Research Project Presentation

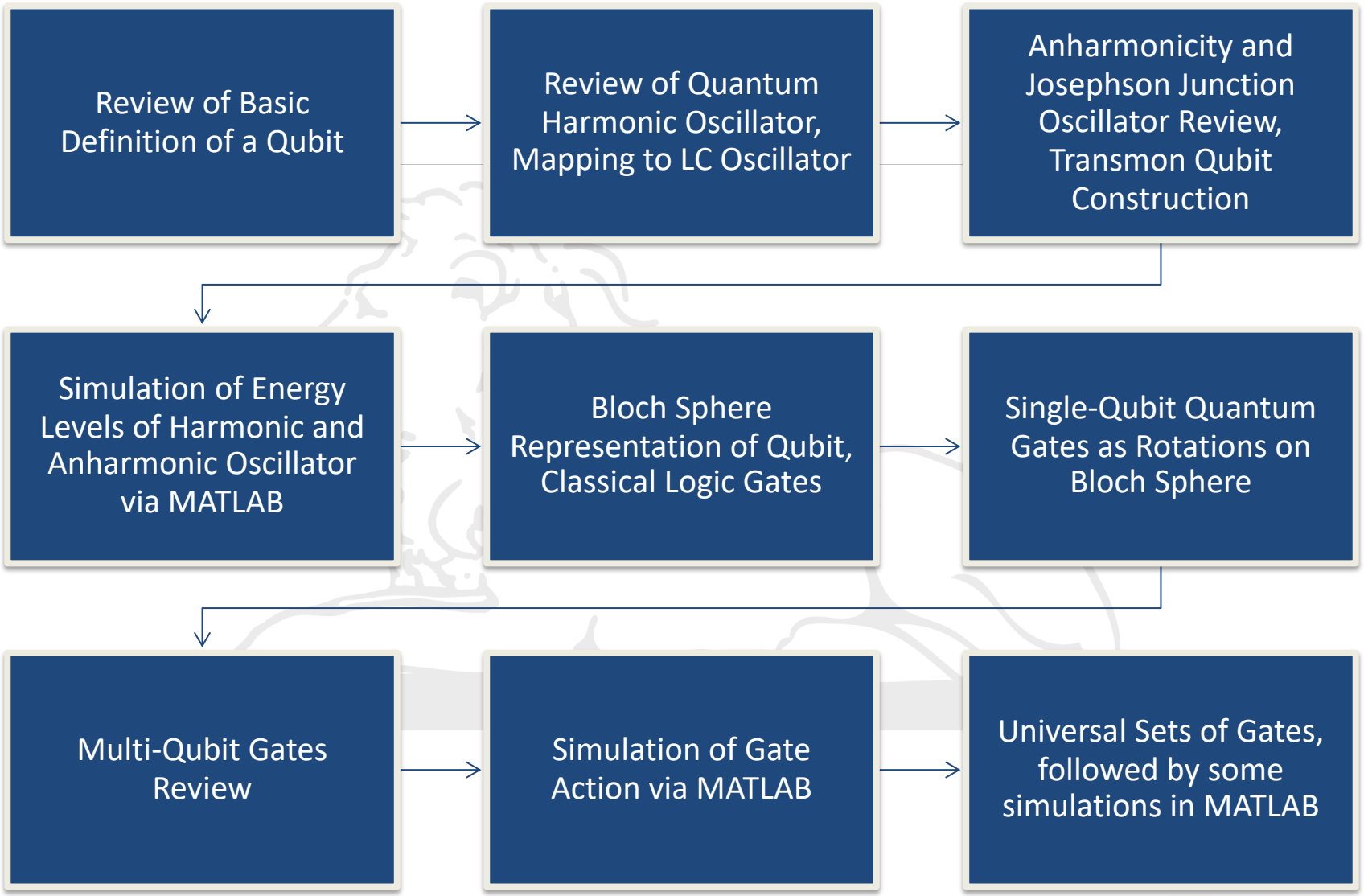
A Comparative Study of Quantum Logic Gates Suited For Current Algorithms

Aarav Ratra (21122002)

B.Tech Engineering Physics, IIT Roorkee

Supervisor: Dr. Ajay, Dept. of Physics

Roadmap


Review of Basic Definition of a Qubit
Review of Quantum Harmonic Oscillator, Mapping to LC Oscillator
Anharmonicity and Josephson Junction Oscillator Review, Transmon Qubit Construction
Simulation of Energy Levels of Harmonic and Anharmonic Oscillator via MATLAB
Bloch Sphere Representation of Qubit, Classical Logic Gates
Single-Qubit Quantum Gates as Rotations on Bloch Sphere
Multi-Qubit Gates Review
Simulation of Gate Action via MATLAB
Universal Sets of Gates, followed by some simulations in MATLAB

What is a Qubit?

Definition of a Qubit

- Fundamental unit of Quantum Information
- Combination of "quantum" and "bit”
- Unlike classical bits, a qubit can exist in the superposition of two states (i.e., 0 and 1), hence representing both simultaneously.

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

- Properties of quantum mechanics such as superposition and entanglement followed by these qubits would allow us to develop algorithms which outperform their classical counterparts.

How are Qubits implemented?

Any Two-Level Quantum System

We can implement Qubits by isolating two levels from a Multi-Level Quantum System

This leads to multiple implementations such as:

- Photonic Qubits
- Single Atom Qubits
- Ion Trap Qubits
- Superconducting Qubits
- Spintronic Qubits
- NV Centre Qubits

Review – Quantum Harmonic Oscillator

- Hamiltonian of Quantum Harmonic Oscillator

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

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

- $\hat{a}^\dagger$ and $\hat{a}$ = Creation and Annihilation Operators
- $\hat{a}^\dagger = \frac{-i\hat{p}+m\omega\hat{x}}{\sqrt{2m\hbar\omega}}$ and $\hat{a} = \frac{i\hat{p}+m\omega\hat{x}}{\sqrt{2m\hbar\omega}}$.
- On finding Hamiltonian of $\hat{a}^\dagger|\psi_n\rangle$ and $\hat{a}\,|\psi_n\rangle$ we get

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

$$\hat{H}(\hat{a}|\psi_n\rangle) = (E_n - \hbar\omega)(\hat{a}|\psi_n\rangle)$$

- By definition, if $|0\rangle$ is ground state then $\hat{a}\,|0\rangle = 0$. Using this with Hamiltonian we obtain zero-point energy = $\frac{1}{2}\hbar\omega$.

[2] Principles of Quantum Mechanics Tyagi, S. Tyagi, I.S. 9789332517721 2012 Dorling Kindersley (India)

Review – Quantum Harmonic Oscillator

- 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 $\hat{a}$ 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:

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

- We solve for eigenvalue of $\hat{a}$ and $\hat{a}^\dagger$ by assuming them to be variables.

$$\hat{a}^\dagger|n\rangle = \lambda_n|n+1\rangle\,, \hat{a}|n\rangle = \mu_n|n-1\rangle\ (0\ for\ n=0)$$

- Calculate $\langle n|\hat{a}^\dagger\hat{a}|n\rangle$ and $\langle n|\hat{H}|n\rangle$ to obtain λ_n and μ_n. Final Result:

$$\hat{a}^\dagger|n\rangle = \sqrt{n+1}|n+1\rangle$$
$$\hat{a}|n\rangle = \sqrt{n}|n-1\rangle\ (0\ for\ n=0)$$

[2] Principles of Quantum Mechanics Tyagi, S. Tyagi, I.S. 9789332517721 2012 Dorling Kindersley (India)

Review – Quantum Harmonic Oscillator

- Define Number Operator $\hat{n} = \hat{a}^\dagger \hat{a}$.
- Calculation of Matrix Elements for $\hat{a}^\dagger, \hat{a}$:

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

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

- Hence we obtain the matrix form of $\hat{a}^\dagger$, $\hat{a}$ and hence H

$$\hat{a}^\dagger = \begin{bmatrix} 0 & 0 & 0 & \\ \sqrt{1} & 0 & 0 & \cdots \\ 0 & \sqrt{2} & 0 & \\ 0 & 0 & \sqrt{3} & \\ & \vdots & & \ddots \end{bmatrix} \quad \hat{a} = \begin{bmatrix} 0 & \sqrt{1} & 0 & 0 & \\ 0 & 0 & \sqrt{2} & 0 & \cdots \\ 0 & 0 & 0 & \sqrt{3} & \\ & \vdots & & & \ddots \end{bmatrix} \quad H = \begin{bmatrix} \frac{1}{2}\hbar\omega & 0 & 0 & & \\ 0 & \frac{3}{2}\hbar\omega & 0 & & \cdots \\ 0 & 0 & \frac{5}{2}\hbar\omega & & \\ & & & \frac{7}{2}\hbar\omega & \\ & \vdots & & & \ddots \end{bmatrix}$$

[2] Principles of Quantum Mechanics Tyagi, S. Tyagi, I.S. 9789332517721 2012 Dorling Kindersley (India)

Review – Quantum Harmonic Oscillator

- Now we solve for Wave-Function $\psi_n(x)$.
- Let $\beta = \sqrt{\frac{m\omega}{\hbar}}$,we get $\hat{a} = \frac{1}{\beta\sqrt{2}}\frac{d}{dx} + \frac{\beta}{\sqrt{2}}\hat{x}$ $\hat{a}^\dagger = -\frac{1}{\beta\sqrt{2}}\frac{\partial}{\partial x} + \frac{\beta}{\sqrt{2}}\hat{x}$
- We use $\hat{a}\,|0\rangle = 0$ to solve for $\psi_0(x)$ and normalize

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

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

- Hence, general expression for $\psi_n(x)$ can be written as:

$$\psi_n(x) = \hat{a}^\dagger\,\psi_{n-1}(x)/\sqrt{n}\;\psi_n(x) = \left(\hat{a}^\dagger\right)^n\psi_0(x)/\sqrt{n!}$$

We now move on to simulating these levels using MATLAB

[2] Principles of Quantum Mechanics Tyagi, S. Tyagi, I.S. 9789332517721 2012 Dorling Kindersley (India)

Simulation of QHO Energy Levels using MATLAB

Obtaining Energy Levels

```
%% Matrix Formalism

N = 10; %Number of energy levels we are restricting ourselves to for simplicity


%Defining creation and anhillation operators usin matrices
a = diag(sqrt(1:N-1),1);
ad= diag(sqrt(1:N-1),-1);

num = ad*a; %Number operator = ad*a

hcr = 6.582119569E-16; %Reduced Planck's Const (in eV)
vbar = 300; %cm^-1
c = 2.99792458E10; %cm/s
w = 2*pi*vbar*c;
m = 6.2224e-26; %Arbitary value chosen

Id = diag(ones(1,N)); %Identity Matrix

H = (num + Id/2)*hcr*w;
%X = sqrt(hcr/(2*m*w))*(ad + a);

fprintf('Given that ħω = %1.5f \n',hcr*w)
%eigenvectors and eigenvalues of the Energy Operator
fprintf('The energy eigenvalues are as follows:')
H_eigenvalues = eig(H)
fprintf('The Hamiltonian Matrix is as described:\n')
disp(H)
```

```
Given that ħω = 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
```

Simulation of QHO Energy Levels using MATLAB

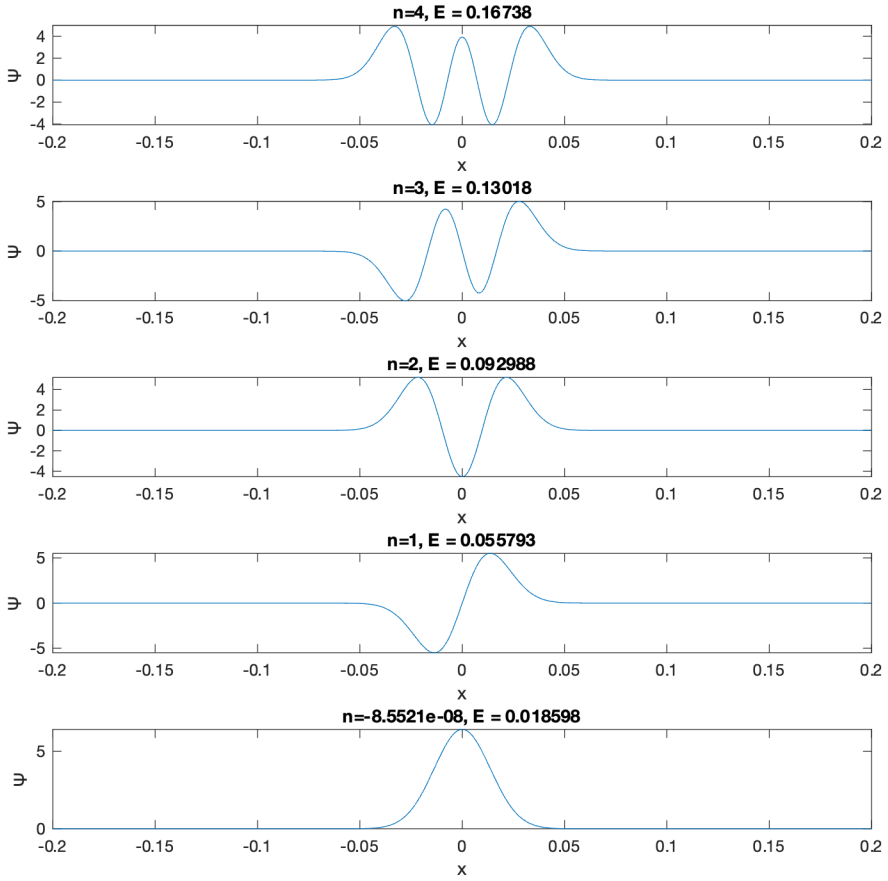
Obtaining Eigenstates

```
%%
% On obtaining the eigenvalues of the above, we can attempt to show the
% energy levels and wavefunctions.
%We first solve for 0th eigenstate, then we use the creation operator to
%derive the rest.

%a|0> = 0 => we obtain Ψ0 through a first order differential equation.
%I shall be using the Symbolic Math library of MATLAB which allows us to
%deal with functions and derivatives in a more convenient manner.
syms x;
beta = sqrt(m*w/hcr)

%Declaring ad_ and a_ as anonymous symbolic functions
ad_ = @(ps) -diff(ps)/(beta*sqrt(2)) + beta*x*ps/sqrt(2);
a_ = @(ps) diff(ps)/(beta*sqrt(2)) + beta*x*ps/sqrt(2);
n_ = @(ps) ad_(a_(ps)); %Number Operator

n_avg = @(ps) int(n_(ps)/ps,-1000,1000)/2000 ; %I have computed the average eigenvalue of the n operator.
%Note: This function requires more computational resources than my local
%system. This does not really return suitable value for n >=3 within
%feasable time and ends up crashing matlab. Hence, i shall only demonstate
%it for n = 0,1,2

psi_0 = sqrt(beta/sqrt(pi))*exp(-0.5*beta^2*x^2); %Manually calculated

```

```
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
```

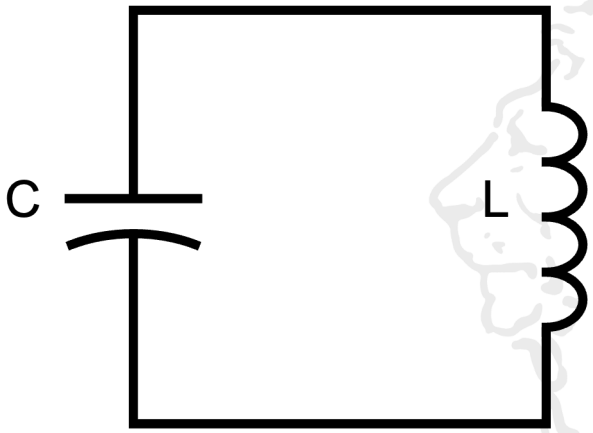
Simulation of QHO Energy Levels using MATLAB

```
%%
%I shall be running a loop for eigenstates we are dealing with. I would be
%dealing with only the first 5 eigenstates, hence, I am reassigning N = 5

N=5;
p = psi_0;

subplot(N+1,1,N+1)
xline(0)
yline(0)
fplot (psi_0,[-.2,.2])


for i=1:N
    if(i<=3)
        n = eval(n_avg(p))
        disp('Above n evaluated using the operator')
    else
        n = i-1
    end
    E = (n +1/2);
    p = ad_(p)/sqrt(i);
    subplot(N+1,1,N+1-i)
    fplot (p,[-.2,.2])
    xlabel('x')
    ylabel('Ψ')
    title('n='+string(n)+', E = '+string(E))
end
```



Quantum Mechanical Treatment of the LC Oscillator Circuit

- We can derive the Hamiltonian of the LC Circuit and observe similarities to the Linear Harmonic Oscillator



Kirchhoff's Laws

$$\sum \dot{Q}_i = 0$$
$$\sum \dot{\phi}_i = 0$$

$$\dot{\phi}_c - \dot{\phi}_L = 0 \Rightarrow \phi_c = \phi_L \, (= \phi)$$

$$\dot{Q}_c + \dot{Q}_L = 0 \Rightarrow C\ddot{\phi} + \frac{\phi}{L} = 0$$

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

This bears similarity to $\ddot{x} = -\omega^2 x$

$$H = \mathcal{E}_C + \mathcal{E}_L = \frac{Q^2}{2C} + \frac{\phi^2}{2L}$$

Compare with $H = \frac{p^2}{2m} + \frac{1}{2}kx^2$

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

Quantum Mechanical Treatment of the LC Oscillator Circuit

- This proves that there is a clear correspondence between the LC Circuit and the Linear Harmonic Oscillator.
- $\phi \Leftrightarrow x\ ; Q \Leftrightarrow p\ ; m \Leftrightarrow C\ ; \frac{1}{L} \Leftrightarrow k$
- Also define $\omega = \frac{1}{\sqrt{LC}}$, $Z = L/C$
- Writing Hamiltonian in terms of operators (using method of Classical Analogy)

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

Note that $\{O, O'\} = \frac{1}{i\hbar}[O, O']$
Hence, the term of ħ comes in the final solution.
We have used Classical Analogy to obtain the operator form.

- This proves that the solution would be of similar form as QHO.

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

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

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

Quantum Mechanical Treatment of the LC Oscillator Circuit

- We can express operators $\hat{\phi}$ and $\hat{Q}$ in terms of â and â†:

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

- Zero-Point Fluctuation: variance in flux/charge for eigenstate $|0\rangle$
- Expectation Value of operators $\hat{\phi}$ and $\hat{Q}$ found to be 0:

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

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

Hence, the ZPF Values are calculated as follows:

$$(\phi_{ZPF})^2 = \langle \hat{\phi}^2 \rangle = \langle 0|\hat{\phi}^2|0\rangle = \hbar Z/2 \qquad (Q_{ZPF})^2 = \langle \hat{Q}^2 \rangle = \langle 0|\hat{Q}^2|0\rangle = \hbar/2Z$$

Rewriting operators $\hat{\phi}$ and $\hat{Q}$:

$$\hat{\phi} = \phi_{ZPF}(\hat{a} + \hat{a}^\dagger) \qquad \hat{Q} = -iQ_{ZPF}(\hat{a} - \hat{a}^\dagger)$$

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

Limitations of Harmonic Oscillator Systems

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.

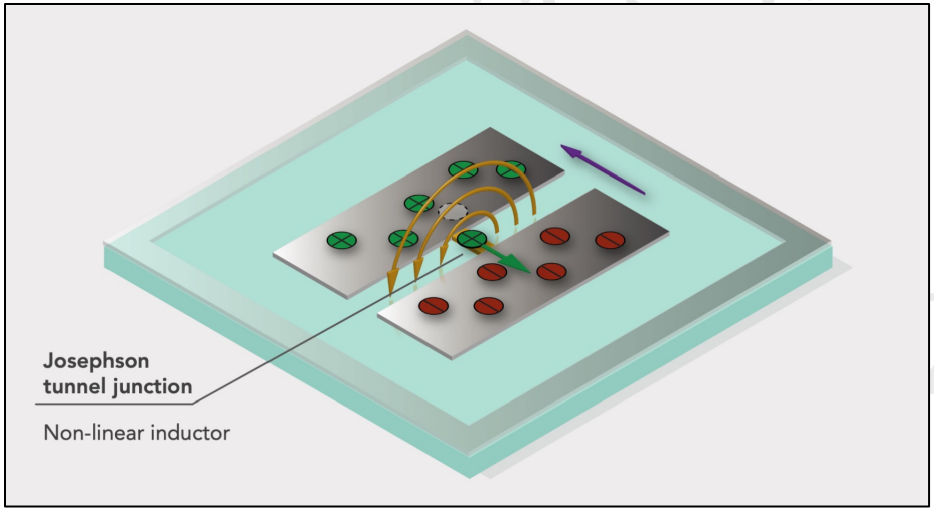
Anharmonicity and Introduction to the Transmon Qubit

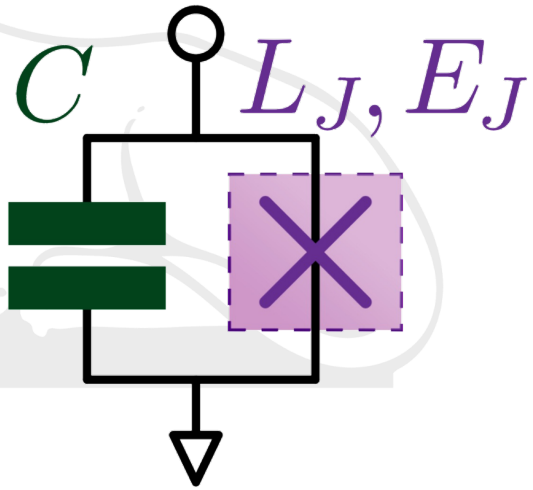
- The Hamiltonian of the Anharmonic Oscillator is given by:

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

(We shall not solve it, but shall attempt to numerically find its eigenstates and energy eigenvalues)

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.





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

Anharmonicity and Introduction to the Transmon Qubit

- Energy of Josephson Junction:

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

- Hence, Hamiltonian of the JJ-Oscilator:

$$H = \mathcal{E}_C + \mathcal{E}_j = \frac{Q^2}{2C} + E_j\left(\frac{\left(\frac{\phi_j}{\phi_0}\right)^2}{2!} - \frac{\left(\frac{\phi_j}{\phi_0}\right)^4}{4!} + \cdots\right)$$

Substituting $E_j = \phi_0^2/L_j$ In Hamiltonian

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

- L_j = Effective Linear Inductance
- Clearly, the system resembles an anharmonic oscillator on neglecting higher order terms.

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

Anharmonicity and Introduction to the Transmon Qubit

- We can write our Hamiltonian as:

$$\hat{H} = \frac{\hat{Q}^2}{2C} + \frac{\hat{\phi}_j{}^2}{2L} - \frac{\hat{\phi}_j{}^4}{L_j 4!} = \hat{H}_{QHO} - \frac{\hat{\phi}_j{}^4}{L_j 4!}$$

- 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.

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

Simulating 1-D Potentials using MATLAB

- Solving these equations can be cumbersome. Hence, we rely on numerical methods. One such method is described below:
- 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 ħ=1, and non-dimensionalise the equation by expressing x = L*y,

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

Note that Boundary Conditions still hold, so $\psi(y = 0) = \psi(y = 1) = 0$

[3] Luke Polson Physics Blog - Numerically Finding the Eigenstates/Energies of a 1D Quantum System

Simulating 1-D Potentials using MATLAB

- We use a discrete definition of the 2nd Derivative:

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

- Substitute in S.E.:

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

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

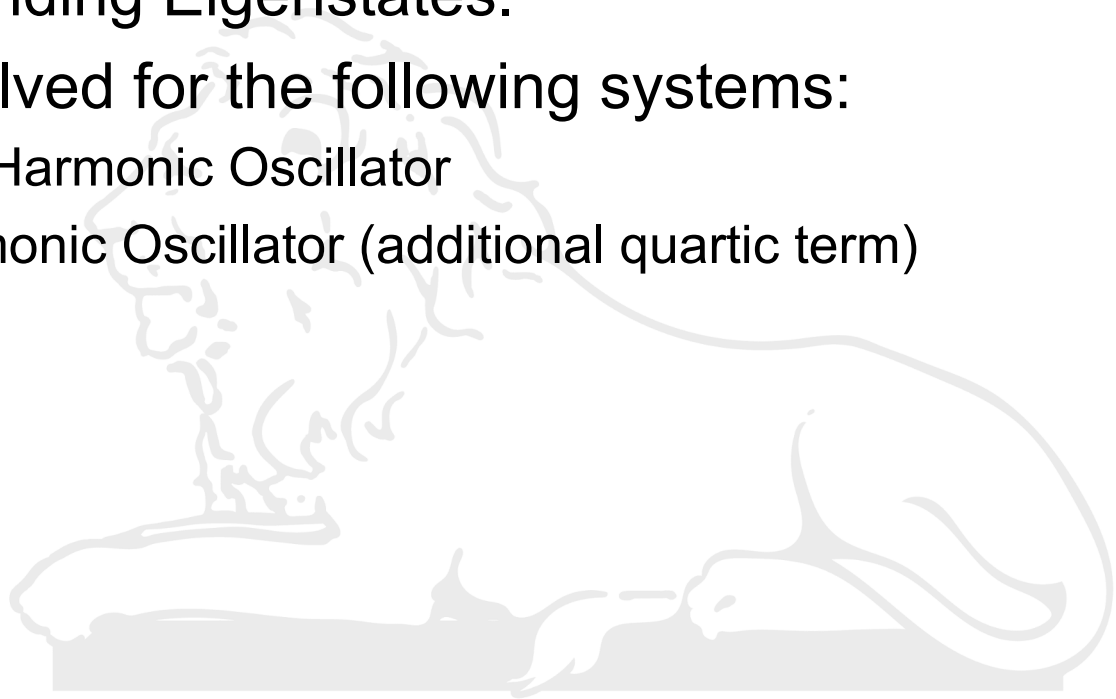
Express above family in matrix form for j = 1 to N-1

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

[3] Luke Polson Physics Blog - Numerically Finding the Eigenstates/Energies of a 1D Quantum System

Simulating 1-D Potentials using MATLAB

- Using this equation, we can obtain the energy levels through the eigenvalues of our LHS Matrix and hence find the corresponding Eigenstates.
- I have solved for the following systems:
  - Linear Harmonic Oscillator
  - Anharmonic Oscillator (additional quartic term)



[3] Luke Polson Physics Blog - Numerically Finding the Eigenstates/Energies of a 1D Quantum System

Simulating 1-D Potentials using MATLAB

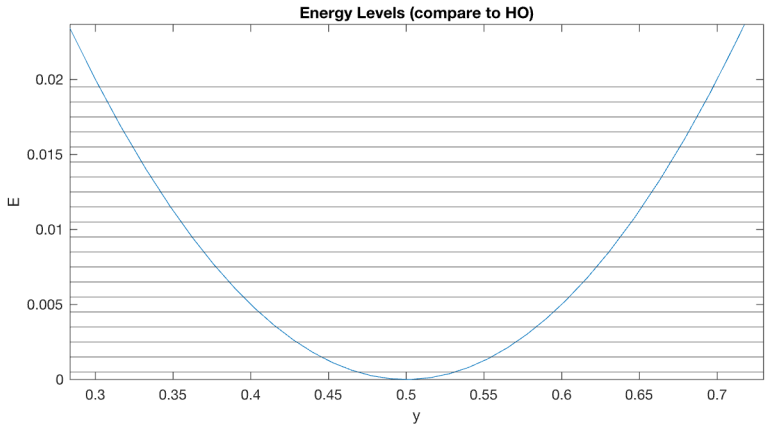
Linear Harmonic Oscillator

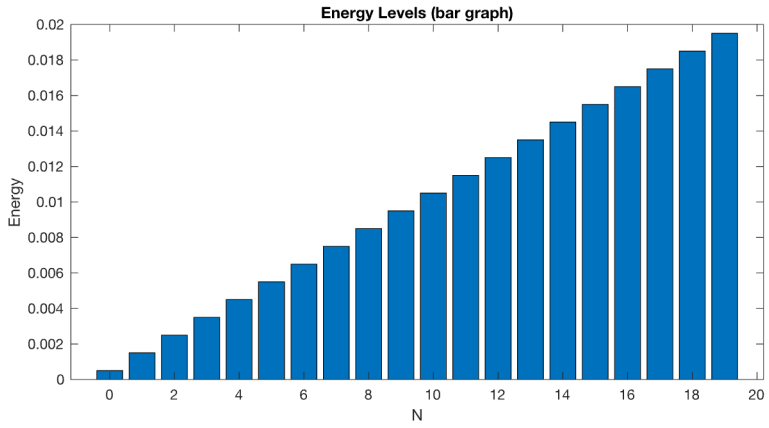
```
%% Part 1 : QHO
f1 = figure();
figure(f1)
N = 10000;
dy=1/N;
y = linspace(0,1,N+1);
L = 1000;
m=1;
k=1;

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

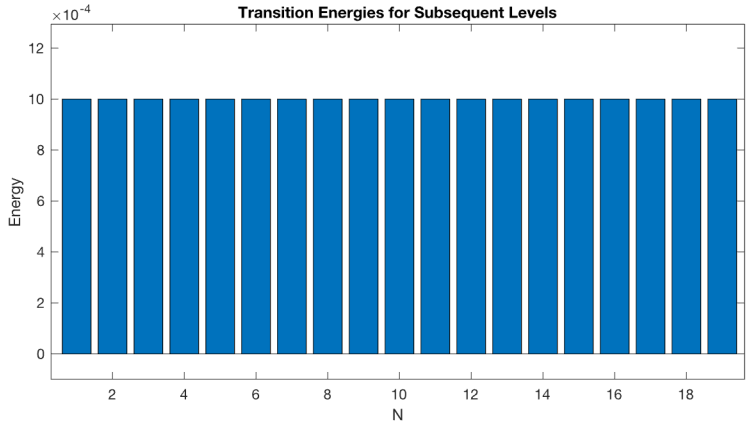
```
Lines = [];
for i=1:19
    Lines = [Lines , E(i+1)-E(i)];
end
subplot(2,2,3)
bar(Lines)
title('Transition Energies for Subsequent Levels')
xlabel('N')
ylabel('Energy')

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')
```

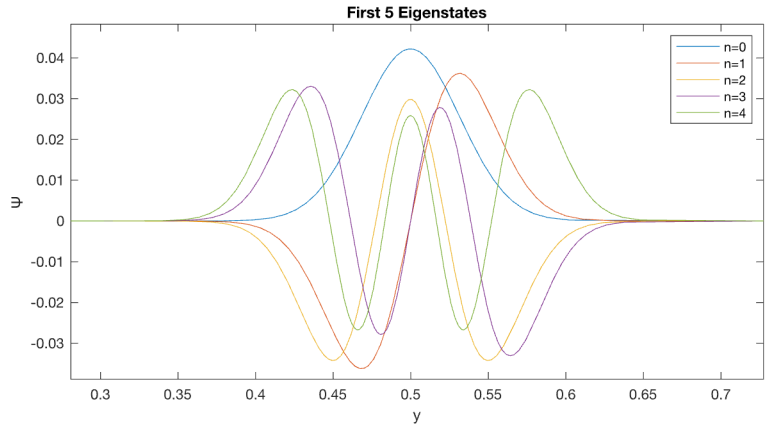
Simulating 1-D Potentials using MATLAB

Result for Linear Harmonic Oscillator









Simulating 1-D Potentials using MATLAB

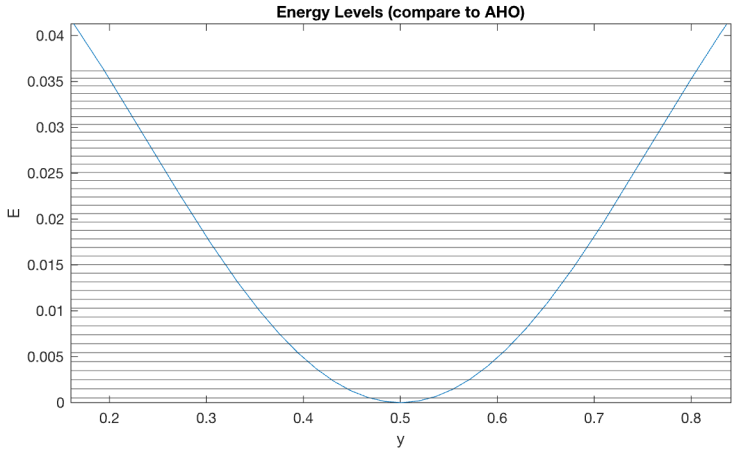
Anharmonic Oscillator

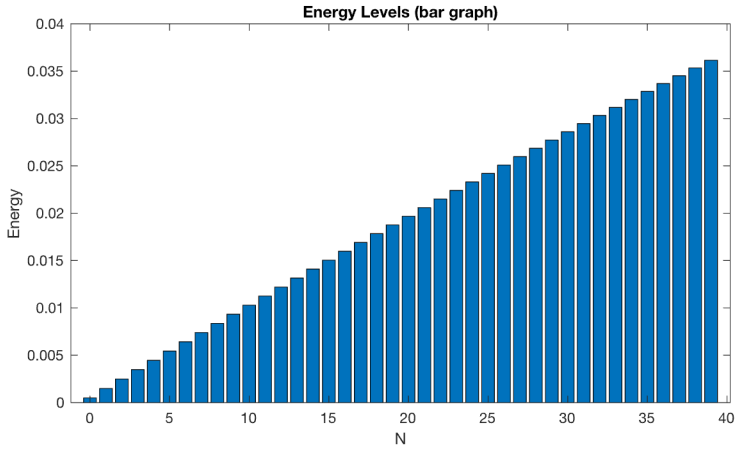
```
%% Part 2: Recreating the same for a potential with an additional anharmonic term
f2 = figure();
figure(f2)

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
yline(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')
```

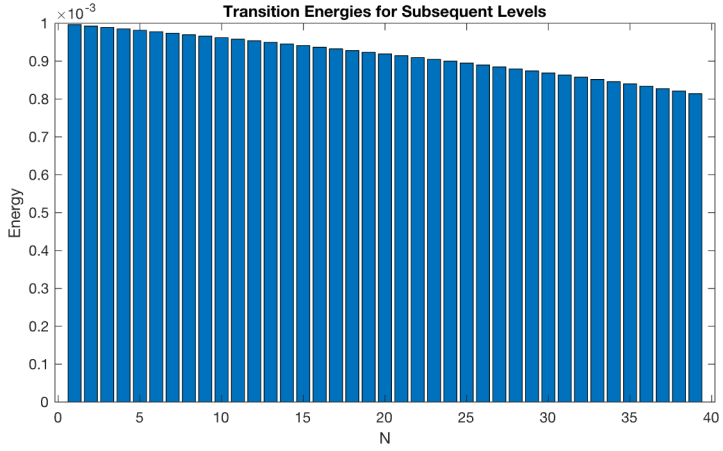
```
Lines = [];
for i=1:39
    Lines = [Lines , E(i+1)-E(i)];
end
subplot(2,2,3)
bar(Lines)
title('Transition Energies for Subsequent Levels')
xlabel('N')
ylabel('Energy')

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')
```

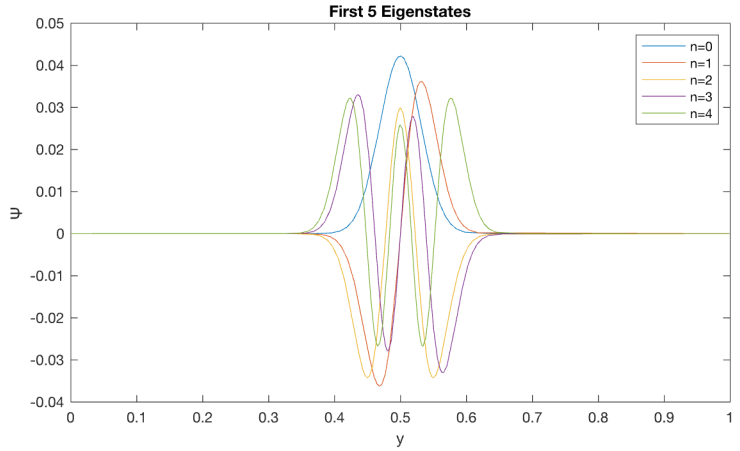
Simulating 1-D Potentials using MATLAB

Result for Anharmonic Oscillator









Simulating 1-D Potentials using MATLAB

Findings:

- We were able to get a fairly accurate approximation for the Harmonic Oscillator Potential as the energy levels were equally spaced for lower levels of N.
- The anharmonic oscillator has roughly similar eigenstates but the energy levels are spaced unequally and the gap between energy levels decreases as we go on to higher energy eigenvalues.
- Hence, in an anharmonic oscillator type system, we can separate the lower two energy states and use them as our basis states.
- Since the Josephson Junction Oscillator has a Hamiltonian similar to the Anharmonic Oscillator, we can use it to construct our Transmon Qubit

Basics of the Qubit

For some time, we shall move away from the physical picture of the qubit and shift towards a more mathematical 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:

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

- Here 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:

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

Bloch Sphere Representation

- We now wish to visualize the state of the qubit geometrically. Note that α and β are complex coefficients and hence have their own magnitudes and arguments, giving us 4 degrees-of-freedom required to visualise information. Hence, we would need to use 4 dimensions to visualise the state of a qubit.
- 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. This reduces our problem to 3 degrees-of-freedom.

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:

$$|\psi\rangle = ae^{i\varphi_a}|0\rangle + be^{i\varphi_b}|1\rangle = e^{i\varphi_a}\left(a|0\rangle + be^{i(\varphi_b - \varphi_a)}|1\rangle\right)$$

We neglect $e^{i\varphi_a}$ since it contains information only about global phase. Hence our state reduces to:

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

Where $\varphi_b - \varphi_a = \varphi$ and is called 'Local Phase'

Bloch Sphere Representation

- We have already reduced our problem to 3-degrees of freedom and the state of the system is a function of a, b and ϕ.
- We also have another constraint $a^2 + b^2 = 1$, which gives us the idea of a trigonometric relationship between a and b. If we substitute a and b as functions of some angle θ such that any arbitrary state can be represented by a unique point on a unit sphere.

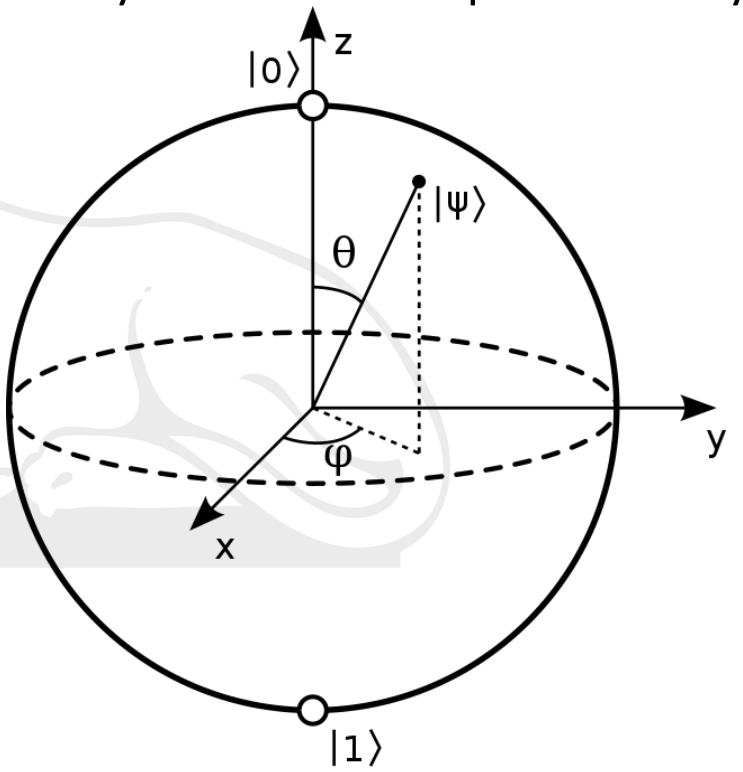
We substitute a=cos(θ/2) and b=sin(θ/2)

Hence, our state is represented as:

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

Refer to diagram attached:

Any point (r=1, θ, ϕ) can be used to represent the state of a qubit.



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

Density Matrix Representation of a Qubit

The state of a qubit can be represented by its density matrix as well.

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

The density matrix can also represent mixed and entangled states as well, and is hence also frequently used.

[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

Primer into Classical Gates

Before diving into Quantum Gates, we shall see a brief summary of Classical Logic Gates.

There are 6 Main Classical Logic Gates

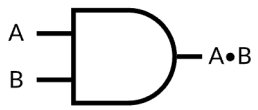
- AND Gate
- OR Gate
- NOT Gate
- NAND Gate
- NOR Gate
- XOR Gate

These are non-reversible. There also exist some reversible gates such as Toffoli and Fredkin Gates. However, these gates are also central to Quantum Computing and we shall see these gates when reviewing Multi-Qubit Quantum Gates

Primer into Classical Gates

AND GATE

Diagram



Boolean Representation

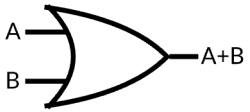
A•B

Truth Table

A	B	A•B
0	0	0
0	1	0
1	0	0
1	1	1

OR GATE

Diagram



Boolean Representation

A+B

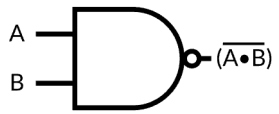
Truth Table

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

Primer into Classical Gates

NAND GATE

Diagram



Boolean Representation

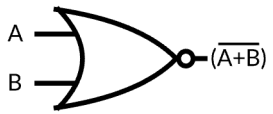
$(A \bullet B)' = A' + B'$

Truth Table

A	B	(A•B)'
0	0	1
0	1	1
1	0	1
1	1	0

NOR GATE

Diagram



Boolean Representation

$(A + B)' = A' \bullet B'$

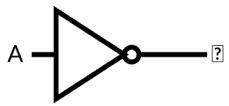
Truth Table

A	B	(A+B)'
0	0	1
0	1	0
1	0	0
1	1	0

Primer into Classical Gates

NOT GATE

Diagram



Boolean Representation

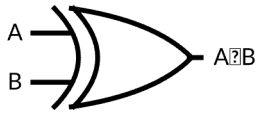
A’

Truth Table

A	A'
0	1
1	0

XOR GATE

Diagram



Boolean Representation

A⊕B

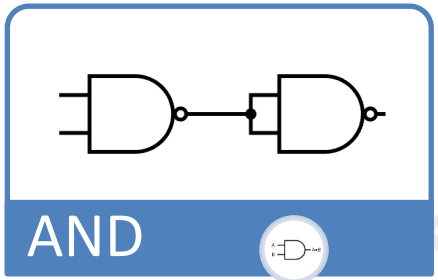
Truth Table

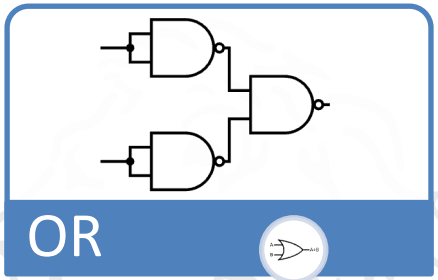
A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

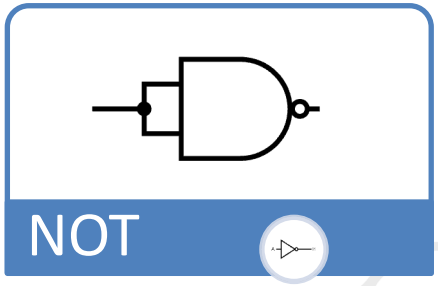
Universality in Classical Gates

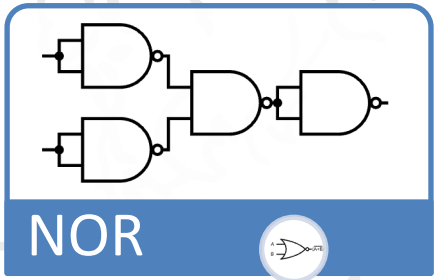
The NAND and NOR Gates are called universal since you can build any gate using NAND or NOR Gates

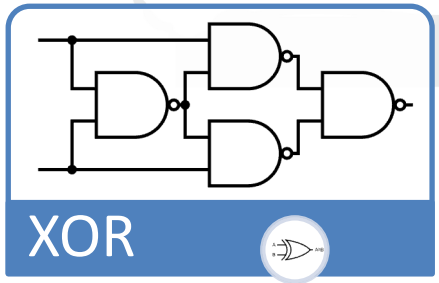
NAND



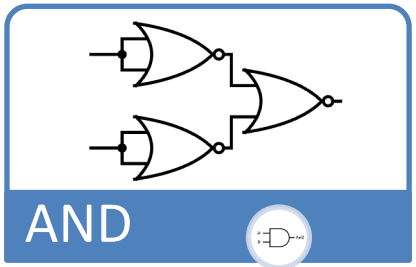


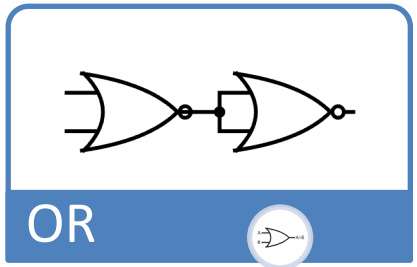


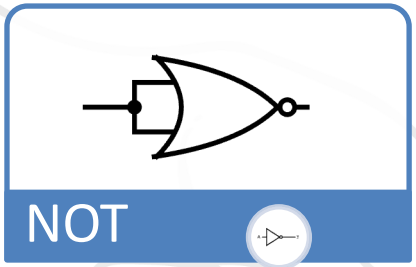


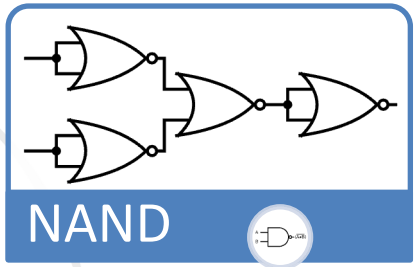


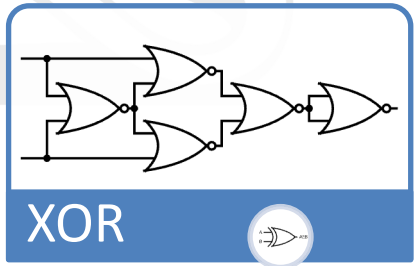
NOR











Classical v/s Quantum Gates

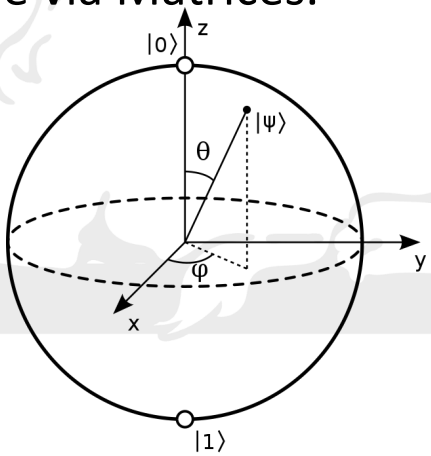
- Classical Gates are implemented using Transistors. They operate on Boolean Logic and are hence used to solve Boolean Functions.
- The major difference between Classical and Quantum Gates are their implementation.
- A qubit's construction is fundamentally different as compared to that of a classical 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.

Single-Qubit Quantum Gates as Rotations on 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.

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

- The action of a single-qubit gate can be expressed as a series of rotations on the Bloch Sphere.
- We can also represent gates by matrices. We shall start with representing rotations on the Bloch Sphere via Matrices.



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

Single-Qubit Quantum Gates as Rotations on Bloch Sphere

- 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:

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

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

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

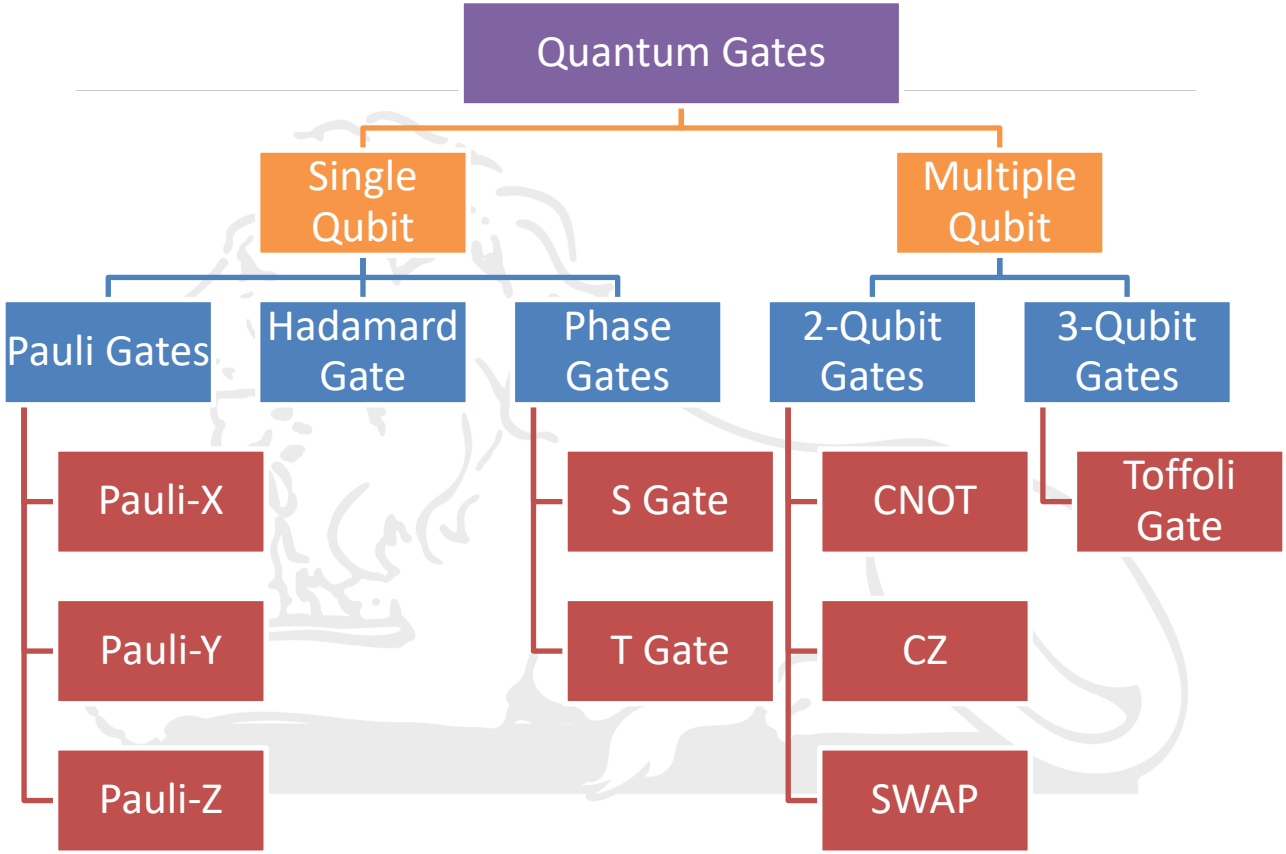
$$R_y(\Omega) = \begin{bmatrix} \cos\left(\frac{\Omega}{2}\right) & -\sin\left(\frac{\Omega}{2}\right) \\ \sin\left(\frac{\Omega}{2}\right) & \cos\left(\frac{\Omega}{2}\right) \end{bmatrix}$$

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

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.

Types of Gates

We shall see the following categories of gates



Pauli Gates

Pauli Gates are described as 180º rotations along the X, Y and Z Axes.

These comprise of 3 Gates:

- Pauli-X
- Pauli-Y
- Pauli-Z

We shall look into each of these gates one-by-one.

(Note: The Gate Symbols have mostly been drawn using IBM's Quantum Composer, and hence I have referenced IBM as well.)

Pauli-X Gate

- The Pauli-X Gate represents a 180º Rotation about the X-Axis.
- This gate is also called the 'Bit-Flip' Gate and is the Quantum Analogue of the classical NOT gate.

The matrix representation and gate action is shown below:

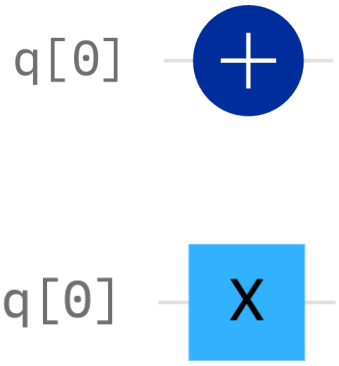
$$X = R_x(\pi) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = |0\rangle\langle 1| + |1\rangle\langle 0|$$

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

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

$$X(\alpha|0\rangle + \beta|1\rangle) = \alpha|1\rangle + \beta|0\rangle$$

Symbol:



This gate is reversible.

[7] IBM Quantum. https://quantum-computing.ibm.com/, 2021

Pauli-Z Gate

- The Pauli-X Gate represents a 180º Rotation about the Z-Axis.
- This gate is also called the 'Phase-Flip' Gate as it changes the phase by a factor of π
- The matrix representation and gate action is shown below:

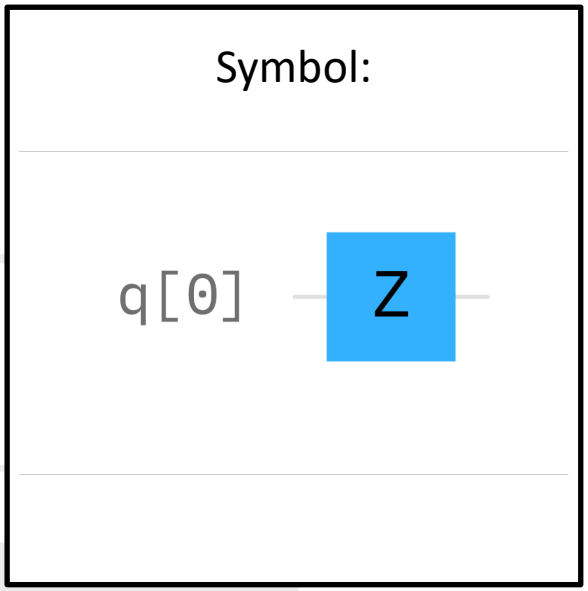
$$Z = R_Z(\pi) = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1|$$

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

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

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

Symbol:



This gate is reversible.

[7] IBM Quantum. https://quantum-computing.ibm.com/, 2021

Pauli-Y Gate

- The Pauli-Y Gate represents a 180º Rotation about the Y-Axis.
- This gate does both a bit and a phase flip. It can be used when we wish to flip the bits and bring about an additional phase of π.

The matrix representation and gate action is shown below:

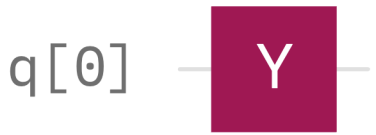
$$Y = R_Y(\pi) = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = i|1\rangle\langle 0| - i|0\rangle\langle 1|$$

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

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

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

Symbol:



This gate is reversible.

[7] IBM Quantum. https://quantum-computing.ibm.com/, 2021

Hadamard Gate

- The Hadamard Gate is one of the most important gates as it has the ability to generate superposition states.
- It can be represented either by a 180º rotation about the line z=x, or via a rotation of π/2 about Y axis followed by a rotation of π about the X axis.

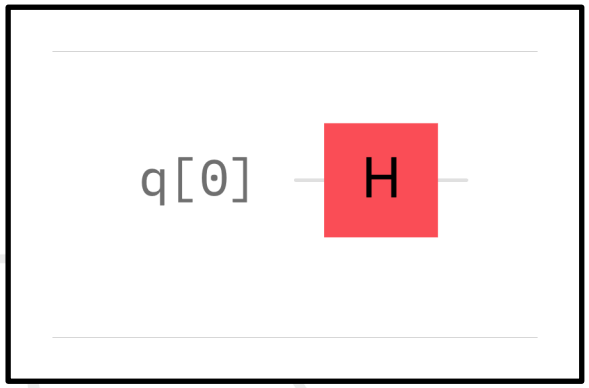
$$H = R_x(\pi)R_y\left(\frac{\pi}{2}\right) = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle$$

$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle$$

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

Symbol:



[7] IBM Quantum. https://quantum-computing.ibm.com/, 2021

Hadamard Gate

- 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-Jozsaa Algorithm, Bernstein-Vazirani Algorithm and Grover's Algorithm.
- The Hadamard Gate is also reversible.

Phase Shift Gates

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 Gate / Phase Gate

- Provides a phase shift of π/2

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

Symbol: q[0] — S

T Gate / π/8 Gate

- Provides a phase shift of π/4

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

Symbol: q[0] — T

These gates are not reversible.

[7] IBM Quantum. https://quantum-computing.ibm.com/, 2021

Some Useful Identities Related to 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

- 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:

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}, |01\rangle = |0\rangle \otimes |1\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}$$

$$|10\rangle = |1\rangle \otimes |0\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}, |11\rangle = |1\rangle \otimes |1\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

- The tensor-product can also be used with gates to represent action of multiple single-qubit gates simultaneously.
- Multiple-qubit Gates are represented by matrices as well.

CNOT 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. It is used in various algorithms.
- 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 C-NOT gate is considered to be the classical analogue of the XOR gate.

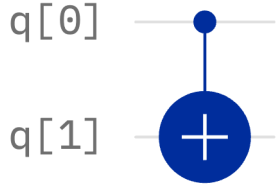
The matrix representation and the gate action is shown alongside,

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

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

$$CNOT|00\rangle = |00\rangle, CNOT|01\rangle = |01\rangle,$$
$$CNOT|10\rangle = |11\rangle, CNOT|11\rangle = |10\rangle$$

Symbol:



[7] IBM Quantum. https://quantum-computing.ibm.com/, 2021

CNOT Gate

Listed below are some well known applications of the CNOT Gate

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:

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

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 next.

CZ Gate

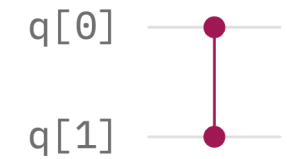
- The controlled-Z gate, also known as the CZ gate, is another important two-qubit gate.
- Like the CNOT Gate, the CZ gate also takes in two inputs i.e., the control and target qubits, but instead applies a phase flip to the target qubit when the control qubit is in state '1'.

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

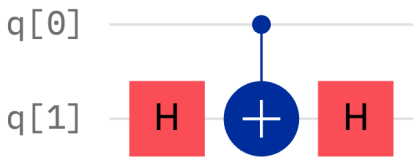
- $CZ|00\rangle = |00\rangle$
- $CZ|01\rangle = |01\rangle$
- $CZ|10\rangle = |10\rangle$
- $CZ|11\rangle = -|11\rangle$

The matrix representation and the gate action is shown alongside.

Symbol:



We can also construct the CZ Gate using the help of CNOT and H Gates. We use the identity Z=HXH:



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.

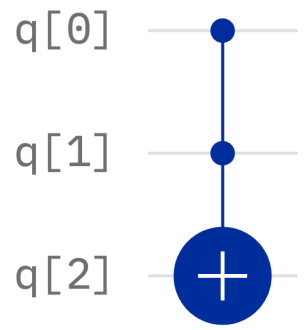
[7] IBM Quantum. https://quantum-computing.ibm.com/, 2021

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

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

Symbol:



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.

[7] IBM Quantum. https://quantum-computing.ibm.com/, 2021

SWAP Gate

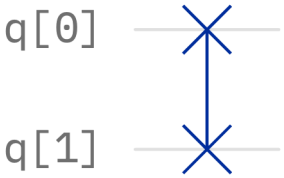
The SWAP Gate is a 2-Qubit Gate and is used to swap the states of two qubits.

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

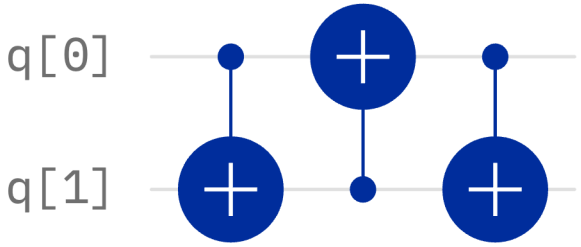
$$\text{SWAP}|q_0 q_1\rangle = |q_1 q_0\rangle$$

The matrix form and the symbol have been shown on the RHS.

Symbol:



The SWAP Gate can also be implemented using 3 CNOT Gates:

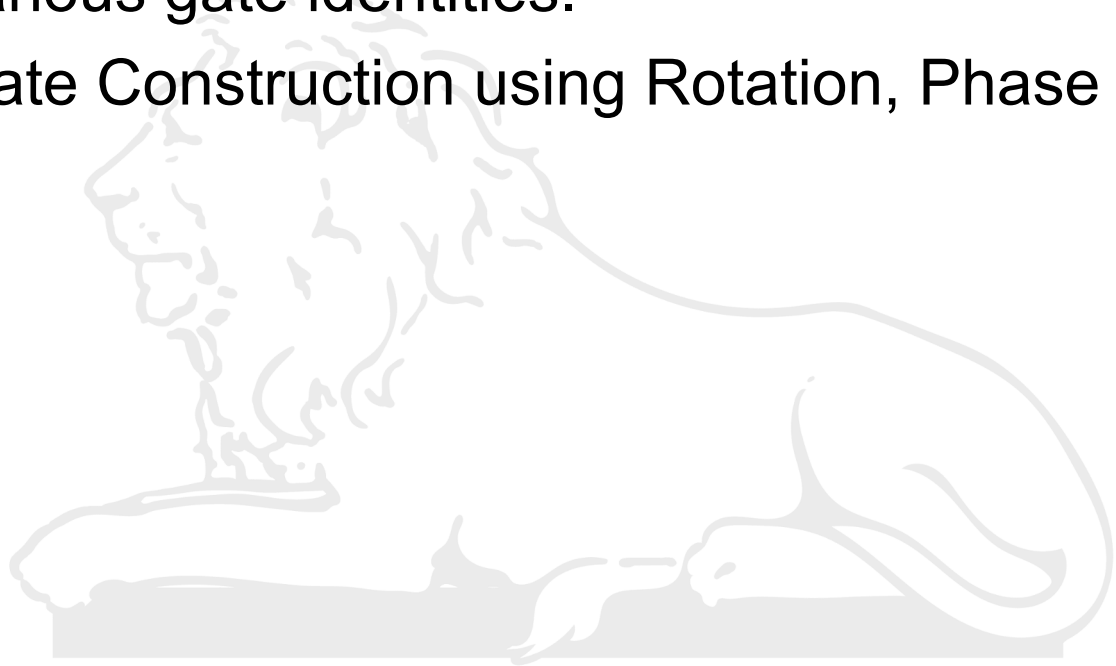


[7] IBM Quantum. https://quantum-computing.ibm.com/, 2021

Simulation of Single and Multiple Qubit Gates using MATLAB

Now, we shall use MATLAB to:

1. Simulate the action of single and multi-qubit gates.
2. Verify various gate identities.
3. Verify Gate Construction using Rotation, Phase and CNOT Gates.



Simulation of Single and Multiple Qubit Gates using 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 esure that
%we can truely understand gate action on arbitary state.

format shortG
```

```
psi =
a
b
```

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>:
     0
     1
Action of X gate on state |1>:
     1
     0
Action of X gate on state a|0> + b|1>:
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>:
            0 +           0i
            0 +           1i
Action of Y gate on state |1>:
            0 -           1i
            0 +           0i
Action of Y gate on state a|0> + b|1>:
-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>:
     1
     0
Action of Z gate on state |1>:
     0
    -1
Action of Z gate on state a|0> + b|1>:
 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>:
       0.70711
       0.70711
Action of H gate on state |1>:
       0.70711
      -0.70711
Action of H gate on state a|0> + b|1>:
(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>:
     1
     0
Action of S gate on state |1>:
            0 +           0i
            0 +           1i
Action of S gate on state a|0> + b|1>:
   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>:
     1
     0
Action of T gate on state |1>:
            0 +           0i
      0.70711 +     0.70711i
Action of T gate on state a|0> + b|1>:
                          a
2^(1/2)*b*(1/2 + 1i/2)
```

Simulation of Single and Multiple Qubit Gates using MATLAB

Verification of basic gate identites

```
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 =
     0     1
     1     0
HXH =
     1     0
     0    -1
(X+Z)/√2=
    0.70711    0.70711
    0.70711   -0.70711
XZ =
     0    -1
     1     0
ZX =
     0     1
    -1     0
iY =
     0     1
    -1     0
```

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>:
     1
     0
     0
     0
Action of CNOT gate on state |01>:
     0
     1
     0
     0
Action of CNOT gate on state |10>:
     0
     0
     0
     1
Action of CNOT gate on state |11>:
     0
     0
     1
     0
Action of CNOT gate on state a|00> + b|01> + c|10> + d|11>:
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>:
     1
     0
     0
     0
Action of CZ gate on state |01>:
     0
     1
     0
     0
Action of CZ gate on state |10>:
     0
     0
     1
     0
Action of CZ gate on state |11>:
     0
     0
     0
    -1
Action of CZ gate on state a|00> + b|01> + c|10> + d|11>:
 a
 b
 c
-d
```

Simulation of Single and Multiple Qubit Gates using MATLAB

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>:
     1
     0
     0
     0
Action of SWAP gate on state |01>:
     0
     0
     1
     0
Action of SWAP gate on state |10>:
     0
     1
     0
     0
Action of SWAP gate on state |11>:
     0
     0
     0
     1
Action of SWAP gate on state a|00> + b|01> + c|10> + d|11>:
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> + b|001> + c|010> + d|011>
+ e|100> + f|101> + g|110> + h|111>
a
b
c
d
e
f
h
g
```

Verification of Gate Construction using Rotation, Phase and CNOT Gates

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];
```

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(π) = ')
disp(Rx(pi))
disp('Here, an additional global phase of -1i is obtained')

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

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


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

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

% Building CZ using H(=Rx(π)Ry(π/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
%interechanged 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(π) =
   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(π/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 = P(pi/2) =
   1.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 1.0000i
T = P(pi/4) =
   1.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.7071 + 0.7071i
CZ = I⊗H * CNOT * I⊗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
```

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.

[6] Williams, C.P. (2011). Quantum Gates. In: Explorations in Quantum Computing. Texts in Computer Science. Springer, London

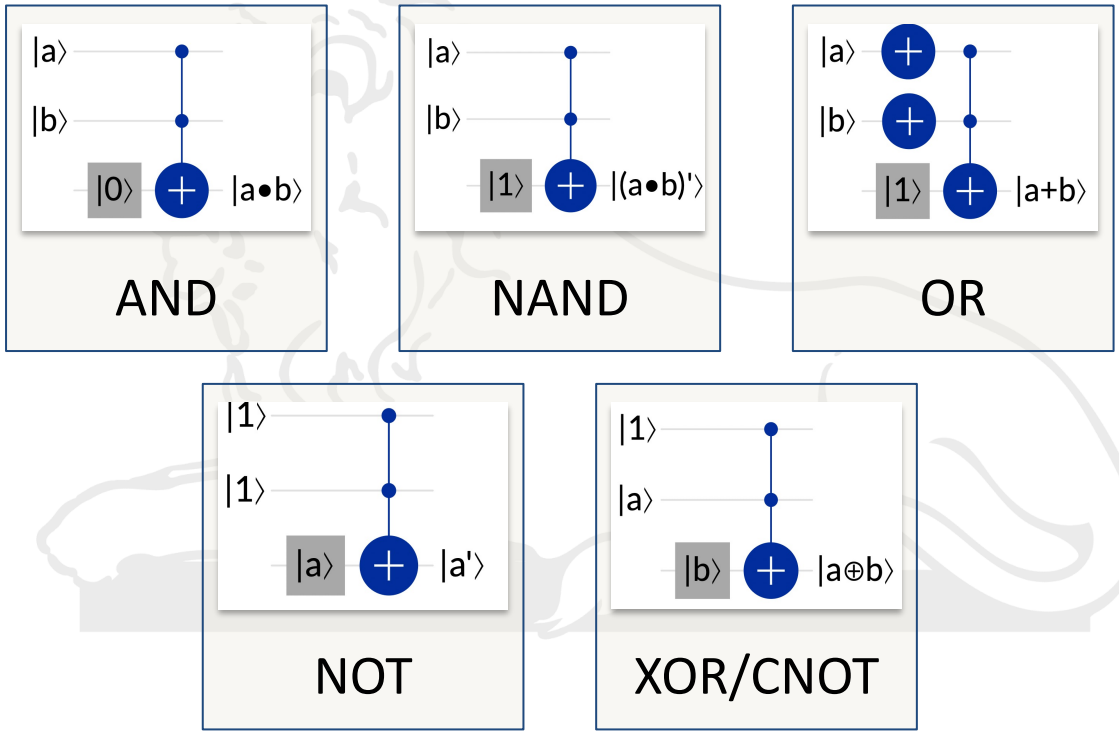
Family 1: Rx(θ), Ry(θ), Rz(θ), P(ϕ), CNOT

- 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.
- 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 before.
- 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 identiy 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.
- We can also implement the Toffoli Gate as well using these gates but it would become too complicated. We shall see a Toffoli Gate Decomposition later on.

[6] Williams, C.P. (2011). Quantum Gates. In: Explorations in Quantum Computing. Texts in Computer Science. Springer, London

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 (in addition to some auxiliary qubits in '0' and '1' states) by itself is self sufficient.

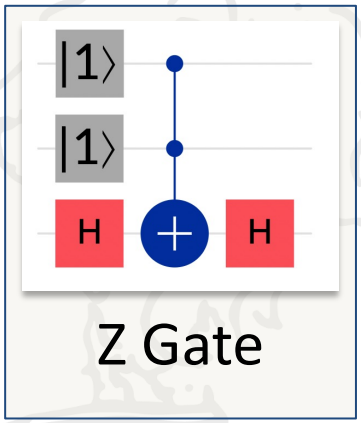


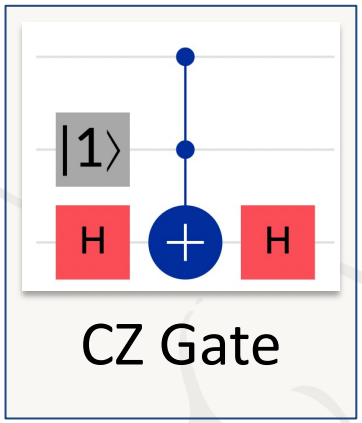
[7] IBM Quantum. https://quantum-computing.ibm.com/, 2021

[6] Williams, C.P. (2011). Quantum Gates. In: Explorations in Quantum Computing. Texts in Computer Science. Springer, London

Family 2: Toffoli (CCX), H

- 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.



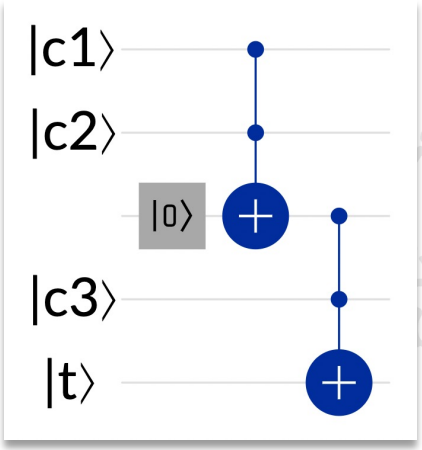


- Hence, this set of gates can be used for a much wider variety of purposes. This however, does not encompass gates such as the S and T gates which are non-reversible fixed-angle gates, which is the main limitation of this set of gates.

[7] IBM Quantum. https://quantum-computing.ibm.com/, 2021
[6] Williams, C.P. (2011). Quantum Gates. In: Explorations in Quantum Computing. Texts in Computer Science. Springer, London

Family 2: Toffoli (CCX), H

- The Toffoli Gate can also be used to implement an n-controlled generalized Toffoli Gate. This requires an additional auxiliary qubit in state $|0\rangle$ for each additional control qubit.



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

[7] IBM Quantum. https://quantum-computing.ibm.com/, 2021
[6] Williams, C.P. (2011). Quantum Gates. In: Explorations in Quantum Computing. Texts in Computer Science. Springer, London

Family 3: CNOT, H, S and T

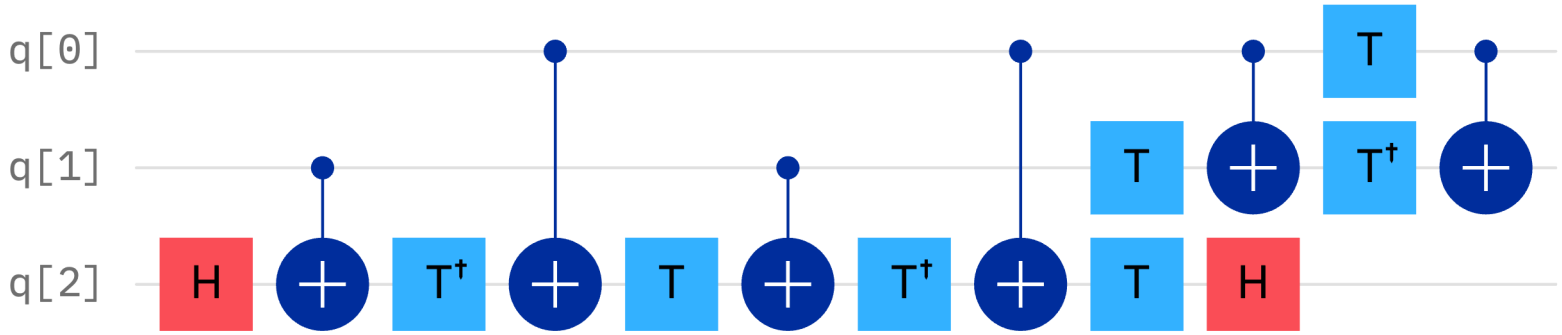
- The set {CNOT, H, S} is referred to as the Clifford Set. This set in itself is not a universal set but the addition of T to the set would allow us to represent any unitary to an approximate level.
- It is an implication of the Solovay–Kitaev theorem that any arbitrary single-qubit gate can be well approximated by the discrete gate set {H, S, T} and an accuracy of ε can be achieved by using $O(\log^c(1/\varepsilon))$ gates.

- The S gate represents a rotation on the Bloch sphere about the Z axis by an angle of π/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.

[12] A. Y. Kitaev, "Quantum Computations: Algorithms and Error Correction," Russ. Math. Surv., Volume 52, Issue 6 (1997) pp. 1191–1249.
[6] Williams, C.P. (2011). Quantum Gates. In: Explorations in Quantum Computing. Texts in Computer Science. Springer, London

Family 3: CNOT, H, S and T

Below is Toffoli Gate Decomposition via the use of H, T and $T^{\dagger}$, where $T^{\dagger}$ refers to the inverse of T and can be written as T^7 or S^3T.



While this is not the most intuitive implementation, this proves that universality can be achieved using this set since the Toffoli Gate itself compasses a large set of operations. We shall attempt to simulate this circuit.

[12] A. Y. Kitaev, "Quantum Computations: Algorithms and Error Correction," Russ. Math. Surv., Volume 52, Issue 6 (1997) pp. 1191–1249.
[7] IBM Quantum. https://quantum-computing.ibm.com/, 2021
[6] Williams, C.P. (2011). Quantum Gates. In: Explorations in Quantum Computing. Texts in Computer Science. Springer, London

DEUTSCH Gate

- 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:

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

- 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 where the Unitary is of the form

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

[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)

[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

[6] Williams, C.P. (2011). Quantum Gates. In: Explorations in Quantum Computing. Texts in Computer Science. Springer, London

DEUTSCH Gate

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

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

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

- 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, θ = π/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(θ) and approximate all known gates.

[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)

[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

[6] Williams, C.P. (2011). Quantum Gates. In: Explorations in Quantum Computing. Texts in Computer Science. Springer, London

BARENCO Gate

- 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.

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

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

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

- This unitary can be much more universal as well. For example, setting θ = π/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 allows you to do a bit-flip with a simultaneous phase change.

[10] Barenco Adriano 1995A universal two-bit gate for quantum computation Proc. R. Soc. Lond. A449679–683
[6] Williams, C.P. (2011). Quantum Gates. In: Explorations in Quantum Computing. Texts in Computer Science. Springer, London

BARENCO Gate

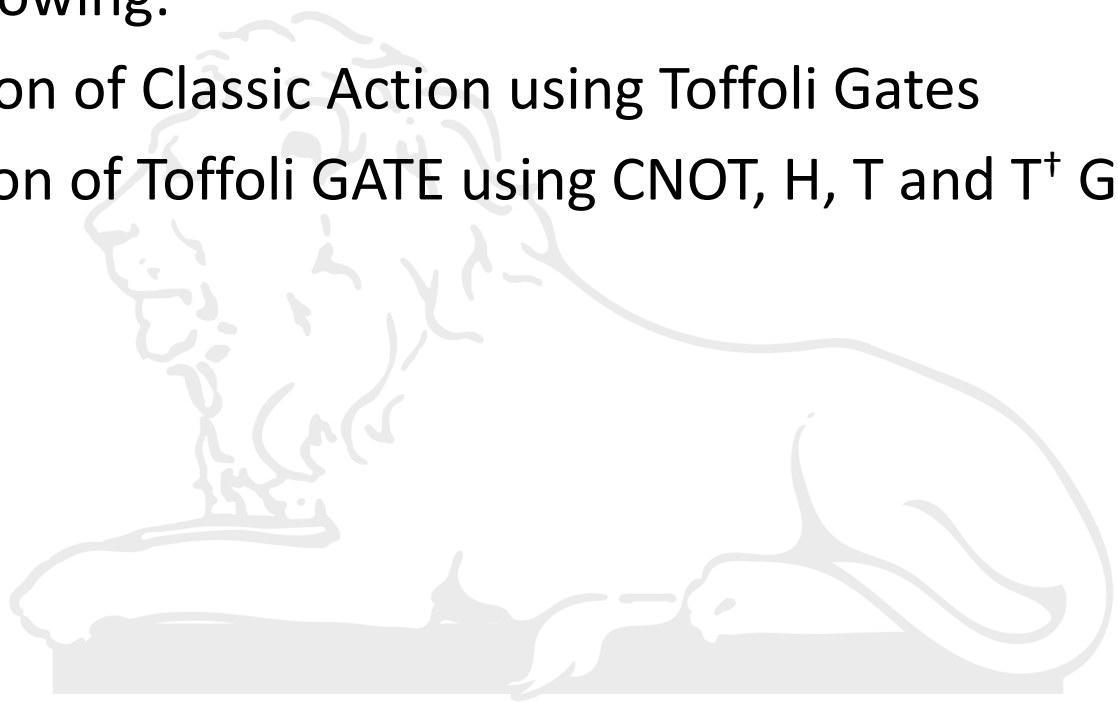
- Hence, this allows us to encompass the sets of CX gates, CY Gates, and if used together with the help of auxiliary qubits, even CZ gates and controlled Phase-Shift gates can be developed, which encompass various universal sets.
- In the case where ϕ, α and θ are chosen fixed irrational multiples of π and each other, we would be able to approximate any unitary just using a single gate. This is similar to the Deutsch Gate.
- It is also known that the BARENCO Gate can be used to build a DEUTSCH Gate.

[10] Barenco Adriano 1995A universal two-bit gate for quantum computation Proc. R. Soc. Lond. A449679–683
[6] Williams, C.P. (2011). Quantum Gates. In: Explorations in Quantum Computing. Texts in Computer Science. Springer, London

Some more simulations on MATLAB

I have run some MATLAB Scripts related to some of these Universal Sets. I did not do a simulation of all the universal sets, but I have done the following:

1. Verification of Classic Action using Toffoli Gates
2. Generation of Toffoli GATE using CNOT, H, T and $T^{\dagger}$ Gates



Verification of Classic Action using Toffoli Gates

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.

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

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.
```

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/LogicGates.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.

Thank You