Quantum Computing

Chapter 05: Leveraging Superposition, Entanglement, and Error Correction

Prachya Boonkwan

Version: January 14, 2026

Sirindhorn International Institute of Technology
Thammasat University, Thailand

Who? Me?

- Nickname: Arm (P'/N' Arm, etc.)
- Born: Aug 1981
- Work
  - Researcher at NECTEC 2005-2024
  - Lecturer at SIIT, Thammasat University 2025-now
- Education
  - B.Eng & M.Eng in Computer Engineering, Kasetsart University, Thailand
  - Obtained Ministry of Science and Technology Scholarship of Thailand in early 2008
  - Did a PhD in Informatics (AI & Computational Linguistics) at University of Edinburgh, UK from 2008 to 2013

Table of Contents

1. Bernstein-Vazirani Algorithm

2. Deutsch-Jozsa Algorithm

3. Quantum Teleportation

4. Superdense Coding

5. Quantum Noisy Theorem

6. Quantum Error Correction

7. Conclusion

Bernstein-Vazirani Algorithm

Oracle Separation

- Suppose there is a combination padlock, where you have to key in a code to unlock it
- Objective: We want to find the right combination code, called the oracle, of this padlock
- In classical computing, we have to try all possible combination codes in the worst case
- Ethan Bernstein and Umesh Vazirani introduced a quantum algorithm for oracle separation (1997) that solves this problem in $O(1)$ time complexity

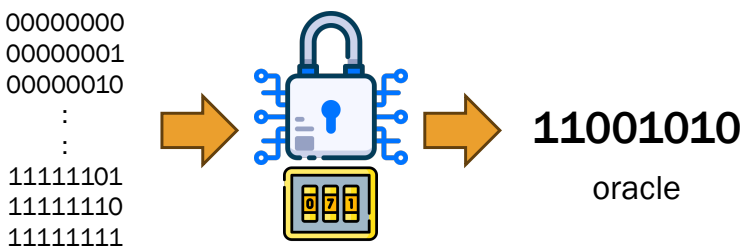


Figure 1: Oracle separation problem

Quantum Parallelism

- In quantum computing, we enable parallelism by:
  1. Converting a bit string into a sequence of binary distributions
  2. Manipulating these distributions with complex matrix operations
  3. Sharing data structures via quantum entanglement and lazy evaluation
- Quantum state for binary distribution

$$|\psi\rangle = \alpha^{(0)}|0\rangle + \alpha^{(1)}|1\rangle$$

where α, β are complex coefficients, and

$$P(0|\psi) = \left|\alpha^{(0)}\right|^2 \quad P(1|\psi) = \left|\alpha^{(1)}\right|^2$$

- We can represent a set of all possible N-bit input strings with a mixed state

$$\begin{aligned} |\psi\rangle &= \frac{1}{\sqrt{2^N}} \sum_{j=0}^{2^N-1} \beta_j |\mathbb{B}_N(j)\rangle \\ &= |+\rangle^{\otimes N} \end{aligned}$$

where each $\mathbb{B}_N(j)$ is the N-bit binary string of decimal number j, and $\beta_j = \prod_{k=1}^{N} \alpha_j^{(x_k)}$ is its coefficient

- The probability for each binary string $\mathbb{B}_N(j) = |x_1, \ldots, x_N\rangle$ is

$$P(x_1, \ldots, x_N|\psi) = \left|\beta_j\right|^2$$

Bernstein-Vazirani Problem (BV Problem)

- Suppose we have a black-box function f that matches an input binary with a secret string $s_{1:N}$ (i.e. oracle) [% is modulo]

$$\begin{aligned} f_s(x_{1:N}) &= x_{1:N} \cdot s_{1:N} \\ &= \left(\sum_{j=1}^{N} x_j s_j \right) \%2 \end{aligned}$$

N.B. $f_s(x_{1:N})$ returns the number of matched 1's modulo by 2

- Example: Let oracle $s = 110$

$$\begin{aligned} f_s(100) &= \left[(1 \cdot 1) + (0 \cdot 1) + (0 \cdot 0) \right] \%2 \\ &= 1 \end{aligned}$$

- We can recover the oracle s from the function f_s with bitwise measurement

$$s_k = f_s(\mathbb{B}_N(2^k))$$

where $\mathbb{B}_N(j)$ is the N-bit binary of decimal number j — e.g. $\mathbb{B}_6(15) = 001111$

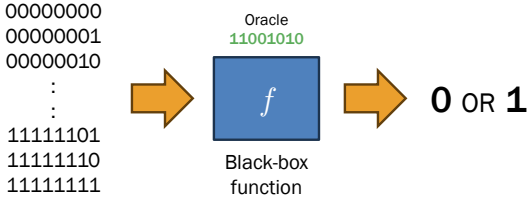


Figure 2: Bernstein-Vazirani problem

Bernstein-Vazirani Problem

- This problem is a prime example of the complexity class BQP (bounded-error quantum polynomial problems)
- We can leverage the quantum parallelism to separate the oracle $s_{1:N}$ from the function f

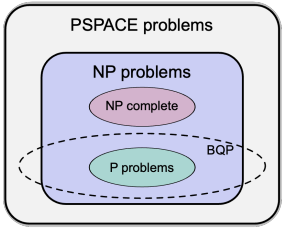


Figure 3: Problem space

- Let's define a quantum operator $\mathbf{U}_{fs}$ in response to the function f

$$
\begin{aligned}
\mathbf{U}_{fs} &= |x_{1:N}\rangle\langle x_{1:N}| \otimes |y \otimes f_s(x_{1:N})\rangle\langle y| \\
&= |x_{1:N}\rangle\langle x_{1:N}| \otimes \left(|0 \otimes f_s(x_{1:N})\rangle\langle 0| \right. \\
&\quad \left. + |1 \otimes f_s(x_{1:N})\rangle\langle 1|\right) \\
&= \mathrm{I}_N \otimes \begin{bmatrix} 0 \oplus f_s(x_{1:N}) & 0 \\ 0 & 1 \oplus f_s(x_{1:N}) \end{bmatrix}
\end{aligned}
$$

The ancillary qubit y works as a carry flag in the phase kickback technique

Phase Kickback Technique

- In the phase kickback technique, a phase applied on the ancillary qubit gets kicked-back (i.e. taking a repercussion) due to interference
- Suppose we have an operator $\mathbf{U}$ that applies a phase onto the input qubit $|\psi\rangle$

$$\mathbf{U}|\psi\rangle = \operatorname{cis}\theta|\psi\rangle$$

- Let's define a multiple-control operator $\mathbf{C}^N\mathbf{U}$, where $|x_{1:N}\rangle \neq |1\rangle^{\otimes N}$ and

$$\begin{aligned}\mathbf{C}^N\mathbf{U} &= |x_{1:N}, y\rangle\langle x_{1:N}, y| \\ &\quad + \left(|1\rangle^{\otimes N} \otimes \operatorname{cis}\theta|y\rangle\right)\left(\langle 1|^{\otimes N} \otimes \langle y|\right)\end{aligned}$$

- Note that the phase is kicked-back to $|1\rangle^{\otimes N}$ when $|x_{1:N}\rangle$ is superposition

$$\begin{aligned}&\mathbf{C}^N\mathbf{U}\left(|+\rangle^{\otimes N} \otimes |\psi\rangle\right) \\ &= \frac{1}{2^{N/2}}\sum_{j=0}^{2^N-1}|\mathbb{B}_N(j)\rangle \otimes |\psi\rangle \\ &= \frac{1}{2^{(N+1)/2}}\left(\sum_{j=0}^{2^N-2}|\mathbb{B}_N(j)\rangle \otimes |\psi\rangle\right) \\ &\quad + \frac{1}{2^{(N+1)/2}}\left(|1\rangle^{\otimes N} \otimes \operatorname{cis}\theta|\psi\rangle\right) \\ &= \frac{\left(\sum_{j=0}^{2^N-2}|\mathbb{B}_N(j)\rangle\right) + \operatorname{cis}\theta|1\rangle^{\otimes N}}{2^{(N+1)/2}} \otimes |\psi\rangle\end{aligned}$$

Phase Kickback Technique

- Instead of a single oracle $|1\rangle^{\otimes N}$, let's generalize $\mathbf{C}^N\mathbf{U}$ by a function f with an oracle $|s_{1:N}\rangle$

$$\begin{aligned}\mathbf{U}_f &= |x_{1:N}, y\rangle\langle x_{1:N}, y| \\ &\quad + |s_{1:N}, y\,\mathrm{cis}\,\theta\rangle\langle s_{1:N}, y| \\ &= |x_{1:N}, y\rangle\langle x_{1:N}, y| \\ &\quad + \mathrm{cis}\,\theta\,|s_{1:N}, y\rangle\langle s_{1:N}, y|\end{aligned}$$

  where all $x_{1:N} \neq s_{1:N}$

- Let $f_{sk}(x_{1:N})$ be a probing function of each k-th bit of the oracle s

$$f_{sk}(x_{1:N}) = x_{1:N} \cdot \mathbb{B}_N(s_k 2^k)$$

- Kickback property:

$$\begin{aligned}&\mathbf{U}_f\left(|+\rangle^{\otimes N} \otimes |y\rangle\right) \\ =\ &\mathbf{U}_f\left(\sum_{j=0}^{2^N-1} |\mathbb{B}_N(j), y\rangle\right) \\ =\ &\sum_{k=1}^{N}\sum_{j=0}^{2^N-1} (\mathrm{cis}\,\theta)^{\mathbb{B}_N(j)\cdot\mathbb{B}_N(2^k)} |\mathbb{B}_N(j), y\rangle \\ =\ &\sum_{k=1}^{N}\sum_{j=0}^{2^N-1} (\mathrm{cis}\,\theta)^{f_{sk}(\mathbb{B}_N(j))} |\mathbb{B}_N(j), y\rangle\end{aligned}$$

- Note that each k-th matched bit get kicked-back by $\mathrm{cis}\,\theta$

Generalized Phase Kickback Technique

- For an oracle function f with an oracle $|s_{1:N}\rangle$, we can construct its kickback operator of f as follows

$$\begin{aligned}
\mathbf{U}_f &= |x_{1:N}, y \oplus f(x_{1:N})\rangle \langle x_{1:N}, y| \\
&= |x_{1:N}, 0 \oplus f(x_{1:N})\rangle \langle x_{1:N}, 0| \\
&\quad + |x_{1:N}, 1 \oplus f(x_{1:N})\rangle \langle x_{1:N}, 1| \\
&= \mathbf{I}_N \otimes \begin{bmatrix} 0 \oplus f(x_{1:N}) & 0 \\ 0 & 1 \oplus f(x_{1:N}) \end{bmatrix}
\end{aligned}$$

- Recall that each k-th bit of the oracle can be recovered by

$$s_k = f_{sk}(2^k)$$

$$\begin{aligned}
&\mathbf{U}_f \left(|+\rangle^{\otimes N} \otimes |-\rangle \right) \\
&= \frac{1}{\sqrt{2^N}} \sum_{j=0}^{2^N-1} \mathbf{U}_f \left(|\mathbb{B}_N(j)\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) \\
&= \frac{1}{\sqrt{2^{N+1}}} \sum_{j=0}^{2^N-1} \sum_{k=1}^{N} |\mathbb{B}_N(j)\rangle \otimes |0 \oplus f_{sk}(j)\rangle \\
&\quad - \frac{1}{\sqrt{2^{N+1}}} \sum_{j=0}^{2^N-1} \sum_{k=1}^{N} |\mathbb{B}_N(j)\rangle \otimes |1 \oplus f_{sk}(j)\rangle \\
&= \left(\bigotimes_{k=1}^{N} \frac{|0\rangle + (-1)^{f_{sk}(2^k)} |1\rangle}{\sqrt{2}} \right) \otimes |-\rangle \\
&= \mathbf{H}|s_1\rangle \otimes \ldots \otimes \mathbf{H}|s_N\rangle \otimes \mathbf{H}|1\rangle
\end{aligned}$$

Implementation of Oracle Operator

- The oracle operator $\mathbf{U}_f$ can be implemented as a series of **CCX** for each bit of $|s_{1:N}\rangle$

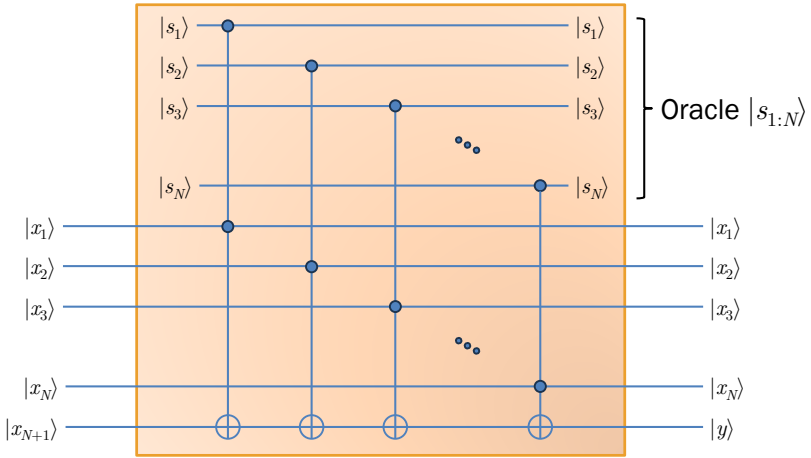


Figure 4: Oracle operator $\mathbf{U}_f$

Bernstein-Vazirani Algorithm (BV Algorithm)

- If we have an oracle function f whose oracle is $|s_{1:N}\rangle$, we construct its kickback operator

$$\mathbf{U}_f \;=\; \mathbf{I}_N \otimes \begin{bmatrix} 0 \oplus f(x_{1:N}) & 0 \\ 0 & 1 \oplus f(x_{1:N}) \end{bmatrix}$$

- The Bernstein-Vazirani algorithm for f is as follows

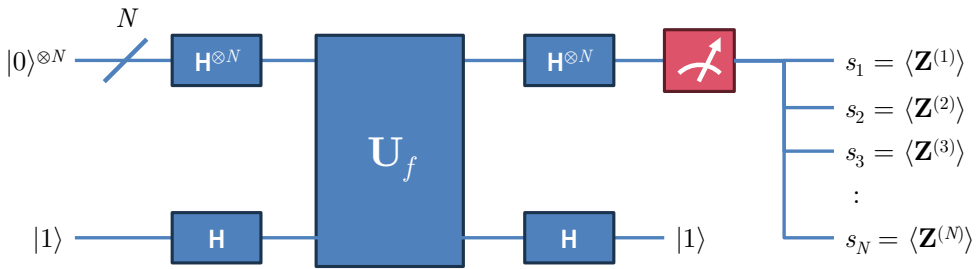


Figure 5: Bernstein-Vazirani algorithm

Summary

- BV Algorithm demonstrates the efficiency of exponential queries via superposition
- One multi-qubit query $|+\rangle^N$ in quantum computing represents 2^N binary strings in classical computing

$$|+\rangle^N = \frac{1}{\sqrt{2^N}} \sum_{j=0}^{2^N-1} |\mathbb{B}_N(j)\rangle$$

- State superposition allows parallel evaluation of independent queries

$$\mathbf{U}|+\rangle^N = \sum_{j=0}^{2^N-1} \mathbf{U}|\mathbb{B}_N(j)\rangle$$

- For any blackbox function f guaranteed to have a single oracle, we construct

$$\mathbf{U}_f = \mathbf{I}_N \otimes \begin{bmatrix} 0 \oplus f(x_{1:N}) & 0 \\ 0 & 1 \oplus f(x_{1:N}) \end{bmatrix}$$

  The last qubit is dedicated for the phase kickback technique
- BV Algorithm separates the oracle $|s_{1:N}\rangle$ from an unseen quantum operator $\mathbf{U}_f$ using quantum parallelism and the phase kickback technique

$$|s_{1:N}\rangle \otimes |1\rangle = \mathbf{H}\mathbf{U}_f\mathbf{H}\left(|0\rangle^{\otimes N} \otimes |1\rangle\right)$$

Bernstein-Vazirani Algorithm in PennyLane

```
secret_code = [0,1,0,0,1,1,0,1]
no_bits = len(secret_code)
dev = qml.device('default.qubit', wires=no_bits + 1, shots=500)

def oracle(secret_code, wires):
    for i in range(len(secret_code)):              # CCX for each bit of the oracle
        if secret_code[i] == 1:
            qml.CNOT(wires=[wires[i], wires[-1]])

@qml.qnode(dev)
def bernstein_vazirani(secret_code):
    qml.X(wires=no_bits)                           # Prepare state |+>^N (*) |-> from |0>^(N+1)
    qml.Hadamard(wires=no_bits)
    for i in range(no_bits):
        qml.Hadamard(wires=[i])
    oracle(secret_code, range(no_bits + 1))        # Apply the oracle operator U_f
    for i in range(no_bits):                       # Extract the oracle from qubits[1:N]
        qml.Hadamard(wires=[i])
    return qml.counts(wires=range(no_bits))        # Return the oracle

bernstein_vazirani(secret_code)
# Expected result: {'01001101': 500}
```

Exercise 5.1: Bernstein-Vazirani Algorithm

Answer the following questions.

1. Suggest how to modify the oracle operator $\mathbf{U}_f$ to possess multiple oracles.
2. Show that

$$\mathbf{H}^{\otimes N} = \frac{1}{\sqrt{2^N}} \sum_{j=0}^{2^N-1} |\mathbb{B}_N(j)\rangle$$

3. Show that for any oracle s of length N,

$$s = \sum_{k=0}^{N-1} \left[2^k f_s(\mathbb{B}_N(2^k))\right]$$

where $f_s(x_{1:N}) = x_{1:N} \cdot s_{1:N}$.

4. Regarding the implementation of the oracle operator $\mathbf{U}_f$, show that

$$\mathrm{CCX}^{(N+1)}_{s_k,x_k,x_{N+1}} \left[\left| \mathbb{B}_N(2^k) \right\rangle \otimes |1\rangle \right]$$
$$= \mathbf{I}_N \otimes \begin{bmatrix} 0 \oplus x_k s_k & 0 \\ 0 & 1 \oplus x_k s_k \end{bmatrix}$$

5. In connection with the previous question, show that

$$\prod_{k=1}^{N} \mathrm{CCX}^{(N+1)}_{s_k,x_k,x_{N+1}} \left[|+\rangle^{\otimes N} \otimes |-\rangle \right]$$
$$= \mathbf{H}|s_1\rangle \otimes \ldots \otimes \mathbf{H}|s_N\rangle \otimes \mathbf{H}|1\rangle$$

Deutsch-Jozsa Algorithm

Balance Testing

- In cryptography, a hash function converts an input string into a binary representation
- A good hash function should be balanced, where all outputs are equally distributed
- David Deutsch and Richard Jozsa introduced a quantum algorithm for testing all input combinations (1992) in $O(1)$ time complexity

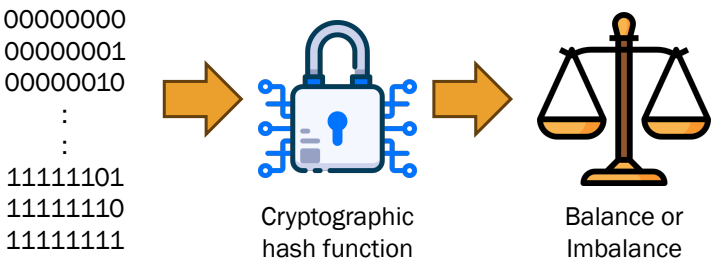


Figure 6: Balance testing problem

Deutsch-Jozsa Problem

- Suppose we have a black-box function f that takes an input binary string and returns either 0 or 1

$$f : \{0,1\}^N \mapsto \{0,1\}$$

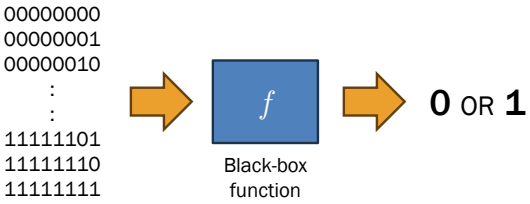


Figure 7: Deutsch-Jozsa problem

- We are guaranteed that this function is either constant or balance
  - Constant: It always returns either 0's or 1's for all inputs
  - Balanced: It returns equal amounts of 0's and 1's
- Best case: we can classify the function in first two samples – $[0,0]$ and $[1,1]$ for constant, while $[0,1]$ and $[1,0]$ for balanced
- Worse case: we have to enumerate at most half of all inputs and count the amounts of 0's and 1's $\Rightarrow O(N^2)$ space
- This problem also belongs to the BQP complexity class

Deutsch-Jozsa Algorithm

- Let's define a quantum operator $\mathbf{U}_f$ in response to the function f

$$\begin{aligned}
\mathbf{U}_f &= (|x_{1:N}\rangle\langle x_{1:N}|) \otimes (|y \oplus f(x_{1:N})\rangle\langle y|) \\
&= \mathbf{I}_N \otimes (|0 \oplus f(x_{1:N})\rangle\langle 0| \\
&\qquad + |1 \oplus f(x_{1:N})\rangle\langle 1|) \\
&= \mathbf{I}^N \otimes \begin{bmatrix} 0 \oplus f(x_{1:N}) & 0 \\ 0 & 1 \oplus f(x_{1:N}) \end{bmatrix}
\end{aligned}$$

Notice the phase kickback technique

- We setup a balance score of f as follows

$$b(f) = \frac{1}{\sqrt{2^N}} \sum_{j=0}^{2^N-1} (-1)^{f(\mathbb{B}_N(j))}$$

$$\begin{aligned}
&\mathbf{U}_f \left(|+\rangle^{\otimes N} \otimes |-\rangle \right) \\
&= \frac{1}{\sqrt{2^N}} \sum_{j=0}^{2^N-1} |\mathbb{B}_N(j)\rangle \otimes \left(\frac{0 \oplus f(\mathbb{B}_N(j))}{\sqrt{2}} |0\rangle \right) \\
&\qquad - |\mathbb{B}_N(j)\rangle \otimes \left(\frac{1 \oplus f(\mathbb{B}_N(j))}{\sqrt{2}} |1\rangle \right) \\
&= \frac{1}{\sqrt{2^N}} \sum_{j=0}^{2^N-1} (-1)^{f(\mathbb{B}_N(j))} |\mathbb{B}_N(j)\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}} \\
&= \left(\frac{1}{\sqrt{2^N}} \sum_{j=0}^{2^N-1} (-1)^{f(\mathbb{B}_N(j))} |\mathbb{B}_N(j)\rangle \right) \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}} \\
&= b(f) \left(\sum_{j=0}^{2^N-1} |\mathbb{B}_N(j)\rangle \right) \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}
\end{aligned}$$

Deutsch-Jozsa Algorithm

- Algorithm:

$$
\begin{aligned}
& \mathbf{H}^{\otimes N+1}\mathbf{U}_f\mathbf{H}^{\otimes N+1}\left(|0\rangle^{\otimes N}\otimes|1\rangle\right) \\
= \; & \mathbf{H}^{\otimes N+1}\mathbf{U}_f\left(|+\rangle^{\otimes N}\otimes|-\rangle\right) \\
= \; & \mathbf{H}^{\otimes N+1}\left[\left(b(f)\sum_{j=0}^{2^N-1}|\mathbb{B}_N(j)\rangle\right)\otimes|-\rangle\right] \\
= \; & \left(b(f)\,|0\rangle^{\otimes N}\right)\otimes|1\rangle \\
= \; & b(f)\left(|0\rangle^{\otimes N}\otimes|1\rangle\right)
\end{aligned}
$$

- Difference from BV Algorithm: black-box function f is either constant or balanced, implying there might be multiple oracles

- We can now solve the Deutsch-Jozsa problem by measuring the amplitude of pure state $|0\rangle^{\otimes N}$

$$
\left[b(f)\right]^2 = \left|\frac{1}{\sqrt{2^N}}\sum_{j=0}^{2^N-1}(-1)^{f(\mathbb{B}_N(j))}\right|^2
$$

- Evaluation
  - $b(f)$ evaluates to 1 if $f(x_{1:N})$ is constant
  - $b(f)$ evaluates to 0 if it is balanced
- Advantage: We have evaluated all possible inputs at the same time via quantum parallelism — Just apply the Hadamard gate to $|0\rangle^{\otimes N}$

Deutsch-Jozsa Algorithm

- If we have an oracle function f whose oracle is $|s_{1:N}\rangle$, we construct its kickback operator

$$\mathbf{U}_f \quad = \quad \mathbf{I}_N \otimes \begin{bmatrix} 0 \oplus f(x_{1:N}) & 0 \\ 0 & 1 \oplus f(x_{1:N}) \end{bmatrix}$$

- The Deutsch-Jozsa algorithm for f is as follows

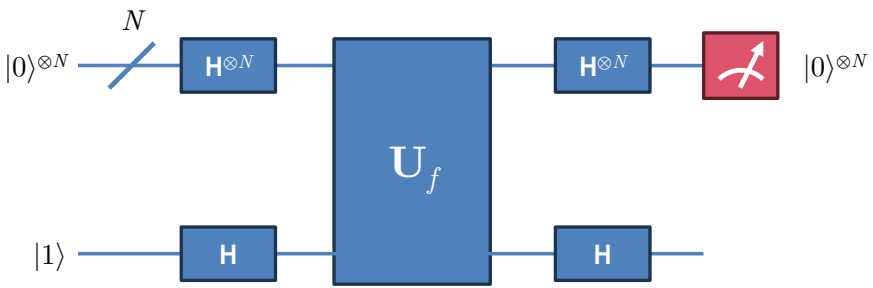


Figure 8: Deutsch-Jozsa Algorithm

Summary

- DJ Algorithm speeds up the balance testing problem by exponential queries via superposition
- The balance score of f is the sum between 0's and 1's yielded from f

$$b(f) \quad = \quad \frac{1}{\sqrt{2^N}} \sum_{j=0}^{2^N-1} (-1)^{f(\mathbb{B}_N(j))}$$

- Evaluation
  - If $b(f) = 1$, then f is constant
  - If $b(f) = 0$, then f is balanced
  - Otherwise, f is neither of them

- For any blackbox function f that yields either 0 or 1, we construct

$$\mathbf{U}_f \quad = \quad \mathbf{I}_N \otimes \begin{bmatrix} 0 \oplus f(x_{1:N}) & 0 \\ 0 & 1 \oplus f(x_{1:N}) \end{bmatrix}$$

- DJ Algorithm evaluates the balance score $b(f)$ from an unseen quantum operator $\mathbf{U}_f$ using quantum parallelism and the phase kickback technique

$$b(f)\,|0\rangle^{\otimes N} \otimes |1\rangle \quad = \quad \mathbf{H}\mathbf{U}_f\mathbf{H}\left(|0\rangle^{\otimes N} \otimes |1\rangle\right)$$

- We obtain $b(f)$ by measuring the amplitude of $|0\rangle^{\otimes N}$

Deutsch-Jozsa Algorithm in PennyLane

```
no_bits = 4                                     # Length of the hash code
dev = qml.device('default.qubit', wires=no_bits + 1, shots=500)

def oracle(theta1, theta2, theta3):
    # Compute a hash code from the input wires
    qml.RX(theta1, wires=[0]);    qml.CNOT(wires=[0, 1])
    qml.RZ(theta2, wires=[2]);    qml.CNOT(wires=[2, 3])
    qml.RY(theta3, wires=4)
    qml.Toffoli(wires=[0, 2, no_bits])          # Last bit is the type of this hash code

@qml.qnode(dev)
def deutsch_jozsa():
    qml.X(wires=no_bits)                        # Prepare state |+>^N (*) |-> from |0>^(N+1)
    qml.Hadamard(wires=no_bits)
    for i in range(no_bits):
        qml.Hadamard(wires=[i])
    oracle(np.pi / 4, np.pi / 4, np.pi / 4)     # Apply the oracle operator U_f
    for i in range(no_bits):                    # Extract the balance from the last qubit
        qml.Hadamard(wires=[i])
    return qml.probs(wires=[no_bits])           # Return the balance

deutsch_jozsa()
# Expected result: [0.65, 0.35]
```

Exercise 5.2: Deutsch-Jozsa Algorithm

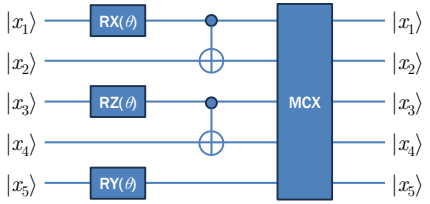
Let's define the following operators

$$\mathrm{MCX}^{(N+1)} = \prod_{k=1}^{N} \mathrm{CNOT}_{k,N+1}^{(N+1)}$$

$$\mathrm{MCZ}^{(N+1)} = \prod_{k=1}^{N} \mathrm{CZ}_{k,N+1}^{(N+1)}$$

Draw quantum circuits for the following oracle operators. Then implement them in PennyLane to check if each of them is balanced or not.

1. $\mathrm{MCX}^{(5)}$
2. $\mathrm{MCZ}^{(5)}$
3. $\mathrm{MCX}^{(5)}\ \mathrm{MCZ}^{(5)}$
4. $\mathrm{MCX}^{(5)}\ \mathrm{SWAP}_{1,5}^{(5)}$
5. $\mathrm{CCX}_{3,4,5}^{(5)}\ \mathrm{CCX}_{1,2,3}^{(5)}$
6. $\mathrm{MCX}^{(5)}\ \mathrm{X}_{3}^{(5)}\ \mathrm{X}_{1}^{(5)}$
7. $\mathrm{MCZ}^{(5)}\ \mathrm{Z}_{4}^{(5)}\ \mathrm{Z}_{2}^{(5)}$
8. Let $\mathrm{U}_f(\theta)$ be this quantum circuit.



Vary θ to make it balanced.

Quantum Teleportation

Quantum Teleportation

- Quantum teleportation is a communication technique, where information is transferred over a long distance via entangled states
- In traditional communication, information is transferred by state cloning
- No-cloning theorem: It is impossible to create an independent and identical copy of an unknown quantum state
  - We measure it by collapsing onto the axes, and construct a new state accordingly
  - However, we have completely destroyed the original state by measurement, resulting in no perfect clone

- Spoiler: This is NOT the teleportation in your favorite scifi movies

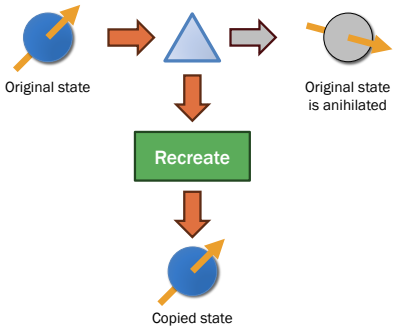


Figure 9: No-cloning theorem

Secure Communication via Entanglement

- State cloning can be emulated with entangled states, offering a secure means of communication
- Ingredients:
  - An unknown qubit $|x\rangle = a|0\rangle + b|1\rangle$
  - Entanglement $|\Phi^+\rangle$ distributed over quantum networks, e.g. fiber optics
  - Classical communication channel
- Setup:
  - Sender and receiver share an entangled pair $|\Phi\rangle = \frac{|00\rangle+|11\rangle}{2}$: $\left|\Phi_A^+\right\rangle$ for sender and $\left|\Phi_B^+\right\rangle$ for receiver
  - Note that $\left|\Phi_A^+\right\rangle$ and $\left|\Phi_B^+\right\rangle$ are either both 0's or both 1's at the same time
  - We have an unseen qubit $|x\rangle$

Sender: If we collapse $|y_1, y_2\rangle$, we will get $|y_3\rangle$

$$
\begin{aligned}
& |y_1, y_2, y_3\rangle \\
= \; & \mathbf{H}_1^{(3)}\,\mathbf{CNOT}_{1,2}^{(3)}\left(|x\rangle \otimes |\Phi^+\rangle\right) \\
= \; & \mathbf{H}_1^{(3)}\,\mathbf{CNOT}_{1,2}^{(3)}\left((a|0\rangle + b|1\rangle) \otimes (|00\rangle + |11\rangle)\right) \\
= \; & \mathbf{H}_1^{(3)}\,\mathbf{CNOT}_{1,2}^{(3)}(a|000\rangle + b|100\rangle \\
& \quad + a|011\rangle + b|111\rangle) \\
= \; & \mathbf{H}_1^{(3)}\,(a|000\rangle + b|110\rangle + a|011\rangle + b|101\rangle) \\
= \; & a|000\rangle + a|100\rangle + b|010\rangle - b|110\rangle \\
& + a|011\rangle + a|111\rangle + b|001\rangle - b|101\rangle \\
= \; & |00\rangle \otimes (a|0\rangle + b|1\rangle) + |01\rangle \otimes (b|0\rangle + a|1\rangle) \\
& + |10\rangle \otimes (a|0\rangle - b|1\rangle) + |11\rangle \otimes (-b|0\rangle + a|1\rangle)
\end{aligned}
$$

Secure Communication via Entanglement

- State cloning can be emulated with entangled states, offering a secure means of communication
- Ingredients:
  - An unknown qubit $|x\rangle = a|0\rangle + b|1\rangle$
  - Entanglement $|\Phi^+\rangle$ distributed over quantum networks, e.g. fiber optics
  - Classical communication channel
- Setup:
  - Sender and receiver share an entangled pair $|\Phi\rangle = \frac{|00\rangle+|11\rangle}{2}$: $\left|\Phi_A^+\right\rangle$ for sender and $\left|\Phi_B^+\right\rangle$ for receiver
  - Note that $\left|\Phi_A^+\right\rangle$ and $\left|\Phi_B^+\right\rangle$ are either both 0's or both 1's at the same time
  - We have an unseen qubit $|x\rangle$

Receiver: If we know $|y_1, y_2\rangle$, let's decode $|y_3\rangle$

$$
\begin{aligned}
& |z_1, z_2, z_3\rangle \\
= \; & \mathrm{CZ}_{1,3}^{(3)}\, \mathrm{CNOT}_{2,3}^{(3)} |y_1, y_2, y_3\rangle \\
= \; & \mathrm{CZ}_{1,3}^{(3)}\, \mathrm{CNOT}_{2,3}^{(3)} \\
& (|00\rangle \otimes (a|0\rangle + b|1\rangle) + |01\rangle \otimes (b|0\rangle + a|1\rangle) \\
& + |10\rangle \otimes (a|0\rangle - b|1\rangle) + |11\rangle \otimes (-b|0\rangle + a|1\rangle)) \\
= \; & \mathrm{CZ}_{1,3}^{(3)} \\
& (|00\rangle \otimes (a|0\rangle + b|1\rangle) + |01\rangle \otimes (a|0\rangle + b|1\rangle) \\
& + |10\rangle \otimes (a|0\rangle - b|1\rangle) + |11\rangle \otimes (a|0\rangle - b|1\rangle)) \\
= \; & |00\rangle \otimes (a|0\rangle + b|1\rangle) + |01\rangle \otimes (a|0\rangle + b|1\rangle) \\
& + |10\rangle \otimes (a|0\rangle + b|1\rangle) + |11\rangle \otimes (a|0\rangle + b|1\rangle)
\end{aligned}
$$

Quantum Teleportation Algorithm for Secure Communication

1. Sender and receiver are far apart, but they share a pair of secure keys ($\left|\Phi_A^+\right\rangle, \left|\Phi_B^+\right\rangle$)
2. We apply $\mathbf{CNOT}$ on $\left|\Phi_A^+\right\rangle$ conditioned on the message $|x\rangle$, and then apply $\mathbf{H}$ on $|x\rangle$
3. Encoding: We measure the message and the sender's secure key as (p, q), and send them over through classical channel $\Rightarrow$ Measurement causes immediate collapse $\left|\Phi_B^+\right\rangle := \left|\Phi_A^+\right\rangle$
4. Decoding: On $\left|\Phi_B^+\right\rangle$, we apply $\mathbf{CNOT}$ conditioned on q and $\mathbf{CZ}$ conditioned on p

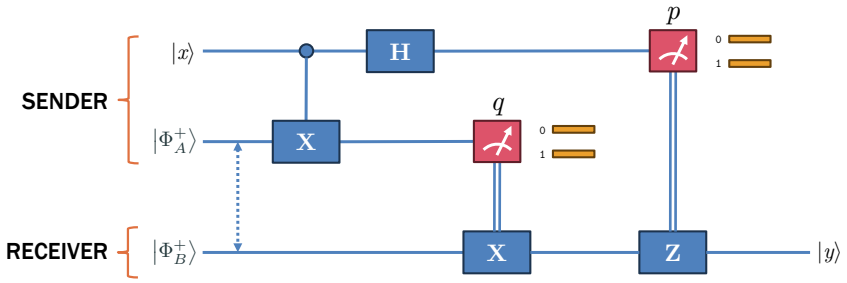


Figure 10: Quantum teleportation

Cartoon from JPL, NASA

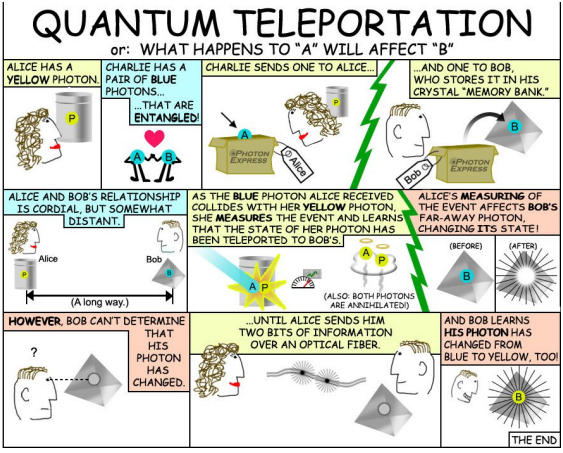


Figure 11: Quantum teleportation as a cartoon

Source: `https://www.jpl.nasa.gov/news/researchers-advance-quantum-teleportation/`

Quantum Teleportation in PennyLane

```
dev = qml.device('default.qubit', wires=['S', 'A', 'B'], shots=500)

def prepare_bellstate():
    qml.Hadamard(wires='A')
    qml.CNOT(wires=['A', 'B'])
def sender(info):
    qml.cond(info==1, qml.X)(wires=['S'])
    qml.CNOT(wires=['S', 'A'])
    qml.Hadamard(wires='S')
def receiver(p, q):
    qml.cond(q, qml.X)(wires='B')
    qml.cond(p, qml.Z)(wires='B')

@qml.qnode(dev)
def quantum_teleportation(info):
    prepare_bellstate()                              # Bell state |Phi+> as carrier
    sender(info)                                     # Encrypt the message
    (p, q) = (qml.measure('S'), qml.measure('A'))    # Measure (p,q) and send them
    receiver(p, q)                                   # via classical communication
    return qml.probs(wires='B')                      # Receiver extracts the message

quantum_teleportation(info=1)
# Expected result: [0.0, 1.0]
```

Quantum Gate Teleportation

- Quantum gate teleportation is an extension, where an entangled state is transferred over a long distance
- We cannot separate counterparts of an entangled state in teleportation because that will destroy the entanglement
- Ingredients
  - An entangled state $|x_A, x_B\rangle$
  - 4-qubit entanglement $|\chi\rangle$, where

$$|\chi_A, \chi_B, \chi_C, \chi_D\rangle = \frac{1}{2}(|0000\rangle + |0111\rangle + |1100\rangle + |1011\rangle)$$

  - Classical communication channel

- Setup:
  - Sender and receiver share a 4-qubit entangled state: $|\chi_A, \chi_D\rangle$ for sender, and $|\chi_B, \chi_C\rangle$ for receiver $\Rightarrow$ secure keys
  - $|\chi_A, \chi_D\rangle$ corresponds with $|\chi_B, \chi_C\rangle$:

$$\begin{aligned} |00\rangle &\mapsto |00\rangle \\ |01\rangle &\mapsto |11\rangle \\ |10\rangle &\mapsto |10\rangle \\ |11\rangle &\mapsto |01\rangle \end{aligned}$$

  - We have an unseen entanglement $|x_A, x_B\rangle$

Quantum Gate Teleportation

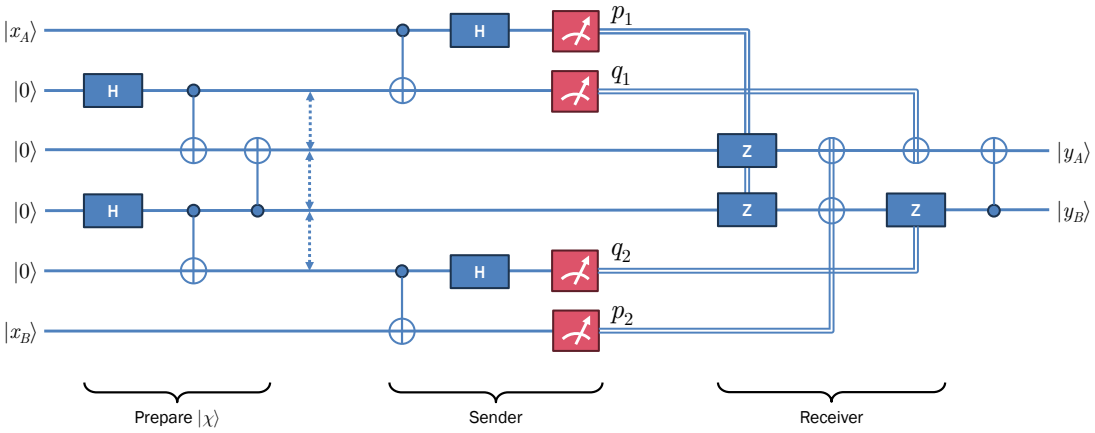


Figure 12: Quantum gate teleportation

Quantum Gate Teleportation in PennyLane/1

```
import pennylane as qml
from pennylane import numpy as np

dev = qml.device('default.qubit', wires=['S1', 'S2', 'X1', 'X2', 'X3', 'X4'])

def prepare_bellstate(wire1, wire2, minus=False):
    qml.Hadamard(wires=wire1)
    if minus:
        qml.PauliX(wires=wire2)
    qml.CNOT(wires=[wire1, wire2])

def prepare_chistate(wire1, wire2, wire3, wire4):
    prepare_bellstate(wire1, wire2, minus=False)
    prepare_bellstate(wire3, wire4, minus=False)
    qml.CNOT(wires=[wire3, wire2])

def sender(input1, input2, x1, x2, x3, x4):
    qml.CNOT(wires=[input1, x1])
    qml.CNOT(wires=[x4, input2])
    qml.Hadamard(wires=input1)
    qml.Hadamard(wires=x4)
```

Quantum Gate Teleportation in PennyLane/2

```
def receiver(p1, q1, p2, q2, x2, x3):
    qml.cond(p1, qml.Z)(wires=x2)
    qml.cond(p1, qml.Z)(wires=x3)
    qml.cond(p2, qml.X)(wires=x2)
    qml.cond(p2, qml.X)(wires=x3)
    qml.cond(q1, qml.X)(wires=x2)
    qml.cond(q2, qml.Z)(wires=x3)
    qml.CNOT(wires=[x3, x2])

@qml.qnode(dev, shots=500)
def quantum_gate_teleportation(info):
    prepare_bellstate('S1', 'S2', minus=info)     # Input entangled state
    prepare_chistate('X1', 'X2', 'X3', 'X4')      # Prepare the chi state
    sender('S1', 'S2', 'X1', 'X2', 'X3', 'X4')    # Sender part
    p1 = qml.measure('S1')                        # Measure four qubits
    p2 = qml.measure('S2')
    q1 = qml.measure('X1')
    q2 = qml.measure('X4')
    receiver(p1, p2, q1, q2, 'X2', 'X3')          # Receiver part
    return qml.counts(wires=['X2', 'X3'])

quantum_gate_teleportation(info=True)             # False = Bell-plus, True = Bell-minus
# Expected result: {'01': 250, '10': 250}
```

Exercise 5.3: Quantum Teleportation

Answer the following questions.

1. Explain why physical transmission of qubits over quantum networks is insecure.
2. Explain why quantum teleportation addresses the problem of eavesdropping in physical transmission with the no-cloning theorem.
3. Explain why the input qubit is destroyed after encrypting it into two classical bits (p, q).
4. Explain how we reconstruct the encrypted qubit from two bits (p, q) received from a classical communication channel.
5. Modify the quantum teleportation algorithm so that it uses $|\Psi^-\rangle$ as secure keys. Implement a PennyLane code to validate your idea.
6. Explain why we cannot teleport separate qubits of the entangled state.
7. Experiment with the PennyLane code for quantum gate teleportation with variations of Bell states $|\Psi^+\rangle, |\Psi^-\rangle, |\Phi^+\rangle, |\Phi^-\rangle$. Observe the algorithm's behaviors and outputs.

Superdense Coding

Superdense Coding

- Superdense coding is a compression method whereby two classical bits can be transmitted with a single quantum state
- Ingredients
  - Two classical bits $b_1 b_2$
  - Entanglement $|\Phi^+\rangle$ distributed over quantum networks, e.g. fiber optics
  - Quantum communication channel
- Setup:
  - Sender and receiver share an entangled pair $|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$: $\left|\Phi_A^+\right\rangle$ for sender and $\left|\Phi_B^+\right\rangle$ for receiver
  - Sender wants to send two classical bits $b_1 b_2$ to the receiver

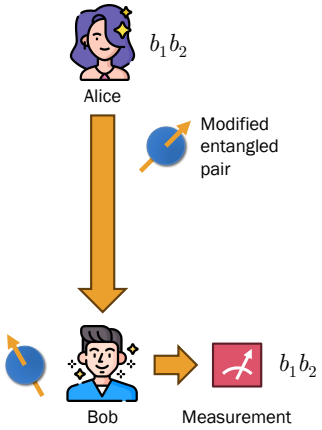


Figure 13: Overview of superdense coding

Superdense Coding

- Sender: We encode b_1 with sign-flipping and b_2 as bit-flipping

$$
\begin{aligned}
& Q(b_1, b_2) \\
= \; & \mathrm{CZ}^{(2)}_{b_1,1}\, \mathrm{CNOT}^{(2)}_{b_2,1}\, \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\
= \; & \mathrm{CZ}^{(2)}_{b_1,1} \begin{cases} \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) & [00] \\ \frac{1}{\sqrt{2}}(|10\rangle + |01\rangle) & [01] \\ \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) & [10] \\ \frac{1}{\sqrt{2}}(|10\rangle + |01\rangle) & [11] \end{cases}
\end{aligned}
$$

- (Cont'd)

$$
= \begin{cases} \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) & [00] \\ \frac{1}{\sqrt{2}}(|10\rangle + |01\rangle) & [01] \\ \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) & [10] \\ \frac{1}{\sqrt{2}}(|10\rangle - |01\rangle) & [11] \end{cases}
$$

Message b_1, b_2	State $Q(b_1, b_2)$
00	$\frac{1}{\sqrt{2}}(\|00\rangle + \|11\rangle)$
01	$\frac{1}{\sqrt{2}}(\|10\rangle + \|01\rangle)$
10	$\frac{1}{\sqrt{2}}(\|00\rangle - \|11\rangle)$
11	$\frac{1}{\sqrt{2}}(\|10\rangle - \|01\rangle)$

Table 1: Superdense coding

Superdense Coding

- Receiver: We decode sign-flipping and bit-flipping into b_1, b_2

$$
\begin{aligned}
& |b_1, b_2\rangle \\
= \; & \mathrm{H}_1^{(2)} \, \mathrm{CNOT}_{1,2}^{(2)} \begin{cases} \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) & [00] \\ \frac{1}{\sqrt{2}}(|10\rangle + |01\rangle) & [01] \\ \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) & [10] \\ \frac{1}{\sqrt{2}}(|10\rangle - |01\rangle) & [11] \end{cases} \\
= \; & \mathrm{H}_1^{(2)} \begin{cases} \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle) & [00] \\ \frac{1}{\sqrt{2}}(|11\rangle + |01\rangle) & [01] \\ \frac{1}{\sqrt{2}}(|00\rangle - |10\rangle) & [10] \\ \frac{1}{\sqrt{2}}(|11\rangle - |01\rangle) & [11] \end{cases}
\end{aligned}
$$

- (Cont'd)

$$
\begin{aligned}
= \; & \mathrm{H}_1^{(2)} \begin{cases} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle & [00] \\ \frac{1}{\sqrt{2}}(|1\rangle + |0\rangle) \otimes |1\rangle & [01] \\ \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \otimes |0\rangle & [10] \\ \frac{1}{\sqrt{2}}(|1\rangle - |0\rangle) \otimes |1\rangle & [11] \end{cases} \\
= \; & \begin{cases} |0\rangle \otimes |0\rangle & [00] \\ |0\rangle \otimes |1\rangle & [01] \\ |1\rangle \otimes |0\rangle & [10] \\ |1\rangle \otimes |1\rangle & [11] \end{cases}
\end{aligned}
$$

Superdense Coding

- We can transmit two classical bits using only one qubit over physical networks

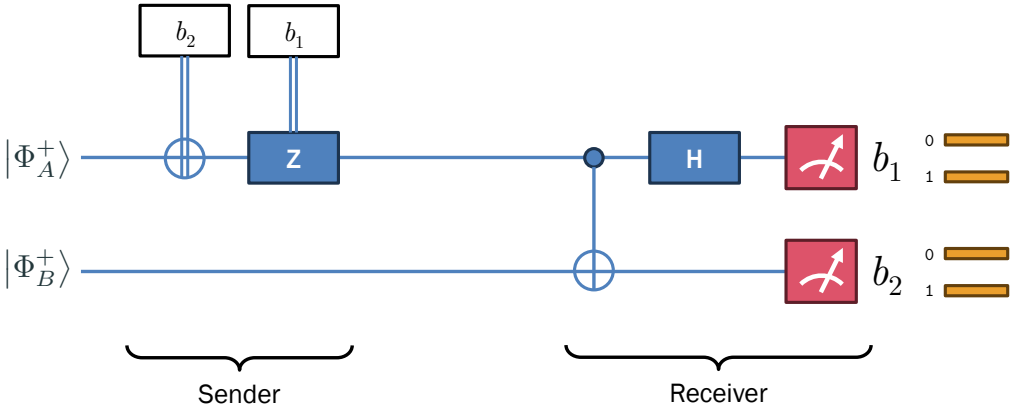


Figure 14: Superdense coding

Superdense Coding in PennyLane

```
dev = qml.device('default.qubit', wires=['A', 'B'], shots=500)

def prepare_bellstate():
    qml.Hadamard(wires='A')
    qml.CNOT(wires=['A', 'B'])

@qml.qnode(dev)
def superdense_coding(bit1, bit2):
    prepare_bellstate()

    # Sender
    qml.cond(bit2==1, qml.X)(wires='A')
    qml.cond(bit1==1, qml.Z)(wires='A')

    # Receiver
    qml.CNOT(wires=['A', 'B'])
    qml.Hadamard(wires='A')
    return qml.counts(wires=['A', 'B'])

superdense_coding(1, 0)
# Expected result: {'10': 500}
```

Exercise 5.4: Superdense Coding

Answer the following questions.

1. Explain why superdense coding is tolerant to eavesdropping, where the transmitted qubit $\left|\Phi_A^+\right\rangle$ is intercepted before the receiver receives it.
2. Modify the quantum teleportation algorithm so that it uses $|\Psi^-\rangle$ as entanglement. Implement a PennyLane code to validate your idea.
3. Since there are three axes in a quantum state, it is tempting to incorporate the third classical bit (as $\mathbf{CY}$) into superdense coding. Explain why doing so would perplex the decryption process.
4. To enhance security, we can integrate the quantum teleportation algorithm with the superdense coding. Implement with PennyLane the ultra-secure quantum communication. On the sender side, a qubit is encrypted as two classical bits and then transmitted via superdense coding. On the receiver side, the received qubit is decrypted into two classical bits and then decoded to the original quantum state. Design this algorithm and implement it with PennyLane to validate your idea.

Quantum Noisy Theorem

Density Matrix

- Quantum states constantly interact with the environment, collectively a.k.a. noise
- State vector $|\psi\rangle = \sum_{k=1}^{N} a_k |\psi_k\rangle$, where each $|\psi_k\rangle$ is a pure state, is extended to the density matrix to accommodate noise

$$\rho^{(\psi)} = \sum_{k=1}^{N} a_k^2 |\psi_k\rangle \langle\psi_k|$$

- Density matrix of mixed states is

$$\rho = \sum_{j=0}^{2^N-1} \sum_{k=0}^{2^N-1} q_{j,k} |\mathbb{B}_N(j)\rangle \langle\mathbb{B}_N(k)|$$

where $\sum_j \sum_k q_{j,k} = 1$

- Example: The density matrix of $|+\rangle$ is

$$\begin{aligned} \rho^{(+)} &= \frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1| \\ &= \frac{1}{2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{aligned}$$

- `qml.math.dm_from_state_vector`
- `qml.density_matrix(wires=...)`
- Noise: Interaction with the environment makes the off-diagonal non-zero, e.g.

$$\tilde{\rho}^{(+)} = \begin{bmatrix} 0.4 & 0.2 \\ 0.1 & 0.3 \end{bmatrix}$$

Common Types of Quantum Noisy Channel

- Bit-flip channel (X-noise)

$$\tilde{\rho} = (1-q)\rho + q\mathbf{X}\rho\mathbf{X}$$

- Sign-flip channel (Z-noise): most common

$$\tilde{\rho} = (1-q)\rho + q\mathbf{Z}\rho\mathbf{Z}$$

- Bit-sign-flip channel (Y-noise)

$$\tilde{\rho} = (1-q)\rho + q\mathbf{Y}\rho\mathbf{Y}$$

- Pauli channel

$$\tilde{\rho} = (1-q_x-q_y-q_z)\rho + q_x\mathbf{X}\rho\mathbf{X} + q_y\mathbf{Y}\rho\mathbf{Y} + q_z\mathbf{Z}\rho\mathbf{Z}$$

- Depolarization (information loss):

$$\tilde{\rho} = (1-q)\rho + \frac{q}{2}\mathbf{I}$$

$\Rightarrow$ Worse-case scenario!

- Amplitude damping channel (energy loss)

$$|0\rangle \mapsto |0\rangle \qquad |1\rangle \mapsto |0\rangle\sqrt{q} + |1\rangle\sqrt{1-q}$$

$\Rightarrow$ In superconducting and optical qubits

- Phase damping channel (coherence loss)

$$\tilde{\rho} = \begin{bmatrix} \rho_{11} & (1-q)\rho_{12} \\ (1-q)\rho_{21} & \rho_{22} \end{bmatrix}$$

$\Rightarrow$ Also most common

Purity

- Properties of density matrix ρ
  - It is Hermitian: $\rho = \rho^*$
  - It has a unit trace: $\mathrm{tr}(\rho) = 1$
  - It is positive-semidefinite: for all quantum states $|\psi\rangle$, we have $\langle\psi|\rho|\psi\rangle \geq 0$
- Purity of density matrix

$$\gamma(\rho) = \mathrm{tr}(\rho^2)$$

- PennyLane: `qml.math.purity`
- If a quantum state $|\psi\rangle$ is pure, we then have $\gamma(\rho^{(\psi)}) = 1$; otherwise, $\gamma(\rho^{(\psi)}) < 1$

$$\begin{aligned}\gamma(\rho^{(0)}) &= \mathrm{tr}(|0\rangle\langle 0|0\rangle\langle 0|) = 1\\ \gamma(\rho^{(1)}) &= \mathrm{tr}(|1\rangle\langle 1|1\rangle\langle 1|) = 1\end{aligned}$$

- Maximally mixed state is a quantum state with maximal noise

$$\rho_{\max} = \sum_{j=0}^{2^N-1}\sum_{k=0}^{2^N-1}\frac{1}{2^N}|\mathbb{B}_N(j)\rangle\langle\mathbb{B}_N(k)|$$

- Expectation: Expectation measured with a projector $\mathbf{U} = \sum_{k=1}^{N}\lambda_k|k\rangle\langle k|$ is

$$\begin{aligned}\left\langle \mathbf{U}^{(\rho)} \right\rangle &= \sum_{k=1}^{N}\lambda_k q_{k,k}\\ &= \sum_{k=1}^{N}\lambda_k \,\mathrm{tr}(\rho\,|k\rangle\langle k|)\\ &= \mathrm{tr}(\rho\mathbf{U})\end{aligned}$$

Partial Trace

- Partial trace of multi-qubit density matrix

$$\mathrm{tr}_j(\rho) = \sum_{k\in\{0,1\}} (\mathbf{U}_{j,k,N})^* \rho (\mathbf{U}_{j,k,N})$$

where

$$\mathbf{U}_{j,k,N} = \mathbf{I}_{j-1} \otimes |k\rangle \otimes \mathbf{I}_{N-j}$$

- Reduced density matrix: $\mathrm{tr}_j(\rho)$ removes the j-th qubit from ρ and sums the remaining reduced states
- This concept imitates quantum state transmission via quantum channel

- Example: Our density matrix is:

$$\rho = \frac{1}{2}|00\rangle + \frac{1}{4}|01\rangle + \frac{1}{6}|10\rangle + \frac{1}{8}|11\rangle$$

Reduced density matrices of ρ are:

$$\begin{aligned}\mathrm{tr}_1(\rho) &= \frac{1}{2}|\not{0}0\rangle + \frac{1}{4}|\not{0}1\rangle + \frac{1}{6}|\not{1}0\rangle + \frac{1}{8}|\not{1}1\rangle \\ &= \left(\frac{1}{2}+\frac{1}{6}\right)|0\rangle + \left(\frac{1}{4}+\frac{1}{8}\right)|1\rangle \\ \mathrm{tr}_2(\rho) &= \frac{1}{2}|0\not{0}\rangle + \frac{1}{4}|0\not{1}\rangle + \frac{1}{6}|1\not{0}\rangle + \frac{1}{8}|1\not{1}\rangle \\ &= \left(\frac{1}{2}+\frac{1}{4}\right)|0\rangle + \left(\frac{1}{6}+\frac{1}{8}\right)|1\rangle\end{aligned}$$

Purification

- Purification of quantum state $|\psi\rangle$ can be done by finding partial traces

$$\begin{aligned} |\psi_1\rangle &= \mathrm{tr}_2(|\psi\rangle\langle\psi|) \\ |\psi_2\rangle &= \mathrm{tr}_1(|\psi\rangle\langle\psi|) \end{aligned}$$

- If a density matrix ρ has K eigen-pairs $(\lambda_k, |e_k\rangle)$, i.e.

$$\rho = \sum_{k=1}^{K} \lambda_k |e_k\rangle\langle e_k|$$

we can purity ρ into each pure state $|i_k\rangle$

$$|\psi\rangle = \sum_{k=1}^{K} \sqrt{\lambda_k}\,(|e_k\rangle \otimes |i_k\rangle)$$

- Example: We have a noisy density matrix

$$\rho = \begin{bmatrix} 0.4 & 0.2 \\ 0.1 & 0.3 \end{bmatrix}$$

- We decompose ρ into two eigen-pairs

$$\left(0.5, \frac{1}{\sqrt{5}}\begin{bmatrix}2 & 1\end{bmatrix}^\top\right) \quad \left(0.2, \frac{1}{\sqrt{2}}\begin{bmatrix}1 & -1\end{bmatrix}^\top\right)$$

- We can purify ρ into pure states $|0\rangle$ and $|1\rangle$

$$\begin{aligned} |\psi\rangle = &\sqrt{0.5} \times \left(\frac{1}{\sqrt{5}}\begin{bmatrix}2 & 1\end{bmatrix}^\top \otimes |0\rangle\right) \\ &+ \sqrt{0.2} \times \left(\frac{1}{\sqrt{2}}\begin{bmatrix}1 & -1\end{bmatrix}^\top \otimes |1\rangle\right) \end{aligned}$$

Fidelity

- Fidelity: similarity between two density matrices ρ and σ

$$F(\rho,\sigma) = \left(\text{tr}\sqrt{\sqrt{\rho}\sigma\sqrt{\rho}}\right)^2$$

- Case 1: Both ρ and σ are pure states

$$F(\rho,\sigma) = |\langle\rho|\sigma\rangle|^2$$

- Case 2: ρ and/or σ are noisy
  1. Diagonalize ρ into $(\mathbf{V},\mathbf{D})$
  2. Compute $\sqrt{\rho} = \mathbf{V}\mathbf{D}^{1/2}\mathbf{V}^*$
  3. Compute $\mathbf{M} = \sqrt{\rho}\sigma\sqrt{\rho}$
  4. Diagonalize $\mathbf{M}$ into $(\mathbf{W},\Lambda)$
  5. Compute $\sqrt{\mathbf{M}} = \mathbf{W}\Lambda^{1/2}\mathbf{W}^*$
  6. Compute $F(\rho,\sigma) = \text{tr}\sqrt{\mathbf{M}}$

- Diagonalization
  1. Decompose ρ into K eigen-pairs $(\lambda_k, \mathbf{v}_k)$
  2. Construct singular matrix $\mathbf{V} = [\mathbf{v}_1|\ldots|\mathbf{v}_K]$ and diagonal matrix $\mathbf{D} = \text{diag}(\lambda_1,\ldots,\lambda_K)$
  3. Return $(\mathbf{V},\mathbf{D})$

- Example: Let's compute $F(|+\rangle, |0\rangle)$

$$\begin{aligned} F(|+\rangle,|0\rangle) &= |\langle+|0\rangle|^2 \\ &= \left|\left(\frac{1}{\sqrt{2}}\langle 0| + \frac{1}{\sqrt{2}}\langle 1|\right)|0\rangle\right|^2 \\ &= \left|\frac{1}{\sqrt{2}}\langle 0|0\rangle + \frac{1}{\sqrt{2}}\langle 1|0\rangle\right|^2 \\ &= \left|\frac{1}{\sqrt{2}} + 0\right|^2 = \frac{1}{2} \end{aligned}$$

Fidelity

- Example: For noisy states

$$\rho = \begin{bmatrix} 0.4 & 0.2 \\ 0.1 & 0.3 \end{bmatrix} \quad \sigma = \begin{bmatrix} 1/\sqrt{2} & 0 \\ 0 & 1/\sqrt{2} \end{bmatrix}$$

Let's compute the fidelity $F(\rho, \sigma)$

- We diagonalize ρ into

$$\mathbf{V} = \begin{bmatrix} 2/\sqrt{5} & 1/\sqrt{2} \\ 1/\sqrt{5} & -1/\sqrt{2} \end{bmatrix} \quad \mathbf{D} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.2 \end{bmatrix}$$

- We compute $\sqrt{\rho} = \mathbf{V}\mathbf{D}^{1/2}\mathbf{V}^*$

$$\sqrt{\rho} = \begin{bmatrix} 0.79 & 0.06 \\ 0.06 & 0.37 \end{bmatrix}$$

- We compute $\mathbf{M} = \sqrt{\rho}\sigma\sqrt{\rho}$

$$\mathbf{M} = \begin{bmatrix} 0.44 & 0.05 \\ 0.05 & 0.10 \end{bmatrix}$$

- We diagonalize $\mathbf{M}$ into

$$\mathbf{W} = \begin{bmatrix} 0.99 & -0.14 \\ 0.14 & 0.99 \end{bmatrix} \quad \Lambda = \begin{bmatrix} 0.45 & 0 \\ 0 & 0.09 \end{bmatrix}$$

- We compute $\sqrt{\mathbf{M}} = \mathbf{W}\Lambda^{1/2}\mathbf{W}^*$

$$\sqrt{\mathbf{M}} = \begin{bmatrix} 0.66 & 0.05 \\ 0.05 & 0.31 \end{bmatrix}$$

- Fidelity $F(\rho, \sigma) = \mathrm{tr}\sqrt{M} = 0.66 + 0.31$

Fidelity of Density Matrices in PennyLane

```
import pennylane as qml
from pennylane import numpy as np
from pennylane import math as qmath    # We need this qmath, too

# Depolarization channel
def noisy_dm(mat, p_noise):
    noise = p_noise / 2 * np.eye(2)
    new_mat = (1 - p_noise) * mat + noise
    return new_mat

def main():
    # Convert a state vector into a density matrix (dm)
    mat = qmath.dm_from_state_vector([0.9, 0.1])
    mat = mat / qmath.norm(mat)

    for p_noise in range(10):
        noisy_mat = noisy_dm(mat, p_noise * 0.1)    # Noisy matrix with noise parameter
        f = qmath.fidelity(mat, noisy_mat)          # Fidelity of quantum states
        print(f'p_noise = {p_noise * 0.1:4.2f}: fidelity = {f:6.4f}')

main()
```

Fidelity of State Vectors in PennyLane

```
import pennylane as qml
from pennylane import numpy as np
from pennylane import math as qmath

# Phase damping channel
def noisy_vec(vec, p_noise):
    noise = np.array([ [0.0, 1 - p_noise],
                       [1 - p_noise, 0.0] ])
    new_vec = noise @ vec
    return new_vec

def main():
    vec = np.array([1.0, 1.0])
    vec = vec / qmath.norm(vec)

    for p_noise in range(10):
        new_vec = noisy_vec(vec, p_noise * 0.1)         # Noisy vector with noise parameter
        f = qmath.fidelity_statevector(vec, new_vec)    # Fidelity of quantum states
        print(f'p_noise = {p_noise * 0.1:4.2f}: fidelity = {f:6.4f}')

main()
```

Exercise 5.5: Quantum Noisy Theorem

Convert the following state vectors into equivalent density matrices. Then separately compute their X-, Z-, and Y-noises with the parameter $q = 0.25$.

1. $|0\rangle$
2. $|1\rangle$
3. $|+\rangle$
4. $|-\rangle$
5. $|\Psi^+\rangle$
6. $|\Phi^-\rangle$
7. $|{+}{+}\rangle$
8. $|{+}{-}\rangle$
9. $|{+}{-}{+}\rangle$

Implement PennyLane code.

10. Let $\rho^{(-)}$ be the density matrix of $|-\rangle$. Let $\tilde{\rho}^{(-)}$ be the sign-flip channel of $\rho^{(-)}$ with $q = 0.25$. Then compute the purity $\gamma(\tilde{\rho}^{(-)})$.
11. Regarding the previous question, purify $\tilde{\rho}^{(-)}$. [Hint: We can decompose a matrix with `numpy.linalg.eig`.]

Answer the following questions.

12. Explain why a noisy density matrix cannot be converted to an equivalent state vector.
13. For any two pure states $|\rho\rangle$ and $|\sigma\rangle$, where

$$|\sigma\rangle = |\rho\rangle \operatorname{cis}\alpha \qquad \langle\rho|\sigma\rangle = \cos\theta$$

show that their fidelity is

$$F(\rho,\sigma) \quad = \quad \cos^2\theta$$

14. Regarding the previous question, discuss how the fidelity could be used as a loss function in machine learning.

Quantum Error Correction

Noisy Intermediate-Scale Quantum (NISQ) Computers

- In current technology (mid-2020s), quantum computers are not fully noise-tolerant due to noisy hardware implementation of quantum gates
- Quantum decoherence hinders the reliability of multi-qubit computers, where more qubits may exponentially introduce noise to the quantum gates
- State-of-the-art milestone is the noisy intermediate-scale quantum (NISQ) computers equipped with efficient quantum error correction
- Current NISQ devices contain 50-1,000 physical qubits and they are noise-intolerant because of noisy hardware implementation and quantum decoherence
- Gate fidelities for one-qubit operations is 99.5% and that for two-qubit operations is 95-99%, while the error rate is 0.1%/gate
- For a quantum algorithm, we prefer less number of quantum gates due to this exponential increase of noise due to per-gate error rate

Classical Error Correction

- We use a redundant correcting code to detect errors

$$\begin{aligned} \text{Encode:} \quad \mathbf{y} &= \left[\mathrm{I}_N | \mathrm{A}^\top\right]^\top \mathbf{x} \\ \text{Decode:} \quad \mathbf{p} &= \left[\mathrm{A} | \mathrm{I}_L\right] \hat{\mathbf{y}} \end{aligned}$$

where $\mathbf{x}$ is a binary-string block of length N and $\left[\mathrm{A}\right]_{L \times N}$ is a parity-check matrix

- Hamming Code: For a 4-bit block $b_1 b_2 b_3 b_4$, there are 3 parity check bits $p_1 p_2 p_3$

$$\mathrm{A} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}_{3 \times 4}$$

This is called the Hamming(7,4) code

- Encoding: We can encode a block `1011` by

$$\begin{aligned} & \left[\mathrm{I}_4 | \mathrm{A}^\top\right]^\top \begin{bmatrix} 1 & 0 & 1 & 1 \end{bmatrix}^\top \\ = & \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hline 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{array}\right] \times \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \\ = & \left[\begin{array}{cccc|ccc} 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{array}\right]^\top \end{aligned}$$

The parity-check bits are `010`

- Decoding: Via noise channel, we receive a corrupted block $\mathtt{1010\underline{0}010}$

$$
\begin{aligned}
&\left[\mathbf{A}|\mathbf{I}_3\right]\hat{\mathbf{y}} \\
&= \left[\begin{array}{cccc|ccc} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right] \times \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^\top \triangleq \begin{bmatrix} p_1 & p_2 & p_3 \end{bmatrix}^\top
\end{aligned}
$$

- Syndrome: Regarding $p_1p_2p_3$, we consult the Hamming(7,4) parity-check diagram
- With $p_1p_2p_3 = \mathtt{111}$, we identify the corrupted position by the intersection of three circles p_1, p_2, p_3, resulting in b_4

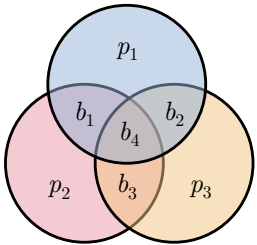


Figure 15: Hamming(7,4) parity-check diagram

Quantum Error Correction

- Quantum error correction (QEC) is a technique for protecting quantum information from decoherence and noise
- Sources of noise: faulty state storage, faulty gates, faulty state preparation, and faulty measurement ⇒ low fidelity
- Syndrome: types of quantum error
  - Random bit-flipping via $\mathbf{RX}$, e.g.

$$a|0\rangle + b|1\rangle \quad \Rightarrow \quad b|0\rangle + a|1\rangle$$

  - Random sign-flipping via $\mathbf{RZ}$, e.g.

$$a|0\rangle + b|1\rangle \quad \Rightarrow \quad a|0\rangle - b|1\rangle$$

  - Both types may happen at the same time

- Bit-flipping noise with probability q

$$\mathbf{E}_{\text{bit}}\rho \quad = \quad (1-q)\rho + q\mathbf{X}\rho\mathbf{X}$$

- Sign-flipping noise: with probability q

$$\mathbf{E}_{\text{sign}}\rho \quad = \quad (1-q)\rho + q\mathbf{Z}\rho\mathbf{Z}$$

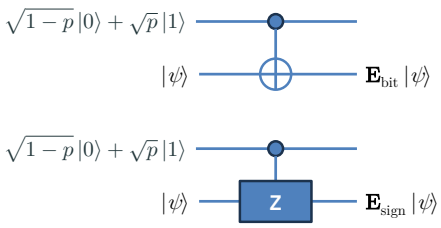


Figure 16: Implementation of noisy channels

Correction for Bit-Flipping

- We can apply the 3-repetition code for bit-flipping correction

$$\begin{aligned} & \mathrm{CNOT}^{(3)}_{1,3}\mathrm{CNOT}^{(3)}_{1,2}(a\,|0\rangle + b\,|1\rangle) \otimes |00\rangle \\ = \;& \mathrm{CNOT}^{(3)}_{1,3}(a\,|00\rangle + b\,|11\rangle) \otimes |0\rangle \\ = \;& a\,|000\rangle + b\,|111\rangle \end{aligned}$$

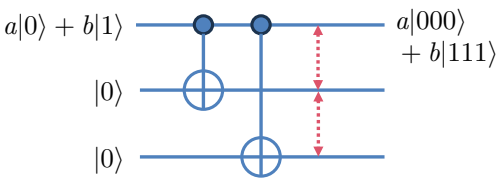


Figure 17: Encoder for bit-flipping

- When receiving a block $|b_1, b_2, b_3\rangle$ via noisy channel, we determine the syndrome by

$$|s1, s_2\rangle \quad = \quad |b_1 \oplus b_2, b_1 \oplus b_3\rangle$$

where each bit determines if $b_1 = b_2$ and $b_1 = b_3$, respectively

Syndromes	Corrections
00	No correction
01	Bit-flip b3
10	Bit-flip b2
11	Bit-flip b1

Table 2: Correction table for bit-flipping

Correction for Bit-Flipping

- We can address the correction table by the decoder circuit below

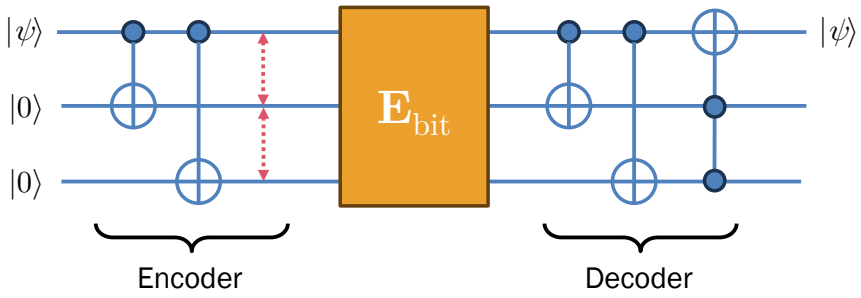


Figure 18: Encoder-decoder for bit-flipping correction

Correction for Sign-Flipping

- Sign-flipping is equivalent to bit-flipping when applied with the Hadamard gate

$$\begin{aligned} \mathrm{H}(a|+\rangle + b|-\rangle) &\Rightarrow \mathrm{H}(a|-\rangle + b|+\rangle) \\ \frac{a+b}{\sqrt{2}}|0\rangle + \frac{a-b}{\sqrt{2}}|1\rangle &\Rightarrow \frac{a+b}{\sqrt{2}}|0\rangle - \frac{a-b}{\sqrt{2}}|1\rangle \end{aligned}$$

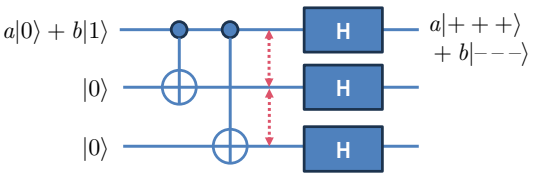


Figure 19: Encoder for sign-flipping

- When receiving a block $|b_1, b_2, b_3\rangle$ via noisy channel, we determine the syndrome by

$$|s_1, s_2\rangle \quad = \quad |b_1 \oplus b_2, b_1 \oplus b_3\rangle$$

where each bit determines if $b_1 = b_2$ and $b_1 = b_3$, respectively

Syndromes	Corrections
$++$	No correction
$+-$	Sign-flip b3
$-+$	Sign-flip b2
$--$	Sign-flip b1

Table 3: Correction table for sign-flipping

Correction for Sign-Flipping

- We can address the correction table by the decoder circuit below

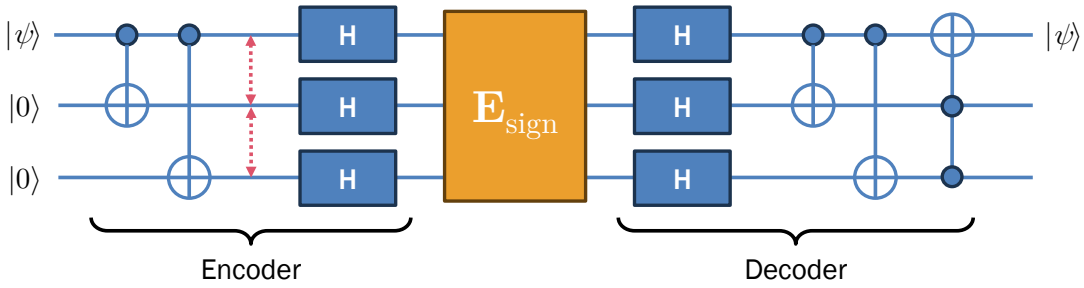


Figure 20: Encoder-decoder for bit-flipping correction

Shor Code

- Shor code is a 9-repetitive code that interleaves bit-flipping and sign-flipping to accommodate any arbitrary error $\mathbf{E} = \mathrm{I} + c_x \mathrm{X} + c_y \mathrm{Y} + c_z \mathrm{Z}$

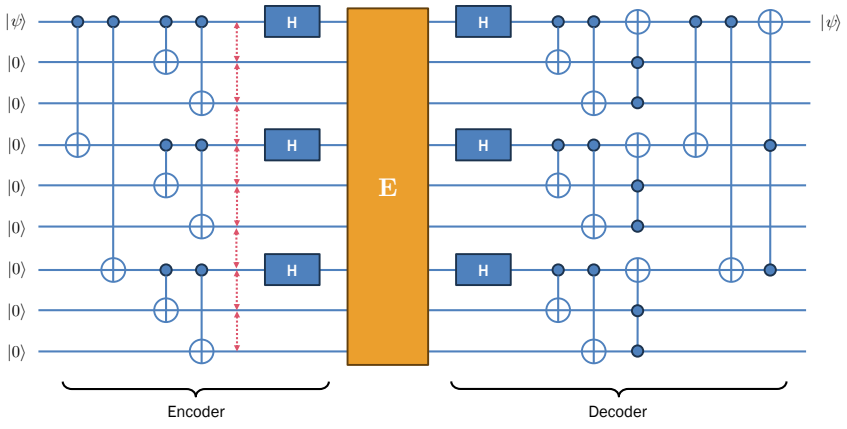


Figure 21: Shor code

Surface Code (Dennis et al., 2002)

- State-of-the-art error correction algorithm for large-scale quantum computing (i.e. high data-per-parity ratio)
- Data qubits are open circles (○), while Z- and X-measurements are filled circles (●)
- Both types of measurement qubits are computed from surrounding data qubits

$$Z_{abcd} = \begin{cases} +1 & \text{even number of 1's} \\ -1 & \text{otherwise} \end{cases}$$

$$X_{abcd} = \begin{cases} +1 & \text{even number of 1's} \\ -1 & \text{otherwise} \end{cases}$$

- These qubits are measured via projection

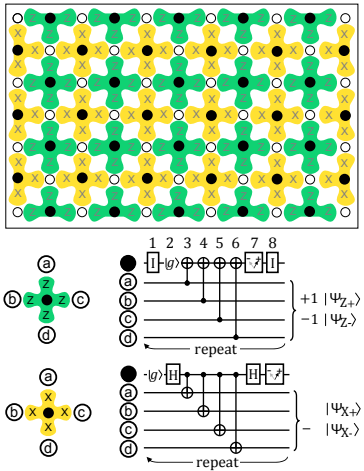


Figure 22: Surface code (Fowler et al., 2012)

Syndrome Detection in Surface Code

- Syndrome: Error is detected in each cycle throughout the network, where the current Z- and X-measures are compared with those from the previous timestep (stored in classical computer)
- Single qubit error always causes a mismatched pair of adjacent measures
  - c has a bit-flip error

$$Z_{abcd}^{(t+1)} \neq Z_{abcd}^{(t)} \qquad Z_{ecfg}^{(t+1)} \neq Z_{ecfg}^{(t)}$$

  - c has a sign-flip error

$$X_{abcd}^{(t+1)} \neq X_{abcd}^{(t)} \qquad X_{ecfg}^{(t+1)} \neq X_{ecfg}^{(t)}$$

  - Both errors may happen at the same time

- Multiple qubit errors result in a mismatch pair of separate measures, and we have to find the shortest path between them with the minimum-weight perfect-matching algorithm (Edmonds and Canad, 1965)

Figure 23: Error detection as Minesweeper

Error Detection in Surface Code

- Isolated qubit error results in a square of measure mismatches
- Once a qubit error is detected, we can bit-flip it (Z-measure) or sign-flip it (X-measure)

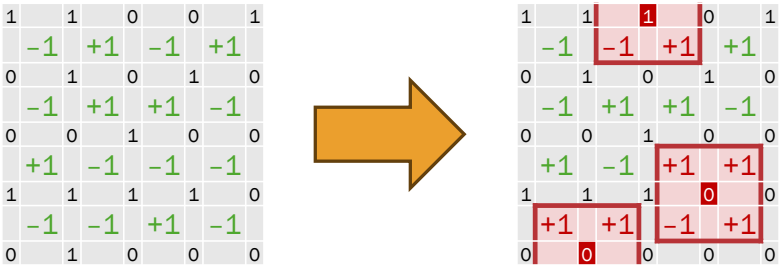


Figure 24: The change of Z- and X- measures is used to locate error bits

Error Detection in Surface Code

- Consecutive qubit errors result in measure mismatches at the corners
- We try to draw a rectangle that covers these qubit errors
- Some of these rectangles may also overlap each other

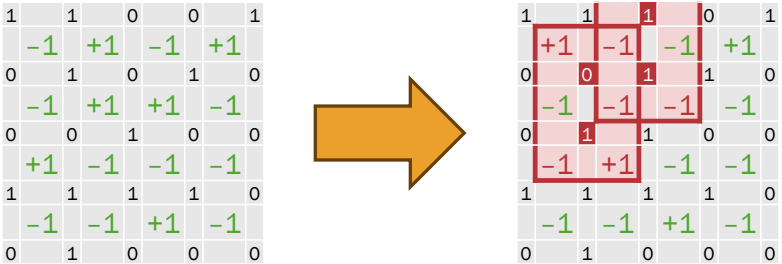


Figure 25: Consecutive qubit errors

Exercise 5.6: Quantum Error Correction

Encode the following codes with the Hamming(7,4) code.

1. `0000`
2. `0100`
3. `1010`
4. `1100`
5. `1101`
6. `0101`
7. `0111`
8. `1111`

Decode the following codes with the Hamming(7,4) code. Determine which bit needs to be corrected.

9. `1000000`
10. `0100111`
11. `1010001`
12. `1101011`
13. `1111100`
14. `0101011`
15. `0111001`
16. `1111110`

Answer the following questions.

17. Let's compute the redundancy rate r of an error correction method.

$$r = \frac{\text{num of parity bits}}{\text{num of total bits}}$$

For example, the redundancy rate of Hamming(7,4) is $3/7$. Compute the redundancy rate of Shor's code.

18. We want to store N qubits in our data storage equipped with the Surface code. Compute the redundancy rate of this data storage. Discuss why the Surface code outperforms Shor's code.

Conclusion

Conclusion

- Unlocking a combination padlock with Bernstein-Vazirani Algorithm and the phase kickback technique
- Measuring the balance of hash functions with Deutsch-Jozsa Algorithm
- Transferring a qubit securely with quantum teleportation
- Transferring two classical bits in one qubit with superdense coding
- Quantum noisy theorem, density matrix, state purification, and state fidelity
- Correcting error qubits with redundancy codes: Shor's code and Surface code

Questions?

Target Learning Outcomes/1

- Quantum Algorithm Analysis and Implementation
  1. Differentiate between the Bernstein-Vazirani and Deutsch-Jozsa problems, specifically identifying when a function is constant, balanced, or possesses a secret oracle.
  2. Apply the phase kickback technique to manipulate ancillary qubits for state evaluation.
  3. Construct and simulate these algorithms using quantum programming frameworks like PennyLane.
- Quantum Communication Protocols
  1. Explain the limitations imposed by the No-Cloning Theorem and how quantum teleportation bypasses this to achieve secure communication.
  2. Execute the superdense coding protocol to transmit two classical bits of information using a single qubit.
  3. Contrast standard teleportation with Quantum Gate Teleportation for long-distance entangled state transfer.

Target Learning Outcomes/2

- Noise Modeling and Characterization
  1. Formulate the density matrix to represent mixed states and environmental interactions (noise).
  2. Identify and model common quantum noisy channels, including bit-flip (X), sign-flip (Z), and depolarization.
  3. Calculate state fidelity and purity to measure the similarity between ideal and noisy quantum states.
- Error Correction and Fault Tolerance
  1. Analyze the Hamming Code as a classical foundation for understanding quantum redundancy.
  2. Design error-correction circuits using the Shor Code (9-qubit) and 3-repetition code to mitigate bit-flip and sign-flip errors.
  3. Evaluate the efficiency of the surface code in the context of NISQ hardware and "Minesweeper-style" syndrome detection.