Bachelor's Thesis (Academic Year 2025)

Entanglement Ejection: Logical Entanglement Generation for Distributed Quantum Computing on Surface Codes

Keio University
Faculty of Environment and Information Studies

Kento Samuel Soon

January 2026

Bachelor's Thesis Academic Year 2025

Entanglement Ejection: Logical Entanglement Generation for Distributed Quantum Computing on Surface Codes

Abstract

Achieving practical quantum advantage demands both fault-tolerance and scaling beyond a single processor. Crucially, this relies on the generation of logical entanglement between distributed nodes, where existing methods typically have high resource overheads. In this thesis, I introduce *Entanglement Ejection*, a novel surface code scheme that transforms a physical Bell pair into shared logical entanglement via an intermediate physical–logical state. Numerical simulations demonstrate that our method outperforms existing protocols for moderately noisy regimes and maintains superiority across all noise levels when input entanglement quality is low. From such observations, I expect entanglement ejection to serve as a primitive for fault-tolerant distributed quantum computing.

Keywords

Quantum Error Correction, Fault-Tolerant Quantum Computing, Distributed Quantum Computing, Quantum Entanglement

Faculty of Environment and Information Studies
Keio University

Kento Samuel Soon

学士論文　2025 年度（令和 7 年度）

Entanglement Ejection: 表面符号を用いた分散量子計算のための論理量子エンタングルメント生成

概要

実用的な量子優位性を実現するには，誤り耐性と単一プロセッサの枠を超えたスケーリングの両方が不可欠である．そのためには分散ノード間での論理エンタングルメントの生成が必要となるが，既存手法は多大なリソースオーバーヘッドを伴う傾向がある．本論文では，中間的な物理・論理エンタングルメント状態を介して物理ベルペアを共有された論理エンタングルメントへと変換する新規手法「Entanglement Ejection」を提案する．数値シミュレーションの結果，Entanglement Ejection は既存手法に対し，中程度のノイズ環境下で優れた性能を示した．さらに，入力エンタングルメントが低品質な場合においては，全てのノイズ領域にわたって優位性を示した．これらの結果から，提案手法「Entanglement Ejection」は誤り耐性分散量子計算の基本的な構成要素として有効であることが期待される．

キーワード

量子誤り訂正，謝り耐性量子計算，分散量子計算，量子エンタングルメント

慶應義塾大学 環境情報学部 環境情報学科

スーン 憲人サミュエル

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Quantum computers are known to have more advanced capabilities than their classical counterparts in several problem domains [1, 2]. However, advantageous quantum computing is extremely difficult to achieve in the presence of the inherent noise in quantum systems. Engineering the quantum system to possess lower physical noise does work up to a certain level but is not scalable. Consequently, we perform quantum error correction [3, 4], encoding physical qubits into logical qubits. This allows us to achieve error rates of practical regime, performing quantum algorithms fault-tolerantly.

Despite the theoretical guarantees of quantum error correction, scaling a monolithic quantum computer to the level required for running quantum algorithms in a robust manner, remains a formidable task. To counter this scaling issue, we require a distributed quantum computing environment [5] rather than a monolithic device. In this architecture, multiple smaller quantum systems are interconnected via a quantum mechanical resource known as entanglement.

To enable such fault-tolerant distributed quantum computing, it is therefore necessary to generate error-corrected logical entanglement between spatially separated nodes. Specifically, for the widely used surface code [6], there exist various protocols for achieving such logical entanglement. These include state injection [7, 8], which directly encodes physical entanglement into its logical form, and remote implementations of lattice surgery [9, 10]. However, these methods typically exhibit high physical resource overheads; state injection is probabilistic, and remote lattice surgery consumes entanglement resources proportional to the code size.

1.2 Contribution

This thesis proposes a novel logical entanglement generation scheme termed **entanglement ejection**. This method bridges the gap between physical Bell pairs and logical surface code qubits by effectively “ejecting” a qubit entangled with the original surface code. By performing entanglement swapping between the ejected qubit and one half of a distributed Bell pair, we establish logical entanglement between distant nodes. Due to its deterministic nature, this scheme functions with one Bell pair at minimum, and with further modifications to the protocol, we can adjust our scheme to consume multiple Bell pairs at once.

To evaluate the behavior of our protocols, I perform numerical simulations. I define the “achievable” logical error rate as a comparative metric to account for the disparity in physical Bell pair consumption across protocols. This metric incorporates entanglement distillation [11] as a post-processing step, allowing us to suppress the logical error at the cost of consuming additional logical entanglement.

Under this metric, I find that entanglement ejection demonstrates superior performance across a wide variety of parameter regimes, especially for low-fidelity input Bell pairs. Even for high-fidelity input Bell pairs, ejection remained superior against remote lattice surgery as long as the circuit noise was not too weak. Such strong results suggest that entanglement ejection can serve as an effective primitive for fault-tolerant distributed quantum computation, especially for near-term quantum devices.

1.3 Thesis Structure

This thesis is organized as follows. In Chapter 2, I provide the preliminary background on quantum theory, quantum computing, and quantum communication. In Chapter 3, I introduce the topics of error correction and fault tolerance in quantum computers. In Chapter 4, I define the problem within the context of distributed quantum computing, review related prior work, and describe the proposed method. Chapter 5 describes the methodology for evaluation, while Chapter 6 presents the results and discussion, including potential research questions originating from this work. This thesis concludes with Chapter 7, which summarizes the overall contributions and findings.

Chapter 2

Preliminaries

This chapter outlines the theoretical foundations of the thesis, beginning with the fundamental principles of quantum theory. I then introduce the concept of quantum computation and examine specific algorithms that demonstrate computational advantages over classical methods. This chapter concludes with a discussion on quantum communication and its applications.

2.1 Quantum Theory

From my perspective, it[quantum mechanics] is about information and probabilities and observables, and how they relate to each other.

– Scott Aaronson

Wrestling with mind-bending differential equations or pondering a quantum mechanical philosophical interpretation of the world often leads to the common misconception that quantum mechanics is a field only for pure geniuses. Viewing quantum theory as the study of abstract objects described by linear algebra, rather than constantly relying on physics analogies, makes the field much more approachable. In this section, I introduce the basic concepts of quantum theory, alongside the necessary mathematical foundations such as (complex) linear algebra.

2.1.1 Mathematical Background

I did claim that quantum theory is not as hard as you think, but of course, we cannot describe quantum without any mathematical tools. Quantum mechanics can be described using a slightly abstract format of linear algebra, specifically with complex numbers. I would first like to introduce some crucial notations for dealing with complex matrices.

(1) Matrices/Operators

I will use capital letters to represent a matrix, or, in a more abstract term, an operator, like A, B, and so on. As we would want a concrete example for explaining the mathematics of linear algebra, let us define A as follows:

$$A = \begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix}, \quad a, b, c, d, e, f \in \mathbb{C}. \tag{2.1}$$

The transpose of A, denoted as A^T, can be written as follows. We simply swap the row index and the column index of the operator.

$$A^T = \begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix}^T = \begin{pmatrix} a & d \\ b & e \\ c & f \end{pmatrix} \tag{2.2}$$

The conjugate transpose of A is denoted as $A^\dagger$, where the symbol † is pronounced as "dagger." In complex linear algebra, this operation corresponds to taking the adjoint of one operator. We take the transpose of the operator and then take the complex conjugate of all elements.

$$A^\dagger = \begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix}^\dagger = \begin{pmatrix} a^* & d^* \\ b^* & e^* \\ c^* & f^* \end{pmatrix} \tag{2.3}$$

An operator that is equivalent to its adjoint form is called **Hermitian**. For any Hermitian operator H, $H = H^\dagger$ holds.

(2) The Hilbert Space

Quantum states are defined as a unit vector within some kind of **Hilbert space** $\mathcal{H}$. A Hilbert space is a complete inner product vector space over $\mathbb{F}$. However, as far as we are dealing with quantum theory used within this thesis, there is no need to worry about completeness or any other kind of properties related to infinite dimensions. Within this thesis, it is sufficient to interpret a "Hilbert space" as a d-dimensional complex vector space $\mathbb{C}^d$.

How do we actually represent such kinds of vectors? Of course, we can write them in the usual way we do in any other field dealing with linear algebra, such as lining up the elements in a parenthesis as a column, like $\begin{pmatrix} a \\ b \\ c \end{pmatrix}$, or in a row, $\begin{pmatrix} a & b & c \end{pmatrix}$, according to our preference. With this, we can represent an element in $\mathbb{C}^3$ for $a, b, c \in \mathbb{C}$.

(3) The Bra-ket Notation

However, writing out every single element in this format becomes tedious, especially in higher dimensions. Such situations are very common in dealing with quantum theory, and therefore, we have the **bra-ket notation**. The bra-ket notation, also known as the **Dirac notation**, is a specific formalism that is used to represent vectors defined in a Hilbert space. The bra-ket notation is formalized as follows.

Symbol	**Mathematical meaning**
$\|\psi\rangle$	Represents a column vector in $\mathcal{H}$. Pronounced as "ket-psi."
$\langle\psi\|$	A row vector in $\mathcal{H}^*$, obtained via $(\|\psi\rangle)^\dagger$. Pronounced as "bra-psi."
$\langle\psi\|\phi\rangle$	Inner product between the vectors $\|\psi\rangle$ and $\|\phi\rangle$.
$\|\psi\rangle\langle\phi\|$	Outer product between the vectors $\|\psi\rangle$ and $\|\phi\rangle$.
$\|\psi\rangle\otimes\|\phi\rangle$	Tensor product of $\|\psi\rangle$ and $\|\phi\rangle$.

Table 2.1: Representing linear algebra operations using the bra-ket notation.

Let's look at some concrete examples where we choose some specific elements from a two-dimensional Hilbert space $\mathcal{H}$. Let $|\psi\rangle = \begin{pmatrix} a \\ b \end{pmatrix}$ where $a, b \in \mathbb{C}$. Here, $\langle\psi|$ can be obtained as follows:

$$\langle\psi| = (|\psi\rangle)^\dagger = \begin{pmatrix} a \\ b \end{pmatrix}^\dagger = \begin{pmatrix} a^* & b^* \end{pmatrix}. \tag{2.4}$$

Pick another vector $|\phi\rangle = \begin{pmatrix} c \\ d \end{pmatrix}$ where $c, d \in \mathbb{C}$. The **inner product** between $|\psi\rangle$ and $|\phi\rangle$ can be obtained as

$$\langle\psi|\phi\rangle = \begin{pmatrix} a^* & b^* \end{pmatrix} \begin{pmatrix} c \\ d \end{pmatrix} = a^*c + b^*d, \tag{2.5}$$

and the **outer product** between $|\psi\rangle$ and $|\phi\rangle$ as

$$|\psi\rangle\langle\phi| = \begin{pmatrix} a \\ b \end{pmatrix} \begin{pmatrix} c^* & d^* \end{pmatrix} = \begin{pmatrix} ac^* & ad^* \\ bc^* & bd^* \end{pmatrix}. \tag{2.6}$$

From this, we can notice that the *inner product gives us a complex number*, and the *outer product gives us a linear operator.* Following the bra-ket notation allows us to reduce inner products and outer products to a simple matrix multiplication problem, which helps a lot in describing various aspects of quantum theory.

The **tensor product** between $|\psi\rangle$ and $|\phi\rangle$ can be calculated in the vector form as follows:

$$|\psi\rangle \otimes |\phi\rangle = \begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} a\begin{pmatrix} c \\ d \end{pmatrix} \\ b\begin{pmatrix} c \\ d \end{pmatrix} \end{pmatrix} = \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix}. \tag{2.7}$$

As we can see, taking a tensor product increases the dimension of the Hilbert space. For $\mathcal{H}_A$ and $\mathcal{H}_B$, the dimension of the tensor product space taken between these two spaces is $\dim \mathcal{H}_A \otimes \mathcal{H}_B = \dim \mathcal{H}_A \dim \mathcal{H}_B$.

(4) Basis of the Hilbert Space

Using the bra-ket notation, we can describe more advanced mathematical concepts in linear algebra with simple terms. First, we introduce the concept of a *basis*. A basis is a set of vectors that can represent any arbitrary vector via a linear combination. For example, any arbitrary vector in the two-dimensional Hilbert space $|\psi\rangle \in \mathcal{H}^2$ can be decomposed into a linear combination as

$$|\psi\rangle = \begin{pmatrix} a \\ b \end{pmatrix} = a \begin{pmatrix} 1 \\ 0 \end{pmatrix} + b \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{2.8}$$

Alternatively, we can choose a different set of basis vectors. For example, we can define another basis set for the same two-dimensional Hilbert space as

$$\left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix} \right\}. \tag{2.9}$$

Using this set, $|\psi\rangle$ can be represented as

$$|\psi\rangle = \begin{pmatrix} a \\ b \end{pmatrix} = \frac{a+b}{2} \begin{pmatrix} 1 \\ 1 \end{pmatrix} + \frac{a-b}{2} \begin{pmatrix} 1 \\ -1 \end{pmatrix}. \tag{2.10}$$

For a d-dimensional Hilbert space, as long as we have d linearly independent vectors, those vectors constitute a valid basis.

If we refer to an **orthogonal** basis, it means that all vectors in the basis set are orthogonal to each other (i.e., the inner products of different elements are 0). If we refer to a **normalized** basis, it means that the inner product of a vector with itself results in 1. Typically, we choose an **orthonormal** basis (which is both orthogonal and normalized) due to its mathematical simplicity. Moreover, even if we are given a basis set that is neither orthogonal nor normal, we can easily reconstruct an orthonormal basis from those vectors using a technique known as Gram-Schmidt orthogonalization.

We typically denote an orthonormal basis set for a d-dimensional Hilbert space as $\{|i\rangle\}_{i=1}^{d}$. For arbitrary elements $|i\rangle$ and $|j\rangle$ in this basis set, the following relation holds[1].

$$\langle i|j\rangle = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{2.11}$$

One important relation between operators and an orthonormal basis is the **completeness relation**, as shown as follows:

$$I = \sum_{i=1}^{d} |i\rangle \langle i|. \tag{2.12}$$

[1]δ_{ij} is a function known as the Kronecker delta.

This can be verified by the following derivation. Let an arbitrary vector be represented by a linear combination $|\psi\rangle = \sum_i c_i |i\rangle$. Then, applying $\sum_i |i\rangle \langle i|$ yields the following.

$$\left(\sum_{i=1}^{d} |i\rangle \langle i| \right) |\psi\rangle = \sum_{i=1}^{d} |i\rangle \langle i| \sum_{j=1}^{d} c_j |j\rangle \tag{2.13}$$

$$= \sum_{i,j=1}^{d} c_j |i\rangle \langle i|j\rangle \tag{2.14}$$

$$= \sum_{i=1}^{d} c_i |i\rangle = |\psi\rangle \tag{2.15}$$

Here, we used the property of δ_{ij} when calculating $\langle i|j\rangle$, removing elements such that $i \neq j$.

(5) Eigenvalues and Eigenvectors

Given some operator A, we can think of a vector $|\psi_i\rangle$ where the following equation holds.

$$A |\psi\rangle = \lambda |\psi\rangle, \quad \lambda \in \mathbb{C} \tag{2.16}$$

We call $|\psi\rangle$ as the **eigenvector**, and λ as the **eigenvalue** of A. For operators that have the property of $AA^\dagger = A^\dagger A$[2], there exists a **diagonal representation** that allows decomposing the operator as follows:

$$A = \sum_i \lambda_i |i\rangle \langle i| . \tag{2.17}$$

Here, the set $\{|i\rangle\}_i$ forms an orthonormal set of eigenvectors with eigenvalues $\{\lambda_i\}_i$.

(6) Projection Operators

Consider a d-dimensional Hilbert space $\mathcal{H}$ and a k-dimensional subspace $\mathcal{V} \subseteq \mathcal{H}$ where $k \leq d$. Then, with the Gram-Schmidt orthogonalization, we can construct an orthonormal basis for $\mathcal{H}$ as $\{|i\rangle\}_{i=1}^{d}$ and for $\mathcal{V}$ as $\{|i\rangle\}_{i=1}^{k}$. The **projection operator** (or the **projector**) P that forcefully projects a vector in $\mathcal{H}$ to $\mathcal{V}$ is given as follows:

$$P = \sum_{i}^{k} |i\rangle \langle i| . \tag{2.18}$$

[2]We call such A as **normal** operators. Hermitian operators are a special class of normal operators.

The action of P on some vector $|\psi\rangle \in \mathcal{H}$, where $|\psi\rangle = \sum_i^d c_i |i\rangle$, is as follows:

$$P|\psi\rangle = \sum_i^k |i\rangle\langle i| \sum_j^d c_j |j\rangle, \tag{2.19}$$

$$= \sum_i^k \sum_j^d c_j |i\rangle \langle i|j\rangle, \tag{2.20}$$

$$= \sum_i^k c_i |i\rangle\langle i|. \tag{2.21}$$

Therefore, the projector can be viewed as deleting orthonormal basis elements that exist in $\mathcal{H}$ but not in $\mathcal{V}$. We can easily confirm P is Hermitian, $P^\dagger = P$. Moreover, $P^2 = P$, as

$$P^2 = \sum_i^k |i\rangle\langle i| \sum_j^k |j\rangle\langle j| \tag{2.22}$$

$$= \sum_{i,j}^k |i\rangle\langle i|j\rangle\langle j|, \tag{2.23}$$

$$= \sum_i^k |i\rangle\langle j| = P. \tag{2.24}$$

(7) The Trace and the Partial Trace

The **trace** of an operator is defined as the sum of all elements along the diagonal. For an operator in the d-dimensional Hilbert space $\mathcal{H}$ with an orthonormal basis $\{|i\rangle\}_{i=1}^d$, the trace can be written as:

$$\mathrm{tr}(A) = \sum_{i=1}^d A_{ii}, \tag{2.25}$$

$$= \sum_{i=1}^d \langle i|A|i\rangle. \tag{2.26}$$

The concept of a trace can be easily generalized to taking the trace on a subspace. For example, consider that A is an operator defined on the composite Hilbert space $\mathcal{H}_A \otimes \mathcal{H}_B$. Let the orthonormal basis of $\mathcal{H}_B$ be $\{|i\rangle_B\}_{i=1}^{\dim(\mathcal{H}_B)}$. Then, the **partial trace** of A over $\mathcal{H}_B$ can be written as follows:

$$\mathrm{tr}_{\mathcal{H}_B}(A) = \sum_{i=1}^{\dim(\mathcal{H}_B)} \langle i|_B A |i\rangle_B. \tag{2.27}$$

An interesting application of the trace is determining the dimension of a subspace given its projector. Given a projector P, the dimension of the projected subspace can be calculated as $k = \mathrm{tr}(P)$. This is demonstrated as follows. Since P is a projector, there necessarily exists an orthonormal basis $\{|i\rangle\}_{i=1}^{d}$ for the full space such that the first k vectors span the projected subspace. Note that we do not need to explicitly find this basis or know k beforehand to perform the calculation; we simply rely on the fact that, mathematically, P can be written as $P = \sum_{i=1}^{k} |i\rangle \langle i|$. Taking the trace of P over the full d-dimensional space yields the following.

$$\begin{aligned}
\mathrm{tr}(P) &= \mathrm{tr}\left(\sum_{i=1}^{k} |i\rangle \langle i|\right) && (2.28)\\
&= \sum_{j=1}^{d} \langle j| \left(\sum_{i=1}^{k} |i\rangle \langle i|\right) |j\rangle && (2.29)\\
&= \sum_{j=1}^{d} \sum_{i=1}^{k} \langle j|i\rangle \langle i|j\rangle && (2.30)\\
&= \sum_{i=1}^{k} \langle i|i\rangle^2 = \sum_{i=1}^{k} 1 = k && (2.31)
\end{aligned}$$

From this, the dimension of the subspace can be directly obtained from the trace of P, without prior knowledge of the basis or k.

(8) The Commutator and the Anti-Commutator

The **commutator** between two operators A and B is defined as follows:

$$[A, B] = AB - BA. \quad (2.32)$$

When $[A, B] = 0$, we typically refer to A and B that they commute with each other. This is because $AB = BA$ holds, which is not a general property for all operators. The **anti-commutator** between A and B is given as:

$$\{A, B\} = AB + BA. \quad (2.33)$$

As like commutators, we claim that A and B anti-commute with each other when $\{A, B\} = 0$.

There is an interesting property that connects the commutativity of operators to the diagonal representation of operators. Provided that A and B are Hermitian operators, A and B commute if and only if A and B are **simultaneously diagonalizable**. This means that A and B have a diagonal representation regarding the same orthonormal vectors $\{|i\rangle\}$, as:

$$A = \sum_i a_i |i\rangle \langle i|, \quad B = \sum_i b_i |i\rangle \langle i|. \quad (2.34)$$

(9) Operator Functions

Operator functions take operators instead of scalars as inputs. For generic operators, finding the function value might be difficult. However, for operators that have a diagonal representation, an operator function can be easily defined. Let a normal operator A have a diagonal representation $A = \sum_i \lambda_i |i\rangle\langle i|$. Then, for a function $f(A)$, the resulting function can be written as:

$$f(A) = \sum_i f(\lambda_i) |i\rangle\langle i| . \tag{2.35}$$

In short, applying a function on the eigenvalues corresponds to applying a function on the entire operator.

Such operator functions have entirely different features from scalar functions. For example, consider the matrix exponential function $f(A) = e^A$. This can be expressed as:

$$f(A) = \sum_i e^{\lambda_i} |i\rangle\langle i| . \tag{2.36}$$

Recall that for real numbers $x, y \in \mathbb{R}$, the identity $e^x e^y = e^{x+y}$ holds. This implies commutativity, as $e^x e^y = e^{x+y} = e^{y+x} = e^y e^x$. However, when generalizing to operators, $e^A e^B$ is not necessarily equivalent to e^{A+B}. This is because multiplication is generally non-commutative, even though addition is $(A + B = B + A)$.

Now, consider the case where $[A, B] = 0$. By the simultaneous diagonalization theorem, A and B share a common eigenbasis $\{|i\rangle\}_i$ such that $A = \sum_i a_i |i\rangle\langle i|$ and $B = \sum_i b_i |i\rangle\langle i|$. Then, the value $e^A e^B$ can be calculated as follows:

$$e^A e^B = \sum_i e^{a_i} |i\rangle\langle i| \sum_j e^{b_j} |j\rangle\langle j| , \tag{2.37}$$

$$= \sum_{i,j} e^{a_i} e^{b_j} |i\rangle\langle i|j\rangle\langle j| , \tag{2.38}$$

$$= \sum_i e^{a_i + b_i} |i\rangle\langle i| = e^{A+B}. \tag{2.39}$$

Thus, when A and B commute with each other, the identity $e^A e^B = e^{A+B}$ holds.

2.1.2 Axioms of Quantum Theory

We can frame quantum mechanics as an axiomatic mathematical theory. Here, I formalize quantum theory by laying out four axioms. Building on this, I will expand some of our axioms to noisy states, providing a complete description of quantum theory. Specifically, this thesis follows the descriptions of axioms introduced by Nakada [12], with slight modifications.

(1) Quantum States

Mathematically, every quantum state is given as an element in the Hilbert space.

Axiom 2.1 (*Quantum State*)

For any physical system, there exists a Hilbert space $\mathcal{H}$ which is called the **state space**. *A unit vector $|\psi\rangle \in \mathcal{H}$ gives a* **quantum state** *in that system.*

This means that a unit vector in a Hilbert space $\mathcal{H}$ corresponds to some particular quantum state. However, quantum states have a freedom of **global phase**, which means that the equivalence relation $|\psi\rangle \sim e^{i\phi}|\psi\rangle, \forall |\psi\rangle \in \mathcal{H}$ holds. This is because such global phases have *no observable effects* on our quantum state.

As an example, we can think of a two-dimensional Hilbert space $\mathcal{H}$ as our state space, and define an arbitrary quantum state $|\psi\rangle$ as follows. Here, the set $\{|0\rangle, |1\rangle\}$ serves as the orthonormal basis for $\mathcal{H}$.

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \text{ where } |0\rangle = \begin{pmatrix}1\\0\end{pmatrix}, |1\rangle = \begin{pmatrix}0\\1\end{pmatrix}, \text{ and } |\alpha|^2 + |\beta|^2 = 1 \tag{2.40}$$

If α and β were real numbers, a geometric picture of our quantum state can be represented as an arbitrary point on the unit circle. However, as we are dealing with complex numbers here, this unit vector in a two-dimensional Hilbert space can be rewritten as follows.

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle \tag{2.41}$$

Plotting this point in the spherical coordinate system (where the radius is 1) $(1, \theta, \phi)$, we obtain the following geometric representation shown in Fig. 2.1, known as the **Bloch sphere**. We can observe from this that a quantum state is in some sense a continuous state, where any single point on the surface of the sphere is a valid physical state.

Such elements in the two-dimensional Hilbert spaces are typically referred to as **qubits**. I will explain this notion in detail in future sections where I introduce quantum computing, but for now, just consider this to be a substitute for calling *an element in the two-dimensional Hilbert space.*

(2) Time Evolution

The evolution of a quantum state is known to be governed by the Schrödinger equation, a differential equation written in the following form. Here, $\hat{H}$ is the Hamiltonian of the system, a Hermitian operator that has a complete description of some particular quantum state.

$$i\hbar\frac{d}{dt}|\psi(t)\rangle = \hat{H}|\psi(t)\rangle \tag{2.42}$$

By solving this equation, you can notice that the time evolution of a quantum state can be simplified, provided that the Hamiltonian is independent of time.

$$|\psi(t)\rangle = e^{-i\hat{H}t}|\psi(0)\rangle \tag{2.43}$$

As $\hat{H}$ is Hermitian, $e^{-i\hat{H}t}$ has a property that is called **unitary**; this is the property where the product of this operator and its adjoint form gives us an identity operator. In other

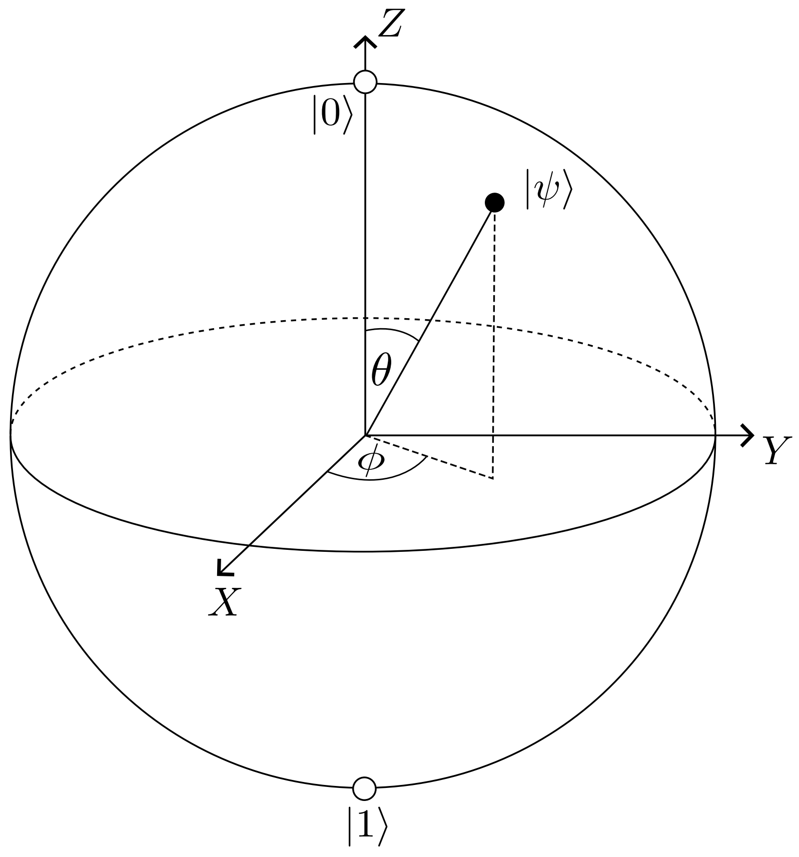


Figure 2.1: Bloch sphere representation of a qubit, an arbitrary quantum state in the two-dimensional Hilbert space. The X, Y, and Z axes on the sphere correspond to quantum states that are the eigenvectors of their respective Pauli matrices. The direction pointed to by an arrow signifies the $+1$ eigenvector, with the opposite side representing the -1 eigenvector.

words, any unitary matrix U has the following property.

$$UU^\dagger = U^\dagger U = I \tag{2.44}$$

We can notice that $e^{-i\hat{H}t}$ is indeed a unitary operator, as the following equation holds.

$$\begin{aligned} UU^\dagger &= (e^{-i\hat{H}t})(e^{-i\hat{H}t})^\dagger && (2.45)\\ &= e^{-i\hat{H}t}e^{i\hat{H}t} && (2.46)\\ &= e^{-i\hat{H}t+i\hat{H}t} \quad \because [\hat{H},\hat{H}] = 0 && (2.47)\\ &= e^0 = I && (2.48) \end{aligned}$$

Therefore, we can frame the axiom of time evolution as follows.

Axiom 2.2 (*Time Evolution*)

There exists a process that transforms a quantum state $|\psi\rangle$ to $U|\psi\rangle$ where U is a ***unitary*** *that operates on the Hilbert space $\mathcal{H}$.*

A famous class of quantum time evolutions are the Pauli matrices. They can be expressed in the following forms.

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.49}$$

These matrices are all unitary operators, hence satisfying the axiom of time evolution. Let's see the transformations on our previously introduced qubit $|\psi\rangle$.

$$I|\psi\rangle = I(\alpha|0\rangle + \beta|1\rangle) = \alpha|0\rangle + \beta|1\rangle \tag{2.50}$$

$$X|\psi\rangle = X(\alpha|0\rangle + \beta|1\rangle) = \alpha|1\rangle + \beta|0\rangle \tag{2.51}$$

$$Y|\psi\rangle = Y(\alpha|0\rangle + \beta|1\rangle) = \alpha|1\rangle - \beta|0\rangle \tag{2.52}$$

$$Z|\psi\rangle = Z(\alpha|0\rangle + \beta|1\rangle) = \alpha|0\rangle - \beta|1\rangle \tag{2.53}$$

I have ignored the global phase i for the action of the Y operator. In the following section where I introduce quantum computing, we will look into specific unitary matrices that allow us to use such quantum states as a computational resource.

(3) Quantum Measurements

To obtain any information from the quantum world to the classical world in which we live, there is a need to perform *quantum measurements.* It is usually misconceived that a measurement is something very theoretical, only existing in the level of abstract mathematics. But it is not something new to any single person. Say you would like to measure the length of your height or weigh how much mass an object has. In such situations, we are measuring a **physical observable**. The same principle applies to quantum theory as well. We measure **quantum mechanical observables** of some particular quantum state, such as whether the *spin* along the Z axis is "up" or "down," or what *energy level* an atom is in.

Axiom 2.3 (*Quantum Measurement*)

Let a quantum state space be defined on a d-dimensional Hilbert space $\mathcal{H}$. For an orthonormal basis $\{|i\rangle\}_{i=1}^{d}$ of $\mathcal{H}$ and a quantum state $|\psi\rangle$, there exists a measurement that returns the result i and changes the post-measurement state to $|i\rangle$ with probability $p_i = |\langle i|\psi\rangle|^2$.

This axiom might be a little difficult to understand straightforwardly, so let us take our qubit state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ as an example again. The set of vectors $\{|0\rangle, |1\rangle\}$ forms an orthonormal basis, so there exists a quantum measurement that satisfies Tab. 2.2.

How does this connect to the notion of observables? In fact, for quantum mechanics, an observable is given as some Hermitian operator H. As all Hermitian operators have diagonal representations, measuring a quantum mechanical observable corresponds to performing quantum measurement regarding the eigenvectors of H. For example, the

Result	Probability	Post-measurement state
0	$p_0 = \lvert \langle 0\vert\psi\rangle \rvert^2 = \lvert\alpha\rvert^2$	$\lvert 0\rangle$
1	$p_1 = \lvert \langle 1\vert\psi\rangle \rvert^2 = \lvert\beta\rvert^2$	$\lvert 1\rangle$

Table 2.2: Measuring $|\psi\rangle$ in the orthonormal basis $\{|0\rangle, |1\rangle\}$.

measurement $\{|0\rangle, |1\rangle\}$ corresponds to measuring the observable Z, as the Pauli Z matrix has the diagonal representation of:

$$Z = |0\rangle\langle 0| - |1\rangle\langle 1|. \tag{2.54}$$

On the other hand, measuring the observable X corresponds to a measurement in the orthonormal basis of $\{(|0\rangle + |1\rangle)/\sqrt{2}, (|0\rangle - |1\rangle)/\sqrt{2}\}$, as the Pauli X matrix has the diagonal representation of:

$$Z = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)\left(\frac{\langle 0| + \langle 1|}{\sqrt{2}}\right) - \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)\left(\frac{\langle 0| - \langle 1|}{\sqrt{2}}\right). \tag{2.55}$$

With this axiom, we can understand that *quantum measurement changes the post-measurement state.* This is one of the key features of quantum theory, typically called the *collapse of the wave function.* As an example, consider we measure the observable Z. If we obtain 0 as the measurement result, this projects the state to $|0\rangle$, where further measurements will also return 0, regardless of the input state.

Moreover, quantum theory only tells us the *probability distributions* of the measurement results, not their exact values. A single shot of measurement does not provide sufficient information about our initial state[3]. What we know from this axiom is that the measurement result follows a certain probability distribution determined by its initial state.

Such features are nothing like what we know from classical mechanics. Due to this peculiarity, there are many wild speculations around quantum theory, which typically dive into "deep and philosophical" debates around the interpretation of what a quantum measurement actually is. However, just by admitting that the measurement results give us a probability distribution instead of the exact value of the physical observable, the theory coincides with every kind of physical experiment that has been conducted in the past, implying that we should adhere to the current form of quantum theory. There is nothing more to be said beyond that, and in the end, whether you believe in the Copenhagen interpretation or the multiverse interpretation does not matter at all.

(4) Composite Systems

Composite systems, where we think of a combination between two or more different state spaces, can be mathematically described with the **tensor products** of vectors.

[3]However, now that we have changed the post-measurement state according to the measurement result, we have full knowledge about the current state.

Axiom 2.4 (*Composite Systems*)

Given two Hilbert spaces $\mathcal{H}_A$ and $\mathcal{H}_B$ where $|\psi\rangle \in \mathcal{H}_A$ and $|\phi\rangle \in \mathcal{H}_B$, the composite system of quantum states $|\psi\rangle$ and $|\phi\rangle$ is given by $|\psi\rangle \otimes |\phi\rangle$.

One prominent example in quantum theory that also causes debates, where even the renowned physicist Albert Einstein disagreed with [13], is about the property of **quantum entanglement**. This phenomenon can be described with the axiom of measurement and composite systems. Think of a composite system between $\mathcal{H}_A$ and $\mathcal{H}_B$, $\mathcal{H}_A \otimes \mathcal{H}_B$, where both are qubit spaces. Let the state $|\Phi^+\rangle$ be defined as follows.

$$|\Phi^+\rangle \equiv \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) \tag{2.56}$$

This is indeed a valid state in $\mathcal{H}_A \otimes \mathcal{H}_B$. Let's say we measure this state with the orthonormal basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. The quantum measurement can be given as shown in Tab 2.3. Therefore, if we measure the state space corresponding to $\mathcal{H}_B$, we

Result	Probability	Post-measurement state
0	$p_0 = \lvert\langle 00\vert\Phi^+\rangle\rvert^2 = 1/2$	$\vert 00\rangle$
1	$p_1 = \lvert\langle 01\vert\Phi^+\rangle\rvert^2 = 0$	$\vert 01\rangle$
2	$p_2 = \lvert\langle 10\vert\Phi^+\rangle\rvert^2 = 0$	$\vert 10\rangle$
3	$p_3 = \lvert\langle 11\vert\Phi^+\rangle\rvert^2 = 1/2$	$\vert 11\rangle$

Table 2.3: Measuring $|\Phi^+\rangle$ in the orthonormal basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$.

always obtain the same measurement result that was obtained in measuring the state space of $\mathcal{H}_A$. *This indicates a strong correlation between each other's quantum state!!* Even if these two qubits are physically far away, *the measurement results between them always coincide.* As long as we are limited to classical mechanics, such a phenomenon can never occur.

In fact, telling whether a state is entangled does not require knowing the measurement properties of that particular state. The determining factor is whether the state *can be written in tensor products* of its individual subsystems. For example, the state $|00\rangle$ can be written as the tensor products of $|0\rangle$ and $|0\rangle$, so it's not entangled. However, the state $|\Phi^+\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$ cannot be decomposed into a tensor product of two subsystem states, meaning $|\Phi^+\rangle \neq |\psi\rangle \otimes |\phi\rangle$. When a quantum state has this property, we say that it is **entangled**.

In particular, for a composite system $\mathcal{H}_A \otimes \mathcal{H}_B$ where $\mathcal{H}_A$ and $\mathcal{H}_B$ are both qubit spaces, there are four well-known entangled states, which are called **Bell pairs**.

$$|\Phi^+\rangle \equiv \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{2.57}$$

$$|\Phi^-\rangle \equiv \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \tag{2.58}$$

$$|\Psi^+\rangle \equiv \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \tag{2.59}$$

$$|\Psi^-\rangle \equiv \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \tag{2.60}$$

The set $\{|\Phi^+\rangle, |\Phi^-\rangle, |\Psi^+\rangle, |\Psi^-\rangle\}$ forms a complete orthonormal basis in this state space. Therefore, it is possible to perform quantum measurement in this basis. Such measurements are called Bell state measurements. I will introduce more properties of the Bell pairs in later sections.

2.1.3 Axioms of Noisy Quantum Theory

Have I completely described quantum theory? It turns out that the answer is no, because I omitted the description of how **noise** affects our system. Noiseless quantum states, which I have previously described, are called **pure states**.

In this section, I will describe noisy, or **mixed states**, by expanding on some axioms from our previous section. Specifically, by slightly altering the axioms for 1. quantum states, 2. time evolution, and 3. quantum measurements, we can obtain a complete quantum theory with the presence of noise.

(1) Noisy Quantum States

A noisy quantum state can be described as a classical probabilistic ensemble of several quantum states. We represent it in the following format, $\{p_i, |\psi_i\rangle\}_i$. This means that with probability p_i, we have the state $|\psi_i\rangle$.

It does seem like simply writing those states as a superposition state $|\psi\rangle = \sum_i \sqrt{p_i}\,|\psi_i\rangle$ gives us the probabilistic mixture. However, this is completely untrue, and, for some ensembles, this $|\psi\rangle$ might be an invalid quantum state. What we have written is a **quantum superposition**, not a **classical mixture**. To represent such mixtures, we need a different mathematical framework from vectors. Indeed, we can represent a mixed quantum state as **linear operators**. We write this state as ρ, which is defined as follows:

$$\rho = \sum_i p_i \,|\psi_i\rangle\langle\psi_i|\,. \tag{2.61}$$

What kind of properties does this mixed state have? As we are dealing with probabilities, for any i, $p_i \geq 0$, and for the set $\{p_i\}_i$, $\sum_i p_i = 1$. Such properties can be translated into linear algebraic expressions on ρ.

1. For any vector $|\psi\rangle \in \mathcal{H}$, $\langle\psi|\rho|\psi\rangle = \sum_i \left(p_i \left|\langle\psi|\psi_i\rangle\right|^2\right) \geq 0$, so ρ is a positive semi-definite operator. We can write this as $\rho \geq 0$.
2. $\mathrm{tr}(\rho) = \sum_i p_i \mathrm{tr}\left(|\psi_i\rangle\langle\psi_i|\right) = \sum_i p_i = 1$, so a density operator is a unit trace operator.

With these properties, the axiom of a noisy quantum state is as follows.

Axiom 2.5 (*Noisy Quantum State*)

*For any physical system, there exists a Hilbert space $\mathcal{H}$ which is called the state space. A quantum state of the system can be given in the form of a **density operator** ρ on $\mathcal{H}$, which satisfies the following properties.*

$$\rho \geq 0, \quad \mathrm{tr}(\rho) = 1 \tag{2.62}$$

We denote $\mathcal{M}(\mathcal{H})$ as the set of density operators on the Hilbert space $\mathcal{H}$.

(2) Noisy Time Evolution

For pure states, unitary operators change quantum states. This is generally the same for mixed states, as we can represent the change of state ρ under a unitary U as follows.

$$\rho \mapsto U\rho U^\dagger \tag{2.63}$$

If we only consider unitary operations, this is enough. However, not all operations in the world are unitary. Even if we have the highest-grade equipment, what we can produce in the lab is still a probabilistic ensemble of quantum states. In the mixed state formalism, we move one step away from unitary matrices and introduce the following axiom to describe time evolution including quantum noise.

Axiom 2.6 (*Noisy Time Evolution*)

*The time evolution of a quantum state ρ is given as a **quantum channel** $\mathcal{E}$, which mathematically represents a linear **CPTP map**. A linear map $\mathcal{E} : \mathcal{M}(\mathcal{H}_{in}) \mapsto \mathcal{M}(\mathcal{H}_{out})$ is a CPTP map if it satisfies the following properties:*

1. ***Completely Positive**: For any auxiliary Hilbert space $\mathcal{H}_R$ and for any density operator $\rho_{R,in} \in \mathcal{M}(\mathcal{H}_R \otimes \mathcal{H}_{in})$,*

$$(Id_R \otimes \mathcal{E})(\rho_{R,in}) \geq 0 \tag{2.64}$$

2. ***Trace Preserving**: For any density operator $\rho \in \mathcal{M}(\mathcal{H}_{in})$,*

$$tr(\mathcal{E}(\rho)) = tr(\rho) = 1 \tag{2.65}$$

Here, Id_R denotes the identity map on the auxiliary Hilbert space $\mathcal{H}_R$. Such properties ensure that $\mathcal{E}$ maps any valid density operator to another valid density operator, representing the most general physical operation possible on a quantum system.

Any CPTP map $\mathcal{E}$ can be represented by a set of **Kraus operators**, $\{K_i\}_i$ as:

$$\rho \mapsto \mathcal{E}(\rho) = \sum_i K_i \rho K_i^\dagger. \tag{2.66}$$

Kraus operators are linear operators that satisfy the completeness relation:

$$\sum_i K_i K_i^\dagger = I. \tag{2.67}$$

We adopt the Kraus operator formalism to describe quantum channels, as it provides a more intuitive and convenient representation than the abstract CPTP map.

An example of a quantum channel on a qubit is the bit-flip channel. This channel is the quantum analog of the classical bit-flip channel, where a qubit space undergoes a Pauli-X flip with probability p. Defined by the Kraus operators $M_0 = \sqrt{1-p}I$ and $M_1 = \sqrt{p}X$, its action is written as follows.

$$\mathcal{E}_{\text{bit}}(\rho) = (1-p)\rho + pX\rho X^\dagger \tag{2.68}$$

On the Bloch sphere, this transformation shrinks the state space towards the invariant X-axis, as shown in Fig. 2.2a. Another well-known example of a quantum channel is the phase-flip channel (or dephasing channel[4]), where the setting is the same for a bit-flip channel, but we replace the X operator with a Z operator. It is defined by the Kraus operators $\{\sqrt{1-p}I, \sqrt{p}Z\}$ and evolves the state as follows.

$$\mathcal{E}_{\text{phase}}(\rho) = (1-p)\rho + pZ\rho Z^\dagger \tag{2.69}$$

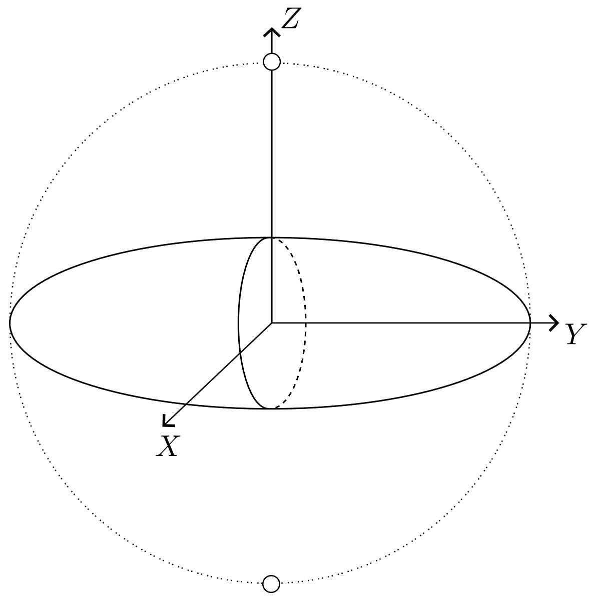
This results in a shrink towards the Z-axis in the Bloch sphere picture as shown in Fig. 2.2b.

The past two channels were an example of a biased noise channel. The depolarizing channel is a good example that represents uniform noise. For a single qubit, the Kraus operators are $\{\sqrt{1-3p}I, \sqrt{p}X, \sqrt{p}Y, \sqrt{p}Z\}$, changing the quantum state as follows.

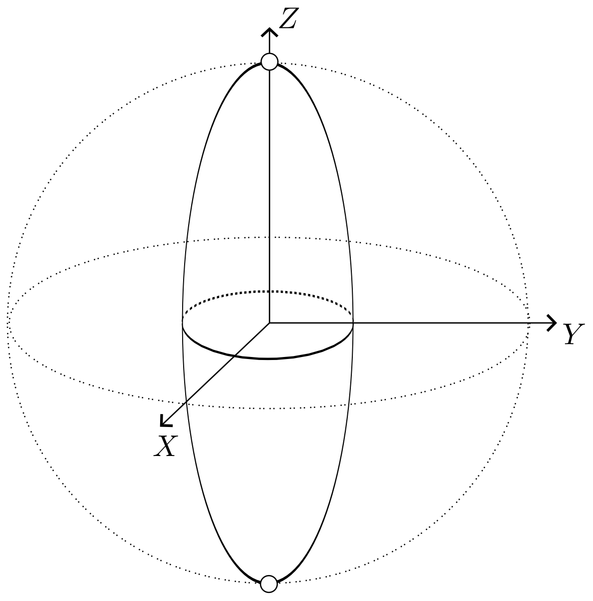
$$\begin{aligned} \rho \mapsto \mathcal{E}_{\text{dep}}(\rho) &= (1-3p)\rho + pX\rho X + pY\rho Y + pZ\rho Z & (2.70) \\ &= \left(1 - \frac{4p}{3}\right)\rho + \frac{4p}{3}\frac{I}{2} & (2.71) \end{aligned}$$

This results in a shrink towards the center of the Bloch sphere, as shown in Fig. 2.3a. The depolarizing channel can easily be generalized to arbitrary dimensions of the Hilbert space, by normalizing the identity operator in equation 2.71. The Pauli channel is a generalization of the error parameters for the depolarizing channel. The Kraus operators of this channel are given as $\{\sqrt{p_I}I, \sqrt{p_X}X, \sqrt{p_Y}Y, \sqrt{p_Z}Z\}$. It changes a quantum state ρ

[4] This name is due to this channel's significance in physics. Dephasing refers to the loss of quantum coherence.

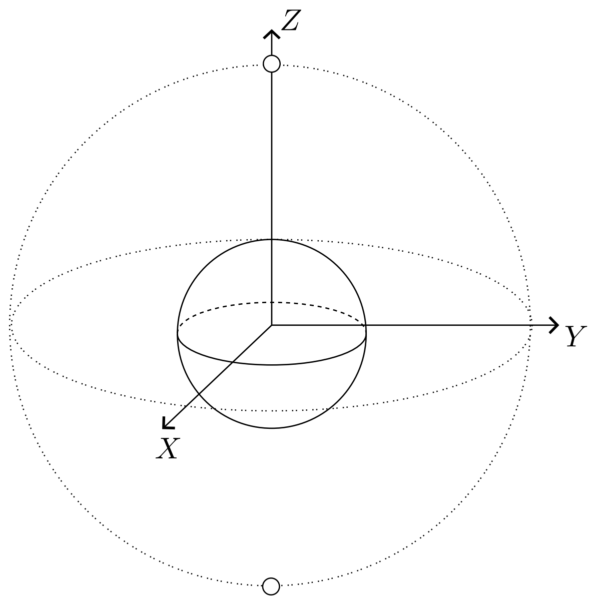


(a) Bit-flip channel

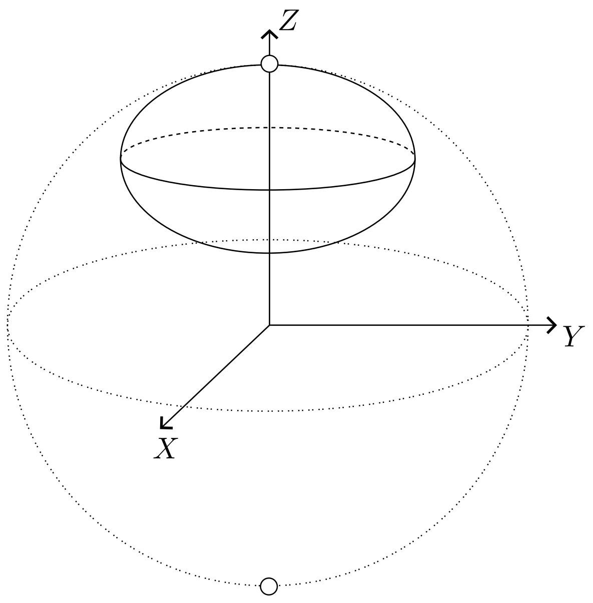


(b) Phase-flip channel

Figure 2.2: Visualization of the bit/phase-flip channels on the Bloch sphere.



(a) Depolarizing channel



(b) Amplitude damping channel

Figure 2.3: Visualization of the depolarizing/amplitude damping channels on the Bloch sphere.

to a probabilistic mixture where either of the Pauli gates has occurred, as shown in the following equation.

$$\rho \mapsto \mathcal{E}_{\text{Pauli}}(\rho) = p_I\rho + p_X X\rho X + p_Y Y\rho Y + p_Z Z\rho Z \tag{2.72}$$

The amplitude damping channel is a representation of a common type of error that occurs in experimental quantum systems: the excited state probabilistically collapses to the ground state. The Kraus operators for this channel are given as

$$A_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-p} \end{pmatrix}, \quad A_1 = \begin{pmatrix} 0 & \sqrt{p} \\ 0 & 0 \end{pmatrix} \tag{2.73}$$

and the quantum state is changed as

$$\rho \mapsto \mathcal{E}_{\text{damp}}(\rho) = A_0\rho A_0^\dagger + A_1\rho A_1^\dagger. \tag{2.74}$$

In the Bloch sphere, the amplitude damping channel represents a squash towards the upper region where the state $|0\rangle$ is located, as shown in Fig. 2.3b.

So far, the CPTP maps discussed have been limited to operation within the same Hilbert space, mapping a qubit state to another qubit state. However, physical implementation often involves higher-dimensional spaces, as we must consider leakage or loss to the environment. We can model this "lost" state by introducing an auxiliary state $|\perp\rangle$ that is orthogonal to both $|0\rangle$ and $|1\rangle$. This extends the system from a qubit to a **qutrit** (three-level system), where $|\perp\rangle$ corresponds to the third basis vector.

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad |\perp\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{2.75}$$

For such qubit erasure channels, we map a qubit state into a qutrit state with the following Kraus operators.

$$A_0 = \sqrt{1-p}\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad A_1 = \sqrt{p}\begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{pmatrix} A_1 = \sqrt{p}\begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \tag{2.76}$$

The evolution of a qubit density matrix ρ under this channel is given by

$$\rho \mapsto \mathcal{E}_{\text{loss}}(\rho) = A_0\rho A_0^\dagger + A_1\rho A_1^\dagger + A_2\rho A_2^\dagger = (1-p)\rho + p\,|\perp\rangle\langle\perp|\,. \tag{2.77}$$

(3) Noisy Quantum Measurements

Performing quantum measurements on mixed states might sound complicated. However, the same principle applies to our case of dealing with pure states. Just keep in mind that quantum measurement gives us a probability distribution of some physical observable and changes the post-measurement state according to the measured result.

Axiom 2.7 (*Noisy Quantum Measurement*)

For a quantum state ρ, the probabilistic change of quantum states caused by measurements can be written with Kraus operators $\{K_i\}_i$. Specifically, we obtain a result i with probability $p_i = \mathrm{tr}(K_i^\dagger K_i \rho)$, where the post-measurement state changes as follows.

$$\rho \mapsto \frac{K_i \rho K_i^\dagger}{p_i} \tag{2.78}$$

If we are not interested in the post-measurement state, quantum measurements can be described with a POVM (Positive Operator-Valued Measure). A POVM is a set of Hermitian operators $\{M_i\}_i$ that satisfy $M_i \geq 0$ and $\sum_i M_i = I$. With these operators, we obtain a result i with probability $p_i = \mathrm{tr}(M_i \rho)$. Given a Kraus operator, the corresponding POVM is $M_i = K_i^\dagger K_i$.

2.1.4 Properties of Quantum Theory

Here, I list several interesting properties of quantum theory that are used throughout the thesis.

(1) Bell Pairs and Entanglement

Bell pairs have the property of being bipartite **maximally entangled** states. This property can be understood by taking the partial trace of one qubit. The partial trace of the Bell state $|\Phi^+\rangle$ over the subsystem $\mathcal{H}_B$ is calculated as follows:

$$\mathrm{tr}_{\mathcal{H}B}(|\Phi^+\rangle\langle\Phi^+|) = \frac{1}{2}\mathrm{tr}\mathcal{H}_B\left(|00\rangle\langle 00| + |00\rangle\langle 11| + |11\rangle\langle 00| + |11\rangle\langle 11|\right), \tag{2.79}$$

$$= \frac{1}{2}\langle 0|_B(\dots)|0\rangle_B + \frac{1}{2}\langle 1|_B(\dots)|1\rangle_B, \tag{2.80}$$

$$= \frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1| = \frac{I}{2}. \tag{2.81}$$

Taking the partial trace of a quantum system corresponds to erasing all information regarding the traced-out space. Therefore, if we possess only one qubit of a Bell pair, the partial state corresponds to the maximally mixed state. This leads to an interesting phenomenon, where locally, the state appears to be in the extreme of noise, but provided the full description of the global state, it becomes a noiseless, pure state. Such states are called maximally entangled states[5].

[5]For bipartite entanglement, verifying that the partial trace is maximally mixed suffices. However, defining maximally entangled states for multipartite systems is significantly complex, as there are various witnesses to characterize the strength of entanglement [14].

(2) Quantum State Tomography

As previously described, any quantum state can be written in the form of a density operator. However, quantum measurements only provide a single measurement value drawn from some probability distribution, not the exact description of the state itself. Under such circumstances, one might wonder whether it is even possible to find the density operator via measurements.

Fortunately, it is possible to fully characterize a general quantum state by **quantum state tomography**, provided that identical copies of the state can be prepared [15]. This process performs multiple measurements in a set of complementary bases to reconstruct the density matrix. It is important to note, however, that this method scales poorly. In fact, the number of required measurements (and therefore the required copies of the state) grows exponentially with the number of qubits.

(3) Measures of Quantum States

How do we evaluate whether two quantum states are similar or not? Here, we will introduce the notion of **trace distance** and **fidelity** as measures that capture how different or how similar quantum states are.

The trace distance is defined as the following.

Definition 2.1 (*Trace Distance*)

The ***trace distance*** *between two quantum states ρ, σ is defined as follows.*

$$D(\rho,\sigma) = \frac{1}{2}\mathrm{tr}(|\rho-\sigma|) \tag{2.82}$$

Here, $|A| = \sqrt{AA^\dagger}$.

The trace distance satisfies the axioms of distances, which makes it a valid metric for the space of density operators.

We do have another measure to quantify the closeness of quantum states, known as the fidelity. The fidelity is defined as the following.

Definition 2.2 (*Fidelity*)

The ***fidelity*** *between two quantum states ρ, σ is defined as follows.*

$$F(\rho,\sigma) = \left(\mathrm{tr}\sqrt{\rho^{1/2}\sigma\rho^{1/2}}\right)^2 \tag{2.83}$$

It might not be obvious from this definition, but the fidelity is symmetric on its inputs, meaning that $F(\rho,\sigma) = F(\sigma,\rho)$. Moreover, if we limit either one of the inputs to a pure state by considering $\rho = |\psi\rangle\langle\psi|$, we obtain[6]

$$F(|\psi\rangle\langle\psi|, \sigma) = \left(\mathrm{tr}\sqrt{(|\psi\rangle\langle\psi|)^{1/2}\sigma(|\psi\rangle\langle\psi|)^{1/2}}\right)^2 \tag{2.84}$$

$$= \left(\mathrm{tr}\sqrt{|\psi\rangle\langle\psi|\sigma|\psi\rangle\langle\psi|}\right)^2 \tag{2.85}$$

$$= \left(\mathrm{tr}\sqrt{\langle\psi|\sigma|\psi\rangle \times |\psi\rangle\langle\psi|}\right)^2 \tag{2.86}$$

$$= \left(\sqrt{\langle\psi|\sigma|\psi\rangle}\right)^2 = \langle\psi|\sigma|\psi\rangle\,. \tag{2.87}$$

This expression is valuable in an experimental setting where we want to evaluate the quality of state generation. This is because the ideal target state is typically a pure state represented by a state vector, while the experimentally noisy state can be obtained as a density matrix, often reconstructed via quantum state tomography. Therefore, the fidelity between such pure and mixed states serves as a reasonable measure of the accuracy of the target state.

(4) No-Cloning Theorem

Consider a scenario where we would like to use some unitary to copy an arbitrary quantum state to another Hilbert space. This is proven to be impossible [16], formalized by the following theorem.

Theorem 2.1 (*No-Cloning Theorem*)

There does not exist a unitary operator U that can perfectly copy arbitrary quantum states. Specifically, for two non-orthogonal states $|\psi\rangle$ and $|\phi\rangle$, there is no single unitary U such that

$$U(|\psi\rangle \otimes |0\rangle) = |\psi\rangle \otimes |\psi\rangle \tag{2.88}$$

$$U(|\phi\rangle \otimes |0\rangle) = |\phi\rangle \otimes |\phi\rangle \tag{2.89}$$

Proof: We prove this by contradiction. Assume that such a cloning unitary U exists. Then, equations 2.88 and 2.89 hold. Since unitary operators preserve inner products, we evaluate the inner product of the left-hand side and the right-hand side of the equation. The inner product of the LHS is

$$(\langle\phi| \otimes \langle 0|)U^\dagger U(|\psi\rangle \otimes |0\rangle) = \langle\phi|\psi\rangle\langle 0|0\rangle = \langle\phi|\psi\rangle\,. \tag{2.90}$$

The inner product of the RHS is

$$(\langle\phi| \otimes \langle\phi|)(|\psi\rangle \otimes |\psi\rangle) = \langle\phi|\psi\rangle^2\,. \tag{2.91}$$

Equating these results implies $\langle\phi|\psi\rangle = \langle\phi|\psi\rangle^2$, the solutions to which are

$$\langle\phi|\psi\rangle = 0 \text{ or } 1. \tag{2.92}$$

[6]For two pure states $|\psi\rangle$ and $|\phi\rangle$, this relation simplifies to $F(|\psi\rangle, |\phi\rangle) = |\langle\psi|\phi\rangle|^2$.

However, since $|\psi\rangle$ and $|\phi\rangle$ are arbitrary non-orthogonal distinct states, $0 < |\langle\phi|\psi\rangle| < 1$, leading to a contradiction. Therefore, a cloning unitary U does not exist. ■

2.2 Quantum Computing

Quantum information science has developed since the latter half of the 20th century, following the completion of the theoretical framework of quantum theory [17]. In its early days, researchers attempted to extend classical information theory to the quantum regime [18] and derived upper bounds on the accessible information in a quantum system [19]. However, these efforts were primarily treated as a sub-branch of mathematical physics, distinct from the paradigm of practical computing.

The earliest documented use of quantum states as information resources appeared in the context of conjugate coding[7] [20], which is a cryptographic tool to achieve secure communication. Parallel to these developments, attempts to bridge physics and computing [21] evolved into formal definitions of computational models incorporating quantum mechanical principles [22, 23]. Independently, Feynman argued that simulating quantum systems on classical computers would require enormous computational resources, proposing instead the construction of computers based on quantum mechanics itself [24]. This seminal insight established the concept of the **quantum computer**, a machine that uses **qubits** as the fundamental unit of information processing. In this section, I introduce how the axioms and properties of quantum theory presented in the previous section can help us perform information processing.

2.2.1 Qubits

In classical computing, we deal with **bits**, which are objects that take values of either 0 or 1. Mathematically, these objects can be defined as the elements of the finite field $\mathbb{F}_2$. There is a quantum mechanical representation of this bit, simply called **qubits**[8] Specifically, a single qubit is a single element from the two-dimensional Hilbert space. Classical bits can be mapped to qubits as follows:

$$\text{bit: } 0 \to \text{qubit: } |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \tag{2.93}$$

$$\text{bit: } 1 \to \text{qubit: } |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{2.94}$$

[7]The original idea of Wiesner was first established in the late 1960s; however, it remained unpublished until 1983 as the concept was considered ahead of its time.

[8]The term "qubit," representing a two-level quantum system, was coined by Schumacher and Wootters [25].

These two states form a complete orthonormal basis, $\{|0\rangle, |1\rangle\}$. In quantum computation, this basis is typically referred to as the **computational basis**. However, as quantum state can be in a **superposition** between these two states, the following state $|\psi\rangle$ is also a valid qubit.

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \text{ where } |\alpha|^2 + |\beta|^2 = 1 \tag{2.95}$$

Together with the axiom of measurement, we can tell this qubit will be in $|0\rangle$ with probability $p_0 = |\alpha|^2$, and in $|1\rangle$ with probability $p_1 = |\beta|^2$, as long as we perform quantum measurement in the computational basis.

When we deal with multiple bits, we simply line them up in a row like 101001. For multiple qubits, we deal with them as a *composite system formed by the tensor product of their individual state spaces.* The state $|101001\rangle$ is an example of a multiple-qubit system. In general, as the collection of all bit strings forms an orthonormal basis for the Hilbert space of n qubits, an arbitrary n-qubit system can be written as follows.

$$|N\rangle = \sum_{i\in\mathbb{F}_2^n} c_i\,|i\rangle \text{ where } \sum_i |c_i|^2 = 1 \tag{2.96}$$

Here, $\mathbb{F}_2^n$ is the set containing all classical n-bit strings. In other words, an n-qubit system can represent all n-bit strings at once. Do not misunderstand that due to this parallelism, quantum computers solve problems quickly. When actually extracting information from this state, we need to perform quantum measurement, which only returns us *one sample* from the probability distribution following our measurement basis[9].

2.2.2 Quantum Circuit Model

In classical computation, Turing machines [26] represent a generalized model of computation. In quantum computing, where we deal with qubits instead of bits, there also is the quantum equivalent of a Turing machine, known as the quantum Turing machine [27]. However, such Turing machines might be useful for proving computability, but it is difficult to interpret them as explicit computation processes. Instead, we write programs, pseudocode, flowcharts, and, at the lowest level, Boolean logic circuits. Using these tools makes our lives much easier rather than living in the mathematically abstract world of infinite tapes and state machines.

Therefore, it is natural to expand the classical notation of Boolean logic circuits into quantum circuits [28, 29]. Quantum circuits read the same way as classical logic circuits, where time flows from left to right. The only difference is that rather than using classical logic gates such as AND, OR, and NOT, we use quantum gates, which are unitary operators. Measurement operations are also included in the quantum circuit notation. A list of various commonly used quantum gates and measurements is given in Tab. 2.4.

[9]In fact, we can only read out at most one bit of classical information from such superposition state. This limitation follows from an information theoretical bound known as the Holevo bound [19].

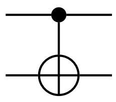
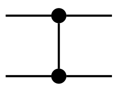
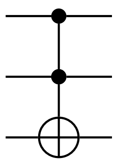
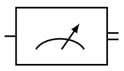
Gate Name	**Symbol**	**Matrix Representation**
Pauli X gate	$\boxed{X}$	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
Pauli Y gate	$\boxed{Y}$	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$
Pauli Z gate	$\boxed{Z}$	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
Hadamard gate	$\boxed{H}$	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
Phase (S) gate	$\boxed{S}$	$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$
CNOT (Controlled NOT) gate		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$
CZ (Controlled Z) gate		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$
T ($\pi/8$) gate	$\boxed{T}$	$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$
Toffoli (CCNOT)		$\begin{pmatrix} 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&1&0 \end{pmatrix}$
Computational basis measurement		Measurement in $\{\|0\rangle, \|1\rangle\}$

Table 2.4: Gate and matrix representations of common quantum operators.

There are several interesting gates I would like to describe in detail. We can notice that the Pauli X gate corresponds to a bit-flip operation in the computational basis, as $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$. This means that a NOT operation can be performed on qubits as long as they are in the computational basis. In contrast, the Pauli Z gate acts on the phase rather than the bit value. This is typically called performing a phase-flip. For example, applied to an equal superposition, it flips the sign of the state, $Z(|0\rangle + |1\rangle) = |0\rangle - |1\rangle$.

Another interesting gate is the CNOT gate, which is a conditional gate whose action depends on its control qubit. If the control qubit is in state $|0\rangle$, the target qubit remains unchanged. If the control qubit is in state $|1\rangle$, a Pauli X gate is applied to the target qubit. When we consider inputs from the computational basis, the CNOT gate ends up writing the result of an XOR operation of the two input bits to the target qubit[10].

$$\text{CNOT}_{1,2}|q_0\rangle \otimes |q_1\rangle = |q_0\rangle \otimes |q_0 \oplus q_1\rangle \tag{2.97}$$

With these quantum gates, we can describe any kind of quantum computation we would like to perform on our input qubits. For example, a circuit performing quantum teleportation [30], a protocol that allows quantum information flow from one location to another, can be written as shown in Fig. 2.4.

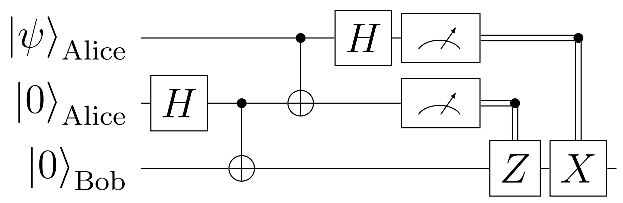


Figure 2.4: Quantum circuit performing state teleportation. Here, Alice is preparing the arbitrarily generated state she wishes to teleport, and Bob is trying to reconstruct the same state on his qubit.

The quantum state teleportation protocol can be explained as follows. Think of the scenario where Alice has prepared a single qubit in the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. The first two quantum gates, the Hadamard gate and the CNOT gate, alter the input qubits as follows:

$$|0\rangle_{\text{Alice}}|0\rangle_{\text{Bob}} \overset{H}{\mapsto} \frac{1}{\sqrt{2}}(|0\rangle_{\text{Alice}}|0\rangle_{\text{Bob}} + |1\rangle_{\text{Alice}}|0\rangle_{\text{Bob}}) \overset{\text{CNOT}}{\mapsto} |\Phi^+\rangle_{\text{Alice,Bob}}\,. \tag{2.98}$$

[10]In some sense, the CNOT gate represents a reversible XOR gate, as the output bits have not erased any information of the input bits

Rewriting the entire composite system, we obtain[11].

$$
\begin{aligned}
|\psi\rangle \otimes |\Phi^+\rangle &= \frac{1}{\sqrt{2}} \left(\alpha |0\rangle |00\rangle + \beta |1\rangle |00\rangle + \alpha |0\rangle |11\rangle + \beta |1\rangle |11\rangle\right) && (2.99)\\
&= \{|\Phi^+\rangle (\alpha |0\rangle + \beta |1\rangle) + |\Phi^-\rangle (\alpha |0\rangle - \beta |1\rangle) + \\
&\qquad |\Psi^+\rangle (\alpha |1\rangle + \beta |0\rangle) + |\Psi^-\rangle (\alpha |1\rangle - \beta |0\rangle)\} && (2.100)\\
&= \left(|\Phi^+\rangle |\psi\rangle + |\Phi^-\rangle Z |\psi\rangle + |\Psi^+\rangle X |\psi\rangle + |\Psi^-\rangle XZ |\psi\rangle\right). && (2.101)
\end{aligned}
$$

The CNOT between Alice's two qubits, and the Hadamard gate, corresponds to performing a Bell state measurement on the two qubits residing on Alice's side. After this operation, Bob's state results in the following state.

$$
\begin{aligned}
\Pr(\Phi^+) = 1/4, &\quad |\text{Bob}\rangle = |\psi\rangle && (2.102)\\
\Pr(\Phi^-) = 1/4, &\quad |\text{Bob}\rangle = Z |\psi\rangle && (2.103)\\
\Pr(\Psi^+) = 1/4, &\quad |\text{Bob}\rangle = X |\psi\rangle && (2.104)\\
\Pr(\Psi^-) = 1/4, &\quad |\text{Bob}\rangle = XZ |\psi\rangle && (2.105)\\
& && (2.106)
\end{aligned}
$$

Bob applies the appropriate Pauli operators depending on Alice's Bell state measurement[12], and as a result, Bob reconstructs Alice's initial qubit $|\psi\rangle$ on his qubit! Within the circuit diagram, this is represented with the double line, where the measured result controls the Pauli gate on Bob's qubit.

This might seem like an obvious example, at least in a local picture. However, if we separate Alice and Bob far away, then this protocol gains significance. In the circuit picture, we have performed multiple qubit operations to generate a Bell pair between Alice and Bob, but as long as we can have that single pair shared between them by some other method, we can teleport quantum information from one side to another just using **LOCC** (Local Operation and Classical Communication). This means that, as long as we have some shared entanglement and a method to communicate with each other, quantum information can be transferred from one location to another!

2.2.3 Universal Quantum Computation

For classical computers, a specific set of logic gates is sufficient to perform universal computation on bits. Similarly, one might ask whether all quantum operations can be represented by their equivalent quantum circuits. The axiom of quantum time evolution states that any (pure) quantum operation can be written as a unitary operation. Therefore, to achieve **universal quantum computation** with quantum circuits, we must be able to synthesize an arbitrary unitary from discrete quantum gates.

[11] I have omitted the subscripts for simplicity. The qubits are ordered in $|\psi\rangle_{\text{Alice}} |0\rangle_{\text{Alice}} |0\rangle_{\text{Bob}}$.

[12] This indeed does perform correction of the X/Z factor in front of $|\psi\rangle$, as for any Pauli operator, the square of itself is equivalent to an identity, i.e., $X^2 = Y^2 = Z^2 = I$.

A natural concern is whether universality is even achievable for quantum systems. While the space of possible unitaries for an n-qubit system is continuous and infinite, a quantum gateset is discrete. Fortunately, this problem simplifies to the ability to perform a CNOT gate and an arbitrary single-qubit unitary, as this combination can generate any unitary operator within an n-qubit system. Therefore, being able to perform a single-qubit unitary suffices.

However, because arbitrary single-qubit rotations are continuous, an exact representation using a finite gateset is often impossible. To address this issue, we rely on approximation. We say that V approximates U with ε accuracy when:

$$\max_{|\psi\rangle} \|(U - V)\,|\psi\rangle)\| \leq \varepsilon. \tag{2.107}$$

In fact, the combination of gates from the set $\{H, T\}$ generates a unitary V that approximates any single-qubit unitary U, while arbitrarily suppressing the error ε. From this, we can claim the gateset $\{H, T, \text{CNOT}\}$ generates a universal unitary on n-qubits.

There exists an efficient algorithm for achieving such approximation known as the Solovay-Kitaev decomposition [31]. This algorithm allows efficient gate construction, stating that an arbitrary single-qubit unitary U with ε error can be approximated by $\mathcal{O}(\log^c(1/\varepsilon))$[13] gates from the gateset $\{H, S, T, S^\dagger, T^\dagger\}$ with some constant c. This allows us to synthesize an arbitrary unitary without adding significant overhead to our computation. Other modern methods further optimize this decomposition by reducing the total gate count or the circuit depth of resource-intensive gates, while also achieving lower time complexity for the decomposition process itself [32].

2.3 Quantum Algorithms

In the previous section, I described the foundations of how to use quantum systems as computers. However, we have not yet answered the question of what they are actually useful for. Some might argue that since quantum computers were originally proposed for quantum simulation, their primary use should be as experimental devices to study quantum mechanics. While such usage is certainly possible, there have been a significant number of discoveries in the domain of algorithms, some of which hold the potential to catastrophically alter our modern civilization in profound ways. In this section, I will discuss the utility of quantum computers by outlining some major quantum algorithms.

2.3.1 Scaling of Algorithms

Before presenting specific examples of quantum algorithms, we need a reasonable framework to compare them. One might consider using the execution time of an algorithm;

[13]The notation used here is known as the big-O notation, which will be introduced in the following section.

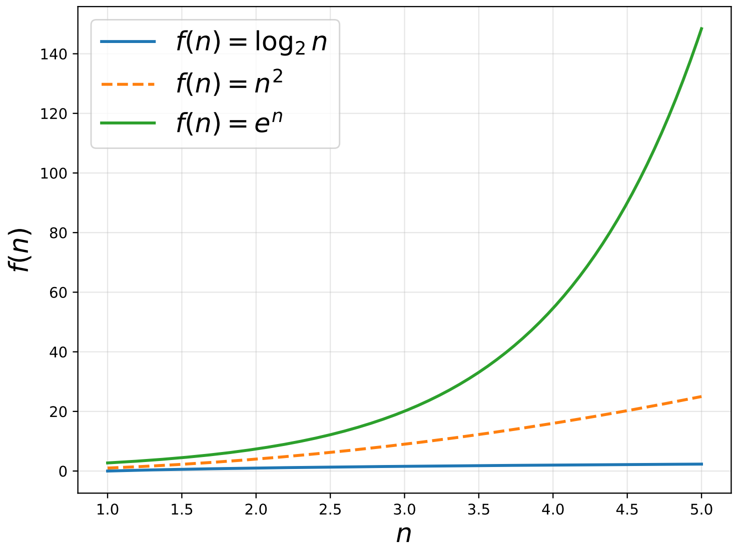


Figure 2.5: Scaling differences between $\log n$, n^2, and e^n.

however, such metrics heavily depend on the specific hardware implementation details[14]. Instead, we focus on the fundamental theoretical differences between algorithms independent of hardware and architecture design. Specifically, our primary interest lies in computational complexity, which represents how the required computational time/space scales as the size of the input becomes arbitrarily large. This scaling behavior is formalized using the **Big-O notation**.

Let $f(n)$ represent the cost of an algorithm (such as the required computation steps) for an input of size n. We say that $f(n)$ is of the order of $g(n)$, denoted as $f(n) = \mathcal{O}(g(n))$, if $f(n)$ is asymptotically bounded above by $g(n)$ up to a constant factor.

$$f(n) = \mathcal{O}(g(n)) \iff \exists c, n_0 > 0 \text{ s.t. } \forall n \geq n_0, \quad |f(n)| \leq c \cdot |g(n)|. \tag{2.108}$$

This behavior can be interpreted as extracting the leading order term of the function $f(n)$, while omitting the constant coefficient. For example, a computational cost function $f(n) = 5n^3 + 2n^2 + n - 1$ is simplified to $\mathcal{O}(n^3)$ in Big-O notation.

To understand the significance of computational complexity scaling, consider a scenario where three distinct algorithms solve the same problem with costs of $\mathcal{O}(\log n)$, $\mathcal{O}(n^2)$, and $\mathcal{O}(e^n)$, respectively. The scaling behaviors of these functions are compared in Fig. 2.5. As illustrated, exponential functions grow rapidly, while logarithmic functions do not see this growth, especially for a large n. From this, the algorithm with $\mathcal{O}(\log n)$ complexity is the best choice among the three.

[14]It is important to clarify that "hardware implementation details" here refers to performance differences within the same computational model (e.g., processor clock speed or memory size), rather than the fundamental theoretical differences between quantum and classical computers.

We can also consider other notations such as Big-Ω (Omega) and Big-Θ (Theta), which describe lower and tight bounds, respectively. However, in our case of evaluating algorithms, it is practical to focus on the upper bound captured by the Big-O notation. This allows us to be sufficiently "pessimistic" in our predictions, providing a guarantee on the worst-case performance.

2.3.2 Classical Search and Grover's Algorithm

Search algorithms are fundamental to computer science, serving as a canonical example of the importance of appropriately designed algorithms. To understand this, consider the following scenario. We are given an array of size n and we wish to find the index of a specific target number x. For simplicity, assume there are no redundant numbers in the array and that we can query the value at any given index k in $\mathcal{O}(1)$ time. How efficiently can we find x? If our array is sorted (in either ascending or descending order; here we will stick to ascending order), there exists a well-known classical algorithm called binary search. A single iteration of this procedure cuts the searching space in half. Hence, we can say that after $\log_2 n$ steps, we will find the index for x (or determine that x does not exist in the array), so the computational complexity for this algorithm is $\mathcal{O}(\log n)$.

However, performing the same task on an unsorted database is much more complex. This is because, there is nothing else to do but to tediously search every index one-by-one. If we are lucky enough, we might find x right after we begin searching, but on average, it should take $n/2$ steps. In the worst case, we can spend n searches as we might miss every single element until we reach the last index. Therefore, the complexity is $\mathcal{O}(n)$.

For quantum computers, Grover's algorithm [1] provides a more efficient solution. Provided access to an oracle, this algorithm has a computational complexity of $\mathcal{O}(\sqrt{n})$, which is a quadratic speedup compared to the classical case. Here, an "oracle" is an entity that returns us the answer to a certain query in $\mathcal{O}(1)$ time. For the specific case of Grover's algorithm, it is a unitary operator O such that $O\,|x\rangle = (-1)^{f(x)}\,|x\rangle$. In our case of searching the index of an unstructured database, $f(k)$ is a Boolean function which returns 1 if the input index k corresponds to the target item, and 0 otherwise. Furthermore, Grover's algorithm is not limited to searching for a number in an array; it is applicable to general search problems.

2.3.3 Shor's Algorithm

Previously, we discussed an example where quantum computers can be more efficient than classical computers. However, for that case, the improvement was limited to a polynomial speedup over the classical case; specifically, it was a quadratic speedup. While accelerating the search of a massive unordered dictionary is significant, we must also consider the practical overhead of quantum computers. When we include the cost of fault tolerance, which we will introduce in the next chapter, this quadratic speedup can be easily canceled out [33].

This raises the question of, do significantly more efficient and impactful quantum algorithms even exist? The answer is yes. A prime example is Shor's algorithm, which is a quantum algorithm that solves the mathematical problem of integer factoring [2]. For classical algorithms, there are no known instances of solving this problem in polynomial time. For example, consider factoring a number N. Then, the best-known classical algorithm for factoring an integer, known as the general number field sieve, costs approximately $\mathcal{O}(e^{1.9(\log N)^{1/3}(\log\log N)^{2/3}})$ time [34]. Relying on this computational hardness, integer factorization serves as the foundation for secure communication protocols, such as RSA encryption [35]. In contrast, the quantum counterpart, Shor's algorithm, runs in $\mathcal{O}((\log N)^3)$ time. This suggests that quantum computers may possess[15] fundamental computational capabilities distinct from classical computers. Furthermore, this capability could compromise our modern digital society, which has a heavy dependence on the assumption that efficient integer factorization is almost impossible. Numerous variants of quantum factoring algorithms have since been developed, building upon Shor's foundational work and offering various improvements [36, 37, 38, 39].

2.3.4 Other Quantum Algorithms

Beyond the foundational algorithms discussed so far, there exist many other quantum algorithms that show some level of quantum advantage. Aligning with the original motivation for quantum computing, there is a handful of quantum algorithms designed for quantum physics simulations [40, 41, 42]. In the domain of linear algebra, the HHL algorithm provides a method for solving systems of linear equations with exponential speedup under specific conditions [43]. Additionally, there exists a wide class of heuristic quantum algorithms designed to find approximate solutions to combinatorial optimization problems [44, 45]. Moreover, recent theoretical work suggests that these seemingly distinct quantum algorithms can be described under a unified framework [46].

2.3.5 Hardness of Simulating Quantum Computation

We have discussed how quantum computers have an advantage compared to classical computers, in terms of quantum algorithms showing better scaling than their classical counterparts. But, how different are they in a broader sense? One might come up with the following question: "Can quantum computers be simulated by classical computers? If that is possible, then what is the point of quantum computers?" This is indeed a very reasonable claim. If we think of "computation" itself, then yes, quantum computers can be simulated with classical computers. In the end, the problems these two models

[15]Note that this is not a proven fact. We have just shown an example where classical computers are outperformed by quantum computers, but we do not know whether there exists a much more efficient classical algorithm for factoring.

can solve are known to be equivalent, in the same way for many other computational models[16].

However, the theoretical "efficiency" of computation seems to differ from classical computers. In fact, naive simulation of quantum processes with a classical computer requires exponential time. Simulation of quantum computing becomes intractable as the system size grows, which led to Feynman's original proposal of building a quantum computer [24].

It is important to note that there is currently no theoretical proof that provides a strict separation between the computational power of classical and quantum computers. This open problem is known as the question of whether $\mathsf{BQP} = \mathsf{P}$.[17] If $\mathsf{BQP} = \mathsf{P}$, the fundamental motivation for constructing quantum computers were to vanish, this would imply the existence of efficient classical algorithms for hard problems such as integer factoring. Conversely, if $\mathsf{BQP} \neq \mathsf{P}$, it would theoretically confirm that quantum computers do have a definitive computational advantage over their classical counterparts.

In fact, there actually exist specific classes of quantum computing that can be efficiently simulated by classical computers. Obviously, it is impossible to show any quantum advantage with them, but they turn out to be useful when analyzing the behavior of specific quantum computing procedures, which we will explore in the following sections.

2.4 Quantum Communication

Previously, we have reviewed quantum computing and quantum algorithms. In this section, I introduce quantum communication, another application of treating quantum systems as information processing resources.

2.4.1 Quantum Repeaters

How do we realize quantum communication in the real world? Although direct transmission of quantum states is theoretically possible, it suffers from exponential attenuation in optical fibers. For the classical case, this issue of attenuation was simply dealt with by amplifying the signal. However, unlike classical signals, quantum information cannot be amplified, as the no-cloning theorem [16] prohibits copying arbitrary quantum states. This makes direct transmission of quantum states impractical.

Instead, we employ entanglement as the communication resource, utilizing quantum state teleportation to transmit arbitrary quantum information at the cost of one Bell pair per qubit. Consequently, the challenge of quantum communication can be reduced

[16]This is more of a definition rather than a claim. We define a task as computable if it can be executed by a Turing machine. This principle is known as the Church-Turing thesis [47].

[17]Here, BQP (Bounded-error Quantum Polynomial time) and P (Polynomial time) are complexity classes that represent the problems that can be solved in polynomial time, for quantum computers and classical computers, respectively.

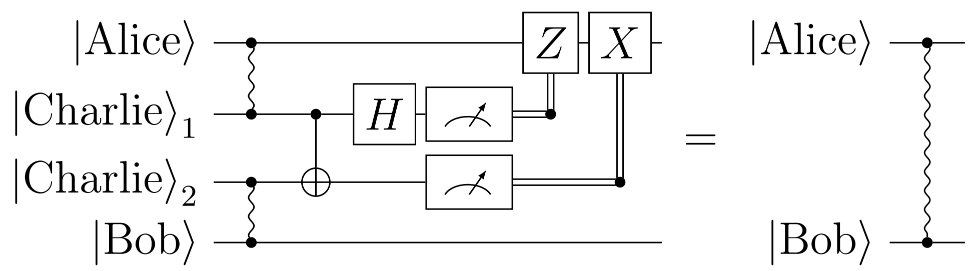


Figure 2.6: Quantum circuit for entanglement swapping. Wavy lines between parties denote shared entanglement, which is assumed to be the Bell state $|\Phi^+\rangle$. We are applying Pauli correction to Alice in this figure, but note that the same correction can be applied to Bob as well.

to the task of distributing entanglement efficiently. To achieve this, we combine a technique known as entanglement swapping with error management, establishing a unified framework known as quantum repeaters [48, 49].

Consider the scenario where we wish to share entanglement between two distant parties, Alice and Bob. If the distance between them is too long for direct transmission, we introduce an intermediate node, Charlie, to serve as the repeater. First, link-level Bell pairs are generated between Alice and Charlie and Bob and Charlie, respectively, using a quantum link architecture [50]. Performing a Bell state measurement on Charlie's two qubits and applying the appropriate Pauli correction on Alice or Bob, end-to-end entanglement is established. This process can be written in the quantum circuit representation as shown in Fig. 2.6. This approach effectively splits the distance photons need to travel in half, reducing the attenuation error, and by repeatedly placing quantum repeaters in appropriate locations, entanglement can be efficiently shared among remote parties.

However, even with entanglement swapping, the quality of the distributed Bell pairs is still not perfect, as no quantum operations are noise-free. The link-level generation process incurs inherent noise, and the swapping operations at Charlie can further degrade the fidelity of the end-to-end Bell pair. To circumvent this issue and make these states useful for practical applications, we employ methods known as entanglement distillation[18] [11, 51]. This allows us to distill a smaller number of high-fidelity pairs from a larger pool of low-fidelity inputs.

2.4.2 Applications of Quantum Communication

The most well-known application of quantum communication is in the field of security [52]. Quantum communication allows for the generation of an information-theoretically secure classical shared key, a process typically referred to as Quantum Key Distribution (QKD). Here, information-theoretically secure means that the security holds regardless of the adversary's computational power. This stands in contrast to computational security

[18]Depending on the literature, distillation is often referred to as purification.

schemes such as RSA encryption [35], which are vulnerable to quantum attacks. The information-theoretic security of QKD protocols can be attributed to quantum mechanical properties. Some exploit the state-collapsing feature of quantum measurement [53], while others rely on non-classical correlations from entanglement[19] [55, 56].

However, the field of quantum communication is not limited to secure communication. Applications of this field extend to physics-related domains such as high-precision sensing [57, 58] and clock synchronization [59]. Moreover, it enables computational protocols such as blind quantum computing[20] [60, 61], and distributed consensus algorithms such as Byzantine agreement [62]. More importantly, for achieving scalable computation, they provide the backbone for distributed quantum computing [5, 63, 64].

[19]These seemingly distinct protocols can be mapped to one another for rigorous security analysis [54].

[20]Blind quantum computing is a framework for delegating quantum computation to a remote server without the server learning the input, algorithm, or output of the computation.

Chapter 3

Quantum Error Correction and Fault-Tolerant Quantum Computing

Quantum systems are subject to unavoidable environmental noise. Although we can run noisy quantum circuits with current technology known as Noisy Intermediate-Scale Quantum (NISQ) devices [65], the usefulness of NISQ is debated. For example, claims of NISQ utility in physical simulations do show an advantage against brute-force classical algorithms [66], but are not advantageous against optimized algorithms [67].

On the other hand, quantum algorithms that show theoretical advantage over classical counterparts typically require more qubits and operations. This means that more errors should accumulate on the qubit, requiring a low error rate for the individual operations. For example, it is known that the required error rate per operation is at the level of 10^{-15} for cracking RSA-2048 [68]. While directly reducing the error rates of individual qubits might be beneficial, physical refinements themselves are insufficient for achieving such stringent requirements. Therefore, it is essential to pursue Fault-Tolerant Quantum Computing (FTQC) using Quantum Error-Correcting Codes (QECCs).

This chapter will first review quantum error correction. Then, I will introduce various topics on fault-tolerant quantum computing, which makes quantum error correction work in realistic settings. This chapter concludes with a description of what it actually means for a quantum system to be fault-tolerant.

3.1 Quantum Error-Correcting Codes

This section provides the theoretical background on coding theory. I progress from classical error correction to quantum error correction. Then, I introduce stabilizer codes and surface codes. A discussion on decoding concludes the section.

3.1.1 Repetition Codes

Error correction, whether classical or quantum, relies on introducing redundancy by mapping **logical** information into a larger **physical** space. This redundancy allows for the recovery of logical information, typically through operations known as parity checks or syndrome measurements. I refer to the valid representation of logical information as **codewords** for both quantum and classical error correction theory.

An error-correcting code is defined by the parameters n, k, and d. Here, n represents the number of physical bits (or qubits), k represents the number of encoded logical bits (or qubits), and d denotes the code distance. Conventionally, the notation $[n, k, d]$ is used for classical codes and $[[n, k, d]]$ for quantum codes. In the classical setting, the code distance d corresponds to the minimum Hamming distance[1] between any two distinct codewords. In the quantum setting, the definition of distance is generalized as the minimum weight[2] of an operator required to map one codeword to another, rather than by simple bit differences.

(1) Classical Repetition Code

In classical coding theory, the **repetition code** is a simple example of an error-correcting code that has the ability to correct the bit-flip channel. The bit-flip channel is a noise process where, with probability p, a single bit flips from 0 to 1 (or vice versa), as illustrated in Fig 3.1. This channel is assumed to be i.i.d.[3], so every bit has a probability p of flipping and $1 - p$ of remaining unchanged.

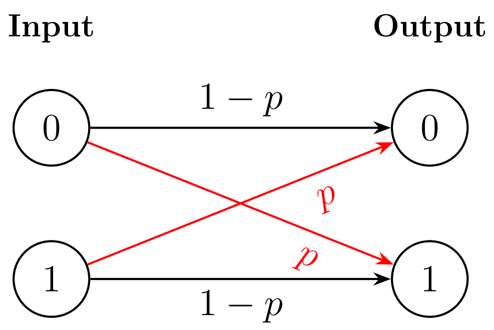


Figure 3.1: The bit-flip channel. p is the probability of a bit-flip error.

The three-bit repetition code introduces redundancy by encoding the logical bits 0

[1]The Hamming distance is defined as the number of bit positions at which the two strings differ. For example, the Hamming distance between 000 and 111 is three, 101 and 111 is one.

[2]Here, the weight of an operator is defined as the number of qubits on which the operator acts non-trivially.

[3]i.i.d. stands for independent and identically distributed. In this context, it implies that every bit has the same independent probability p of flipping.

and 1 as follows.

$$0 \mapsto 000 \tag{3.1}$$

$$1 \mapsto 111 \tag{3.2}$$

To correct errors, we perform a majority vote and correct the state to the most frequently appearing bit in the string. Therefore, if we observe 011, we correct this to 111 (logical 1), and for 010, we correct to 000 (logical 0).

To see how this encoding/decoding scheme is resilient against the bit-flip channel, let us consider an example where the initial state is 000. All possible bit-flip channel errors and decoding outcomes are summarized in Tab. 3.1. From this table, we can see that the

Bit-Flips	Resulting State	Probability	Decoded As
0	000	$(1-p)^3$	0 (Success)
1	100 010 001	$p(1-p)^2$	0 (Success)
2	110 101 011	$p^2(1-p)$	1 (Failure)
3	111	p^3	1 (Fail)

Table 3.1: Possible correction outcomes of classical repetition codes, where the initial state 000 passes through a bit-flip channel with error probability p.

success probability is the sum of the probabilities for the cases of zero and one bit-flip errors.

$$\Pr(\text{success}) = (1-p)^3 + 3p(1-p)^2 \tag{3.3}$$

Therefore, if $\Pr(\text{success}) > p$, the overall error rate reduces from the case where we did not do encoding. This inequality holds when $p < 0.5$ (as $0 \le p \le 1$), so the repetition code functions when our bit-flip channel has less than 50% chance of flipping each bit. The code distance gives a good measure of how resilient our error-correcting code is against errors. If we let the number of errors as t, we can relate this value to the code distance by $t = \lfloor d/2 \rfloor$. The two logical codewords for the three-bit repetition code are given as 000 and 111, so the code distance is three. From this, we can correct up to one error by using the $[3, 1, 3]$ repetition code.

(2) The Quantum Repetition Code

The quantum counterpart of this repetition code might be difficult to think of. By simple thought, we might need to think of "copying" our logical information, which is prohibited due to the no-cloning theorem. However, there is a way to overcome this issue by introducing redundancy not into the original state itself, but only for the basis states $|0\rangle$ and $|1\rangle$. For a three-qubit bit repetition code, we encode the basis states as follows.

$$|0\rangle \mapsto |000\rangle \tag{3.4}$$
$$|1\rangle \mapsto |111\rangle \tag{3.5}$$

One might worry this violates the no-cloning theorem. Yet, this theorem only applies to the cloning of arbitrary states. Since we are performing a specific unitary transformation, the encoding is physically valid, illustrated by the quantum circuit in Fig. 3.2. If we check

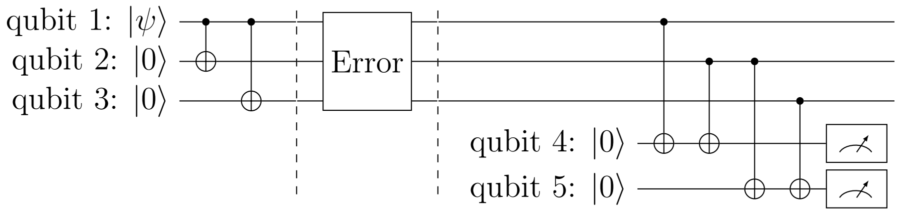


Figure 3.2: The bit repetition code. The encoding circuit is on the left, and the parity check is on the right of the error.

the resulting state of this error-correcting code, it is not cloning some arbitrary state $|\psi\rangle$, but rather simply mapping $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ to a state $|\overline{\psi}\rangle = \alpha|000\rangle + \beta|111\rangle$.

However, the correction part is performed slightly differently from our classical counterpart, as we cannot simply take a majority vote between the three bits that form our codeword. This is because measuring qubits collapses the state and might change the logical information that was initially encoded. Instead, we employ a parity check method using two ancilla qubits. Denote whether there was a bit-flip on the i-th data qubit as $x_i \in \{0, 1\}$, where $x_i = 1$ indicates an error. The two ancilla qubits measure the parity of two data qubits, where their state before measurement corresponds to the XOR[4] of the errors.

$$|\text{ancilla}_1\rangle = |x_1 \oplus x_2\rangle \tag{3.6}$$
$$|\text{ancilla}_2\rangle = |x_2 \oplus x_3\rangle \tag{3.7}$$

By measuring these ancilla qubits, we obtain a two-bit string called the error syndrome. The error-correcting schematic depending on the syndrome value is shown in Tab. 3.2.

From this, we can see that the error syndrome allows us to uniquely identify single-qubit errors (remember, the classical three-bit repetition code also only had the ability to correct up to one error), without collapsing the superposition of the logical data. You might think that the code parameters for the quantum repetition code might be the same as the classical case, but that is actually untrue. We can only say this is a $[[3, 1]]$ code; as for quantum error-correcting, the notion of distance needs to account for all kinds of quantum errors, not only Pauli X errors.

[4]XOR is a binary operation that stands for exclusive OR. Mathematically, this represents the addition of bits in modulo 2.

Error syndrome $(x_1 \oplus x_2, x_2 \oplus x_3)$	Diagnosed Error	Correction Action
$(0,0)$	No Error	No Correction (I)
$(1,0)$	Error on Qubit 1	Flip Qubit 1 (X_1)
$(1,1)$	Error on Qubit 2	Flip Qubit 2 (X_2)
$(0,1)$	Error on Qubit 3	Flip Qubit 3 (X_3)

Table 3.2: Syndrome table for the 3-qubit bit repetition code. The syndrome bits indicate the parity between qubits $(1,2)$ and $(2,3)$.

In the quantum world, there are many other kinds of errors. For simplicity, let us think about correcting a phase-flip error, or a Pauli Z error. This might look challenging at first, but a simple modification to the circuit of the quantum repetition code suffices for our initial goal. We just need to think of the basis $\{|+\rangle, |-\rangle\}$, where $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ and $|-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$. Transforming the computational basis $\{|0\rangle, |1\rangle\}$ to the basis $\{|+\rangle, |-\rangle\}$ just requires adding Hadamard gates to the beginning and the ending of the circuit, so we obtain Fig. 3.3 as our encoding/parity check circuit for the phase repetition code.

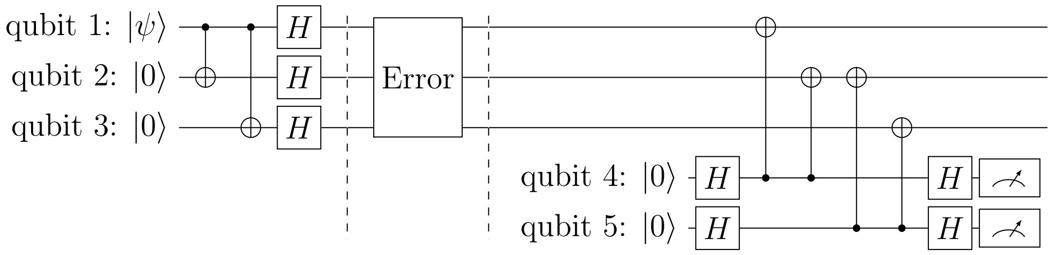


Figure 3.3: The phase repetition code. The encoding circuit is on the left, and the parity check is on the right of the error.

Under this encoding scheme, we map $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ to the state $|\overline{\psi}\rangle = \alpha|+++\rangle + \beta|---\rangle$. Using parity check measurements allows us to detect which qubit suffered a phase-flip error and correct it. This strategy is identical to that of the bit repetition code, except that the correction target is replaced from a Pauli X error to a Pauli Z error.

3.1.2 The Shor Code

However, by changing our basis, we have now lost the ability to correct against bit-flip errors. This might sound daunting, as we can only protect against either type of error, but in fact, we can overcome this issue by **concatenating** error-correcting codes. We encode our initial qubit in one type of error-correcting code, in this case the bit repetition code, and then encode the already encoded state into another error-correcting code that can serve for the other type of error (in this case, the phase repetition code). This concatenated error-correcting code is also known as the Shor code [3].

The encoding and parity check circuit for the Shor code is shown in Fig. 3.4. Combining bit/phase repetition codes allows us to correct all kinds of Pauli errors. X and Z errors are dealt with straightforwardly, as they can be corrected by their respective codes. Moreover, Y errors are simultaneous bit-flip and phase-flip errors, as $Y = iXZ$, independently correcting the X and Z components suffices.

Even with the Shor code, some might speculate that we can't correct quantum errors because qubits have a much wider error space. What about if one qubit suffers an H error? What about continuous errors that rotate the qubit some random angle around the Z axis? I will answer that question in the following section by introducing the discretization of errors.

3.1.3 Discretization of Errors

There exists a computational model called analog computers. While digital computers are resilient to noise due to their quantized behavior of being only able to take one or zero per bit, analog computers suffered from continuous errors due to their inherent feature of encoding information in fluctuating real values. One might point out that, quantum computers are also yet another analog computer, where error correction is impossible due to this issue of continuity vs. error correction.

However, unlike analog computers, quantum computers are not subject to this continuous error issue. In fact, we can perform quantum error correction on any arbitrary CPTP map as long as we can correct arbitrary Pauli errors. We will look at this example by using a repetition code that corrects Z type errors on a single logical qubit. For simplicity, consider the following Z rotation continuous error R_θ described as follows.

$$R_\theta = \begin{pmatrix} e^{-i\theta} & 0 \\ 0 & e^{i\theta} \end{pmatrix} \tag{3.8}$$

Here, the special case $\theta = \pi/2$ gives the Pauli Z gate. This operator can be decomposed into a linear combination of Pauli operators,

$$R_\theta = \cos\theta\, I - i \sin\theta\, Z \tag{3.9}$$

If this error acts on an encoded state $|\overline{\psi}\rangle$, then

$$R_\theta |\overline{\psi}\rangle = \cos\theta\, |\overline{\psi}\rangle - i \sin\theta\, Z |\overline{\psi}\rangle . \tag{3.10}$$

One might point out that the error R_θ is not correctable, as the state is in a mixture between $|\overline{\psi}\rangle$ and $Z|\overline{\psi}\rangle$, where in our previous examples, we only discussed correcting discrete Pauli errors. However, correcting such continuous errors turns out to be possible, even with the same scheme as before. The key for discretization is the **linearity** of quantum operations and the **projective** nature of quantum measurements.

Denote the ancilla qubit's error syndrome by representing no error as $|I\rangle_{\text{syn}}$ and Z error on data qubit i as $|Z_i\rangle_{\text{syn}}$. We also ensure these two states are orthogonal to each

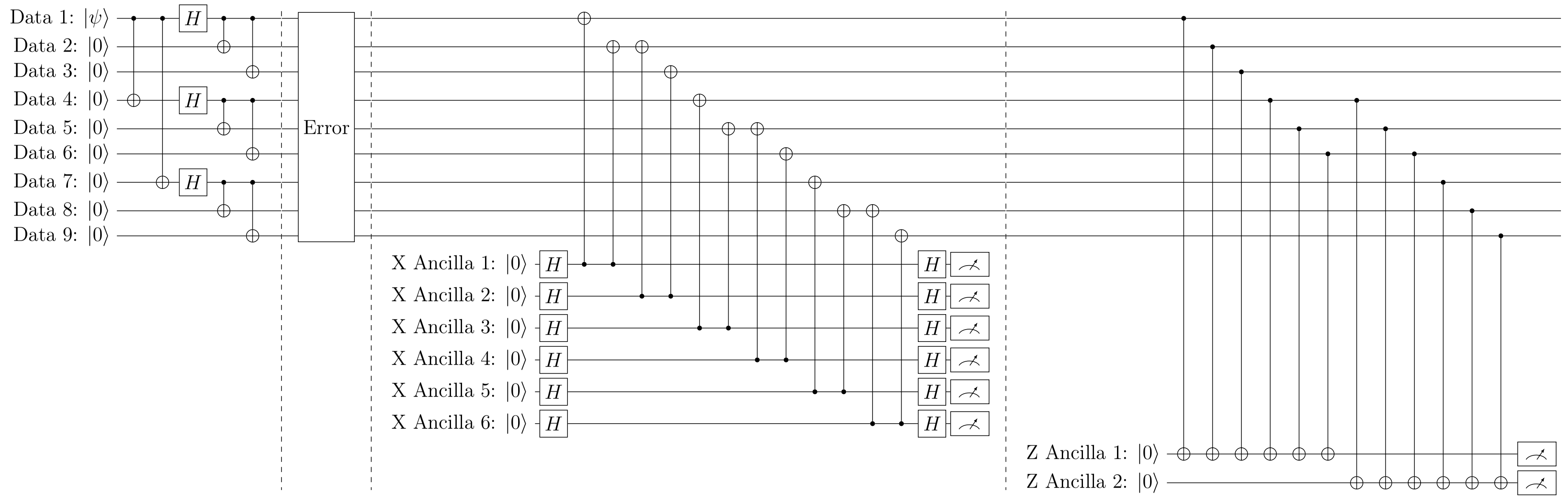


Figure 3.4: The Shor code. The encoding circuit is on the left, X-error syndrome extraction circuit is in the middle, and Z-error syndrome extraction circuit is on the right.

other. After the parity check, the ancilla's state changes as follows:

$$|\overline{\psi}\rangle\,|0\rangle \overset{\text{syn}}{\mapsto} |\overline{\psi}\rangle\,|I\rangle_{\text{syn}}\,, \qquad Z_i\,|\overline{\psi}\rangle\,|0\rangle \overset{\text{syn}}{\mapsto} Z_i\,|\overline{\psi}\rangle\,|Z_i\rangle_{\text{syn}}\,. \tag{3.11}$$

Due to the **linear** nature of quantum operations, the action of the error R_θ can be described as

$$\begin{aligned} R_\theta\,|\overline{\psi}\rangle\,|0\rangle &= \cos\theta\,\,|\overline{\psi}\rangle\,|0\rangle - i\sin\theta\,Z_i\,|\overline{\psi}\rangle\,|0\rangle\,, &(3.12)\\ &\overset{\text{syn}}{\mapsto} \cos\theta\,\,|\overline{\psi}\rangle\,|I\rangle_{\text{syn}} - i\sin\theta\,Z_i\,|\overline{\psi}\rangle\,|Z_i\rangle_{\text{syn}}\,. &(3.13)\end{aligned}$$

Performing a measurement on the ancilla qubit **projects** the states into the following.

$$\Pr(|I\rangle_{\text{syn}}) = \cos^2\theta, \quad \text{resulting state: } |\overline{\psi}\rangle \tag{3.14}$$

$$\Pr(|Z_i\rangle_{\text{syn}}) = \sin^2\theta, \quad \text{resulting state: } Z\,|\overline{\psi}\rangle \tag{3.15}$$

Therefore, the effect of the continuous error rotation is effectively discretized into either the identity (I) or a definitive Pauli Z_i error. If the measured syndrome gives out a Z_i error, the correction is applied by simply applying a Z_i gate; otherwise, we leave the state unchanged. This principle ensures that continuous phase errors are correctable by a phase repetition code. This error discretization property can be generalized by the following theorem.

Theorem 3.1 (*Discretization of Quantum Errors*)

If $\mathcal{C}$ is a quantum error-correcting code for a set of errors $\mathcal{E}$, then $\mathcal{C}$ is also a valid QECC for the error space span($\mathcal{E}$). *This implies that if a QECC can correct two different errors E and F, it can also correct any linear combination $\alpha E + \beta F$.*

Since the Shor code is designed to correct all single-qubit Pauli errors (X, Y, and Z), the theorem above tells us we can correct any single-qubit error. This holds because the Pauli operators form a basis for the space of all single-qubit operators[5].

$$\forall E \in \mathsf{SU}(2), \quad \exists \alpha, \beta, \gamma, \delta \in \mathbb{C} \text{ such that } E = \alpha I + \beta X + \gamma Y + \delta Z \tag{3.16}$$

This linearity principle can be extended into multi-qubit errors as well. Thus, we can correct not only continuous local errors but also highly correlated errors affecting multiple qubits at the same time. For rigorous proofs and detailed discussion on the correction of arbitrary errors, refer to Chapters 1 and 2 of Gottesman's textbook on QECC and fault tolerance [69].

3.1.4 Stabilizer Codes

(1) The Stabilizer Group and Stabilizer Codes

The bit/phase repetition codes, as well as the Shor code, belong to a broad category of QECCs known as stabilizer codes [4]. I will first introduce the Pauli group[6], which serves

[5]$\mathsf{SU}(2)$ is the group of single-qubit unitary gates. This is typically referred to as "special unitary", where this "special" comes from the fact that the global phase is not in our interest.

[6]For a review on the mathematical principles of group theory, refer to the appendix.

as the foundation of stabilizer theory.

Definition 3.1 (*Pauli Group*)

The n-qubit Pauli group P_n is defined as the set of all possible tensor products of Pauli operators, including the global phase.

$$\mathsf{P}_n = \{\alpha \cdot P_1 \otimes P_2 \otimes \cdots \otimes P_n \mid \alpha \in \{\pm 1, \pm i\}, P_j \in \{I, X, Y, Z\}\} \tag{3.17}$$

The set P_n forms a group because it satisfies the necessary axioms: it contains the identity element ($I^{\otimes n}$), every element possesses an inverse (since Pauli matrices are unitary), and the set is closed under matrix multiplication.

Keeping this Pauli group in mind, the **stabilizer** of a QECC is defined as follows.

Definition 3.2 (*Stabilizer of a QECC*)

Let C be the code space of a QECC defined on n qubits. The stabilizer of this QECC, denoted as $\mathsf{S}(C)$, is defined as:

$$\mathsf{S}(C) = \{M \in \mathsf{P}_n \mid M |\psi\rangle = |\psi\rangle, \forall |\psi\rangle \in C\}. \tag{3.18}$$

In other words, the stabilizer is the set of Pauli operators for which every codeword in C is a $+1$ eigenvector.

The stabilizer $\mathsf{S}(C)$ possesses the following three properties.

1. **$-I \notin \mathsf{S}(C)$:** As the operator $-I$ does not permit any $+1$ eigenvectors, $-I$ cannot be an element of any stabilizer.

2. **Group:** Let $\forall M, N \in \mathsf{S}(C)$. Their combined action on a codeword is $MN|\psi\rangle = M(N|\psi\rangle) = M|\psi\rangle = |\psi\rangle$. Thus, the product MN is also an element of the stabilizer, ensuring the closure condition. The existence of identity and inverse follows directly from the properties of the Pauli group.

3. **Abelian:** Let $\forall M, N \in \mathsf{S}(C)$. As $MN|\psi\rangle = |\psi\rangle$ and $NM|\psi\rangle = |\psi\rangle$, we have $MN|\psi\rangle = NM|\psi\rangle$. Since M and N are Pauli group elements, they must either commute or anticommute. If they anticommute ($MN = -NM$), then $2MN|\psi\rangle = 0$ so $|\psi\rangle = 0$. This contradicts the assumption of a non-trivial code space. Thus, every element in $\mathsf{S}(C)$ must commute, so the stabilizer group is Abelian.

We then define the stabilized subspace from a "stabilizer," an Abelian Pauli subgroup without $-I$ which I denote as S.

Definition 3.3 (*The Stabilized Subspace*)

Let a subset $\mathsf{S} \subseteq \mathsf{P}_n$ be an Abelian group, such that $-I \notin \mathsf{S}$. Then the stabilized subspace space $V(\mathsf{S})$ is defined as

$$V(\mathsf{S}) = \{|\psi\rangle \mid M|\psi\rangle = |\psi\rangle, \forall M \in \mathsf{S}\} \tag{3.19}$$

If a code space C is equivalent to the stabilized subspace of the stabilizer of C, the corresponding QECC has a special name.

Definition 3.4 (*Stabilizer Code*)

Let C be a QECC. C is a **stabilizer code** *if*

$$C = V(\mathsf{S}(C)). \tag{3.20}$$

Furthermore, any Abelian subgroup of the Pauli group without $-I$, uniquely determines a stabilizer code as shown in the following theorem.

Theorem 3.2 (*Stabilizer Code Theorem*)

Let $\mathsf{S} \subseteq \mathsf{P}_n$ be an Abelian subgroup such that $-I \notin \mathsf{S}$. Then,

$$\mathsf{S} = \mathsf{S}(V(\mathsf{S})). \tag{3.21}$$

The proof of this theorem is beyond the scope of this thesis. For a detailed discussion, refer to Chapter 3 of Gottesman's textbook [69]. From this point forward, I will often identify a stabilizer code by its defining stabilizer S.

(2) Properties of Stabilizer Codes

The error-correcting properties of a stabilizer code are determined by the **generators**[7] of the stabilizer group S. Let $\mathsf{S} = \langle g_1, g_2, \ldots, g_r \rangle$ be a stabilizer group defined by r generators acting on an n-qubit physical space. The projector that maps an arbitrary state onto the $+1$ eigenspace of a single generator g_i is given by $(I + g_i)/2$. Since the stabilized subspace $V(\mathsf{S})$ is the simultaneous $+1$ eigenspace of all generators, the global projector is the product of these individual generator projectors[8]. Therefore, the overall projection operator onto the subspace $V(\mathsf{S})$ can be written as follows:

$$\Pi_{\mathsf{S}} = \prod_{i=1}^{r} \frac{I + g_i}{2}, \tag{3.22}$$

$$= \frac{1}{2^r} \sum_{M \in \mathsf{S}} M. \tag{3.23}$$

Here, we have used the property of the order of the stabilizer group S being $|\mathsf{S}| = 2^r$, as stabilizer groups are Abelian, and for any $g_i \in \mathcal{G}$, $g_i^2 = I$.

Taking the trace of a projection operator corresponds to the dimension of the projected

[7]Refer to the appendix for a general discussion on Group theory.

[8]Performing projection onto the generators implicitly projects onto the entire group. Separate projectors for the remaining non-generator elements are redundant.

subspace. As the trace of any Pauli operator is zero: $\mathrm{tr}(P) = 0$, the following holds.

$$\mathrm{tr}(\Pi_{\mathsf{S}}) = \mathrm{tr}\left(\frac{1}{2^r}\sum_{M\in\mathsf{S}} M\right) \tag{3.24}$$

$$= \frac{1}{2^r}\sum_{M\in\mathsf{S}} \mathrm{tr}M = \frac{\mathrm{tr}I}{2^r} = 2^{n-r} \tag{3.25}$$

From this, the stabilized subspace has a dimension of 2^{n-r}, corresponding to $n-r$ qubits. With the stabilizer code theorem, this means that the encoded space has $n-r$ logical qubits. We typically set $r = n-k$, making the stabilizer code an $[[n,k]]$ code.

But how do we actually perform syndrome measurements within the stabilizer framework? It turns out that we can do them by "measuring" the stabilizer generators. In our case, we perform repeated measurements of the stabilizer generator to ensure that our codeword is within the code space of the QECC. The quantum circuit for measuring a generator g_i is given in Fig. 3.5. With this, we can project our codeword to either

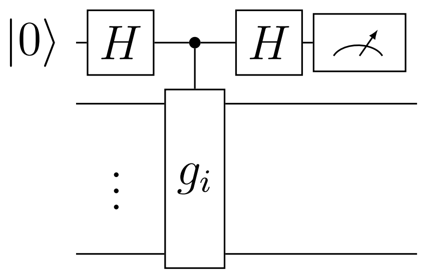


Figure 3.5: Quantum circuit performing stabilizer element measurement.

$+1$ or -1 eigenstates of the stabilizer. For $+1$ eigenstates, we can leave the codeword as it is. For -1 eigenstates, by applying an appropriate Pauli operator, we can map the erroneous state back to one of the codewords[9]. By repeating this for all g_is in the generator of S, we can perform syndrome measurements of a stabilizer code and perform error detection/correction.

However, stabilizer codes also have errors that cannot be corrected nor detected. In fact, the errors that stabilizer codes can detect are limited to Pauli errors[10] which anticommute with any stabilizer element. This is shown as follows. Let $M \in \mathsf{S}$ and $E \in \mathsf{P}_n$, and $ME = -EM$. Then, for any $|\psi\rangle \in C$, the following holds.

$$M(E|\psi\rangle) = -EM|\psi\rangle = -E|\psi\rangle \tag{3.26}$$

This means that $E|\psi\rangle$ turns out to be a -1 eigenstate of the stabilizer, which can be detected by stabilizer measurements. What if M and E commute? For most cases,

[9]In general, we identify the stabilizers that yield -1 eigenvalues for M_i and apply the predicted Pauli error $\tilde{E}$ based on these outcomes. This process is known as decoding, which will be introduced in the next section.

[10]As mentioned previously, the ability to correct Pauli errors corresponds to correcting arbitrary errors.

as stabilizers are Abelian, a commuting E turns out to be a stabilizer, which are $+1$ eigenoperators on a code word. However, if an error commutes and is not a stabilizer element, we cannot even detect it.

To formalize the definition of undetectable errors, we define the normalizer of a stabilizer as follows.

Definition 3.5 (*Normalizer*)

Let S *be a stabilizer group. Then, the* ***normalizer*** $N(\mathsf{S})$*, is defined as follows.*

$$N(\mathsf{S}) = \{N \in \mathsf{P}_n \mid NM = MN,\ \forall M \in \mathsf{S}\} \tag{3.27}$$

Then, errors that are not correctable nor detectable via a stabilizer code are operators within the set $N(\mathsf{S}) \setminus \mathsf{S}$. This also implies that the code distance of a stabilizer code is the minimum weight operator within $N(\mathsf{S}) \setminus \mathsf{S}$.

$$d = \min\{\mathrm{wt}(E) \mid E \in N(\mathsf{S}) \setminus \mathsf{S}\} \tag{3.28}$$

(3) Examples of Stabilizer Codes

Examples of the stabilizer code are the bit/phase repetition codes and the Shor code, which I introduced in the previous section. For the bit repetition code, the stabilizer generators are $\langle Z_1Z_2, Z_2Z_3\rangle$, while for the phase repetition code, the generators are $\langle X_1X_2, X_2X_3\rangle$. The bit parity checks correspond to measuring the Z-stabilizers, and the phase parity checks correspond to measuring the X-stabilizers. For the Shor code, the generators are given in Tab. 3.3. A common trait among these three stabilizer codes

	1	2	3	4	5	6	7	8	9
g_1	Z	Z							
g_2		Z	Z						
g_3				Z	Z				
g_4					Z	Z			
g_5							Z	Z	
g_6								Z	Z
g_7	X	X	X	X	X	X			
g_8				X	X	X	X	X	X

Table 3.3: Stabilizer generators for the Shor code. Each column corresponds to the physical qubit indices (1–9). Empty cells denote the identity operator I.

is that they are CSS codes [70, 71], meaning their stabilizer generators can be separated into strictly X-type and strictly Z-type operators. CSS codes can be easily constructed from classical error-correcting codes known as linear codes.

The five-qubit code is the most compact QECC capable of encoding a single logical qubit and correcting any single-qubit error, achieving a code distance of $d = 3$. It is a stabilizer code defined by the generators listed in Tab. 3.4. Notably, the five-qubit code is

	1	2	3	4	5
g_1	X	Z	Z	X	
g_2		X	Z	Z	X
g_3	X		X	Z	Z
g_4	Z	X		X	Z

Table 3.4: Stabilizer generators for the five-qubit code. The columns correspond to the physical qubit indices (1–5). Empty cells denote the identity operator I.

not a CSS code, as it is impossible to construct a generator set composed exclusively of X-type and Z-type operators. Consequently, it does not correspond to classical linear codes. However, within the framework of non-binary codes, the five-qubit code corresponds to a linear code over GF(4) [72].

(4) Surface Codes

The surface code is a stabilizer code (and also a CSS code) widely considered as the leading candidate for quantum error correction [6]. It is derived from Kitaev's toric code, which was originally defined on a torus encoding two logical qubits [73]. By adapting this structure to a two-dimensional plane[11], the planar surface code is obtained [74]. Both the surface and toric codes belong to the class of topological codes, where code properties are defined by the geometrical layout and interaction of qubits.

As it is a topologically defined code, the surface code can be well understood with a picture. An example of a distance-three unrotated surface code is shown in Fig. 3.6. Here, the circles correspond to data qubits, and the red and blue plaquettes illustrate the stabilizer generators. Red plaquettes correspond to stabilizer generators where Pauli X operators are applied to the adjacent data qubits, which means the generators are either $XXXX$ or XXX. For blue plaquettes, the same principle applies, but with Z-type stabilizers $ZZZZ$ or ZZZ. We can confirm all generators commute with each other, as the support of red and blue plaquettes only share zero or two data qubits[12]. As the number of logical qubits is the number of physical qubits minus stabilizer generators $(13 - 12 = 1)$, the surface code encodes one logical qubit. Generalizing this to arbitrary code distances by placing d data qubits on the edge, we can conclude that the surface code is a $[[d^2 + (d-1)^2, 1, d]]$ code[13].

For implementing the surface code, stabilizer measurements are performed by placing an ancilla qubit in the location of the red and blue plaquettes, realized by the following

[11]Mathematically, this transformation corresponds to replacing the periodic boundary conditions of the torus with open boundary conditions.

[12]Plaquettes of the same color trivially commute.

[13]The surface code can be constructed as a hypergraph product [75] (a method of constructing CSS codes from classical linear codes) of two repetition codes of the same code distances.

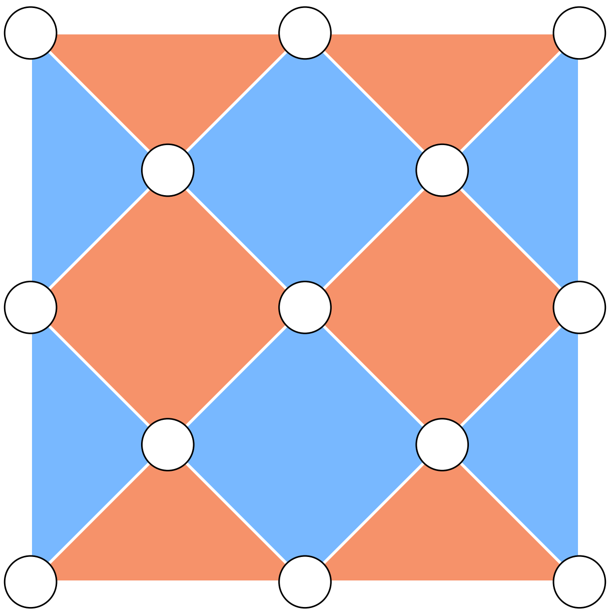
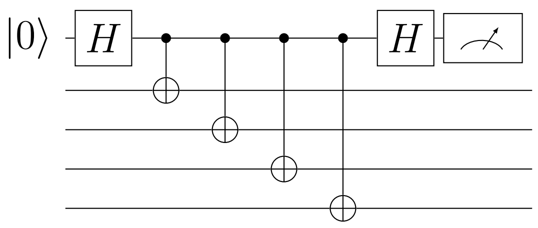
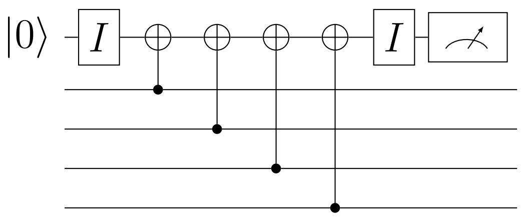


Figure 3.6: An example of an unrotated surface code of code distance $d = 3$. We map XYZ Pauli operators to RGB colors respectively, so in this figure, the red plaquettes correspond to X-stabilizers, while the blue plaquettes represent Z-stabilizers.

circuits as shown in Figs. 3.7. From this, we can notice that the surface code only



(a) X-stabilizer measurement.



(b) Z-stabilizer measurement.

Figure 3.7: Quantum circuits for stabilizer measurements in the surface code. In order to synchronize the time between the two different stabilizer measurements, an identity gate I is inserted for the Z-stabilzier measurement. In practice, this corresponds to waiting for the Hadamard to end.

requires nearest-neighbor connectivity on a 2D grid, where we only apply multi-qubit gates to such pairs of qubits. Because physical platforms like superconducting qubits often possess limited connectivity, having such features makes the surface code a leading candidate for realizing quantum error correction. Furthermore, the surface code is known to have good operational capabilities, where universal quantum computation is fairly easy to achieve, and has high error-suppression ability, also phrased as having a high "threshold," as explained in the following sections of this chapter.

3.1.5 Decoding an Error-Correcting Code

As we saw previously in error correction theory (regardless of classical or quantum), we need to perform decoding. Here, decoding stands for estimating the most likely physical error that has occurred on the logical qubit, using information from the syndrome measurements. For simple codes like the Shor code, creating a lookup table suffices, as we already know what errors cause what kind of syndrome measurement results at what probability. We can then select the most probable error from that table according to our error syndromes.

This is not always the case for larger and more complex codes, such as surface codes. These codes might have several distinct physical errors on the qubits that lead to the identical error syndrome, where we need to choose which is the most probable physical error that has occurred on our qubits. In short, we cannot easily identify which physical errors have caused a specific syndrome measurement. One might assume that we can take the same approaches for simple codes by constructing lookup tables and choosing the most probable error path. However, such approaches do not scale, as the complexity of the lookup table explodes with the size of the code. In fact, it has been shown that for general quantum error-correcting codes, identifying this correspondence between error syndromes and actual errors is an NP-hard problem[14] [77, 78].

Due to this NP-hardness, heuristic methods are often employed to solve this decoding problem. For instance, the Minimum-Weight Perfect Matching (MWPM) algorithm [79] is known to be highly effective for surface codes, represented as in decoders such as Py-Matching [80]. Moreover, despite being a heuristic, solving the MWPM provides provable theoretical guarantees regarding error suppression capabilities [81].

3.2 Towards Fault-Tolerant Quantum Computing

So far, we have looked at how QECCs are composed and how they function to eliminate errors. However, we only had the view that the encoding/error-correcting circuits were not affected by errors, which does not hold in the real world. In reality, they are all noisy quantum gates; every single location in our circuit has a chance of introducing an error. In this section, I step forward from such kind of static error correction to actually establishing Fault-Tolerant Quantum Computing (FTQC) by introducing the various problems one may encounter when implementing QECCs.

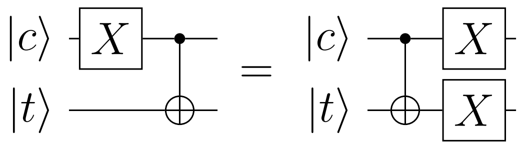
3.2.1 Error Propagation

Error propagation is a major obstacle for achieving fault-tolerant quantum computing. Taking a look back at the circuit for syndrome measurements, one might notice many

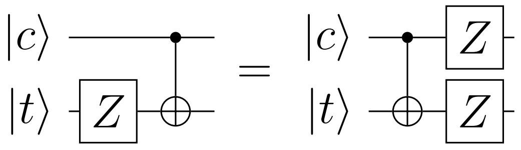
[14]This property is known to be true for classical error-correcting codes [76] as well.

CNOT gates are going through the ancilla qubits and the data qubits. In fact, CNOT gates can propagate noise to each other qubit, which can lead to filling up the system with unintentional errors. For such cases, even if we begin with a QECC with a large code distance, the error can easily grow out of our error-correcting capabilities.

For example, think of applying a CNOT gate between two qubits, but right before the application of that gate, an X error or a Z error happens. In that case, we get the following commutation relation with CNOT gates and Pauli gates shown in Fig. 3.8 for X and Z type errors. As the X or Z gate here was an unintentional gate, the resulting state after an ideal CNOT is followed by not one Pauli error, but two Pauli errors.



(a) X error propagation on the CNOT gate.



(b) Z error propagation on the CNOT gate.

Figure 3.8: Pauli error propagations on the CNOT gate.

This relation is shown as follows.

$$\begin{aligned}
\mathrm{CNOT}_{1,2}(X\otimes I) &= (|0\rangle\langle 0|\otimes I + |1\rangle\langle 1|\otimes X)(X\otimes I) && (3.29)\\
&= |0\rangle\langle 1|\otimes I + |1\rangle\langle 0|\otimes X && (3.30)\\
&= X|1\rangle\langle 1|\otimes XX + X|0\rangle\langle 0|\otimes X && (3.31)\\
&= (X\otimes X)(|1\rangle\langle 1|\otimes X + |0\rangle\langle 0|\otimes I) && (3.32)\\
&= (X\otimes X)\mathrm{CNOT}_{1,2} && (3.33)
\end{aligned}$$

$$\begin{aligned}
\mathrm{CNOT}_{1,2}(I\otimes Z) &= (|0\rangle\langle 0|\otimes I + |1\rangle\langle 1|\otimes X)(I\otimes Z) && (3.34)\\
&= |0\rangle\langle 0|\otimes Z + |1\rangle\langle 1|\otimes XZ && (3.35)\\
&= |0\rangle\langle 0|\otimes Z - |1\rangle\langle 1|\otimes ZX && (3.36)\\
&= Z|0\rangle\langle 0|\otimes Z + Z|1\rangle\langle 1|\otimes ZX && (3.37)\\
&= (Z\otimes Z)(|0\rangle\langle 0|\otimes I + |1\rangle\langle 1|\otimes X) && (3.38)\\
&= (Z\otimes Z)\mathrm{CNOT}_{1,2} && (3.39)
\end{aligned}$$

If we were lucky enough and the X and Z gates were in the opposite location, there would be no error propagation, but as errors are probabilistic, we would like to avoid such error propagation as much as possible.

3.2.2 Dealing with Operational Errors

Real-world quantum circuits are susceptible to faults at various locations. As previously discussed, the issue of error propagation is already a significant challenge. Furthermore, the qubit measurements in syndrome measurement circuits can yield incorrect bit values, leading to erroneous correction operations. Moreover, the entire computation process, including initial encoding, gate operations, and logical measurements, is inherently noisy. This means the simplified model of encoding a state, letting it idle while errors accumulate, and then recovering the state via syndrome measurements is unrealistic. The operations susceptible to errors in quantum circuits are as follows:

1. **State Preparation**: We must prepare an encoded state with low error probability. If this initialization is unreliable, all subsequent operations are compromised.
2. **Logical Gates**: We need to execute logical gates on the encoded qubits to proceed with the computation without accumulating too many errors.
3. **Logical Measurements**: At the end of a quantum computation, we must read out the resulting codeword $|\overline{\psi}\rangle$ reliably.
4. **Idle**: Qubits idling between logical gates or measurements are also subject to error accumulation.
5. **Error Correction**: Error correction itself must also be performed reliably, avoiding the introduction of new errors while preventing erroneous corrections.

We refer to the robust implementation of these operations as **fault-tolerant** implementations. Here, fault tolerance is defined by limited error accumulation within an operation, where the introduced errors are sufficiently small enough to be corrected by some QECC. With such requirements, the error-corrected circuit can be designed as a repeating sequence of fault-tolerant logic locations followed immediately by fault-tolerant error correction[15]. Having this structure prevents errors from accumulating into a logical fault, as errors from all potentially faulty locations are detected and suppressed before they exceed the code's error-correcting capacity.

Let's take a look at designing fault-tolerant error correction on stabilizer codes. For stabilizer codes, the standard error correction cycle involves measuring syndromes for each generator, followed by a classical decoding operation to predict the Pauli error, concluding with a correction operation on the logical qubits. We need to prevent error propagation and handle faulty measurements to make this process fault tolerant. There do exist various protocols to rigorously achieve fault tolerance, but they are typically complicated and resource intensive [82, 83, 84].

On the other hand, fault-tolerant error correction is fairly simple for surface codes. It turns out that just performing d rounds of repeated syndrome measurements is sufficient. This is because repeated stabilizer measurements create a spatio-temporal structure of measurement events, which, with the help of a matching-based decoder, allows the correction of measurement errors in the temporal direction over the course of d rounds [85].

[15]For logical measurement, the quantum state collapses to classical information, so further error correction is unnecessary.

3.3 Logical Gates

How do we perform computation on our logical qubits? In classical computing, we do not really worry about this issue, because direct operations on bits do not introduce as much noise as the quantum world. We might perform encoding when transmitting data or storing data, but computation is more or less treated as raw operations on the decoded information. There do exist methods to perform computation on encrypted data, such as homomorphic encryption [86], but such schemes have more emphasis on performing private computation rather than reducing operational errors. However, for the quantum case, there is a need to perform our desired unitary U on a logical codeword. This is because the goal for encoding is not mainly for security but to perform more accurate and practical computation even under the presence of noise.

For this reason, we need to compose fault-tolerant **logical gates**, implementing some target operation we intended to perform on our codewords. Here, I present various methods to apply logical gates on our encoded quantum states, with a special focus on surface codes.

3.3.1 Transversal Gates

A family of quantum gates that are not heavily affected by error propagation is known as transversal gates. A **transversal** operation is defined as follows.

Definition 3.6 (*Transversal Operation*)

Consider a composite system of m code blocks, where each block consists of n physical qubits. A quantum operation $\mathcal{F}$ is said to be transversal if

$$\mathcal{F} = \bigotimes_{i=1}^{n} \mathcal{G}_i. \tag{3.40}$$

Here, each component $\mathcal{G}_i$ acts exclusively on the subsystem formed by the i-th physical qubits from all m code blocks, ensuring that qubits with different physical indices do not interact.

Transversal gates prevent intra-codeblock error propagation. Even if an error on a single physical qubit occurs, it cannot propagate to other qubits within the same code block. This property ensures fault tolerance, as errors can occur but cannot grow larger due to the structure of the logical gate.

For stabilizer codes, logical Pauli gates X, Y, Z are naturally transversal. This is because the set of undetectable errors $N(\mathsf{S}) \setminus \mathsf{S}$ corresponds to logical Pauli gates, which are Pauli group elements. For example, applying the operation $Z^{\otimes 9}$ on the Shor code changes the state from a logical $|0\rangle$ to a logical $|1\rangle$, implementing a transversal logical X gate.

In surface codes, logical Pauli gates can be transversally implemented by applying physical X or Z gates on a particular chain of physical qubits as shown in Fig. 3.9. Chains of physical Pauli operators that connect opposite boundaries commute with all

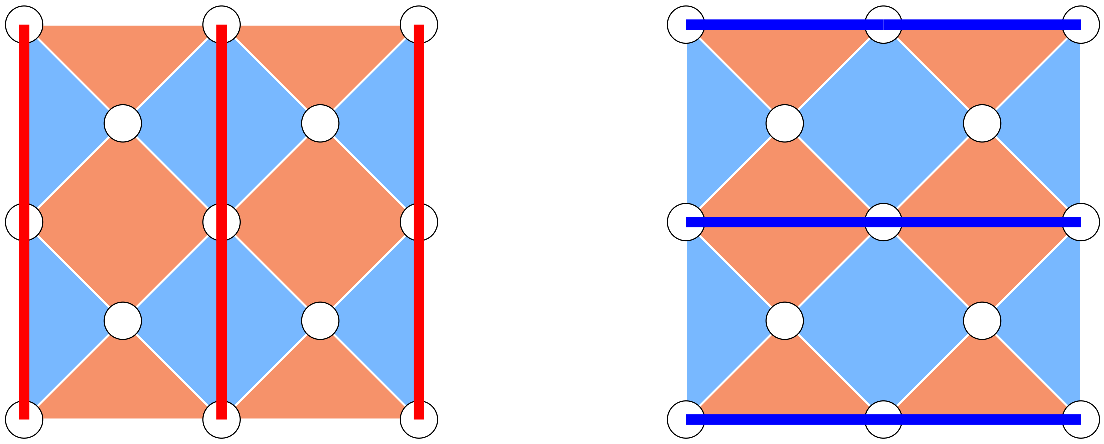

(a) Logical X operators (b) Logical Z operators

Figure 3.9: Examples of logical Pauli operators on the surface code. (a) The logical X gate is implemented by applying physical X gates on a chain of qubits along the red path. (b) The logical Z gate is implemented by physical Z gates along the blue path. These paths are chosen to connect their respective boundaries (e.g., the logical X operator connects X-type boundaries). Note that these chains are not unique, as multiplying a logical operator by any stabilizer element results in an equivalent logical operation.

stabilizers of the surface code, while remaining independent of the stabilizer group itself. For instance, the logical X operators shown in Fig. 3.9a commute with all Z-stabilizers, but they cannot be expressed as a product of any generators. Therefore, such chains correspond to elements in $N(\mathsf{S}) \setminus \mathsf{S}$, and implement our desired logical Pauli gates[16]. The logical Y gate can then be implemented by the combination of logical X and Z as $Y = iXZ$.

There are many other examples of transversal gates in various QECCs. For example, the CNOT gate can be transversally implemented for CSS codes by performing data-qubit-wise physical CNOTs [4]. One might therefore hope to realize a universal fault-tolerant gate set entirely through transversal operations. However, the Eastin-Knill theorem proves this to be impossible, stating that for any QECC, a transversal universal gate set cannot be composed [87].

3.3.2 Surface Codes and Lattice Surgery

As surface codes are CSS codes, logical CNOT gates can be implemented transversally. However, the transversal CNOT on the surface code assumes a high connectivity require-

[16]The reason we choose logical X and Z to be on different directions is to make them anticommute, satisfying requirements of Pauli algebra. As you might notice, every single logical X operator intersects with any logical Z operator at an odd number of data qubit sites.

ment, in which all data qubits are connected to their relevant partner in the other surface code. This is not a desirable behavior, as nearest-neighbor connectivity suffices for stabilizer measurements. Does this suggest that we need to break this good condition when performing error-corrected computation?

Fortunately, there are ways to perform logical CNOT gates in a fault-tolerant manner while preserving connectivity. One well-known method is **lattice surgery** [88], which allows the fault-tolerant measurement of logical Pauli product operators via d rounds of stabilizer measurements between the boundaries of multiple surface code patches. With this, we can generate correlations between surface codes, and as a result, perform CNOT gates with each other.

As an example of lattice surgery, a Bell pair between two qubits can be generated by measuring the observable $X \otimes X$ on $|00\rangle$. Consider the following situation where our logical qubits are initialized in the following state.

$$|0\rangle \otimes |0\rangle = \frac{|{+}{+}\rangle + |{+}{-}\rangle + |{-}{+}\rangle + |{-}{-}\rangle}{\sqrt{2}} \tag{3.41}$$

Measuring the operator $X \otimes X$ yields the following measurement results and post-measurement states.

$$\Pr(+1) = 1/2, \quad \text{resulting state: } |\Phi^+\rangle \tag{3.42}$$

$$\Pr(-1) = 1/2, \quad \text{resulting state: } |\Psi^+\rangle \tag{3.43}$$

From this, we can understand that a Pauli product measurement generates a Bell pair.

One can visualize why measuring the stabilizers between patches corresponds to measuring the logical Pauli product as follows. We first bring two surface code patches adjacent to one another. Then, we introduce two intermediate qubits α and β between the patches, defining X-stabilizers as shown in Fig. 3.10. Using the data qubit labeling provided in the figure, we choose the logical X operators to be $\overline{X}_L = X_3X_8X_{13}$ for the left patch and $\overline{X}_R = X_{14}X_{19}X_{24}$ for the right patch. The column of X-stabilizers bridging the two logical operators, from top to bottom, are:

$$M_{\text{top}} = X_3X_{14}X_\alpha, \quad M_{\text{mid}} = X_\alpha X_8 X_{19} X_\beta, \quad M_{\text{bot}} = X_\beta X_{13} X_{24}. \tag{3.44}$$

When we take the product of these three stabilizers, the operators on the intermediate qubits (X_α and X_β) cancel out:

$$M_{\text{top}} \cdot M_{\text{mid}} \cdot M_{\text{bot}} = X_3X_8X_{13}X_{14}X_{19}X_{24} = \overline{X}_L\overline{X}_R. \tag{3.45}$$

Therefore, the measurement results of these intermediate stabilizers directly yield the measurement result of the logical operator $\overline{X}_L \otimes \overline{X}_R$. Lattice surgery is inherently fault-tolerant as it is solely composed of stabilizer measurements throughout the process, where d rounds of such measurements ensuring reliable readouts.

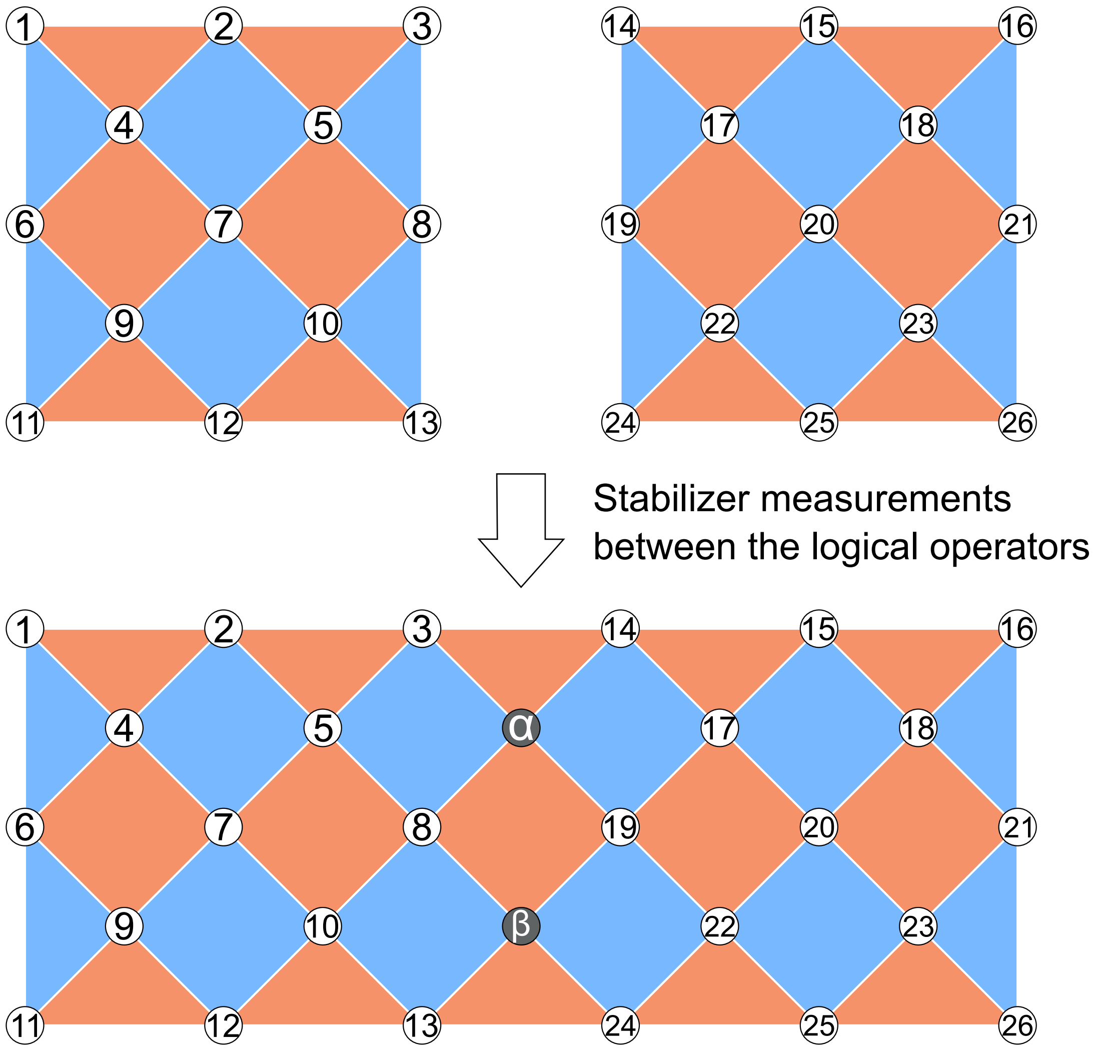


Figure 3.10: Performing lattice surgery on surface codes. The newly introduced intermediate data qubits α and β both need to be initialized in $|0\rangle$, as we do not want them to interfere with the stabilizer measurement results of the adjacent Zstabilizers.

3.3.3 Clifford Gates and Non-Clifford Gates

(1) The Clifford Group

For stabilizer codes, there exists an interesting class of quantum gates known as Clifford gates. I first begin with the definition of a Clifford group, a set of unitary operators that map a Pauli group element to another Pauli group element under conjugation.

Definition 3.7 (*Clifford Group*)

The n-qubit Clifford group C_n is defined as follows.

$$\mathsf{C}_n = \{U \in \mathsf{U}(2^n) \mid UPU^\dagger \in \mathsf{P}_n \forall P \in \mathsf{P}_n\} \tag{3.46}$$

Here, $\mathsf{U}(2^n)$ is the set of unitary operators (including the global phase) on n-qubits.

The set C_n can be confirmed as a group, as for any $U, V \in \mathsf{C}_n$ and $P, Q \in \mathsf{P}_n$, UV yields the following relation.

$$(UV)P(UV)^\dagger = (UV)P(V^\dagger U^\dagger) \tag{3.47}$$
$$= U(VPV^\dagger)U^\dagger \tag{3.48}$$
$$= UQU^\dagger \in \mathsf{P}_n \tag{3.49}$$

The same property also holds for VU. The generator for an n-qubit Clifford group C_n can be written as the gateset as follows:

$$\mathcal{G} = \langle H_i, S_i, \mathrm{CNOT}_{i,j}, e^{i\theta} I\rangle. \tag{3.50}$$

We do not require $e^{i\theta}I$ for quantum computing as it only contributes to the global phase of a unitary. We call an element of the Clifford group (especially an element from this generator) as Clifford gates.

(2) The Gottesman-Knill Theorem

The Gottesman-Knill theorem states that a quantum circuit only with Clifford gates can be efficiently simulated on a classical computer [89].

Theorem 3.3 (*Gottesman-Knill Theorem*)

A quantum circuit can be simulated on a classical computer in polynomial time if it consists solely of state preparation in the computational basis, gates from the Clifford group, and measurements in the computational basis.

The foundation of this efficiency lies in the **stabilizer formalism**. With n stabilizer generators, an $[[n, 0]]$ stabilizer code can be defined where $k = n - n = 0$. This corresponds to the exact description of a single, specific n qubit state, as zero logical qubits imply the dimension of the protected subspace to be $2^0 = 1$. Such states are referred to as stabilizer states[17]. For any Clifford gate $U \in \mathsf{C}_n$ and any stabilizer element $M \in \mathsf{S}_n$, the following holds:

$$UMU^\dagger \in \mathsf{P}_n, \tag{3.51}$$

since any stabilizer M is also an element of the Pauli group. This implies that applying a Clifford gate corresponds to simply updating the n generators of a quantum state. A Clifford gate circuit can therefore be efficiently simulated by tracking the evolution of these n generators, which remain as Pauli strings of length n throughout the computation.

(3) Logical Clifford Gates

Clifford gates are known to have some nice properties when implementing them as logical operations on QECCs. For example, logical Clifford gates, such as the logical S or H gates, are **fold-transversal** on surface codes [90, 91]. Here, fold-transversality is a concept similar to transversability but also allowing permutations of the individual physical qubits. As S and H are fold-transversal, and CNOT being fault-tolerant due to lattice surgery, any logical Clifford gate can be fault-tolerantly implemented on surface codes.

Moreover, there exists a general fault-tolerant construction for any logical Clifford gate on stabilizer codes [92]. Although this construction may not always offer the most efficient implementation for a specific gate in practice, it provides a theoretical guarantee that logical Clifford operations can be performed in a fault-tolerant manner.

(4) Logical Non-Clifford Gates

While Clifford gates have nice properties, a quantum circuit with Clifford gates alone is insufficient for universal quantum computation[18]. We require **non-Clifford gates**, such as the T gate (a Z axis $\pi/4$ rotation gate) or the Toffoli gate (a controlled-controlled-NOT gate). Unfortunately, it turns out that we cannot have a naive implementation of a fault-tolerant non-Clifford gate in the surface code.

Instead, we perform gate teleportation, which is a technique that reduces the problem of gate application to state preparation. With this, a non-Clifford gate can be implemented by preparing a so-called **magic state**, combined with a sequence of Clifford gates and measurements. Here, we define the magic state to be the quantum state $T\left|+\right\rangle$. The gate teleportation circuits for implementing a T gate are shown in Fig. 3.11. In this context, the two qubits $\left|\psi\right\rangle$ and $T\left|+\right\rangle$ are logical qubits encoded in some stabilizer code. Moreover, to ensure gate teleportation is fault-tolerant, preparing the magic state itself must introduce as little noise as possible.

[17]For example, generators $\langle ZZ, XX\rangle$ represent a two-qubit state $\left|\Phi^+\right\rangle = (\left|00\right\rangle + \left|11\right\rangle)/\sqrt{2}$.

[18]In fact, if it were sufficient, by the Gottesman-Knill theorem, quantum computers would be efficiently simulatable with classical computers! It turns out that the complexity class of the Gotteman-Knill procedure is equivalent to $\oplus\mathsf{L}$ [93], implying that it might not even be universal for classical computation.

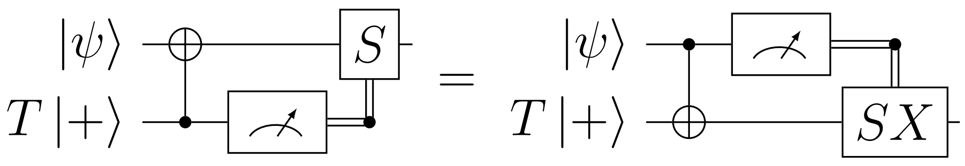


Figure 3.11: Two equivalent quantum circuits implementing a T gate on the state $|\psi\rangle$, using gate teleportation. The right version is more preferred than the left version, as we can continue our computation on a fresh qubit, but when changing the configuration between data qubits and ancilla qubits is difficult, the left version is also chosen.

(5) Magic State Preparation

Previously, I introduced magic state preparation as a necessary subroutine for performing non-Clifford gates, such as the T gate. One might worry that preparing a high-fidelity magic state $T|+\rangle$ inherently requires performing a T gate on the code itself. If this were the case, we would not have changed the problem at all but just added more complications to the process. However, preparing a quantum state is an entirely different problem from applying a gate, and various methods exist to generate a high-fidelity magic state encoded in a QECC.

Typically, magic state preparation consists of two levels: **magic state injection**, followed by **magic state distillation**. Magic state injection is a method to, without any fault-tolerant assurance, encode a noisy magic state into an error-correcting code. There are various proposals of magic state injection that aim to reduce the error as much as possible for surface codes [94, 95, 96, 97]. Then we move on to distillation, where many noisy magic states are refined into a smaller set of high-fidelity magic states. [98, 99, 100, 101]. With this, an encoded, high-fidelity magic state can be prepared, ready to be used for gate teleportation.

Recently, a novel alternative known as magic state cultivation [102] has been proposed. Unlike traditional distillation, cultivation is based on the idea of zero-level distillation, a method to maintain a state's fidelity by performing repeated measurements [103, 104, 105]. Magic state cultivation improves on these ideas by dynamically increasing the code distance of such states, providing an efficient procedure for magic state preparation. In fact, the authors show via numerical simulations that the cost of preparing a high-fidelity magic state is roughly equivalent to performing a logical CNOT gate, drastically reducing the required resources.

(6) Pauli-Based Computation

Another motivation for the distinction between Clifford and non-Clifford gates lies in **Pauli-based computation** (PBC). PBC is a model of universal computation that translates a quantum circuit composed of Clifford gates and non-Clifford gates into a sequence of specific Pauli product measurements, supplemented by magic states [106]. Through

PBC, universal quantum computation on surface codes can be efficiently implemented using lattice surgery and magic states [107, 108].

3.4 What Does It Mean To Be Fault Tolerant?

3.4.1 The Threshold Theorem

So far, I have been describing how to perform various logical operations in a manner that prevents error propagation on encoded qubits, specifically for stabilizer codes. The motivation for constructing such complex gadgets is to demonstrate that errors can be arbitrarily suppressed while maintaining the feasibility of quantum computation, which is proved via the threshold theorem [82, 109, 110, 111, 112], stated as below.

Theorem 3.4 (*Threshold Theorem*)

A quantum circuit comprising T gates can be simulated with a total error probability of at most ϵ using $O(T\,\mathrm{polylog}(T/\epsilon))$ physical gates, provided that the physical error rate p is below a certain threshold value p_{th}.

This theorem proves that arbitrary error suppression is, in principle, achievable, and the overhead induced by error correction does not cancel the quantum advantage offered by algorithms. It also provides a clear goal for experimentalists working towards large-scale, fault-tolerant quantum computing. There are various instances of the threshold theorem, as the proofs rely on different assumptions that can affect the threshold value or the resulting overheads. However, as a general rule of thumb, even when we incorporate stringent assumptions that attempt to model the real world as closely as possible, the threshold theorem holds.

3.4.2 Simulation and Pseudo-Thresholds

Although the threshold theorem is a strong result, it may not accurately reflect the practical behavior of a QECC. This is because theoretical proofs often abstract away many critical implementation details of a code, such as assuming an optimal decoder instead of an actual decoding algorithm. On the other hand, since QECC circuits are primarily composed of Clifford gates, this gives us the motivation to perform simulation using an actual decoder. The motivation for these simulations, in contrast with analytical methods, is detailed in Chapter 5.

For such simulations, we typically estimate the logical error rate and derive the value known as the **pseudo-threshold**. As mentioned, using more physical qubits increases the code distance, improving the error suppression ability. However, this also implies there are more locations where errors can occur compared to codes with smaller distances. Under these circumstances, we can suppose there exists a specific “threshold” for the physical error rate: the physical error rate being small enough that actual error suppression begins to work. This turning point is called the pseudo-threshold region[19]. An example of a pseudo-threshold in a surface code memory experiment, assuming a circuit-level noise model, is shown in Fig 3.12.

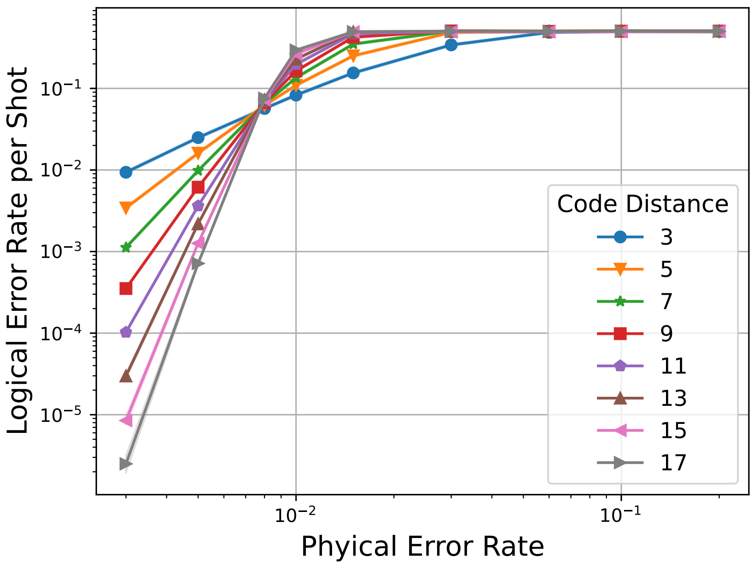


Figure 3.12: An example of a pseudo-threshold simulation. Here, we can observe a pseudo-threshold at a noise strength of $p = 0.8\%$. This simulation is performed on the default unrotated surface code using Stim [113] as the Clifford gate simulator, decoded by PyMatching[20] [80]. These simulation tools will be introduced in the following chapters.

As we can see, for physical error rates higher than $p = 0.8\%$, no error suppression is occurring: increasing code distances constantly worsens the logical error rate. However, for physical error rates smaller than this value, we can see the opposite trend where error correction begins to function. This means that if we build a quantum computer having a physical error rate smaller than the pseudo-threshold error rate, the overall error rate can also be suppressed with the help of an error-correcting code.

Of course, we need to be careful about the assumptions of the error model. For this simulation, we assumed a model in which errors occur after every reset, gate, and measurement location with equal probability p, where p represents the physical error rate plotted on the x-axis. If we consider a more relaxed assumption regarding the error model, the pseudo-threshold region can be much higher. For example, in the code capacity model[21], the pseudo-threshold region for surface codes is around 10% [6].

[19]This is referred to as a pseudo-threshold "region" because the intersection is not a single, well-defined point; rather, it is an interval where the scaling behavior of the error rates shifts.

[20]PyMatching implements a tailored version of the MWPM algorithm for surface codes.

[21]An error model where errors only occur solely on data qubits, and stabilizer measurements or other ancilla-related qubits are noiseless.

Chapter 4

Problem and Proposed Approach

4.1 Motivation: Distributed Quantum Computing

Realizing FTQC for practical applications requires nearly millions of qubits [68]. However, scaling a single quantum device to this level is impractical due to physical limitations, such as the increased noise introduced by densely packing qubits, and engineering challenges, such as qubit wiring bottlenecks. This leads to the idea of splitting computation among smaller nodes that are spatially separated and interconnected to each other by a quantum network, referred to as Distributed Quantum Computing (DQC) [5, 63, 64].

In order to extend quantum computation to a distributed scenario, we must be able to perform non-local CNOT gates[1]. One might suggest achieving this via quantum teleportation: share Bell pairs, teleport a qubit to a remote node, apply a local CNOT, and then teleport it back. However, this round-trip approach consumes two Bell pairs per CNOT, which might sound too costly for such a primitive operation. Fortunately, there exists a non-local CNOT implementation that requires only a single shared Bell pair [114]. This protocol is known as the remote CNOT gate and is illustrated in Fig. 4.1. Therefore,

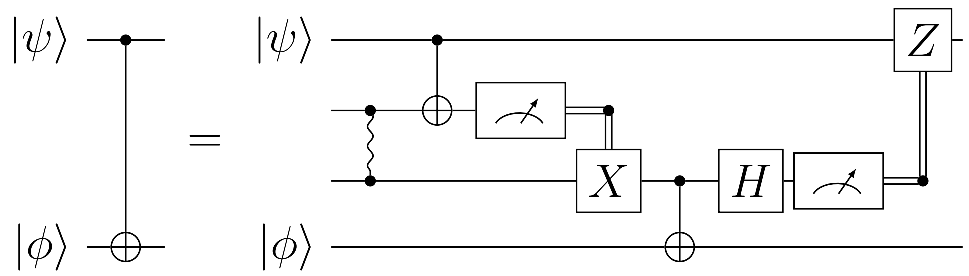


Figure 4.1: Quantum circuit for the remote CNOT gate. The wavy lines between qubits represent a $|\Phi^+\rangle$ state.

provided that the distributed quantum nodes are interconnected via a quantum network

[1]This is because any multi-qubit gate can be decomposed into single-qubit gates and CNOT gates between arbitrary qubits.

that enables the distribution of Bell pairs, DQC is, in principle, achievable.

However, there are many other details to address in realizing a DQC system. For example, the distribution of Bell pairs serves as a major cost when partitioning a large quantum circuit into smaller sub-circuits. Consequently, minimizing this distribution overhead across nodes is necessary. One way is to reduce this circuit distribution problem to the graph partitioning problem and solve that partitioning problem with various heuristics [115, 116]. The network topology, or the connectivity of nodes, also needs heavy consideration, as we would like to minimize latency and loss in designing the interconnect architecture [117, 118]. Benchmarks and performance evaluations [119] of such full-scale modular architectures are also required to serve as a guiding light for choosing directions of future development. Furthermore, mapping fault-tolerant quantum computation and related computational models to this distributed environment is a crucial step for practically running quantum algorithms.

4.2 Importance of Logical Entanglement

When we aim for fault-tolerant distributed quantum computation, the Bell pairs shared between distant nodes must be error-corrected. In other words, generation of **logical entanglement** is a critical requirement to enable practical, scalable quantum computation. However, we do have an issue with generating such logically entangled states. That is, the physical-to-logical interface is often ambiguous, and while there exist methods to generate logical entanglement, it is unclear what is the best method to unify the gap between physical entanglement and logical qubits.

It is worth noting here that error-corrected quantum repeaters [120], typically referred to as 2G repeaters [121], are designed to distribute entanglement in the form of error-corrected codes rather than mere physical entanglement. There also have been prior research that analyzes 2G repeaters speaking in the surface code [122].

In fact, as we will see in the following section, the surface code repeater scheme is practically implementing one of the known methods of logical entanglement distribution, remote lattice surgery.

In our context, I focus on a scenario where a number of physical Bell pairs are distributed among computational nodes, and the objective is to bridge these physical resources to the logical computation space.

4.3 Existing Methods

There have been many proposals for generating logical entanglement, particularly within the context of surface codes. This section provides an overview of two established methods.

(1) Remote Lattice Surgery

One robust way for generating logical entanglement is **remote lattice surgery** [122, 9, 10]. This method is conceptually straightforward. Lattice surgery, as described in the previous chapter, can be used to generate Bell pairs with d rounds of stabilizer measurements. For the remote case, the principle stays the same; we just replace CNOT gates between non-directly connected qubits with the remote CNOT gates presented in the previous section. An example of this remote lattice surgery protocol is shown in Fig. 4.2.

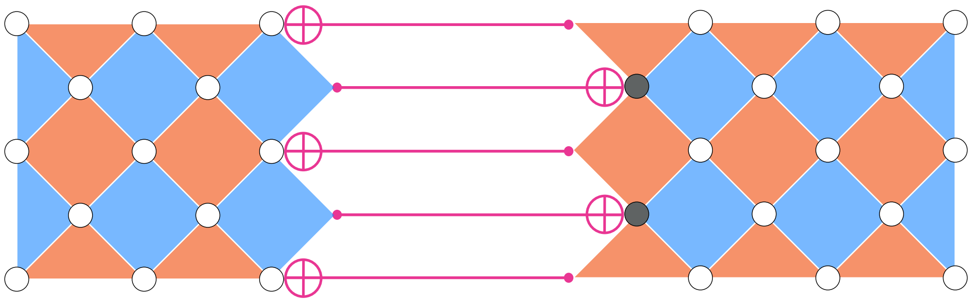


Figure 4.2: An example of remote lattice surgery performed between two unrotated $d = 3$ surface codes. Here, the pink CNOT gates are remote CNOT gates, which are realized using one physical Bell pair and LOCC for each instance. The grey qubits are the intermediate data qubits required for lattice surgery, and they can be located at either side of the connection as long as the ancilla configuration remains symmetric.

For a surface code of code distance d, a single round of remote stabilizer measurement requires $2d - 1$ or d Bell pairs, depending on whether the code is unrotated or rotated. Consequently, repeating d rounds of stabilizer measurements to account for measurement errors, $d(2d - 1)$ or d^2 Bell pairs are required for remote lattice surgery. Moreover, this method is based on straightforward lattice surgery. Therefore, it is well-suited for distributed implementations of architectures where computation is performed exclusively via magic state preparation and lattice surgery.

(2) State Injection

Another approach utilizes **state injection** assisted with entanglement distillation to encode physical entanglement into the logical space [7, 8]. By adapting protocols originally designed for magic state injection, this approach directly converts a physical Bell pair into a logical one. For instance, the injection protocol based on expanding a surface code patch from a physical qubit [95] can be effectively adapted for this purpose.

In terms of resource consumption, state injection requires only a single physical Bell pair per attempt, making it relatively efficient compared to remote lattice surgery. However, because the injected logical Bell pair typically retains too much noise for immediate FTQC operations, we need to post-process the injected state using entanglement distillation. Furthermore, state injection protocols rely on post-selection, making it a probabilistic process. Success is not guaranteed in every attempt, and enforcing overly stringent post-selection conditions can easily negate the resource advantage relative to remote lattice surgery.

A recent proposal named entanglement boosting [123] improves upon the injection scheme in several key ways. The authors report that entanglement boosting can suppress logical Bell pair error rates to practical levels, requiring only approximately 100 noisy Bell pairs and relatively small local computation space[2]. These improvements are attributed to modifying the injection protocol to a cultivation-like protocol and the use of a soft-output decoder that incorporates the complementary gap [124] for post-selection of the noisy logical state. However, such advances also come with the cost of spending d_{Bell}^2 Bell pairs for a single round of logical Bell pair generation.

4.4 Proposal: Entanglement Ejection

In this thesis, I introduce a novel proposal for generating logical entanglement, **entanglement ejection**[3]. Entanglement ejection, which I will typically refer to as "ejection," is a method that generates physical–logical entanglement on surface codes (see step five of Fig. 4.3) and extends that to logical entanglement via entanglement swapping on the physical qubits.

One might think that generating this state does involve state injection, as it looks as if one side of a locally generated physical Bell pair is encoded into a logical surface code patch. However, partial injection keeps the other qubit consisting of the Bell pair not encoded for a very long period of time, which makes it subject to a large amount of noise. Instead, we generate this physical–logical entanglement as follows.

1. Prepare a surface code of distance d composed with $(2d+1)\times(2d-1)$ physical qubits, including the stabilizer ancillas, and run d rounds of stabilizer measurements.
2. Perform lattice surgery, measuring out the data qubits on the second-to-the-rightmost column in the surface code patch.
3. We now have entanglement between a surface code patch of distance d and a repetition code consisting of $2d-1$ physical qubits. Run d rounds of stabilizer measurements on the two entangled logical qubits.
4. Shrink the logical state of the repetition code to a single physical qubit by measuring out all other data qubits, preparing the designated qubit for entanglement swapping with an incoming Bell pair.

[2]Note that assumptions regarding noise models and distillation strategies vary across the literature. Such numerical results should be interpreted within the context of their specific parameters.

[3]The concept of entanglement ejection was originally conceived by Associate Professor Shota Nagayama. I coined the term for the protocol and performed the detailed analysis and evaluation.

5. Based on the measurement results of the ancilla and data qubits, move into the correct Pauli frame. This results in a physical–logical entangled state.

An example of generating physical–logical entanglement using the $d = 3$ unrotated surface code is shown in Fig. 4.3.

With this physical–logical entangled state, we can proceed to generate logical entanglement. We simply perform entanglement swapping between the physical halves of a shared Bell pair and the local "ejected" states, as shown in Fig. 4.4. Although this process is efficient, consuming only one Bell pair, the resulting logical Bell pair is noisy. This is evident from the fact that the protocol forcefully collapses a logical qubit to a partially physical qubit and performs noisy measurements on it. Therefore, entanglement distillation is required as a post-processing step.

By careful analysis on the protocol, one might notice that entanglement ejection has a rate-adaptive feature. Once the logical state is reduced to a repetition code, it can already begin entanglement swapping as long as we have $2d-1$ Bell pairs, as entanglement swapping can be transversally performed on repetition codes. By adaptively shrinking our repetition code to an intermediate k-qubit repetition code, our method can dynamically adjust to the network's physical Bell pair distribution rate. We can also view this as a variant of performing entanglement distillation via stabilizer codes [125], as many physical Bell pairs are combined and consumed into a single higher-fidelity logical surface code Bell pair.

There exist various similarities and differences between ejection and injection. For example, they both include a fragile step during their operations, involving direct operations on a single physical qubit. This can be conceived as a non-fault tolerant behavior, because a single error on this physical qubit can propagate, potentially compromising the entire state. However, injection and ejection also have fundamentally opposite traits, such as the direction of their protocols. In state injection, we initialize a physical state and constructively grow it into a logical state. Conversely, in entanglement ejection, we begin with a logical state and shrink it down to a physical state, allowing it to interact with physical Bell pairs. Another key difference between these two protocols is that entanglement ejection is deterministic, whereas state injection is probabilistic. Such differences can play a crucial role in the overall error suppression capabilities of these two methods.

4.5 Research Objective

For achieving fault-tolerant distributed quantum computing, it is necessary to generate error-corrected logical entanglement between distant devices. We have seen various methods that achieve this goal, where all of them have distinct trends.

In the near term, due to slow and low-quality interconnects between quantum devices, remote lattice surgery may lack access to sufficient high-quality Bell pairs. State injection is an alternative method that addresses such concerns; however, it is probabilistic and its

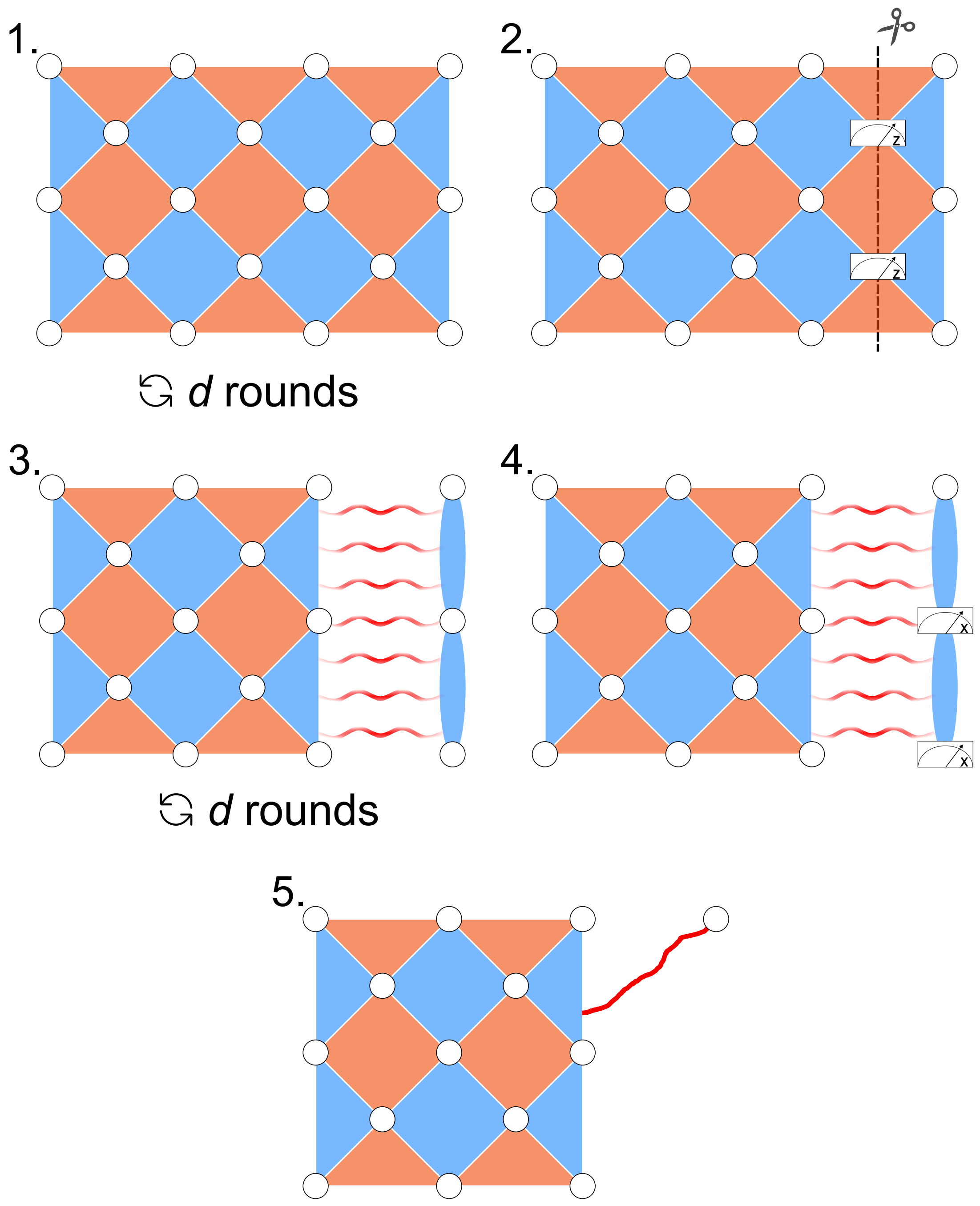


Figure 4.3: An example of entanglement ejection on the $d = 3$ unrotated surface code. (1) Initialization of a $(2d + 1) \times (2d - 1)$ qubit surface code with d rounds of stabilizer measurements. (2) Lattice surgery split operation entangling the surface code (left) with a bit-repetition code (right). The intermediate data qubits are measured in the Z basis in order to not interfere with the adjacent Z stabilizers. (3) d rounds of stabilizer measurements for the surface code and the bit-repetition code. (4) Shrinking the bit-repetition code by measuring its data qubits in the X basis. This basis is specifically chosen to preserve the codeword along the repetition code (opposite stabilizer configurations would require Z basis measurement instead). (5) Resulting state as an entanglement between the surface code and a physical qubit.

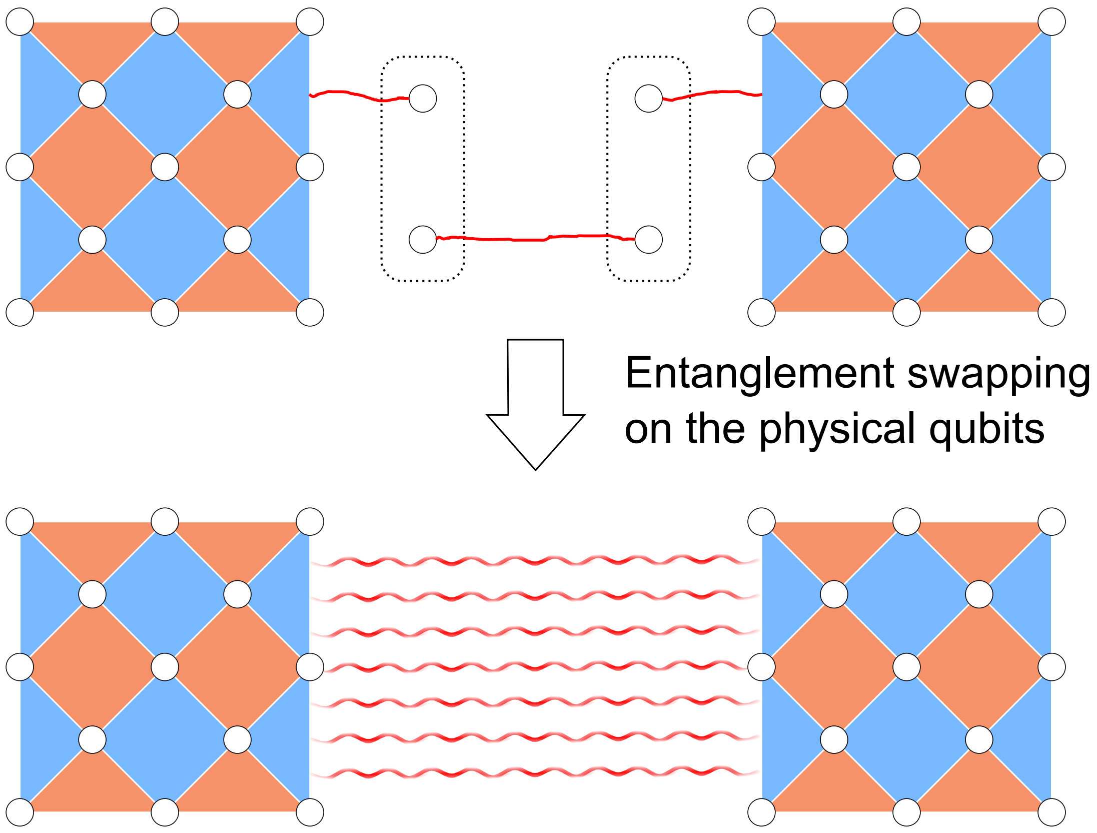


Figure 4.4: Generating logical entanglement via ejection and swapping. Here, entanglement swapping is performed between the physical component, the “ejected” state, and the physical halves of a non-local Bell pair.

overall structure might not be optimal for this task. As an alternative solution, I have introduced entanglement ejection, a deterministic, single Bell pair consuming protocol. Tab. 4.1 summarizes the characteristics of the three methods under investigation.

	State Injection	**Remote L.S.**	**Entanglement Ejection**
Deterministic?	No	Yes	Yes
Fault-tolerant?	No	Yes	No
Required Bell pairs	$1/p_{\text{success}}$	$d(2d-1)$	$1(\sim 2d-1)$

Table 4.1: Comparison among three major methods to generate logical entanglement. p_{success} is the success probability for state injection. Note that the required number of Bell pairs is adjusted for the unrotated surface code. For the rotated surface code, replace $2d-1$ with d.

Such differences among the three protocols lead to the intuition that there is a tradeoff between resource consumption and the residual error in the final logical state. This raises the question: *which method is optimal?* In the following chapters, I answer this question by formalizing a framework to analyze which is the "best" approach and performing numerical simulations.

Chapter 5

Methodology

In this chapter, I discuss the methodology used to evaluate the performance of our proposal. Note that this thesis focuses entirely on theoretical work rather than physical experiments. Implementation on physical hardware is beyond the scope of this study and is left for future investigation, as executing complex DQC protocols remains a significant challenge for current state-of-the-art devices.

5.1 Simulation of FTQC protocols

5.1.1 Simulation vs. Analytics

How should we evaluate the effectiveness of a specific FTQC protocol? Broadly, there are two primary methodologies: analytics and numerical simulation.

I refer to analytical methods as approaches based on mathematical proofs to establish the asymptotic behavior of a protocol. We have previously seen the importance of such methods for proving the threshold theorem, which guarantees that physical errors can be suppressed arbitrarily if they occur below a certain probability threshold. This provides assurance that, as long as we reduce qubit noise, fault-tolerant quantum computation will be eventually possible. However, these proofs often abstract away critical implementation details, such as the complex and heuristic nature of specific decoding algorithms, and also typically rely on inequalities that yield conservative worst-case predictions.

In contrast, simulation-based methods often provide a more realistic (albeit potentially optimistic) evaluation of performance. Moreover, mathematical modeling of topological codes like surface codes remains challenging due to the large number of qubits involved. This is because it is difficult to track exactly how errors propagate through the code and to bound them mathematically, and instead, running simulations allows us to observe the direct behavior. Furthermore, our primary interest lies in the practical, finite-qubit regime rather than asymptotic limits, as our goal is to deploy these protocols in real DQC environments. For example, we can estimate the logical error rate, the probability of a logical fault occurring after error correction, via Monte Carlo simulations.

We must acknowledge that there exist simplifications and assumptions that can degrade the quality of the prediction and, in some cases, can completely diverge from real-world behavior. Nevertheless, given the complexity of the proposed protocols, I take a simulation-based approach to evaluate the logical entanglement generation methods described in this thesis.

5.1.2 Choice of Simulator

Based on the Gottesman-Knill theorem, quantum computation can be simulated in polynomial time as long as operations are restricted to the Clifford group. Since the circuits that appear when dealing with stabilizer codes are mostly composed of Clifford gates, they can be efficiently simulated. Even within FTQC protocols that require non-Clifford resources, Clifford states such as $|+\rangle$ or $S|+\rangle$ are frequently employed as proxies for validation and benchmarking [97, 102].

Various simulators have been implemented based on the Gottesman-Knill theorem, including the CHP (CNOT-Hadamard-Phase) simulator [93] and the graph state-based simulator [126]. The mechanism used by the graph-based simulator is also applied for quantum network simulators such as QuISP [127], due to its ease of dynamically adding and removing qubits from the overall system.

Today, Stim [113] is generally regarded as the standard for simulating Clifford circuits due to its superior simulation speed and native support for various error correction mechanisms such as decoding. Furthermore, it offers visualization support that is particularly well-suited for topological codes, such as the surface code. Due to such benefits, Stim is used as the main simulator for implementing and analyzing entanglement ejection, state injection, and remote lattice surgery.

5.1.3 Stim and the Detector Error Model

Stim distinguishes itself as one of the fastest Clifford gate simulators due to its highly optimized structure. The efficiency can be credited to the detector error model, which was first implemented in Stim and later explored theoretically [128]. The detectors are defined as the parity between syndrome measurement results, where such values must always be equivalent if there is no error within the circuit. Stim evaluates how probabilistically inserted errors propagate throughout a circuit and outputs a detector graph, which includes the information of which detectors had a parity mismatch. Using this detector graph, the problem of decoding an error-correcting code can be translated to the problem of estimating the most likely errors consistent with the observed detector data. With this model, users can test various decoders, and then determine the logical error rate for a specific circuit-decoder combination.

Moreover, Stim's circuit visualization suite offers an excellent interface for analyzing error-correcting codes. It is not limited to canonical circuit diagrams (such as the timeline diagram in Fig. 5.1), but also supports 2D-grid visualizations where operations can be observed time-step by time-step. Figure 5.2a illustrates this diagram known as the timeslice diagram. Furthermore, the detector slice (detslice) diagram allows us to directly observe the detector error model in a grid representation, as shown in Fig. 5.2b. The detector graph is shown in Fig. 5.3. For these examples, the quantum circuit represents an unrotated surface code of distance $d = 3$ with three rounds of stabilizer measurements. Such visualization features are valuable when designing and verifying topological codes and their associated protocols.

5.1.4 Choice of Decoder

For simulation, I will use the standard decoder for surface codes, PyMatching [80]. PyMatching implements an optimized version of the Minimum-Weight-Perfect-Matching (MWPM) algorithm, as introduced in Chapter 3, which has ideal properties for surface codes. Moreover, it has native support for the detector graph produced by Stim. However, PyMatching might not be the best-suited decoder for analyzing logical entanglement, as entanglement boosting [123] showed that a soft-output decoder that considers the complementary gap [124] does enhance the post-selection performance on the injected state. Regardless of this fact, as this thesis is not intended to highlight the difference between decoding strategies, I will use PyMatching for all logical Bell pair generation methods and leave improvement of decoders as future work.

5.2 Simulation Settings

I utilize the unrotated surface code for the simulations. While unrotated surface codes exhibit a slightly lower encoding rate than the rotated counterparts for a fixed logical error rate target [129], I select this version for its implementation simplicity and native compatibility with the state injection protocol [95], which serves as our comparative baseline.

5.2.1 Details of Logical Bell Pair Generating Protocols

(1) State Injection

For the injection-based protocol, I employ the magic state injection protocol by Li [95]. To adapt this method for Bell pair injection, I generate distributed Bell pairs at the start of the protocol and grow our surface code from those physical qubits.

A key feature of the Li injection scheme is the ability to reduce the logical error rate by tightening the post-selection criteria. This is controlled by the parameter d_{inject}, which defines the size of the injection region where post-selection is performed. A larger d_{inject}

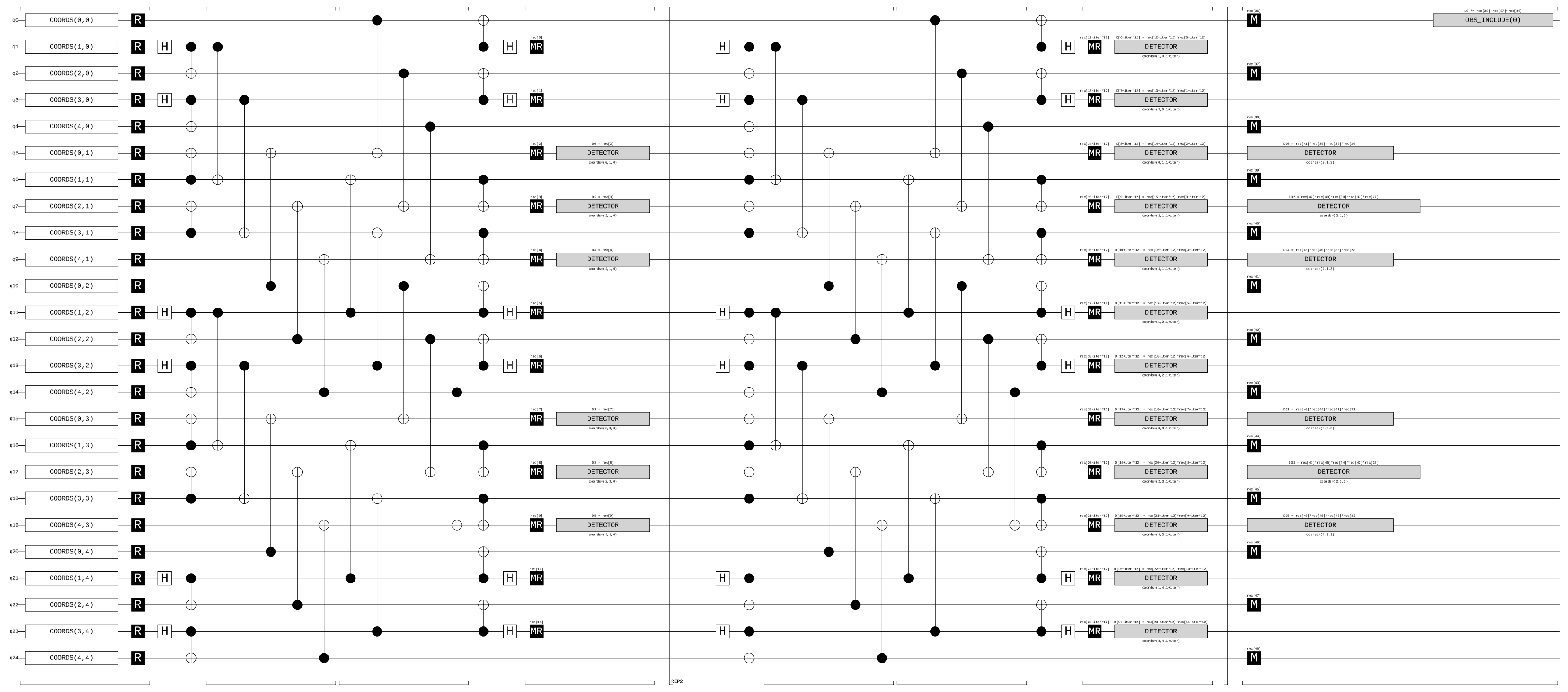
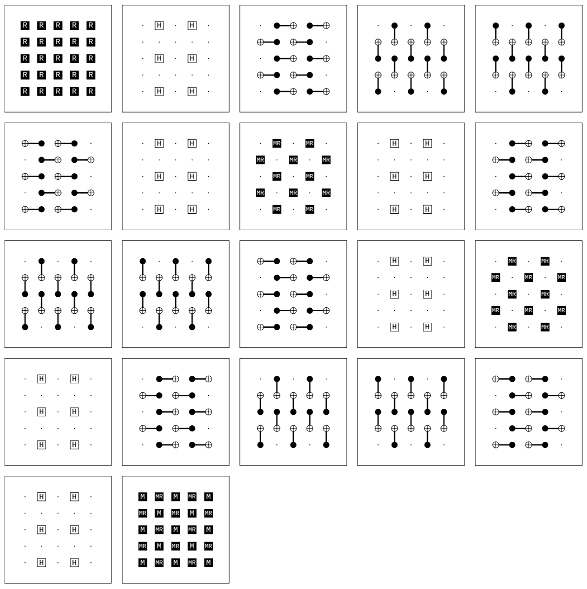


Figure 5.1: An example of a surface code timeline circuit in Stim. R represents resetting a qubit to $|0\rangle$, and M denotes a Z basis measurement. MR is a combination of the measurement M followed immediately by the reset R. DETECTOR indicates a detector declaration, which checks the parity of the measurement results listed on top of the box. COORDS annotations specify the spatial coordinates of qubits or detectors in a 2D grid, allowing the translation of the circuit into timeslice or detslice visualizations.


(a) Timeslice diagram

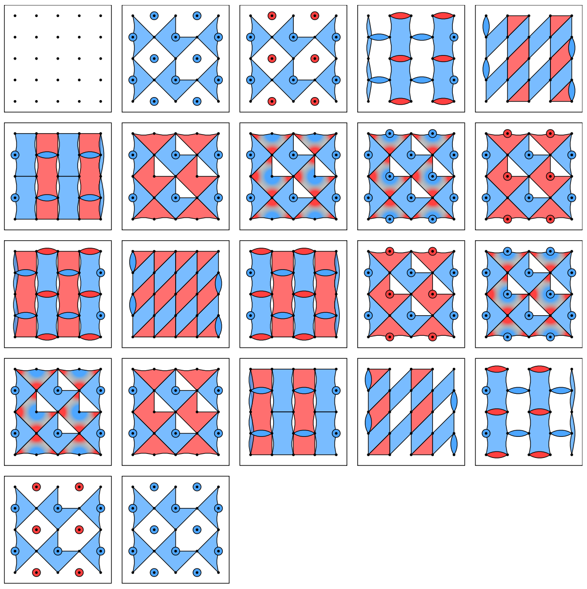

(b) Detslice diagram

Figure 5.2: Various visualizations of the surface code circuit from Fig. 5.1. (a) A timeslice diagram showing the spatial arrangement of qubits and operations at a specific time step. (b) A detslice diagram highlighting the detectors active during a specific time step, with the red and blue colors indicating X or Z basis checks, respectively.

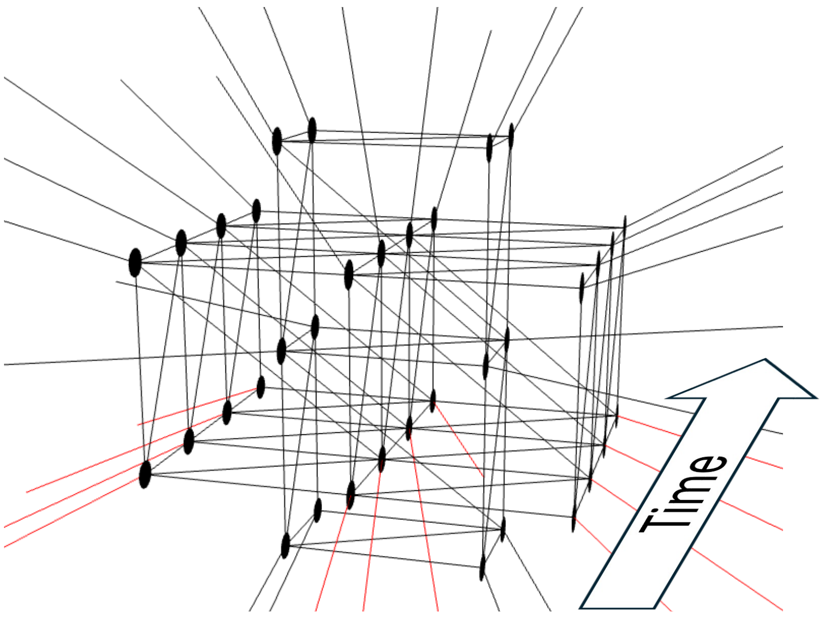


Figure 5.3: A detector graph for the surface code circuit, with uniform errors applied to each operation. The nodes represent a detector, and the edges are weighted edges that represent the probability of errors between the detectors.

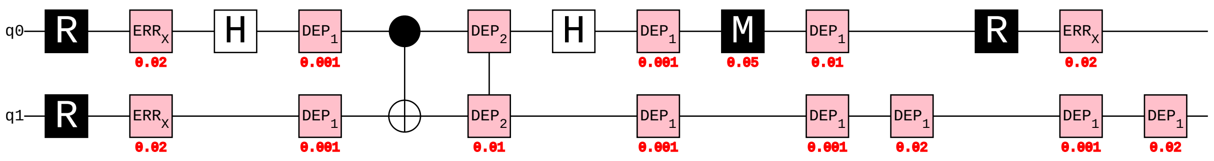


Figure 5.4: An example of a quantum circuit being subjected to an SI1000 noise. Here, the sample circuit is implementing a subroutine known as the Hadamard test for $U = X$. This figure is generated by the visualization suite of Stim. Here, the noise parameter is set to $p = 0.01$. We can observe that during initialization and measurement, idle qubits are subjected to a fair amount of depolarizing noise representing the long waiting time.

yields a logical state with a lower error rate at the cost of a high discard rate; conversely, a smaller d_{inject} improves the yield but suffers a high logical error rate. In our simulations, I test for injection instances of $d_{\text{inject}} = 3, 4, 5, 6, 7, 8, 9$.

(2) Remote Lattice Surgery

For remote lattice surgery, I implement a quantum circuit performing lattice surgery between surface codes, replacing CNOTs with remote CNOTs for qubits in the seam. As previously mentioned, the remote CNOT gate is implemented as the standard proposal by Eisert et al [114]. Furthermore, I assume that the physical Bell pairs are inserted right before performing the remote CNOT gates, so they are not subject to decoherence for a large amount of time.

(3) Entanglement Ejection

For entanglement ejection, the ejected states at both nodes are generated in parallel. The physical Bell pair is generated only after the ejected state (physical–logical in-node entanglement) has been prepared at both sides. This timing strategy helps reduce the accumulation of decoherence error on the physical pair.

5.2.2 Noise Models

(1) Circuit Noise Model

The circuit-level noise model I apply in this thesis is called the SI1000 model [130]. SI1000 stands for Superconducting-Inspired 1000, and as the name suggests, this model mimics behavior typical to superconducting qubits. For example, measurements and qubit initialization are noisier than applying physical gates, and take a relatively long time. Features of the SI1000 model are summarized in Tab. 5.1, and an example of applying this error model to a sample circuit is shown in Fig. 5.4.

It is important to note that for DQC protocols, the SI1000 model represents a particularly stringent requirement, as DQC protocols involve a fair amount of measurement

Ideal Operation	**Noisy Operation Under SI1000**
One-qubit Clifford gate	1. Ideal one-qubit Clifford gate 2. One-qubit depolarizing channel ($p/10$)
Two-qubit Clifford gate	1. Ideal two-qubit Clifford gate 2. Two-qubit depolarizing channel (p)
Reset ($\|0\rangle$)	1. Reset qubit to $\|0\rangle$ 2. Bit-flip channel ($2p$)
Measurement (Z)	1. Bit-flip channel ($5p$) 2. Ideal Z measurement 3. One-qubit depolarizing channel (p)
Idling	One-qubit depolarizing channel ($p/10$)
Resonator idling	One-qubit depolarizing channel ($2p$)

Table 5.1: Specifications of the SI1000 noise model. Here, p is the noise strength of the model. We ensure $0 \leq p < 0.125$ for a reasonable error model, as the parameter for the bit-flip channel applied before measurement must be less than 0.5. *Idling* refers to a qubit waiting during unitary operations on other qubits. *Resonator Idling* refers to waiting during measurement or reset operations on other qubits.

on Bell pairs and other qubits. However, as we do not typically specify the exact physical implementation of our logical entanglement generation schemes, adopting a relatively conservative evaluation of the errors is a reasonable choice.

(2) Bell Pair Noise Model

The ideal physical Bell pair I use in my simulations is the phi-plus state[1] $|\Phi^+\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$. To model the noisy behavior of such physical Bell pairs, I first prepare the ideal state and then apply a two-qubit depolarizing channel of channel parameter p. This is because the resulting fidelity between the noisy Bell pair and the ideal state corresponds exactly to $F = 1 - p$, allowing us to directly map the simulation parameter p to experimental fidelity metrics.

This relationship is analytically derived as follows. The action of a two-qubit depolarizing channel $\mathcal{E}_{\text{dep}}$ on $|\Phi^+\rangle$ is defined as

$$\mathcal{E}_{\text{dep}}(|\Phi^+\rangle) = \left(1 - \frac{4p}{3}\right) |\Phi^+\rangle \langle\Phi^+| + \frac{4p}{3}\frac{I}{4}. \tag{5.1}$$

[1]The choice of Bell pair is arbitrary, and any of the four Bell states can be used. This is because they are equivalent up to local Pauli corrections, which can be tracked by the Pauli frame.

Therefore, the fidelity between $|\Phi^+\rangle$ and $\mathcal{E}_{\text{dep}}(|\Phi^+\rangle)$ is

$$
\begin{aligned}
F(|\Phi^+\rangle, \mathcal{E}_{\text{dep}}(|\Phi^+\rangle)) &= \langle\Phi^+| \left[\left(1 - \frac{4p}{3}\right)|\Phi^+\rangle\langle\Phi^+| + \frac{4p}{3}\frac{I}{4}\right]|\Phi^+\rangle, && (5.2)\\
&= \left(1 - \frac{4p}{3}\right)|\langle\Phi^+|\Phi^+\rangle|^2 + \frac{4p}{3}\frac{1}{4}\langle\Phi^+|\Phi^+\rangle, && (5.3)\\
&= 1 - \frac{4p}{3} + \frac{p}{3}, && (5.4)\\
&= 1 - p. && (5.5)
\end{aligned}
$$

(3) Error Parameters

The error parameters for the noise models used in our simulations are defined as follows. For the circuit-level noise, I sweep the noise strength p of the SI1000 model within the range $[5.0\times10^{-4}, 1.0\times10^{-2}]$. For the Bell pair fidelity F, I vary the value within $[0.86, 0.96]$ to align with state-of-the-art experimental results [131].

5.2.3 Experiment One: Logical Error Rates

As previously mentioned in Chapter 3, simulations allow us to observe the logical error rate and the pseudo-threshold. By evaluating logical error rates across varying code distances, we can identify the pseudo-threshold behavior of the selected FTQC protocol. In this experiment, I generate logical Bell pairs using state injection, remote lattice surgery, and entanglement ejection. The simulations are conducted using the parameters summarized in Tab. 5.2.

Parameter	**Value / Setting**
Fixed Parameters	
Error-Correcting Code	Unrotated Surface Code
Circuit Noise Model	SI1000
Variable Parameters	
Code Distance (d)	$d \in \{3, 5, 7, 9, 11, 13, 15\}$
Noise Strength (p)	$p \in [5.0 \times 10^{-5}, 1.0 \times 10^{-2}]$
Bell Pair Fidelity (F)	$F \in \{0.86, 0.96\}$
Protocol-Specific Parameters	
Injecting Distance (d_{inject})	$d_{\text{inject}} = 7$

Table 5.2: Summary of simulation parameters for experiment one. The noise model and code type are fixed. We sweep the code distance d, noise strength p, and fidelity F. The injecting distance $d_{\text{inject}} = 7$ is selected as this is the canonical example presented in the original paper [95].

5.2.4 Experiment Two: "Achievable" Logical Error Rates

Experiment one evaluates the logical error rates and pseudo-threshold behaviors of the three methods. However, focusing only on how much we can suppress the error does not serve as a fair comparison, as each method has entirely different characteristics. Theoretically, error rates could be arbitrarily suppressed assuming infinite computational time and space; however, relying on such unbounded resources would cancel out the computational advantage of quantum algorithms, making such assumptions impractical.

When evaluating fault-tolerant protocols, it is therefore necessary to consider the resource constraints. Resource is a broad concept which can include computation space, computation time, or, for distributed scenarios, the rate of obtaining physical distributed Bell pairs. For our case of logical entanglement generation, we need a good metric to capture the behavior of the logical error rate, the entanglement distribution cost, and the local operation space-time usage.

For evaluating entanglement ejection and its counterparts, I introduce the **achievable logical error rate** as a normalized comparative metric. This metric is defined through the following procedure, also shown in Fig. 5.5. I first allocate a fixed pool of physical Bell pairs, and estimate how many (noisy) logical Bell pairs can be produced by each method. Then, entanglement distillation on these pairs is performed to produce a single, high-fidelity logical Bell pair. The final logical error rate of this distilled pair is the achievable logical error rate.

This metric enables a fair comparison of logical entanglement generation protocols by accounting for their distinct Bell pair consumption behaviors. In fact, within the context of distributed quantum computing, the generation and distribution of physical entanglement acts as the primary bottleneck for overall system performance [132]. Therefore, logical error rates while normalizing Bell pair consumption is a necessary evaluation for a logical entanglement distribution protocol.

For this simulation, we need a normalized code distance for the target state. I set the code distance of the logical Bell pair to $d = 25$, as this level of protection is required for achieving quantum advantage in critical applications, such as factoring RSA-2048 [68] or utilizing the QAOA algorithm to solve non-toy optimization problems [133].

(1) Resource Constraints

Since the three approaches differ in their physical Bell pair consumption, I normalize the resource budget based on remote lattice surgery, as it is the most demanding method. Therefore, I assume a fixed pool of $d(2d-1)$ physical Bell pairs to be provided at first. Under this constraint, the yield of the logical Bell pairs can be calculated as shown in Tab. 5.3.

Method	**Expected number of logical Bell pairs**
State Injection	$p_{\text{success}} \cdot d(2d-1)$
Remote Lattice Surgery	1
Entanglement Ejection	$d(2d-1)$

Table 5.3: Comparison of the number of yielded logical Bell pairs under a fixed budget of $d(2d-1)$ physical Bell pairs. p_{success} is the post-selection success probability for state injection.

(2) Incorporating Entanglement Distillation to Logical Error Rates

Entanglement distillation protocols are typically quantified by analyzing the input/output fidelity of the Bell pairs. The simulation results obtained from Stim are in the format of logical error rates, but thankfully, the fidelity F and the logical error rate p_L can be converted one-to-one, as

$$F = 1 - p_L. \tag{5.6}$$

This is explained as follows. The logical error rate p_L is the probability of a logical state $|\psi\rangle$ being projected to an orthogonal[2] logical state $|\psi^{\perp}\rangle$. Therefore, a quantum state ρ that has a logical error rate of p_L can be written as

$$\rho = (1-p_L)\,|\psi\rangle\langle\psi| + p_L\,|\psi^{\perp}\rangle\langle\psi^{\perp}|\,. \tag{5.7}$$

Here, the fidelity between ρ and $|\psi\rangle$ is calculated as follows:

$$\begin{aligned}
F(|\psi\rangle, \rho) &= \langle\psi|\,\rho\,|\psi\rangle\,, && (5.8)\\
&= \langle\psi|\left((1-p_L)\,|\psi\rangle\langle\psi| + p_L\,|\psi^{\perp}\rangle\langle\psi^{\perp}|\right)|\psi\rangle\,, && (5.9)\\
&= (1-p_L)|\langle\psi|\psi\rangle|^2 + p_L|\langle\psi|\psi^{\perp}\rangle|^2, && (5.10)\\
&= 1 - p_L. && (5.11)
\end{aligned}$$

Hence, $F = 1 - p_L$ can be used as the input fidelity for distillation.

(3) Distillation Strategies

I choose the original BBPSSW protocol [11] as the actual entanglement distillation method applied to the logical Bell pairs. The quantum operations for distillation are assumed to be noiseless, as they are performed on fault-tolerant logical qubits where gate errors are negligible compared to the infidelity of the input pairs. Therefore, the output fidelity and success probability can be analytically derived instead of resorting to

[2]This is due to the error-discretization feature of QECCs that projects the resulting state to a logical codeword state. Due to orthogonality, $\langle\psi|\psi^{\perp}\rangle = 0$.

circuit-level simulation, as follows:

$$p_{\text{success}} = F^2 + \frac{2}{3}F(1-F) + \frac{5}{9}(1-F)^2, \tag{5.12}$$

$$F_{\text{out}} = \frac{F^2 + \frac{1}{9}(1-F)^2}{F^2 + \frac{2}{3}F(1-F) + \frac{5}{9}(1-F)^2}. \tag{5.13}$$

By applying this fidelity evolution recursively, I calculate the improved fidelity and the required number of logical Bell pairs. Once the resource cost exceeds the available supply of logical Bell pairs, further distillation is aborted, and the final logical error rate is reported as the achievable logical error rate.

(4) Summary of Simulation Two

In summary, I perform simulations of generating logical Bell pairs by state injection, entanglement ejection, and remote lattice surgery, considering the achievable logical error rate. For state injection, I test multiple instances among the injecting distances $d_{\text{inject}} = 3, 4, 5, 6, 7, 8, 9$. The parameters and conditions for simulation are shown in Tab. 5.4. I then calculate and compare the achievable logical error rate considering the number

Parameter	**Value/Setting**
Fixed Parameters	
Error-Correcting Code	Unrotated Surface Code $d = 25$
Circuit Noise Model	SI1000
Variable Parameters	
Noise Strength (p)	$p \in [5.0 \times 10^{-5}, 1.0 \times 10^{-2}]$
Bell Pair Fidelity (F)	$F \in [0.86, 0.96]$
Protocol-Specific Parameters	
Injecting Distance (d_{inject})	$d_{\text{inject}} \in \{3, 4, 5, 6, 7, 8, 9\}$

Table 5.4: Simulation parameters for experiment two. The code distance is fixed. We sweep the noise parameters, and for the injection protocol, the injecting distance d_{inject}.

of yielded logical Bell pairs for each protocol, shown in Tab. 5.3, utilizing BBPSSW distillation, where the overall process is shown in Fig. 5.5.

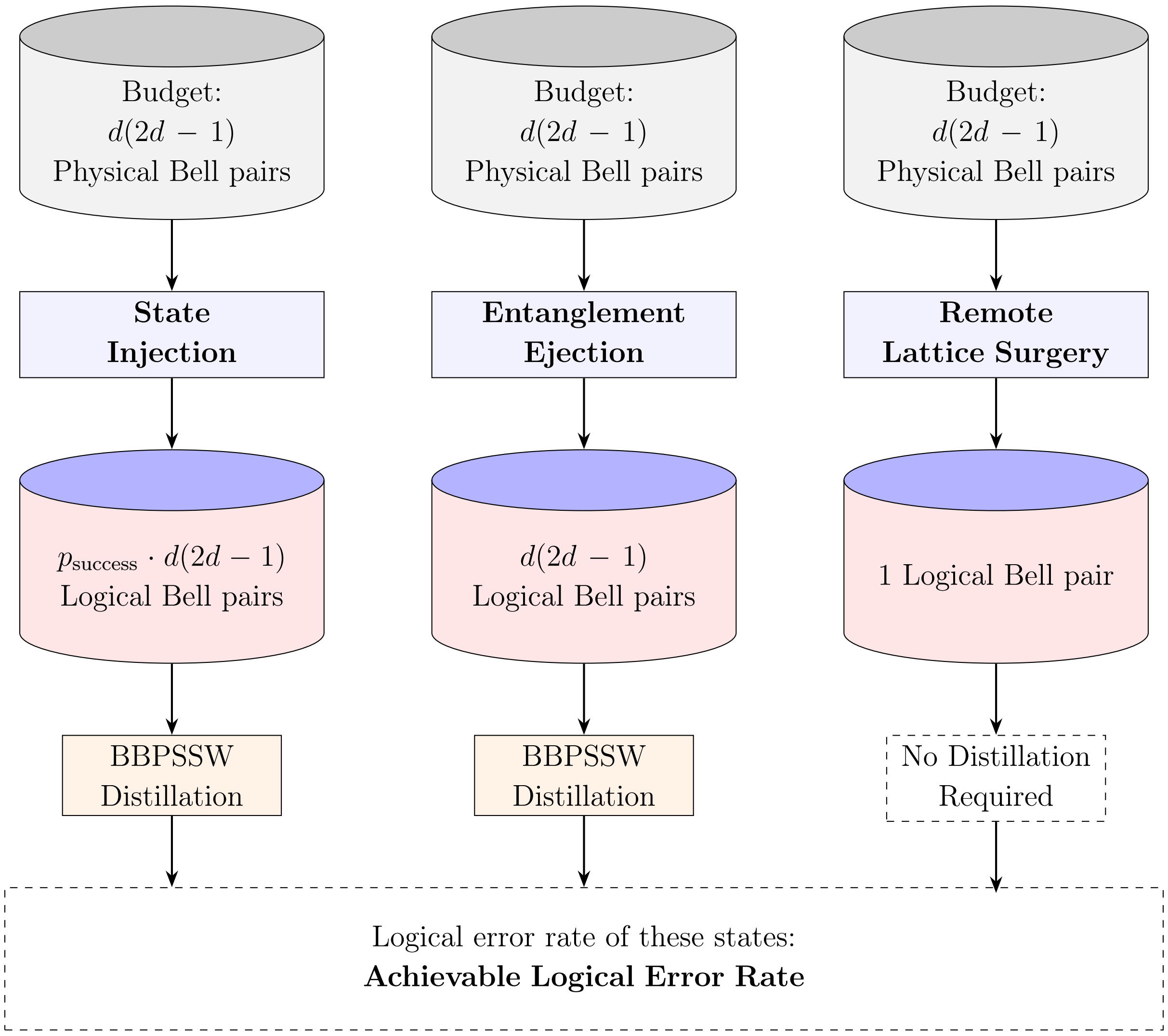


Figure 5.5: Deriving the achievable logical error rate. Each logical Bell pair generation method is initially allocated $d(2d-1)$ physical Bell pairs. State injection generates $p_{\text{success}} \cdot d(2d-1)$ logical pairs, entanglement ejection generates $d(2d-1)$ pairs, and remote lattice surgery produces a single logical Bell pair. For injection and ejection, all available logical Bell pairs undergo entanglement distillation via the BBPSSW protocol. The achievable logical error rate is then determined by the logical error rates of the resulting final states.

Chapter 6

Evaluation and Discussion

6.1 Implementation Details

The logical error rates are calculated as the number of failed decoding instances divided by the total number of accepted (non-discarded) simulation instances. To achieve statistical confidence in our results, we performed 10^7 rounds of simulations per condition. However, to optimize computational resources, we terminate the simulation early if the accumulated number of logical errors exceeds 10,000. This is to avoid excessive computing in high-error regimes where convergence is quickly reached. All simulations were conducted under the conditions detailed in Tab. 6.1.

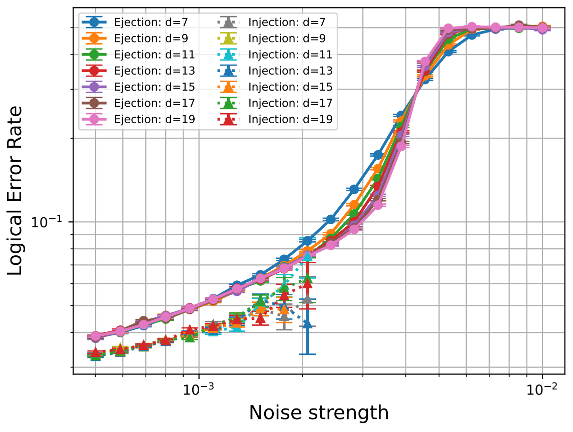
Component	**Specification**
OS	Ubuntu 24.04.2 LTS
CPU	4×AMD EPYC 9684X (96 cores each, 384 cores in total) 300 cores were allocated for this simulation.
Memory (RAM)	1.48 TB
Python	Version 3.12.3
Stim	Version 1.13.0
PyMatching	Version 2.2.1

Table 6.1: Computational resources and software versions used for the simulation.

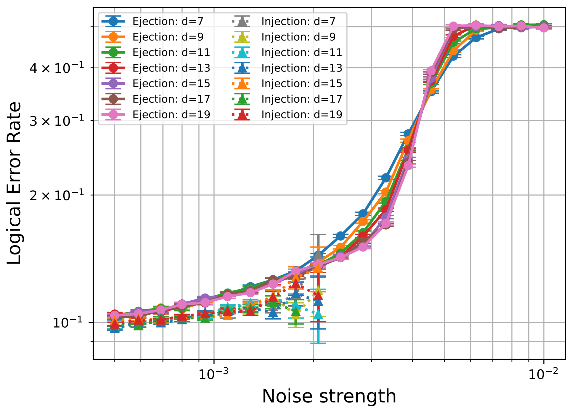
6.2 Results: Experiment One

Fig. 6.1 presents the logical error rates for the three different protocols: state injection, remote lattice surgery, and entanglement ejection. The results are plotted as a function of the noise strength p under the SI1000 model, where each subfigure represents an input Bell pair fidelity of $F = 0.96$ and $F = 0.86$.

Injection/Ejection

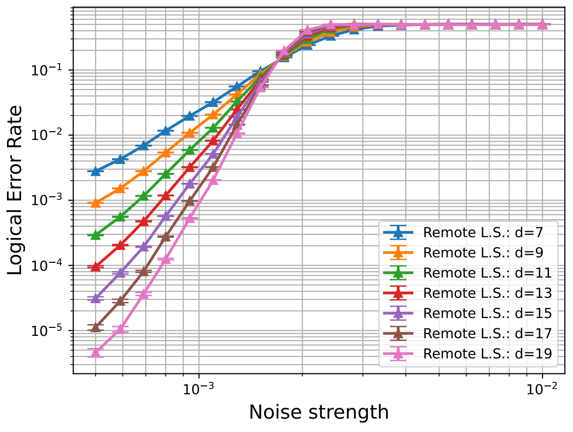


(a) Input Bell pair fidelity $F = 0.96$.

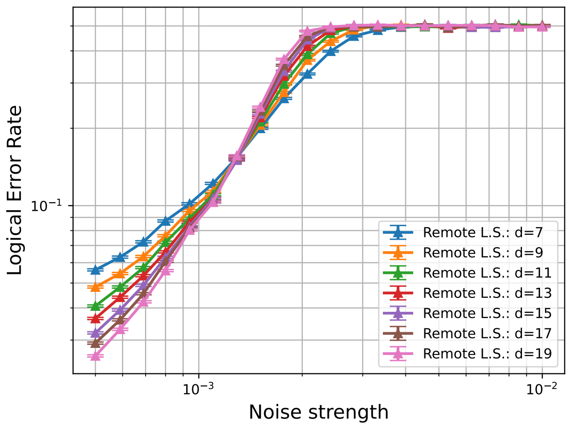


(b) Input Bell pair fidelity $F = 0.86$.

Remote Lattice Surgery



(c) Input Bell pair fidelity $F = 0.96$



(d) Input Bell pair fidelity $F = 0.86$.

Figure 6.1: Simulation results for Experiment 1 with $F = 0.96$ and $F = 0.86$. State injection and entanglement ejection are plotted in the same figure as they exhibit similar trends. Here, solid lines represent ejection and dotted lines represent injection. The x-axis represents the physical noise strength p of the SI1000 model, and the y-axis represents the logical error rate. Error bars indicate the standard deviation of the data. Missing data points for the state injection results are due to low post-selection probability.

Note that the number of consumed Bell pairs significantly varies across protocols, and here I do not aim to compare which is the best. The goal of this experiment is to characterize features of each individual protocol, highlighting the difference between them.

6.2.1 Error Suppression

A distinct trend in error suppression emerges when comparing injection/ejection-based schemes and remote lattice surgery. For injection and ejection, logical error rates are suppressed as noise strength is reduced; however, they eventually exhibit a floor, where increasing the code distance fails to provide further suppression. In contrast, remote lattice surgery does not suffer from this saturation behavior within the parameters of our simulation.

This discrepancy is primarily due to the difference in resource consumption between each method. While injection and ejection focus on converting a single[1] Bell pair into a logical Bell pair, remote lattice surgery consumes $d(2d-1)$ for the same task. This means that remote lattice surgery naturally consumes more qubits as code distances increase, while other methods are forced to perform operations that exhaust a single, limited resource even for large code distances.

Moreover, we observe a "floor" for error suppression in injection/ejection schemes. This is likely due to such protocols relying on unprotected raw qubit operations unlike remote lattice surgery. Noise introduced during such operations serve as single points of failure that cannot be suppressed by the error-correcting code. Due to this feature, the logical error rates hit a "floor," where the error rate decreases only proportionally to the noise strength.

An interesting fact about entanglement ejection is that it can be adapted to consume at most $2d-1$ Bell pairs in a single shot. For such instances, we can expect that the trends in error suppression would align more closely with the behavior of remote lattice surgery.

6.2.2 Pseudo-Thresholds

Each scheme has a different trend in their behavior of pseudo-thresholds. Entanglement ejection exhibits a clear pseudo-threshold region at a noise strength around $p \simeq 4.2\times10^{-3}$, for both high and low fidelity situations. For state injection, a pseudo-threshold behavior is not observed from simulation results because the post-selection is almost constantly failing in high-noise regimes. The pseudo-threshold for remote lattice surgery is found to be around $p \simeq 1.8 \times 10^{-3}$ for $F = 0.96$ and $p \simeq 1.4 \times 10^{-3}$ for $F = 0.86$. From this, we can tell that remote lattice surgery exhibits a lower pseudo-threshold than ejection,

[1]Note that injection fails with probability p_{inject}, requiring $1/p_{\text{inject}}$ Bell pairs on average.

which means that it has much more stringent requirements on the physical qubits to be less noisy.

This phenomenon is possibly due to remote lattice surgery having more locations that can potentially introduce errors in a single attempt. Remote lattice surgery includes $d(2d-1)$ remote CNOT gates, which are much noisier than local CNOT gates due to the physical Bell pairs and the measurement errors. On the other hand, entanglement ejection has fewer noise sources, where entanglement swapping is the only location where we actually interface with the physical Bell pairs. Of course, even within ejection, there are many noise sources, such as the code shrinking process or the entanglement swapping process, as they involve direct measurements of physical qubits. However, the total noisy locations are fewer than that of remote lattice surgery. Therefore, remote lattice surgery naturally presents more opportunities for errors to propagate through the circuit, and therefore leads to a lower pseudo-threshold region.

6.3 Results: Experiment Two

Fig. 6.2 shows the comparison of the achievable logical error rates for the three different protocols: state injection, remote lattice surgery, and entanglement ejection. The results are plotted as a function of the noise strength p under the SI1000 model, with each subfigure representing a different input physical Bell pair fidelity. In this section, I present the results for the specific cases of $F = 0.96$ and $F = 0.86$, and the complete simulation results for input fidelities in the range $F \in [0.86, 0.96]$ are provided in Appendix B.

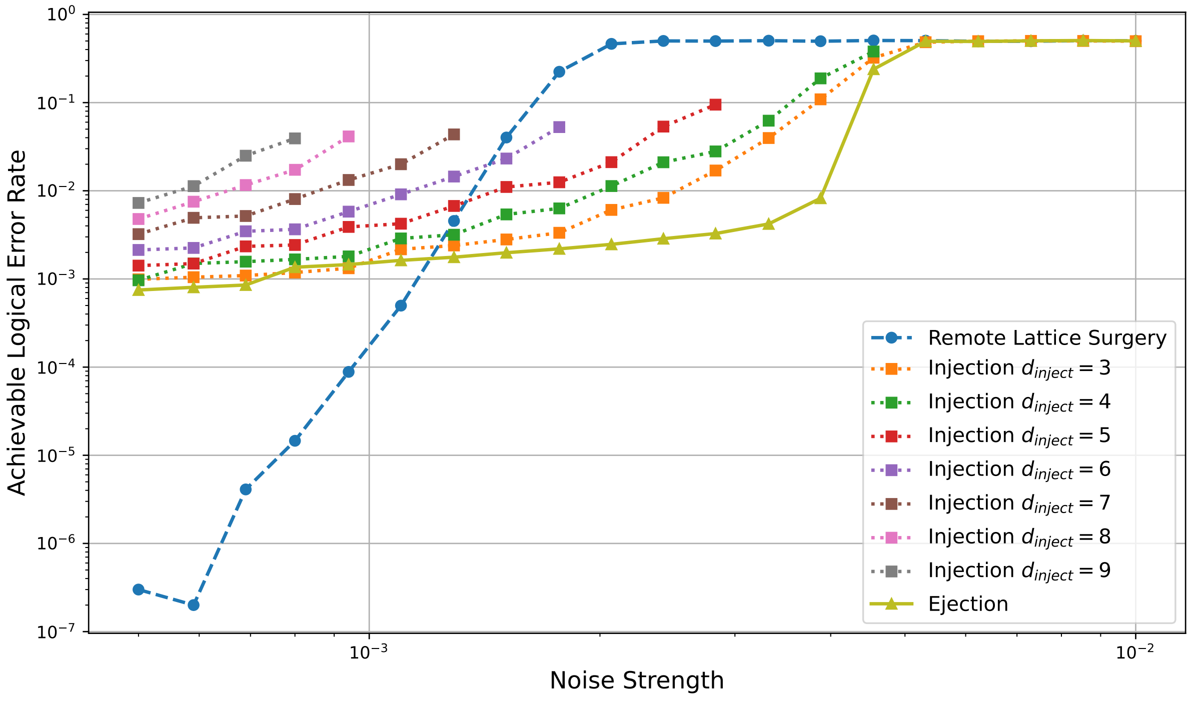
6.3.1 State Injection vs. Entanglement Ejection

In terms of achievable logical error rates, ejection consistently outperforms injection across nearly all parameter regimes. This is somewhat a counterintuitive result, as raw data from the first experiment actually shows injection achieving a slightly lower logical error rate than ejection. However, when resource constraints are applied and post-protocol distillation is performed, ejection yields slightly lower logical error rates.

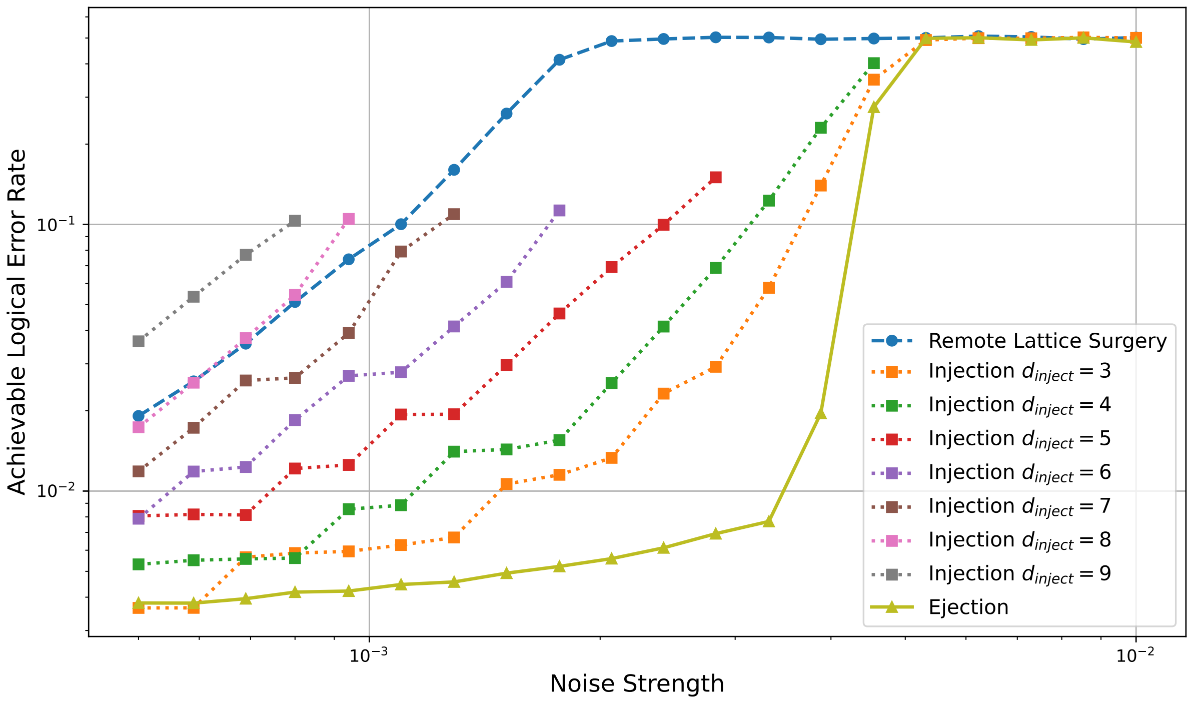
(1) Disadvantages of Probabilistic Protocols

One reason for this is due to the difference between *deterministic* and *probabilistic* protocols. Because ejection is deterministic, it produces more (albeit noisy) logical Bell pairs from the same resource budget compared to the probabilistic injection protocol. This provides more logical entanglement to be distilled, allowing further suppression of the logical error rate.

This behavior of probabilistic protocols performing poorly can be further confirmed by comparing the state injection of different injecting distances. As a larger injecting distance implies better logical error rates, one might consider protocols with large d_{inject}



(a) Input Bell pair fidelity $F = 0.96$.



(b) Input Bell pair fidelity $F = 0.86$.

Figure 6.2: Simulation results for Experiment 2 with $F = 0.96$ and $F = 0.86$. The y-axis represents the achievable logical error rate, and the x-axis is the noise strength parameter p of the SI1000 model. Solid lines represent entanglement ejection, dotted lines represent state injection, and dash-dotted lines represent remote lattice surgery. Missing data points for state injection are due to low post-selection probability.

should have better error suppression. This is indeed true for an unlimited supply of entanglement, but this higher logical error rate comes at the cost of much lower success probability per attempt. Therefore, when the resources are constrained (and also when we perform distillation afterwards), a protocol closer to deterministic seems to show better performance.

(2) Structural Advantages of Ejection

However, this still does not completely answer the question of why entanglement ejection is better than state injection in terms of the achievable logical error rate. After all, the consumed resources stay the same, so this implies that entanglement ejection should have a superior structure compared to state injection.

One major difference between these two methods is that state injection is a bottom-up approach that grows a noisy physical state into a logical state, whereas ejection is a top-down approach that begins with an already encoded state and breaks it down to the physical level. I hypothesize that this top-down approach provides robustness; even though heavy noise is applied to the physical qubits during ejection, this structure seems to be favorable for fault tolerance rather than forcefully converting a physical entanglement to a logical one.

6.3.2 Remote Lattice Surgery vs. Entanglement Ejection

The relative performance between entanglement ejection and remote lattice surgery depends on both the circuit-noise strength and the fidelity of the input Bell pair. For high-fidelity input Bell pairs, remote lattice surgery does outperform entanglement ejection in the low-circuit-noise regime (in the order of 10^{-4}), but this trend reverses as circuit noise increases. In contrast, when considering low-fidelity input Bell pairs, remote lattice surgery is consistently outperformed by nearly all other logical entanglement generation protocols.

(1) High-Fidelity Input Bell Pairs

The disadvantage of ejection in the low noise regime can be attributed to the logical error rate saturation behavior. Due to this, the logical error rate of the ejected-and-distilled state remains higher. This creates a large gap between achievable logical error rates for remote lattice surgery and ejection.

Conversely, the advantage of ejection in higher noise regimes is because remote lattice surgery has a much lower pseudo-threshold. In other words, remote lattice surgery has higher sensitivity against circuit noise. Due to this phenomenon, logical error rates of remote lattice surgery do not decrease for moderate noise regions, and as a result, is outperformed by ejection.

(2) Low-Fidelity Input Bell Pairs

For low-fidelity input Bell pairs, remote lattice surgery loses its dominance against other methods. In fact, for $F = 0.86$, remote lattice surgery is the least effective method among almost all. This can mainly be attributed to its sensitivity against the fidelity of physical Bell pairs, which is not so obvious for ejection or injection.

For example, at a fixed circuit noise of $p = 1.0 \times 10^{-3}$, decreasing the input fidelity from $F = 0.96$ to $F = 0.86$ causes the logical error rate of remote lattice surgery to increase by a factor of 400 (from 2.0×10^{-4} to 8.0×10^{-2}). In contrast, ejection only sees an increase by a factor of 2.7 (from 1.5×10^{-3} to 4.0×10^{-3}) under the same conditions.

This behavior can be attributed to the same reason remote lattice surgery has a low pseudo-threshold: it inherently possesses more noise sources that can compromise the logical state. This also means that if each of these noise sources themselves become stronger, the protocol is more severely impacted. Due to this behavior, remote lattice surgery is ineffective for low-fidelity input Bell pairs.

6.4 Recap and Future Directions

Overall, my analysis shows that entanglement ejection is the predominant method among the three compared protocols: ejection, state injection, and remote lattice surgery. In particular, ejection performs well in regimes characterized by either low-fidelity input Bell pairs, or high-fidelity inputs with moderate circuit noise. The only parameter regime where ejection is outperformed is when input fidelity is high and circuit noise is low; in this case, remote lattice surgery yields a lower achievable logical error rate.

These findings imply that for the near-term and the foreseeable future, where quantum networks and quantum computers remain noisy and operate near the break-even threshold, entanglement ejection offers a resource-efficient and error-resilient solution for logical entanglement generation.

While these strong results provide a promising outlook for entanglement ejection, they also open opportunities for future work. In the following subsections, I outline potential paths to build upon this research.

6.4.1 Extension of the Protocol

One promising direction of extending entanglement ejection is to modify it to be adaptive against the quantity of the input Bell pairs. As emphasized several times, ejection can be adapted to generate entanglement between surface codes and a repetition code of k qubits, where $1 \leq k \leq 2d - 1$. This allows for the simultaneous consumption of k Bell pairs during entanglement swapping. I hypothesize that such cases may have similar traits to remote lattice surgery, especially for the case of $k = 2d - 1$, potentially offering better error suppression with the cost of having lower pseudo-thresholds and vulnerability against low-fidelity input Bell pairs.

Another direction is to build a tailored decoder towards logical Bell pair generation, rather than relying on generic decoders such as PyMatching. Recent research on entanglement boosting [123] has demonstrated that specialized decoders can significantly improve the logical error rate of post-selected logical entanglement. This suggests that applying similar "tweaked" decoders or imposing carefully determined post-selection conditions on the ejection process could further drive down logical error rates.

Moreover, I did not perform a thorough analysis of entanglement distillation. Instead, I simply picked one of the most generic protocols, the BBPSSW protocol [11], and derived the estimated end fidelity via simple calculations. In other words, I took a conservative approach where distillation has the minimum effect on the achievable logical error rates. This implies that there remain potential benefits for methods that can perform distillation on a larger number of logical Bell pairs, which in particular is ejection among the three compared protocols, making this an interesting step forward.

Finally, assessing the generality of ejection across different QECCs is also necessary. While I focused on unrotated surface codes, the protocol can, in principle, be generalized to function on any code where lattice surgery is possible. Therefore, implementing and analyzing the behavior of entanglement ejection on a wider variety of codes, such as rotated surface codes or high-rate quantum Low-Density Parity-Check (LDPC) codes, would be beneficial for the field.

6.4.2 A More Realistic Analysis

A crucial direction for entanglement ejection is a feasibility study considering real-world hardware characteristics. While I have used a realistic noise model (the SI1000 model) to evaluate the protocols, the current analysis is still conducted at a high level of abstraction. For example, I did not consider an actual implementation of ejection under a specific qubit platform, such as superconducting chips, ion traps, or neutral atoms. The operational characteristics of these platforms differ drastically; for instance, neutral atom qubits can be physically moved in batches via optical tweezers, while superconducting qubits are constrained to a static, fixed connectivity. Furthermore, the interfacing efficiency with quantum networks, and therefore the quality of the physical Bell pairs, also differs across various physical media. Therefore, such a feasibility analysis across various qubit platforms would be valuable.

Beyond hardware specifics, future research can build upon this foundation by integrating the dynamic constraints inherent to distributed quantum computing. In a practical setting, it can be expected that physical Bell pairs are provided at some given rate, rather than a specific number of Bell pairs being provided at once. Such rates are expected to be dominated by the structure of the underlying quantum network. Furthermore, devices face strict limitations on the number of physical qubits they can store locally[2]. Future studies can incorporate such network performance behavior and local computational constraints to provide a system-level evaluation and comparison between protocols.

6.4.3 Alternative Applications of Ejection

The utility of entanglement ejection can be extended beyond the context of distributed quantum computing. This is because the ejected state, an entangled state of a physical qubit and a logical qubit, is already an interesting resource on its own right.

For example, ejection can be adapted for magic state injection by performing state teleportation through the physical end of the ejected pair. Furthermore, this structure might be beneficial for quantum sensing; the physical qubit gathers information from the environment[3], while the logical state serves as a robust, error-protected storage that interfaces with fault-tolerant quantum computers.

[2]Recall that this was one of the motivations for distributed quantum computation; space is limited, and therefore we want to distribute.

[3]Within the context of DQC and FTQC, this information was treated as noise.

Chapter 7

Conclusion

Generating logical Bell pairs is a critical task in fault-tolerant distributed quantum computing, as they serve as the fundamental building blocks for communication between distant nodes. While numerous methods exist for generating logical entanglement, no single protocol is universally optimal, and there remains potential for further optimization.

In this thesis, I introduced **entanglement ejection**: a novel protocol for generating logical Bell pairs in surface codes. Entanglement ejection allows the conversion of physical to logical entanglement by first generating a physical—logical entanglement within a single computational node. This is followed by entanglement swapping between the "ejected" physical qubit and one half of a physically distributed Bell pair, resulting in a (slightly noisy) logical pair. We then perform entanglement distillation on the logical pair to reduce the error of such states.

To evaluate the efficacy of this approach, I performed a comparative analysis against two established methods: state injection, which probabilistically converts a physical pair into a logical pair, and remote lattice surgery, which performs lattice surgery by replacing local CNOT gates with remote CNOT gates. As the primary metric for comparing between these methods, I defined the **achievable logical error rate**, which allows us to compare logical error rates while accounting for the disparity in Bell pair consumption across each protocol.

Through numerical simulations assuming superconducting-inspired noise and realistic input Bell pair fidelities, I demonstrated that entanglement ejection achieves superior performance compared to both state injection and remote lattice surgery across a wide parameter regime. In fact, ejection constantly outperforms state injection, and while there exist situations where remote lattice surgery has a lower achievable logical error rate than ejection, it is only in the specific parameter regime of high-fidelity Bell pairs and low circuit noise.

This significant outcome implies that, entanglement ejection is a resource-efficient and robust protocol for generating logical entanglement on near-term quantum devices. This further allows it to serve as a primitive for realizing fault-tolerant distributed quantum computation for the foreseeable future.

There exist several potential directions for future research, including extending the ejection protocol to consume multiple physical Bell pairs simultaneously, optimizing decoding and distillation strategies, and generalizing the method to other error-correcting codes. Moreover, conducting a feasibility analysis on real devices and a system-level analysis that considers dynamic resource constraints remains a crucial milestone for the realization of entanglement ejection. Finally, exploring alternative applications beyond distributed quantum computing, such as quantum sensing, represents an important step forward of ejection.

Acknowledgment

I would like to express my gratitude to the faculty of the Kusumoto-Takashio-Van Meter-Uehara-Mitsugi-Nakazawa-Okoshi joint research group (RG) for their advice and feedback during numerous research presentations.

I would like to thank my primary supervisor, Rodney Van Meter, for his guidance over the past four years. Although I was the kind of student who constantly brought up new, trendy topics that I didn't always fully understand, he always encouraged me to pursue them, providing me with appropriate references. I am also deeply grateful for his guidance on technical writing and presentation skills, as well as for the opportunity to attend numerous international conferences.

My sincere thanks go to Shota Nagayama, Naphan Benchasattabuse, Michal Hajdušek, and Fumiyoshi Kobayashi for their support regarding the research on entanglement ejection. Specifically, I am grateful to Dr. Nagayama for initially proposing the concept of "ejection" and allowing me to work on it over the course of the year. I appreciate the assistance of Dr. Benchasattabuse and Dr. Hajdušek for their continuous guidance in shaping this thesis, and Dr. Kobayashi for his technical advice on quantum circuit simulation.

I would also like to thank other faculty members of AQUA: Hiroyuki Kusumoto, Bernard Ousmane Sane, Amin Taherkhani, Daisuke Sakuma, and Andrew Todd, for their technical feedback and valuable advice both in and outside of regular meetings.

Furthermore, I am grateful to my seniors in AQUA: Naphan Benchasattabuse, Yasuhiro Ohkura, Ryosuke Satoh, Kentaro Teramoto, Makoto Nakai, and many others who welcomed me as a new undergraduate student in AQUA. Without them, I would not have come this far in the group, and they will continue to be my role models.

I am also thankful to fellow AQUA members, specifically Poramet Pathumsoot, Takafumi Oka, Monet Tokuyama Friedrich, Hikaru Yokomori, Marii Koyama, Junseo Kwak, Yuta Imahoko, Ruka Ishikawa, Akash Prasad, Siddartha Bhandari, Sohma Kobayashi, and many others. I am lucky that I had the opportunity to discuss various topics, ranging from quantum computing to daily life.

I would also like to acknowledge fellow RG members, specifically Korry Luke, Fumiaki Mizuno, Takumi Ishihara, Riku Mochizuki, Hiroki Takemura, Shota Suzuki, Yoshiki Uchida, Atsuya Osaki, Kentaro Muta, Shiori Hoshina, Harunari Takata, Taihei Yamaguchi, Shota Imamura, Kota Yatagai, Raku Saito, and many others, with whom I had the opportunity to discuss research, hobbies, and socialize in and outside the Delta building.

I would also like to express my gratitude to Shun Hasegawa and Futa Watanabe for the study group where we studied quantum computing and quantum error correction together. Such an environment contributed to building my foundational skills in the field of quantum information.

On a practical note, I must also mention Nissin Food Products for developing the *Kanzen Meshi* series. Their highly engineered, tasty-yet-balanced products supported my health and sustained me during busy periods of writing.

Moreover, I would like to thank *Ling Tosite Sigure* for their incredible music. Whenever I felt lost or simply fatigued, their music brought me back to life. Although I have not been able to spend enough time playing the guitar recently, I hope to eventually play a full cover of one of their songs.

Last but not least, I would like to thank my family for their continuous support throughout my life.

Bibliography

[1] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of Computing*, pages 212–219, 1996.

[2] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[3] Peter W. Shor. Scheme for reducing decoherence in quantum computer memory. *Physical Review A*, 52(4):2493–2496, 1995.

[4] Daniel Gottesman. *Stabilizer codes and quantum error correction.* California Institute of Technology, 1997.

[5] Rodney Van Meter and Simon Devitt. The path to scalable distributed quantum computing. *Computer*, 49(9):31–42, 2016.

[6] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*, 86(3):032324, 2012.

[7] Shota Nagayama, Byung-Soo Choi, Simon Devitt, Shigeya Suzuki, and Rodney Van Meter. Interoperability in encoded quantum repeater networks. *Physical Review A*, 93(4):042338, 2016.

[8] Shinichi Sunami, Shiro Tamiya, Ryotaro Inoue, Hayata Yamasaki, and Akihisa Goban. Scalable networking of neutral-atom qubits: Nanofiber-based approach for multiprocessor fault-tolerant quantum computers. *PRX Quantum*, 6(1):010101, 2025.

[9] Joshua Ramette, Josiah Sinclair, Nikolas P. Breuckmann, and Vladan Vuletić. Fault-tolerant connection of error-corrected qubits with noisy links. *npj Quantum Information*, 10(1):58, 2024.

[10] Josiah Sinclair, Joshua Ramette, Brandon Grinkemeyer, Dolev Bluvstein, Mikhail D. Lukin, and Vladan Vuletić. Fault-tolerant optical interconnects for neutral-atom arrays. *Physical Review Research*, 7(1):013313, 2025.

[11] Charles H. Bennett, Gilles Brassard, Sandu Popescu, Benjamin Schumacher, John A. Smolin, and William K. Wootters. Purification of noisy entanglement and faithful teleportation via noisy channels. *Physical Review Letters*, 76(5):722, 1996.

[12] 中田芳史. **量子情報理論 情報から物理現象の理解まで**. 朝倉書店, 2024.

[13] Albert Einstein, Boris Podolsky, and Nathan Rosen. Can quantum-mechanical description of physical reality be considered complete? *Physical Review*, 47(10):777–780, 1935.

[14] Martin B. Plenio and Shashank Virmani. An introduction to entanglement measures. *arXiv preprint quant-ph/0504163*, 2005.

[15] Daniel F. V. James, Paul G. Kwiat, William J. Munro, and Andrew G. White. Measurement of qubits. *Physical Review A*, 64(5):052312, 2001.

[16] William K. Wootters and Wojciech H. Zurek. A single quantum cannot be cloned. *Nature*, 299(5886):802–803, 1982.

[17] John Von Neumann. *Mathematical foundations of quantum mechanics: New edition.* Princeton university press, 2018.

[18] Roman S. Ingarden. Quantum information theory. *Reports on Mathematical Physics*, 10(1):43–72, 1976.

[19] Alexander S. Holevo. Bounds for the quantity of information transmitted by a quantum communication channel. *Problemy Peredachi Informatsii*, 9(3):3–11, 1973.

[20] Stephen Wiesner. Conjugate coding. *ACM Sigact News*, 15(1):78–88, 1983.

[21] Rolf Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 5(3):183–191, 1961.

[22] Paul Benioff. The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. *Journal of statistical physics*, 22(5):563–591, 1980.

[23] Yuri I. Manin. *The computable and the non-computable. (Vychislimoe I Nevychislimoe).* Kibernetika. Sovetskoe Radio, 1980.

[24] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6):467–488, 1982.

[25] Benjamin Schumacher. Quantum coding. *Physical Review A*, 51(4):2738–2747, 1995.

[26] Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936.

[27] David Deutsch. Quantum theory, the Church–Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London A*, 400(1818):97–117, 1985.

[28] David Deutsch. Quantum computational networks. *Proceedings of the Royal Society of London A*, 425(1868):73–90, 1989.

[29] Andrew C. Yao. Quantum circuit complexity. In *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*, pages 352–361, 1993.

[30] Charles H. Bennett, Gilles Brassard, Claude Crépeau, Richard Jozsa, Asher Peres, and William K. Wootters. Teleporting an unknown quantum state via dual classical and Einstein–Podolsky–Rosen channels. *Physical Review Letters*, 70(13):1895–1899, 1993.

[31] Christopher M. Dawson and Michael A. Nielsen. The Solovay–Kitaev algorithm. *Quantum Information & Computation*, 6(1):81–95, 2006.

[32] Soichiro Yamazaki and Seiseki Akibue. Clifford+V synthesis for multi-qubit unitary gates. *arXiv preprint arXiv:2510.08312*, 2025.

[33] Ryan Babbush, Jarrod R. McClean, Michael Newman, Craig Gidney, Sergio Boixo, and Hartmut Neven. Focus beyond quadratic speedups for error-corrected quantum advantage. *PRX Quantum*, 2(1):010103, 2021.

[34] Eric W. Weisstein. Number field sieve. From MathWorld–A Wolfram Web Resource. Accessed: 2026-01-05.

[35] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[36] David Beckman, Amalavoyal N. Chari, Srikrishna Devabhaktuni, and John Preskill. Efficient networks for quantum factoring. *Physical Review A*, 54(2):1034–1063, 1996.

[37] Martin Ekerå and Johan Håstad. Quantum algorithms for computing short discrete logarithms and factoring RSA integers. In *8th International Workshop on Post-Quantum Cryptography, PQCrypto 2017*, pages 347–363, 2017.

[38] Oded Regev. An efficient quantum factoring algorithm. *Journal of the ACM*, 72(1):1–13, 2025.

[39] Clémence Chevignard, Pierre-Alain Fouque, and André Schrottenloher. Reducing the number of qubits in quantum factoring. In *Advances in Cryptology – CRYPTO 2025: 45th Annual International Cryptology Conference, Proceedings, Part II*, pages 384–415, 2025.

[40] Stephen P. Jordan, Keith S. M. Lee, and John Preskill. Quantum algorithms for quantum field theories. *Science*, 336(6085):1130–1133, 2012.

[41] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5(1):4213, 2014.

[42] Andrew M. Childs, Dmitri Maslov, Yunseong Nam, Neil J. Ross, and Yuan Su. Toward the first quantum simulation with quantum speedup. *Proceedings of the National Academy of Sciences*, 115(38):9456–9461, 2018.

[43] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15):150502, 2009.

[44] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[45] Stephen P. Jordan, Noah Shutty, Mary Wootters, Adam Zalcman, Alexander Schmidhuber, Robbie King, Sergei V. Isakov, Tanuj Khattar, and Ryan Babbush. Optimization by decoded quantum interferometry. *Nature*, 646(8086):831–836, 2025.

[46] John M. Martyn, Zane M. Rossi, Andrew K. Tan, and Isaac L. Chuang. Grand unification of quantum algorithms. *PRX Quantum*, 2(4):040203, 2021.

[47] B. Jack Copeland. The Church–Turing thesis. In *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Winter 2024 edition, 2024.

[48] Hans J. Briegel, Wolfgang Dür, Juan I. Cirac, and Peter Zoller. Quantum repeaters: The role of imperfect local operations in quantum communication. *Physical Review Letters*, 81(26):5932–5935, 1998.

[49] Koji Azuma, Sophia E. Economou, David Elkouss, Paul Hilaire, Liang Jiang, Hoi-Kwong Lo, and Ilan Tzitrin. Quantum repeaters: From quantum networks to the quantum internet. *Reviews of Modern Physics*, 95(4):045006, 2023.

[50] Kento Samuel Soon, Michal Hajdušek, Shota Nagayama, Naphan Benchasattabuse, Kentaro Teramoto, Ryosuke Satoh, and Rodney Van Meter. An implementation and analysis of a practical quantum link architecture utilizing entangled photon sources. In *2024 International Conference on Quantum Communications, Networking, and Computing (QCNC)*, pages 25–32, 2024.

[51] David Deutsch, Artur K. Ekert, Richard Jozsa, Chiara Macchiavello, Sandu Popescu, and Anna Sanpera. Quantum privacy amplification and the security of quantum cryptography over noisy channels. *Physical Review Letters*, 77(13):2818–2821, 1996.

[52] Stefano Pirandola, Ulrik L. Andersen, Leonardo Banchi, Mario Berta, Darius Bunandar, Roger Colbeck, Dirk Englund, Tobias Gehring, Cosmo Lupo, Carlo Ottaviani, Jason L. Pereira, Mohsen Razavi, J. Shamsul Shaari, Marco Tomamichel, Vladyslav C. Usenko, Giuseppe Vallone, Paolo Villoresi, and Petros Wallden. Advances in quantum cryptography. *Advances in Optics and Photonics*, 12(4):1012–1236, 2020.

[53] Charles H. Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. *Theoretical computer science*, 560:7–11, 2014.

[54] Marco Tomamichel and Anthony Leverrier. A largely self-contained and complete security proof for quantum key distribution. *Quantum*, 1:14, 2017.

[55] Artur K. Ekert. Quantum cryptography based on bell's theorem. *Physical Review Letters*, 67(6):661–663, 1991.

[56] Charles H. Bennett, Gilles Brassard, and N. David Mermin. Quantum cryptography without Bell's theorem. *Physical Review Letters*, 68(5):557–559, 1992.

[57] Timothy J. Proctor, Paul A. Knott, and Jacob A. Dunningham. Multiparameter estimation in networked quantum sensors. *Physical Review Letters*, 120(8):080501, 2018.

[58] Daniel Gottesman, Thomas Jennewein, and Sarah Croke. Longer-baseline telescopes using quantum repeaters. *Physical Review Letters*, 109(7):070503, 2012.

[59] Ebubechukwu O. Ilo-Okeke, Louis Tessler, Jonathan P. Dowling, and Tim Byrnes. Remote quantum clock synchronization without synchronized clocks. *npj Quantum Information*, 4(1):40, 2018.

[60] Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. Universal blind quantum computation. In *Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 517–526, 2009.

[61] Tomoyuki Morimae and Keisuke Fujii. Blind quantum computation protocol in which Alice only makes measurements. *Physical Review A*, 87(5):050301, 2013.

[62] Michael Ben-Or and Avinatan Hassidim. Fast quantum byzantine agreement. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, page 481–485, 2005.

[63] Marcello Caleffi, Michele Amoretti, Davide Ferrari, Jessica Illiano, Antonio Manzalini, and Angela Sara Cacciapuoti. Distributed quantum computing: A survey. *Computer Networks*, 254:110672, 2024.

[64] David Barral, F. Javier Cardama, Guillermo Díaz-Camacho, Daniel Faílde, Iago F. Llovo, Mariamo Mussa-Juane, Jorge Vázquez-Pérez, Juan Villasuso, César Piñeiro, Natalia Costas, Juan C. Pichel, Tomás F. Pena, and Andrés Gómez. Review of distributed quantum computing: from single QPU to high performance quantum computing. *Computer Science Review*, 57:100747, 2025.

[65] John Preskill. Quantum computing in the NISQ era and beyond. *Quantum*, 2:79, 2018.

[66] Youngseok Kim, Andrew Eddins, Sajant Anand, Ken Xuan Wei, Ewout Van Den Berg, Sami Rosenblatt, Hasan Nayfeh, Yantao Wu, Michael Zaletel, Kristan Temme, and Abhinav Kandala. Evidence for the utility of quantum computing before fault tolerance. *Nature*, 618(7965):500–505, 2023.

[67] Joseph Tindall, Matthew Fishman, E. Miles Stoudenmire, and Dries Sels. Efficient tensor network simulation of IBM's Eagle kicked Ising experiment. *PRX Quantum*, 5(1):010308, 2024.

[68] Craig Gidney. How to factor 2048 bit RSA integers with less than a million noisy qubits. *arXiv preprint arXiv:2505.15917*, 2025.

[69] Daniel Gottesman. *Surviving as a quantum computer in a classical world.* 2024.

[70] A. Robert Calderbank and Peter W. Shor. Good quantum error-correcting codes exist. *Physical Review A*, 54(2):1098–1105, 1996.

[71] Andrew M. Steane. Multiple-particle interference and quantum error correction. *Proceedings of the Royal Society of London A*, 452(1954):2551–2577, 1996.

[72] A. Robert Calderbank, Eric M. Rains, Peter W. Shor, and Neil J. A. Sloane. Quantum error correction via codes over GF(4). *IEEE Transactions on Information Theory*, 44(4):1369–1387, 1998.

[73] Alexei Kitaev. Fault-tolerant quantum computation by anyons. *Annals of physics*, 303(1):2–30, 2003.

[74] Sergey Bravyi and Alexei Kitaev. Quantum codes on a lattice with boundary. *arXiv preprint quant-ph/9811052*, 1998.

[75] Jean-Pierre Tillich and Gilles Zémor. Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength. *IEEE Transactions on Information Theory*, 60(2):1193–1202, 2013.

[76] Elwyn Berlekamp, Robert McEliece, and Henk Van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.

[77] Min-Hsiu Hsieh and François Le Gall. NP-hardness of decoding quantum error-correction codes. *Physical Review A*, 83(5):052331, 2011.

[78] Pavithran Iyer and David Poulin. Hardness of decoding quantum stabilizer codes. *IEEE Transactions on Information Theory*, 61(9):5209–5223, 2015.

[79] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[80] Oscar Higgott. Pymatching: A python package for decoding quantum codes with minimum-weight perfect matching. *ACM Transactions on Quantum Computing*, 3(3), June 2022.

[81] Austin G. Fowler. Proof of finite surface code threshold for matching. *Physical Review Letters*, 109(18):180502, 2012.

[82] Peter W. Shor. Fault-tolerant quantum computation. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 56–65, 1996.

[83] Andrew M. Steane. Active stabilization, quantum computation, and quantum state synthesis. *Physical Review Letters*, 78(11):2252–2255, 1997.

[84] Emanuel Knill. Quantum computing with realistically noisy devices. *Nature*, 434(7029):39–44, 2005.

[85] Robert Raussendorf, Jim Harrington, and Kovid Goyal. Topological fault-tolerance in cluster state quantum computation. *New Journal of Physics*, 9(6):199, 2007.

[86] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys*, 51(4):1–35, 2018.

[87] Bryan Eastin and Emanuel Knill. Restrictions on transversal encoded quantum gate sets. *Physical Review Letters*, 102(11):110502, 2009.

[88] Dominic Horsman, Austin G. Fowler, Simon Devitt, and Rodney Van Meter. Surface code quantum computing by lattice surgery. *New Journal of Physics*, 14(12):123011, 2012.

[89] Daniel Gottesman. The Heisenberg representation of quantum computers. *arXiv preprint quant-ph/9807006*, 1998.

[90] Nikolas P. Breuckmann and Simon Burton. Fold-transversal clifford gates for quantum codes. *Quantum*, 8:1372, 2024.

[91] Zi-Han Chen, Ming-Cheng Chen, Chao-Yang Lu, and Jian-Wei Pan. Transversal logical clifford gates on rotated surface codes with reconfigurable neutral atom arrays. *arXiv preprint arXiv:2412.01391*, 2024.

[92] Daniel Gottesman and Isaac L. Chuang. Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations. *Nature*, 402(6760):390–393, 1999.

[93] Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5):052328, 2004.

[94] Andrew J. Landahl and Ciaran Ryan-Anderson. Quantum computing by color-code lattice surgery. *arXiv preprint arXiv:1407.5103*, 2014.

[95] Ying Li. A magic state's fidelity can be superior to the operations that created it. *New Journal of Physics*, 17(2):023037, 2015.

[96] Justyna Lodyga, Paweł Mazurek, Andrzej Grudka, and Michał Horodecki. Simple scheme for encoding and decoding a qubit in unknown state for various topological codes. *Scientific reports*, 5(1):8975, 2015.

[97] Craig Gidney. Cleaner magic states with hook injection. *arXiv preprint arXiv:2302.12292*, 2023.

[98] Sergey Bravyi and Alexei Kitaev. Universal quantum computation with ideal clifford gates and noisy ancillas. *Physical Review A*, 71(2):022316, 2005.

[99] Sergey Bravyi and Jeongwan Haah. Magic-state distillation with low overhead. *Physical Review A*, 86(5):052329, 2012.

[100] Craig Gidney and Austin G. Fowler. Efficient magic state factories with a catalyzed $|\mathrm{CCZ}\rangle$ to $2|\mathrm{T}\rangle$ transformation. *Quantum*, 3:135, 2019.

[101] Daniel Litinski. Magic state distillation: Not as costly as you think. *Quantum*, 3:205, 2019.

[102] Craig Gidney, Noah Shutty, and Cody Jones. Magic state cultivation: growing t states as cheap as cnot gates. *arXiv preprint arXiv:2409.17595*, 2024.

[103] Hayato Goto. Minimizing resource overheads for fault-tolerant preparation of encoded states of the steane code. *Scientific reports*, 6(1):19578, 2016.

[104] Christopher Chamberland and Kyungjoo Noh. Very low overhead fault-tolerant magic state preparation using redundant ancilla encoding and flag qubits. *npj Quantum Information*, 6(1):91, 2020.

[105] Tomohiro Itogawa, Yugo Takada, Yutaka Hirano, and Keisuke Fujii. Efficient magic state distillation by zero-level distillation. *PRX Quantum*, 6(2):020356, 2025.

[106] Sergey Bravyi, Graeme Smith, and John A. Smolin. Trading classical and quantum computational resources. *Physical Review X*, 6(2):021043, 2016.

[107] Austin G. Fowler and Craig Gidney. Low overhead quantum computation using lattice surgery. *arXiv preprint arXiv:1808.06709*, 2018.

[108] Daniel Litinski. A game of surface codes: Large-scale quantum computing with lattice surgery. *Quantum*, 3:128, 2019.

[109] Dorit Aharonov and Michael Ben-Or. Fault-tolerant quantum computation with constant error. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 176–188, 1997.

[110] Emanuel Knill, Raymond Laflamme, and Wojciech H. Zurek. Resilient quantum computation: error models and thresholds. *Proceedings of the Royal Society of London A*, 454(1969):365–384, 1998.

[111] Alexei Kitaev. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys*, 52(6):1191, 1997.

[112] Panos Aliferis, Daniel Gottesman, and John Preskill. Quantum accuracy threshold for concatenated distance-3 codes. *arXiv preprint quant-ph/0504218*, 2005.

[113] Craig Gidney. Stim: a fast stabilizer circuit simulator. *Quantum*, 5:497, 2021.

[114] Jens Eisert, Kurt Jacobs, Polykarpos Papadopoulos, and Martin B. Plenio. Optimal local implementation of nonlocal quantum gates. *Physical Review A*, 62(5):052317, 2000.

[115] Pablo Andrés-Martínez and Chris Heunen. Automated distribution of quantum circuits via hypergraph partitioning. *Physical Review A*, 100(3):032308, 2019.

[116] Eneet Kaur, Shahrooz Pouryousef, Hassan Shapourian, Jiapeng Zhao, Michael Kilzer, Ramana Kompella, and Reza Nejabati. Optimized quantum circuit partitioning across multiple quantum processors. *IEEE Transactions on Quantum Engineering*, 6:1–17, 2025.

[117] Daisuke Sakuma, Tomoki Tsuno, Hikaru Shimizu, Yuki Kurosawa, Monet Tokuyama Friedrich, Kentaro Teramoto, Amin Taherkhani, Andrew Todd, Yosuke Ueno, Michal Hajdušek, Rikizo Ikuta, Rodney Van Meter, Toshihiko Sasaki, and Shota Nagayama. Q-fly: An optical interconnect for modular quantum computers. *arXiv preprint arXiv:2412.09299*, 2024.

[118] Hassan Shapourian, Eneet Kaur, Troy Sewell, Jiapeng Zhao, Michael Kilzer, Ramana Kompella, and Reza Nejabati. Quantum data center infrastructures: A scalable architectural design perspective. *arXiv preprint arXiv:2501.05598*, 2025.

[119] James Ang, Gabriella Carini, Yanzhu Chen, Isaac L. Chuang, Michael Demarco, Sophia E. Economou, Alec Eickbusch, Andrei Faraon, Kai-Mei Fu, Steven Girvin, Michael Hatridge, Andrew Houck, Paul Hilaire, Kevin Krsulich, Ang Li, Chenxu Liu, Yuan Liu, Margaret Martonosi, David McKay, Jim Misewich, Mark Ritter, Robert Schoelkopf, Samuel Stein, Sara Sussman, Hong Tang, Wei Tang, Teague Tomesh, Norm Tubman, Chen Wang, Nathan Wiebe, Yongxin Yao, Dillon Yost, and Yiyu Zhou. ARQUIN: Architectures for multinode superconducting quantum computers. *ACM Transactions on Quantum Computing*, 5(3), September 2024.

[120] Liang Jiang, Jacob M. Taylor, Kae Nemoto, William J. Munro, Rodney Van Meter, and Mikhail D. Lukin. Quantum repeater with encoding. *Physical Review A*, 79(3):032325, 2009.

[121] Sreraman Muralidharan, Linshu Li, Jungsang Kim, Norbert Lütkenhaus, Mikhail D. Lukin, and Liang Jiang. Optimal architectures for long distance quantum communication. *Scientific reports*, 6(1):20463, 2016.

[122] Austin G. Fowler, David S. Wang, Charles D. Hill, Thaddeus D. Ladd, Rodney Van Meter, and Lloyd C. L. Hollenberg. Surface code quantum communication. *Physical Review Letters*, 104(18):180503, 2010.

[123] Shinichi Sunami, Yutaka Hirano, Toshihide Hinokuma, and Hayata Yamasaki. Entanglement boosting: Low-volume logical bell pair preparation for distributed fault-tolerant quantum computation. *arXiv preprint arXiv:2511.10729*, 2025.

[124] Craig Gidney, Michael Newman, Peter Brooks, and Cody Jones. Yoked surface codes. *Nature Communications*, 16(1):4498, 2025.

[125] Ryutaroh Matsumoto. Conversion of a general quantum stabilizer code to an entanglement distillation protocol. *Journal of Physics A: Mathematical and General*, 36(29):8113, 2003.

[126] Simon Anders and Hans J. Briegel. Fast simulation of stabilizer circuits using a graph-state representation. *Physical Review A*, 73(2):022334, 2006.

[127] Ryosuke Satoh, Michal Hajdušek, Naphan Benchasattabuse, Shota Nagayama, Kentaro Teramoto, Takaaki Matsuo, Sara Ayman Metwalli, Poramet Pathumsoot, Takahiko Satoh, Shigeya Suzuki, and Rodney Van Meter. QuISP: a quantum internet simulation package. In *2022 IEEE international conference on quantum computing and engineering (QCE)*, pages 353–364, 2022.

[128] Peter-Jan H. S. Derks, Alex Townsend-Teague, Ansgar G. Burchards, and Jens Eisert. Designing fault-tolerant circuits using detector error models. *Quantum*, 9:1905, 2025.

[129] Anthony Ryan O'Rourke and Simon Devitt. Compare the pair: Rotated versus unrotated surface codes at equal logical error rates. *Physical Review Research*, 7(3):033074, 2025.

[130] Craig Gidney, Michael Newman, and Matt McEwen. Benchmarking the planar honeycomb code. *Quantum*, 6:813, 2022.

[131] Can M. Knaut, Aziza Suleymanzade, Yan-Cheng Wei, Daniel R. Assumpcao, Pieter-Jan Stas, Yan Qi Huan, Bartholomeus Machielse, Erik N. Knall, Madison Sutula, Gefen Baranes, Neil Sinclair, Chawina De-Eknamkul, David S. Levonian, Mihir K. Bhaskar, Hongkun Park, Marko Lončar, and Mikhail D. Lukin. Entanglement of nanophotonic quantum memory nodes in a telecom network. *Nature*, 629(8012):573–578, 2024.

[132] Rodney Van Meter, William J. Munro, Kae Nemoto, and Kohei M. Itoh. Arithmetic on a distributed-memory quantum multicomputer. *ACM Journal on Emerging Technologies in Computing Systems*, 3(4), 2008.

[133] Sivaprasad Omanakuttan, Zichang He, Zhiwei Zhang, Tianyi Hao, Arman Babakhani, Sami Boulebnane, Shouvanik Chakrabarti, Dylan Herman, Joseph Sullivan, Michael A. Perlin, Ruslan Shaydulin, and Marco Pistoia. Threshold for fault-tolerant quantum advantage with the quantum approximate optimization algorithm. *arXiv preprint arXiv:2504.01897*, 2025.

Appendix A

Introduction to Group Theory

Quantum error correction has a deep connection with the mathematics of **group theory**. Here, I aim to present the bare minimum essence of group theory that is required for understanding stabilizer theory. Let us begin with a definition of a **group**.

Definition A.1

A group is a set G of elements combined with a binary operation $\cdot$ that satisfies the following properties.

1. ***Closure**: For any $M, N \in G$:*

$$M \cdot N \in G. \tag{A.1}$$

2. ***Associativity**: For any $M, N, O \in G$:*

$$(M \cdot N) \cdot O = M \cdot (N \cdot O). \tag{A.2}$$

3. ***Identity**: For any $M \in G$, there exists an identity $I \in G$ such that:*

$$I \cdot M = M \cdot I = M. \tag{A.3}$$

4. ***Inverse**: For any $M \in G$, there exists a unique inverse $M^{-1} \in G$ such that:*

$$M \cdot M^{-1} = M^{-1} \cdot M = I. \tag{A.4}$$

An obvious example of a group is a set of integers $\mathbb{Z} = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$ with addition $(+)$ as the binary operation. We can easily confirm that $(\mathbb{Z}, +)$ does indeed form a group as follows.

1. For any $a, b \in \mathbb{Z}$, $a + b \in \mathbb{Z}$.

2. For any $a, b, c \in \mathbb{Z}$, $(a + b) + c = a + (b + c)$.

3. For any $a \in \mathbb{Z}$, $0 \in \mathbb{Z}$ serves as an identity as $a + 0 = 0 + a = a$.

4. For any $a \in \mathbb{Z}$, there exists an inverse $-a \in \mathbb{Z}$ as $a + (-a) = (-a) + a = 0$.

On the other hand, the set of integers $\mathbb{Z}$ with multiplication $(\cdot)$, does not form a group. This is because for any $a \in \mathbb{Z}$, $1 \in \mathbb{Z}$ satisfies the identity property $a \cdot 1 = 1 \cdot a = a$, but for any $a \in \mathbb{Z}$ with $|a| > 1$, there does not exist an inverse $a^{-1} \in \mathbb{Z}$ such that $a \cdot a^{-1} = a^{-1} \cdot a = 1$.

There exist several types of groups that are referred to by specific names.

Definition A.2

*We call H a **subgroup** of a group G if a subset of elements $H \subseteq G$ forms a group with respect to the binary operation of G. This relation is denoted as $H \leq G$.*

For example, although the set $\mathbb{Z}_{|n| \leq 3} = \{-3, -2, -1, 0, 1, 2, 3\}$ is a subset of $\mathbb{Z}$, it does not form a subgroup of $(\mathbb{Z}, +)$ because it is not closed under addition. One counterexample is $3 + 3 = 6 \notin \mathbb{Z}_{|n| \leq 3}$.

Definition A.3

*A group is said to be a commutative group, or an **Abelian group,** if all the elements commute. That is, for any $M, N \in G$:*

$$M \cdot N = N \cdot M. \tag{A.5}$$

The group $(\mathbb{R}, +)$ is an Abelian group, as for any $a, b \in \mathbb{R}$, $a + b = b + a$. For Abelian groups, the sign $+$ instead of $\cdot$ can also be used for representing generic binary operations between elements.

For understanding stabilizer codes, the idea of **generators** is beneficial.

Definition A.4

*A set $\mathcal{G}$ is said to **generate** a group G if every element of G can be written as a product of elements from $\mathcal{G}$. Such a set $\mathcal{G}$ is called a **generator set**, and its members are referred to as **generators**. Typically, a group G generated by the set $\mathcal{G} = \{g_1, g_2, \ldots, g_r\}$ is denoted as:*

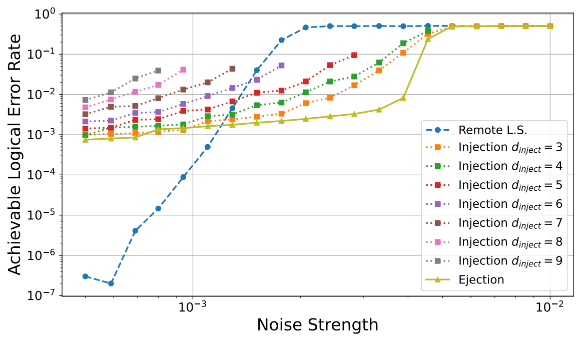
$$G = \langle g_1, g_2, \ldots, g_r \rangle. \tag{A.6}$$

For example, the group of integers under addition, $(\mathbb{Z}, +)$, can be generated by the set $\{1, -1\}$, denoted as $(\mathbb{Z}, +) = \langle 1, -1 \rangle$. This is because every integer can be reached through the repeated addition of 1 or -1.

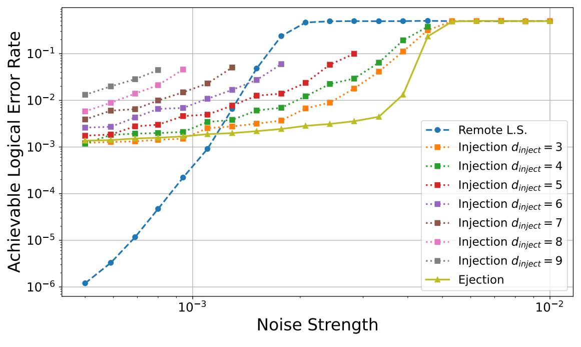
Appendix B

Supplementary Data

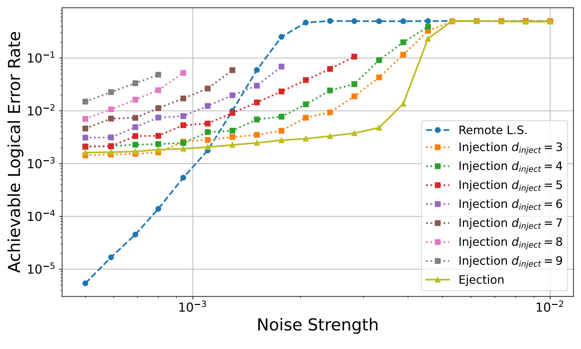
This appendix provides the full comparative results for experiment two, where the achievable logical error rates for each protocol are evaluated for input Bell pair fidelities ranging from $F = 0.86$ to $F = 0.96$.



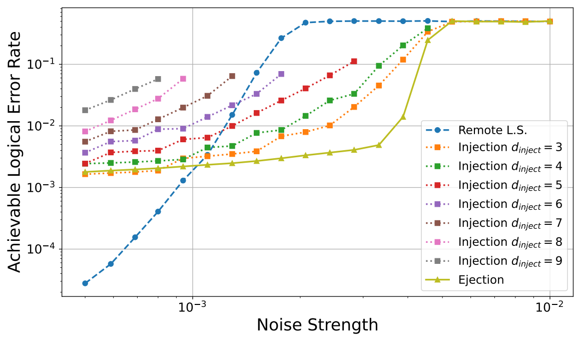
(a) $F = 0.96$



(b) $F = 0.95$

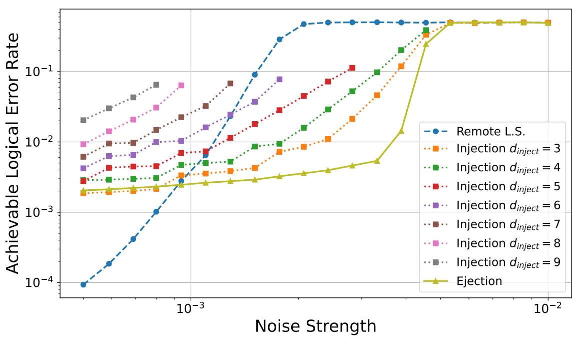


(c) $F = 0.94$

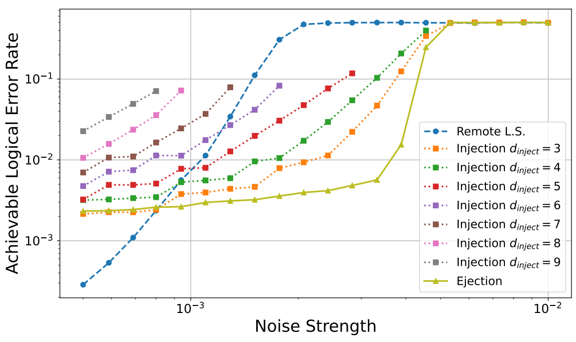


(d) $F = 0.93$

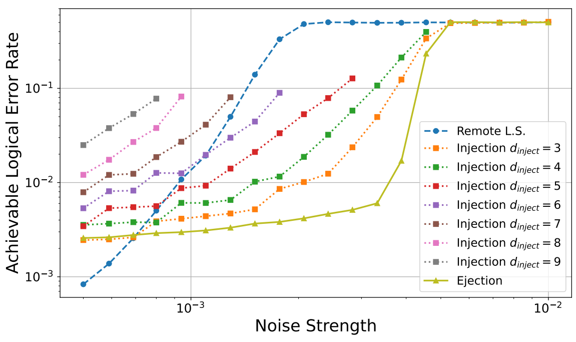
Figure B.1: Simulation results of experiment two ($F = 0.96$ to $F = 0.86$).



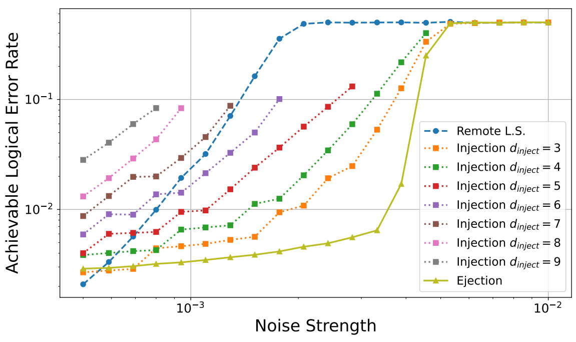
(e) $F = 0.92$



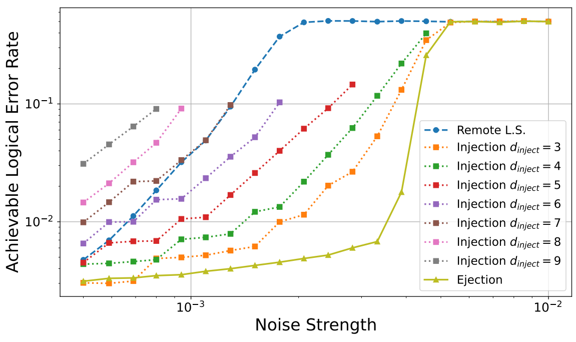
(f) $F = 0.91$



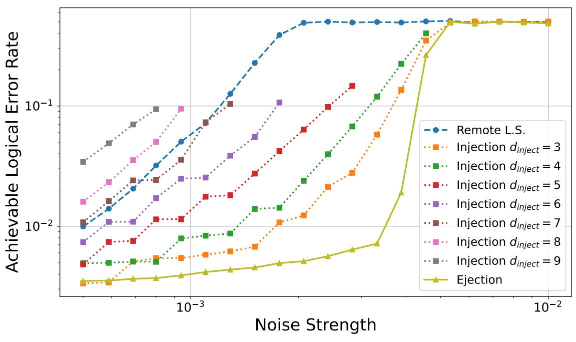
(g) $F = 0.90$



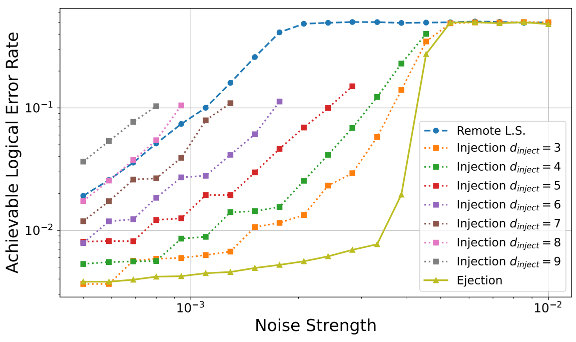
(h) $F = 0.89$



(i) $F = 0.88$



(j) $F = 0.87$



(k) $F = 0.86$

Figure B.1: Simulation results of experiment two ($F = 0.92$ to $F = 0.86$).

Appendix C

List of Publications

Conference Papers (Peer-Reviewed)

- Kento Samuel Soon*, Michal Hajdušek, Shota Nagayama, Naphan Benchasattabuse, Kentaro Teramoto, Ryosuke Satoh, and Rodney Van Meter, "An Implementation and Analysis of a Practical Quantum Link Architecture Utilizing Entangled Photon Sources," in *2024 International Conference on Quantum Communications, Networking, and Computing (QCNC)*, pages 25–32, Jul. 2024.

- Kento Samuel Soon*, Naphan Benchasattabuse, Michal Hajdušek, Kentaro Teramoto, Shota Nagayama, and Rodney Van Meter, "Performance of Quantum Networks Using Heterogeneous Link Architectures," in *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 1914–1923, Sep. 2024.

- Joaquin Chung, Michal Hajdušek, Naphan Benchasattabuse, Alexander Kolar, Ansh Singal, Kento Samuel Soon, Kentaro Teramoto, Allen Zang, Raj Kettimuthu, and Rodney Van Meter, "Cross-Validating Quantum Network Simulators," in *IEEE INFOCOM 2025 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6, May 2025.

Posters with Proceedings (Peer-Reviewed)

- Kento Samuel Soon*, Fumiyoshi Kobayashi, Naphan Benchasattabuse, and Shota Nagayama, "Surface Code Entanglement Ejection," in *2025 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 648–649, Aug. 2025.

*Indicates the paper was presented by Kento Samuel Soon.