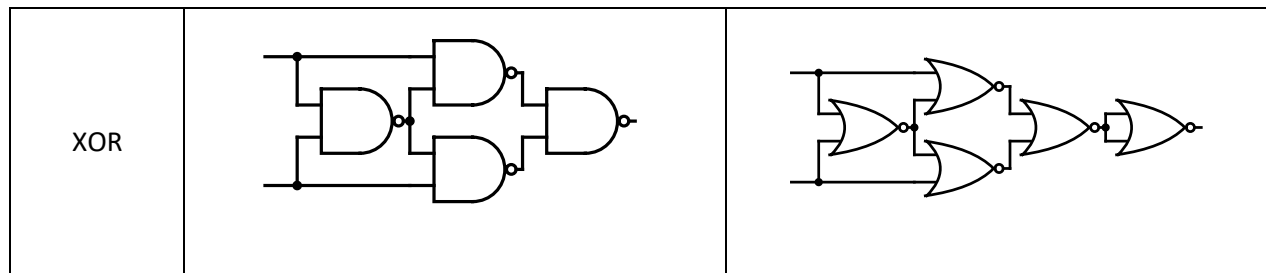
Before diving further into Quantum Logic Gates, we shall do a brief review of Classical Logic Gates. We shall not go deep into the physical picture and restrict ourselves to the Truth Table.

GATE	Diagram	Boolean Expression(s)	Truth Table															
AND		$A \bullet B$	<table><tr><th>A</th><th>B</th><th>$A \bullet B$</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	$A \bullet B$	0	0	0	0	1	0	1	0	0	1	1	1
A	B	$A \bullet B$																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		$A + B$	<table><tr><th>A</th><th>B</th><th>$A + B$</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	$A + B$	0	0	0	0	1	1	1	0	1	1	1	1
A	B	$A + B$																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
NOT		\bar{A}	<table><tr><th>A</th><th>\bar{A}</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	\bar{A}	0	1	1	0									
A	\bar{A}																	
0	1																	
1	0																	
NAND		$\overline{(A \bullet B)}$ $= \bar{A} + \bar{B}$	<table><tr><th>A</th><th>B</th><th>$\overline{(A \bullet B)}$</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	$\overline{(A \bullet B)}$	0	0	1	0	1	0	1	0	0	1	1	0
A	B	$\overline{(A \bullet B)}$																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
NOR		$\overline{(A + B)}$ $= \bar{A} \bullet \bar{B}$	<table><tr><th>A</th><th>B</th><th>$\overline{(A + B)}$</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	$\overline{(A + B)}$	0	0	0	0	1	0	1	0	0	1	1	1
A	B	$\overline{(A + B)}$																
0	0	0																
0	1	0																
1	0	0																
1	1	1																

XOR		$A \oplus B = A\bar{B} + \bar{A}B$	A	B	$A \oplus B$
			0	0	0
			0	1	1
			1	0	1
			1	1	0

The NAND (and NOR) Gate are known as Universal Gates, since any of these basic gates can be built using just NAND (or NOR) Gates. We shall see how:

GATE	In Terms of NAND	In Terms of NOR
AND		
OR		
NOT		
NAND	-	
NOR		-



Classical Gates are implemented using Transistors. They operate on Boolean Logic and are hence used to solve Boolean Functions. These gates are irreversible.

There also exist reversible gates such as the Toffoli and Fredkin Gates. We see that Toffoli Gate is an important gate for Quantum Computation as well, however it does encompass classical computing as well. Hence, we shall discuss about the Toffoli Gate when we are dealing with Multi-Qubit Quantum Gates.

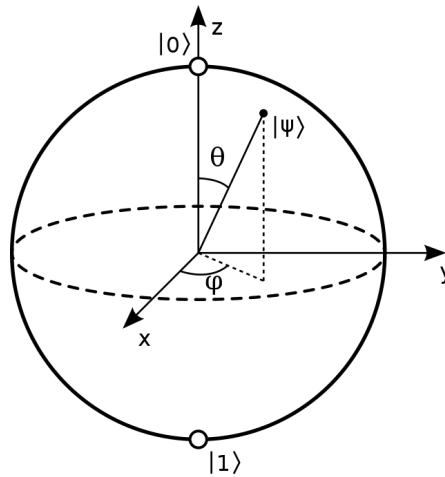
The major difference between Classical and Quantum Gates are their implementation. A qubit's construction is fundamentally different as compared to that of a classical bit, which is normally just stored in terms of voltages 'High' and 'Low'. We tend to use quantum mechanical systems in qubits where the state is determined by the energy level or the spin associated with the system. Hence, unlike classical gates, Quantum Gates are realised via pulses. We have already talked about the Bloch sphere representation of the qubit before taking a detour into classical gates. We shall see how a Quantum Gate operation is represented.

Introduction to Quantum Gates as Rotations on the Bloch Sphere

A quantum gate can be described as a unitary evolution of the state of the qubit. They are designed such that the resulting unitary evolution of the qubit implements the target gate. [1]

$$\hat{U}(t) = e^{-\frac{i\hat{H}t}{\hbar}}$$

A single qubit gate is normally represented as a rotation (or a series of rotations) on the Bloch Sphere to achieve a particular outcome. Gates can also be represented as matrices. To get us a good head start on Quantum Gates, we shall first talk about rotations about the primary axes.



(Photograph by Smite-Meister, distributed under a CC BY-SA 3.0 license[11])

There are namely 3 axes, i.e., the X, Y and Z axes. The coordinates of the state-vector in terms of cartesian coordinates can be written as:

$$(\sin(\theta) \cos(\varphi), \sin(\theta) \sin(\varphi), \cos(\theta))$$

The matrices associated with Rotation by angle Ω about X, Y and Z axes are shown below:

$$R_x(\Omega) = \begin{bmatrix} \cos\left(\frac{\Omega}{2}\right) & -i \sin\left(\frac{\Omega}{2}\right) \\ -i \sin\left(\frac{\Omega}{2}\right) & \cos\left(\frac{\Omega}{2}\right) \end{bmatrix}$$

$$R_y(\Omega) = \begin{bmatrix} \cos\left(\frac{\Omega}{2}\right) & -\sin\left(\frac{\Omega}{2}\right) \\ \sin\left(\frac{\Omega}{2}\right) & \cos\left(\frac{\Omega}{2}\right) \end{bmatrix}$$

$$R_z(\Omega) = \begin{bmatrix} e^{-\frac{i\Omega}{2}} & 0 \\ 0 & e^{\frac{i\Omega}{2}} \end{bmatrix} \left(\text{or} \begin{bmatrix} 1 & 0 \\ 0 & e^{i\Omega} \end{bmatrix} \right)$$

These are all unitary matrices. Note that these are not necessarily reversible. All the conventional gates used in most Quantum Computing can be expressed as a sequence of rotations.

We shall now begin our discussion of gates. We will start off with single qubit gates and then discuss CNOT gate and other multi-qubit gates. We shall review its implementation and representation.

Pauli Gates – X, Y, Z

The Pauli Gates represent 180° rotations about the respective axes. We shall study them one-by-one:

Pauli-X Gate

This gate is also called the ‘Bit-Flip’ Gate and is the Quantum Analogue of the classical NOT gate. The reason for this is because it changes state $|0\rangle$ to state $|1\rangle$ and vice-versa.

$$X = R_x(\pi) = \begin{bmatrix} \cos\left(\frac{\pi}{2}\right) & -i \sin\left(\frac{\pi}{2}\right) \\ -i \sin\left(\frac{\pi}{2}\right) & \cos\left(\frac{\pi}{2}\right) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

We shall now see the action of the X gate on states $|0\rangle$ and $|1\rangle$:

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

We can also express the Pauli-X Gate in Dirac Notation:

$$\hat{X} = |0\rangle\langle 1| + |1\rangle\langle 0|$$

And hence, in a general state:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

$$X|\psi\rangle = \alpha|1\rangle\langle 0|0\rangle + \beta|0\rangle\langle 1|1\rangle = \alpha|1\rangle + \beta|0\rangle$$

(Note: $\langle 0|1\rangle$ and $\langle 1|0\rangle = 0$ and hence omitted)

It is clearly visible that the X gate is reversible since $X^2=I$.

Pauli-Z Gate

This gate is frequently called the ‘Phase-Flip’ Gate. It does not bring about any change to the magnitude of the amplitudes of the basis states but brings about an additional phase change of π .

$$Z = R_z(\pi) \text{ OR } \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

We shall now see the action of the Z gate on states $|0\rangle$ and $|1\rangle$:

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

We can represent the Pauli-Z Gate in Dirac notation as:

$$\hat{Z} = |0\rangle\langle 0| - |1\rangle\langle 1|$$

And hence, for a general state:

$$Z(\alpha|0\rangle + \beta|1\rangle) = \alpha|0\rangle\langle 0|0\rangle - \beta|1\rangle\langle 1|1\rangle = \alpha|0\rangle - \beta|1\rangle$$

which clearly shows a phase flip.

The Z Gate is also a reversible gate, since $Z^2=I$

Pauli-Y Gate

The Pauli-Y Gate is known to do both a Bit-Flip as well as a Phase-Flip. It refers to a 180° rotation of the Bloch-vector about the Y-axis.

$$Y = R_Y(\pi) = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

We can judge the action of the Y gate on states $|0\rangle$ and $|1\rangle$ and hence on a general state

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

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

Hence,

$$Y(\alpha|0\rangle + \beta|1\rangle) = i\alpha|1\rangle - i\beta|0\rangle = i(\alpha|1\rangle - \beta|0\rangle)$$

(Note: the term i in the final step only stores information about the local phase)

The final term has a local phase of π as well as a bit-flip.

In Dirac notation, we can express the Y gate as:

$$\hat{Y} = i|1\rangle\langle 0| - i|0\rangle\langle 1|$$

This gate is also reversible, since $Y^2=I$.

Other Important Single-Qubit Gates

The Pauli Gates are fundamental to most Quantum Algorithms we have developed today.

However, these are not sufficient alone. We find that other gates such as the Hadamard Gate, the S gate and the T gate also hold a lot of importance.

The Hadamard Gate (H)

The Hadamard Gate is one of the most important gates when it comes to Quantum Computation. It is used to create superposition states.

We shall first see the matrix representation of the H gate:

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

We see the power of the Hadamard Gate when we apply it to basis states $|0\rangle$ and $|1\rangle$:

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

As you can see, both the states created are superposition states. The states $|+\rangle$ and $|-\rangle$ have very significant role in most quantum algorithms. Some examples are listed below:

- ⇒ If we apply the H gate on every qubit of an n-qubit quantum register, we generate a superposition of 2^n inputs. This allows us to achieve Quantum Parallelism, which gives a significant quantum advantage.
- ⇒ The $|-\rangle$ state is usually used as an ancilla qubit to induce phase kickback when using quantum oracles. We observe this in a lot of algorithms such as Deutsch-Jozsa Algorithm, Bernstein-Vazirani Algorithm and Grover's Algorithm.

The Hadamard Gate can also be represented as a series of rotations. It is usually expressed as a $\pi/2$ rotation about the y axis followed by a π rotation about the x axis:

$$H = R_x(\pi)R_y\left(\frac{\pi}{2}\right)$$

We can also say that the Hadamard Gate represents a π rotation about the line $x=z$

In Dirac notation, we can represent the Hadamard Gate as:

$$\hat{H} = \frac{|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|}{\sqrt{2}}$$

One should realise that $H^2 = I$ and hence the Hadamard Gate is reversible.

Phase Shift Gates: S and T

A class of quantum gates called phase shift gates also exists which are responsible for inducing a change in phase of the qubit. These do not affect the magnitude of the complex amplitudes.

A general notation of a Phase-Shift Gate is:

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

Two examples of phase shift gates are S and T gates. (It is important to know that the Z gate also falls in this category)

$$S = P\left(\frac{\pi}{2}\right) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{2}} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$$

$$T = P\left(\frac{\pi}{4}\right) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix}$$

The above gates are non-Hermitian and non-reversible.

Some Interesting Relations between Gates

Some identities would be of use to us in various algorithms and when dealing with multiple qubit circuits:

$$X = HZH$$

$$Z = HXH$$

$$H = (X+Z)/\sqrt{2}$$

$$ZX = -XZ = iY$$

Multiple Qubit Gates

We have already done a review of basic single qubit gates. We now go on to study multiple-qubit circuits. When we are using more than 1 qubit, the combined state of the qubits is represented using the tensor product of the states of the individual qubits.

$$|ab\rangle = |a\rangle \otimes |b\rangle$$

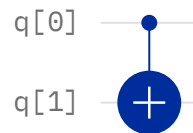
Hence, in the case of two qubits, the basis states are written as follows:

$$\begin{aligned} |00\rangle &= |0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} & |01\rangle &= |0\rangle \otimes |1\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \\ |10\rangle &= |1\rangle \otimes |0\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} & |11\rangle &= |1\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \end{aligned}$$

We normally represent multiple qubit gates using matrices as well.

Controlled-NOT Gate:

The Controlled-NOT Gate, also dubbed as the C-NOT Gate or CX Gate, is one of the most important two-qubit gates. It is responsible for generation and disentanglement of Bell pairs. Here is the diagrammatic representation of the C-NOT gate:



(Picture Source: IBM Quantum Composer [7])

The C-NOT Gate has two inputs: control qubit and target qubit. The gate applies a bit-flip on the target qubit whenever the control qubit is in state '1'.

The matrix representation of the C-NOT Gate is as follows:

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & X \end{bmatrix}$$

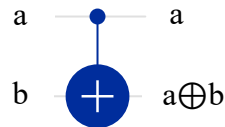
We shall see the result of application of the C-NOT Gate for all the possible basis state inputs now:

Control Input	Target Input	Control Output	Target Output
$ 0\rangle$	$ 0\rangle$	$ 0\rangle$	$ 0\rangle$
$ 0\rangle$	$ 1\rangle$	$ 0\rangle$	$ 1\rangle$
$ 1\rangle$	$ 0\rangle$	$ 1\rangle$	$ 1\rangle$
$ 1\rangle$	$ 1\rangle$	$ 1\rangle$	$ 0\rangle$

In other words:

$$\begin{aligned} CNOT|00\rangle &= |00\rangle, CNOT|01\rangle = |01\rangle, \\ CNOT|10\rangle &= |11\rangle, CNOT|11\rangle = |10\rangle \end{aligned}$$

While the control qubit is interchanged, the target qubit gets flipped. We can compare the output of the target qubit to that of the classical XOR gate. Hence, the C-NOT gate is considered to be the classical analogue of the XOR gate:



(Picture Source: IBM Quantum Composer [7])

The CNOT Gate is very central to Quantum Computation. Some examples are listed below:

1. The CNOT Gate alongside H gate can be used to prepare and dis-entangle entangled states. This makes it central in certain quantum communication algorithms such as Quantum Teleportation, which deals with entanglement swapping.
2. The CNOT Gate is used for oracle design and is frequently used alongside the $|-\rangle$ state as an ancilla qubit to provide phase kickback. This is because of the fact that $X|-\rangle = -|-\rangle$.

Note: As observed by the matrix representation, we can generalize for a controlled-U gate.

If we have a Unitary Gate $U = \begin{bmatrix} U_1 & U_2 \\ U_3 & U_4 \end{bmatrix}$ then we can represent a controlled-U gate using the matrix:

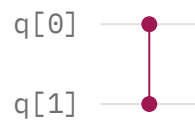
$$CU = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & U_1 & U_2 \\ 0 & 0 & U_3 & U_4 \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & U \end{bmatrix}$$

A wide variety of gates belong to this category, such as the CX, CZ, CS and CT gates. We shall discuss about the C-Z gate now.

Controlled-Z Gate:

The controlled-Z gate, also known as the CZ gate, is another important two-qubit gate. Like the CNOT Gate, the CZ gate also takes in two inputs i.e., the control and target qubits, but instead applies a phase flip to the target qubit when the control qubit is in state '1'.

Below is a diagrammatic representation of the CZ gate:



(Picture Source: IBM Quantum Composer [7])

The CZ Gate Matrix can be expressed as shown below:

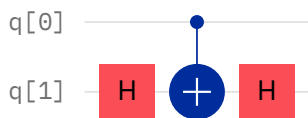
$$CZ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & Z \end{bmatrix}$$

We shall see the result of application of the C-Z Gate for all the possible basis state inputs now:

$$\begin{aligned} CZ|00\rangle &= |00\rangle, CZ|01\rangle = |01\rangle, \\ CZ|10\rangle &= |10\rangle, CZ|11\rangle = -|11\rangle \end{aligned}$$

The CZ gate can also be constructed using CNOT Gate. We made ourselves familiar with the identity that $Z = HXH$

Hence, we can construct a CZ gate using by applying the H gate on both sides of the CZ gate on the target qubit.

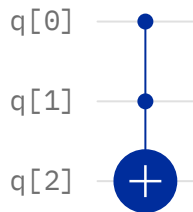


(Picture Source: IBM Quantum Composer[7])

If the control qubit is '0' then the identity $H^2 = I$ tells us that the state of the target qubit remains unchanged, and if the control qubit is '1' then the phase flip is applied.

Toffoli Gate

The Toffoli Gate is an extended version of the CNOT gate which has 2 control qubit inputs instead of one. It is hence also called the CCNOT or CCX gate. Its action is similar to the CNOT gate, except it only flips the target if both the control qubits are in '1' state. The circuit representation and matrix representation are shown below:



(Picture Source: IBM Quantum Composer[7])

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

The Toffoli Gate finds a lot of applications in various algorithms such as Grover's Algorithm.

It is important to note that the term Toffoli Gate might also refer to the Generalised Toffoli Gate which has multiple control qubit inputs + 1 target qubit input. We generally refer to these as C3X, C4X, C5X... or C^nX .

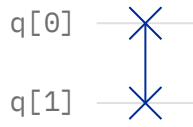
SWAP Gate

The SWAP Gate is a 2-Qubit Gate and is used to swap the states of two qubits.

Its application is as follows:

$$\text{SWAP}|q_0q_1\rangle = |q_1q_0\rangle$$

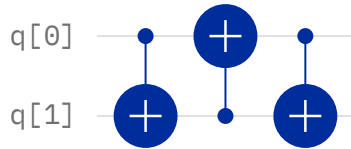
The matrix representation and diagrammatic representation are shown below:



(Picture Source: IBM Quantum Composer[7])

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

The SWAP Gate can also be implemented with the help of CNOT Gates as follows:



(Picture Source: IBM Quantum Composer[7])

The calculation is shown below:

On application of the first CX Gate:

$$CX|q_0q_1\rangle = |q_0\rangle|q_0\oplus q_1\rangle$$

Then on applying the second CX gate (note that the control and target qubit are switched so I switched them here.):

$$CX(|q_0\oplus q_1\rangle|q_0\rangle) = |q_0\oplus q_1\rangle|q_0\oplus q_0\oplus q_1\rangle = |q_0\oplus q_1\rangle|0\oplus q_1\rangle = |q_0\oplus q_1\rangle|q_1\rangle$$








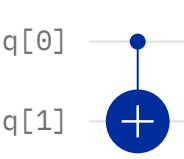
We then apply the 3rd CX gate (re-switching the control and target qubits)

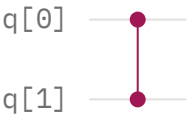
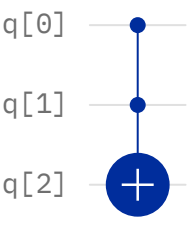
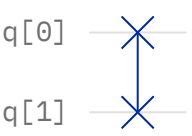
$$CX(|q_1\rangle|q_0\oplus q_1\rangle) = |q_1\rangle|q_0\oplus q_1\oplus q_1\rangle = |q_1\rangle|q_0\oplus 0\rangle = |q_1\rangle|q_0\rangle = |q_1q_0\rangle$$

Hence, the swap has been achieved.

Summary of Quantum Gates

We have now discussed most of the important Multi-Qubit Gates. Here, I have summarised all the gates. Note that all the symbols here have been drawn with the help of IBM's Quantum Composer[7], which is an open-source tool for working with Quantum Circuits:

Gate Name	Number of Inputs	Symbol	Function	Matrix
Pauli X	1	 or 	$R_x(\pi)$	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli Y	1		$R_y(\pi)$	$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli Z	1		$R_z(\pi)$	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard	1		$R_x(\pi)R_y\left(\frac{\pi}{2}\right)$	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase Gate (S)	1		$R_z\left(\frac{\pi}{2}\right)$	$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$ Gate (T)	1		$R_z\left(\frac{\pi}{4}\right)$	$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix}$
CNOT Gate	2		$\hat{X} \psi_t\rangle \text{ if } \psi_c\rangle = 1$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

CZ Gate	2		$\hat{Z} \psi_t\rangle \text{ if } \psi_c\rangle = 1\rangle$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
Toffoli Gate	3(+)		$\hat{X} \psi_t\rangle \text{ if } \psi_{c_1}\psi_{c_2}\rangle = 11\rangle$	$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$
SWAP Gate	2		$\text{SWAP} q_0q_1\rangle = q_1q_0\rangle$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

With this we complete our review of the basic gates we use. We shall now move on to simulations via MATLAB.

Simulation of Basic Quantum Gates via MATLAB

Here, I have attempted to simulate all the above gates on MATLAB.

Simulation of Single and Multi-Qubit Gates

```
%In this section, I attempt to simulate the
action of gates. I shall be
%using matrix representation of the gates.
```

```
ket0 = [1;0];
ket1 = [0;1];
```

```
syms a
syms b
psi = a*ket0 + b*ket1
%I have defined a and b as symbolic variables,
making advantage of the
%Symbolic Math Toolbox of MATLAB. The reason
for the same is to ensure that
%we can truly understand gate action on
arbitrary state.
```

```
format shortG
```

```
psi =
```

```
a
```

```
b
```

Single Qubit Gate Action

```
%First, we shall be simulating gate action on
single qubits.
```

1. Pauli-X

```
X = [0 1;
```

```
1 0];
```

```
disp('Action of X gate on state |0>:')
disp(X*ket0)
```

```
disp('Action of X gate on state |1>:')
disp(X*ket1)
```

```
disp('Action of X gate on state a|0> + b|1>:')
disp(X*psi)
```

Action of X gate on state $|0\rangle$:

```
0
```

```
1
```

Action of X gate on state $|1\rangle$:

```
1
```

```
0
```

Action of X gate on state $a|0\rangle + b|1\rangle$:

```
b
```

```
a
```

2. Pauli-Y

```
Y = [0 -1i;
```

```
1i 0];
```

```
disp('Action of Y gate on state |0>:')
disp(Y*ket0)
```

```
disp('Action of Y gate on state |1>:')
disp(Y*ket1)
```

```
disp('Action of Y gate on state a|0> + b|1>:')
disp(Y*psi)
```

Action of Y gate on state $|0\rangle$:

```
0 + 0i
```

```
0 + 1i
```

Action of Y gate on state $|1\rangle$:

```
0 - 1i
```

```
0 + 0i
```

Action of Y gate on state $a|0\rangle + b|1\rangle$:

```
-b*1i
```

```
a*1i
```

3. Pauli-Z

```
Z = [1 0;
```

```
0 -1];
```

```
disp('Action of Z gate on state |0>:')
disp(Z*ket0)
```

```
disp('Action of Z gate on state |1>:')
disp(Z*ket1)
```

```
disp('Action of Z gate on state a|0> + b|1>:')
disp(Z*psi)
```

Action of Z gate on state $|0\rangle$:

1

0

Action of Z gate on state $|1\rangle$:

0

-1

Action of Z gate on state $a|0\rangle + b|1\rangle$:

a

-b

4. Hadamard Gate

```
H = (1/sqrt(2))*[1 1;
```

```
1 -1];
```

```
disp('Action of H gate on state |0>:')
disp(H*ket0)
```

```
disp('Action of H gate on state |1>:')
disp(H*ket1)
```

```
disp('Action of H gate on state a|0> + b|1>:')
disp(H*psi)
```

Action of H gate on state $|0\rangle$:

0.70711

0.70711

Action of H gate on state $|1\rangle$:

0.70711

-0.70711

Action of H gate on state $a|0\rangle + b|1\rangle$:

$(2^{1/2}*a)/2 + (2^{1/2}*b)/2$

$(2^{1/2}*a)/2 - (2^{1/2}*b)/2$

5. S Gate

```
S = [1 0;
```

```
0 j];
```

```
disp('Action of S gate on state |0>:')
disp(S*ket0)
```

```
disp('Action of S gate on state |1>:')
disp(S*ket1)
```

```
disp('Action of S gate on state a|0> + b|1>:')
disp(S*psi)
```

Action of S gate on state $|0\rangle$:

1

0

Action of S gate on state $|1\rangle$:

0 + 0i

0 + 1i

Action of S gate on state $a|0\rangle + b|1\rangle$:

a

b*1i

6. T Gate

```
T = [1 0;
```

```
0 exp(i*pi/4)];
```

```
disp('Action of T gate on state |0>:')
disp(T*ket0)
```

```
disp('Action of T gate on state |1>:')
disp(T*ket1)
```

```
disp('Action of T gate on state a|0> + b|1>:')
disp(T*psi)
```

Action of T gate on state $|0\rangle$:

1

0

Action of T gate on state $|1\rangle$:

$$\begin{pmatrix} 0 & + & 0i \\ 0.70711 & + & 0.70711i \end{pmatrix}$$

Action of T gate on state $a|0\rangle + b|1\rangle$:

$$2^{(1/2)} * b * (1/2 + 1i/2)$$

Verification of basic gate identities

HZH = H*Z*H

HXH = H*X*H

```
disp('(X+Z)/√2=')
disp((X+Z)/sqrt(2))
XZ = X*Z
ZX = Z*X
iY = i*Y
```

HZH =

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

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

HXH =

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

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

(X+Z)/√2=

$$\begin{pmatrix} 0.70711 & 0.70711 \\ 0.70711 & -0.70711 \end{pmatrix}$$

$$\begin{pmatrix} 0.70711 & -0.70711 \\ 0.70711 & 0.70711 \end{pmatrix}$$

XZ =

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

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

ZX =

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

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

iY =

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

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

Action of 2-Qubit Gates

%For 2-qubit gates, we have four basis states which stem from tensor % product of the basis states of a single qubit

```
syms c
syms d

k00 = kron(ket0,ket0);
k01 = kron(ket0,ket1);
k10 = kron(ket1,ket0);
k11 = kron(ket1,ket1);

psi_ = a*k00+b*k01+c*k10+d*k11
```

psi_ =

a

b

c

d

CNOT-Gate

CNOT = [1 0 0 0;

0 1 0 0;

0 0 0 1

0 0 1 0];

```
disp('Action of CNOT gate on state |00>:')
disp(CNOT*k00)
```

```
disp('Action of CNOT gate on state |01>:')
disp(CNOT*k01)
```

```
disp('Action of CNOT gate on state |10>:')
disp(CNOT*k10)
```

```
disp('Action of CNOT gate on state |11>:')
disp(CNOT*k11)
```

```
disp('Action of CNOT gate on state a|00> + b|01> + c|10> + d|11>:')
disp(CNOT*psi_)
```

Action of CNOT gate on state $|00\rangle$:

1

0

0

0

Action of CNOT gate on state $|01\rangle$:

0
1
0
0

Action of CNOT gate on state $|10\rangle$:

0
0
0
1

Action of CNOT gate on state $|11\rangle$:

0
0
1
0

Action of CNOT gate on state $a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle$:

a
b
d
c

CZ Gate

```
CZ = [1 0 0 0;
      0 1 0 0;
      0 0 1 0;
      0 0 0 -1];
```

```
disp('Action of CZ gate on state |00>:')
disp(CZ*k00)
```

```
disp('Action of CZ gate on state |01>:')
disp(CZ*k01)
```

```
disp('Action of CZ gate on state |10>:')
disp(CZ*k10)

disp('Action of CZ gate on state |11>:')
disp(CZ*k11)

disp('Action of CZ gate on state a|00> + b|01> + c|10> + d|11>:')
disp(CZ*psi_)
```

Action of CZ gate on state $|00\rangle$:

1
0
0
0

Action of CZ gate on state $|01\rangle$:

0
1
0
0

Action of CZ gate on state $|10\rangle$:

0
0
1
0

Action of CZ gate on state $|11\rangle$:

0
0
0
-1

Action of CZ gate on state $a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle$:

a
b
c

-d

SWAP GATE

```
SWAP = [1 0 0 0;
```

```
0 0 1 0;
```

```
0 1 0 0
```

```
0 0 0 1];
```

```
disp('Action of SWAP gate on state |00>:')
disp(SWAP*k00)
```

```
disp('Action of SWAP gate on state |01>:')
disp(SWAP*k01)
```

```
disp('Action of SWAP gate on state |10>:')
disp(SWAP*k10)
```

```
disp('Action of SWAP gate on state |11>:')
disp(SWAP*k11)
```

```
disp('Action of SWAP gate on state a|00> +
b|01> + c|10> + d|11>:')
disp(SWAP*psi_)
```

Action of SWAP gate on state $|00\rangle$:

```
1
```

```
0
```

```
0
```

```
0
```

Action of SWAP gate on state $|01\rangle$:

```
0
```

```
0
```

```
1
```

```
0
```

Action of SWAP gate on state $|10\rangle$:

```
0
```

```
1
```

```
0
```

```
0
```

Action of SWAP gate on state $|11\rangle$:

```
0
```

```
0
```

```
0
```

```
1
```

Action of SWAP gate on state $a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle$:

```
a
```

```
c
```

```
b
```

```
d
```

3 Qubit Gate: Toffoli Gate

```
syms e
syms f
syms g
syms h
%first, we shall construct our 3-qubit basis
states
```

```
k000 = kron(ket0,k00);
```

```
k001 = kron(ket0,k01);
```

```
k010 = kron(ket0,k10);
```

```
k011 = kron(ket0,k11);
```

```
k100 = kron(ket1,k00);
```

```
k101 = kron(ket1,k01);
```

```
k110 = kron(ket1,k10);
```

```
k111 = kron(ket1,k11);
```

```
psi__ = a*k000 + b*k001 + c*k010 + d*k011 +
e*k100 + f*k101 + g*k110 + h*k111
```

```
TOFFOLI = [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];
```

```
% We shall not be testing out the Toffoli for
individual cases, but we
% shall instead be checking it for our general
state only and make our
% inferences from there. It shall be verified
if g and h would swap in
% the final statevector.
```

```
disp('Action of TOFFOLI gate on general state
a|000> + b|001> + c|010> + d|011>')
disp(' + e|100> + f|101> + g|110> + h|111>')
disp(TOFFOLI*psi__)
```

```
psi__ =
```

```
a
```

b
c
d
e
f
g
h

Action of TOFFOLI gate on general state $a|000\rangle$
 $+ b|001\rangle + c|010\rangle + d|011\rangle$

$+ e|100\rangle + f|101\rangle + g|110\rangle + h|111\rangle$

a
b
c
d
e
f
h
g

Published with MATLAB® R2023b

Verification of Gate Construction via Rotation, Phase and CNOT Gates

We have seen that gates can be described using Rotations as well. Hence, it is always possible to express gates in terms of Rotation and Phase Change Gates. The same has been done below via MATLAB.

Redefining Basis States and Basic Gates

```
format short

ket0 = [1;0];
ket1 = [0;1];
k00 = kron(ket0,ket0);
k01 = kron(ket0,ket1);
k10 = kron(ket1,ket0);
k11 = kron(ket1,ket1);
k000 = kron(ket0,k00);
k001 = kron(ket0,k01);
k010 = kron(ket0,k10);
k011 = kron(ket0,k11);
k100 = kron(ket1,k00);
k101 = kron(ket1,k01);
k110 = kron(ket1,k10);
k111 = kron(ket1,k11);

syms a
syms b
syms c
syms d
syms e
syms f
syms g
syms h
psi = a*ket0 + b*ket1;
psi_ = a*k00+b*k01+c*k10+d*k11;
psi__ = a*k000 + b*k001 + c*k010 + d*k011 +
e*k100 + f*k101 + g*k110 + h*k111;

I = diag(ones(1,2));
X = [0 1;
     1 0];
Y = [0 -1i;
     1i 0];
Z = [1 0;
     0 -1];
```

```
H = (1/sqrt(2))*[1 1;
                 1 -1];
S = [1 0;
     0 1i];
T = [1 0;
     0 exp(1i*pi/4)];
CNOT = [1 0 0 0;
        0 1 0 0;
        0 0 0 1;
        0 0 1 0];
CZ = [1 0 0 0;
      0 1 0 0;
      0 0 1 0;
      0 0 0 -1];
SWAP = [1 0 0 0;
        0 0 1 0;
        0 1 0 0;
        0 0 0 1];
TOFFOLI = [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];
```

Defining some additional gates: Rotation and Phase Gates

```
Rx = @(th) [cos(th/2) , -1i*sin(th/2);
            -1i*sin(th/2), cos(th/2)];

Ry = @(th) [cos(th/2) , -sin(th/2);
            sin(th/2), cos(th/2)];
```

```
Rz = @(th) [exp(-1i*th/2) , 0;
           0, exp(+1i*th/2)];

P = @(phi) [1, 0;
           0, exp(i*phi)];
```

We shall obtain our basic gates using Rx(θ), Ry(θ), Rz(θ), P(ϕ), CNOT

For this family we will be attempting to obtain the gates through rotations and shall be verifying their matrices. For our purposes, we shall ignore any additional Global Phase which might get added on since it does not have significance for our computations.

```
% Pauli-X
disp('X = Rx(pi) = ')
disp(Rx(pi))
disp('Here, an additional global phase of -1i is obtained')

% Pauli-Y
disp('Y = Ry(pi) = ')
disp(Ry(pi))
disp('Here, an additional global phase of -1i is obtained')

% Pauli-Z
disp('Z = P(pi) = ')
disp(P(pi))

%Hadamard
disp('H = Rx(pi)Ry(pi/2) = ')
disp(Rx(pi)*Ry(pi/2))
disp('Here, an additional global phase of -1i is obtained')

%S and T Gates:
disp('S = Ph(pi/2) = ')
disp(Ph(pi/2))
disp('T = Ph(pi/4) = ')
disp(Ph(pi/4))

% Building CZ using H(=Rx(pi)Ry(pi/2)) and CNOT
H_ = Rx(pi)*Ry(pi/2);
disp('CZ = I⊗H * CNOT * I⊗H')
CZ_ = kron(I,H_)*CNOT*kron(I,H_)

% Building SWAP using 2 CNOT Gates
%NOTE: The CNOT Gate Matrix for when the ctrl
and the target qubit are
%interchanged would be different. The
construction of the inverted CNOT
%matrix would be:
CNOT_Flip = [1 0 0 0;
            0 0 0 1;
            0 0 1 0;
            0 1 0 0];

disp('SWAP = CNOT*CNOT_Flip*CNOT = ')
disp(CNOT*CNOT_Flip*CNOT)

X = Rx(pi) =
```

```
0.0000 + 0.0000i    0.0000 - 1.0000i
```

```
0.0000 - 1.0000i    0.0000 + 0.0000i
```

Here, an additional global phase of -1i is obtained

Y = Ry(π) =

```
0.0000    -1.0000
1.0000     0.0000
```

Here, an additional global phase of -1i is obtained

Z = P(π) =

```
1.0000 + 0.0000i    0.0000 + 0.0000i
0.0000 + 0.0000i   -1.0000 + 0.0000i
```

H = Rx(π)Ry($\pi/2$) =

```
0.0000 - 0.7071i   -0.0000 - 0.7071i
0.0000 - 0.7071i    0.0000 + 0.7071i
```

Here, an additional global phase of -1i is obtained

S = Ph($\pi/2$) =

```
1.0000 + 0.0000i    0.0000 + 0.0000i
0.0000 + 0.0000i    0.0000 + 1.0000i
```

T = Ph($\pi/4$) =

```
1.0000 + 0.0000i    0.0000 + 0.0000i
0.0000 + 0.0000i    0.7071 + 0.7071i
```

CZ = I \otimes H * CNOT * I \otimes H

CZ_ =

```
-1.0000 - 0.0000i   -0.0000 - 0.0000i
0.0000 + 0.0000i    0.0000 + 0.0000i
```

```
-0.0000 - 0.0000i   -1.0000 + 0.0000i
0.0000 + 0.0000i    0.0000 + 0.0000i
```

```
0.0000 + 0.0000i    0.0000 + 0.0000i   -
1.0000 - 0.0000i   -0.0000 + 0.0000i
```

```
0.0000 + 0.0000i    0.0000 + 0.0000i   -
0.0000 - 0.0000i    1.0000 - 0.0000i
```

SWAP = CNOT*CNOT_Flip*CNOT =

```
1    0    0    0
```

0	0	1	0
0	1	0	0
0	0	0	1

Published with MATLAB® R2023b

With this, we have completed our review of gates. We have successfully shown the action of various gates on arbitrary states. We have also verified various constructions via rotations etc.

Now we shall look at some sets of universal gates. There are a lot of cases where just a few of these gates can create a set of gates which are useful for most circuits. We shall look at some examples now.

Universal Sets of Gates

Any set of gates S is referred to as a “universal set” if any feasible communication can be achieved in the circuit using solely the gates from the set S .

In classical computing, we have learnt that the NAND Gate (and similarly NOR Gate) itself is a universal set since it can be used to effectively design the other logic gates. However, in classical electronics, the number of gates can be limited, while the possible number of gates in Quantum Computing is uncountable. Hence, the ‘universal sets’ known to us either comprise of parameterised gates or can be fair approximations.

We shall look at some well-known families of universal gates used in Quantum Computation. Note that these might not be the most exhaustive gates.

Family 1: $R_x(\theta)$, $R_y(\theta)$, $R_z(\theta)$, $P(\phi)$, $CNOT$ [6]

Here, $R_x(\theta)$, $R_y(\theta)$ and $R_z(\theta)$ are parametrised gates which refer to the rotation operators about the X, Y and Z axes, while $P(\phi)$ refers to the parameterised phase shift gate which changes phase by a factor of ϕ . As you have seen, most single qubit gates can be expressed as combinations of R_x , R_y and R_z gates as they are basically just rotations. Hence, this set already encompasses a wide variety of gates, including all the single-qubit gates mentioned above. We can attempt to express any unitary as a set of rotations, and hence achieve universality to some degree.

The addition of the C-NOT gate in this set allows for interaction in two-qubit systems. We already know that the C-Z gate can be built using a C-NOT and H gate using the identity $Z = HXH$. Here, we can decompose the H gate in terms of the rotation gates as $R_x(\pi)R_y\left(\frac{\pi}{2}\right)$, and hence the CZ gate can be implemented using this. Similarly, we can attempt to create controlled-unitary type gates as well. We have also seen gates like SWAP being implemented using 3 C-NOT gates.

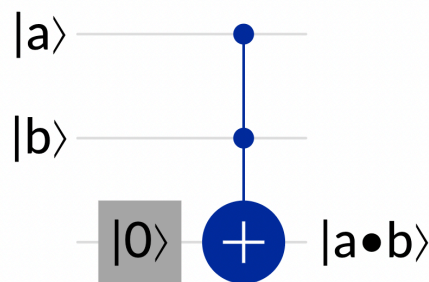
A possible question which arose my mind was how 3-qubit gates could be implemented using these. The famous CCX Gate (Toffoli Gate) can also be constructed using single and 2-qubit

gates (both of which can be constructed using this set of gates), but I have discussed this in more detail when dealing with Family #3, which utilises the H,S,T and CNOT Gates.

Family 2: Toffoli (CCX), H

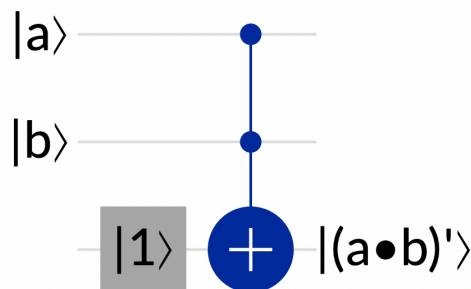
The Toffoli Gate, also dubbed as the CCX gate is of great importance since by itself it can act as a universal set which encompasses all classical computations. If we wish to implement classical analogues of logic gates, then the Toffoli Gate by itself is self sufficient. We shall see the following applications.

(Note that these demand usage of auxiliary qubits as well which are in either $|0\rangle$ or $|1\rangle$.)



Implementation of AND Gate via Toffoli Gate

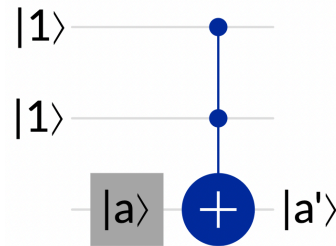
(Picture Source: IBM Quantum Composer [7])



Implementation of NAND Gate via Toffoli Gate

(Picture Source: IBM Quantum Composer [7])

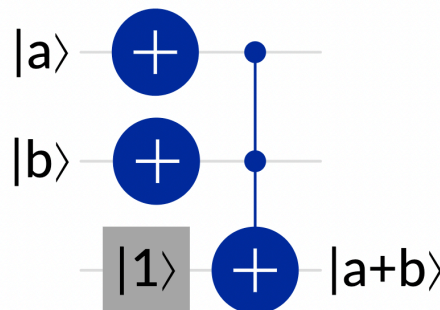
(since the NAND Gate can be represented using Toffoli Gates, we have effectively proven that a Toffoli Gate can represent all known classical gates. However, we shall still see some more examples.)



Implementation of the NOT/X Gate via Toffoli Gate

(Picture Source: IBM Quantum Composer [7])

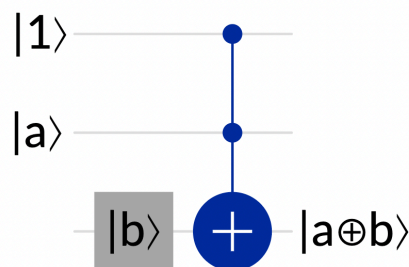
(Note: Since we have successfully represented the X Gate using Toffoli Gates, we shall use the X gate as well to implement other logic gates, but it is to be assumed that the X gate has been built via the above construction)



Implementation of the OR Gate via Toffoli Gate + X Gate

(Picture Source: IBM Quantum Composer [7])

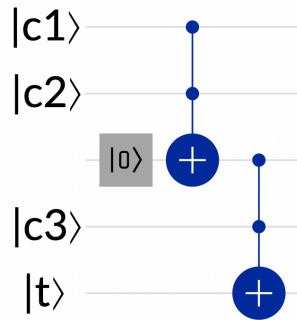
From the above examples, it can be clearly understood that basic logic gates can be implemented using just Toffoli Gates. We can also build the CNOT/XOR Gate as well, where all we do is let one of the control qubits be in state $|1\rangle$.



Implementation of the CNOT/XOR Gate via Toffoli Gate

(Picture Source: IBM Quantum Composer [7])

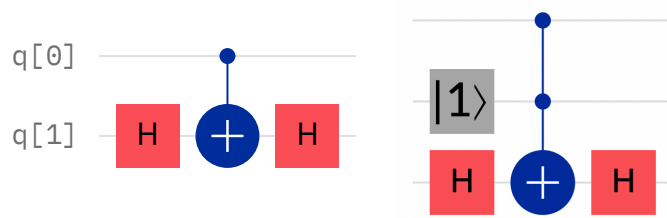
The above circuits clearly imply the universality of the Toffoli Gate when it comes to Boolean operations on qubits. We can also implement a generalised n-bit Toffoli using the help of auxiliary qubits in $|0\rangle$ state. The following example shows implementation of a C3X gate using the help of 2 CCX gates and 1 auxiliary qubit in state $|0\rangle$.



(Picture Source: IBM Quantum Composer [7])

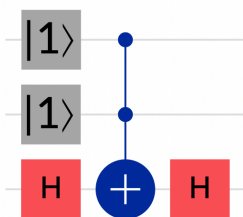
Here, the $|c_1\rangle$, $|c_2\rangle$, $|c_3\rangle$ qubits are the control qubits and the $|t\rangle$ represents the target qubit.

Once we have seen that the CNOT Gate and the basic logic gates have been successfully implemented, the addition of the H gate to this set would allow us to play with the phase of qubits as well. We can implement the Z gate using the identity $Z=HXH$ and also use the H gate to generate superpositions as and when needed. Using these, we can also construct the CZ gate and other important gates.



(Picture Source: IBM Quantum Composer [7])

The above examples are the construction of the CZ from the CNOT gate and the Toffoli Gates respectively. Setting the other control-qubit as 1 would allow implementation of the Z gate, as shown below.



(Picture Source: IBM Quantum Composer [7])

Hence, this set of gates can be used for a much wider variety of purposes. This however, does not encompass gates such as the S and T gates which are non-reversible fixed-angle gates, which is the main limitation of this set of gates.

Family 3: CNOT, H, S and T [6]

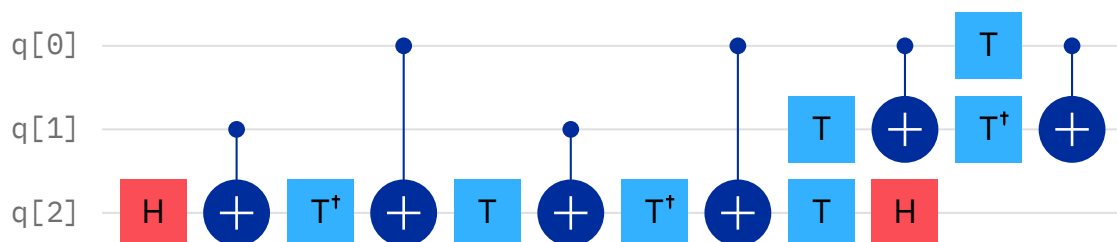
The set {CNOT, H, S} is referred to as the Clifford Set. This set in itself is not a universal set but the addition of T to the set would allow us to represent any unitary to an approximate level. It is an implication of the Solovay–Kitaev theorem [6][12] that any arbitrary single-qubit gate can be well approximated by the discrete gate set {H, S, T} and an accuracy of ϵ can be achieved by using $O(\log^c(1/\epsilon))$ gates. We shall not be going into the proof of the same for now but rather go through certain examples.

The S gate represents a rotation on the Bloch sphere about the Z axis by an angle of $\pi/2$, and hence it is obvious that $S^2 = Z$. Hence, the Z gate can be implemented very conveniently.

Using the identity $X = HZH$, we can also derive the X gate from the obtained Z gate, and hence obtain all the possible gates which we have summarised above.

This universal set also allows one to form multi-qubit gates. As an example, we shall use these gates to obtain the Toffoli Gate.

Note that T^\dagger refers to the inverse of T (and can also be written as T^7 or S^3T). We observe this distinction since the T gate is not reversible.



(Picture Source: IBM Quantum Composer [7])

While this representation looks very complicated and is not the most intuitive, this proves that we can produce Toffoli gates using these set of gates and hence this set can be used to obtain any quantum gates.

The DEUTSCH Gate [6][9][8]

In classical computation, the NAND and NOR gates are by themselves universal sets and they require 3-bits (2-input + 1 output) to be universal. Hence, we expect that if there were a singleton universal set of gates, it would comprise of a 3-qubit gate. It has been found that the DEUTSCH Gate is indeed universal for quantum computing. The DEUTSCH Gate has the following matrix representation:

$$D(\theta) = \begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \end{bmatrix}$$

Here, θ is a parameter.

The Deutsch Gate looks like a controlled-unitary type 3-qubit gate with 2 control qubits and 1 target qubit. Hence, it is like a CC-U Gate.

We can describe the unitary that would act on the target qubit as:

$$U(\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

Let us review first how $U(\theta)$ would act on states $|0\rangle$ and $|1\rangle$:

$$\begin{aligned} U(\theta)|0\rangle &= \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} = \cos(\theta) |0\rangle + \sin(\theta) |1\rangle \\ U(\theta)|1\rangle &= \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \sin(\theta) \\ \cos(\theta) \end{pmatrix} = \sin(\theta) |0\rangle + \cos(\theta) |1\rangle \end{aligned}$$

Hence the unitary $U(\theta)$ on application on a general state returns:

$$U(\theta)(\alpha|0\rangle + \beta|1\rangle) = (\alpha \cos(\theta) + \beta \sin(\theta))|0\rangle + (\beta \cos(\theta) + \alpha \sin(\theta))|1\rangle$$

We can clearly understand that certain substitutions of θ would returns some gates which are known to us and would hence help us achieve universality to a greater degree. For example, $\theta = \pi/2$ would give us $U = X$ and hence the DEUTSCH Gate would act like a Toffoli Gate, which is a very diverse gate in itself.

We can also observe the following property:

$$D(\alpha)D(\alpha') = iD(\alpha + \alpha')$$

This property allows us to further approximate any unitary just using a single gate. If we let θ/π be irrational, then we would be able to create a single universal gate $D(\theta)$ and approximate all known gates. [6]

The BARENCO Gate [6][10]

On further investigation, it turns out that the DEUTSCH Gate can be built from another set of 2-Qubit Gates. This is called the BARENCO Gate. The matrix form of the BARENCO Gate is shown below:

$$A(\theta, \alpha, \phi) = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & e^{i\alpha} \cos(\theta) & -ie^{i(\alpha-\phi)} \sin(\theta) \\ & & -ie^{i(\alpha+\phi)} \sin(\theta) & e^{i\alpha} \cos(\theta) \end{bmatrix}$$

This also has a similar construct to the Controlled-Unitary gate. Here the unitary is of the form:

$$U(\theta, \alpha, \phi) = \begin{bmatrix} e^{i\alpha} \cos(\theta) & -ie^{i(\alpha-\phi)} \sin(\theta) \\ -ie^{i(\alpha+\phi)} \sin(\theta) & e^{i\alpha} \cos(\theta) \end{bmatrix}$$

We now assess the action of this unitary on states $|0\rangle$ and $|1\rangle$ and hence on a general state.

$$U(\theta, \alpha, \phi)|0\rangle = \begin{bmatrix} e^{i\alpha} \cos(\theta) & -ie^{i(\alpha-\phi)} \sin(\theta) \\ -ie^{i(\alpha+\phi)} \sin(\theta) & e^{i\alpha} \cos(\theta) \end{bmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} e^{i\alpha} \cos(\theta) \\ -ie^{i(\alpha+\phi)} \sin(\theta) \end{pmatrix}$$

$$U(\theta, \alpha, \phi)|1\rangle = \begin{bmatrix} e^{i\alpha} \cos(\theta) & -ie^{i(\alpha-\phi)} \sin(\theta) \\ -ie^{i(\alpha+\phi)} \sin(\theta) & e^{i\alpha} \cos(\theta) \end{bmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -ie^{i(\alpha-\phi)} \sin(\theta) \\ e^{i\alpha} \cos(\theta) \end{pmatrix}$$

This unitary can be much more universal as well. For example, setting $\theta = \pi/2$ would make the unitary as:

$$U(\pi/2, \alpha, \phi) = \begin{bmatrix} 0 & -ie^{i(\alpha-\phi)} \\ -ie^{i(\alpha+\phi)} & 0 \end{bmatrix} = -ie^{i(\alpha-\phi)} \begin{bmatrix} 0 & 1 \\ e^{2i(\phi)} & 0 \end{bmatrix}$$

This has a global phase term so we shall ignore that for now. However, it clearly shows that we can achieve both a bit as well as phase flip. Hence, this allows us to encompass the sets of CX gates, CY Gates, and if used together with the help of auxiliary qubits, even CZ gates and controlled Phase-Shift gates can be developed, which encompass various universal sets.

In the case where ϕ , α and θ are chosen fixed irrational multiples of π and each other, we would be able to approximate any unitary just using a single gate. [6] [10]

Some Other MATLAB Simulations

I have done a few more calculations on MATLAB to verify a few of the properties we have learnt. Due to complications, I have not done simulations of all the Universal Gates but I have attempted to verify certain properties of the same. They are demonstrated below:

Verification of Classic Action using Toffoli Gates

```
ket0 = [1;0];
```

```
ket1 = [0;1];
```

```
k00 = kron(ket0,ket0);
```

```
k01 = kron(ket0,ket1);
```

```
k10 = kron(ket1,ket0);
```

```
k11 = kron(ket1,ket1);
```

```
k000 = kron(ket0,k00);
```

```
k001 = kron(ket0,k01);
```

```
k010 = kron(ket0,k10);
```

```
k011 = kron(ket0,k11);
```

```
k100 = kron(ket1,k00);
k101 = kron(ket1,k01);
k110 = kron(ket1,k10);
k111 = kron(ket1,k11);
```

```
X = [0 1;
      1 0];
```

```
% Here we are redefining the Toffolli as a
function of the control qubits
% c1 c2 and target qubit qt.
```

```
TOFFOLI_ = @(c1c2,t) all(c1c2==k11)*X*t +
~all(c1c2==k11)*t;
```

AND GATE for AB = 00, 01, 10, 11

```
disp('For AB=00')
disp(TOFFOLI_(k00,ket0))
disp('For AB=01')
disp(TOFFOLI_(k01,ket0))
disp('For AB=10')
disp(TOFFOLI_(k10,ket0))
disp('For AB=11')
disp(TOFFOLI_(k11,ket0))
```

```
%Note: The results would be printed in the
form of matrix representation of
%|0> and |1> and are hence verified.
```

For AB=00

```
1
0
```

For AB=01

```
1
0
```

For AB=10

```
1
0
```

For AB=11

```
0
1
```

NAND GATE for AB = 00, 01, 10, 11

```
disp('For AB=00')
disp(TOFFOLI_(k00,ket1))
disp('For AB=01')
disp(TOFFOLI_(k01,ket1))
disp('For AB=10')
disp(TOFFOLI_(k10,ket1))
disp('For AB=11')
disp(TOFFOLI_(k11,ket1))
```

```
%Note: The results would be printed in the
form of matrix representation of
%|0> and |1> and are hence verified.
```

For AB=00

```
0
1
```

For AB=01

```
0
1
```

For AB=10

```
0
1
```

For AB=11

```
1
0
```

NOT Gate for A = 0, 1

```
disp('For A=0')
disp(TOFFOLI_(k11,ket0))
disp('For A=1')
disp(TOFFOLI_(k11,ket1))
```

For A=0

```
0
1
```

For A=1

1
0

OR Gate for AB = 00, 01, 10, 11

X2 = kron(X,X); %Since here we have the X Gate acting on both Qubits

```
disp('For AB=00')
disp(TOFFOLI_(X2*k00,ket1))
disp('For AB=01')
disp(TOFFOLI_(X2*k01,ket1))
disp('For AB=10')
disp(TOFFOLI_(X2*k10,ket1))
disp('For AB=11')
disp(TOFFOLI_(X2*k11,ket1))
```

For AB=00

1
0

For AB=01

0
1

For AB=10

0
1

For AB=11

0
1

XOR (CNOT) Gate for AB = 00, 01, 10, 11

```
disp('For AB=00')
disp(TOFFOLI_(kron(ket1,ket0),ket0))
disp('For AB=01')
disp(TOFFOLI_(kron(ket1,ket0),ket1))
disp('For AB=10')
disp(TOFFOLI_(kron(ket1,ket1),ket0))
disp('For AB=11')
disp(TOFFOLI_(kron(ket1,ket1),ket1))
```

For AB=00

1
0

For AB=01

0
1

For AB=10

0
1

For AB=11

1
0

Hence, we can conclude that Toffoli Gates can be used to obtain any classical operation. We can use Toffoli Gates to develop Half and Full Adders for Quantum Circuits.

Published with MATLAB® R2023b

Generation of Toffoli GATE using CNOT, H, T and T† Gates

%Here, we shall attempt to construct the CCX Gate using CNOT,H,T and T† Gates. We can use that fact that T† = inverse of T

format short

```
ket0 = [1;0];
ket1 = [0;1];
k00 = kron(ket0,ket0);
k01 = kron(ket0,ket1);
k10 = kron(ket1,ket0);
k11 = kron(ket1,ket1);
```

```

k000 = kron(ket0,k00);
k001 = kron(ket0,k01);
k010 = kron(ket0,k10);
k011 = kron(ket0,k11);
k100 = kron(ket1,k00);
k101 = kron(ket1,k01);
k110 = kron(ket1,k10);
k111 = kron(ket1,k11);

CNOT = [1 0 0 0;
        0 1 0 0;
        0 0 0 1;
        0 0 1 0];
I = [1 0; 0 1];
H = (1/sqrt(2))*[1 1;
                 1 -1];

% Since it is a 3-Qubit System, we shall define CNOT as a 3-Qubit Gate with
% different sets of Ctrl and Target Qubits used.

CNOT_12 = kron(CNOT,I);
CNOT_23 = kron(I,CNOT);

CNOT_13 = [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 0 1
           0 0 0 0 0 0 1 0]; %This is derivable from the solutions.

T = [1 0;
      0 exp(1i*pi/4)];
Td = inv(T);

%Now we shall multiply the matrices in appropriate order. Will do it
%stepwise and display the matrix after the final operation is done

TOFFOLI_Const = kron(kron(I,I),H); %Step 1
TOFFOLI_Const = CNOT_23*TOFFOLI_Const; %Step 2
TOFFOLI_Const = kron(kron(I,I),Td)*TOFFOLI_Const; %Step 3
TOFFOLI_Const = CNOT_13*TOFFOLI_Const; %Step 4
TOFFOLI_Const = kron(kron(I,I),T)*TOFFOLI_Const; %Step 5
TOFFOLI_Const = CNOT_23*TOFFOLI_Const; %Step 6
TOFFOLI_Const = kron(kron(I,I),Td)*TOFFOLI_Const; %Step 7
TOFFOLI_Const = CNOT_13*TOFFOLI_Const; %Step 8
TOFFOLI_Const = kron(kron(I,T),T)*TOFFOLI_Const; %Step 9
TOFFOLI_Const = kron(CNOT,H)*TOFFOLI_Const; %Step 10
TOFFOLI_Const = kron(kron(T,Td),I)*TOFFOLI_Const; %Step 11
TOFFOLI_Const = CNOT_12*TOFFOLI_Const ;%Step 12

disp('The FINAL Matrix is as follows:')
disp(round(TOFFOLI_Const,10))

```

The FINAL Matrix is as follows:

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

```
%%Hence, it is verified that we obtain the original Toffoli Matrix.
```

Published with MATLAB® R2023b

These are just a few examples where we used universal gate sets for computation. With this, we conclude our reviews of Universal Gates as well. We did not go over the DEUTSCH and BARENCO sets of gates as they are not only parametric but would also require to go much deeper and it would be out of scope of the project.

Conclusion

Through this project, we were able to explore various gates, gate compositions and universal gate sets while also briefly diving into the construction of a basic Transmon qubit. There are still couple of spots which have not been filled up yet, which really tells us how much research is left in the field. While there are a fixed number of classical gates, the number of potential quantum gates are uncountable since there are infinitely many states that can exist in a two-level system. Limiting to some basic gates would allow us to make use of various quantum algorithms such as the famous Grover's and Shor's algorithms, while advanced applications would require us to synthesize gates as to our requirements. Luckily, the availability of various universal gate sets would allow us to obtain basically any unitary from just a few known gates, and hence combats the issue of hardware limitation.

References

- [1] Sangil Kwon, Akiyoshi Tomonaga, Gopika Lakshmi Bhai, Simon J. Devitt, Jaw-Shen Tsai; Gate-based superconducting quantum computing. *Journal of Applied Physics* 28 January 2021; 129 (4): 041102. <https://doi.org/10.1063/5.0029735>
- [2] Principles of Quantum Mechanics Tyagi, S. Tyagi, I.S. 9789332517721 <https://books.google.co.in/books?id=Pwg4nQAACAAJ> 2012 Dorling Kindersley (India)
- [3] Luke Polson Physics Blog - Numerically Finding the Eigenstates/Energies of a 1D Quantum System <https://lukepolsonphysicsblog.wordpress.com/2020/10/29/example-post-3/>
- [4] Lectures 16-21 (Zlatko K. Minev), 2020 Qiskit Global Summer School on Quantum Computing and Quantum Hardware <https://learn.qiskit.org/summer-school/2020/superconducting-qubits-ii-circuit-electrodynamics-readout-calibration-methods>
- [5] Nielsen, M.A. & Chuang, I.L., 2011. Quantum Computation and Quantum Information: 10th Anniversary Edition, Cambridge University Press.
- [6] Williams, C.P. (2011). Quantum Gates. In: Explorations in Quantum Computing. Texts in Computer Science. Springer, London. https://doi.org/10.1007/978-1-84628-887-6_2
- [7] IBM Quantum. <https://quantum-computing.ibm.com/>, 2021
- [8] Journal Article Deutsch, David Elieser, Penrose, Roger, Quantum computational networks, 1989, Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences, 73-90, 425, 1868, <https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1989.0099>
- [9] Ashok Muthukrishnan, "Classical and Quantum Logic Gates: An Introduction to Quantum Computing" Quantum Information Seminar, Friday, Sep. 3, 1999, Rochester Center for Quantum Information (RCQI) http://www2.optics.rochester.edu/~stroud/presentations/muthukrishnan991/LogicGate_s.pdf
- [10] Barenco Adriano 1995A universal two-bit gate for quantum computation Proc. R. Soc. Lond. A449679–683 <http://doi.org/10.1098/rspa.1995.0066>
- [11] Photograph by Smite-Meister, distributed under a CC BY-SA 3.0 license https://commons.wikimedia.org/wiki/File:Bloch_sphere.svg
- [12] A. Y. Kitaev, "Quantum Computations: Algorithms and Error Correction," Russ. Math. Surv., Volume 52, Issue 6 (1997) pp. 1191–1249.