Quantum Coupling: The Implementation of Two Qubit Gate with Josephson Junction

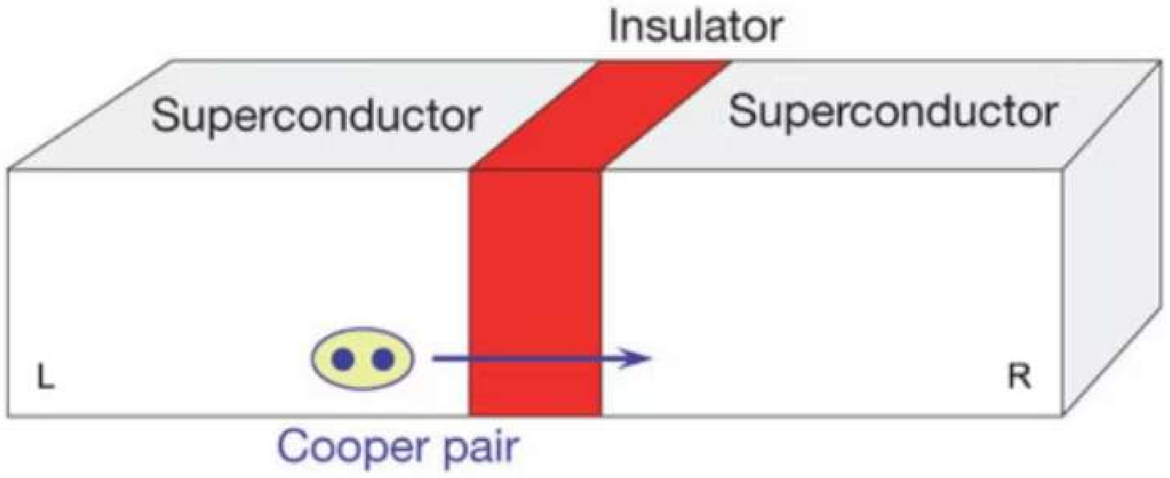
Nguyen Xuan Tung Mat. 2005491 xuantung.nguyen@studenti.unipd.it

Quantum Information and Computing
(a.y. 2022/23)

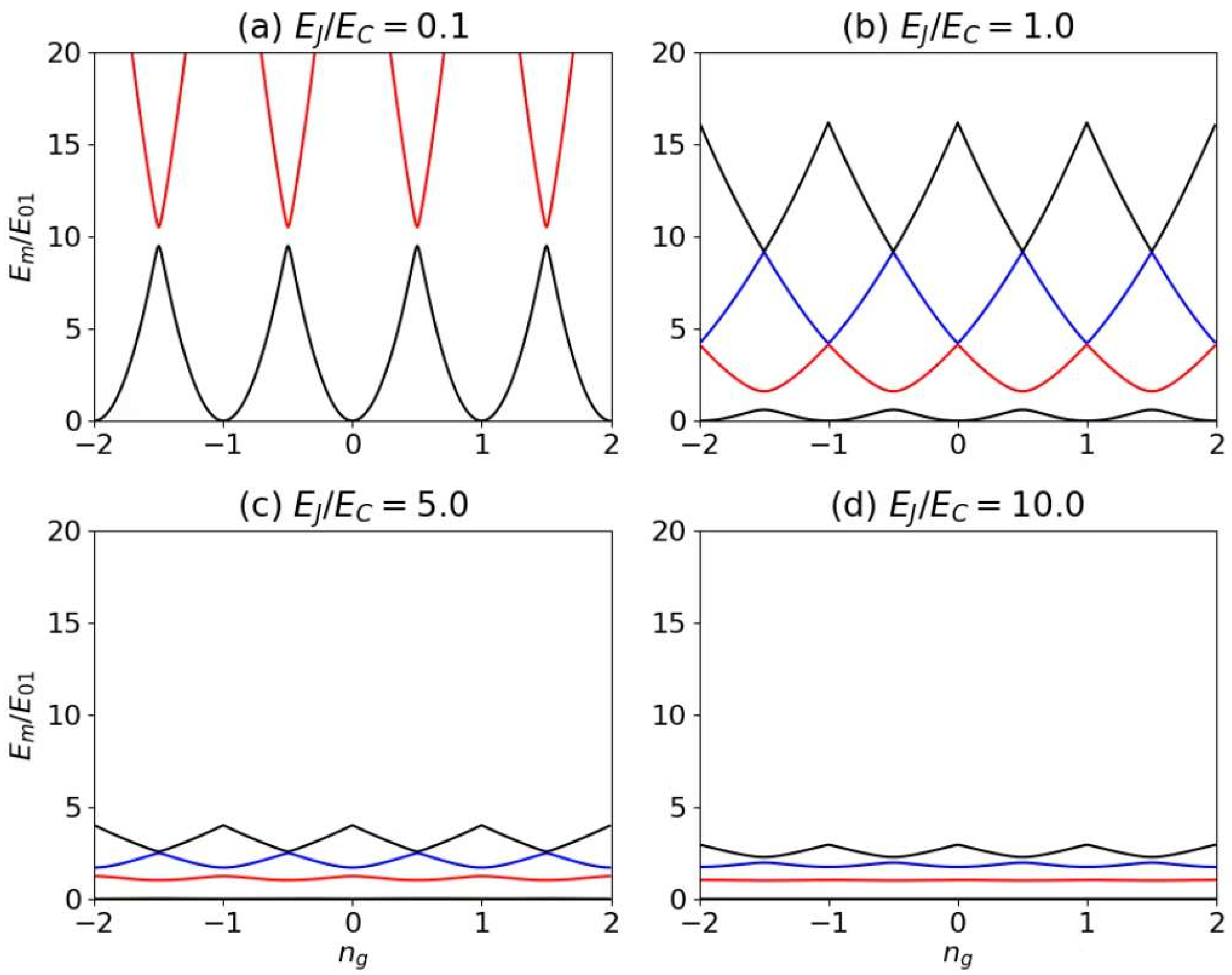
1. Introduction
2. Josephson Junctions
3. Two Qubit Gate
4. Implementation
5. Result
6. Conclusion

Introduction

- Two superconducting electrodes are separated by a thin insulating barrier. When a voltage is applied across the junction, Cooper pairs of electrons tunnel through the barrier, leading to a super current.
- The Josephson effect describes the relationship between the voltage across the junction and the phase difference between the superconducting electrodes, is responsible for the nonlinear behavior of the junction.
- By applying external magnetic fields or controlling the bias voltage across the junction, the Josephson junction can be used to implement single-qubit gates, such as rotations around the Z-axis or the X-axis, and two-qubit gates, such as the controlled-X.



Josephson Junctions: Eigenenergies



Two Qubit Gates

- The CNOT gate, a quantum gate, acts on two qubits by flipping the second qubit only if the first qubit is in state $|1>$. A notable outcome is entanglement, evident when it's used on the separable state $\alpha|00> + \beta|10>$.
- This generates $\alpha|00> + \beta|11>$, which is entangled. Josephson junctions are often used in a superconducting quantum computer to implement CNOT gates.
- These are made by two superconducting electrodes separated by a thin insulating barrier; the flow of super currents through the junction is initiated when a voltage is applied.
- To use a CNOT gate, two qubits are required, each typically encoded in superconducting loops. One of the qubits is a control qubit, and the other one is a target qubit.

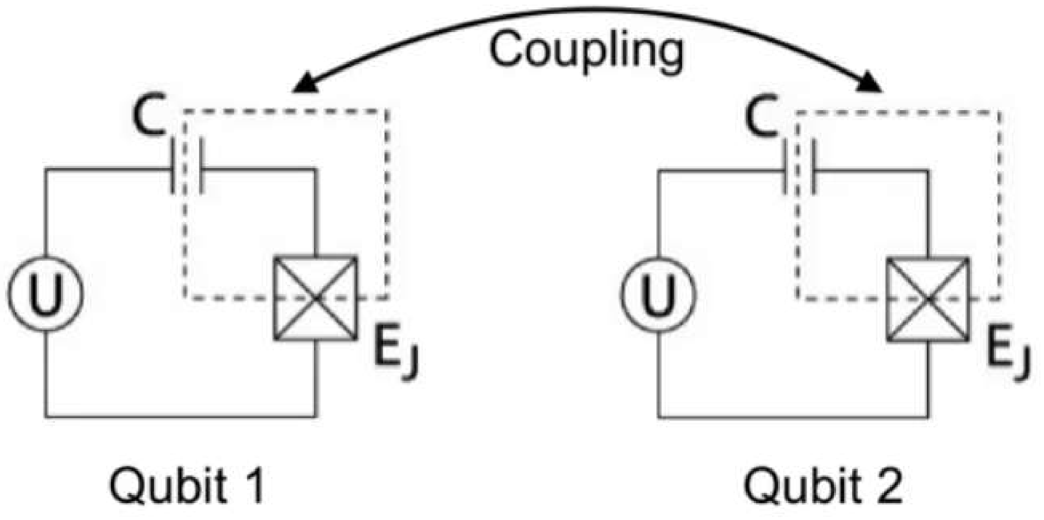
Implementation of CNOT gate

- The Hamiltonian describes the energy of the system and how it evolves over time, and can be used to simulate the behavior of the qubits during the gate operation.
- In order to implement Coupled Josephson qubits, we start with Hamiltonian function.

$$H = H_1 + H_2 + H_{coupling}$$

$$H_i = \sum_{n_i} [4E_c(n_i - n_g^{(i)})^2 \sigma_z^{(i)} - \frac{E_J^{(i)}}{2} \sigma_x^{(i)}]$$

$$H_{coupling} = E_{cc} \sigma_z^{(1)} \sigma_z^{(2)}$$



Implementation of CNOT gate

- The two-qubit time-evolution operator is defined similarly as:

$$U = exp(-\frac{it}{\hbar}H)$$

- CNOT gate can be implemented:

$$CNOT \propto H^2[U_z^{(1)}(-\frac{\pi}{2})U_z^{(2)}(-\frac{\pi}{2})exp(i\frac{\pi}{4}\sigma_z^{(i)})]H^2$$

- In order to quantify the accuracy of the gate, we need to compute the fidelity:

$$F = |<\Psi_1|U^+U|\Psi_2>|^2$$

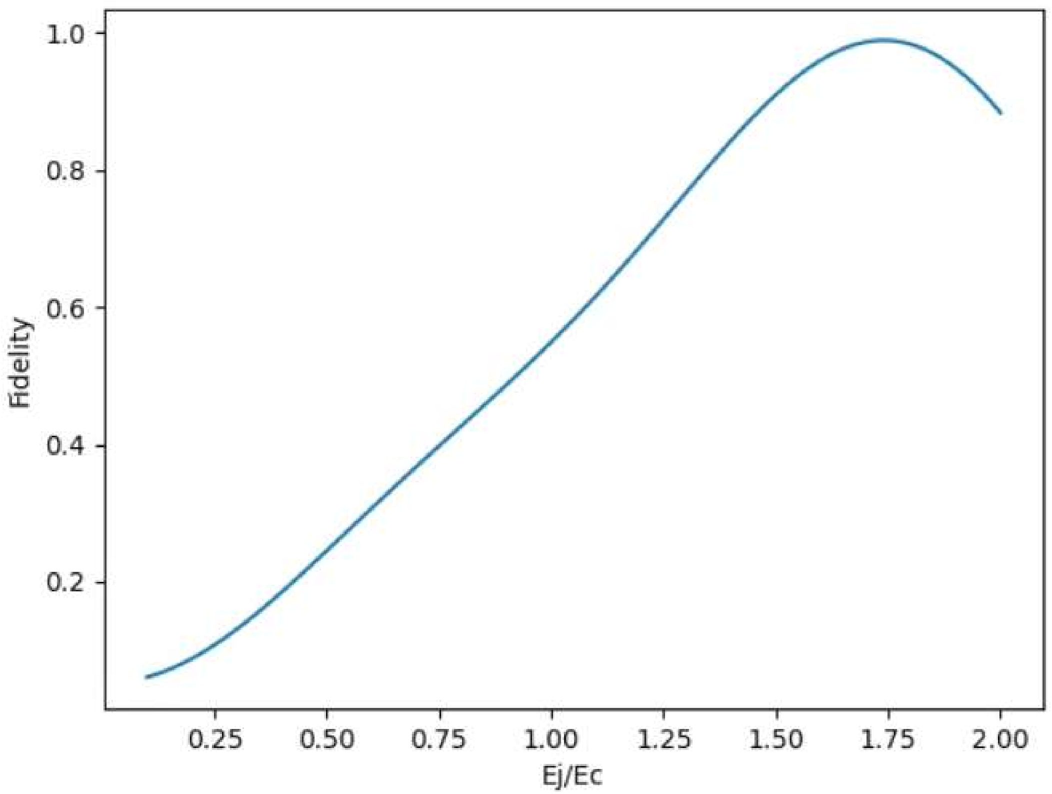
Implementation of CNOT gate

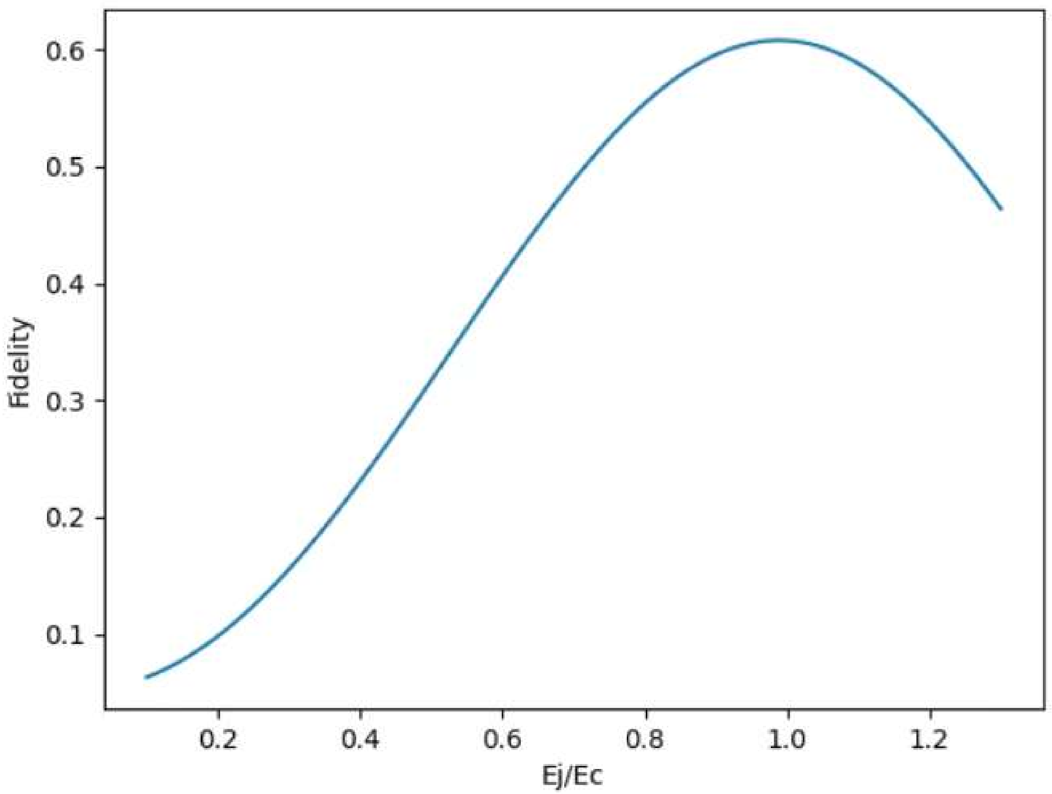
```python
# Define parameters for Hamiltonian
Ej1 = 0.005
Ej2 = 0.005
ng = 0
Ec1 = 1
Ec2 = 1

# Define Pauli matrices
sigmax = np.array([[0, 1], [1, 0]])
sigmay = np.array([[0, -1j], [1j, 0]])
sigmaz = np.array([[1, 0], [0, -1]])
Had = np.sqrt(0.5)*np.array([[1, 1], [1, -1]])

# Define ideal CNOT gate matrix
CNOT_ideal = np.array([[1, 0, 0, 0],
                       [0, 1, 0, 0],
                       [0, 0, 0, 1],
                       [0, 0, 1, 0]])

# Definetwo-qubits Hamiltonian
#t: time dependence of the function
#Y: function variable
def Hamil_noisy(t,Y,Ej1,Ec1,Ej2,Ec2,ng,Ecc,Beta1,Beta2):

    H = -0.5*(Ej1)*np.matmul(np.kron(sigmax,np.identity(2)), Y) +
    (- 0.5*(Ec1*(1-2*ng))+Beta1)*np.matmul(np.kron(sigmaz,np.identity(2)), Y) -
    0.5*(Ej2)*np.matmul(np.kron(np.identity(2),sigmax), Y) +
    (- 0.5*(Ec2*(1-2*ng))+Beta2)*np.matmul(np.kron(np.identity(2),sigmaz), Y) +Ecc*np.matmul(np.kron(sigmaz,sigmaz), Y)
    return H

# Define utility function to be used in time evolution described by Schroedinger equation
def H2Q_noisy_divI(t,Y,Ej1,Ec1,Ej2,Ec2,ng,Ecc,Beta1,Beta2):


    return Hamil_noisy(t,Y,Ej1,Ec1,Ej2,Ec2,ng,Ecc,Beta1,Beta2)/(0+1.0j)

# Define Unperturbed two-qubits Hamiltonian
def Hamil(t,Y,Ej1,Ec1,Ej2,Ec2,ng,Ecc):

    return Hamil_noisy(t,Y,Ej1,Ec1,Ej2,Ec2,ng,Ecc,0,0)

# Define utility function to be used in time evolution described by Schroedinger equation
def H2Q_divI(t,Y,Ej1,Ec1,Ej2,Ec2,ng,Ecc):

    return H2Q_noisy_divI(t,Y,Ej1,Ec1,Ej2,Ec2,ng,Ecc,0,0)
```

```python
# apply Hadamard gate
ket00_t = np.matmul(np.kron(np.identity(2),Had), ket00_t)
ket01_t = np.matmul(np.kron(np.identity(2),Had), ket01_t)
ket10_t = np.matmul(np.kron(np.identity(2),Had), ket10_t)
ket11_t = np.matmul(np.kron(np.identity(2),Had), ket11_t)



# apply exp(i pi/4 sigmazDOTsigmaz) Where Bx = 0 = Bz, E_CC ≠ 0, tau = pi / (4 |E_CC|) if E_CC<0

sol = integrate.solve_ivp(H2Q_divI, t_span=(0,tau1), y0=ket00_t, args=(0,0,0,0,0,ECC), method='DOP853')
ket00_t = sol.y[:,-1]

sol = integrate.solve_ivp(H2Q_divI, t_span=(0,tau1), y0=ket01_t, args=(0,0,0,0,0,ECC), method='DOP853')
ket01_t = sol.y[:,-1]

sol = integrate.solve_ivp(H2Q_divI, t_span=(0,tau1), y0=ket10_t, args=(0,0,0,0,0,ECC), method='DOP853')
ket10_t = sol.y[:,-1]

sol = integrate.solve_ivp(H2Q_divI, t_span=(0,tau1), y0=ket11_t, args=(0,0,0,0,0,ECC), method='DOP853')
ket11_t = sol.y[:,-1]



# apply Uz Where: Bx = 0 = E_CC, Bz ≠ 0, tau  = pi / (2 |Bz|) if Bz<0

sol = integrate.solve_ivp(H2Q_divI, t_span=(tau1,tau1+tau2), y0=ket00_t, args=(0,BZ1,0,BZ2,0,0), method='DOP853')
ket00_t = sol.y[:,-1]

sol = integrate.solve_ivp(H2Q_divI, t_span=(tau1,tau1+tau2), y0=ket01_t, args=(0,BZ1,0,BZ2,0,0), method='DOP853')
ket01_t = sol.y[:,-1]

sol = integrate.solve_ivp(H2Q_divI, t_span=(tau1,tau1+tau2), y0=ket10_t, args=(0,BZ1,0,BZ2,0,0), method='DOP853')
ket10_t = sol.y[:,-1]

sol = integrate.solve_ivp(H2Q_divI, t_span=(tau1,tau1+tau2), y0=ket11_t, args=(0,BZ1,0,BZ2,0,0), method='DOP853')
ket11_t = sol.y[:,-1]



# apply Hadamard gate
ket00_t = np.matmul(np.kron(np.identity(2),Had), ket00_t)
ket01_t = np.matmul(np.kron(np.identity(2),Had), ket01_t)
ket10_t = np.matmul(np.kron(np.identity(2),Had), ket10_t)
ket11_t = np.matmul(np.kron(np.identity(2),Had), ket11_t)
```

Results

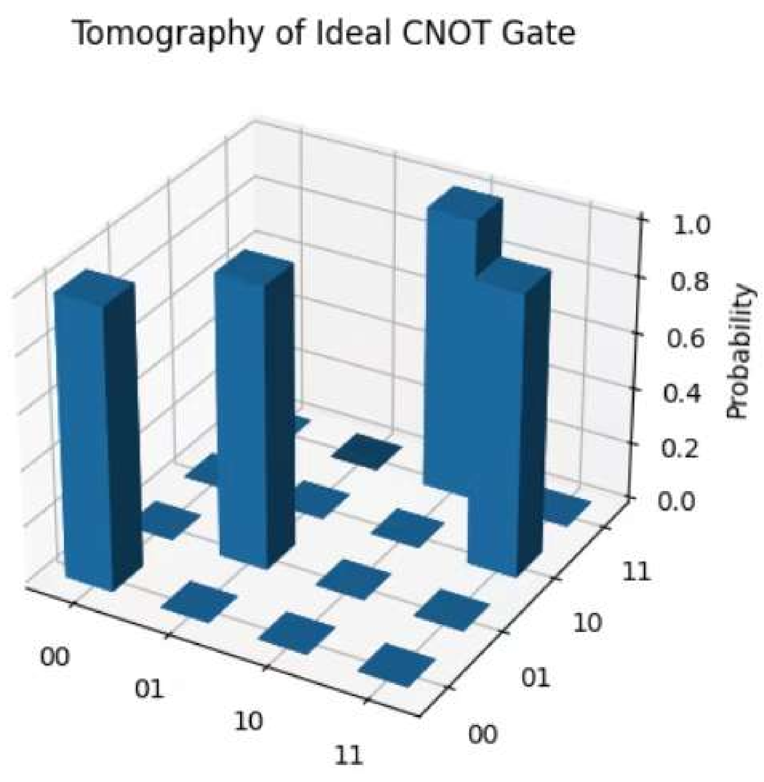
Fidelity of the ideal and noisy CNOT gate with the fraction of EJ/EC respectively.

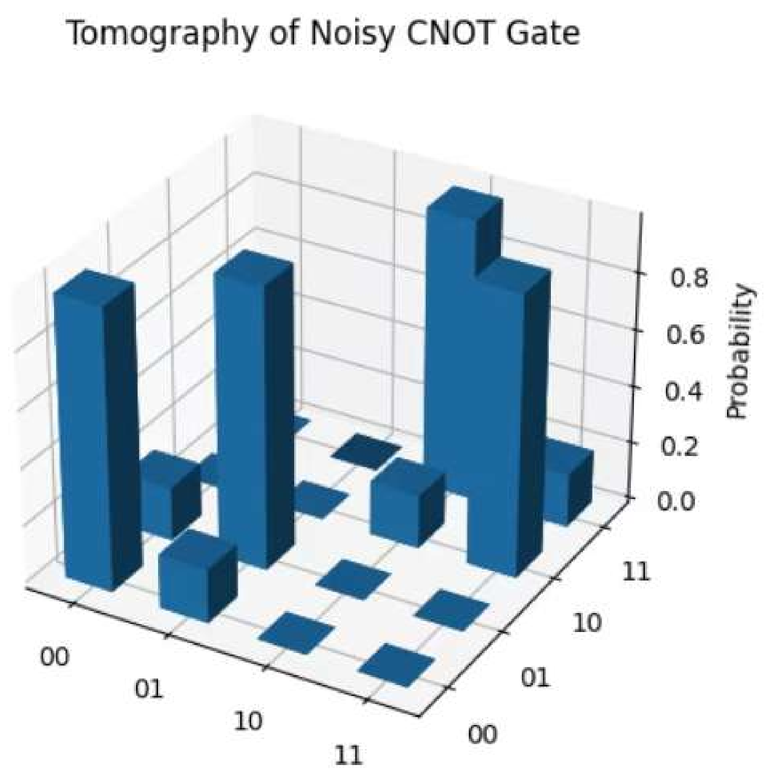




Results

3D histogram of the probabilities of the ideal and perturbed CNOT gate acting on each of the four possible input states |00 >, |01 >, |10 >, and|11 >)





Conclusions

- Successfully implement the 2 qubit gate coupled with Josephson junction.
- Josephson junctions can be used as qubits in quantum computation.
- Find the fidelity of the gate with respect to Josephson energy.
- The charge qubit setup allows to implement the CNOT gate.

Backup slides

Backup

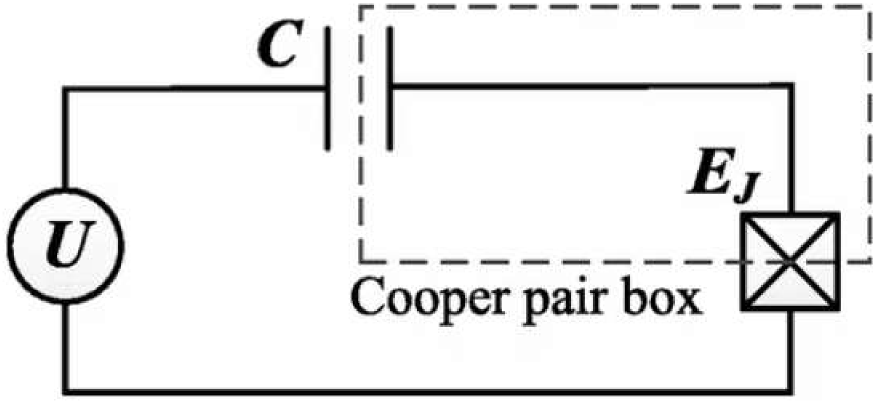


Figure: Super conducting charged qubit

$$H_1 = [4E_c(n - n_g^2\sigma_z^{(i)} - E_J cos\theta]$$

- Where n: number operator of excess Cooper-pair charges
- Where θ: phase of the superconducting order parameter

$$E_c = \frac{e^2}{2(C + C_j)}$$

$$n_g = \frac{CU}{2e}$$

$$E_J = \frac{\hbar}{2e} I_c$$

- Where I_c = critical current of the junction

Backup

$$H_1 = [4E_c(n - n_g^2\sigma_z^{(i)} - E_J cos\theta]$$

$$C_j < 10^{-15}F$$

$$C \ll C_j$$

$$\frac{E_j}{k_b} \sim 100mK$$

Superconducting with gap energy:

$$\Delta \gg E_c$$

Hamiltonian can be simplified:

$$H = -\frac{1}{2}B_z\sigma_z - \frac{1}{2}B_x\sigma_x$$

Where: $n_g \in [0, 1]$
$B_z = 4E_c(n - n_g^2$
$B_x = E_j$

Backup

$$CNOT \propto H^2[U_z^{(1)}(-\frac{\pi}{2})U_z^{(2)}(-\frac{\pi}{2})exp(i\frac{\pi}{4}\sigma_z^{(i)})]H^2$$

$$H = -\frac{1}{2}B_z\sigma_z - \frac{1}{2}B_x\sigma_x$$

for $\tau_2 = \frac{\hbar\pi}{2(-B_z^1)}$

$$H = E_{cc}\sigma_z^1\sigma_z^2$$

for $\tau_1 = \frac{\hbar\pi}{4(-E_{cc})}$