Quantum Coding with Entanglement

by

Mark McMahon Wilde

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements of the Degree
DOCTOR OF PHILOSOPHY
(Electrical Engineering)

August 2008

Dedication

To my parents

Gregory and Sharon Wilde

And to my grandparents

Joseph and Rose McMahon

Norbert Jay and Mary Wilde

Acknowledgements

The completion of this Ph.D. has been a long and difficult road with many wonderful memories along the way. I can truly say that there is no way I could have completed it without the help and encouragement from many people.

I first thank my advisor Todd Brun. His help and deep knowledge have been indispensable throughout and I am grateful that he took me on as his student. Our many research meetings have been valuable and I am thankful that Todd gave his time for me. I cannot imagine what a better advisor would be like. I can recall visiting USC in March 2004 for a Ph.D. recruitment trip. I saw a posting on the wall that said there would be a seminar on quantum computing and that Todd Brun was the host. It was from this point that I became fascinated by quantum information, had a desire to contribute original research to the field, and thought it would be great if Todd Brun were my advisor. I look back now and am pleased that all of these dreams have come to fruition.

I next thank Igor Devetak. He was the first to teach me quantum information theory and I am grateful for the many times that he met with me. I met with Igor about one hour per week in the academic semesters of 2005-2006 and these formative meetings helped shape and refine my knowledge of quantum information theory. I am also grateful that Igor invited me to join Todd and Igor's research group meetings.

I thank Jonathan Dowling for accepting me into his research group at LSU during the summers of 2005-2006. I learned so much about quantum optics by conversing with him and the other members of his lab. The research environment that Jon creates by having so many seminars and bringing in visitors is a great one to be involved in.

I am grateful to Daniel Lidar for his collaboration and for teaching his course on quantum error correction at USC. After this class, I can say that my research really took off because I was able to solidify my knowledge of quantum error correction.

I am in debt to Bart Kosko for working with me during the initial period of my Ph.D. He gave valuable advice for how to be a good teacher and researcher and taught me how to have the

discipline required of a good researcher. When I was a teaching assistant, I simply tried to teach like he does and I think I won the Best Teaching Assistant Award for this reason.

I thank Diane Demetras for being a great grad student "mom" away from home. I owe thanks to Antonio Ortega for initially giving me advice. I also thank the fifth floor staff—Milly Montenegro, Mayumi Thrasher, and Gerrielyn Ramos—for providing a pleasant working environment and for filing all those technical reports that I asked to file. I hope our softball team wins! Shock 'em, Volts!

I thank my collaborators Hari Krovi and Austin Lund for helping me along the way. Austin was invaluable during the first summer at LSU when he answered my questions. Hari was a big help in all of our collaborations and especially when we were first trying to figure out what Todd, Igor, and Min-Hsiu had done with entanglement-assisted quantum codes. I also thank everyone else in the group, Min-Hsiu Hsieh, Zhicheng Luo, Ognyan Oreshkov, Shesha Raghunathan, Bilal Shaw, Martin Varbanov, for being a good group of guys to work with. I learned a lot from our debates and conversations.

I finally thank my family for their support. I especially thank my dad "Jack" for the many long Ph.D. conversations that we have had on the phone and for being excited about my research. I thank all of the wonderful friends I have made while in the City of Angels, especially Ali, Ben, Mary L., and Mary B., and I thank Fr. Lawrence for our spiritual conversations. I lastly give a "shout out" to Santa Monica power yoga studios for providing a great environment for destressing and to Pete Carroll for having an awesome Trojan football team! Fight on!

Table of Contents

Dedication ii

Acknowledgements iii

List of Tables viii

List of Figures ix

Abstract xi

1 Introduction 1
- 1.1 Quantum Error Correction . . . 2
  - 1.1.1 Stabilizer Quantum Coding . . . 3
  - 1.1.2 Entanglement-Assisted Stabilizer Quantum Coding . . . 3
  - 1.1.3 Convolutional Stabilizer Quantum Coding . . . 4
- 1.2 Connections to Quantum Shannon Theory . . . 6
  - 1.2.1 Communication Protocols . . . 7
  - 1.2.2 Distillation Protocols . . . 9
- 1.3 Organization of this Thesis . . . 10

2 Stabilizer Quantum Codes 11
- 2.1 Review of the Stabilizer Formalism . . . 12
- 2.2 Clifford Encoding Unitary . . . 15
- 2.3 The Logical Operators and the Information Qubits . . . 16
- 2.4 The Pauli-to-Binary Isomorphism . . . 17
- 2.5 Example . . . 20
- 2.6 Closing Remarks . . . 20

3 Entanglement-Assisted Stabilizer Quantum Codes 21
- 3.1 Review of the Entanglement-Assisted Stabilizer Formalism . . . 22
- 3.2 The Symplectic Product Matrix . . . 25
- 3.3 The Rate of an Entanglement-Assisted Quantum Code . . . 26
- 3.4 Example of an Entanglement-Assisted Code . . . 27
- 3.5 Encoding Algorithm . . . 29
- 3.6 Optimal Entanglement Formulas . . . 34
- 3.7 Closing Remarks . . . 40

4 Quantum Convolutional Codes **41**
4.1 Review of the Convolutional Stabilizer Formalism . . . 42
4.1.1 Quantum Convolutional Code Definition . . . 43
4.1.2 Quantum Convolutional Code Operation . . . 45
4.2 The Pauli-to-Binary Isomorphism for Quantum Convolutional Codes . . . 46
4.3 The Shifted Symplectic Product . . . 49
4.4 Row and Column Operations . . . 54
4.5 Finite-Depth Clifford Operations . . . 56
4.6 Infinite-Depth Clifford Operations . . . 57
4.6.1 Examples of Infinite-Depth Operations . . . 58
4.6.2 General Infinite-Depth Operations . . . 63
4.6.3 Infinite-Depth Operations in Practice . . . 67
4.7 Closing Remarks . . . 68

5 Entanglement-Assisted Quantum Convolutional Coding: The CSS Case **69**
5.1 Operation of an Entanglement-Assisted Quantum Convolutional Code . . . 70
5.2 The CSS Construction . . . 72
5.3 Codes with Finite-Depth Encoding and Decoding Circuits . . . 77
5.4 Codes with Infinite-Depth Encoding and Finite-Depth Decoding Circuits . . . 83
5.4.1 Special Case with Coherent Teleportation Decoding . . . 88
5.4.2 Discussion . . . 96
5.5 Closing Remarks . . . 96

6 Entanglement-Assisted Quantum Convolutional Coding: The General Case **98**
6.1 The Expansion of Quantum Convolutional Generators . . . 100
6.1.1 Example of the Expansion . . . 100
6.1.2 General Technique for Expansion . . . 103
6.2 Polynomial Symplectic Gram-Schmidt Orthogonalization Procedure . . . 104
6.2.1 Example of the Procedure . . . 104
6.2.2 The Procedure for General Codes . . . 108
6.3 Encoding and Decoding Circuits . . . 113
6.3.1 Discussion . . . 120
6.3.2 Importing Classical Convolutional Codes over $GF(4)$ . . . 120
6.4 Example . . . 121
6.5 Optimal Entanglement Formulas . . . 125
6.6 Closing Remarks . . . 126

7 Entanglement-Assisted Quantum Convolutional Coding: Free Entanglement **128**
7.1 Construction . . . 129
7.2 Example . . . 136
7.3 Closing Remarks . . . 139

8 Unified Quantum Convolutional Coding **140**
8.1 Grandfather Quantum Convolutional Codes . . . 141
8.2 Example . . . 144
8.3 Closing Remarks . . . 148

9 Convolutional Entanglement Distillation **149**
9.1 Stabilizer Entanglement Distillation without Entanglement Assistance 151
9.2 Stabilizer Entanglement Distillation with Entanglement Assistance 153
9.3 Convolutional Entanglement Distillation without Entanglement Assistance 154
9.4 Convolutional Entanglement Distillation with Entanglement Assistance 157
9.4.1 Yield (n-1)/n Convolutional Entanglement Distillation 157
9.4.2 Yield (n-m)/n Convolutional Entanglement Distillation 162
9.4.3 CSS-Like Construction for Convolutional Entanglement Distillation 166
9.5 Closing Remarks 170

10 Conclusion **172**

BIBLIOGRAPHY **174**

List of Tables

8.1 A list of possible single-qubit errors in a particular frame and the corresponding syndrome vector. The syndrome corresponding to any single-qubit error is unique. The code therefore corrects an arbitrary single-qubit error in every other frame. . . 147

9.1 The convolutional entanglement distillation protocol for Example 9.4.1 corrects for a single-qubit error in every fourth frame. Here we list the syndromes corresponding to errors X_1, Y_1, and Z_1 on the first qubit and to errors X_2, Y_2, and Z_2 on the second qubit. The syndromes are unique so that the receiver can identify which error occurs. 159

List of Figures

2.1 The operation of a stabilizer code. Thin lines denote quantum information and thick lines denote classical information. Slanted bars denote multiple qubits. A sender encodes a multi-qubit state $|\psi\rangle$ with the help of some ancilla qubits $|0\rangle$. She sends the encoded state over a noisy quantum channel. The receiver performs multi-qubit measurements to extract information about the errors. He finally performs a recovery operation R to reverse the channel error. 13

3.1 The operation of an entanglement-assisted quantum error-correcting code. The sender encodes quantum information in state $|\psi\rangle$ with the help of local ancilla qubits $|0\rangle$ and her half of a set of shared ebits $|\Phi^+\rangle$. She then sends her qubits over a noisy quantum channel. The channel does not corrupt the receiver's half of the set of shared ebits. The receiver performs multi-qubit measurements on all of the qubits to diagnose the channel error. He performs a recovery unitary R to reverse the estimated channel error. 24

3.2 Encoding circuit for the entanglement-assisted code in the example of Section 3.4. The "H" gate is a Hadamard gate and the "P" gate is a phase gate. 32

4.1 The operation of a quantum convolutional code. The sender applies the same unitary successively to a stream of information qubits and ancilla qubits. The convolutional structure implies that the unitary overlaps some of the same qubits. The sender transmits her qubits as soon as the unitary finishes processing them. The noisy quantum channel corrupts the transmitted qubits. The receiver performs overlapping multi-qubit measurements to diagnose channel errors and corrects for them. The receiver performs an online decoding circuit to recover the sender's original stream of information qubits. 45

4.2 An example of an infinite-depth operation. A sequence of CNOT gates acts on the third qubit of every frame. This infinite-depth operation effectively multiplies the third column of the "X" side of the binary polynomial matrix by the rational polynomial $1/(1+D)$ and multiplies the third column of the "Z" side of the binary polynomial matrix by $1+D^{-1}$. 61

4.3 Another example of an infinite-depth operation. An infinite-depth operation acts on qubit i in every frame. This particular infinite-depth operation multiplies column i on the "X" side of the binary polynomial matrix by $1/\left(1+D+D^3\right)$ and multiplies column i on the "Z" side of the binary polynomial matrix by $1+D^{-1}+D^{-3}$. 67

5.1 (Color online) An entanglement-assisted quantum convolutional code operates on a stream of qubits partitioned into a countable number of frames. The sender encodes the frames of information qubits, ancilla qubits, and half of shared ebits with a repeated, overlapping encoding circuit U. The noisy channel affects the sender's encoded qubits but does not affect the receiver's half of the shared ebits. The receiver performs overlapping measurements on both the encoded qubits and his half of the shared ebits. These measurements produce an error syndrome which the receiver can process to determine the most likely error. The receiver reverses the errors on the noisy qubits from the sender. The final decoding circuit operates on all qubits in a frame and recovers the original stream of information qubits. 71
5.2 (Color online) The finite-depth encoding circuit for the entanglement-assisted quantum convolutional code in Example 5.3.1. The above operations in reverse order give a valid decoding circuit. 81

7.1 (Color online) An online encoding circuit for an entanglement-assisted quantum convolutional code. The receiver Bob possesses the first two qubits in the two ebits and the sender Alice possesses the second two qubits in the two ebits. The sender encodes two information qubits per frame with the help of her half of the two ebits. 138

9.1 An example of a block entanglement distillation protocol. A sender creates a set of noisy ebits by sending half of a set of Bell states through a noisy quantum channel. Both sender and receiver perform multi-qubit measurements to diagnose channel error. The sender transmits her measurement results to the receiver over a classical communication channel. Both perform recovery and decoding operations to obtain a set of noiseless ebits. 151
9.2 An example of a convolutional entanglement distillation protocol taken from the quantum convolutional code in Ref. [1]. The code in Ref. [1] has rate 1/3 and can correct for single-qubit errors in every other frame. Alice and Bob first measure the operators in the stabilizer for the quantum convolutional code. Alice performs conditional unitaries on her qubits to restore them to the $+1$ eigenspace of the stabilizer code. Alice forwards her measurement results to Bob. Bob performs a maximum-likelihood decoding procedure such as Viterbi decoding [2] to determine the qubit errors. He corrects for these errors. He restores his qubits to the $+1$ eigenspace of the stabilizer code. Alice and Bob both perform online decoding to obtain ebits with yield 1/3. 154
9.3 The above figure illustrates Bob's side of the convolutional entanglement distillation protocol that uses entanglement assistance. The noise affects the first and second of every three ebits that Bob shares with Alice. Every third ebit that Alice and Bob share are noiseless. The measurements correspond to those in Example 9.4.1. 161

Abstract

Quantum error-correcting codes will be the ultimate enabler of a future quantum computing or quantum communication device. This theory forms the cornerstone of practical quantum information theory. We provide several contributions to the theory of quantum error correction—mainly to the theory of "entanglement-assisted" quantum error correction where the sender and receiver share entanglement in the form of entangled bits (*ebits*) before quantum communication begins. Our first contribution is an algorithm for encoding and decoding an entanglement-assisted quantum block code. We then give several formulas that determine the optimal number of ebits for an entanglement-assisted code. The major contribution of this thesis is the development of the theory of entanglement-assisted quantum convolutional coding. A convolutional code is one that has memory and acts on an incoming stream of qubits. We explicitly show how to encode and decode a stream of information qubits with the help of ancilla qubits and ebits. Our entanglement-assisted convolutional codes include those with a Calderbank-Shor-Steane structure and those with a more general structure. We then formulate convolutional protocols that correct errors in noisy entanglement. Our final contribution is a unification of the theory of quantum error correction—these unified convolutional codes exploit all of the known resources for quantum redundancy.

Chapter 1

Introduction

If computers that you build are quantum,
Then spies of all factions will want 'em.
Our codes will all fail,
And they'll read our email,
Till we've crypto that's quantum, and daunt 'em.
—Jennifer and Peter Shor

QUANTUM computation and communication have enormous potential to revolutionize the way we compute and communicate. From communicating by teleportation of quantum information [3] to breaking RSA encryption [4, 5], quantum technological breakthroughs will change society in a way that is difficult to predict.

One might say that the field of quantum computation began when Richard Feynman and others suggested that a computer operating according to the principles of quantum mechanics would have advantages over a computer operating according to the laws of classical physics [6, 7, 8]. They suggested that it might be able to simulate quantum-mechanical processes more efficiently than a classical computer could. Later work then showed that it was possible to achieve this simulation speedup [9]. Other major advances in the theory of quantum computing followed—examples are Shor's algorithm for breaking RSA encryption [4, 5] and Grover's algorithm for searching a database [10, 11].

The field of modern quantum communication began with the discovery of quantum key distribution [12]. Major discoveries for quantum communication followed—examples are the ability to send a quantum bit using two classical bits and a bit of entanglement (quantum teleportation [3]) and the ability to send two classical bits by sending a quantum bit and consuming a bit of entanglement (quantum superdense coding [13]). Quantum information theorists then began to think more deeply about the ways in which we could combine the resources of classical communication, quantum communication, and entanglement to formulate new communication protocols.

In spite of these spectacular advances for the theories of quantum computation and communication, a dark shadow lingered over them. A few authors disputed that reliable quantum computation or communication would be possible because small quantum errors would accumulate as the computation proceeds or channel errors would destroy the quantum bits as we communicate them [14]. Rolf Landauer even urged his colleagues to include the following disclaimer in their papers on quantum computation: "This scheme, like all other schemes for quantum computation, relies on speculative technology, does not in its current form take into account all possible sources of noise, unreliability and manufacturing error, and probably will not work." Various other obstacles such as the no-cloning theorem [15] and measurement destroying a quantum state seemed to pose an insurmountable barrier to a protocol for quantum error correction.

Despite the aforementioned obstacles, Shor demonstrated the first quantum error-correcting code that reduces the negative effects of decoherence on a quantum bit [16]. Shor's code overcame all of the above difficulties and established the basic principles for constructing a general theory of quantum error correction [17, 18, 19]. Shor's code exploits many of the signature principles of quantum mechanics: superposition, entanglement, unitary evolution, and measurement. Mermin proclaims it a "miracle" that quantum error correction is even possible [20].

1.1 Quantum Error Correction

Quantum error correction theory [16, 21, 22, 17, 19] now plays a prominent role in the practical realization and engineering of quantum computing and communication devices. The first quantum error-correcting codes [16, 21, 22, 19] are strikingly similar to classical block codes [23] in their operation and performance. Quantum error-correcting codes restore a noisy, decohered quantum

state to a pure quantum state. Any future quantum information processing device will operate faithfully only if it employs an error correction scheme. This scheme can be an active scheme [17], a passive scheme [24, 25, 26], or a combination of both techniques [27, 28, 29, 30, 31].

The field of quantum error correction has rapidly expanded in recent years because there are so many ways in which one can correct quantum error. We briefly introduce three notable developments in the theory—these three are by no means exhaustive. The developments include the stabilizer formalism, the entanglement-assisted stabilizer formalism, and the convolutional stabilizer formalism.

1.1.1 Stabilizer Quantum Coding

Gottesman formalized the theory of quantum block coding by establishing the stabilizer formalism [17]. A *stabilizer* quantum error-correcting code appends ancilla qubits to information qubits that we want to protect. A unitary encoding circuit rotates the Hilbert space of the information qubits into a subspace of a larger Hilbert space. This highly entangled, encoded state corrects for local noisy errors. A quantum error-correcting code makes quantum computation and quantum communication practical by providing a way for a sender and receiver to simulate a noiseless qubit channel given a noisy qubit channel that has a particular error model.

The stabilizer theory of quantum error correction allows one to import some classical binary or quaternary codes for use as a quantum code [19]. The Calderbank-Shor-Steane (CSS) construction is the name for the method for importing classical binary codes [32]. This idea of importing codes is useful because quantum code designers can utilize classical block codes with high performance to construct quantum codes with high performance. The only "catch" when importing is that the classical code must satisfy the dual-containing or self-orthogonality constraint. Researchers have found many examples of classical codes satisfying this constraint [19], but most classical codes do not.

1.1.2 Entanglement-Assisted Stabilizer Quantum Coding

Bowen was the first to extend the stabilizer formalism by providing an example of a code that exploits entanglement shared between a sender and a receiver [33]. The underlying assumption of Bowen's code is that the sender and receiver share a set of noiseless ebits (entangled qubits)

before quantum communication begins. Many quantum protocols such as teleportation [3] and superdense coding [13] are "entanglement-assisted" protocols because they assume that noiseless ebits are available.

Brun, Devetak, and Hsieh extended the standard stabilizer theory of quantum error correction by developing the entanglement-assisted stabilizer formalism [34, 35]. They included entanglement as a resource that a sender and receiver can exploit for a quantum error-correcting code. They provided a "direct-coding" construction in which a sender and receiver can use ancilla qubits and ebits in a quantum code. An ebit is a nonlocal bipartite Bell state

$$\left|\Phi^{+}\right\rangle = \left(|00\rangle + |11\rangle\right)/\sqrt{2}.$$

Gottesman later showed that their construction is optimal [36]—it gives the minimum number of ebits required for the entanglement-assisted quantum code.

The major benefit of the entanglement-assisted stabilizer formalism is that we can construct an entanglement-assisted quantum code from two arbitrary classical binary block codes or from an arbitrary classical quaternary block code. The rate and error-correcting properties of the classical codes translate to the resulting quantum codes. The entanglement-assisted stabilizer formalism may be able to reduce the problem of finding high-performance quantum codes approaching the quantum capacity [37, 38, 39, 40, 41] to the problem of finding good classical linear codes approaching the classical capacity [42]. The entanglement-assisted stabilizer formalism thus is a significant and powerful extension of the stabilizer formalism.

1.1.3 Convolutional Stabilizer Quantum Coding

Another extension of the theory of quantum error correction protects a potentially-infinite stream of quantum information against the corruption induced by a noisy quantum communication channel [43, 44, 45, 46, 47, 48, 1, 49, 50, 51, 52, 53]. Quantum convolutional codes have numerous benefits. The periodic structure of the encoding and decoding circuits for a quantum convolutional code ensures a low complexity for encoding and decoding while also providing higher performance than a block code with equivalent encoding complexity [1]. The encoding and decoding circuits have the property that the sender Alice and the receiver Bob can respectively send and receive

qubits in an "online" fashion. Alice can encode an arbitrary number of information qubits without worrying beforehand how many she may want to send over the quantum communication channel.

We believe that quantum convolutional coding theory is one step along the way to finding quantum error-correcting codes that approach the capacity of a noisy quantum communication channel for sending quantum information [37, 38, 39, 54, 55, 56, 57]. Poulin et al. have recently incorporated some of this well-developed theory of quantum convolutional coding into a theory of quantum serial-turbo coding [58] with the goal of designing quantum codes that come close to achieving capacity.

The development of quantum convolutional codes has been brief but successful. Chau was the first to construct some quantum convolutional codes [59, 60], though some authors [1] argue that his construction does not give a true quantum convolutional code. Several authors have established a working theory of quantum convolutional coding based on the stabilizer formalism and classical self-orthogonal codes over the finite field $GF(4)$ [43, 44, 48, 1]. Others have also provided a practical way for realizing "online" encoding and decoding circuits for quantum convolutional codes [43, 44, 46, 45].

Forney et al. have determined a method for importing an arbitrary classical self-orthogonal quaternary code for use as a quantum convolutional code [48, 1]. The technique is similar to that for importing a classical block code as a quantum block code [19]. Forney et al. showed how to produce quantum convolutional codes from classical convolutional codes by extending the ideas from the CSS construction to the convolutional setting [1]; but again, these imported classical convolutional codes have to satisfy the restrictive dual-containing constraint in order to form valid quantum codes. The dual-containing constraint is actually quite a bit more restrictive in the convolutional case because each generator for the quantum convolutional code must commute not only with the other generators, but it must commute also with any arbitrary shift of itself and any arbitrary shift of the other generators. Forney et al. performed specialized searches to determine classical quaternary codes that satisfy the restrictive self-orthogonality constraint [1].

1.2 Connections to Quantum Shannon Theory

Quantum error correction theory gives practical ways to build codes for protection of quantum information against decoherence. A quantum code typically encodes some number k of information qubits into a larger number n of physical qubits. We say that the rate of quantum communication for this code is k/n.

One may ask the question: what is the maximum rate of quantum communication for a given noisy quantum communication channel? This question is the fundamental question of quantum Shannon theory and the answer to it is a quantity known as the capacity of the quantum channel for quantum communication, or more simply, the quantum capacity [37, 38, 39, 61]. Quantum Shannon theory has many other questions. What is the rate of classical communication for a noisy quantum communication channel? What is the rate if the sender and receiver have access to entanglement? What is the rate of simultaneous classical and quantum communication? The list of questions goes on and on.

The goal of quantum Shannon theory is to quantify the amount of quantum communication, classical communication, and entanglement required for various information processing tasks [37, 38, 39, 61]. Quantum teleportation and superdense coding provided the initial impetus for quantum Shannon theory because these protocols demonstrate that we can combine entanglement, noiseless quantum communication, and noiseless classical communication to transmit quantum or classical information. In practice, the above resources are not noiseless because quantum systems decohere by interacting with their surrounding environment. Quantum Shannon theory is a collection of capacity theorems that determine the ultimate limits for noisy quantum communication channels. Essentially all quantum protocols have been unified as special cases of a handful of abstract protocols [61].

The techniques in quantum Shannon theory determine the asymptotic limits for communication, but these techniques do not produce practical ways of realizing these limits. This same practical problem exists with classical Shannon theory because random coding techniques determine the limit of a noisy classical communication channel [62].

1.2.1 Communication Protocols

The fundamental theorem of quantum Shannon theory is the celebrated Lloyd-Shor-Devetak quantum capacity theorem. It determines the capacity of a given noisy quantum communication channel for reliable quantum communication. Suppose that a noisy quantum channel $\mathcal{N}$ connects a sender Alice A to a receiver Bob B. Let $[q \rightarrow q]$ denote one qubit of noiseless quantum communication. We can state the coding theorem as a resource inequality:

$$\langle \mathcal{N} \rangle \geq Q\left[q \rightarrow q\right], \tag{1.1}$$

The above resource inequality states that n uses of the noisy quantum channel $\mathcal{N}$ are sufficient to communicate nQ noiseless qubits in the limit of a large number n of channel uses. The rate Q is equal to a quantity known as the coherent information: $I\left(A\rangle B\right)$ [32]. The entropic quantity is maximized with respect to any resource state $|\psi\rangle^{ABE}$ associated to the noisy channel $\mathcal{N}$ and shared between Alice, Bob, and the environment E. The goal of the stabilizer codes and convolutional stabilizer codes mentioned respectively in Sections 1.1.1 and 1.1.3 is for their rates to approach the maximum capacity given above.

Another example of an important capacity theorem from quantum Shannon theory results from the "father" protocol [61]. The father capacity theorem determines the optimal trade-off between the rate E of ebits (entangled qubits in the state $|\Phi^{+}\rangle^{AB} \equiv (|00\rangle^{AB} + |11\rangle^{AB})/\sqrt{2}$) and the rate Q of qubits in entanglement-assisted quantum communication. This protocol is the "father" protocol because it generates many of the protocols in the family tree of quantum information theory [61]. The nickname "father" is a useful shorthand for classifying the protocol—there exists a mother, grandmother, and grandfather protocol as well. Let $[qq]$ denote one ebit of entanglement. The following resource inequality is a statement of the capability of the father protocol:

$$\langle \mathcal{N} \rangle + E\left[qq\right] \geq Q\left[q \rightarrow q\right]. \tag{1.2}$$

The above resource inequality states that n uses of the noisy quantum channel $\mathcal{N}$ and nE noiseless ebits are sufficient to communicate nQ noiseless qubits in the limit of large n. The rates E and Q are respectively equal to $\frac{1}{2}I\left(A;E\right)$ and $\frac{1}{2}I\left(A;B\right)$. The entropic quantities are maximized with

respect to any resource state $|\psi\rangle^{ABE}$ associated to the noisy channel $\mathcal{N}$ and shared between Alice, Bob, and the environment E (the rate E is different from the environment E). The father capacity theorem gives the optimal limits on the resources, but it does not provide a useful quantum coding technique for approaching the above limits. The goal of the entanglement-assisted stabilizer formalism mentioned in Section 1.1.2 is to develop quantum codes that approach the maximal rates given in the father capacity theorem. We spend a significant portion of this thesis (Chapters 5, 6, and 7) developing entanglement-assisted convolutional stabilizer codes. Future research might be able to incorporate these convolutional codes into a framework for designing codes that come close to achieving the maximal rates.

Another important capacity theorem determines the ability of a noisy quantum channel to send "classical-quantum" states [63]. Let $[c \rightarrow c]$ denote one bit of noiseless classical communication. The result of the classical-quantum capacity theorem is also a resource inequality:

$$\langle \mathcal{N} \rangle \geq Q\left[q \rightarrow q\right] + R\left[c \rightarrow c\right]. \tag{1.3}$$

The resource inequality states that n uses of the noisy quantum channel $\mathcal{N}$ are sufficient to communicate nQ noiseless qubits and nR noiseless classical bits in the limit of large n. The rates Q and R are respectively equal to $I\left(X;B\right)$ and $I\left(A\rangle BX\right)$. The entropic quantities are with respect to any state resulting from sending the A' system of the following classical-quantum state

$$\sum_x p_x \left|x\right\rangle \left\langle x\right|^X \otimes \left|\phi_x\right\rangle \left\langle \phi_x\right|^{AA'} \tag{1.4}$$

through the quantum channel $\mathcal{N}^{A'\rightarrow B}$. X and A are systems that the sender keeps. This theorem proves that we can devise clever classical-quantum codes that perform better than time-sharing a noisy quantum channel $\mathcal{N}$ between purely quantum codes and purely classical codes.

The "grandfather" capacity theorem determines the optimal triple trade-off between qubits, ebits, and classical bits for simultaneous transmission of classical and quantum information using an entanglement-assisted noisy quantum channel $\mathcal{N}$ [64]. The grandfather resource inequality is as follows:

$$\langle \mathcal{N} \rangle + E\left[qq\right] \geq Q\left[q \rightarrow q\right] + R\left[c \rightarrow c\right]. \tag{1.5}$$

The above resource inequality is again an asymptotic statement and its meaning is similar to that in (1.1), (1.2), and (1.3). The optimal rates in the above resource inequality coincide with the father inequality (1.2) when $R = 0$, with the classical-quantum inequality (1.3) when $E = 0$, and with the quantum capacity (1.1) when both $R = 0$ and $E = 0$. The optimal strategy for the grandfather protocol is not time-sharing the channel between father codes and entanglement-assisted classical codes. It remains to be proven whether this optimal strategy outperforms time-sharing [64]. We develop grandfather convolutional codes in Chapter 8. Again, future research might be able to incorporate these grandfather convolutional codes into a framework for designing codes that come close to achieving the maximal limits given in the grandfather capacity theorem.

1.2.2 Distillation Protocols

The aforementioned protocols employ a noisy quantum communication channel in their operation. We now ask the question: given noisy entanglement, is it possible to produce noiseless entanglement using classical communication? Indeed, it is possible, and this procedure is entanglement distillation [65, 66]. The following resource inequality summarizes the resources consumed and produced in an entanglement distillation protocol:

$$\left\langle \rho^{AB} \right\rangle + R\left[c \rightarrow c\right] \geq Q\left[qq\right].$$

The statement of the resource inequality is as follows: given a large number n of noisy bipartite states ρ^{AB} shared between two parties, it is possible to generate nQ noiseless ebits by consuming nR classical bits in the limit of a large number n of copies of the noisy state ρ^{AB}. The optimal rates are $R = I\left(A;E\right)$ and $Q = I\left(A\rangle B\right)$ where the entropic quantities are with respect to a state $\left|\psi\right\rangle^{ABE}$ that purifies the state ρ^{AB}. In Chapter 9, we develop convolutional entanglement distillation protocols whose goal is to approach the optimal rates given in the above resource inequality.

1.3 Organization of this Thesis

This thesis represents a significant contribution to quantum error correction theory. In it, we give several methods for designing quantum codes. These methods should be useful in the future when coherent quantum encoding devices are available. The quantum code designer will have a plethora of techniques to exploit for protecting quantum information and the techniques in this thesis should be part of the arsenal.

A quick glance over the thesis reveals that much of the mathematics involves only linear algebra over matrices of binary numbers or binary polynomials. It is a testament to the strength of the theory that it reduces to this simple form. We show in detail especially in Chapters 2 and 4 how to reduce the theory to have this simple form. This thesis has contributions from the following publications [52, 67, 68, 69, 53, 51, 70].

We structure the thesis as follows. The next chapter introduces the stabilizer formalism for quantum error correction. We show in this chapter how to represent quantum codes with binary matrices. Chapter 3 introduces the entanglement-assisted stabilizer formalism and shows how to enhance the stabilizer formalism by assuming that the sender and receiver share entanglement. This chapter gives two contributions: a method for determining the encoding circuit for a quantum block code and a method to determine the minimal amount of entanglement that an entanglement-assisted code requires. Chapter 4 reviews the convolutional stabilizer formalism. In particular, it details the operation of a quantum convolutional code and shows how to simplify the mathematics with matrices whose entries are binary polynomials. Chapters 5, 6, and 7 form the heart of this thesis. They show how to construct entanglement-assisted quantum codes that have a convolutional structure. The techniques developed represent a significant extension of the entanglement-assisted stabilizer formalism. Chapter 8 presents a unifying structure for quantum error correction under which most current schemes for quantum error correction fall. We present a method for distilling entanglement in a convolutional manner in Chapter 9. The major benefit of all of the above convolutional methods is that we can import arbitrary high-performance classical codes to construct high-performance quantum codes.

Chapter 2

Stabilizer Quantum Codes

"Fight entanglement with entanglement," you say,
'Cause entanglement keeps Eve away,
The hell that she raises,
She damps and dephases,
She's no match—like UCLA!

The stabilizer formalism is a mathematical framework for quantum error correction [71, 17]. This framework has many similarities with classical coding theory, and it is even possible to import a classical code for use in quantum error correction by employing the CSS construction [21, 22, 32]. We review the stabilizer theory for quantum block codes.

The stabilizer formalism is versatile because it appears not only in quantum error correction but also in other subjects of quantum information theory such as stabilizer or graph states [72], cluster-state quantum computation [73], and entanglement theory [74]. The stabilizer formalism simplifies the theory of quantum error correction by reducing the mathematics to linear algebra over binary matrices.

2.1 Review of the Stabilizer Formalism

The stabilizer formalism exploits elements of the Pauli group Π. The set $\Pi = \{I, X, Y, Z\}$ consists of the Pauli operators:

$$I \equiv \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \; X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \; Y \equiv \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \; Z \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

The above operators act on a single qubit—a vector in a two-dimensional Hilbert space—and are the most important in formulating a quantum error-correcting code. Two crucial properties of these matrices are useful: each matrix in Π has eigenvalues equal to $+1$ or -1, and any two matrices in Π either commute or anticommute.

In general, a quantum error-correcting code uses n physical qubits to protect a smaller set of information qubits against decoherence or quantum noise. An n-qubit quantum error-correcting code employs elements of the Pauli group Π^n. The set Π^n consists of n-fold tensor products of Pauli operators:

$$\Pi^n = \left\{ e^{i\phi} A_1 \otimes \cdots \otimes A_n : \forall j \in \{1, \ldots, n\}, \quad A_j \in \Pi, \quad \phi \in \{0, \pi/2, \pi, 3\pi/2\} \right\}. \tag{2.1}$$

Matrices in Π^n act on a 2^n-dimensional complex vector, or equivalently, an n-qubit quantum register. We omit tensor product symbols in what follows so that $A_1 \cdots A_n \equiv A_1 \otimes \cdots \otimes A_n$. The above two crucial properties for the single-qubit Pauli group Π still hold for the Pauli group Π^n (up to an irrelevant phase for the eigenvalue property). The n-fold Pauli group Π^n plays an important role in both the encoding circuit and the error-correction procedure of a quantum stabilizer code over n qubits.

We can phrase the theory of quantum error correction in purely mathematical terms using elements of Π^n. Consider a matrix $g_1 \in \Pi^n$ that is not equal to $\pm I$. Matrix g_1 then has two eigenspaces each of dimension 2^{n-1}. We can identify one eigenspace with the eigenvalue $+1$ and the other eigenspace with eigenvalue -1. Consider a matrix $g_2 \in \Pi^n$ different from both g_1 and the identity that commutes with g_1. Matrix g_2 also has two eigenspaces each of size 2^{n-1} and identified similarly by its eigenvalues ± 1. Then g_1 and g_2 have simultaneous eigenspaces because they commute. These matrices together have four different eigenspaces, each of size 2^{n-2} and

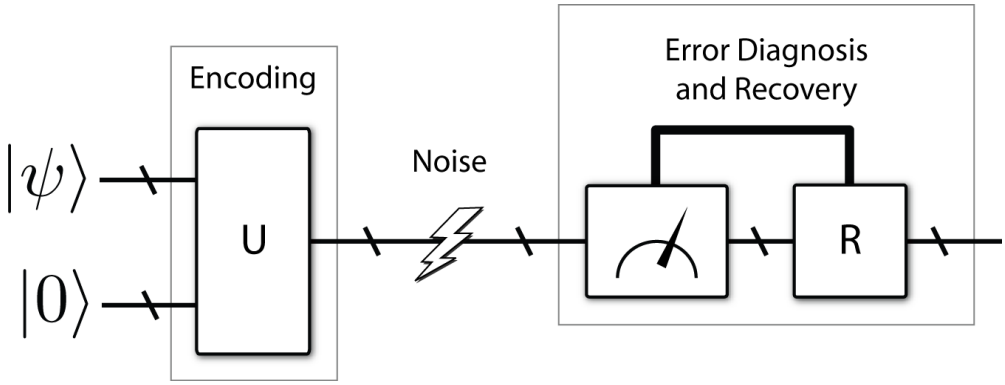

Figure 2.1: The operation of a stabilizer code. Thin lines denote quantum information and thick lines denote classical information. Slanted bars denote multiple qubits. A sender encodes a multi-qubit state $|\psi\rangle$ with the help of some ancilla qubits $|0\rangle$. She sends the encoded state over a noisy quantum channel. The receiver performs multi-qubit measurements to extract information about the errors. He finally performs a recovery operation R to reverse the channel error.

identified by the eigenvalues $\pm 1, \pm 1$ of g_1 and g_2 respectively. We can continue this process of adding more commuting and independent matrices to a set $\mathcal{S}$. The matrices in $\mathcal{S}$ are independent in the sense that no matrix in $\mathcal{S}$ is a product of two or more other matrices in $\mathcal{S}$. Adding more matrices from Π^n to $\mathcal{S}$ continues to divide the eigenspaces of matrices in $\mathcal{S}$. In general, suppose $\mathcal{S}$ consists of $n-k$ independent and commuting matrices $g_1, \ldots, g_{n-k} \in \Pi^n$. These $n-k$ matrices then have 2^{n-k} different eigenspaces each of size 2^k and identified by the eigenvalues $\pm 1, \ldots, \pm 1$ of $g_1, \ldots, g_{n-k}$ respectively. Consider that the Hilbert space of k qubits has size 2^k. A dimension count immediately suggests that we can encode k qubits into one of the eigenspaces of $\mathcal{S}$. We typically encode these k qubits into the simultaneous $+1$-eigenspace of $g_1, \ldots, g_{n-k}$. This eigenspace is the *codespace*. The operators $g_1, \ldots, g_{n-k}$ function in the same way as a parity check matrix does for a classical linear block code. An $[n, k]$ quantum error-correcting code encodes k information qubits into the simultaneous $+1$-eigenspace of $n-k$ matrices $g_1, \ldots, g_{n-k} \in \Pi^n$. Its stabilizer $\mathcal{S}$ is an abelian subgroup of the n-fold Pauli group Π^n: $\mathcal{S} \subset \Pi^n$. Note that it is possible to multiply the generators in the stabilizer together and obtain an equivalent representation of the stabilizer in much the same way that we can add vectors in a basis together and obtain an equivalent basis.

The operation of an $[n, k]$ quantum error-correcting code consists of four steps. Figure 2.1 highlights these steps.

(i). A unitary operation U encodes both k information qubits in a general state $|\psi\rangle$ and $n-k$ ancilla qubits in the state $|0\rangle^{\otimes n-k}$ into the simultaneous $+1$-eigenspace of the matrices g_1, $\ldots$, g_{n-k}.

(ii). The sender transmits the n encoded qubits by using the noisy quantum communication channel n times.

(iii). The receiver performs quantum measurements of the $n-k$ matrices g_1, $\ldots$, g_{n-k} in the stabilizer $\mathcal{S}$. These measurements learn only about errors that may occur and do not disturb the encoded quantum information. Each measurement gives a bit result equal to $+1$ or -1, and the result of all the measurements is to project the n-qubit quantum register into one of the 2^{n-k} different eigenspaces of g_1, $\ldots$, g_{n-k}. Suppose that no error occurs. Then the measurements project the n qubits into the simultaneous $+1$-eigenspace and return a bit vector consisting of $n-k$ ones. This "projection of the error concept" is one of the fundamental notions in quantum error correction theory. It suffices to correct a discrete error set with support in the Pauli group Π^n [16]. Now suppose that a quantum error in an error set $\mathcal{E} \subset \Pi^n$ occurs. The error takes the encoded quantum state out of the codespace and into one of the other $2^{n-k}-1$ orthogonal eigenspaces. The measurements can detect that an error has occurred because the result of the measurements is a bit vector differing from the all ones vector. The receiver can identify uniquely which error in $\mathcal{E}$ has occurred if the set $\mathcal{E}$ satisfies the following quantum error correction conditions:

$$\forall E_a, E_b \in \mathcal{E} \qquad \exists\ g_i \in \mathcal{S} : \left\{g_i, E_a^\dagger E_b\right\} = 0 \text{ or } E_a^\dagger E_b \in \mathcal{S}. \tag{2.2}$$

The first condition corresponds to the active error-correcting capability of the code, and the second condition corresponds to its passive error-correcting capability.

(iv). If the receiver can identify which error occurs, he can then apply a unitary operation R that is the inverse of the error. He finally performs a decoding unitary that decodes the k information qubits.

2.2 Clifford Encoding Unitary

We comment briefly on the encoding operation U. The encoding operation U is a special type of unitary matrix called a Clifford operation. A Clifford operation U is one that preserves elements of the Pauli group under conjugation: $A \in \Pi^n \Rightarrow UAU^\dagger \in \Pi^n$. The CNOT gate, the Hadamard gate H, and the phase gate P suffice to implement any unitary matrix in the Clifford group [17]. A quantum code with the CSS structure needs only the CNOT and Hadamard gates for encoding and decoding. The matrix for the CNOT gate acting on two qubits is

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \tag{2.3}$$

the matrix for the Hadamard gate H acting on a single qubit is

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \tag{2.4}$$

and the matrix for the phase gate P acting on a single qubit is

$$P = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}. \tag{2.5}$$

The standard basis for Π^1 is X and Z because any element of Π^1 is a product of elements in this generating set up to an irrelevant phase. The standard basis for elements of the two-qubit Pauli group Π^2 is as follows

$$Z \quad I\ , \quad I \quad Z\ , \quad X \quad I\ , \quad I \quad X\ , \tag{2.6}$$

for the same reasons.

The Hadamard gate H transforms the standard basis of Π^1 under conjugation as follows:

$$Z \to X, \quad X \to Z \tag{2.7}$$

and the phase gate P transforms the standard basis as follows:

$$Z \to Z, \quad X \to Y. \tag{2.8}$$

For the CNOT gate, the first qubit is the "control" qubit and the second qubit is the "target" qubit. The CNOT gate transforms the standard basis of Π^2 under conjugation as follows

$$Z\ \ I \ \to\ Z\ \ I\ , \quad I\ \ Z\ \to\ Z\ \ Z\ , \quad X\ \ I\ \to\ X\ \ X\ , \quad I\ \ X\ \to\ I\ \ X\ , \tag{2.9}$$

where the first qubit is the control and the second qubit is the target.

Section 3.5 of Chapter 3 details an algorithm that determines a Clifford encoding circuit for any stabilizer code or any entanglement-assisted stabilizer code (we review entanglement-assisted codes in the next chapter).

2.3 The Logical Operators and the Information Qubits

Another aspect of the theory of quantum error correction is later useful for our purposes in quantum convolutional coding. This aspect concerns the information qubits and the operators that change them. Consider that the initial unencoded state of a quantum error-correcting code is a simultaneous $+1$-eigenstate of the matrices $Z_{k+1}, \ldots, Z_n$ where Z_i has a Z matrix operating on qubit i and the identity I on all other qubits. Therefore, the matrices $Z_{k+1}, \ldots, Z_n$ constitute a stabilizer for the unencoded state. The initial unencoded logical operators for the k information qubits are $Z_1, X_1, \ldots, Z_k, X_k$. The encoding operation U rotates the unencoded stabilizer matrices $Z_{k+1}, \ldots, Z_n$ and the unencoded logical operators $Z_1, X_1, \ldots, Z_k, X_k$ to the encoded stabilizer operators $\bar{Z}_{k+1}, \ldots, \bar{Z}_n$ and the encoded logical operators $\bar{Z}_1, \bar{X}_1, \ldots, \bar{Z}_k, \bar{X}_k$ respectively. The encoded matrices $\bar{Z}_{k+1}, \ldots, \bar{Z}_n$ are respectively equivalent to the matrices $g_1, \ldots, g_{n-k}$ in the discussion of the previous section.

The encoded operators obey the same commutation relations as their unencoded counterparts. We would violate the uncertainty principle if this invariance did not hold. Therefore, each of the encoded logical operators commutes with elements of the stabilizer $\mathcal{S}$. Let A denote an arbitrary logical operator from the above set and let $\bar{Z}_i$ denote an arbitrary element of the stabilizer $\mathcal{S}$. The operator $A\bar{Z}_i$ (or equivalently $\bar{Z}_i A$) is an equivalent logical operator because $A\bar{Z}_i$ and A have the same effect on an encoded state $\left|\bar{\psi}\right\rangle$:

$$\bar{Z}_i A \left|\bar{\psi}\right\rangle = A\bar{Z}_i \left|\bar{\psi}\right\rangle = A \left|\bar{\psi}\right\rangle . \tag{2.10}$$

We make extensive use of the above fact in our work.

The logical operators also provide a useful way to characterize the information qubits. Gottesman showed that the logical operators for the information qubits provide a straightforward way to characterize the information qubits as they progress through a quantum circuit [17]. As an example of this technique, he develops quantum teleportation in the stabilizer formalism. The logical operators at the beginning of the protocol are X_1 and Z_1 and become X_3 and Z_3 at the end of the protocol. This transformation implies that the quantum information in qubit one teleports to qubit three because the logical operators act on only qubit three at the end of the protocol. We use the same idea throughout this thesis to determine if our decoding circuits have truly decoded the information qubits.

2.4 The Pauli-to-Binary Isomorphism

A simple but useful mapping exists between elements of Π and the binary vector space $(\mathbb{Z}_2)^2$. This mapping gives a simplification of quantum error correction theory. It represents quantum codes with binary vectors and binary operations rather than with Pauli operators and matrix operations respectively. We first give the mapping for the one-qubit case. Suppose $[A]$ is a set of equivalence classes of an operator A that have the same phase:

$$[A] = \{\beta A \mid \beta \in \mathbb{C},\ |\beta| = 1\} . \tag{2.11}$$

Let $[\Pi]$ be the set of phase-free Pauli operators where $[\Pi] = \{[A] \mid A \in \Pi\}$. Define the map $N : (\mathbb{Z}_2)^2 \to \Pi$ as

Π	I	X	Y	Z
$(\mathbb{Z}_2)^2$	00	01	11	10

. (2.12)

Suppose $u, v \in (\mathbb{Z}_2)^2$. Let us employ the shorthand $u = [z|x]$ and $v = [z'|x']$ where z, x, z', $x' \in \mathbb{Z}_2$. For example, suppose $u = [0|1]$. Then $N(u) = X$. The map N induces an isomorphism $[N] : (\mathbb{Z}_2)^2 \to [\Pi]$ because addition of vectors in $(\mathbb{Z}_2)^2$ is equivalent to multiplication of Paulis up to a global phase:

$$[N(u+v)] = [N(u)][N(v)]. \tag{2.13}$$

We name this isomorphism the *P2B isomorphism* (for Pauli-to-binary isomorphism).

Let $\odot$ denote the *symplectic product* between two elements $u, v \in (\mathbb{Z}_2)^2$:

$$u \odot v \equiv zx' - xz'. \tag{2.14}$$

The symplectic product $\odot$ gives the commutation relations of elements of Π:

$$N(u) N(v) = (-1)^{(u \odot v)} N(v) N(u).$$

The symplectic product and the P2B isomorphism $[N]$ thus give a useful way to phrase Pauli relations in terms of binary algebra.

The extension of the above definitions and the P2B isomorphism $[N]$ to multiple qubits is straightforward. Let $\mathbf{A} = A_1 \otimes \cdots \otimes A_n$ denote an arbitrary element of Π^n. We can similarly define the phase-free n-qubit Pauli group $[\Pi^n] = \{[\mathbf{A}] \mid \mathbf{A} \in \Pi^n\}$ where

$$[\mathbf{A}] = \{\beta \mathbf{A} \mid \beta \in \mathbb{C},\ |\beta| = 1\}. \tag{2.15}$$

The group operation $*$ for the above equivalence class is as follows:

$$[\mathbf{A}] * [\mathbf{B}] \equiv [A_1] * [B_1] \otimes \cdots \otimes [A_n] * [B_n] = [A_1 B_1] \otimes \cdots \otimes [A_n B_n] = [\mathbf{AB}]. \tag{2.16}$$

The equivalence class $[\Pi^n]$ forms a commutative group under operation $*$. Consider the $2n$-dimensional vector space

$$(\mathbb{Z}_2)^{2n} = \{(\mathbf{z}, \mathbf{x}) : \mathbf{z}, \mathbf{x} \in (\mathbb{Z}_2)^n\}. \tag{2.17}$$

It forms the commutative group $((\mathbb{Z}_2)^{2n}, +)$ with operation $+$ defined as binary vector addition. We employ the notation $\mathbf{u} = [\mathbf{z}|\mathbf{x}]$ and $\mathbf{v} = [\mathbf{z}'|\mathbf{x}']$ to represent any two vectors $\mathbf{u}, \mathbf{v} \in (\mathbb{Z}_2)^{2n}$ respectively. Each vector $\mathbf{z}$ and $\mathbf{x}$ has elements $(z_1, \ldots, z_n)$ and $(x_1, \ldots, x_n)$ respectively with similar representations for $\mathbf{z}'$ and $\mathbf{x}'$. The *symplectic product* $\odot$ of $\mathbf{u}$ and $\mathbf{v}$ is

$$\mathbf{u} \odot \mathbf{v} \equiv \sum_{i=1}^{n} z_i x_i' - x_i z_i' = \sum_{i=1}^{n} u_i \odot v_i, \tag{2.18}$$

where $u_i = [z_i|x_i]$ and $v_i = [z_i'|x_i']$. Let us define a map $\mathbf{N} : (\mathbb{Z}_2)^{2n} \to \Pi^n$ as follows:

$$\mathbf{N}(\mathbf{u}) \equiv N(u_1) \otimes \cdots \otimes N(u_n). \tag{2.19}$$

Let

$$\mathbf{X}(\mathbf{x}) \equiv X^{x_1} \otimes \cdots \otimes X^{x_n}, \qquad \mathbf{Z}(\mathbf{z}) \equiv Z^{z_1} \otimes \cdots \otimes Z^{z_n}, \tag{2.20}$$

so that $\mathbf{N}(\mathbf{u})$ and $\mathbf{Z}(\mathbf{z})\mathbf{X}(\mathbf{x})$ belong to the same equivalence class:

$$[\mathbf{N}(\mathbf{u})] = [\mathbf{Z}(\mathbf{z})\mathbf{X}(\mathbf{x})]. \tag{2.21}$$

The map $[\mathbf{N}] : (\mathbb{Z}_2)^{2n} \to [\Pi^n]$ is an isomorphism for the same reason given in (2.13):

$$[\mathbf{N}(\mathbf{u} + \mathbf{v})] = [\mathbf{N}(\mathbf{u})][\mathbf{N}(\mathbf{v})], \tag{2.22}$$

where $\mathbf{u}, \mathbf{v} \in (\mathbb{Z}_2)^{2n}$. The symplectic product captures the commutation relations of any operators $\mathbf{N}(\mathbf{u})$ and $\mathbf{N}(\mathbf{v})$:

$$\mathbf{N}(\mathbf{u})\mathbf{N}(\mathbf{v}) = (-1)^{(\mathbf{u}\odot\mathbf{v})}\mathbf{N}(\mathbf{v})\mathbf{N}(\mathbf{u}). \tag{2.23}$$

The above P2B isomorphism and symplectic algebra are useful in making the relation between classical linear error correction and quantum error correction more explicit.

2.5 Example

An example of a stabilizer code is the five qubit $[5, 1]$ stabilizer code [75, 66]. It encodes $k = 1$ logical qubit into $n = 5$ physical qubits and protects against an arbitrary single-qubit error. Its stabilizer consists of $n - k = 4$ Pauli operators:

$$\begin{array}{rcccccc} g_1 & = & X & Z & Z & X & I \\ g_2 & = & I & X & Z & Z & X \\ g_3 & = & X & I & X & Z & Z \\ g_4 & = & Z & X & I & X & Z \end{array} \tag{2.24}$$

The above operators commute. Therefore the codespace is the simultaneous $+1$-eigenspace of the above operators. Suppose a single-qubit error occurs on the encoded quantum register. A single-qubit error is in the set $\{X_i, Y_i, Z_i\}$ where A_i denotes a Pauli error on qubit i. It is straightforward to verify that any arbitrary single-qubit error has a unique syndrome. The receiver corrects any single-qubit error by identifying the syndrome and applying a corrective operation.

2.6 Closing Remarks

The mathematics developed in this chapter form the basis for the work in later chapters. In particular, the binary representation of quantum codes is important. The role of the "twisted" symplectic product plays an important role for the entanglement-assisted codes of the next chapter because it helps in determining the commutation relations of an arbitrary (not necessarily commuting) set of Pauli generators.

Chapter 3

Entanglement-Assisted Stabilizer Quantum Codes

Classical to quantum made dinero,
But the twisted strange product was zero,
Little did we know,
What Entanglement could show,
And again become quantum's good hero.

The entanglement-assisted stabilizer formalism is a significant extension of the standard stabilizer formalism that incorporates shared entanglement as a resource for protecting quantum information [35, 34]. The advantage of entanglement-assisted stabilizer codes is that the sender can exploit the error-correcting properties of an arbitrary set of Pauli operators. The sender's Pauli operators do not necessarily have to form an abelian subgroup of Π^n. The sender can make clever use of her shared ebits so that the global stabilizer is abelian and thus forms a valid quantum error-correcting code. Figure 3.1 demonstrates the operation of a generic entanglement-assisted stabilizer code.

Several references provide a review of this technique and generalizations of the basic theory to block entanglement distillation [76], continuous-variable codes [70], and entanglement-assisted operator codes for discrete-variable [30, 31] and continuous-variable systems [68]. Chapters 5, 6, 7, 8, and 9 of this thesis extend the entanglement-assisted stabilizer formalism to many different "convolutional" coding scenarios. We explain what we mean by "convolutional" in the next chapter.

For now, we provide a review of the basics of entanglement-assisted coding. This chapter also gives two original contributions: a method to determine the encoding and decoding circuit for an entanglement-assisted code and several formulas that determine the number of ebits that an arbitrary entanglement-assisted block code requires.

3.1 Review of the Entanglement-Assisted Stabilizer Formalism

The fundamental unit of bipartite entanglement is the *ebit*. We must first understand the commutativity properties of the ebit before developing the entanglement-assisted formalism. We express the state $|\Phi^+\rangle$ of an ebit shared between a sender Alice and a receiver Bob as follows:

$$\left|\Phi^+\right\rangle \equiv \frac{|00\rangle^{AB} + |11\rangle^{AB}}{\sqrt{2}}. \tag{3.1}$$

The two operators that stabilize this ebit state are $X^A X^B$ and $Z^A Z^B$. These two operators commute,

$$\left[X^A X^B, Z^A Z^B\right] = 0,$$

but the local operators (operating only on either party) anticommute,

$$\left\{X^A, Z^A\right\} = \left\{X^B, Z^B\right\} = 0.$$

The above commutation relations hint at a way that we can resolve anticommutativity in a set of generators. Suppose that we have two generators that anticommute. We can use an ebit of entanglement to resolve the anticommutativity in the two generators. We explain this idea in more detail in what follows.

We review the general construction of an entanglement-assisted code. An $[n, k; c]$ entanglement-assisted code employs c ebits and a ancilla qubits to encode k information qubits. Suppose that there is a nonabelian subgroup $\mathcal{S} \subset \Pi^n$ of size $2c + a$. Application of the fundamental theorem of

symplectic geometry* [77] (Lemma 1 in [34]) states that there exists a minimal set of independent generators $\{\bar{Z}_1, \ldots, \bar{Z}_{a+c}, \bar{X}_{a+1}, \ldots, \bar{X}_{a+c}\}$ for $\mathcal{S}$ with the following commutation relations:

$$\begin{aligned} \forall i,j \quad & [\bar{Z}_i, \bar{Z}_j] = 0, \quad \forall i,j \quad [\bar{X}_i, \bar{X}_j] = 0, \\ \forall i \neq j \quad & [\bar{X}_i, \bar{Z}_j] = 0, \quad \forall i \quad \{\bar{X}_i, \bar{Z}_i\} = 0. \end{aligned} \tag{3.2}$$

The decomposition of $\mathcal{S}$ into the above minimal generating set determines that the code requires a ancilla qubits and c ebits. The parameters a and c generally depend on the set of generators in $\mathcal{S}$.

There exists a symplectic Gram-Schmidt orthogonalization procedure that gives the decomposition [35, 34, 51, 78]. Specifically, the algorithm performs row operations (multiplication of the Pauli generators) that do not affect the code's error-correcting properties and thus gives a set of generators that form an equivalent code. The decomposition also minimizes the number of ebits required for the code and we prove this optimality in Section 3.6.

We present a simple stabilizer version of the symplectic Gram-Schmidt algorithm. Suppose we have a set of m generators $g_1, \ldots, g_m$. Consider g_1. It either commutes with all other generators or it anticommutes with at least one other generator. Remove it from the set if it commutes with all other generators. Now suppose that it does not, i.e., it anticommutes with one other generator g_i. Relabel g_2 as g_i and vice versa. Recall that we can multiply generators without changing the error-correcting properties of the set of generators. So we perform several multiplications to change the commutation relations to have the standard commutation relations in (3.2). We perform the following manipulations on the generators $g_3, \ldots, g_m$:

$$g_j = g_j \cdot g_1^{f(g_2,g_j)} \cdot g_2^{f(g_1,g_j)} \qquad \forall\ i \in \{3, \ldots, m\}$$

where the function f is equal to zero if its two arguments commute and it is equal to one if its two arguments anticommute. (This function is the symplectic product.) Remove the generators g_1 and g_2 from the set. Repeat the algorithm on the remaining generators. When the algorithm

*We loosely refer to this theorem as the fundamental theorem of symplectic geometry because of its importance in symplectic geometry and in quantum coding theory.

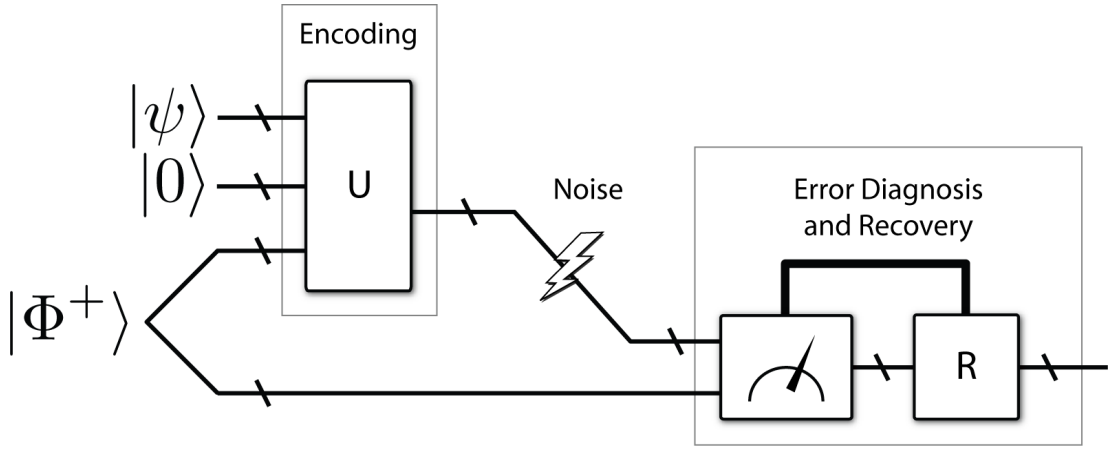

Figure 3.1: The operation of an entanglement-assisted quantum error-correcting code. The sender encodes quantum information in state $|\psi\rangle$ with the help of local ancilla qubits $|0\rangle$ and her half of a set of shared ebits $|\Phi^+\rangle$. She then sends her qubits over a noisy quantum channel. The channel does not corrupt the receiver's half of the set of shared ebits. The receiver performs multi-qubit measurements on all of the qubits to diagnose the channel error. He performs a recovery unitary R to reverse the estimated channel error.

finishes, all the "removed' generators constitute the generators for the code and have the standard commutation relations in (3.2).

We can partition the nonabelian group $\mathcal{S}$ into two subgroups: the isotropic subgroup $\mathcal{S}_I$ and the entanglement subgroup $\mathcal{S}_E$. The isotropic subgroup $\mathcal{S}_I$ is a commuting subgroup of $\mathcal{S}$ and thus corresponds to ancilla qubits: $\mathcal{S}_I = \{\bar{Z}_1, \ldots, \bar{Z}_a\}$. The elements of the entanglement subgroup $\mathcal{S}_E$ come in anticommuting pairs and thus correspond to halves of ebits: $\mathcal{S}_E = \{\bar{Z}_{a+1}, \ldots, \bar{Z}_{a+c}, \bar{X}_{a+1}, \ldots, \bar{X}_{a+c}\}$. The two subgroups $\mathcal{S}_I$ and $\mathcal{S}_E$ play a role in the error-correcting conditions for the entanglement-assisted stabilizer formalism. An entanglement-assisted code corrects errors in a set $\mathcal{E} \subset \Pi^n$ if

$$\forall E_1, E_2 \in \mathcal{E} \qquad E_1^\dagger E_2 \in \mathcal{S}_I \text{ or } E_1^\dagger E_2 \in \Pi^n - \mathcal{Z}\left(\langle \mathcal{S}_I, \mathcal{S}_E \rangle\right).$$

The conditions correspond to error pairs E_1, E_2 in an error set $\mathcal{E}$. The first condition corresponds to the passive error-correcting capability of the code, and the second condition corresponds to its active error-correcting capability.

Figure 3.1 illustrates the operation of an entanglement-assisted stabilizer code. The following four steps explain the operation of an $[n, k; c]$ entanglement-assisted quantum code:

(i). The sender and receiver share c ebits before quantum communication begins and the sender has a ancilla qubits. The unencoded state is a simultaneous $+1$-eigenstate of the following operators:

$$\{Z_{a+1}|Z_1, \ldots, Z_{a+c}|Z_c, X_{a+1}|X_1, \ldots, X_{a+c}|X_c, Z_1, \ldots, Z_a\}. \tag{3.3}$$

The operators to the right of the vertical bars indicate the receiver's half of the shared ebits. The sender encodes her k information qubits with the help of a ancilla qubits and her half of the c ebits. The encoding operation is U in Figure 3.1. The encoding unitary transforms the unencoded operators to the following encoded operators:

$$\{\bar{Z}_{a+1}|Z_1, \ldots, \bar{Z}_{a+c}|Z_c, \bar{X}_{a+1}|X_1, \ldots, \bar{X}_{a+c}|X_c, \bar{Z}_1, \ldots, \bar{Z}_a\}. \tag{3.4}$$

(ii). The sender sends her n qubits over a noisy quantum communication channel. The noisy channel affects these n qubits only and does not affect the receiver's half of the c ebits.

(iii). The receiver combines his half of the c ebits with those he receives from the noisy quantum channel. He performs measurements on all $n + c$ qubits to diagnose an error that may occur on the n qubits.

(iv). After estimating which error occurs, the receiver performs a recovery operation that reverses the estimated error.

3.2 The Symplectic Product Matrix

Let us write the local operators X^A and Z^A on the sender Alice's side as respective symplectic vectors h_1 and h_2:

$$h_1 = \left[\begin{array}{c|c} 0 & 1 \end{array}\right], \qquad h_2 = \left[\begin{array}{c|c} 1 & 0 \end{array}\right].$$

The "symplectic product matrix" Ω of these two symplectic vectors is as follows:

$$\Omega = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{3.5}$$

We call this matrix the symplectic product matrix because its entries enumerate the symplectic products:

$$[\Omega]_{ij} = h_i \odot h_j \quad \forall i, j \in \{1, 2\}. \tag{3.6}$$

The symplectic product matrix in (3.5) is so special for the purposes of this thesis that we give it the name $J \equiv \Omega$. It means that two generators have commutation relations that are equivalent to half of an ebit.

A general set of generators for a quantum block code can have complicated commutation relations. For a given set of generators, the commutation relations for the generators resulting from the symplectic Gram-Schmidt orthogonalization procedure have a special form. Their symplectic product matrix Ω has the standard form:

$$\Omega = \bigoplus_{i=1}^{c} J \oplus \bigoplus_{j=1}^{a} [0] \tag{3.7}$$

where the large and small $\oplus$ correspond to the direct sum operation and $[0]$ is the one-element null matrix. The standard form above implies that the commutation relations of the $2c + a$ generators are equivalent to those of c halves of ebits and a ancilla qubits—the commutation relations are equivalent to those in (3.2).

3.3 The Rate of an Entanglement-Assisted Quantum Code

We can interpret the rate of an entanglement-assisted code in three different ways [34, 35, 51]. Suppose that an entanglement-assisted quantum code encodes k information qubits into n physical qubits with the help of c ebits.

(i). The "entanglement-assisted" rate assumes that entanglement shared between sender and receiver is free. Bennett et al. make this assumption when deriving the entanglement-assisted capacity of a quantum channel for sending quantum information [40, 41]. The entanglement-assisted rate is k/n for a code with the above parameters.

(ii). The "trade-off' rate assumes that entanglement is not free and a rate pair determines performance. The first number in the pair is the number of noiseless qubits generated per channel use, and the second number in the pair is the number of ebits consumed per channel use. The rate pair is $(k/n, c/n)$ for a code with the above parameters. Quantum information theorists have computed asymptotic trade-off curves that bound the rate region in which achievable rate pairs lie [61]. Brun et al.'s construction for an entanglement-assisted quantum block code minimizes the number c of ebits given a fixed number k and n of respective information qubits and physical qubits [35, 34].

(iii). The "catalytic rate" assumes that bits of entanglement are built up at the expense of transmitted qubits [35, 34]. A noiseless quantum channel or the encoded use of noisy quantum channel are two different ways to build up entanglement between a sender and receiver. The catalytic rate of an $[n, k; c]$ code is $(k - c)/n$.

Which interpretation is most reasonable depends on the context in which we use the code. In any case, the parameters n, k, and c ultimately govern performance, regardless of which definition of the rate we use to interpret that performance.

3.4 Example of an Entanglement-Assisted Code

We present an example of an entanglement-assisted code that corrects an arbitrary single-qubit error [34]. This example highlights the main features of the theory given above. Suppose the sender wants to use the quantum error-correcting properties of the following nonabelian subgroup of Π^4:

$$\begin{matrix} Z & X & Z & I \\ Z & Z & I & Z \\ X & Y & X & I \\ X & X & I & X \end{matrix} \tag{3.8}$$

The first two generators anticommute. We obtain a modified third generator by multiplying the third generator by the second. We then multiply the last generator by the first, second, and modified third generators. The error-correcting properties of the generators are invariant under these operations. The modified generators are as follows:

$$\begin{array}{rcccc} g_1 & = & Z & X & Z & I \\ g_2 & = & Z & Z & I & Z \\ g_3 & = & Y & X & X & Z \\ g_4 & = & Z & Y & Y & X \end{array} \tag{3.9}$$

The above set of generators have the commutation relations given by the fundamental theorem of symplectic geometry:

$$\{g_1, g_2\} = [g_1, g_3] = [g_1, g_4] = [g_2, g_3] = [g_2, g_4] = [g_3, g_4] = 0.$$

The above set of generators is unitarily equivalent to the following canonical generators:

$$\begin{array}{cccc} X & I & I & I \\ Z & I & I & I \\ I & Z & I & I \\ I & I & Z & I \end{array} \tag{3.10}$$

We can add one ebit to resolve the anticommutativity of the first two generators:

$$\begin{array}{cccc|c} X & I & I & I & X \\ Z & I & I & I & Z \\ I & Z & I & I & I \\ I & I & Z & I & I \end{array} \tag{3.11}$$

A +1-eigenstate of the above stabilizer is the following state

$$\left|\Phi^+\right\rangle^{AB} |00\rangle^A |\psi\rangle^A, \tag{3.12}$$

where $|\psi\rangle^A$ is a qubit that the sender wants to encode. A local encoding unitary then rotates the generators in (3.11) to the following set of globally commuting generators:

$$\left[\begin{array}{cccc|c} Z & X & Z & I & X \\ Z & Z & I & Z & Z \\ Y & X & X & Z & I \\ Z & Y & Y & X & I \end{array}\right] \tag{3.13}$$

The receiver measures the above generators upon receipt of all qubits to detect and correct errors.

3.5 Encoding Algorithm

Suppose we have an arbitrary set of Pauli matrices in Π^n whose error-correcting properties we would like to exploit. The algorithm in this section (in Ref. [51]) determines the encoding and decoding circuit for the set of Pauli generators.

It has two additional benefits. We do not necessarily know beforehand how many ebits we require for the Pauli matrices to form a commuting set. Several methods exist such as the Gram-Schmidt algorithm outlined in Section 3.1 and the others in Refs. [35, 34, 30, 31, 51], but the algorithm here also determines the optimal number of ebits and the measurements the receiver performs to diagnose errors. It "kills three birds with one stone."

We continue with the example in Section 3.4. We detail an algorithm for determining an encoding circuit and the optimal number of ebits for the entanglement-assisted code. The operators in (3.8) have the following representation as a binary matrix according to the P2B isomorphism:

$$H = \left[\begin{array}{cccc|cccc} 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{array}\right]. \tag{3.14}$$

Call the matrix to the left of the vertical bar the "Z" matrix and the matrix to the right of the vertical bar the "X" matrix.

The algorithm consists of row and column operations on the above matrix. Row operations do not affect the error-correcting properties of the code but are crucial for arriving at the optimal decomposition from the fundamental theorem of symplectic geometry. The operations available for manipulating columns of the above matrix are Clifford operations (discussed at the end of Section 2.2). The operations have the following effect on entries in the binary matrix:

(i). A CNOT gate from qubit i to qubit j adds column i to column j in the X matrix and adds column j to column i in the Z matrix.

(ii). A Hadamard gate on qubit i swaps column i in the Z matrix with column i in the X matrix and vice versa.

(iii). A phase gate on qubit i adds column i in the X matrix to column i in the Z matrix.

(iv). Three CNOT gates implement a qubit swap operation [32]. The effect of a swap on qubits i and j is to swap columns i and j in both the X and Z matrix.

The algorithm begins by computing the symplectic product between the first row and all other rows. Leave the matrix as it is if the first row is not symplectically orthogonal to the second row or if the first row is symplectically orthogonal to all other rows. Otherwise, swap the second row with the first available row that is not symplectically orthogonal to the first row. In our example, the first row is not symplectically orthogonal to the second so we leave all rows as they are.

Arrange the first row so that the top left entry in the X matrix is one. A CNOT, swap, Hadamard, or combinations of these operations can achieve this result. We can have this result in our example by swapping qubits one and two. The matrix becomes

$$\left[\begin{array}{cccc|cccc} 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{array}\right]. \tag{3.15}$$

Perform CNOTs to clear the entries in the X matrix in the top row to the right of the leftmost entry. These entries are already zero in this example so we need not do anything. Proceed to the clear the entries in the first row of the Z matrix. Perform a phase gate to clear the leftmost entry

in the first row of the Z matrix if it is equal to one. It is equal to zero in our example so we need not do anything. We then use Hadamards and CNOTs to clear the other entries in the first row of the Z matrix.

For our example, perform a Hadamard on qubits two and three. The matrix becomes

$$\left[\begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right]. \tag{3.16}$$

Perform a CNOT from qubit one to qubit two and from qubit one to qubit three. The matrix becomes

$$\left[\begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right]. \tag{3.17}$$

The first row is complete. We now proceed to clear the entries in the second row. Perform a Hadamard on qubits one and four. The matrix becomes

$$\left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \end{array}\right]. \tag{3.18}$$

Perform a CNOT from qubit one to qubit two and from qubit one to qubit four. The matrix becomes

$$\left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{array}\right]. \tag{3.19}$$

The first two rows are now complete. They need one ebit to compensate for their anticommutativity or their nonorthogonality with respect to the symplectic product.

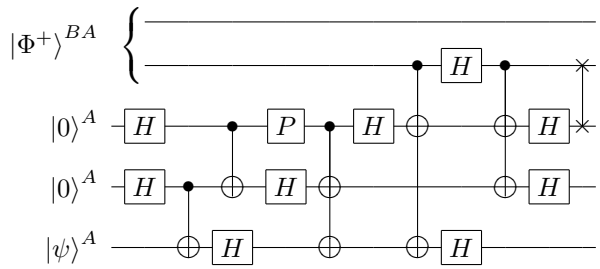

Figure 3.2: Encoding circuit for the entanglement-assisted code in the example of Section 3.4. The "H" gate is a Hadamard gate and the "P" gate is a phase gate.

Now we perform row operations that are similar to the "symplectic Gram-Schmidt orthogonalization." Add row one to any other row that has one as the leftmost entry in its Z matrix. Add row two to any other row that has one as the leftmost entry in its X matrix. For our example, we add row one to row four and we add row two to rows three and four. The matrix becomes

$$\left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{array}\right]. \tag{3.20}$$

The first two rows are now symplectically orthogonal to all other rows per the fundamental theorem of symplectic geometry.

We proceed with the same algorithm on the next two rows. We can deal with the next two rows individually because they are symplectically orthogonal to each other. Perform a Hadamard on qubit two. The matrix becomes

$$\left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{array}\right]. \tag{3.21}$$

Perform a CNOT from qubit two to qubit three and from qubit two to qubit four. The matrix becomes

$$\left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{array}\right]. \tag{3.22}$$

Perform a phase gate on qubit two:

$$\left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{array}\right]. \tag{3.23}$$

Perform a Hadamard on qubit three followed by a CNOT from qubit two to qubit three:

$$\left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{array}\right]. \tag{3.24}$$

Add row three to row four and perform a Hadamard on qubit two:

$$\left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{array}\right]. \tag{3.25}$$

Perform a Hadamard on qubit four followed by a CNOT from qubit three to qubit four. End by performing a Hadamard on qubit three:

$$\left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{array}\right] . \tag{3.26}$$

The above matrix now corresponds to the canonical Paulis in (3.10). Adding one half of an ebit to the receiver's side gives the canonical stabilizer in (3.11) whose simultaneous +1-eigenstate is the state in (3.12).

Figure 3.2 gives the encoding circuit corresponding to the above operations. The above operations in reverse order take the canonical stabilizer (3.11) to the encoded stabilizer (3.13).

3.6 Optimal Entanglement Formulas

Entanglement is a valuable resource, and we would like to minimize the amount that a sender and receiver need to consume for entanglement-assisted quantum coding. Hsieh, Devetak, and Brun first addressed this issue by determining a useful formula that gives the optimal number of ebits required by a Calderbank-Shor-Steane (CSS) entanglement-assisted code [31]. In particular, the number c of ebits that a CSS entanglement-assisted code requires is

$$c = \operatorname{rank}\left(HH^T\right), \tag{3.27}$$

where H corresponds to the parity check matrix of a classical binary block code that we import to correct quantum bit and phase flips. The same authors also determined an upper bound for the number of ebits that an entanglement-assisted quantum LDPC code requires [79].

In this section, we present several generalizations of the above formula. Our first theorem gives a formula for the optimal number of ebits that an arbitrary (non-CSS) entanglement-assisted quantum block code requires. We find special cases of this formula that apply to an entanglement-assisted quantum block code produced from two arbitrary classical binary block codes, and to

codes from a classical block code over $GF(4)$. Our last formula applies to *continuous-variable* entanglement-assisted codes [70].

Theorem 3.6.1. *Suppose we want to build an entanglement-assisted quantum code from generators corresponding to the rows in a quantum check matrix*

$$H = \left[\begin{array}{c|c} H_Z & H_X \end{array} \right], \tag{3.28}$$

where H is an $(n-k) \times 2n$-dimensional binary matrix representing the quantum code, and both H_Z and H_X are $(n-k) \times n$-dimensional binary matrices. Then the resulting code is an $[[n, k+c; c]]$ entanglement-assisted code and requires c ebits, where

$$c = \operatorname{rank}\left(H_X H_Z^T + H_Z H_X^T\right)/2, \tag{3.29}$$

and addition is binary.

Proof. Consider the symplectic product matrix $\Omega_H = H_X H_Z^T + H_Z H_X^T$. Its entries are the symplectic products between all rows of H so that

$$[\Omega_H]_{ij} = h_i \odot h_j, \tag{3.30}$$

where h_i is the i^{th} row of H. The matrix Ω_H is a $(n-k) \times (n-k)$-dimensional binary matrix.

Refs. [35, 34] and the algorithm in Section 3.1 outline a symplectic Gram-Schmidt orthogonalization procedure (SGSOP) that uniquely determines the optimal (i.e., minimal) number of ebits that the code requires, and Ref. [36] proves that the SGSOP gives the optimal number of ebits. The code construction in Refs. [35, 34] shows that the resulting entanglement-assisted quantum code requires at most c ebits. The essence of the argument in Ref. [36] is that the resulting entanglement-assisted quantum code requires at least c ebits because any fewer ebits would not be able to resolve the anticommutativity of the generators on Alice's side of the code.

The SGSOP performs row operations that do not change the error-correcting properties of the quantum code (because the code is additive), but these row operations do change the symplectic product relations. These row operations are either a row swap $S(i,j)$, where $S(i,j)$ is a full-rank $(n-k) \times (n-k)$ matrix that swaps row i with j, or a row addition $A(i,j)$, where $A(i,j)$ is a

full-rank $(n-k) \times (n-k)$ matrix that adds row i to row j. These row operations multiply the matrix H from the left. The SGSOP then is equivalent to a full-rank $(n-k) \times (n-k)$ matrix G that contains all of the row operations and produces a new quantum check matrix $H' = GH$ with corresponding symplectic product matrix $\Omega_{H'} = G\left(H_X H_Z^T + H_Z H_X^T\right) G^T$. In particular, the resulting symplectic product matrix $\Omega_{H'}$ is in the standard form in (3.7) so that

$$\Omega_{H'} = \bigoplus_{i=1}^{c} J \oplus \bigoplus_{j=1}^{n-k-2c} [0]\,. \tag{3.31}$$

Each matrix J in the direct sum corresponds to half of an ebit as described in Section 3.2 and has rank two. Each matrix $[0]$ has rank zero and corresponds to an ancilla qubit. The optimal number of ebits required for the code is $\operatorname{rank}\left(\Omega_{H'}\right)/2$:

$$\operatorname{rank}\left(\Omega_{H'}\right) = \operatorname{rank}\left(\bigoplus_{i=1}^{c} J \oplus \bigoplus_{j=1}^{n-k-2c} [0]\right) = \sum_{i=1}^{c} \operatorname{rank}\left(J\right) + \sum_{j=1}^{n-k-2c} \operatorname{rank}\left([0]\right) = 2c. \tag{3.32}$$

The second equality follows because the rank of a direct sum is the sum of the individual matrix ranks, and the third equality follows from the individual matrix ranks given above. The number c of ebits is also equal to $\operatorname{rank}\left(\Omega_H\right)/2$ because the matrix G is full rank. The code is an $[[n, k+c; c]]$ entanglement-assisted quantum block code by the construction in Refs. [35, 34].

□

Our formula (3.29) is equivalent to the formula at the top of page 14 in Ref. [35], but it provides the quantum code designer with a quick method to determine how many ebits an entanglement-assisted code requires, by simply "plugging in" the generators of the code.

The formula (3.29), like the CSS formula in (3.27), is a measure of how far a set of generators is from being a commuting set, or equivalently, how far it is from giving a standard stabilizer code.

Corollary 3.6.1 below gives a formula for the optimal number of ebits required by a CSS entanglement-assisted quantum code. It is generally a bit less difficult to compute than the above formula in (3.29). This reduction in complexity occurs because of the special form of a CSS quantum code and because the size of the matrices involved are generally smaller for a CSS code than for a general code with the same number of generators and physical qubits.

Corollary 3.6.1. *Suppose we import two classical* $[n, k_1, d_1]$ *and* $[n, k_2, d_2]$ *binary codes with respective parity check matrices* H_1 *and* H_2 *to build an entanglement-assisted quantum code. The resulting code is an* $[[n, k_1 + k_2 - n + c, \min(d_1, d_2); c]]$ *entanglement-assisted code, and requires* c *ebits where*

$$c = \operatorname{rank}\left(H_1 H_2^T\right). \tag{3.33}$$

Proof. The quantum check matrix has the following form:

$$H = \left[\begin{array}{c|c} H_1 & 0 \\ 0 & H_2 \end{array}\right]. \tag{3.34}$$

The symplectic product matrix Ω_H is then

$$\Omega_H = \begin{bmatrix} H_1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 & H_2^T \end{bmatrix} + \begin{bmatrix} 0 \\ H_2 \end{bmatrix} \begin{bmatrix} H_1^T & 0 \end{bmatrix} = \begin{bmatrix} 0 & H_1 H_2^T \\ H_2 H_1^T & 0 \end{bmatrix}. \tag{3.35}$$

The above matrix is equivalent by a full rank permutation matrix to the matrix $H_1 H_2^T \oplus H_2 H_1^T$, so the rank of Ω_H is

$$\operatorname{rank}(\Omega_H) = \operatorname{rank}\left(H_1 H_2^T \oplus H_2 H_1^T\right) = \operatorname{rank}\left(H_1 H_2^T\right) + \operatorname{rank}\left(H_2 H_1^T\right) = 2 \operatorname{rank}\left(H_1 H_2^T\right) \tag{3.36}$$

The second equality follows because the rank of a direct sum is equivalent to the sum of the individual ranks, and the third equality follows because the rank is invariant under matrix transposition. The number of ebits required for the resulting entanglement-assisted quantum code is $\operatorname{rank}\left(H_1 H_2^T\right)$, using the result of the previous theorem. The construction in Refs. [35, 34] produces an $[[n, k_1 + k_2 - n + c, \min(d_1, d_2); c]]$ entanglement-assisted quantum block code.

□

Corollary 3.6.2. *Suppose we import an* $[n, k, d]_4$ *classical code over* $GF(4)$ *with parity check matrix* H *for use as an entanglement-assisted quantum code according to the construction in Refs. [35, 34]. Then the resulting quantum code is an* $[[n, 2k - n + c; c]]$ *entanglement-assisted quantum code where* $c = rank\left(HH^\dagger\right)$ *and* $\dagger$ *denotes the conjugate transpose operation over matrices in* $GF(4)$.

Proof. Ref. [34] shows how to produce an entanglement-assisted quantum code from the parity check matrix H of a classical code over $GF(4)$. The resulting quantum parity check matrix H_Q in symplectic binary form is

$$H_Q = \gamma\left(\begin{bmatrix}\omega H \\ \bar{\omega} H\end{bmatrix}\right), \tag{3.37}$$

where γ denotes the isomorphism between elements of $GF(4)$ and symplectic binary vectors that represent Pauli matrices. We augment the table in (2.12) to include the entries from $GF(4)$:

Π	I	X	Y	Z
$(\mathbb{Z}_2)^2$	00	01	11	10
$GF(4)$	0	ω	1	$\bar{\omega}$

(3.38)

The isomorphism γ is the mapping from the third row to the second row in the above table. The symplectic product between binary vectors is equivalent to the trace product of their $GF(4)$ representations (see, e.g., Ref. [34]):

$$h_i \odot h_j = \operatorname{tr}\left\{\gamma^{-1}(h_i) \cdot \overline{\gamma^{-1}(h_j)}\right\}, \tag{3.39}$$

where h_i and h_j are any two rows of H_Q, $\cdot$ denotes the inner product, the overbar denotes the conjugate operation, and $\operatorname{tr}\{x\} = x + \bar{x}$ denotes the trace operation over elements of $GF(4)$. We exploit these correspondences to write the symplectic product matrix Ω_{H_Q} for the quantum check matrix H_Q as follows:

$$\Omega_{H_Q} = \operatorname{tr}\left\{\begin{bmatrix}\omega H \\ \bar{\omega} H\end{bmatrix}\begin{bmatrix}\omega H \\ \bar{\omega} H\end{bmatrix}^\dagger\right\} = \operatorname{tr}\left\{\begin{bmatrix}\omega H \\ \bar{\omega} H\end{bmatrix}\begin{bmatrix}\bar{\omega} H^\dagger & \omega H^\dagger\end{bmatrix}\right\} \tag{3.40}$$

$$= \operatorname{tr}\left\{\begin{bmatrix}HH^\dagger & \bar{\omega} HH^\dagger \\ \omega HH^\dagger & HH^\dagger\end{bmatrix}\right\} = \operatorname{tr}\left\{\begin{bmatrix}1 & \bar{\omega} \\ \omega & 1\end{bmatrix} \otimes HH^\dagger\right\} \tag{3.41}$$

$$= \begin{bmatrix}1 & \bar{\omega} \\ \omega & 1\end{bmatrix} \otimes HH^\dagger + \begin{bmatrix}1 & \omega \\ \bar{\omega} & 1\end{bmatrix} \otimes \bar{H}H^T \tag{3.42}$$

where the "tr" operation above is an element-wise trace operation over $GF(4)$ (it is not the matrix trace operation.) The matrix Ω_{H_Q} is over the field $GF(2)$, but we can consider it as being over the field $GF(4)$ without changing its rank. Therefore, we can multiply it by matrices over the field $GF(4)$. Consider the following full-rank $GF(4)$ matrices:

$$A_1 = \begin{bmatrix} 1 & \bar{\omega} \\ 0 & 1 \end{bmatrix} \otimes I, \quad A_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \otimes I. \tag{3.43}$$

We premultiply and postmultiply the matrix Ω_{H_Q} as follows and obtain a matrix with the same rank as Ω_{H_Q}:

$$A_2 A_1 \Omega_{H_Q} A_1^\dagger A_2^\dagger = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \otimes HH^\dagger + \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \otimes \bar{H}H^T = \begin{bmatrix} \bar{H}H^T & 0 \\ 0 & HH^\dagger \end{bmatrix} = \bar{H}H^T \oplus HH^\dagger \tag{3.44}$$

Therefore, the rank of Ω_{H_Q} is

$$\operatorname{rank}\left(\Omega_{H_Q}\right) = \operatorname{rank}\left(\bar{H}H^T \oplus HH^\dagger\right) = \operatorname{rank}\left(\bar{H}H^T\right) + \operatorname{rank}\left(HH^\dagger\right) = 2 \operatorname{rank}\left(HH^\dagger\right). \tag{3.45}$$

The second equality holds because the rank of a direct sum is the sum of the individual ranks and the third holds because the rank is invariant under the matrix transpose operation. Therefore, the resulting entanglement-assisted quantum code requires $c = \operatorname{rank}\left(HH^\dagger\right)$ ebits by applying the result of the original theorem. The construction in Refs. [35, 34] produces an $[[n, 2k - n + c; c]]$ entanglement-assisted quantum code.

□

Corollary 3.6.3. *We can construct a continuous-variable entanglement-assisted quantum code from generators corresponding to the rows in quantum check matrix* $H = \left[\begin{array}{c|c} H_Z & H_X \end{array}\right]$ *where* H *is* $(n-k) \times 2n$*-dimensional,* H *is a real matrix representing the quantum code [70], and both* H_Z *and* H_X *are* $(n-k) \times n$*-dimensional. The resulting code is an* $[[n, k+c; c]]$ *continuous-variable entanglement-assisted code and requires* c *entangled modes where*

$$c = \operatorname{rank}\left(H_X H_Z^T - H_Z H_X^T\right)/2. \tag{3.46}$$

Proof. The proof is similar to the proof of the first theorem but requires manipulations of real vectors instead of binary vectors. See Ref. [70] for details of the symplectic geometry required for continuous-variable entanglement-assisted codes. □

Remark 3.6.1. A similar formula holds for entanglement-assisted qudit codes by replacing the subtraction operation above with subtraction modulo d. Specifically, we can construct a qudit entanglement-assisted quantum code from generators corresponding to the rows in check matrix $H = \left[\begin{array}{c|c} H_Z & H_X \end{array}\right]$ whose matrix entries are elements of the finite field $\mathbb{Z}_d$. The code requires c edits (a d-dimensional state $\left(\sum_{i=0}^{d-1} |i\rangle\,|i\rangle\right)/\sqrt{d}$) where

$$c = \operatorname{rank}\left(H_X H_Z^T \ominus_d H_Z H_X^T\right)/2$$

and $\ominus_d$ is subtraction modulo d. We use subtraction modulo d because the symplectic form over d-dimensional variables includes subtraction modulo d.

3.7 Closing Remarks

This chapter reviewed the theory of entanglement-assisted coding for block codes. We showed how to manipulate these codes with both a Pauli representation and a binary one. Our contributions included an algorithm for computing an encoding circuit and a method to determine the optimal number of ebits for an entanglement-assisted code. Much of this thesis extends the entanglement-assisted technique to the domain of quantum convolutional coding. We introduce quantum convolutional coding in the next chapter.

Chapter 4

Quantum Convolutional Codes

Forney, Viterbi, and Elias,
To Andy the Trojans are pious,
Their methods, we want 'em,
Who'd think they'd go quantum?
So now they still stupefy us.

The block codes in the previous two chapters are useful in quantum computing and in quantum communication. One of the drawbacks of the block-coding technique for quantum communication is that, in general, the sender must have all her qubits ready before encoding takes place. For a large block code, this preparation may be a heavy demand on the sender.

Quantum convolutional coding theory [43, 44, 48, 1] offers a different paradigm for coding quantum information. The convolutional structure is useful in a quantum communication scenario where a sender possesses a stream of qubits to send to a receiver. The encoding circuit for a quantum convolutional code has a much lower complexity than an encoding circuit needed for a large block code. It also has a repetitive pattern so that the same physical devices or the same routines can encode the stream of quantum information in an online fashion as soon as information qubits are available for encoding.

Quantum convolutional stabilizer codes borrow heavily from the structure of their classical counterparts [43, 44, 48, 1]. Quantum convolutional codes are similar because some of the qubits feed back into a repeated encoding unitary and give the code a memory structure like that of a

classical convolutional code. The quantum codes feature online encoding and decoding of qubits. This feature gives quantum convolutional codes both their low encoding and decoding complexity.

Our techniques for encoding and decoding are an expansion of previous techniques from quantum convolutional coding theory. Previous techniques for encoding and decoding include finite-depth operations only. A finite-depth operation propagates errors to a finite number of neighboring qubits in the qubit stream. We introduce an infinite-depth operation to the set of shift-invariant Clifford operations and explain it in detail in Section 4.6. We must be delicate when using infinite-depth operations because they can propagate errors to an infinite number of neighboring qubits in the qubit stream. We explain our assumptions in detail when we include infinite-depth operations in entanglement-assisted quantum convolutional codes in Section 5.4 of the next chapter. An infinite-depth operation gives more flexibility when designing encoding circuits—similar to the way in which an infinite-impulse response filter gives more flexibility in the design of classical convolutional circuits.

We structure this chapter as follows. In Section 4.1, we introduce the definition of a quantum convolutional code and discuss how it operates. Section 4.2 introduces the Pauli-to-binary isomorphism. This mapping simplifies the mathematics of a quantum convolutional code by showing how to represent it with a matrix of binary polynomials. Section 4.3 discusses the important "shifted symplectic product" that gives the commutation relations of a set of convolutional generators. We discuss in Section 4.4 how to manipulate the binary polynomial matrix that represents a quantum convolutional code. In Sections 4.5 and 4.6, we respectively review finite-depth and infinite-depth operations for encoding a quantum convolutional code.

4.1 Review of the Convolutional Stabilizer Formalism

We review the theory of convolutional stabilizer codes by considering a set of Pauli matrices that stabilize a stream of encoded qubits. We first give the mathematical definition of a quantum convolutional code and follow by discussing the various steps involved in the operation of it.

4.1.1 Quantum Convolutional Code Definition

A quantum convolutional stabilizer code acts on a Hilbert space $\mathcal{H}$ that is a countably infinite tensor product of two-dimensional qubit Hilbert spaces $\{\mathcal{H}_i\}_{i\in\mathbb{Z}^+}$ where

$$\mathcal{H} = \bigotimes_{i=0}^{\infty} \mathcal{H}_i. \tag{4.1}$$

and $\mathbb{Z}^+ \equiv \{0, 1, \ldots\}$. A sequence $\mathbf{A}$ of Pauli matrices $\{A_i\}_{i\in\mathbb{Z}^+}$, where

$$\mathbf{A} = \bigotimes_{i=0}^{\infty} A_i, \tag{4.2}$$

can act on states in $\mathcal{H}$. Let $\Pi^{\mathbb{Z}^+}$ denote the set of all Pauli sequences. The support $\text{supp}(\mathbf{A})$ of a Pauli sequence $\mathbf{A}$ is the set of indices of the entries in $\mathbf{A}$ that are not equal to the identity. The weight of a sequence $\mathbf{A}$ is the size $|\text{supp}\,(\mathbf{A})|$ of its support. The delay $\text{del}(\mathbf{A})$ of a sequence $\mathbf{A}$ is the smallest index for an entry not equal to the identity. The degree $\deg(\mathbf{A})$ of a sequence $\mathbf{A}$ is the largest index for an entry not equal to the identity. E.g., the following Pauli sequence

$$I \quad X \quad I \quad Y \quad Z \quad I \quad I \quad \cdots, \tag{4.3}$$

has support $\{1, 3, 4\}$, weight three, delay one, and degree four. A sequence has finite support if its weight is finite. Let $F(\Pi^{\mathbb{Z}^+})$ denote the set of Pauli sequences with finite support. The following definition for a quantum convolutional code utilizes the set $F(\Pi^{\mathbb{Z}^+})$ in its description.

Definition 4.1.1. *A rate k/n-convolutional stabilizer code with $0 \leq k \leq n$ is specified by a commuting set $\mathcal{G}$ of all n-qubit shifts of a basic generator set $\mathcal{G}_0$. The commutativity requirement is necessary for the same reason that standard stabilizer codes require it [32]. The basic generator set $\mathcal{G}_0$ has $n-k$ Pauli sequences of finite support:*

$$\mathcal{G}_0 = \left\{ \mathbf{G}_i \in F(\Pi^{\mathbb{Z}^+}) : 1 \leq i \leq n-k \right\}. \tag{4.4}$$

The constraint length ν of the code is the maximum degree of the generators in $\mathcal{G}_0$. A frame of the code consists of n qubits. The definition of a quantum convolutional code as n-qubit shifts of the basic set $\mathcal{G}_0$ is what gives the code its periodic structure.

A quantum convolutional code admits an equivalent definition in terms of the delay transform or D-transform. The D-transform captures shifts of the basic generator set $\mathcal{G}_0$. Let us define the n-qubit delay operator D acting on any Pauli sequence $\mathbf{A} \in \Pi^{\mathbb{Z}^+}$ as follows:

$$D(\mathbf{A}) = I^{\otimes n} \otimes \mathbf{A}. \tag{4.5}$$

We can write j repeated applications of D as a power of D:

$$D^j(\mathbf{A}) = I^{\otimes jn} \otimes \mathbf{A}. \tag{4.6}$$

Let $D^j(\mathcal{G}_0)$ be the set of shifts of elements of $\mathcal{G}_0$ by j. Then the full stabilizer $\mathcal{G}$ for the convolutional stabilizer code is

$$\mathcal{G} = \bigcup_{j \in \mathbb{Z}^+} D^j(\mathcal{G}_0). \tag{4.7}$$

Example 4.1.1. Forney et al. provided an example of a rate-1/3 quantum convolutional code by importing a particular classical quaternary convolutional code [48, 1]. Grassl and Rötteler determined a noncatastrophic encoding circuit for Forney et al.'s rate-1/3 quantum convolutional code [45]. The basic stabilizer and its first shift are as follows:

$$\cdots \left| \begin{array}{c|c|c|c|c} III & XXX & XZY & III & III \\ III & ZZZ & ZYX & III & III \\ III & III & XXX & XZY & III \\ III & III & ZZZ & ZYX & III \end{array} \right| \cdots \tag{4.8}$$

The code consists of all three-qubit shifts of the above generators. The vertical bars are a visual aid to illustrate the three-qubit shifts of the basic generators. The code can correct for an arbitrary single-qubit error in every other frame.

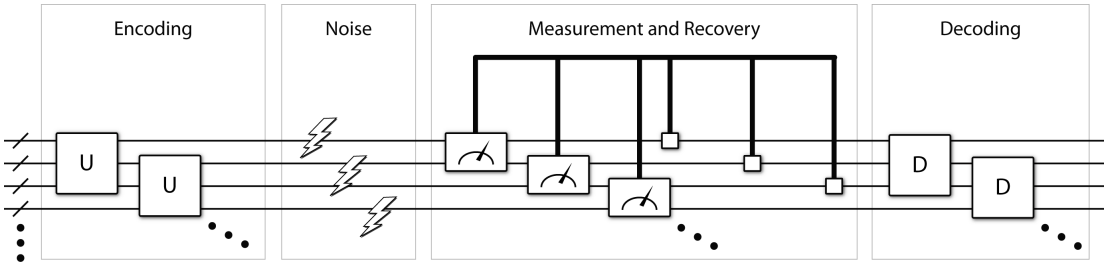

Figure 4.1: The operation of a quantum convolutional code. The sender applies the same unitary successively to a stream of information qubits and ancilla qubits. The convolutional structure implies that the unitary overlaps some of the same qubits. The sender transmits her qubits as soon as the unitary finishes processing them. The noisy quantum channel corrupts the transmitted qubits. The receiver performs overlapping multi-qubit measurements to diagnose channel errors and corrects for them. The receiver performs an online decoding circuit to recover the sender's original stream of information qubits.

4.1.2 Quantum Convolutional Code Operation

Figure 4.1 illustrates the basic operation of a quantum convolutional code. The operation of a rate-k/n quantum convolutional code consists of several steps:

(i). The protocol begins with the sender encoding a stream of information qubits with an online encoding circuit. The sender encodes $n-k$ ancilla qubits and k information qubits per frame [47, 45]. The encoding circuit is "online" if it acts on a few frames at a time.

(ii). The sender transmits a set of initial qubits as soon as the first unitary finishes processing them and continues to transmit later qubits after the next part of the online encoding circuit finishes processing them.

(iii). The above basic set $\mathcal{G}_0$ and all of its n-qubit shifts act like a parity check matrix for the quantum convolutional code. The receiver measures the generators in the stabilizer to determine an error syndrome. It is important that the generators in $\mathcal{G}_0$ have finite weight so that the receiver can perform the measurements and produce an error syndrome. It is also important that the generators have a block-band form so that the receiver can perform the measurements online as the noisy encoded qubits arrive.

(iv). The receiver processes the error syndromes with an online classical error estimation algorithm such as the Viterbi algorithm [2] or any other decoding algorithm [80] to determine the

most likely error for each frame of quantum data. Syndrome-based versions of the Viterbi algorithm that are appropriate for quantum coding are available in Refs. [43, 44, 81].

(v). The receiver performs unitary recovery operations that reverse the estimated errors.

(vi). He finally processes the encoded qubits with a decoding circuit to recover the original stream of information qubits. The qubits decoded from this convolutional procedure should be error free and ready for quantum computation at the receiving end.

In the above description, each step is online—the sender and receiver do not have to employ the above procedure sequentially. Processing of one step can begin once the previous step has finished some amount of processing. All steps above are always active during processing once the initial rounds have been completed.

4.2 The Pauli-to-Binary Isomorphism for Quantum Convolutional Codes

This section contains the most important part of this quantum convolutional coding review—the isomorphism from the set of Pauli sequences to the module over the ring of binary polynomials [44, 45, 1]. We also name it the Pauli-to-binary (P2B) isomorphism (whether we are considering the P2B isomorphism for block codes or convolutional codes should be clear from the context). The P2B isomorphism is important because it is easier to perform manipulations with vectors of binary polynomials than with Pauli sequences.

We first define the phase-free Pauli group $\left[\Pi^{\mathbb{Z}}\right]$ on a sequence of qubits. Recall that the delay transform D in (4.5) shifts a Pauli sequence to the right by n. Let us assume for now that $n = 1$. Let $\Pi^{\mathbb{Z}}$ denote the set of all countably infinite Pauli sequences. The set $\Pi^{\mathbb{Z}}$ is equivalent to the set of all one-qubit shifts of arbitrary Pauli operators:

$$\Pi^{\mathbb{Z}} = \left\{ \prod_{i \in \mathbb{Z}} D^i \left(A_i \right) : A_i \in \Pi \right\}. \tag{4.9}$$

We remark that $D^i\left(A_i\right) = D^i\left(A_i \otimes I^{\otimes\infty}\right)$. We make this same abuse of notation in what follows. We can define the equivalence class $\left[\Pi^{\mathbb{Z}}\right]$ of phase-free Pauli sequences:

$$\left[\Pi^{\mathbb{Z}}\right] = \left\{\beta\mathbf{A} \mid \mathbf{A} \in \Pi^{\mathbb{Z}}, \beta \in \mathbb{C}, |\beta| = 1\right\}. \tag{4.10}$$

We develop the Pauli-to-binary (P2B) isomorphism between binary polynomials and Pauli sequences. The P2B isomorphism is useful for representing the shifting nature of quantum convolutional codes. Suppose $z\left(D\right)$ and $x\left(D\right)$ are arbitrary finite-degree and finite-delay polynomials in D over $\mathbb{Z}_2$:

$$z\left(D\right) = \sum_i z_i D^i, \qquad x\left(D\right) = \sum_i x_i D^i, \qquad z_i, x_i \in \mathbb{Z}_2 \;\; \forall i \in \mathbb{Z}, \tag{4.11}$$

where $\mathrm{del}(z\left(D\right))$, $\mathrm{del}(x\left(D\right))$, $\deg(z\left(D\right))$, $\deg(x\left(D\right)) < \infty$. Suppose

$$u\left(D\right) = \left(z\left(D\right), x\left(D\right)\right) \in \left(\mathbb{Z}_2\left(D\right)\right)^2, \tag{4.12}$$

where $\left(\mathbb{Z}_2\left(D\right)\right)^2$ indicates the Cartesian product $\mathbb{Z}_2\left(D\right) \times \mathbb{Z}_2\left(D\right)$. Let us employ the following shorthand:

$$u\left(D\right) = \left[z\left(D\right)|x\left(D\right)\right]. \tag{4.13}$$

Let N be a map from the binary polynomials to the Pauli sequences, $N : \left(\mathbb{Z}_2\left(D\right)\right)^2 \to \Pi^{\mathbb{Z}}$, where

$$N\left(u\left(D\right)\right) = \prod_i D^i\left(Z^{z_i}X^{x_i}\right). \tag{4.14}$$

Let $v\left(D\right) = \left[z'\left(D\right)|x'\left(D\right)\right]$ where $v\left(D\right) \in \left(\mathbb{Z}_2\left(D\right)\right)^2$. The map N induces an isomorphism

$$\left[N\right] : \left(\mathbb{Z}_2\left(D\right)\right)^2 \to \left[\Pi^{\mathbb{Z}}\right], \tag{4.15}$$

because addition of binary polynomials is equivalent to multiplication of Pauli elements up to a global phase:

$$\left[N\left(u\left(D\right) + v\left(D\right)\right)\right] = \left[N\left(u\left(D\right)\right)\right]\left[N\left(v\left(D\right)\right)\right]. \tag{4.16}$$

The above isomorphism is a powerful way to capture the infiniteness and shifting nature of convolutional codes with finite-degree and finite-delay polynomials over the binary field $\mathbb{Z}_2$.

A quantum convolutional code in general consists of generators with n qubits per frame. Therefore, we consider the n-qubit extension of the definitions and isomorphism given above. Let the delay transform D now shift a Pauli sequence to the right by an arbitrary positive integer n. Consider a $2n$-dimensional vector $\mathbf{u}(D)$ of binary polynomials where $\mathbf{u}(D) \in (\mathbb{Z}_2(D))^{2n}$. Let us write $\mathbf{u}(D)$ as follows

$$\mathbf{u}(D) = [\mathbf{z}(D) | \mathbf{x}(D)] = \left[\begin{array}{ccc|ccc} z_1(D) & \cdots & z_n(D) & x_1(D) & \cdots & x_n(D) \end{array} \right],$$

where $\mathbf{z}(D), \mathbf{x}(D) \in (\mathbb{Z}_2(D))^n$. Suppose

$$z_i(D) = \sum_j z_{i,j} D^j, \qquad x_i(D) = \sum_j x_{i,j} D^j.$$

Define a map $\mathbf{N} : (\mathbb{Z}_2(D))^{2n} \to \Pi^{\mathbb{Z}}$:

$$\mathbf{N}(\mathbf{u}(D)) = \prod_j D^j \left(Z^{z_{1,j}} X^{x_{1,j}}\right) D^j \left(I \otimes Z^{z_{2,j}} X^{x_{2,j}}\right) \cdots D^j \left(I^{\otimes n-1} \otimes Z^{z_{n,j}} X^{x_{n,j}}\right).$$

$\mathbf{N}$ is equivalent to the following map (up to a global phase)

$$\mathbf{N}(\mathbf{u}(D)) = N(u_1(D)) \left(I \otimes N(u_2(D))\right) \cdots \left(I^{\otimes n-1} \otimes N(u_n(D))\right),$$

where

$$u_i(D) = [z_i(D) | x_i(D)]. \tag{4.17}$$

Suppose

$$\mathbf{v}(D) = [\mathbf{z}'(D) | \mathbf{x}'(D)], \tag{4.18}$$

where $\mathbf{v}(D) \in (\mathbb{Z}_2(D))^{2n}$. The map $\mathbf{N}$ induces an isomorphism $[\mathbf{N}] : (\mathbb{Z}_2(D))^{2n} \to \left[\Pi^{\mathbb{Z}}\right]$ for the same reasons given in (4.16):

$$[\mathbf{N}(\mathbf{u}(D) + \mathbf{v}(D))] = [\mathbf{N}(\mathbf{u}(D))] [\mathbf{N}(\mathbf{v}(D))]. \tag{4.19}$$

The P2B isomorphism [**N**] is again useful because it allows us to perform binary calculations instead of Pauli calculations.

4.3 The Shifted Symplectic Product

We consider the commutative properties of quantum convolutional codes in this section. We develop some mathematics for the important "shifted symplectic product." The shifted symplectic product reveals the commutation relations of an arbitrary number of shifts of a set of Pauli sequences.

Recall from Definition 4.1.1 that a commuting set comprising a basic set of Paulis and all their shifts specifies a quantum convolutional code. How can we capture the commutation relations of a Pauli sequence and all of its shifts? The *shifted* symplectic product $\odot$, where

$$\odot : \left(\mathbb{Z}_2\left(D\right)\right)^2 \times \left(\mathbb{Z}_2\left(D\right)\right)^2 \rightarrow \mathbb{Z}_2\left(D\right), \tag{4.20}$$

is an elegant way to do so. The shifted symplectic product maps two vectors $u\left(D\right) = \left[z\left(D\right)|x\left(D\right)\right]$ and $v\left(D\right) = \left[z'\left(D\right)|x'\left(D\right)\right]$ to a binary polynomial with finite delay and finite degree:

$$\left(u \odot v\right)\left(D\right) = z\left(D\right)x'\left(D^{-1}\right) - x\left(D\right)z'\left(D^{-1}\right). \tag{4.21}$$

The shifted symplectic product is not a proper symplectic product because it fails to be alternating [77]. The alternating property requires that

$$\left(u \odot v\right)\left(D\right) = -\left(v \odot u\right)\left(D\right), \tag{4.22}$$

but we find instead that the following holds:

$$\left(u \odot v\right)\left(D\right) = -\left(v \odot u\right)\left(D^{-1}\right). \tag{4.23}$$

Every vector $u\left(D\right) \in \mathbb{Z}_2\left(D\right)^2$ is self time-reversal antisymmetric with respect to $\odot$:

$$\left(u \odot u\right)\left(D\right) = -\left(u \odot u\right)\left(D^{-1}\right) \quad \forall u\left(D\right) \in \mathbb{Z}_2\left(D\right)^2. \tag{4.24}$$

Every binary vector is also self time-reversal symmetric with respect to $\odot$ because addition and subtraction are the same over $\mathbb{Z}_2$. We employ the addition convention from now on and drop the minus signs. The shifted symplectic product is a binary polynomial in D. We write its coefficients as follows:

$$(u \odot v)(D) = \sum_{i \in \mathbb{Z}} (u \odot v)_i \ D^i. \tag{4.25}$$

The coefficient $(u \odot v)_i$ captures the commutation relations of two Pauli sequences for i one-qubit shifts of one of the sequences:

$$N(u(D))\, D^i\, (N(v(D))) = (-1)^{(u \odot v)_i}\, D^i\, (N(v(D)))\, N(u(D)). \tag{4.26}$$

Thus two Pauli sequences $N(u(D))$ and $N(v(D))$ commute for all shifts if and only if the shifted symplectic product $(u \odot v)(D)$ vanishes.

The next example highlights the main features of the shifted symplectic product and further emphasizes the relationship between Pauli commutation and orthogonality of the shifted symplectic product.

Example 4.3.1. Consider two vectors of binary polynomials:

$$u(D) = \left[\ D \ \mid \ 1 + D^3 \ \right], \qquad v(D) = \left[\ 1 + D \ \mid \ D^3 \ \right].$$

The P2B isomorphism maps the above polynomials to the following Pauli sequences:

$$N(u(D)) = (\cdots |I|X|Z|I|X|I| \cdots), \qquad N(v(D)) = (\cdots |I|Z|Z|I|X|I| \cdots). \tag{4.27}$$

The vertical bars between every Pauli in the sequence indicate that we are considering one-qubit shifts. We determine the commutation relations of the above sequences by inspection. $N(u(D))$ anticommutes with a shift of itself by one or two to the left or right and commutes with all other shifts of itself. $N(v(D))$ anticommutes with a shift of itself by two or three to the left or right and commutes with all other shifts of itself. $N(u(D))$ anticommutes with $N(v(D))$ shifted to

the left by one or two, with the zero-shifted $N(v(D))$, and with $N(v(D))$ shifted to the right by two or three. The following shifted symplectic products give us the same information:

$$\begin{aligned}
&(u \odot u)(D) = D^{-2} + D^{-1} + D + D^2, \qquad (v \odot v)(D) = D^{-3} + D^{-2} + D^2 + D^3, \\
&(v \odot u)(D) = D^{-3} + D^{-2} + 1 + D + D^2.
\end{aligned} \tag{4.28}$$

The nonzero coefficients indicate the commutation relations just as (4.26) claims.

We can again define a shifted symplectic product for the case of n-qubits per frame. Let $\odot$ denote the shifted symplectic product between vectors of binary polynomials:

$$\odot : (\mathbb{Z}_2(D))^{2n} \times (\mathbb{Z}_2(D))^{2n} \to \mathbb{Z}_2(D). \tag{4.29}$$

It maps $2n$-dimensional vectors $\mathbf{u}(D)$ and $\mathbf{v}(D)$ of binary polynomials to a finite-degree and finite-delay binary polynomial

$$(\mathbf{u} \odot \mathbf{v})(D) = \sum_{i=1}^{n} (u_i \odot v_i)(D), \tag{4.30}$$

where

$$u_i(D) = [z_i(D) | x_i(D)], \qquad v_i(D) = [z_i'(D) | x_i'(D)].$$

The standard inner product gives an alternative way to define the shifted symplectic product:

$$(\mathbf{u} \odot \mathbf{v})(D) = \mathbf{z}(D) \cdot \mathbf{x}'\left(D^{-1}\right) - \mathbf{x}(D) \cdot \mathbf{z}'\left(D^{-1}\right). \tag{4.31}$$

Every vector $\mathbf{u}(D) \in \mathbb{Z}_2(D)^{2n}$ is self time-reversal symmetric with respect to $\odot$:

$$(\mathbf{u} \odot \mathbf{u})(D) = (\mathbf{u} \odot \mathbf{u})\left(D^{-1}\right) \qquad \forall \mathbf{u}(D) \in \mathbb{Z}_2(D)^{2n}. \tag{4.32}$$

The shifted symplectic product for vectors of binary polynomials is a binary polynomial in D. We write its coefficients as follows:

$$(\mathbf{u} \odot \mathbf{v})(D) = \sum_{i \in \mathbb{Z}} (\mathbf{u} \odot \mathbf{v})_i \ D^i. \tag{4.33}$$

The coefficient $(\mathbf{u} \odot \mathbf{v})_i$ captures the commutation relations of two Pauli sequences for i n-qubit shifts of one of the sequences:

$$\mathbf{N}(\mathbf{u}(D)) D^i (\mathbf{N}(\mathbf{v}(D))) = (-1)^{(\mathbf{u} \odot \mathbf{v})_i} D^i (\mathbf{N}(\mathbf{v}(D))) \mathbf{N}(\mathbf{u}(D)).$$

The shifted symplectic product between two vectors of binary polynomials vanishes if and only if their corresponding Pauli sequences commute.

Example 4.3.2. We consider the case where the frame size $n = 4$. Consider the following vectors of polynomials:

$$\left[\begin{array}{c} \mathbf{u}(D) \\ \mathbf{v}(D) \end{array}\right] = \left[\begin{array}{cccc|cccc} 1+D & D & 1 & D & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1+D & 1+D & 1 & D \end{array}\right]. \tag{4.34}$$

The P2B isomorphism maps $\mathbf{u}(D)$ and $\mathbf{v}(D)$ to the following Pauli sequences:

$$\begin{aligned} \mathbf{N}(\mathbf{u}(D)) &= (\cdots | IIII | ZXZI | ZZIZ | IIII | \cdots), \\ \mathbf{N}(\mathbf{v}(D)) &= (\cdots | IIII | XYXI | XXIX | IIII | \cdots). \end{aligned} \tag{4.35}$$

We can determine the commutation relations by inspection of the above Pauli sequences. $\mathbf{N}(\mathbf{u}(D))$ anticommutes with itself shifted by one to the left or right, $\mathbf{N}(\mathbf{v}(D))$ anticommutes with itself shifted by one to the left or right, and $\mathbf{N}(\mathbf{u}(D))$ anticommutes with $\mathbf{N}(\mathbf{v}(D))$ shifted by one to the left. The following shifted symplectic products confirm the above commutation relations:

$$(\mathbf{u} \odot \mathbf{u})(D) = D^{-1} + D, \qquad (\mathbf{v} \odot \mathbf{v})(D) = D^{-1} + D, \qquad (\mathbf{u} \odot \mathbf{v})(D) = D^{-1}.$$

We note two useful properties of the shifted symplectic product $\odot$. Suppose $f(D) \in \mathbb{Z}_2(D)$ with $\deg(f) \geq 0$. Let us denote scalar polynomial multiplication as follows:

$$(f\ \mathbf{u})(D) = \left[f(D) u_1(D) \quad \cdots \quad f(D) u_n(D)\right]. \tag{4.36}$$

The following identities hold.

$$((f\ \mathbf{u}) \odot \mathbf{v})(D) = f(D)(\mathbf{u} \odot \mathbf{v})(D), \tag{4.37}$$

$$(\mathbf{u} \odot (f\ \mathbf{v}))(D) = f\left(D^{-1}\right)(\mathbf{u} \odot \mathbf{v})(D). \tag{4.38}$$

We also remark that

$$(\mathbf{u} \odot \mathbf{v})(D) = (\mathbf{v} \odot \mathbf{u})(D),$$

iff

$$(\mathbf{u} \odot \mathbf{v})(D) = (\mathbf{u} \odot \mathbf{v})\left(D^{-1}\right).$$

The shifted symplectic product $\odot$ vanishes for all generators in a quantum convolutional code because of the commutativity requirement in Definition 4.1.1. The cases where the shifted symplectic product does not vanish (where the two Pauli sequences anticommute for one or more shifts) are important for constructing entanglement-assisted quantum convolutional codes.

In general, we represent a rate-k/n quantum convolutional code with an $(n-k)\times 2n$-dimensional quantum check matrix $H(D)$ according to the P2B isomorphism. The entries of $H(D)$ are binary polynomials where

$$H(D) = \left[\begin{array}{c|c} Z(D) & X(D) \end{array}\right],$$

and $Z(D)$ and $X(D)$ are both $(n-k)\times n$-dimensional binary polynomial matrices. The following matrix $\Omega(D)$ captures the commutation relations of the generators in $H(D)$:

$$\Omega(D) = Z(D)\, X^T\left(D^{-1}\right) + X(D)\, Z^T\left(D^{-1}\right).$$

The reader can verify that the matrix elements $[\Omega(D)]_{ij}$ of $\Omega(D)$ are the shifted symplectic products between the i^{th} and j^{th} respective rows $h_i(D)$ and $h_j(D)$ of $H(D)$:

$$[\Omega(D)]_{ij} = (h_i \odot h_j)(D).$$

We call the matrix $\Omega(D)$ the *shifted symplectic product matrix* because it encodes all of the shifted symplectic products or the commutation relations of the code. This matrix is equal to the null

matrix for a valid quantum convolutional code because all generators commute with themselves and with all n-qubit shifts of themselves and each other. For a general set of generators, $\Omega(D)$ is not equal to the null matrix and obeys the symmetry: $\Omega(D) = \Omega^T\left(D^{-1}\right)$.

4.4 Row and Column Operations

We can perform row operations on binary polynomial matrices for quantum convolutional codes. A row operation is merely a "mental" operation that has no effect on the states in the codespace or on the error-correcting properties of the code. It just changes the rows of the check matrix for a code. We have three types of row operations:

(i). An elementary row operation multiplies a row times an arbitrary binary polynomial and adds the result to another row. This additive invariance holds for any code that admits a description within the stabilizer formalism. Additive codes are invariant under multiplication of the stabilizer generators in the "Pauli picture" or under row addition in the "binary-polynomial picture."

(ii). Another type of row operation is to multiply a row by an arbitrary power of D. Ollivier and Tillich discuss such row operations as "multiplication of a line by D" and use them to find encoding operations for their quantum convolutional codes [44]. Grassl and Rötteler use this type of operation to find a subcode of a given quantum convolutional code with an equivalent asymptotic rate and equivalent error-correcting properties [45].

(iii). We also employ row operations that multiply a row by an arbitrary polynomial (not necessarily a power of D). This type of row operation occurs when we have generators with infinite weight that we would like to reduce to finite weight so that the receiver can perform measurements in an online fashion as qubits arrive from the noisy channel.

We can encode all of the above types of row operations in a full-rank matrix $R(D)$ with rational polynomial entries. Let $H'(D)$ denote the resulting check matrix after performing a set of row operations in the matrix $R(D)$ where $H'(D) = R(D)H(D)$. The resulting effect on the shifted

symplectic product matrix $\Omega(D)$ is to change it to another shifted symplectic product matrix $\Omega'(D)$ related to $\Omega(D)$ by

$$\Omega'(D) = R(D)\,\Omega(D)\,R^T\left(D^{-1}\right). \tag{4.39}$$

Row operations do not change the commutation relations of a valid quantum convolutional code because its shifted symplectic product matrix is equal to the null matrix. But row operations do change the commutation relations of a set of generators whose corresponding shifted symplectic product matrix is not equal to the null matrix. This ability to change the commutation relations through row operations is crucial for constructing entanglement-assisted quantum convolutional codes from an arbitrary set of generators. We use entanglement to resolve any anticommutativity in the generators.

We can also perform column operations on binary polynomial matrices for quantum convolutional codes. Column operations do change the error-correcting properties of the code and are important for realizing a periodic encoding circuit for the code. We have two types of column operations:

(i). An elementary column operation multiplies one column by an arbitrary binary polynomial and adds the result to another column. We implement elementary column operations with gates from the shift-invariant Clifford group [47, 45].

(ii). Another column operation is to multiply column i in both the "X" and "Z" matrix by D^l where $l \in \mathbb{Z}$. We perform this operation by delaying or advancing the processing of qubit i by l frames relative to the original frame.

(iii). An infinite-depth column operation multiplies one column in the "X" matrix by a rational polynomial whose numerator is one and multiplies the corresponding column in the "Z" matrix by a corresponding finite polynomial.

A column operation implemented on the "X" side of the binary polynomial matrix has a corresponding effect on the "Z" side of the binary polynomial matrix. This corresponding effect is a manifestation of the Heisenberg uncertainty principle because commutation relations remain invariant with respect to the action of unitary quantum gates. The shifted symplectic product is

therefore invariant with respect to column operations from the shift-invariant Clifford group. The next two sections describe possible column operations for implementing encoding circuits.

4.5 Finite-Depth Clifford Operations

One of the main advantages of a quantum convolutional code is that its encoding circuit has a periodic form. We can encode a stream of quantum information with the same physical routines or devices and therefore reduce encoding and decoding complexity.

Ollivier and Tillich were the first to discuss encoding circuits for quantum convolutional codes [43, 44]. They provided a set of necessary and sufficient conditions to determine when an encoding circuit is noncatastrophic. A *noncatastrophic encoding circuit* does not propagate uncorrected errors infinitely through the decoded information qubit stream. Classical convolutional coding theory has a well developed theory of noncatastrophic encoding circuits [80].

Grassl and Rötteler later showed that Ollivier and Tillich's conditions for a circuit to be noncatastrophic are too restrictive [45, 46, 47]. They found subcodes of quantum convolutional codes that admit noncatastrophic encoders. The noncatastrophic encoders are a sequence of Clifford circuits with finite depth. They developed a formalism for encapsulating the periodic structure of an encoding circuit by decomposing the encoding circuit as a set of elementary column operations. Their decoding circuits are exact inverses of their encoding circuits because their decoding circuits perform the encoding operations in reverse order.

Definition 4.5.1. *A finite-depth operation transforms every finite-weight stabilizer generator to one with finite weight.*

This property is important because we do not want the decoding circuit to propagate uncorrected errors into the information qubit stream [80]. A finite-depth decoding circuit corresponding to any stabilizer for a quantum convolutional code exists by the algorithm given in Ref. [45].

We review the finite-depth operations in the shift-invariant Clifford group [45, 46, 47]. The shift-invariant Clifford group is an extension of the Clifford group operations mentioned in Section 2.2. We describe how finite-depth operations in the shift-invariant Clifford group affect the binary polynomial matrix for a code. Each of the following operations acts on every frame of a quantum convolutional code.

(i). The sender performs a CNOT from qubit i to qubit j in every frame where qubit j is in a frame delayed by k. The effect on the binary polynomial matrix is to multiply column i by D^k and add the result to column j in the "X" matrix and to multiply column j by D^{-k} and add the result to column i in the "Z" matrix.

(ii). A Hadamard on qubit i swaps column i in the "X" matrix with column i in the "Z" matrix.

(iii). A phase gate on qubit i adds column i from the "X" matrix to column i in the "Z" matrix.

(iv). A controlled-phase gate from qubit i to qubit j in a frame delayed by k multiplies column i in the "X" matrix by D^k and adds the result to column j in the "Z" matrix. It also multiplies column j in the "X" matrix by D^{-k} and adds the result to column i in the "Z" matrix.

(v). A controlled-phase gate from qubit i to qubit i in a frame delayed by k multiplies column i in the "X" matrix by $D^k + D^{-k}$ and adds the result to column i in the "Z" matrix.

We use finite-depth operations extensively in the next few chapters. Figure 5.2 gives an example of an entanglement-assisted quantum convolutional code that employs several finite-depth operations. The circuit encodes a stream of information qubits with the help of ebits shared between sender and receiver.

Multiple CNOT gates can realize an elementary column operation as described at the end of the previous section. Suppose the elementary column operation multiplies column i in the "X" matrix by $f(D)$ and adds the result to column j. Polynomial $f(D)$ is a summation of some finite set $\{l_1, \ldots, l_n\}$ of powers of D:

$$f(D) = D^{l_1} + \cdots + D^{l_n}.$$

We can realize $f(D)$ by performing a CNOT gate from qubit i to qubit j in a frame delayed by l_i for each $i \in \{1, \ldots, n\}$.

4.6 Infinite-Depth Clifford Operations

We now introduce an infinite-depth operation to the set of operations in the shift-invariant Clifford group available for encoding and decoding quantum convolutional codes.

Definition 4.6.1. *An infinite-depth operation can transform a finite-weight stabilizer generator to one with infinite weight (but does not necessarily do so to every finite-weight generator).*

A decoding circuit with infinite-depth operations on qubits sent over the noisy channel is undesirable because it spreads uncorrected errors infinitely into the decoded information qubit stream. But an encoding circuit with infinite-depth operations is acceptable if we assume a communication paradigm in which the only noisy process is the noisy quantum channel.

The next chapter shows several examples of circuits that include infinite-depth operations. Infinite-depth operations expand the possibilities for quantum convolutional circuits in much the same way that incorporating feedback expands the possibilities for classical convolutional circuits.

We illustrate the details of several infinite-depth operations by first providing some specific examples of infinite-depth operations and then show how to realize an arbitrary infinite-depth operation.

We consider both the stabilizer and the logical operators for the information qubits in our analysis. Tracking both of these sets of generators is necessary for determining the proper decoding circuit when including infinite-depth operations.

4.6.1 Examples of Infinite-Depth Operations

Our first example of an infinite-depth operation involves a stream of information qubits and ancilla qubits. We divide the stream into frames of three qubits where each frame has two ancilla qubits and one information qubit. The following two generators and each of their three-qubit shifts stabilize the initial qubit stream:

$$\cdots \left| \begin{array}{ccc|ccc|ccc} I & I & I & Z & I & I & I & I & I \\ I & I & I & I & Z & I & I & I & I \end{array} \right| \cdots \tag{4.40}$$

The binary polynomial matrix corresponding to this stabilizer is as follows:

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{array} \right]. \tag{4.41}$$

We obtain any Pauli sequence in the stabilizer by multiplying the above rows by a power of D and applying the inverse of the P2B isomorphism. The logical operators for the information qubits are as follows:

$$\cdots \left|\begin{array}{ccc|ccc|ccc} I & I & I & I & I & X & I & I & I \\ I & I & I & I & I & Z & I & I & I \end{array}\right| \cdots$$

They also admit a description with a binary polynomial matrix:

$$\left[\begin{array}{ccc|ccc} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{array}\right]. \tag{4.42}$$

We refer to the above matrix as the "information-qubit matrix."

Encoding

Suppose we would like to encode the above stream so that the following generators stabilize it:

$$\cdots \left|\begin{array}{ccc|ccc|ccc} I & I & I & X & X & X & X & X & I \\ I & I & I & Z & Z & I & I & I & I \end{array}\right| \cdots,$$

or equivalently, the following binary polynomial matrix stabilizes it:

$$\left[\begin{array}{ccc|ccc} 0 & 0 & 0 & D+1 & D+1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{array}\right]. \tag{4.43}$$

We encode the above stabilizer using a combination of finite-depth operations and an infinite-depth operation. We perform a Hadamard on the first qubit in each frame and follow with a CNOT from the first qubit to the second and third qubits in each frame. These operations transform the matrix in (4.41) to the following matrix

$$\left[\begin{array}{ccc|ccc} 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{array}\right],$$

or equivalently transform the generators in (4.40) to the following generators:

$$\cdots \left| \begin{array}{ccc|ccc|ccc|} I & I & I & X & X & X & I & I & I \\ I & I & I & Z & Z & I & I & I & I \end{array} \right| \cdots$$

The information-qubit matrix becomes

$$\left[\begin{array}{ccc|ccc} 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{array} \right].$$

We now perform an infinite-depth operation: a CNOT from the third qubit in one frame to the third qubit in a delayed frame and repeat this operation for all following frames. Figure 4.2 shows this operation acting on our stream of qubits with three qubits per frame. The effect of this operation is to translate the above stabilizer generators as follows:

$$\cdots \left| \begin{array}{ccc|ccc|ccc|ccc|} I & I & I & X & X & X & I & I & X & I & I & X \\ I & I & I & Z & Z & I & I & I & I & I & I & I \end{array} \right| \cdots$$

The first generator above and each of its three-qubits shifts is an infinite-weight generator if the above sequence of CNOT gates acts on the entire countably-infinite qubit stream. We represent the above stabilizer with the binary *rational* polynomial matrix

$$\left[\begin{array}{ccc|ccc} 0 & 0 & 0 & 1 & 1 & 1/(1+D) \\ 1 & 1 & 0 & 0 & 0 & 0 \end{array} \right], \tag{4.44}$$

where $1/(1+D) = 1 + D + D^2 + \ldots$ is a repeating fraction. The operation is infinite-depth because it translates the original finite-weight stabilizer generator to one with infinite weight.

It is possible to perform a row operation that multiplies the first row by $D+1$. This operation gives a stabilizer matrix that is equivalent to the desired stabilizer in (4.43). The receiver of the encoded qubits measures the finite-weight stabilizer generators in (4.43) to diagnose errors. These measurements do not disturb the information qubits because they also stabilize the encoded stream.

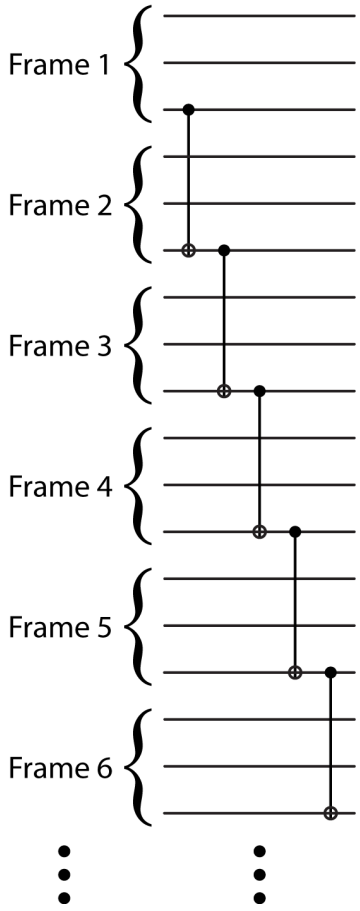

Figure 4.2: An example of an infinite-depth operation. A sequence of CNOT gates acts on the third qubit of every frame. This infinite-depth operation effectively multiplies the third column of the "X" side of the binary polynomial matrix by the rational polynomial $1/(1+D)$ and multiplies the third column of the "Z" side of the binary polynomial matrix by $1+D^{-1}$.

The above encoding operations transform the information-qubit matrix as follows:

$$\left[\begin{array}{ccc|ccc} 0 & 0 & 0 & 0 & 0 & 1/(1+D) \\ 1 & 0 & 1+D^{-1} & 0 & 0 & 0 \end{array}\right]. \tag{4.45}$$

The infinite-depth operation on the third qubit has an effect on the "Z" or left side of the information-qubit matrix as illustrated in the second row of the above matrix. The effect is to multiply the third column of the "Z" matrix by $f\left(D^{-1}\right)$ if the operation multiplies the third column of the "X" matrix by $1/f\left(D\right)$. This corresponding action on the "Z" side occurs because the commutation relations of the Pauli operators remain invariant under quantum gates (due to the Heisenberg uncertainty principle), or equivalently, the shifted symplectic product remains invariant under column operations. The original shifted symplectic product for the logical operators is one, and it remains as one because $f((D^{-1})^{-1})/f\left(D\right) = 1$.

Decoding

We perform finite-depth operations to decode the stream of information qubits. Begin with the stabilizer and information-qubit matrix in (4.44) and (4.45) respectively. Perform a CNOT from the first qubit to the second qubit. The stabilizer becomes

$$\left[\begin{array}{ccc|ccc} 0 & 0 & 0 & 1 & 0 & 1/(1+D) \\ 0 & 1 & 0 & 0 & 0 & 0 \end{array}\right],$$

and the information-qubit matrix does not change. Perform a CNOT from the third qubit to the first qubit in the same frame and in a delayed frame. These gates multiply column three in the "X" matrix by $1 + D$ and add the result to column one. The gates also multiply column one in the "Z" matrix by $1 + D^{-1}$ and add the result to column three. The effect is as follows on both the stabilizer

$$\left[\begin{array}{ccc|ccc} 0 & 0 & 0 & 0 & 0 & 1/(1+D) \\ 0 & 1 & 0 & 0 & 0 & 0 \end{array}\right], \tag{4.46}$$

and the information-qubit matrix

$$\left[\begin{array}{ccc|ccc} 0 & 0 & 0 & 1 & 0 & 1/(1+D) \\ 1 & 0 & 0 & 0 & 0 & 0 \end{array}\right]. \tag{4.47}$$

We can multiply the logical operators by any element of the stabilizer and obtain an equivalent logical operator [17]. We perform this multiplication in the "binary-polynomial picture" by adding the first row of the stabilizer in (4.46) to the first row of (4.47). The information-qubit matrix becomes

$$\left[\begin{array}{ccc|ccc} 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{array}\right], \tag{4.48}$$

so that the resulting logical operators act only on the first qubit of every frame. We have successfully decoded the information qubits with finite-depth operations. The information qubits teleport coherently [82, 83] from being the third qubit of each frame as in (4.42) to being the first qubit of each frame as in (4.48). We exploit the above method of encoding with infinite-depth operations and decoding with finite-depth operations for the class of entanglement-assisted quantum convolutional codes in Section 5.4.1.

4.6.2 General Infinite-Depth Operations

We discuss the action of a general infinite-depth operation on two weight-one "X" and "Z" Pauli sequences where each frame has one Pauli matrix. Our analysis then determines the effect of an infinite-depth operation on an arbitrary stabilizer or information-qubit matrix. The generators in the "Pauli picture" are as follows:

$$\cdots \left|\begin{array}{c} I \\ I \end{array}\right| \left|\begin{array}{c} X \\ Z \end{array}\right| \left|\begin{array}{c} I \\ I \end{array}\right| \cdots, \tag{4.49}$$

or as follows in the "binary-polynomial picture":

$$\left[\begin{array}{c|c} 0 & 1 \\ 1 & 0 \end{array}\right].$$

An infinite-depth $1/f(D)$ operation, where $f(D)$ is an arbitrary polynomial, transforms the above matrix to the following one:

$$\left[\begin{array}{c|c} 0 & 1/f(D) \\ f\left(D^{-1}\right) & 0 \end{array}\right].$$

A circuit that performs this transformation preserves the shifted symplectic product because $f\left(D^{-1}\right) \cdot 1/f\left(D^{-1}\right) = 1$. The circuit operates on a few qubits at a time and is shift-invariant—the same device or physical routines implement it.

First perform the long division expansion of binary rational polynomial $1/f(D)$. This expansion has a particular repeating pattern with period l. For example, suppose that $f(D) = 1 + D + D^3$. Its long-division expansion is $1 + D + D^2 + D^4 + D^7 + D^8 + D^9 + D^{11} + \ldots$ and exhibits a repeating pattern with period seven. We want a circuit that realizes the following Pauli generators

$$\cdots \left|\begin{array}{c} I \\ Z \end{array}\right| \begin{array}{c} I \\ I \end{array} \left|\begin{array}{c} I \\ Z \end{array}\right| \begin{array}{c} X \\ Z \end{array} \left|\begin{array}{c} X \\ I \end{array}\right| \begin{array}{c} X \\ I \end{array} \left|\begin{array}{c} I \\ I \end{array}\right| \begin{array}{c} X \\ I \end{array} \left|\begin{array}{c} I \\ I \end{array}\right| \cdots, \tag{4.50}$$

where the pattern in the X matrices is the same as the repeating polynomial $1/f(D)$ and continues infinitely to the right, and the pattern on the Z matrices is the same as that in $f\left(D^{-1}\right)$ and terminates at the left. The above Pauli sequence is equivalent to the following binary rational polynomial matrix:

$$\left[\begin{array}{c|c} 0 & 1/\left(1 + D + D^3\right) \\ 1 + D^{-1} + D^{-3} & 0 \end{array}\right].$$

We now discuss a method that realizes an arbitrary rational polynomial $1/f(D)$ as an infinite-depth operation. Our method for encoding the generators in (4.50) from those in (4.49) consists of a "sliding-window" technique that determines transformation rules for the circuit. The circuit is an additive, shift-invariant filtering operation. It resembles an infinite-impulse response filter because the resulting sequence extends infinitely. In general, the number N of qubits that the encoding unitary operates on is as follows

$$N = \deg\left(f(D)\right) - \mathrm{del}\left(f(D)\right) + 1,$$

where $\deg(f(D))$ and $\mathrm{del}(f(D))$ are the respective highest and lowest powers of polynomial $f(D)$. Therefore, our exemplary encoding unitary operates on four qubits at a time. We delay the original sequence in (4.49) by three frames. These initial frames are "scratch" frames that give the encoding unitary enough "room" to generate the desired Paulis in (4.50). The first set of transformation rules is as follows

$$\left.\begin{array}{c|c|c|c} I & I & I & X \\ I & I & I & Z \end{array}\right. \rightarrow \left.\begin{array}{c|c|c|c} I & I & I & X \\ Z & I & Z & Z \end{array}\right., \tag{4.51}$$

and generates the first four elements of the pattern in (4.50). Now that the encoding unitary has acted on the first four frames, we need to shift our eyes to the right by one frame in the sequence in (4.50) to determine the next set of rules. So we shift the above outputs by one frame to the *left* (assuming that only identity matrices lie to the right) and determine the next set of transformation rules that generate the next elements of the sequence in (4.50):

$$\left.\begin{array}{c|c|c|c} I & I & X & I \\ I & Z & Z & I \end{array}\right. \rightarrow \left.\begin{array}{c|c|c|c} I & I & X & X \\ I & Z & Z & I \end{array}\right..$$

Shift the above outputs to the left by one frame to determine the next set of transformation rules:

$$\left.\begin{array}{c|c|c|c} I & X & X & I \\ Z & Z & I & I \end{array}\right. \rightarrow \left.\begin{array}{c|c|c|c} I & X & X & X \\ Z & Z & I & I \end{array}\right..$$

We obtain the rest of the transformation rules by continuing this sliding process, and we stop when the pattern in the sequence in (4.50) begins to repeat:

$$\left.\begin{array}{c|c|c|c} X & X & X & I \\ Z & I & I & I \\ X & X & I & I \\ X & I & X & I \\ I & X & I & I \\ X & I & I & I \end{array}\right. \rightarrow \left.\begin{array}{c|c|c|c} X & X & X & I \\ Z & I & I & I \\ X & X & I & X \\ X & I & X & I \\ I & X & I & I \\ X & I & I & X \end{array}\right..$$

The above set of rules determines the encoding unitary and only a few of them are actually necessary. We can multiply the rules together to form equivalent rules because the circuit obeys additivity (in the "binary-polynomial picture"). The rules become as follows after rearranging into a standard form:

$$\left.\begin{array}{c|c|c|c} Z & I & I & I \\ I & Z & I & I \\ I & I & Z & I \\ I & I & I & Z \\ X & I & I & I \\ I & X & I & I \\ I & I & X & I \\ I & I & I & X \end{array}\right. \rightarrow \left.\begin{array}{c|c|c|c} Z & I & I & I \\ I & Z & I & I \\ I & I & Z & I \\ Z & I & Z & Z \\ X & I & I & X \\ I & X & I & I \\ I & I & X & X \\ I & I & I & X \end{array}\right. .$$

A CNOT from qubit one to qubit four and a CNOT from qubit three to qubit four suffice to implement this circuit. We repeatedly apply these operations shifting by one frame at a time to implement the infinite-depth operation. We could have observed that these gates suffice to implement the "Z" transformation in the first set of transformation rules in (4.51), but we wanted to show how this method generates the full periodic "X" sequence in (4.50). Figure 4.3 shows how the above encoding unitary acts on a stream of quantum information.

We can determine the encoding unitary for an arbitrary rational polynomial $1/f\left(D\right)$ using a similar method. Suppose that $\mathrm{del}\left(f\left(D\right)\right) = n$ and suppose $n \neq 0$ as in the above case. First delay or advance the frames if $n > 0$ or if $n < 0$ respectively. Determine the CNOT gates that transform the "Z" Pauli sequence

$$\left[\begin{array}{c|c} 1 & 0 \end{array}\right]$$

to

$$\left[\begin{array}{c|c} D^n f\left(D^{-1}\right) & 0 \end{array}\right].$$

These CNOT gates form the encoding circuit that transform both the "X" and "Z" Pauli sequences. We perform the encoding unitary, shift by one frame, perform it again, and keep repeating. Our method encodes any arbitrary polynomial $1/f\left(D\right)$ on the "X" side and $f\left(D^{-1}\right)$ on the "Z" side.

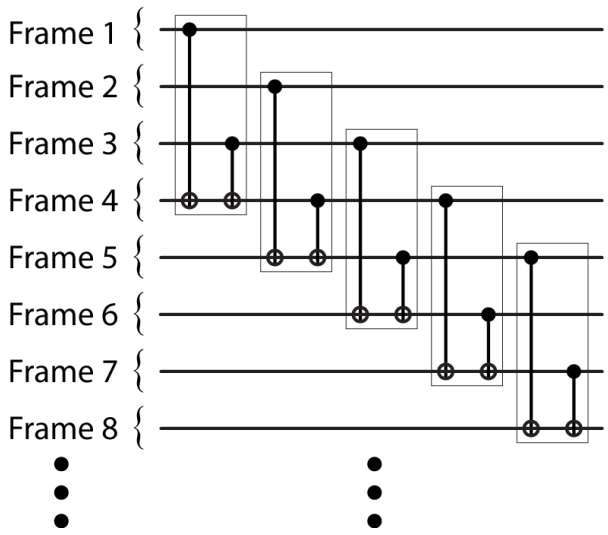

Figure 4.3: Another example of an infinite-depth operation. An infinite-depth operation acts on qubit i in every frame. This particular infinite-depth operation multiplies column i on the "X" side of the binary polynomial matrix by $1/\left(1+D+D^3\right)$ and multiplies column i on the "Z" side of the binary polynomial matrix by $1+D^{-1}+D^{-3}$.

We can implement the "time-reversed" polynomial $1/f\left(D^{-1}\right)$ on the "X" side by first delaying the frames by $m = \deg\left(f\left(D\right)\right) - \mathrm{del}\left(f\left(D\right)\right)$ frames and performing the circuit corresponding to $1/D^m\left(f\left(D^{-1}\right)\right)$. These operations implement the circuit $D^m/D^m\left(f\left(D^{-1}\right)\right) = 1/f\left(D^{-1}\right)$.

4.6.3 Infinite-Depth Operations in Practice

We assume above that each of the infinite-depth operations acts on the entire countably-infinite stream of qubits. In practice, each infinite-depth operation acts on a finite number of qubits at a time so that the encoding and decoding circuits operate in an "online" manner. Therefore, each infinite-depth operation approximates its corresponding rational polynomial. This approximation does not pose a barrier to implementation. We can implement each of the above infinite-depth operations by padding the initial qubits of the information qubit stream with some "scratch" qubits. We first transmit these "scratch" qubits that contain no useful quantum information so that the later information qubits enjoy the full protection of the code. These scratch qubits do not affect the asymptotic rate of the code and merely serve as a convenience for implementing the

infinite-depth operations. From now on, we adhere to describing infinite-depth operations with binary rational polynomials because it is more convenient to do so mathematically.

4.7 Closing Remarks

This chapter reviewed in detail the method of quantum convolutional coding. We discussed how to represent a quantum convolutional code with a matrix of binary polynomials. The shifted symplectic product is crucial in several of the forthcoming chapters because it determines the commutation relations for an arbitrary set of convolutional generators. In general, a set of generators do not form a commuting set and we can use entanglement to resolve their anticommutativity. In this chapter, we also developed many of the operations that manipulate quantum convolutional codes. We repeatedly use these operations in the next few chapters.

Chapter 5

Entanglement-Assisted Quantum Convolutional Coding: The CSS Case

O Calderbank, good Shor, and bright Steane,
Methinks your construction is keen,
But it makes me tense,
this Pauli sequence,
I just want our quantum bits clean.

In this chapter, we develop a theory of entanglement-assisted quantum convolutional coding for a broad class of codes. Our major result is that we can produce an entanglement-assisted quantum convolutional code from two *arbitrary* classical binary convolutional codes. The resulting quantum convolutional codes admit a Calderbank-Shor-Steane (CSS) structure [21, 22, 32]. The rates and error-correcting properties of the two binary classical convolutional codes directly determine the corresponding properties of the entanglement-assisted quantum convolutional code.

Our CSS entanglement-assisted quantum convolutional codes divide into two classes based on certain properties of the classical codes from which we produce them. These properties of the classical codes determine the structure of the encoding and decoding circuit for the code, and the structure of the encoding and decoding circuit in turn determines the class of the entanglement-assisted quantum convolutional code.

(i). Codes in the first class admit both a finite-depth encoding and decoding circuit.

(ii). Codes in the second class have an encoding circuit that employs both finite-depth and infinite-depth operations. Their decoding circuits have finite-depth operations only.

We structure this chapter as follows. We outline the operation of an entanglement-assisted quantum convolutional code and present our main theorem in Section 5.1. This theorem shows how to produce a CSS entanglement-assisted quantum convolutional code from two arbitrary classical binary convolutional codes. The theorem gives the rate and error-correcting properties of a CSS entanglement-assisted quantum convolutional code as a function of the parameters of the classical convolutional codes. Section 5.3 completes the proof of the theorem for our first class of entanglement-assisted quantum convolutional codes. Section 5.4 completes the proof of our theorem for the second class of entanglement-assisted quantum convolutional codes. We discuss the implications of the assumptions for the different classes of entanglement-assisted quantum convolutional codes while developing the constructions. Our theory produces high-performance quantum convolutional codes by importing high-performance classical convolutional codes.

5.1 Operation of an Entanglement-Assisted Quantum Convolutional Code

An entanglement-assisted quantum convolutional code operates similarly to a standard quantum convolutional code. The main difference is that the sender and receiver share entanglement in the form of ebits before quantum communication begins. An $[[n, k; c]]$ entanglement-assisted quantum convolutional code encodes k information qubits per frame with the help of c ebits and $n - k - c$ ancilla qubits per frame. Figure 5.1 highlights the main features of the operation of an entanglement-assisted quantum convolutional code. An entanglement-assisted quantum convolutional code operates as follows:

(i). The sender encodes a stream of quantum information using both additional ancillas and ebits. The sender performs the encoding operations on her qubits only (i.e., not including the halves of the ebits in possession of the receiver). The encoding operations have a periodic structure so that the same operations act on qubits in different frames and give the code a

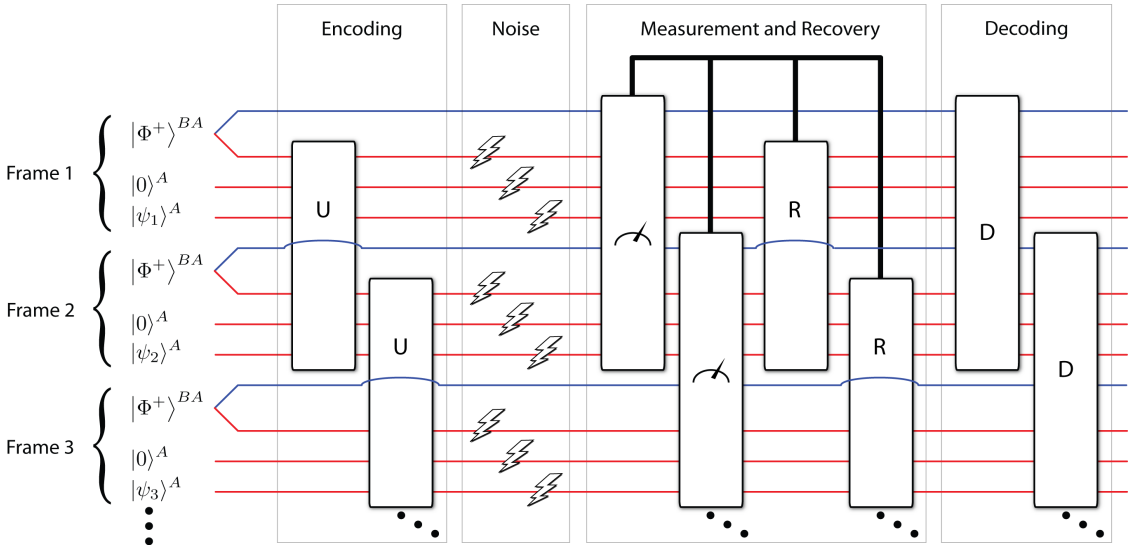

Figure 5.1: (Color online) An entanglement-assisted quantum convolutional code operates on a stream of qubits partitioned into a countable number of frames. The sender encodes the frames of information qubits, ancilla qubits, and half of shared ebits with a repeated, overlapping encoding circuit U. The noisy channel affects the sender's encoded qubits but does not affect the receiver's half of the shared ebits. The receiver performs overlapping measurements on both the encoded qubits and his half of the shared ebits. These measurements produce an error syndrome which the receiver can process to determine the most likely error. The receiver reverses the errors on the noisy qubits from the sender. The final decoding circuit operates on all qubits in a frame and recovers the original stream of information qubits.

memory structure. The sender can perform these encoding operations in an online manner as she places more qubits in the unencoded qubit stream.

(ii). The sender transmits her encoded qubits over the noisy quantum communication channel. The noisy channel does not affect the receiver's half of the shared ebits.

(iii). The receiver combines the received noisy qubits with his half of the ebits and performs measurements to diagnose errors that may occur. These measurements may overlap on some of the same qubits.

(iv). The receiver then diagnoses errors using a classical error estimation algorithm such as Viterbi error estimation [2] and reverses the estimates of the errors.

(v). The receiver finally performs an online decoding circuit that outputs the original information qubit stream.

5.2 The CSS Construction

Our main theorem below allows us to import two arbitrary classical convolutional codes for use as a CSS entanglement-assisted quantum convolutional code. Grassl and Rötteler were the first to construct CSS quantum convolutional codes from two classical binary convolutional codes that satisfy an orthogonality constraint—the polynomial parity check matrices $H_1(D)$ and $H_2(D)$ of the two classical codes are orthogonal with respect to the shifted symplectic product:

$$H_1(D) H_2^T\left(D^{-1}\right) = 0. \tag{5.1}$$

The resulting symplectic code has a self-orthogonal parity-check matrix when we join them together using the CSS construction. Our theorem generalizes the work of Grassl and Rötteler because we can import two *arbitrary* classical binary convolutional codes—the codes do not necessarily have to obey the self-orthogonality constraint.

The theorem gives a direct way to compute the amount of entanglement that the code requires. The number of ebits required is equal to the rank of a particular matrix derived from the check matrices of the two classical codes. It generalizes the earlier theorems discussed in Section 3.6

that determine the amount of entanglement required for an entanglement-assisted quantum block code.

Theorem 5.2.1 also provides a formula to compute the performance parameters of the entanglement-assisted quantum convolutional code from the performance parameters of the two classical codes. This formula ensures that high-rate classical convolutional codes produce high-rate entanglement-assisted quantum convolutional codes. Our constructions also ensure high performance for the "trade-off" and "catalytic" rates by minimizing the number of ebits that the codes require.

We begin the proof of the theorem in this section and complete it in different ways for each of our two classes of entanglement-assisted quantum convolutional codes in Sections 5.3 and 5.4. The proofs detail how to encode a stream of information qubits, ancilla qubits, and shared ebits into a code that has the CSS structure.

Theorem 5.2.1. *Let $H_1(D)$ and $H_2(D)$ be the respective check matrices corresponding to non-catastrophic, delay-free encoders for classical binary convolutional codes C_1 and C_2. Suppose that classical code C_i encodes k_i information bits with n bits per frame where $i = 1, 2$. The respective dimensions of $H_1(D)$ and $H_2(D)$ are thus $(n - k_1) \times n$ and $(n - k_2) \times n$. Then the resulting entanglement-assisted quantum convolutional code encodes $k_1 + k_2 - n + c$ information qubits per frame and is an $[[n, k_1 + k_2 - n + c; c]]$ entanglement-assisted quantum convolutional code. The code requires c ebits per frame where c is equal to the rank of $H_1(D) H_2^T(D^{-1})$.*

Let us begin the proof of the above theorem by constructing an entanglement-assisted quantum convolutional code. Consider the following quantum check matrix in CSS form:

$$\left[\begin{array}{c|c} H_1(D) & 0 \\ 0 & H_2(D) \end{array}\right]. \tag{5.2}$$

We label the above matrix as a "quantum check matrix" for now because it does not necessarily correspond to a commuting stabilizer. The quantum check matrix corresponds to a set of Pauli sequences whose error-correcting properties are desirable. The Pauli sequences inherit these desirable properties from the classical codes C_1 and C_2.

The following lemma begins the proof of the above theorem. It details an initial decomposition of the above quantum check matrix for each of our two classes of entanglement-assisted quantum convolutional codes.

Lemma 5.2.1. *Elementary row and column operations relate the quantum check matrix in (5.2) to the following matrix*

$$\left[\begin{array}{cc|cc} E(D) & F(D) & 0 & 0 \\ 0 & 0 & I & 0 \end{array}\right].$$

where $E(D)$ is dimension $(n-k_1)\times(n-k_2)$, $F(D)$ is $(n-k_1)\times k_2$, the identity matrix is $(n-k_2)\times(n-k_2)$, and the null matrix on the right is $(n-k_2)\times k_2$. We give a definition of $E(D)$ and $F(D)$ in the following proof.

Proof. The Smith form [80] of $H_i(D)$ for each $i=1,2$ is

$$H_i(D) = A_i(D)\left[\begin{array}{cc} I & 0 \end{array}\right] B_i(D), \tag{5.3}$$

where $A_i(D)$ is $(n-k_i)\times(n-k_i)$, the matrix in brackets is $(n-k_i)\times n$, and $B_i(D)$ is $n\times n$. The Smith decomposition is somewhat analogous to the SVD decomposition of a matrix over the reals. The binary polynomials along the diagonal of the center matrix of the Smith decomposition are the *invariant factors*. The matrices $A_i(D)$ and $B_i(D)$ are a product of a sequence of elementary row and column operations respectively [80]. Let $B_{ia}(D)$ be the first $n-k_i$ rows of $B_i(D)$ and let $B_{ib}(D)$ be the last k_i rows of $B_i(D)$:

$$B_i(D) = \left[\begin{array}{c} B_{ia}(D) \\ B_{ib}(D) \end{array}\right].$$

The $(n-k_i)\times(n-k_i)$ identity matrix in brackets in (5.3) indicates that the invariant factors of $H_i(D)$ for each $i=1,2$ are all equal to one [80]. The invariant factors are all unity for both check matrices because the check matrices correspond to noncatastrophic, delay-free encoders [80].

Premultiplying $H_i(D)$ by $A_i^{-1}(D)$ gives a check matrix $H_i'(D)$ for each $i=1,2$. Matrix $H_i'(D)$ is a check matrix for code C_i with equivalent error-correcting properties as $H_i(D)$ because

row operations relate the two matrices. This new check matrix $H_i'(D)$ is equal to the first $n-k_i$ rows of matrix $B_i(D)$:

$$H_i'(D) = B_{ia}(D).$$

The invariant factors of $H_1(D) H_2^T\left(D^{-1}\right)$ are equivalent to those of $H_1'(D) H_2'^T\left(D^{-1}\right)$ because they are related by elementary row and column operations [80]:

$$H_1(D) H_2^T\left(D^{-1}\right) = A_1(D) H_1'(D) H_2'^T\left(D^{-1}\right) A_2^T\left(D^{-1}\right). \tag{5.4}$$

We now decompose the above quantum check matrix into a basic form using elementary row and column operations. The row operations have no effect on the error-correcting properties of the code, and the column operations correspond to elements of an encoding circuit. We later show how to incorporate ebits so that the quantum check matrix forms a valid commuting stabilizer.

Perform the row operations in matrices $A_i^{-1}(D)$ for both check matrices $H_i(D)$. The quantum check matrix becomes

$$\left[\begin{array}{c|c} B_{1a}(D) & 0 \\ 0 & B_{2a}(D) \end{array}\right]. \tag{5.5}$$

The error-correcting properties of the above generators are equivalent to those of the generators in (5.2) because row operations relate the two sets of generators. The matrix $B_2(D)$ corresponds to a sequence $\{B_{2,i}(D)\}_{i=1}^{l}$ of elementary column operations:

$$B_2(D) = B_{2,1}(D) \cdots B_{2,l}(D) = \prod_{i=1}^{l} B_{2,i}(D).$$

The inverse matrix $B_2^{-1}(D)$ is therefore equal to the above sequence of operations in reverse order:

$$B_2^{-1}(D) = B_{2,l}(D) \cdots B_{2,1}(D) = \prod_{i=l}^{1} B_{2,i}(D).$$

Perform the elementary column operations in $B_2^{-1}(D)$ with CNOT and SWAP gates [45]. The effect of each elementary column operation $B_{2,i}(D)$ is to postmultiply the "X" matrix by $B_{2,i}(D)$

and to postmultiply the "Z" matrix by $B_{2,i}^T\left(D^{-1}\right)$. Therefore the effect of all elementary operations is to postmultiply the "Z" matrix by $B_2^T\left(D^{-1}\right)$ because

$$\prod_{i=l}^{1} B_{2,i}^T\left(D^{-1}\right) = \left(\prod_{i=1}^{l} B_{2,i}\left(D^{-1}\right)\right)^T = B_2^T\left(D^{-1}\right).$$

The quantum check matrix in (5.5) becomes

$$\left[\begin{array}{cc|cc} B_{1a}\left(D\right) B_2^T\left(D^{-1}\right) & & 0 & 0 \\ 0 & & I & 0 \end{array}\right]. \tag{5.6}$$

Let $E\left(D\right)$ be equal to the first $n-k_1$ rows and $n-k_2$ columns of the "Z" matrix:

$$E\left(D\right) \equiv B_{1,a}\left(D\right) B_{2,a}^T\left(D^{-1}\right),$$

and let $F\left(D\right)$ be equal to the first $n-k_1$ rows and last k_2 columns of the "Z" matrix:

$$F\left(D\right) \equiv B_{1,a}\left(D\right) B_{2,b}^T\left(D^{-1}\right).$$

The quantum check matrix in (5.6) is then equivalent to the following matrix

$$\left[\begin{array}{cc|cc} E\left(D\right) & F\left(D\right) & 0 & 0 \\ 0 & 0 & I & 0 \end{array}\right], \tag{5.7}$$

where each matrix above has the dimensions stated in the theorem above. □

The above operations end the initial set of operations that each of our two classes of entanglement-assisted quantum convolutional codes employs. We outline the remaining operations for each class of codes in what follows.

5.3 Codes with Finite-Depth Encoding and Decoding Circuits

This section details entanglement-assisted quantum convolutional codes in our first class. Codes in the first class admit an encoding and decoding circuit that employ finite-depth operations only. The check matrices for codes in this class have a property that allows this type of encoding and decoding. The following lemma gives the details of this property, and the proof outlines how to encode and decode this class of entanglement-assisted quantum convolutional codes.

Lemma 5.3.1. *Suppose the Smith form of* $H_1(D) H_2^T\left(D^{-1}\right)$ *is*

$$H_1(D) H_2^T\left(D^{-1}\right) = A(D) \begin{bmatrix} \Gamma(D) & 0 \\ 0 & 0 \end{bmatrix} B(D),$$

where $A(D)$ *is an* $(n-k_1)\times(n-k_1)$ *matrix,* $B(D)$ *is an* $(n-k_2)\times(n-k_2)$ *matrix,* $\Gamma(D)$ *is a diagonal* $c\times c$ *matrix whose entries are powers of* D*, and the matrix in brackets has dimension* $(n-k_1)\times(n-k_2)$*. Then the resulting entanglement-assisted quantum convolutional code has both a finite-depth encoding and decoding circuit.*

Proof. We begin the proof of this lemma by continuing where the proof of Lemma 5.2.1 ends. The crucial assumption for the above lemma is that the invariant factors of $H_1(D) H_2^T\left(D^{-1}\right)$ are all powers of D. The Smith form of $E(D)$ in (5.7) therefore becomes

$$A_1^{-1}(D) A(D) \begin{bmatrix} \Gamma(D) & 0 \\ 0 & 0 \end{bmatrix} B(D) A_2^{-1}(D),$$

by employing the hypothesis of Lemma 5.3.1 and (5.4). The rank of both $H_1(D) H_2^T\left(D^{-1}\right)$ and $E(D)$ is equal to c.

Perform the inverse of the row operations in $A_1^{-1}(D) A(D)$ on the first $n-k_1$ rows of the quantum check matrix in (5.7). Perform the inverse of the column operations in matrix $B(D) A_2^{-1}(D)$ on the first $n-k_2$ columns of the quantum check matrix in (5.7). We execute these column operations with Hadamard, CNOT, and SWAP gates. These column operations have a corresponding effect on columns in the "X" matrix, but we can exploit the identity matrix in the last $n-k_2$ rows

of the "X" matrix to counteract this effect. We perform row operations on the last $n - k_2$ rows of the matrix that act as the inverse of the column operations, and therefore the quantum check matrix in (5.7) becomes

$$\left[\begin{array}{ccc|ccc} \Gamma(D) & 0 & F_1(D) & 0 & 0 & 0 \\ 0 & 0 & F_2(D) & 0 & 0 & 0 \\ 0 & 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & 0 & I & 0 \end{array}\right],$$

where $F_1(D)$ and $F_2(D)$ are the first c and $n - k_1 - c$ respective rows of $A^{-1}(D) A_1(D) F(D)$. We perform Hadamard and CNOT gates to clear the entries in $F_1(D)$ in the "Z" matrix above. The quantum check matrix becomes

$$\left[\begin{array}{ccc|ccc} \Gamma(D) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & F_2(D) & 0 & 0 & 0 \\ 0 & 0 & 0 & I & 0 & 0 \\ 0 & I & 0 & 0 & 0 & 0 \end{array}\right]. \tag{5.8}$$

The Smith form of $F_2(D)$ is

$$F_2(D) = A_F(D) \left[\begin{array}{cc} \Gamma_F(D) & 0 \end{array}\right] B_F(D),$$

where $\Gamma_F(D)$ is a diagonal matrix whose entries are powers of D, $A_F(D)$ is $(n - k_1 - c) \times (n - k_1 - c)$, and $B_F(D)$ is $k_2 \times k_2$. The Smith form of $F_2(D)$ takes this particular form because the original check matrix $H_2(D)$ is noncatastrophic and column operations with Laurent polynomials change the invariant factors only up to powers of D.

Perform row operations corresponding to $A_F^{-1}(D)$ on the second set of $n - k_1 - c$ rows with $F_2(D)$ in (5.8). Perform column operations corresponding to $B_F^{-1}(D)$ on columns $n-k_2+1, \ldots, n$

with Hadamard, CNOT, and SWAP gates. The resulting quantum check matrix has the following form:

$$\left[\begin{array}{cccc|cccc} \Gamma(D) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \Gamma_F(D) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & I & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]. \tag{5.9}$$

We have now completed the decomposition of the original quantum check matrix in (5.2) for this class of entanglement-assisted quantum convolutional codes. It is not possible to perform row or column operations to decompose the above matrix any further. The problem with the above quantum check matrix is that it does not form a valid quantum convolutional code. The first set of rows with matrix $\Gamma(D)$ are not orthogonal under the shifted symplectic product to the third set of rows with the identity matrix on the "X" side. Equivalently, the set of Pauli sequences corresponding to the above quantum check matrix do not form a commuting stabilizer. We can use entanglement shared between sender and receiver to solve this problem. Entanglement adds columns to the above quantum check matrix to resolve the issue. The additional columns correspond to qubits on the receiver's side. We next show in detail how to incorporate ancilla qubits, ebits, and information qubits to obtain a valid stabilizer code. The result is that we can exploit the error-correcting properties of the original code to protect the sender's qubits.

Consider the following check matrix corresponding to a commuting stabilizer:

$$\left[\begin{array}{ccccc|ccccc} I & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I & I & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right], \tag{5.10}$$

where the identity matrices in the first and third sets of rows each have dimension $c \times c$, the identity matrix in the second set of rows has dimension $(n - k_1 - c) \times (n - k_1 - c)$, and the identity matrix in the fourth set of rows has dimension $(n - k_2 - c) \times (n - k_2 - c)$. The first and third sets of c rows stabilize a set of c ebits shared between Alice and Bob. Bob possesses the "left" c qubits and Alice possesses the "right" n qubits. The second and fourth sets of rows stabilize a set of

$2(n-c)-k_1-k_2$ ancilla qubits that Alice possesses. The stabilizer therefore stabilizes a set of c ebits, $2(n-c)-k_1-k_2$ ancilla qubits, and k_1+k_2-n+c information qubits.

Observe that the last n columns of the "Z" and "X" matrices in the above stabilizer are similar in their layout to the entries in (5.9). We can delay the rows of the above stabilizer by an arbitrary amount to obtain the desired stabilizer. So the above stabilizer is a subcode of the following stabilizer in the sense of Ref. [45]:

$$\left[\begin{array}{ccccc|ccccc} \Gamma(D) & \Gamma(D) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \Gamma_F(D) & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I & I & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right].$$

The stabilizer in (5.10) has equivalent error-correcting properties to and the same asymptotic rate as the above desired stabilizer. The above stabilizer matrix is an augmented version of the quantum check matrix in (5.9) that includes entanglement. The sender performs all of the encoding column operations detailed in the proofs of this lemma and Lemma 5.2.1 in reverse order. The result of these operations is an $[[n, k_1+k_2-n+c; c]]$ entanglement-assisted quantum convolutional code with the same error-correcting properties as the quantum check matrix in (5.2). The receiver decodes the original information-qubit stream by performing the column operations in the order presented. The information qubits appear as the last k_1+k_2-n+c in each frame of the stream (corresponding to the k_1+k_2-n+c columns of zeros in both the "Z" and "X" matrices above). □

Example 5.3.1. Consider a classical convolutional code with the following check matrix:

$$H(D) = \left[\begin{array}{cc} 1+D^2 & 1+D+D^2 \end{array}\right].$$

We can use $H(D)$ in an entanglement-assisted quantum convolutional code to correct for both bit-flip errors and phase-flip errors. We form the following quantum check matrix:

$$\left[\begin{array}{cc|cc} 1+D^2 & 1+D+D^2 & 0 & 0 \\ 0 & 0 & 1+D^2 & 1+D+D^2 \end{array}\right]. \tag{5.11}$$

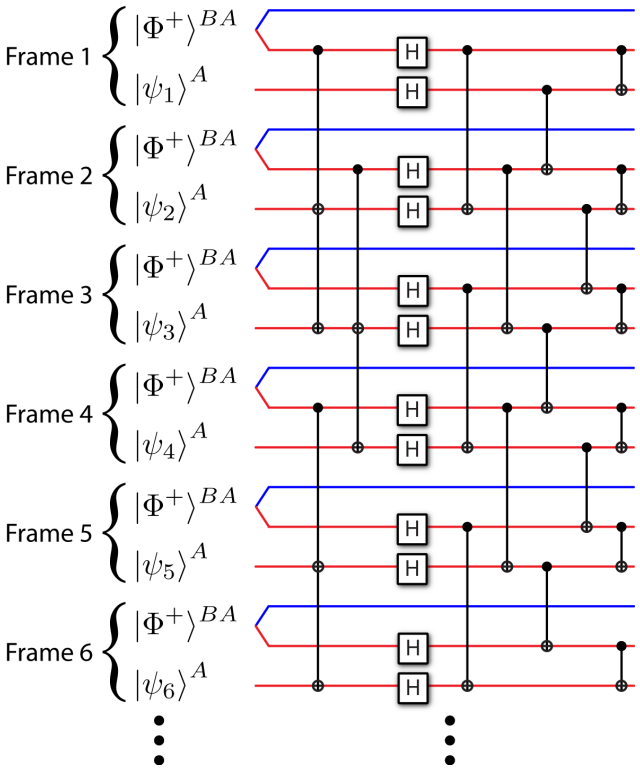

Figure 5.2: (Color online) The finite-depth encoding circuit for the entanglement-assisted quantum convolutional code in Example 5.3.1. The above operations in reverse order give a valid decoding circuit.

This code falls in the first class of entanglement-assisted quantum convolutional codes because $H(D)H^T\left(D^{-1}\right) = 1$.

We do not show the decomposition of the above check matrix as outlined in Lemma 5.3.1, but instead show how to encode it starting from a stream of information qubits and ebits. Each frame has one ebit and one information qubit. Let us begin with a polynomial matrix that stabilizes the unencoded state:

$$\left[\begin{array}{ccc|ccc} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{array}\right].$$

Alice possesses the two qubits on the "right" and Bob possesses the qubit on the "left." We label the middle qubit as "qubit one" and the rightmost qubit as "qubit two." Alice performs a CNOT

from qubit one to qubit two in a delayed frame and a CNOT from qubit one to qubit two in a frame delayed by two. The stabilizer becomes

$$\left[\begin{array}{ccc|ccc} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & D+D^2 \end{array}\right].$$

Alice performs Hadamard gates on both of her qubits. The stabilizer becomes

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & D+D^2 & 1 & 0 & 0 \end{array}\right].$$

Alice performs a CNOT from qubit one to qubit two in a delayed frame. The stabilizer becomes

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 1 & D \\ 0 & D & D+D^2 & 1 & 0 & 0 \end{array}\right].$$

Alice performs a CNOT from qubit two to qubit one in a delayed frame. The stabilizer becomes

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 1+D^2 & D \\ 0 & D & 1+D+D^2 & 1 & 0 & 0 \end{array}\right].$$

Alice performs a CNOT from qubit one to qubit two. The stabilizer becomes

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 1+D^2 & 1+D+D^2 \\ 0 & 1+D^2 & 1+D+D^2 & 1 & 0 & 0 \end{array}\right].$$

A row operation that switches the first row with the second row gives the following stabilizer:

$$\left[\begin{array}{ccc|ccc} 0 & 1+D^2 & 1+D+D^2 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1+D^2 & 1+D+D^2 \end{array}\right].$$

The entries on Alice's side of the above stabilizer have equivalent error-correcting properties to the quantum check matrix in (5.11). Figure 5.2 illustrates how the above operations encode a stream of ebits and information qubits for our example.

Discussion

Codes in the first class are more useful in practice than those in the second because their encoding and decoding circuits are finite depth. An uncorrected error propagates only to a finite number of information qubits in the decoded qubit stream. Codes in the first class therefore do not require any assumptions about noiseless encoding or decoding.

The assumption about the invariant factors in the Smith form of $H_1(D) H_2^T\left(D^{-1}\right)$ holds only for some classical check matrices. Only a subclass of classical codes satisfy this assumption, but it still expands the set of available quantum codes beyond those whose check matrices $H_1(D)$ and $H_2(D)$ are orthogonal. We need further techniques to handle the classical codes for which this assumption does not hold. The following sections provide these further techniques to handle a larger class of entanglement-assisted quantum convolutional codes.

5.4 Codes with Infinite-Depth Encoding and Finite-Depth Decoding Circuits

This section details codes whose encoding circuits have both infinite-depth and finite-depth operations. We therefore assume that encoding is noiseless to eliminate the possibility of encoding errors spreading infinitely into the encoded qubit stream. We later briefly discuss the effects of relaxing this assumption in a realistic system. Their decoding circuits require finite-depth operations only. Some of the decoding circuits are not the exact inverse of their corresponding encoding circuits, but the decoding circuits invert the effect of the encoding circuits because they produce the original stream of information qubits at their output.

Just as with the previous class, this class of codes is determined by the properties of their corresponding classical check matrices, as described in the following lemma.

Lemma 5.4.1. *Suppose the Smith form of $E(D)$ does not admit the form from Lemma 5.3.1. Then the entanglement-assisted quantum convolutional code has an encoding circuit with both infinite-depth and finite-depth operations. Its decoding circuit has finite-depth operations.*

Proof. We perform all of the operations from Lemma 5.2.1. The Smith form of $E(D)$ is in general as follows

$$A_E(D)\begin{bmatrix}\Gamma_1(D) & 0 & 0\\ 0 & \Gamma_2(D) & 0\\ 0 & 0 & 0\end{bmatrix}B_E(D),$$

where $A_E(D)$ is $(n-k_1)\times(n-k_1)$, $\Gamma_1(D)$ is an $s\times s$ diagonal matrix whose entries are powers of D, $\Gamma_2(D)$ is a $(c-s)\times(c-s)$ diagonal matrix whose entries are arbitrary polynomials, and $B_E(D)$ is $(n-k_2)\times(n-k_2)$. Perform the row operations in $A_E^{-1}(D)$ and the column operations in $B_E^{-1}(D)$ on the quantum check matrix in (5.7). Counteract the effect of the column operations on the identity matrix in the "X" matrix by performing row operations. The quantum check matrix in (5.7) becomes

$$\left[\begin{array}{cccc|cc}\Gamma_1(D) & 0 & 0 & F_1(D) & 0 & 0\\ 0 & \Gamma_2(D) & 0 & F_2(D) & 0 & 0\\ 0 & 0 & 0 & F_3(D) & 0 & 0\\ 0 & 0 & 0 & 0 & I & 0\end{array}\right],$$

where $F_1(D)$, $F_2(D)$, and $F_3(D)$ are the respective s, $c-s$, and $n-k_1-c$ rows of $A_E^{-1}(D)F(D)$. The Smith form of $F_3(D)$ is as follows

$$F_3(D)=A_{F_3}(D)\begin{bmatrix}\Gamma_{F_3}(D) & 0\end{bmatrix}B_{F_3}(D),$$

where $A_{F_3}(D)$ is $(n-k_1-c)\times(n-k_1-c)$, $\Gamma_{F_3}(D)$ is an $(n-k_1-c)\times(n-k_1-c)$ diagonal matrix whose entries are powers of D, and $B_{F_3}(D)$ is $k_2\times k_2$. The entries of $\Gamma_{F_3}(D)$ are powers of D because the original check matrix $H_2(D)$ is noncatastrophic and column and row operations

with Laurent polynomials change the invariant factors only by a power of D. Perform the row operations in $A_{F_3}^{-1}(D)$ and the column operations in $B_{F_3}^{-1}(D)$. The quantum check matrix becomes

$$\left[\begin{array}{ccccc|cc} \Gamma_1(D) & 0 & 0 & F_{1a}^{'}(D) & F_{1b}^{'}(D) & 0 & 0 \\ 0 & \Gamma_2(D) & 0 & F_{2a}^{'}(D) & F_{2b}^{'}(D) & 0 & 0 \\ 0 & 0 & 0 & \Gamma_{F_3}(D) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I & 0 \end{array}\right],$$

where $F_{1a}^{'}(D)$, $F_{1b}^{'}(D)$, $F_{2a}^{'}(D)$, $F_{2b}^{'}(D)$ are the matrices resulting from the column operations in $B_{F_3}^{-1}(D)$. Perform row operations from the entries in $\Gamma_{F_3}(D)$ to the rows above it to clear the entries in $F_{1a}^{'}(D)$ and $F_{2a}^{'}(D)$. Use Hadamard and CNOT gates to clear the entries in $F_{1b}^{'}(D)$. The quantum check matrix becomes

$$\left[\begin{array}{ccccc|cc} \Gamma_1(D) & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \Gamma_2(D) & 0 & 0 & F_{2b}^{'}(D) & 0 & 0 \\ 0 & 0 & 0 & \Gamma_{F_3}(D) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I & 0 \end{array}\right].$$

We can reduce $F_{2b}^{'}(D)$ to a lower triangular form with an algorithm consisting of column operations only. The algorithm operates on the last k_2+k_1-n+c columns. It is similar to the Smith algorithm but does not involve row operations. Consider the first row of $F_{2b}^{'}(D)$. Perform column operations between the different elements of the row to reduce it to one non-zero entry. Swap this non-zero entry to the leftmost position. Perform the same algorithm on elements $2, \ldots, k_2 + k_1 - n + c$ of the second row. Continue on for all rows of $F_{2b}^{'}(D)$ to reduce it to a matrix of the following form

$$F_{2b}^{'}(D) \rightarrow \left[\overbrace{L(D)}^{c-s} \quad \overbrace{0}^{k_1+k_2-n+s}\right],$$

where $L(D)$ is a lower triangular matrix. The above quantum check matrix becomes

$$\left[\begin{array}{cccccc|cc} \Gamma_1(D) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \Gamma_2(D) & 0 & 0 & L(D) & 0 & 0 & 0 \\ 0 & 0 & 0 & \Gamma_{F_3}(D) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & I & 0 \end{array}\right].$$

We have completed decomposition of the first set of s rows with $\Gamma_1(D)$, the third set of $n-k_1-c$ rows with $\Gamma_{F_3}(D)$, and rows $n-k_1+1,\ldots,n-k_1+s$ with the identity matrix on the "X" side.

We now consider an algorithm with infinite-depth operations to encode the following submatrix of the above quantum check matrix:

$$\left[\begin{array}{cc|cc} \Gamma_2(D) & L(D) & 0 & 0 \\ 0 & 0 & I & 0 \end{array}\right]. \tag{5.12}$$

We begin with a set of $c-s$ ebits and $c-s$ information qubits. The following matrix stabilizes the ebits

$$\left[\begin{array}{ccc|ccc} I & I & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I & I & 0 \end{array}\right],$$

and the following matrix represents the information qubits

$$\left[\begin{array}{ccc|ccc} 0 & 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I \end{array}\right],$$

where all matrices have dimension $(c-s)\times(c-s)$ and Bob possesses the $c-s$ qubits on the "left" and Alice possesses the $2(c-s)$ qubits on the "right." We track both the stabilizer and the information qubits as they progress through some encoding operations. Alice performs CNOT and Hadamard gates on her $2(c-s)$ qubits. These gates multiply the middle $c-s$ columns of the "Z" matrix by $L(D)$ and add the result to the last $c-s$ columns and multiply the last $c-s$ columns of the "X" matrix by $L^T\left(D^{-1}\right)$ and add the result to the middle $c-s$ columns. The stabilizer becomes

$$\left[\begin{array}{ccc|ccc} I & I & L(D) & 0 & 0 & 0 \\ 0 & 0 & 0 & I & I & 0 \end{array}\right],$$

and the information-qubit matrix becomes

$$\left[\begin{array}{ccc|ccc} 0 & 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & L^T\left(D^{-1}\right) & I \end{array}\right].$$

Alice performs infinite-depth operations on her first $c-s$ qubits corresponding to the rational polynomials $\gamma_{2,1}^{-1}\left(D^{-1}\right)$, ..., $\gamma_{2,c-s}^{-1}\left(D^{-1}\right)$ in $\Gamma_2^{-1}\left(D^{-1}\right)$. The stabilizer matrix becomes

$$\left[\begin{array}{ccc|ccc} I & \Gamma_2(D) & L(D) & 0 & 0 & 0 \\ 0 & 0 & 0 & I & \Gamma_2^{-1}\left(D^{-1}\right) & 0 \end{array}\right],$$

and the information-qubit matrix becomes

$$\left[\begin{array}{ccc|ccc} 0 & 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & L^T\left(D^{-1}\right)\Gamma_2^{-1}\left(D^{-1}\right) & I \end{array}\right].$$

Alice's part of the above stabilizer matrix is equivalent to the quantum check matrix in (5.12) by row operations (premultiplying the second set of rows in the stabilizer by $\Gamma_2(D)$.) Bob can therefore make stabilizer measurements that have finite weight and that are equivalent to the desired stabilizer.

We now describe a method to decode the above encoded stabilizer and information-qubit matrix so that the information qubits appear at the output of the decoding circuit. Bob performs Hadamard gates on his first and third sets of $c-s$ qubits, performs CNOT gates from the first set of qubits to the third set of qubits corresponding to the entries in $L(D)$, and performs the Hadamard gates again. The stabilizer becomes

$$\left[\begin{array}{ccc|ccc} I & \Gamma_2(D) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I & \Gamma_2^{-1}\left(D^{-1}\right) & 0 \end{array}\right], \tag{5.13}$$

and the information-qubit matrix becomes

$$\left[\begin{array}{ccc|ccc} 0 & 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & L^T\left(D^{-1}\right) & L^T\left(D^{-1}\right)\Gamma_2^{-1}\left(D^{-1}\right) & I \end{array}\right].$$

Bob finishes decoding at this point because we can equivalently express the information-qubit matrix as follows

$$\left[\begin{array}{ccc|ccc} 0 & 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I \end{array}\right],$$

by multiplying the last $c-s$ rows of the stabilizer by $L^T\left(D^{-1}\right)$ and adding to the last $c-s$ rows of the information-qubit matrix.

The overall procedure for encoding is to begin with a set of c ebits, $2\left(n-c\right)-k_1-k_2$ ancilla qubits, and k_1+k_2-n+c information qubits. Alice performs the infinite-depth operations detailed in the paragraph with (5.12) for $c-s$ of the ebits. Alice then performs the finite-depth operations detailed in the proofs of this lemma and Lemma 5.2.1 in reverse order. The resulting stabilizer has equivalent error-correcting properties to the quantum check matrix in (5.2).

The receiver decodes by first performing all of the finite-depth operations in the encoding circuit in reverse order. The receiver then decodes the infinite-depth operations by the procedure listed in the paragraph with (5.13) so that the original k_1+k_2-n+c information qubits per frame are available for processing at the receiving end. □

5.4.1 Special Case with Coherent Teleportation Decoding

We now detail a special case of the above codes in this final section. These codes are interesting because the information qubits teleport coherently to other physical qubits when encoding and decoding is complete.

Lemma 5.4.2. *Suppose that the Smith form of $F\left(D\right)$ in (5.7) is*

$$F\left(D\right)=A_F\left(D\right)\left[\Gamma_F\left(D\right)\quad 0\right]B_F\left(D\right),$$

where $A_F\left(D\right)$ is $\left(n-k_1\right)\times\left(n-k_1\right)$, $\Gamma_F\left(D\right)$ is an $\left(n-k_1\right)\times\left(n-k_1\right)$ diagonal matrix whose entries are powers of D, and $B_F\left(D\right)$ is $k_2\times k_2$. Then the resulting entanglement-assisted code admits an encoding circuit with both infinite-depth and finite-depth operations and admits a decoding circuit with finite-depth operations only. The information qubits also teleport coherently to other physical qubits for this special case of codes.

Proof. We perform all the operations in Lemma 5.2.1 to obtain the quantum check matrix in (5.7). Then perform the row operations in $A_F^{-1}(D)$ and the column operations in $B_F^{-1}(D)$. The quantum check matrix becomes

$$\left[\begin{array}{ccc|ccc} E'(D) & \Gamma_F(D) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I & 0 & 0 \end{array}\right],$$

where $E'(D) = A_F^{-1}(D)\,E(D)$. The Smith form of $E'(D)$ is

$$E'(D) = A_{E'}(D) \begin{bmatrix} \Gamma_1(D) & 0 & 0 \\ 0 & \Gamma_2(D) & 0 \\ 0 & 0 & 0 \end{bmatrix} B_{E'}(D),$$

where $A_{E'}(D)$ is $(n-k_1)\times(n-k_1)$, $\Gamma_1(D)$ is an $s\times s$ diagonal matrix whose entries are powers of D, $\Gamma_2(D)$ is a $(c-s)\times(c-s)$ diagonal matrix whose entries are arbitrary polynomials, and $B_{E'}(D)$ is $(n-k_2)\times(n-k_2)$.

Now perform the row operations in $A_{E'}^{-1}(D)$ and the column operations in $B_{E'}^{-1}(D)$. It is possible to counteract the effect of the row operations on $\Gamma_F(D)$ by performing column operations, and it is possible to counteract the effect of the column operations on the identity matrix in the "X" matrix by performing row operations. The quantum check matrix becomes

$$\left[\begin{array}{ccccccc|ccc} \Gamma_1(D) & 0 & 0 & \Gamma_1'(D) & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \Gamma_2(D) & 0 & 0 & \Gamma_2'(D) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \Gamma_3'(D) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 \end{array}\right],$$

where $\Gamma_1'(D)$, $\Gamma_2'(D)$, and $\Gamma_3'(D)$ represent the respective $s\times s$, $(c-s)\times(c-s)$, and $(n-k_1-c)\times(n-k_1-c)$ diagonal matrices resulting from counteracting the effect of row operations $A_{E'}^{-1}(D)$

on $\Gamma_F(D)$. We use Hadamard and CNOT gates to clear the entries in $\Gamma_1'(D)$. The quantum check matrix becomes

$$\left[\begin{array}{ccccccc|ccc} \Gamma_1(D) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \Gamma_2(D) & 0 & 0 & \Gamma_2'(D) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \Gamma_3'(D) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 \end{array}\right].$$

The first s rows with $\Gamma_1(D)$ and rows $n-k_1-c+1,\ldots,n-k_1-c+s$ with the identity matrix on the "X" side stabilize a set of s ebits. The $n-k_1-c$ rows with $\Gamma_3'(D)$ and the $n-k_2-c$ rows with identity in the "X" matrix stabilize a set of $2(n-c)-k_1-k_2$ ancilla qubits (up to Hadamard gates). The s and k_2-n+k_1 columns with zeros in both the "Z" and "X" matrices correspond to information qubits. The decomposition of these rows is now complete.

We need to finish processing the $c-s$ rows with $\Gamma_2(D)$ and $\Gamma_2'(D)$ as entries and the $c-s$ rows of the identity in the "X" matrix. We construct a submatrix of the above quantum check matrix:

$$\left[\begin{array}{cc|cc} \Gamma_2(D) & \Gamma_2'(D) & 0 & 0 \\ 0 & 0 & I & 0 \end{array}\right]. \tag{5.14}$$

We describe a procedure to encode the above entries with $c-s$ ebits and $c-s$ information qubits using infinite-depth operations. Consider the following stabilizer matrix

$$\left[\begin{array}{ccc|ccc} I & I & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I & I & 0 \end{array}\right], \tag{5.15}$$

where all identity and null matrices are $(c-s)\times(c-s)$. The above matrix stabilizes a set of $c-s$ ebits and $c-s$ information qubits. Bob's half of the ebits are the $c-s$ columns on the left in both the "Z" and "X" matrices and Alice's half are the next $c-s$ columns. We also track the logical operators for the information qubits to verify that the circuit encodes and decodes properly. The information-qubit matrix is as follows

$$\left[\begin{array}{ccc|ccc} 0 & 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I \end{array}\right], \tag{5.16}$$

where all matrices are again $(c-s)\times(c-s)$. Alice performs Hadamard gates on her first $c-s$ qubits and then performs CNOT gates from her first $c-s$ qubits to her last $c-s$ qubits to transform (5.15) to the following stabilizer

$$\left[\begin{array}{ccc|ccc} I & 0 & 0 & 0 & I & \Gamma_2'(D) \\ 0 & I & 0 & I & 0 & 0 \end{array}\right].$$

The information-qubit matrix in (5.16) becomes

$$\left[\begin{array}{ccc|ccc} 0 & \Gamma_2'\left(D^{-1}\right) & I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I \end{array}\right].$$

Alice then performs infinite-depth operations on her last $c-s$ qubits. These infinite-depth operations correspond to the elements of $\Gamma_2^{-1}(D)$. She finally performs Hadamard gates on her $2(c-s)$ qubits. The stabilizer becomes

$$\left[\begin{array}{ccc|ccc} I & I & \Gamma_2^{-1}(D)\Gamma_2'(D) & 0 & 0 & 0 \\ 0 & 0 & 0 & I & I & 0 \end{array}\right], \tag{5.17}$$

and the information-qubit matrix becomes

$$\left[\begin{array}{ccc|ccc} 0 & 0 & 0 & 0 & \Gamma_2'\left(D^{-1}\right) & \Gamma_2\left(D^{-1}\right) \\ 0 & 0 & \Gamma_2^{-1}(D) & 0 & 0 & 0 \end{array}\right]. \tag{5.18}$$

The stabilizer in (5.17) is equivalent to the following stabilizer by row operations (premultiplying the first $c-s$ rows by $\Gamma_2(D)$):

$$\left[\begin{array}{ccc|ccc} \Gamma_2(D) & \Gamma_2(D) & \Gamma_2^{'}(D) & 0 & 0 & 0 \\ 0 & 0 & 0 & I & I & 0 \end{array}\right]. \tag{5.19}$$

The measurements that Bob performs have finite weight because the row operations are multiplications of the rows by the arbitrary polynomials in $\Gamma_2(D)$. Alice thus encodes a code equivalent to the desired quantum check matrix in (5.14) using $c-s$ ebits and $c-s$ information qubits.

We now discuss decoding the stabilizer in (5.17) and information qubits. Bob performs CNOTs from the first $c-s$ qubits to the next $c-s$ qubits. The stabilizer becomes

$$\left[\begin{array}{ccc|ccc} 0 & I & \Gamma_2^{-1}(D)\,\Gamma_2'(D) & 0 & 0 & 0 \\ 0 & 0 & 0 & I & 0 & 0 \end{array}\right], \tag{5.20}$$

and the information-qubit matrix does not change. Bob uses Hadamard and finite-depth CNOT gates to multiply the last $c-s$ columns in the "Z" matrix by $\Gamma_2^{'}\left(D^{-1}\right)\Gamma_2(D)$ and add the result to the middle $c-s$ columns. It is possible to use finite-depth operations because the entries of $\Gamma_2'(D)$ are all powers of D so that $\Gamma_2'\left(D^{-1}\right)=\Gamma_2^{'-1}(D)$. The stabilizer in (5.20) becomes

$$\left[\begin{array}{ccc|ccc} 0 & 0 & \Gamma_2^{-1}(D)\,\Gamma_2'(D) & 0 & 0 & 0 \\ 0 & 0 & 0 & I & 0 & 0 \end{array}\right],$$

and the information-qubit matrix in (5.18) becomes

$$\left[\begin{array}{ccc|ccc} 0 & 0 & 0 & 0 & \Gamma_2'\left(D^{-1}\right) & 0 \\ 0 & \Gamma_2'\left(D^{-1}\right) & \Gamma_2^{-1}(D) & 0 & 0 & 0 \end{array}\right].$$

We premultiply the first $c-s$ rows of the stabilizer by $\Gamma_2^{'}\left(D^{-1}\right)$ and add the result to the second $c-s$ rows of the information-qubit matrix. These row operations from the stabilizer to the information-qubit matrix result in the information-qubit matrix having pure logical operators for the middle $c-s$ qubits. Perform Hadamard gates on the second set of $c-s$ qubits. The resulting information-qubit matrix is as follows

$$\left[\begin{array}{ccc|ccc} 0 & \Gamma_2'\left(D^{-1}\right) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \Gamma_2'\left(D^{-1}\right) & 0 \end{array}\right], \tag{5.21}$$

so that the information qubits are available at the end of decoding. Processing may delay or advance them with respect to their initial locations because the matrix $\Gamma_2'\left(D^{-1}\right)$ is diagonal with powers of D. We can determine that the information qubits teleport coherently from the last set of $c-s$ qubits to the second set of $c-s$ qubits in every frame by comparing (5.21) to (5.16).

The overall procedure for encoding is to begin with a set of c ebits, $2(n-c)-k_1-k_2$ ancilla qubits, and k_1+k_2-n+c information qubits. Alice performs the infinite-depth operations detailed in (5.14-5.19) for $c-s$ of the ebits. Alice then performs the finite-depth operations detailed in the proofs of this lemma and Lemma 5.2.1 in reverse order. The resulting stabilizer has equivalent error-correcting properties to the quantum check matrix in (5.2).

The receiver decodes by first performing all of the finite-depth operations in reverse order. The receiver then decodes the infinite-depth operations by the procedure listed in (5.20-5.21) so that the original k_1+k_2-n+c information qubits per frame are available for processing at the receiving end. □

Example 5.4.1. Consider a classical convolutional code with the following check matrix:

$$H(D) = \left[\begin{array}{cc} 1 & 1+D \end{array} \right].$$

We can use the above check matrix in an entanglement-assisted quantum convolutional code to correct for both bit flips and phase flips. We form the following quantum check matrix:

$$\left[\begin{array}{cc|cc} 1 & 1+D & 0 & 0 \\ 0 & 0 & 1 & 1+D \end{array} \right]. \tag{5.22}$$

We first perform some manipulations to put the above quantum check matrix into a standard form. Perform a CNOT from qubit one to qubit two in the same frame and in the next frame. The above matrix becomes

$$\left[\begin{array}{cc|cc} D^{-1}+1+D & 1+D & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array} \right].$$

Perform a Hadamard gate on qubits one and two. The matrix becomes

$$\left[\begin{array}{cc|cc} 0 & 0 & D^{-1}+1+D & 1+D \\ 1 & 0 & 0 & 0 \end{array} \right].$$

Perform a CNOT from qubit one to qubit two. The matrix becomes

$$\left[\begin{array}{cc|cc} 0 & 0 & D^{-1}+1+D & D^{-1} \\ 1 & 0 & 0 & 0 \end{array}\right].$$

Perform a row operation that delays the first row by D. Perform a Hadamard on both qubits. The stabilizer becomes

$$\left[\begin{array}{cc|cc} 1+D+D^2 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right].$$

The above matrix is now in standard form. The matrix $F(D) = 1$ as in (5.7) so that its only invariant factor is equal to one. The code falls into the second class of entanglement-assisted quantum convolutional codes. We begin encoding with one ebit and one information qubit per frame. The stabilizer matrix for the unencoded stream is as follows:

$$\left[\begin{array}{ccc|ccc} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{array}\right],$$

and the information-qubit matrix is as follows:

$$\left[\begin{array}{ccc|ccc} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{array}\right].$$

Perform a Hadamard on qubit two and a CNOT from qubit two to qubit three so that the above stabilizer becomes

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \end{array}\right],$$

and the information-qubit matrix becomes

$$\left[\begin{array}{ccc|ccc} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{array}\right].$$

Perform an infinite-depth operation corresponding to the rational polynomial $1/\left(1+D+D^2\right)$ on qubit three. Follow with a Hadamard gate on qubits two and three. The stabilizer matrix becomes

$$\left[\begin{array}{ccc|ccc} 1 & 1 & 1/\left(1+D+D^2\right) & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{array}\right], \tag{5.23}$$

and the information-qubit matrix becomes

$$\left[\begin{array}{ccc|ccc} 0 & 0 & 1/\left(1+D+D^2\right) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1+D^{-1}+D^{-2} \end{array}\right]. \tag{5.24}$$

Perform the finite-depth operations above in reverse order so that the stabilizer becomes

$$\left[\begin{array}{ccc|ccc} D^{-1} & \frac{1}{1+D+D^2} & \frac{1+D}{1+D+D^2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1+D \end{array}\right],$$

and the information-qubit matrix becomes

$$\left[\begin{array}{ccc|ccc} 0 & \frac{D^{-1}+D^{-2}}{1+D+D^2} & \frac{1}{1+D+D^2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & D^{-1}+D^{-2} & D^{-1} \end{array}\right].$$

The above stabilizer is equivalent to the desired quantum check matrix in (5.22) by a row operation that multiplies its first row by $1+D+D^2$.

The receiver decodes by performing the finite-depth encoding operations in reverse order and gets the stabilizer in (5.23) and the information-qubit matrix in (5.24). The receiver performs a CNOT from qubit one to qubit two and follows with a CNOT from qubit two to qubit three in the same frame, in an advanced frame, and in a twice-advanced frame. Finally perform a Hadamard gate on qubits two and three. The stabilizer becomes

$$\left[\begin{array}{ccc|ccc} 0 & 0 & 0 & 0 & 0 & 1/\left(1+D+D^2\right) \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right],$$

and the information-qubit matrix becomes

$$\left[\begin{array}{ccc|cc|c} 0 & 0 & 0 & 0 & 1 & 1/\left(1+D+D^2\right) \\ 0 & 1 & 0 & 0 & 0 & 0 \end{array}\right].$$

The receiver decodes the information qubits successfully because a row operation from the first row of the stabilizer to the first row of the information-qubit matrix gives the proper logical operators for the information qubits. Figure 7 of Ref. [52] details the above encoding and decoding operations for this entanglement-assisted quantum convolutional code.

5.4.2 Discussion

This second class of codes assumes that noiseless encoding is available. We require this assumption because the encoding circuit employs infinite-depth encoding operations.

If an error does occur during the encoding process, it can propagate infinitely through the encoded qubit stream. The result of a single encoding error can distort both the encoded quantum information, the syndromes that result from measurements, and the final recovery operations based on the syndromes.

We may be able to relax the noiseless encoding assumption if nearly noiseless encoding is available. The probability of an error would have to be negligible in order to ensure that the probability for a catastrophic failure is negligible. One way to lower the probability of an encoding error is to encode first with a quantum block code and then further encode with our quantum convolutional coding method. Many classical coding systems exploit this technique, the most popular of which is a Reed-Solomon encoder followed by a convolutional encoder.

5.5 Closing Remarks

This chapter develops the theory of entanglement-assisted quantum convolutional coding for CSS codes. We show several methods for importing two arbitrary classical binary convolutional codes for use in an entanglement-assisted quantum convolutional code. Our methods outline different ways for encoding and decoding our entanglement-assisted quantum convolutional codes.

Our first class of codes employs only finite-depth operations in their encoding and decoding procedures. These codes are the most useful in practice because they do not have the risk of catastrophic error propagation. An error that occurs during encoding, measurement, recovery, or decoding propagates only to a finite number of neighboring qubits.

Our second class of codes uses infinite-depth operations during encoding. This assumption is reasonable only if noiseless encoding is available. The method of concatenated coding is one way to approach nearly noiseless encoding in practice.

We suggest several lines of inquiry from here. Our codes are not only useful for quantum communication, but also should be useful for private classical communication because of the well-known connection between a quantum channel and private classical channel [39]. It may make sense from a practical standpoint to begin investigating the performance of our codes for encoding secret classical messages. The commercial success of quantum key distribution [12] for the generation of a private shared secret key motivates this investigation. It is also interesting to determine which entanglement-assisted codes can correct for errors on the receiver's side. Codes that possess this property will be more useful in practice.

We hope that our theory of entanglement-assisted quantum convolutional coding provides a step in the direction of finding quantum codes that approach the ultimate capacity of an entanglement-assisted quantum channel.

Chapter 6

Entanglement-Assisted Quantum Convolutional Coding: The General Case

These patterns that anticommute,
They laugh and they think it's so cute,
Expand until fit,
Then ole Gram and Schmidt,
They'll line up with a grand salute.

In this chapter, we show how to encode and decode an entanglement-assisted quantum convolutional code that does not necessarily possess the CSS structure. The methods in this chapter represent a significant extension of our work on CSS entanglement-assisted quantum convolutional codes in the previous chapter.

In particular, we develop a set of techniques that make the commutation relations for the generators of a general code the same as those of entangled qubits (ebits) and ancilla qubits. This procedure first "expands" the check matrix for a quantum convolutional code and applies an extended version of the symplectic Gram-Schmidt orthogonalization procedure [35] that incorporates the binary polynomials representing quantum convolutional codes. We then show how to encode a stream of information qubits, ancilla qubits, and ebits to have the error-correcting properties of the desired code. We follow by detailing the decoding circuit that Bob employs at the receiving end of the channel. The algorithms for encoding and decoding use similar techniques

to those outlined in the previous chapter. The encoding circuits incorporate both finite-depth and infinite-depth operations as discussed in respective Sections 4.5 and 4.6 and the decoding circuits incorporate finite-depth operations only.

One benefit of the techniques developed in this chapter is that the quantum code designer can produce an entanglement-assisted quantum convolutional code from a classical quaternary convolutional code. More generally, quantum convolutional code designers now have the freedom to design quantum convolutional codes to have desirable error-correcting properties without having to search for codes that satisfy a restrictive commutativity constraint.

We structure this chapter as follows. In Section 6.1, we present an example that demonstrates how to expand the check matrix of a set of quantum convolutional generators and then show how to expand an arbitrary check matrix. We show in Section 6.2 how to compute the symplectic Gram-Schmidt orthogonalization procedure for the example in Section 6.1 and then generalize this procedure to work for an arbitrary set of quantum convolutional generators. The Gram-Schmidt orthogonalization technique reduces the quantum check matrix to have the same commutation relations as a set of ebits and ancilla qubits. This technique is essential for determining an encoding circuit that encodes a stream of information qubits, ancilla qubits, and ebits. Section 6.3 gives the algorithms for computing the encoding and decoding circuits for an arbitrary entanglement-assisted quantum convolutional code. We present a detailed example in Section 6.4 that illustrates all of the procedures in this chapter. We finish with a discussion of the practical issues involved in using these entanglement-assisted quantum convolutional codes and present some concluding remarks.

The focus of the next two sections is to elucidate techniques for reducing a set of quantum convolutional generators to have the standard commutation relations in (3.2), or equivalently, the standard symplectic relations in (3.7). The difference between the current techniques and the former techniques [35, 34, 51, 78] is that the current techniques operate on generators that have a convolutional form rather than a block form. Section 6.1 shows how to expand a set of quantum convolutional generators to simplify their commutation relations. Section 6.2 outlines a symplectic Gram-Schmidt orthogonalization algorithm that uses binary polynomial operations to reduce the commutation relations of the expanded generators to have the standard form in (3.7).

6.1 The Expansion of Quantum Convolutional Generators

We begin this section by demonstrating how to expand a particular generator that we eventually incorporate in an entanglement-assisted quantum convolutional code. We later generalize this example and the expansion technique to an arbitrary set of generators. This technique is important for determining how to utilize entanglement in the form of ebits in a quantum convolutional code.

6.1.1 Example of the Expansion

Let us first suppose that we have one convolutional generator:

$$\cdots \left| \; I \; \right| \; X \; \left| \; Z \; \right| \; I \; \left| \cdots \right. \tag{6.1}$$

This generator does not yet represent a valid quantum convolutional code because it anticommutes with a shift of itself to the left or to the right by one qubit. We are not concerned with the error-correcting properties of this generator but merely want to illustrate the technique of expanding it.

Let us for now consider a block version of this code that operates on eight physical qubits with six total generators. The generators are as follows:

$$\begin{array}{c|c|c|c|c|c|c|c} X & Z & I & I & I & I & I & I \\ I & X & Z & I & I & I & I & I \\ I & I & X & Z & I & I & I & I \\ I & I & I & X & Z & I & I & I \\ I & I & I & I & X & Z & I & I \\ I & I & I & I & I & X & Z & I \end{array} \tag{6.2}$$

We still use the vertical bars in the above block code to denote that the frame size of the code is one. Observe that we can view the frame size of the code as two without changing any of the error-correcting properties of the block code:

$$\left.\begin{array}{cc|cc|cc|cc} X & Z & I & I & I & I & I & I \\ I & X & Z & I & I & I & I & I \\ I & I & X & Z & I & I & I & I \\ I & I & I & X & Z & I & I & I \\ I & I & I & I & X & Z & I & I \\ I & I & I & I & I & X & Z & I \end{array}\right. . \tag{6.3}$$

The frame is merely a way to organize our qubits so that we send one frame at a time over the channel after the online encoding circuit has finished processing them. We can extend the above block code with frame size two to have a convolutional structure with the following two convolutional generators:

$$\cdots \left|\begin{array}{cc|cc|cc|cc} I & I & X & Z & I & I & I & I \\ I & I & I & X & Z & I & I & I \end{array}\right| \cdots$$

The above two generators with frame size two have equivalent error-correcting properties to the original generator in (6.1) by the arguments above. We say that we have expanded the original generator by a factor of two or that the above code is a two-expanded version of the original generator. We can also extend the original generator in (6.1) to have a frame size of three and we require three convolutional generators so that they have equivalent error-correcting properties to the original generator:

$$\cdots \left|\begin{array}{ccc|ccc|ccc|ccc} I & I & I & X & Z & I & I & I & I & I & I & I \\ I & I & I & I & X & Z & I & I & I & I & I & I \\ I & I & I & I & I & X & Z & I & I & I & I & I \end{array}\right| \cdots$$

The representation of the original generator in (6.1) as a quantum check matrix in the polynomial formalism is as follows:

$$g\left(D\right) = \left[\begin{array}{c|c} D & 1 \end{array}\right]. \tag{6.4}$$

The two-expanded check matrix is as follows:

$$G_2\left(D\right) = \left[\begin{array}{cc|cc} 0 & 1 & 1 & 0 \\ D & 0 & 0 & 1 \end{array}\right], \tag{6.5}$$

and the three-expanded check matrix is as follows:

$$G_3\left(D\right) = \left[\begin{array}{ccc|ccc} 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ D & 0 & 0 & 0 & 0 & 1 \end{array}\right].$$

An alternative method for obtaining the polynomial representation of the two-expanded check matrix consists of two steps. We first multiply $g\left(D\right)$ as follows:

$$G_2'\left(D\right) = \left[\begin{array}{c} 1 \\ D \end{array}\right] \left[g\left(D\right)\right] \left[\begin{array}{cccc} 1 & D^{-1} & 0 & 0 \\ 0 & 0 & 1 & D^{-1} \end{array}\right].$$

We then "plug in" the fractional delay operator $D^{1/2}$ and apply the flooring operation $\lfloor \cdot \rfloor$ that nulls the coefficients of any fractional power of D:

$$G_2\left(D\right) = \left\lfloor G_2'\left(D^{1/2}\right)\right\rfloor.$$

A similar technique applies to find the check matrix of the three-expanded matrix. We first multiply $g\left(D\right)$ as follows:

$$G_3\left(D\right) = \left[\begin{array}{c} 1 \\ D \\ D^2 \end{array}\right] \left[g\left(D\right)\right] \left[\begin{array}{cccccc} 1 & \frac{1}{D} & \frac{1}{D^2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \frac{1}{D} & \frac{1}{D^2} \end{array}\right]$$

We then "plug in" the fractional delay operator $D^{1/3}$ and apply the flooring operation $\lfloor \cdot \rfloor$:

$$G_3(D) = \left\lfloor G_3'\left(D^{1/3}\right)\right\rfloor$$

We discuss the general method for expanding an arbitrary check matrix in the next subsection.

6.1.2 General Technique for Expansion

We generalize the above example to determine how to expand an arbitrary quantum check matrix by a factor of l. Suppose that we have an $n-k \times 2n$ quantum check matrix $H(D)$ where

$$H(D) = \left[\begin{array}{c|c} Z(D) & X(D) \end{array}\right].$$

Let $\mathbf{D}$ denote a diagonal matrix whose diagonal entries are the delay operator D. We take the convention that $\mathbf{D}^0$ is the identity matrix and $\mathbf{D}^m$ is the matrix $\mathbf{D}$ multiplied m times so that its diagonal entries are D^m. Let $R_l(D)$ and $C_l(D)$ denote the following matrices:

$$R_l(D) \equiv \left[\begin{array}{cccc} \mathbf{D}^0 & \mathbf{D}^1 & \cdots & \mathbf{D}^{l-1} \end{array}\right]^T$$

$$C_l(D) \equiv \left[\begin{array}{cccccc} \mathbf{D}^0 & \cdots & \mathbf{D}^{-(l-1)} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \mathbf{D}^0 & \cdots & \mathbf{D}^{-(l-1)} \end{array}\right]$$

where the diagonal $\mathbf{D}$ matrices in $R_l(D)$ and $C_l(D)$ have respective dimensions $n-k \times n-k$ and $n \times n$. We premultiply and postmultiply the matrix $H(D)$ by respective matrices $R_l(D)$ and $C_l(D)$, evaluate the resulting matrix at a fractional power $1/l$ of the delay operator D, and perform the flooring operation $\lfloor \cdot \rfloor$ to null the coefficients of any fractional power of D. The l-expanded check matrix $H_l(D)$ is as follows:

$$H_l(D) = \left\lfloor R_l\left(D^{1/l}\right) H\left(D^{1/l}\right) C_l\left(D^{1/l}\right)\right\rfloor.$$

The l-expanded quantum check matrix $H_l(D)$ has equivalent error-correcting properties to the original check matrix.

6.2 Polynomial Symplectic Gram-Schmidt Orthogonalization Procedure

In general, a given set of generators may have complicated commutation relations. We have to simplify the commutation relations so that we can encode information qubits with the help of ancilla qubits and halves of ebits shared with the receiver. In this section, we begin with an arbitrary set of convolutional generators. We show how to determine a set of generators with equivalent error-correcting properties and commutation relations that are the same as those of halves of ebits and ancilla qubits. We first show an example of the technique by illustrating it for Pauli sequences, for the quantum check matrix, and with the shifted symplectic product matrix. We then state a polynomial symplectic Gram-Schmidt orthogonalization algorithm that performs this action for an arbitrary set of quantum convolutional generators.

6.2.1 Example of the Procedure

Pauli Picture

Let us consider again our example from the previous section. Specifically, consider the respective expressions in (6.1) and (6.3) for the convolutional generator and the block code. Recall that we can multiply the generators in a block code without changing the error-correcting properties of the code [32]. Therefore, we can multiply the sixth generator in (6.3) to the fourth. We can then multiply the modified fourth to the second to yield the following equivalent code:

$$\left[\begin{array}{cc|cc|cc|cc} X & Z & I & I & I & I & I & I \\ I & X & Z & X & Z & X & Z & I \\ I & I & X & Z & I & I & I & I \\ I & I & I & X & Z & X & Z & I \\ I & I & I & I & X & Z & I & I \\ I & I & I & I & I & X & Z & I \end{array}\right] \tag{6.6}$$

We have manipulated the two-expanded matrix rather than the code in (6.2) because the commutation relations of the above code are equivalent to the commutation relations of the following operators:

$$\left[\begin{array}{cc|cc|cc|cc} I & Z & I & I & I & I & I & I \\ I & X & I & I & I & I & I & I \\ I & I & I & Z & I & I & I & I \\ I & I & I & X & I & I & I & I \\ I & I & I & I & I & Z & I & I \\ I & I & I & I & I & X & I & I \end{array}\right].$$

We can use three ebits to encode the set of generators in (6.6) because they have the same commutation relations as the above operators and the above operators correspond to halves of three ebits. We resolve the anticommutation relations by using the following entanglement-assisted code:

$$\left[\begin{array}{ccc|ccc|ccc|ccc} Z & X & Z & I & I & I & I & I & I & I & I & I \\ X & I & X & I & Z & X & I & Z & X & I & Z & I \\ I & I & I & Z & X & Z & I & I & I & I & I & I \\ I & I & I & X & I & X & I & Z & X & I & Z & I \\ I & I & I & I & I & I & Z & X & Z & I & I & I \\ I & I & I & I & I & I & X & I & X & I & Z & I \end{array}\right].$$

The convention above is that the first qubit of each frame belongs to Bob and corresponds to half of an ebit. The second two qubits of each frame belong to Alice. The overall code forms a commuting stabilizer so that it corresponds to a valid quantum code. Bob could measure the

above operators to diagnose errors or he could measure the following operators that are equivalent by row operations:

$$\left[\begin{array}{ccc|ccc|ccc|ccc} Z & X & Z & I & I & I & I & I & I & I & I & I \\ X & I & X & X & Z & I & I & I & I & I & I & I \\ I & I & I & Z & X & Z & I & I & I & I & I & I \\ I & I & I & X & I & X & I & Z & I & I & I & I \\ I & I & I & I & I & I & Z & X & Z & I & I & I \\ I & I & I & I & I & I & X & I & X & I & Z & I \end{array}\right]$$

One can check that the operators corresponding to the second two qubits of each frame are equivalent to the desired generators in (6.3).

Polynomial Picture

Let us consider the convolutional generator in (6.1). We now use the polynomial formalism because it is easier to perform the manipulations for general codes in this picture rather than in the Pauli picture.

We extend the row operations from the above block code to the polynomial picture. Each row operation multiplied the even-numbered generators by themselves shifted by two qubits. Extending this operation to act on an infinite Pauli sequence corresponds to multiplying the second generator in (6.5) by the rational polynomial $1/(1+D)$. Consider the two-expanded check matrix with the operation described above applied to the second generator:

$$\left[\begin{array}{c} g_1(D) \\ g_2(D) \end{array}\right] = \left[\begin{array}{cc|cc} 0 & 1 & 1 & 0 \\ \frac{D}{1+D} & 0 & 0 & \frac{1}{1+D} \end{array}\right],$$

The shifted symplectic products $(g_1 \odot g_1)(D) = 0$, $(g_1 \odot g_2)(D) = 1$, and $(g_2 \odot g_2)(D) = 0$ capture the commutation relations of the resulting code for all frames. We can therefore resolve this anticommutativity by prepending a column that corresponds to Bob possessing half of an ebit:

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & \frac{D}{1+D} & 0 & 1 & 0 & \frac{1}{1+D} \end{array}\right].$$

Bob possesses the qubit corresponding to column one of both the "Z" and "X" matrix and Alice possesses the two qubits corresponding to the second and third columns. The code above has equivalent error-correcting properties to those of the desired generators.

The generators in the above polynomial setting correspond to Pauli sequences with infinite weight. Infinite-weight generators are undesirable because Bob cannot measure an infinite number of qubits. There is a simple solution to this problem and it is similar to what we did at the end of the previous subsection. We multiply the second row of the above check matrix by $1 + D$ to obtain the following check matrix that has equivalent error-correcting properties to the above one:

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & D & 0 & 1+D & 0 & 1 \end{array}\right].$$

Bob can measure operators that have finite weight because the above check matrix corresponds to Pauli sequences with finite weight.

Shifted Symplectic Product Matrix Picture

We can also determine the row operations that simplify the commutation relations by looking only at the shifted symplectic product matrix. It is easier to determine the row operations that resolve anticommutativity by considering the shifted symplectic product matrix. We would like to perform row operations to reduce the shifted symplectic product matrix to the standard form in (3.7) so that it has commutation relations equivalent to those of halves of ebits and ancilla qubits.

The shifted symplectic product matrix corresponding to the check matrix in (6.4) is a one-element matrix $\Omega_g(D)$ where

$$\Omega_g(D) = \left[D^{-1} + D\right].$$

It is clear that $\Omega_g(D)$ is not reducible by row operations to the 2×2 matrix J in (3.5) because $\Omega_g(D)$ is a one-element matrix. It is also not reducible to the one-element null matrix $[0]$ because there is no valid row operation that can zero out the term $D^{-1} + D$.

We therefore consider the two-expanded check matrix in (6.5) to determine if we can reduce it with row operations to the standard form in (3.7). The shifted symplectic product matrix $\Omega_{G_2}(D)$ for the two-expanded check matrix $G_2(D)$ is as follows:

$$\Omega_{G_2}(D) = \begin{bmatrix} 0 & 1+D^{-1} \\ 1+D & 0 \end{bmatrix}$$

We can formulate the row operation of multiplying the second generator by $1/(1+D)$ as a matrix $R(D)$ where

$$R(D) = \begin{bmatrix} 1 & 0 \\ 0 & 1/(1+D) \end{bmatrix}.$$

The effect on the matrix $\Omega_{G_2}(D)$ is to change it to

$$R(D)\,\Omega_{G_2}(D)\,R^T\left(D^{-1}\right) = J$$

as described in (4.39). The above matrix J has equivalent commutation relations to half of an ebit. We can therefore use one ebit per two qubits to encode this code.

In a later section, we show how to devise encoding circuits beginning from c ebits per frame and a ancilla qubits per frame and also give the algorithm for their online decoding circuits. It is important for us right now to devise a procedure to reduce the commutation relations of any check matrix to those of ebits and ancilla qubits.

6.2.2 The Procedure for General Codes

We detail a polynomial version of the symplectic Gram-Schmidt orthogonalization procedure in this section. It is a generalized version of the procedure we developed for the above example. Before detailing the algorithm, we first prove a lemma that shows how to determine the shifted symplectic product matrix for an l-expanded version of the generators by starting from the shifted symplectic product matrix of the original generators.

The Shifted Symplectic Product Matrix for an l-Expanded Code

Lemma 6.2.1. *Suppose the shifted symplectic product matrix $\Omega(D)$ of a given check matrix $H(D)$ is as follows:*

$$\Omega(D) = Z(D) X^T\left(D^{-1}\right) + X(D) Z^T\left(D^{-1}\right).$$

The shifted symplectic product matrix $\Omega_l(D)$ of the l-expanded check matrix $H_l(D)$ is as follows:

$$\Omega_l(D) = \left\lfloor R_l\left(D^{1/l}\right) \Omega\left(D^{1/l}\right) R_l^T\left(D^{-1/l}\right)\right\rfloor$$

where the flooring operation $\lfloor\cdot\rfloor$ nulls the coefficients of any fractional power of D.

Proof. Consider that the "X" matrix $X_l(D)$ of the l-expanded check matrix $H_l(D)$ is as follows:

$$X_l(D) = \left\lfloor R_l\left(D^{1/l}\right) X\left(D^{1/l}\right) C_l'\left(D^{1/l}\right)\right\rfloor,$$

where

$$C_l'\left(D^{1/l}\right) \equiv \left[\begin{array}{cccc} \mathbf{D}^0 & \mathbf{D}^{-1} & \cdots & \mathbf{D}^{-(l-1)} \end{array}\right],$$

and each diagonal $\mathbf{D}$ matrix is $n \times n$-dimensional. It is also then true that the "Z" matrix of $H_l(D)$ is as follows:

$$Z_l(D) = \left\lfloor R_l\left(D^{1/l}\right) Z\left(D^{1/l}\right) C_l'\left(D^{1/l}\right)\right\rfloor.$$

The matrix transpose operation, the time reversal operation (substituting D^{-1} for D), and matrix addition are all invariant under the flooring operation $\lfloor\cdot\rfloor$ for arbitrary matrices $M(D)$ and $N(D)$:

$$\begin{aligned} \left\lfloor M^T\left(D^{1/l}\right)\right\rfloor &= \left\lfloor M\left(D^{1/l}\right)\right\rfloor^T, \\ \left\lfloor M\left(D^{-1/l}\right)\right\rfloor &= \left\lfloor M\left(D^{1/l}\right)\right\rfloor\Big|_{D=D^{-1}}, \\ \left\lfloor M\left(D^{1/l}\right) + N\left(D^{1/l}\right)\right\rfloor &= \left\lfloor M\left(D^{1/l}\right)\right\rfloor + \left\lfloor N\left(D^{1/l}\right)\right\rfloor. \end{aligned}$$

Additionally, the following property holds for two arbitrary binary polynomials $f(D)$ and $g(D)$:

$$\left\lfloor f\left(D^{1/l}\right) g\left(D^{1/l}\right)\right\rfloor = \sum_{i=0}^{l-1} \left\lfloor D^{-i/l} f\left(D^{1/l}\right)\right\rfloor \left\lfloor D^{i/l} g\left(D^{1/l}\right)\right\rfloor. \tag{6.7}$$

Now consider the product $X_l(D) Z_l^T(D^{-1})$:

$$\begin{aligned}
X_l(D) Z_l^T(D^{-1}) &= \left\lfloor R_l\left(D^{1/l}\right) X\left(D^{1/l}\right) C_l'\left(D^{1/l}\right) \right\rfloor \left(\left\lfloor R_l\left(D^{1/l}\right) Z\left(D^{1/l}\right) C_l'\left(D^{1/l}\right) \right\rfloor \right)^T \Big|_{D=D^{-1}} \\
&= \left\lfloor R_l\left(D^{1/l}\right) X\left(D^{1/l}\right) C_l'\left(D^{1/l}\right) \right\rfloor \left\lfloor C_l'^T\left(D^{-1/l}\right) Z^T\left(D^{-1/l}\right) R_l^T\left(D^{-1/l}\right) \right\rfloor \\
&= \sum_{i=0}^{l-1} \left\lfloor D^{-i/l} R_l\left(D^{1/l}\right) X\left(D^{1/l}\right) \right\rfloor \left\lfloor D^{i/l} Z^T\left(D^{-1/l}\right) R_l^T\left(D^{-1/l}\right) \right\rfloor \\
&= \left\lfloor R_l\left(D^{1/l}\right) X\left(D^{1/l}\right) Z^T\left(D^{-1/l}\right) R_l^T\left(D^{-1/l}\right) \right\rfloor,
\end{aligned}$$

where the second line uses the invariance of the flooring operation with respect to matrix transposition and time reversal, the third line expands the matrix multiplications using the matrix $C_l'(D)$ defined above, and the last line uses the matrix generalization of the multiplication property defined in (6.7). Our final step is to use the invariance of the flooring operation with respect to matrix addition:

$$\begin{aligned}
\Omega_l(D) &= X_l(D) Z_l^T(D^{-1}) + Z_l(D) X_l^T(D^{-1}) \\
&= \left\lfloor R_l\left(D^{\frac{1}{l}}\right) X\left(D^{\frac{1}{l}}\right) Z^T\left(D^{\frac{-1}{l}}\right) R_l^T\left(D^{\frac{-1}{l}}\right) \right\rfloor + \left\lfloor R_l\left(D^{\frac{1}{l}}\right) Z\left(D^{\frac{1}{l}}\right) X^T\left(D^{\frac{-1}{l}}\right) R_l^T\left(D^{\frac{-1}{l}}\right) \right\rfloor \\
&= \left\lfloor R_l\left(D^{\frac{1}{l}}\right) X\left(D^{\frac{1}{l}}\right) Z^T\left(D^{\frac{-1}{l}}\right) R_l^T\left(D^{\frac{-1}{l}}\right) + R_l\left(D^{\frac{1}{l}}\right) Z\left(D^{\frac{1}{l}}\right) X^T\left(D^{\frac{-1}{l}}\right) R_l^T\left(D^{\frac{-1}{l}}\right) \right\rfloor \\
&= \left\lfloor R_l\left(D^{1/l}\right) \Omega\left(D^{1/l}\right) R_l^T\left(D^{-1/l}\right) \right\rfloor.
\end{aligned}$$

□

The Gram-Schmidt Procedure

We now present the polynomial symplectic Gram-Schmidt orthogonalization procedure that reduces the commutation relations of a given set of convolutional generators to have the standard form in (3.7).

Consider the following $n-k \times 2n$-dimensional quantum check matrix $H(D)$:

$$H(D) = \left[\begin{array}{c|c} Z(D) & X(D) \end{array} \right].$$

Label each row as $h_i(D) = \left[\begin{array}{c|c} z_i(D) & x_i(D) \end{array} \right]$ for all $i \in \{1, \ldots, n-k\}$.

We state the Gram-Schmidt procedure in terms of its effect on the shifted symplectic product matrix. It is easier to see how the algorithm proceeds by observing the shifted symplectic product matrix rather than by tracking the generators in the check matrix.

Let l denote the amount by which we expand the check matrix $H(D)$. Suppose first that $l = 1$ (we do not expand the check matrix). Let us say that the check matrix has r generators (we take a number different from $n-k$ because the number of generators may not be equal to $n-k$ for future iterations). There are the following possibilities:

(i). There is a generator $h_i(D)$ such that $(h_i \odot h_j)(D) = 0$ for all $j \in \{1, \ldots, r\}$. In this case, the generator is already decoupled from all the others and corresponds to an ancilla qubit because an ancilla and it share the same commutation relations. Swap $h_i(D)$ to be the first row of the matrix so that it is $h_1(D)$. The shifted symplectic product matrix then has the form:

$$\begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & h_{2,2} & \cdots & h_{2,r} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & h_{r,2} & \cdots & h_{r,r} \end{bmatrix} = \begin{bmatrix} 0 \end{bmatrix} \oplus \begin{bmatrix} h_{2,2} & \cdots & h_{2,r} \\ \vdots & \ddots & \vdots \\ h_{r,2} & \cdots & h_{r,r} \end{bmatrix}$$

where $[0]$ is the one-element zero matrix and we use the shorthand $h_{i,j} = (h_i \odot h_j)(D)$. We remove generator $h_1(D)$ from check matrix $H(D)$ and continue to step two below for the remaining generators in the matrix.

(ii). There are two generators $h_i(D)$ and $h_j(D)$ for which $(h_i \odot h_j)(D) = D^m$ for some integer m and $(h_i \odot h_i)(D) = (h_j \odot h_j)(D) = 0$. In this case these generators correspond exactly to half of an ebit. Multiply generator $h_j(D)$ by D^m. This row operation has the effect of delaying (or advancing) the generator by an amount m and changes the shifted symplectic product to be $(h_i \odot h_j)(D) = 1$. These two generators considered by themselves now have the commutation relations of half of an ebit. Swap the generators $h_i(D)$ and $h_j(D)$ to be

the first and second respective rows of the check matrix $H(D)$. Call them $h_1(D)$ and $h_2(D)$ respectively. The shifted symplectic product matrix is then as follows:

$$\begin{bmatrix} 0 & 1 & h_{1,3} & \cdots & h_{1,r} \\ 1 & 0 & h_{2,3} & \cdots & h_{2,r} \\ h_{3,1} & h_{3,2} & h_{3,3} & \cdots & h_{3,r} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{r,2} & h_{r,2} & h_{r,3} & \cdots & h_{r,r} \end{bmatrix}.$$

We use the following row operations to decouple the other generators from these two generators:

$$h'_i(D) \equiv h_i(D) + [(h_i \odot h_2)(D)]\, h_1(D) + [(h_i \odot h_1)(D)]\, h_2(D) \text{ for all } i \in \{3, \ldots, r\}.$$

The shifted symplectic product matrix becomes as follows under these row operations:

$$\begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & h'_{3,3} & \cdots & h'_{3,r} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & h'_{r,3} & \cdots & h'_{r,r} \end{bmatrix} = \begin{bmatrix} J \end{bmatrix} \oplus \begin{bmatrix} h'_{3,3} & \cdots & h'_{3,r} \\ \vdots & \ddots & \vdots \\ h'_{r,3} & \cdots & h'_{r,r} \end{bmatrix}.$$

The first two generators are now decoupled from the other generators and have the commutation relations of half of an ebit. We remove the first two generators from the check matrix so that it now consists of generators $h'_3(D)$, ..., $h'_r(D)$. We check to see if the conditions at the beginning of the previous step or this step hold for any other generators. If so, repeat the previous step or this step on the remaining generators. If not, see if the conditions for step three hold.

(iii). There are two generators $h_i(D)$ and $h_j(D)$ for which $(h_i \odot h_i)(D) = (h_j \odot h_j)(D) = 0$ but $(h_i \odot h_j)(D) \neq D^m$ for all m and $(h_i \odot h_j)(D) \neq 0$. Multiply generator $h_j(D)$ by

$1/\left(h_j \odot h_i\right)(D)$. Generator $h_j\left(D\right)$ becomes infinite weight because $\left(h_j \odot h_i\right)(D)$ is a polynomial with two or more powers of D with non-zero coefficients. Now the shifted symplectic product relations are as follows: $\left(h_i \odot h_i\right)(D) = \left(h_j \odot h_j\right)(D) = 0$ and $\left(h_i \odot h_j\right)(D) = 1$. We handle this case as we did the previous case after the two generators there had the commutation relations of half of an ebit.

(iv). None of these conditions hold. In this case, we stop this iteration of the algorithm and expand the check matrix by the next factor $l := l + 1$ and repeat the above steps.

We have not proven that this procedure converges on all codes; however, it does converge on all the codes we have tried. We conjecture that this procedure converges for all codes, but even if this is true, in principle it might require expansion to a large number of generators. A simple and practical convergence condition is as follows. In practice, convolutional codes do not act on an infinite stream of qubits but instead act on a finite number of qubits. It may be that there are codes for which this procedure either does not converge or must be expanded until the frame size of the expanded code exceeds the number of qubits that the code acts on. In this case, we would not employ this procedure, and instead would treat the code as a block code, where we could employ the methods from Refs. [34, 35] for encoding and decoding. It is unclear if this practical convergence condition will ever really be necessary. This procedure does converge for a large number of useful codes, so that the frame size of the expanded code is much less than the number of qubits that the code acts on, and we have not found an example where this procedure fails.

6.3 Encoding and Decoding Circuits

This section proves the main theorem of this chapter. The theorem assumes that we have already processed an arbitrary check matrix with the Gram-Schmidt algorithm and that the shifted symplectic product matrix corresponding to the processed check matrix has the standard form in (3.7). The theorem shows how to encode a set of information qubits, ancilla qubits, and halves of ebits into a code that has equivalent error-correcting properties to those of a desired set of convolutional generators. The theorem uses both finite-depth and infinite-depth operations in the encoding circuit and finite-depth operations in the decoding circuit.

Theorem 6.3.1. *Suppose we have a set of quantum convolutional generators in the $n-k \times 2n$-dimensional matrix $H(D)$ where*

$$H(D) = \left[\begin{array}{c|c} Z(D) & X(D) \end{array} \right].$$

Its shifted symplectic product matrix $\Omega(D)$ is as follows:

$$\Omega(D) = Z(D) X^T\left(D^{-1}\right) + X(D) Z^T\left(D^{-1}\right).$$

Suppose check matrix $H(D)$ is the matrix resulting from processing with the polynomial symplectic Gram-Schmidt orthogonalization procedure. Therefore, $\Omega(D)$ has the standard form in (3.7) with parameters c and $a = n-k-2c$. Then there exists an online encoding circuit for the code that uses finite-depth and infinite-depth operations in the shift-invariant Clifford group and there exists an online decoding circuit for the code that uses finite-depth operations. The code encodes $k+c$ information qubits per frame with the help of c ebits and $a = n-k-2c$ ancilla qubits.

Proof. We prove the theorem by giving an algorithm to compute both the encoding circuit and the decoding circuit. The shifted symplectic product matrix $\Omega(D)$ for check matrix $H(D)$ is in standard form so that the first $2c$ rows have the commutation relations of c halves of ebits and the last a rows have the commutation relations of a ancilla qubits. Perform the algorithm outlined in Refs. [46, 45] on the last a generators that correspond to the ancilla qubits. The algorithm uses finite-depth CNOT gates, Hadamard gates, and phase gates. The resulting check matrix has the following form:

$$\left[\begin{array}{cc|cc} Z''(D) & Z'(D) & 0 & X'(D) \\ \Gamma(D) & 0 & 0 & 0 \end{array} \right], \tag{6.8}$$

where the matrix $Z''(D)$ and the null matrix at the top left of the "X" matrix each have dimension $2c \times a$, the matrices $Z'(D)$ and $X'(D)$ each have dimension $2c \times n-a$, and all matrices in the second set of rows each have a rows. The matrix $\Gamma(D)$ may have entries that are rational polynomials. If so, replace each of these entries with a "1" so that the resulting matrix has the following form:

$$\left[\begin{array}{cc|cc} Z''(D) & Z'(D) & 0 & X'(D) \\ I & 0 & 0 & 0 \end{array} \right].$$

This replacement is equivalent to taking a subcode of the original that has equivalent error-correcting properties and rate [45]. We can also think of the replacement merely as row operations with rational polynomials. We then perform row operations from the last a rows to the first $2c$ rows to obtain the following check matrix:

$$\left[\begin{array}{cc|cc} 0 & Z'(D) & 0 & X'(D) \\ I & 0 & 0 & 0 \end{array}\right].$$

The shifted symplectic product matrix still has the standard form in (3.7) because these last row operations do not change its entries. We now focus exclusively on the first $2c$ rows because the previous steps decoupled the last a rows from the first $2c$ rows. Consider the following submatrix:

$$H'(D) = \left[\begin{array}{c|c} Z(D) & X(D) \end{array}\right],$$

where we have reset variable labels so that $Z(D) = Z'(D)$ and $X(D) = X'(D)$. Perform row permutations on the above matrix so that the shifted symplectic product matrix for $H'(D)$ changes from

$$\bigoplus_{i=1}^{c} J,$$

to become

$$\left[\begin{array}{cc} 0 & I \\ I & 0 \end{array}\right], \tag{6.9}$$

where each identity and null matrix in the above matrix are $c \times c$-dimensional. We can employ the algorithm from Ref. [46] on the first c generators because the first c rows of the resulting check matrix form a commuting set (we do not use the row operations in that algorithm). The algorithm employs finite-depth CNOT gates, Hadamard gates, and phase gates and reduces the check matrix to have the following form:

$$\left[\begin{array}{cc|cc} 0 & 0 & L(D) & 0 \\ U(D) & Z_2(D) & X_1(D) & X_2(D) \end{array}\right],$$

where $L(D)$ is a $c \times c$ lower triangular matrix and $U(D)$ is a $c \times c$ upper triangular matrix. The i^{th} diagonal entry $u_{ii}(D)$ of $U(D)$ is equal to $1/l_{ii}\left(D^{-1}\right)$ where $l_{ii}(D)$ is the i^{th} diagonal entry of $L(D)$. This relationship holds because of the shifted symplectic relations in (6.9) and because gates in the shift-invariant Clifford group do not affect the shifted symplectic relations. We now employ several row operations whose net effect is to preserve the shifted symplectic relations in (6.9)—we can therefore include them as a part of the original polynomial symplectic Gram-Schmidt orthogonalization procedure. Multiply row i of the above check matrix by $1/l_{ii}(D)$ and multiply row $i+c$ by $1/u_{ii}(D)$ for all $i \in \{1, \ldots, c\}$. Then use row operations to cancel all the off-diagonal entries in both $L(D)$ and $U(D)$. The resulting check matrix has the following form:

$$\left[\begin{array}{cc|cc} 0 & 0 & I & 0 \\ I & Z_2'(D) & X_1'(D) & X_2'(D) \end{array}\right],$$

where the primed matrices result from all the row operations. One can check that the shifted symplectic relations of the above matrix are equivalent to those in (6.9). Perform Hadamard gates on the first c qubits. The check matrix becomes

$$\left[\begin{array}{cc|cc} I & 0 & 0 & 0 \\ X_1'(D) & Z_2'(D) & I & X_2'(D) \end{array}\right]. \tag{6.10}$$

We show how to encode the above matrix starting from c ebits and $k+c$ information qubits. The following matrix stabilizes a set of c ebits:

$$\left[\begin{array}{ccc|ccc} I & I & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I & I & 0 \end{array}\right], \tag{6.11}$$

where each identity matrix is $c \times c$ and the last column of zeros in each matrix is $c \times (k+c)$. The receiver Bob possesses the first c qubits and the sender Alice possesses the last $k+2c$ qubits. The following matrix is the information-qubit:

$$\left[\begin{array}{ccc|ccc} 0 & 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I \end{array}\right]$$

where each identity matrix is $(k+c) \times (k+c)$ and each column of zeros is $(k+c) \times c$. It is important to track the information-qubit matrix throughout encoding and decoding so that we can determine at the end of the process if we have truly decoded the information qubits. Perform finite-depth CNOT operations from the first c ebits to the last $k+c$ qubits to encode the numerators of the entries in matrix $Z_2'(D)$. Let $Z_{2,N}'(D)$ denote this matrix of the numerators of the entries in $Z_2'(D)$. The stabilizer matrix becomes

$$\left[\begin{array}{ccc|ccc} I & I & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I & I & Z_{2,N}'(D) \end{array}\right],$$

and the information-qubit matrix becomes

$$\left[\begin{array}{ccc|ccc} 0 & Z_{2,N}''(D) & I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I \end{array}\right],$$

where $Z_{2,N}''(D)$ is the matrix that results on the "Z" side after performing the CNOT operations corresponding to the entries in $Z_{2,N}'(D)$. Perform Hadamard gates on the last $k+c$ qubits. The stabilizer matrix becomes

$$\left[\begin{array}{ccc|ccc} I & I & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_{2,N}'(D) & I & I & 0 \end{array}\right],$$

and the information-qubit matrix becomes

$$\left[\begin{array}{ccc|ccc} 0 & Z_{2,N}''(D) & 0 & 0 & 0 & I \\ 0 & 0 & I & 0 & 0 & 0 \end{array}\right].$$

Let $X_{2,N}'(D)$ denote the matrix whose entries are the numerators of the entries in $X_2'(D)$. Perform CNOT gates from the first c qubits to the last $k+c$ qubits corresponding to the entries in $X_{2,N}'(D)$. The stabilizer matrix becomes

$$\left[\begin{array}{ccc|ccc} I & I & 0 & 0 & 0 & 0 \\ 0 & A(D) & Z_{2,N}'(D) & I & I & X_{2,N}'(D) \end{array}\right],$$

where $A(D) = Z'_{2,N}(D) X''_{2,N}(D)$ and $X''_{2,N}(D)$ is the matrix that results on the "Z" side after performing the CNOT operations on the "X" side corresponding to the entries in $X'_{2,N}(D)$. The information-qubit matrix becomes

$$\left[\begin{array}{ccc|ccc} 0 & Z''_{2,N}(D) & 0 & 0 & 0 & I \\ 0 & X''_{2,N}(D) & I & 0 & 0 & 0 \end{array}\right].$$

Let $\Gamma(D)$ be a diagonal matrix whose i^{th} diagonal entry is the denominator of the i^{th} row of $Z'_2(D)$ and $X'_2(D)$. We perform infinite-depth operations corresponding to the entries in $\Gamma(D)$. The stabilizer matrix becomes

$$\left[\begin{array}{ccc|ccc} I & \Gamma^{-1}\left(D^{-1}\right) & 0 & 0 & 0 & 0 \\ 0 & A(D) & Z'_{2,N}(D) & I & \Gamma(D) & X'_{2,N}(D) \end{array}\right],$$

and the information-qubit matrix becomes

$$\left[\begin{array}{ccc|ccc} 0 & Z''_{2,N}(D)\Gamma^{-1}\left(D^{-1}\right) & 0 & 0 & 0 & I \\ 0 & X''_{2,N}(D)\Gamma^{-1}\left(D^{-1}\right) & I & 0 & 0 & 0 \end{array}\right].$$

The above stabilizer matrix is equivalent to the desired one in (6.10) by several row operations. We premultiply the first set of rows by $\Gamma\left(D^{-1}\right)$ and multiply the second set of rows by $\Gamma^{-1}(D)$. We can also use the resulting identity matrix in the first set of rows to perform row operations from the first set of rows to the second set of rows to realize the matrix $X'_1(D)$. The operators that Bob would really measure need to have finite weight so he would measure the operators corresponding to the entries in the following stabilizer matrix:

$$\left[\begin{array}{ccc|ccc} \Gamma\left(D^{-1}\right) & I & 0 & 0 & 0 & 0 \\ 0 & A(D) & Z'_{2,N}(D) & I & \Gamma(D) & X'_{2,N}(D) \end{array}\right]. \tag{6.12}$$

We are done with the encoding algorithm. Alice begins with a set of ebits and performs the encoding operations detailed in (6.11-6.12) and then performs the finite-depth operations detailed in (6.8-6.10) in reverse order. We now detail the steps of the decoding algorithm. Perform CNOT

gates corresponding to the entries in $X'_{2,N}(D)$ from the first set of c qubits to the last set of $k+c$ qubits. The stabilizer matrix becomes

$$\left[\begin{array}{ccc|ccc} I & \Gamma^{-1}\left(D^{-1}\right) & 0 & 0 & 0 & 0 \\ B(D) & A(D) & Z'_{2,N}(D) & I & \Gamma(D) & 0 \end{array}\right],$$

where $B(D) \equiv Z'_{2,N}(D) X''_{2,N}(D)$. The information-qubit matrix becomes

$$\left[\begin{array}{ccc|ccc} 0 & Z''_{2,N}(D)\Gamma^{-1}\left(D^{-1}\right) & 0 & 0 & 0 & I \\ X''_{2,N}(D) & X''_{2,N}(D)\Gamma^{-1}\left(D^{-1}\right) & I & 0 & 0 & 0 \end{array}\right].$$

Perform Hadamard gates on the last set of $k+c$ qubits. The stabilizer matrix becomes

$$\left[\begin{array}{ccc|ccc} I & \Gamma^{-1}\left(D^{-1}\right) & 0 & 0 & 0 & 0 \\ B(D) & A(D) & 0 & I & \Gamma(D) & Z'_{2,N}(D) \end{array}\right],$$

and the information-qubit matrix becomes

$$\left[\begin{array}{ccc|ccc} 0 & Z''_{2,N}(D)\Gamma^{-1}\left(D^{-1}\right) & I & 0 & 0 & 0 \\ X''_{2,N}(D) & X''_{2,N}(D)\Gamma^{-1}\left(D^{-1}\right) & 0 & 0 & 0 & I \end{array}\right].$$

Perform CNOT gates from the first set of c qubits to the last set of $k+c$ qubits. These CNOT gates correspond to the entries in $Z'_{2,N}(D)$. The stabilizer matrix becomes

$$\left[\begin{array}{ccc|ccc} I & \Gamma^{-1}\left(D^{-1}\right) & 0 & 0 & 0 & 0 \\ B(D) & A(D) & 0 & I & \Gamma(D) & 0 \end{array}\right],$$

and the information-qubit matrix becomes

$$\left[\begin{array}{ccc|ccc} Z''_{2,N}(D) & Z''_{2,N}(D)\Gamma^{-1}\left(D^{-1}\right) & I & 0 & 0 & 0 \\ X''_{2,N}(D) & X''_{2,N}(D)\Gamma^{-1}\left(D^{-1}\right) & 0 & 0 & 0 & I \end{array}\right].$$

Row operations from the first set of rows of the stabilizer to each of the two sets of rows in the information-qubit matrix reduce the information-qubit matrix to the following form:

$$\left[\begin{array}{ccc|ccc} 0 & 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I \end{array}\right].$$

Then we perform the finite-depth operations detailed in (6.8-6.10). We have now finished the algorithm for the decoding circuit because the logical operators for the information qubits appear in their original form. □

6.3.1 Discussion

Similar practical issues arise in these circuits as we discussed in the previous chapter. Encoding circuits with infinite-depth operations are acceptable if we assume that noiseless encoding is possible. Otherwise, infinite-depth operations could lead to catastrophic propagation of uncorrected errors. Noiseless encoding is difficult to achieve in practice but we may be able to come close to it by concatenation of codes at the encoder.

There is a dichotomy of these codes similar to that in the previous chapter. Some of the codes may have a simpler form in which the encoding circuit requires finite-depth operations only. These codes fall within the first class of codes discussed in the previous chapter and will be more useful in practice because they propagate errors in the encoding circuit to a finite number of qubits only. The remaining codes that do not have this structure fall within the second class of codes whose encoding circuits have both finite-depth and infinite-depth operations and whose decoding circuits have finite-depth operations only.

6.3.2 Importing Classical Convolutional Codes over $GF(4)$

One benefit of the new entanglement-assisted quantum convolutional codes is that we can produce one from an arbitrary classical convolutional code over $GF(4)$. The error-correcting properties of the classical convolutional code translate to the resulting quantum convolutional code. It is less clear how the rate translates because we use the expansion technique. We know that the term $(2k-n)/n$ lower bounds the "entanglement-assisted" rate where n and k are the parameters from

the imported classical code. The rate should get a significant boost from entanglement—the rate boosts by the number of ebits that the code requires.

The construction for importing an $[n, k]$ classical convolutional code over $GF(4)$ is as follows. Suppose the check matrix for the classical code is an $n-k \times n$-dimensional matrix $H(D)$ whose entries are polynomials over $GF(4)$. We construct the quantum check matrix $\tilde{H}(D)$ according to the following formula:

$$\tilde{H}(D) = \gamma\left(\left[\begin{array}{c} \omega H(D) \\ \bar{\omega} H(D) \end{array}\right]\right)$$

where γ denotes the isomorphism between elements of $GF(4)$ and symplectic binary vectors detailed in (3.38). We use this construction for the example in the next section.

6.4 Example

We take the convolutional generators from Ref. [51] as our example. Ref. [34] originally used these generators in a block code. We import the following classical convolutional code over $GF(4)$:

$$(\cdots|0000|1\bar{\omega}10|1101|0000|\cdots). \tag{6.13}$$

We produce two quantum convolutional generators by multiplying the above generator by ω and $\bar{\omega}$ and applying the map in (3.38). The resulting quantum convolutional generators are as follows:

$$(\cdots|IIII|ZXZI|ZZIZ|IIII|\cdots), \qquad (\cdots|IIII|XYXI|XXIX|IIII|\cdots).$$

These generators have the following representation in the polynomial formalism:

$$\left[\begin{array}{cccc|cccc} 1+D & D & 1 & D & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1+D & 1+D & 1 & D \end{array}\right].$$

The shifted symplectic product matrix $\Omega(D)$ for the above code is as follows:

$$\Omega(D) = \begin{bmatrix} D+D^{-1} & D^{-1} \\ D & D+D^{-1} \end{bmatrix}.$$

The above matrix is not reducible to the standard form by any row operations. We therefore expand the code by a factor of two to give four generators with a frame size of eight. The two-expanded check matrix $H(D)$ is as follows:

$$H(D) = \left[\begin{array}{cccccccc|cccccccc} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ D & D & 0 & D & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & D & D & 0 & D & 1 & 1 & 1 & 0 \end{array}\right].$$

The shifted symplectic product matrix $\Omega_2(D)$ of the two-expanded check matrix is as follows:

$$\Omega_2(D) = \begin{bmatrix} 0 & 0 & 1+D^{-1} & D^{-1} \\ 0 & 0 & 1 & 1+D^{-1} \\ 1+D & 1 & 0 & 0 \\ D & 1+D & 0 & 0 \end{bmatrix}.$$

We proceed with the Gram-Schmidt procedure because this matrix satisfies its initial requirements. We swap generators two and three to be the first and second generators of the check matrix because they have the commutation relations of half of an ebit. The shifted symplectic product matrix becomes

$$\begin{bmatrix} 0 & 1 & 0 & 1+D^{-1} \\ 1 & 0 & 1+D & 0 \\ 0 & 1+D^{-1} & 0 & D^{-1} \\ 1+D & 0 & D & 0 \end{bmatrix}.$$

Multiply generator two by $1+D$ and add to generator four. Multiply generator one by $1+D^{-1}$ and add to generator three. The shifted symplectic matrix becomes

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1+D^{-1}+D^{-2} \\ 0 & 0 & 1+D+D^2 & 0 \end{bmatrix}.$$

We finally divide generator four by $1 + D + D^2$ and the shifted symplectic product matrix then becomes

$$\bigoplus_{i=1}^{2} J,$$

so that it has the commutation relations of halves of two ebits. The check matrix resulting from these operations is as follows:

$$H_2(D) = \left[\begin{array}{c|c} Z_2(D) & X_2(D) \end{array}\right], \tag{6.14}$$

where

$$Z_2(D) = \left[\begin{array}{cccccccc} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ D & D & 0 & D & 1 & 0 & 1 & 0 \\ 1 & \frac{1}{D}+1 & 1 & 0 & 1 & 1 & 0 & 1 \\ \frac{D^2+D}{D^2+D+1} & \frac{D^2+D}{D^2+D+1} & 0 & \frac{D^2+D}{D^2+D+1} & \frac{D+1}{D^2+D+1} & \frac{1}{D^2+D+1} & \frac{D+1}{D^2+D+1} & 0 \end{array}\right],$$

$$X_2(D) = \left[\begin{array}{cccccccc} 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{1}{D}+1 & \frac{1}{D} & \frac{1}{D}+1 & 0 & \frac{1}{D}+1 & \frac{1}{D}+1 & 0 & \frac{1}{D}+1 \\ \frac{D}{D^2+D+1} & \frac{D}{D^2+D+1} & 0 & \frac{D}{D^2+D+1} & \frac{1}{D^2+D+1} & \frac{D}{D^2+D+1} & \frac{1}{D^2+D+1} & 0 \end{array}\right].$$

The error-correcting properties of the above check matrix are equivalent to the error-correcting properties of the original two generators.

This code sends six information qubits and consumes two ebits per eight channel uses. The rate pair for this code is therefore $(3/4, 1/4)$.

We can now apply the algorithm in Theorem 6.3.1 to determine the encoding and decoding circuits for this code. The encoding circuit begins from a set of two ebits and eight information qubits per frame with the following stabilizer matrix:

$$H_0(D) = \left[\begin{array}{cccccccccc|cccccccccc} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right].$$

We label the eight qubits on the right side of each matrix above as $1, \ldots, 8$. We label Bob's two qubits on the left as $B1$ and $B2$. Perform the following finite-depth operations (in order from left to right and then top to bottom):

$$C\left(1,4,D+D^2\right)C\left(1,5,1+D^2\right)C\left(1,6,1\right)C\left(1,7,1+D\right)C\left(2,4,D\right)C\left(2,5,1+D\right)C\left(2,6,1\right)$$
$$H\left(3,\ldots,8\right)C\left(1,4,D+D^2+D^4\right)C\left(1,5,D^2\right)C\left(1,6,1+D\right)C\left(1,7,D^2\right)C\left(2,4,D+D^2\right)$$
$$C\left(2,5,1+D\right)C\left(2,6,1\right)C\left(2,7,1+D\right)$$

where we use the notation $C\left(q_1,q_2,f\left(D\right)\right)$ to represent a finite-depth CNOT gate from qubit one to qubit two that implements the polynomial $f\left(D\right)$, $H\left(q_i,\ldots,q_j\right)$ is a sequence of Hadamard gates applied to qubits q_i through q_j in each frame, $P\left(q\right)$ is a phase gate applied to qubit q in each frame, and $C\left(q,1/f\left(D\right)\right)$ is an infinite-depth CNOT gate implementing the rational polynomial $1/f\left(D\right)$ on qubit q. Alice performs the following infinite-depth operations:

$$H\left(1,2\right)C\left(1,\frac{1}{1+D^{-1}+D^{-2}}\right)C\left(2,\frac{1}{1+D^{-1}+D^{-2}}\right)H\left(1,2\right).$$

She then finishes the encoding circuit with the following finite-depth operations:

$$H\left(1,2\right)C\left(2,3,1\right)C\left(2,5,1\right)C\left(2,6,1\right)C\left(2,8,1\right)P\left(2\right)H\left(3,\ldots,8\right)S\left(2,3\right) \tag{6.15}$$
$$C\left(1,2,1\right)C\left(1,3,1\right)C\left(1,5,1\right)C\left(1,6,1\right)C\left(1,8,1\right)P\left(2\right)$$

The code she encodes has equivalent error-correcting properties to the check matrix in (6.14).

Bob performs the following operations in the decoding circuit. He first performs the operations in (6.15) in reverse order. He then performs the following finite-depth operations:

$$C\left(B1,4,D+D^2+D^4\right)C\left(B1,5,D^2\right)C\left(B1,6,1+D\right)C\left(B1,7,D^2\right)C\left(B2,4,D+D^2\right)$$
$$C\left(B2,5,1+D\right)C\left(B2,6,1\right)C\left(B2,7,1+D\right)H\left(3,\ldots,8\right)$$
$$C\left(B1,4,D+D^2\right)C\left(B1,5,1+D^2\right)C\left(B1,6,1\right)C\left(B1,7,1+D\right)C\left(B2,4,D\right)$$
$$C\left(B2,5,1+D\right)C\left(B2,6,1\right)$$

The information qubits then appear at the output of this online decoding circuit.

6.5 Optimal Entanglement Formulas

We conjecture two formulas for the optimal number of ebits that a general (non-CSS) entanglement-assisted quantum convolutional code or one imported from a classical quaternary convolutional code need to consume per frame of operation. We show that this conjecture holds for a particular example.

Conjecture 6.5.1. *The optimal number c of ebits necessary per frame for an entanglement-assisted quantum convolutional code is*

$$c = \operatorname{rank}\left(H_X(D)H_Z^T(D^{-1}) + H_Z(D)H_X^T(D^{-1})\right)/2 \tag{6.16}$$

where $H(D) = \left[\begin{array}{c|c} H_Z(D) & H_X(D) \end{array}\right]$ *represents the parity check matrix for a set of quantum convolutional generators that do not necessarily form a commuting set.*

It is clear that the above formula holds after we have expanded an original set of generators and their commutation relations are reducible by the polynomial symplectic Gram-Schmidt orthogonalization procedure to the standard form in (3.7). The proof technique follows from the proof technique outlined in Section 3.6. But we are not sure how to apply this formula to an initial set of unexpanded generators.

Conjecture 6.5.2. *The optimal number c of ebits required per frame for an entanglement-assisted quantum convolutional code imported from a classical quaternary convolutional code with parity check matrix $H(D)$ is*

$$c = \operatorname{rank}\left(H(D)H^\dagger(D^{-1})\right). \tag{6.17}$$

We know the number of ebits required for a CSS entanglement-assisted quantum convolutional code is $\operatorname{rank}\left(H_1(D)H_2^T(D^{-1})\right)$ where $H_1(D)$ and $H_2(D)$ correspond to the classical binary convolutional codes that we import to correct respective bit and phase flips. Comparing the CSS convolutional formula, the CSS block formula in Corollary 3.6.1, and the general block formula in Theorem 3.6.1, the above conjecture seems natural.

We finally provide an example of the above conjecture. It is a slight modification of the code presented in Ref. [53].

Example 6.5.1. Consider the quantum convolutional code with quantum check matrix as follows:

$$\left[\begin{array}{ccccc|ccccc} 0 & 0 & 0 & 0 & 0 & h(D) & 0 & D & 1 & h(D) \\ h(D) & D & 0 & 1 & h(D) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & D & D & D & 0 & 1 & 0 & 1 & 1 \\ 0 & \frac{1}{D} & 1 & \frac{1}{D} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{D} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right],$$

where $h(D) = 1 + D$. This code requires two ebits and one ancilla qubit for quantum redundancy and encodes two information qubits. The shifted symplectic product matrix $H_X(D)H_Z^T(D^{-1}) + H_Z(D)H_X^T(D^{-1})$ [51, 52] for this code is as follows:

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{6.18}$$

The rank of the above matrix is four and the code requires two ebits. Therefore, the conjecture holds for this example.

6.6 Closing Remarks

There are several differences between the methods used for general, non-CSS codes discussed in this chapter and the CSS codes used in the previous chapter. It was more straightforward to determine how to use ebits efficiently in CSS entanglement-assisted quantum convolutional codes, but we have had to introduce the expansion technique in Section 6.1 in order to determine how to use ebits efficiently for codes in this chapter. There was also no need for an explicit Gram-Schmidt orthogonalization procedure in the previous chapter. The Smith algorithm implicitly produced the row operations necessary for symplectic orthogonalization.

We do have some methods in the next chapter that do not require expansion of a check matrix or an explicit Gram-Schmidt procedure but these methods do not make efficient use of entanglement and have a lower rate of quantum communication and higher rate of entanglement consumption than the codes discussed in this chapter. Nonetheless, we have determined ways to make these other codes more useful by encoding classical information in the extra entanglement with a superdense-coding-like effect [13]. These other codes are grandfather codes in the sense of Refs. [53, 84] because they consume entanglement to send both quantum and classical information. We discuss these techniques in the next chapter.

One negative implication of the expansion of a code is that the expanded code requires more qubits per frame. The expanded code then requires a larger buffer at both the sender's and receiver's local stations. The increased buffer will be a concern right now because it is difficult to build large quantum memories. This issue will become less of a concern as quantum technology advances. The entanglement-inefficient codes in the next chapter have the advantage that they do not require expansion and thus require smaller buffers. It therefore should be of interest to find solutions in between the entanglement-efficient codes discussed in this chapter and the entanglement-inefficient codes discussed in the previous paragraph.

Some outstanding issues remain. We have not proven the convergence of the polynomial symplectic Gram-Schmidt orthogonalization procedure and have instead provided a practical stopping condition. We have a conjecture for how to proceed with proving convergence. Suppose that we would like to construct a code consisting of one generator that does not commute with shifts of itself. We have found for many examples that the correct expansion factor for the generator is equal to the period of the inverse polynomial of the generator's shifted symplectic product. We do not have a proof that this factor is the correct one and we are not sure what the expansion factor should be when we would like to construct a code starting from more than one generator.

The techniques developed in this chapter represent a useful way for encoding quantum information. This framework will most likely lead to codes with reasonable performance but they will most likely not come close to achieving the quantum capacity of an entanglement-assisted channel. The next step should be to combine the theory in this chapter with Poulin et al.'s recent theory of quantum serial-turbo coding [58]. This combination might lead to entanglement-assisted quantum turbo codes that come close to achieving capacity.

Chapter 7

Entanglement-Assisted Quantum Convolutional Coding: Free Entanglement

Sometimes we've a free cater,
Perhaps on the house we've a waiter,
But free entanglement you say?
Not a price to pay?
Oh assume it for now and pay later.

We mentioned in the previous chapter that the expansion of a set of quantum convolutional generators increases the frame size of a code. This increase implies that each round of transmission in the protocol sends a larger number of encoded qubits and requires a larger quantum memory for its operation. Thus increasing the frame size is somewhat undesirable.

We offer the "free-entanglement" codes in this chapter as an alternative option to the codes in the previous chapter. The free-entanglement codes have a basic generator set that includes an arbitrary set of Pauli sequences. The free-entanglement codes consume more entanglement than is necessary and have a lower quantum information rate, but the benefit is that they do not require a larger quantum memory for each round of transmission. It is up to the quantum coding engineer to decide which option is more desirable for the particular coding application: using less entanglement and having a higher rate of quantum transmission or using a smaller quantum memory to send fewer qubits for each round of transmission.

We make the additional assumption that shared entanglement is freely available. Quantum information theorists make the "free entanglement" assumption when deriving the entanglement-assisted capacity of a quantum channel [40, 41]. This model makes sense when the sender and receiver can use the noisy channel and entanglement distillation protocols [65, 66] during off-peak times to generate a large amount of noiseless entanglement.

The "entanglement-assisted" rate (Section 3.3) of a quantum code is the ratio of the number of encoded information qubits to the number of physical qubits assuming that entanglement is available for free. The entanglement-assisted rate of our codes is at least k/n. Parameter k is the number of information qubits that the code encodes and parameter n is the number of qubits used for encoding. A basic set of $n-k$ generators specifies all of our entanglement-assisted quantum convolutional codes and yields the above entanglement-assisted rate.

We say that the entanglement-assisted rate is at least k/n because the example in Section 7.2 discusses a method to boost the rate. This method encodes some classical information into the extra entanglement along the lines of the "grandfather" coding technique discussed in the next chapter and in Ref. [84]. The sender can use this classical information and the entanglement to teleport [3] additional information qubits and increase the rate of quantum transmission.

7.1 Construction

The proof of our main theorem below outlines how to encode a stream of information qubits, ancilla qubits, and shared ebits so that the encoded qubits have the error-correcting properties of an arbitrary set of Paulis. The receiver may employ an error estimation algorithm such as Viterbi decoding [2] to determine the most likely errors that the noisy quantum communication channel induces on the encoded stream. We then show how to decode the encoded qubit stream so that the information qubits become available at the receiving end of the channel.

The encoding circuits in the proof of our theorem employ both finite-depth and infinite-depth operations. The decoding circuits employ finite-depth operations only. Infinite-depth operations can lead to catastrophic error propagation as discussed in previous chapters. In our proof below, we restrict infinite-depth operations to act on qubits before sending them over the noisy channel.

Catastrophic error propagation does not occur under the ideal circumstance when the operations in the encoding circuit are noiseless.

Our theorem below begins with a "quantum check matrix" that consists of a set of Pauli sequences with desirable error-correcting properties. This quantum check matrix does not necessarily correspond to a commuting stabilizer. The proof of the theorem shows how to incorporate ebits so that the sender realizes the same quantum check matrix for her qubits and the sender and receiver's set of generators form a valid commuting stabilizer.

Theorem 7.1.1. *Suppose the following quantum check matrix*

$$S(D) = \left[\begin{array}{c|c} Z(D) & X(D) \end{array} \right] \in \mathbb{F}_2[D]^{(n-k)\times 2n},$$

where $S(D)$ is of full rank and does not necessarily form a commuting stabilizer and $\mathbb{F}_2[D]$ is the field of binary polynomials. Then an entanglement-assisted quantum convolutional code exists that has the same error-correcting properties as the above quantum check matrix $S(D)$. The entanglement-assisted rate of the code is at least k/n.

Proof. Suppose that the Smith form [80] of $X(D)$ is as follows

$$X(D) = A(D) \begin{bmatrix} \Gamma_1(D) & 0 & 0 \\ 0 & \Gamma_2(D) & 0 \\ 0 & 0 & 0 \end{bmatrix} B(D), \tag{7.1}$$

where $A(D)$ is $(n-k)\times(n-k)$, $B(D)$ is $n\times n$, $\Gamma_1(D)$ is an $s\times s$ diagonal matrix whose entries are powers of D, and $\Gamma_2(D)$ is a $(c-s)\times(c-s)$ diagonal matrix whose entries are arbitrary polynomials. Perform the row operations in $A^{-1}(D)$ and the column operations in $B^{-1}(D)$ on $S(D)$. The quantum check matrix $S(D)$ becomes

$$\left[\begin{array}{c|ccc} & \Gamma_1(D) & 0 & 0 \\ E(D) & 0 & \Gamma_2(D) & 0 \\ & 0 & 0 & 0 \end{array} \right], \tag{7.2}$$

where $E(D) = A^{-1}(D) Z(D) B^T\left(D^{-1}\right)$. Suppose $E_1(D)$ is the first c columns of $E(D)$ and $E_2(D)$ is the next $n-c$ columns of $E(D)$ so that the quantum check matrix is as follows:

$$\left[\begin{array}{cc|ccc} & & \Gamma_1(D) & 0 & 0 \\ E_1(D) & E_2(D) & 0 & \Gamma_2(D) & 0 \\ & & 0 & 0 & 0 \end{array}\right]. \tag{7.3}$$

Perform Hadamard gates on the last $n-c$ qubits so that the quantum check matrix becomes

$$\left[\begin{array}{cc|ccc} & & \Gamma_1(D) & 0 & E_{2,1}(D) \\ E_1(D) & 0 & 0 & \Gamma_2(D) & E_{2,2}(D) \\ & & 0 & 0 & E_{2,3}(D) \end{array}\right], \tag{7.4}$$

where

$$E_2(D) = \left[\begin{array}{c} E_{2,1}(D) \\ E_{2,2}(D) \\ E_{2,3}(D) \end{array}\right]. \tag{7.5}$$

Perform CNOT operations from the first s qubits to the last $n-c$ qubits to clear the entries in $E_{2,1}(D)$. The quantum check matrix becomes

$$\left[\begin{array}{cc|ccc} & & \Gamma_1(D) & 0 & 0 \\ E_1(D) & 0 & 0 & \Gamma_2(D) & E_{2,2}(D) \\ & & 0 & 0 & E_{2,3}(D) \end{array}\right]. \tag{7.6}$$

The Smith form of $E_{2,3}(D)$ is as follows

$$E_{2,3}(D) = A_E(D) \left[\begin{array}{cc} \Gamma(D) & 0 \end{array}\right] B_E(D), \tag{7.7}$$

where $A_E(D)$ is $(n-k-c)\times(n-k-c)$, $B_E(D)$ is $(n-c)\times(n-c)$, and $\Gamma(D)$ is a $(n-k-c)\times(n-k-c)$ diagonal matrix whose entries are polynomials. The Smith form of $E_{2,3}(D)$ is full

rank because the original quantum check matrix $S(D)$ is full rank. Perform the row operations in $A_E^{-1}(D)$ and the column operations in $B_E^{-1}(D)$. The quantum check matrix becomes

$$\left[\begin{array}{cc|cccc} & & \Gamma_1(D) & 0 & 0 & 0 \\ E_1'(D) & 0 & 0 & \Gamma_2(D) & E_{2,2a}'(D) & E_{2,2b}'(D) \\ & & 0 & 0 & \Gamma(D) & 0 \end{array}\right], \tag{7.8}$$

where

$$E_1'(D) = \left[\begin{array}{cc} I & 0 \\ 0 & A_E^{-1}(D) \end{array}\right] E_1(D), \tag{7.9}$$

$$E_{2,2}'(D) = E_{2,2}(D) B_E^{-1}(D) = \left[\begin{array}{cc} E_{2,2a}'(D) & E_{2,2b}'(D) \end{array}\right]. \tag{7.10}$$

Perform a modified version of the Smith algorithm to reduce the $(c-s)\times(n-c)$ matrix $E_{2,2b}'(D)$ to a lower triangular form (discussed in Chapter 5). This modified algorithm uses only column operations to transform

$$E_{2,2b}'(D) \rightarrow \left[\begin{array}{cc} L(D) & 0 \end{array}\right], \tag{7.11}$$

where $L(D)$ is $(c-s)\times(c-s)$ and the null matrix is $(c-s)\times(n+s-2c)$. The quantum check matrix becomes

$$\left[\begin{array}{cc|ccccc} & & \Gamma_1(D) & 0 & 0 & 0 & 0 \\ E_1'(D) & 0 & 0 & \Gamma_2(D) & E_{2,2a}'(D) & L(D) & 0 \\ & & 0 & 0 & \Gamma(D) & 0 & 0 \end{array}\right]. \tag{7.12}$$

We have now completed the decomposition of the quantum check matrix with column and row operations.

We turn to showing how to encode and decode a certain quantum check matrix that proves to be useful in encoding the above quantum check matrix. Consider the following quantum check matrix

$$\left[\begin{array}{cc|cc} I & 0 & 0 & 0 \\ 0 & 0 & \Gamma_2(D) & L(D) \end{array}\right], \tag{7.13}$$

where $\Gamma_2(D)$ and $L(D)$ are from the matrix in (7.12) and each of them, the identity matrix, and the null matrices have dimension $(c-s) \times (c-s)$. We use a method for encoding and decoding the quantum check matrix in (7.13) similar to the method in Chapter 5 for the second class of CSS entanglement-assisted quantum convolutional codes. We begin with a set of $c-s$ ebits and $c-s$ information qubits. The following matrix stabilizes the ebits

$$\left[\begin{array}{ccc|ccc} I & I & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I & I & 0 \end{array}\right], \tag{7.14}$$

where Bob possesses the $c-s$ qubits on the "left," Alice possesses the $2(c-s)$ qubits on the "right," and each matrix is $(c-s) \times (c-s)$. The following matrix represents the information qubits:

$$\left[\begin{array}{ccc|ccc} 0 & 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I \end{array}\right]. \tag{7.15}$$

The above information-qubit matrix represents the logical operators for the information qubits and gives a useful way of tracking the information qubits while processing them. Tracking the information-qubit matrix helps to confirm that the information qubits decode properly at the receiver's end [52]. We track both the above stabilizer and the information-qubit matrix as they progress through some encoding operations. Alice then performs CNOT gates from her first $c-s$ qubits to her next $c-s$ qubits. These gates multiply the middle $c-s$ columns of the "X" matrix by $L(D)$ and add the result to the last $c-s$ columns and multiply the last $c-s$ columns of the "Z" matrix by $L^T\left(D^{-1}\right)$ and add the result to the last $c-s$ columns. The stabilizer becomes

$$\left[\begin{array}{ccc|ccc} I & I & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I & I & L(D) \end{array}\right], \tag{7.16}$$

and the information-qubit matrix becomes

$$\left[\begin{array}{ccc|ccc} 0 & L^T\left(D^{-1}\right) & I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I \end{array}\right]. \tag{7.17}$$

Alice performs infinite-depth operations on her first $c-s$ qubits corresponding to the rational polynomials $\gamma_{2,1}^{-1}\left(D^{-1}\right)$, ..., $\gamma_{2,c-s}^{-1}\left(D^{-1}\right)$ in $\Gamma_2^{-1}\left(D^{-1}\right)$. These operations multiply the middle $c-s$ columns of the "Z" matrix by $\Gamma_2^{-1}\left(D^{-1}\right)$ and multiply the middle $c-s$ columns of the "X" matrix by $\Gamma_2\left(D\right)$. The stabilizer matrix becomes

$$\left[\begin{array}{ccc|ccc} I & \Gamma_2^{-1}\left(D^{-1}\right) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I & \Gamma_2\left(D\right) & L\left(D\right) \end{array}\right], \tag{7.18}$$

and the information-qubit matrix becomes

$$\left[\begin{array}{ccc|ccc} 0 & L^T\left(D^{-1}\right)\Gamma_2^{-1}\left(D^{-1}\right) & I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I \end{array}\right]. \tag{7.19}$$

Alice's part of the above stabilizer matrix is equivalent to the quantum check matrix in (7.13) by row operations (premultiplying the first set of rows by $\Gamma_2\left(D^{-1}\right)$.)

We now illustrate a way to decode the encoded stabilizer in (7.18) and information-qubit matrix in (7.19) so that the information qubits appear at the output of the decoding circuit. Bob performs CNOT gates from the first set of qubits to the third set of qubits corresponding to the entries in $L\left(D\right)$. The stabilizer becomes

$$\left[\begin{array}{ccc|ccc} I & \Gamma_2^{-1}\left(D^{-1}\right) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I & \Gamma_2\left(D\right) & 0 \end{array}\right], \tag{7.20}$$

and the information-qubit matrix becomes

$$\left[\begin{array}{ccc|ccc} L^T\left(D^{-1}\right) & L^T\left(D^{-1}\right)\Gamma_2^{-1}\left(D^{-1}\right) & I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I \end{array}\right]. \tag{7.21}$$

Bob finishes decoding at this point because we can equivalently express the information-qubit matrix as follows

$$\left[\begin{array}{ccc|ccc} 0 & 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I \end{array}\right], \tag{7.22}$$

by multiplying the first $c-s$ rows of the stabilizer by $L^T\left(D^{-1}\right)$ and adding to the first $c-s$ rows of the information-qubit matrix. The information qubits are available at the receiving end of the channel because the above information-qubit matrix is equivalent to the original one in (7.15).

We show how to encode the quantum check matrix in (7.12) using ebits, ancilla qubits, and information qubits. We employ the encoding technique for the submatrix listed above and use some other techniques as well. Suppose that we have the following matrix that stabilizes a set of c ebits per frame, $n-k-c$ ancilla qubits per frame, and k information qubits per frame:

$$\left[\begin{array}{ccccccc|ccccccc} I & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & I & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & I & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 \end{array}\right]. \tag{7.23}$$

The first and third sets of rows have s rows and correspond to s ebits per frame, the second and fourth sets of rows have $c-s$ rows and correspond to $c-s$ ebits per frame, and the last set of $n-k-c$ rows corresponds to $n-k-c$ ancilla qubits per frame. The above matrix has $n+c$ columns on both the "Z" and "X" side so that the above matrix stabilizes k information qubits per frame. Bob possesses the first c qubits and Alice possesses the next n qubits. Alice performs the encoding operations in (7.14-7.19) to get the following stabilizer:

$$\left[\begin{array}{ccccccc|ccccccc} I & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & I & 0 & \Gamma_2^{-1}\left(D^{-1}\right) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & I & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & \Gamma_2(D) & 0 & L(D) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 \end{array}\right]. \tag{7.24}$$

We perform several row operations to get the quantum check matrix in (7.12). Premultiply the middle set of rows by $\Gamma_1(D)$. Premultiply the last set of rows by $E'_{2,2a}(D)$ and add the result to the set of rows above the last set. Premultiply the last set of rows by $\Gamma(D)$. Finally, premultiply

the first two sets of rows by $E_1''(D) = E_1'(D)\left[I \oplus \Gamma_2\left(D^{-1}\right)\right]$ and add the result to the last three sets of rows. The quantum check matrix becomes

$$\left[\begin{array}{ccccccc|ccccccc} I & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & I & 0 & \Gamma_2^{-1}\left(D^{-1}\right) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & 0 & 0 & 0 & \Gamma_1(D) & 0 & \Gamma_1(D) & 0 & 0 & 0 & 0 \\ E_1''(D) & & & E_1'(D) & 0 & 0 & 0 & 0 & I & 0 & \Gamma_2(D) & E_{2,2a}'(D) & L(D) & 0 \\ & & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \Gamma(D) & 0 & 0 \end{array}\right]. \tag{7.25}$$

□

Alice's part of the above quantum check matrix and the last three sets of rows are equivalent to the quantum check matrix in (7.12). Alice then performs all finite-depth encoding operations (column operations) in (7.1-7.12) in reverse order to obtain the desired quantum check matrix in the statement of the theorem. Decoding consists of performing all the operations in (7.1-7.12) and then applying the decoding operations in (7.20-7.22). The entanglement-assisted rate of the above code is k/n because the code uses a noisy quantum communication channel n times per frame to send k information qubits per frame.

7.2 Example

We now present an example that begins with the same generators as those in the example from the previous chapter. We begin with the following two Pauli generators:

$$(\cdots|IIII|ZXZI|ZZIZ|IIII|\cdots), \qquad (\cdots|IIII|XYXI|XXIX|IIII|\cdots).$$

We write the above two generators as a quantum check matrix:

$$\left[\begin{array}{cccc|cccc} 1+D & D & 1 & D & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1+D & 1+D & 1 & D \end{array}\right] \tag{7.26}$$

We encode two information qubits per frame with the help of two ebits. The stabilizer matrix for the unencoded qubit stream is as follows:

$$\left[\begin{array}{cccccc|cccccc} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{array}\right] \tag{7.27}$$

Rows one and three correspond to one ebit and rows two and four correspond to the other. Multiply row one by D and add the result to row three, multiply row one by $1 + D^{-1} + D^2$ and add the result to row four, and multiply row two by $1 + D^{-2}$ and add the result to row four. These row operations give the following equivalent stabilizer:

$$\left[\begin{array}{cccccc|cccccc} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & D & D & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1+D^{-2} & 1+D^{-1}+D^2 & 1+D^{-1}+D^2 & 1+D^{-2} & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{array}\right]. \tag{7.28}$$

Figure 7.1 illustrates the operations that transform the unencoded stabilizer to the encoded one in an online encoding circuit. The final stabilizer is as follows

$$\left[\begin{array}{cccccc|cccccc} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & D & 1+D & D & 1 & D & 0 & 1 & 0 & 1 & 0 & 0 \\ 1+D^{-2} & D^{-1}+1+D & 0 & 1 & 0 & 0 & 1 & 0 & 1+D & 1+D & 1 & D \end{array}\right].$$

Compare Alice's Paulis in the last two rows of the above matrix to the quantum check matrix in (7.26). We have constructed a code with the same error-correcting properties because these two matrices are equivalent. The entanglement-assisted rate of the above code is 1/2 because it encodes two information qubits for every four uses of the noisy quantum channel.

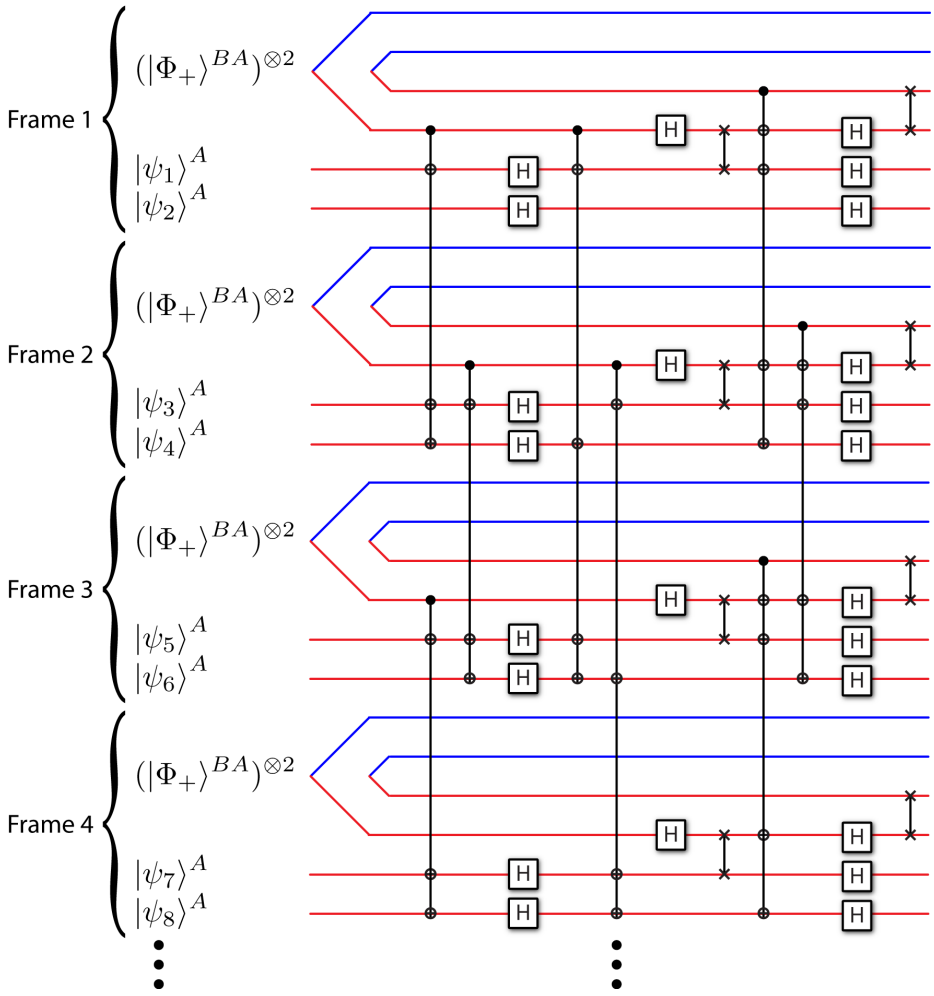

Figure 7.1: (Color online) An online encoding circuit for an entanglement-assisted quantum convolutional code. The receiver Bob possesses the first two qubits in the two ebits and the sender Alice possesses the second two qubits in the two ebits. The sender encodes two information qubits per frame with the help of her half of the two ebits.

Consider the following two operators:

$$\left[\begin{array}{cccccc|cccccc} 0 & 0 & D^{-1} & 1+D+D^{-2} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1+D^{2} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right].$$

The first row anticommutes with the first row in (7.28) and commutes with all other rows in (7.28). The second row anticommutes with the second row in (7.28) and commutes with all other rows in (7.28). These commutation relations imply that the above operators are useful for encoding classical information in a superdense-coding-like fashion. These operators encode two classical bits into the code and make use of the first two rows in (7.28) instead of just "wasting" them. Measuring the first two rows in (7.28) reveals the values of the two classical bits. We can determine the encoded versions of these "classical-information-encoding" operators by tracing how the operators change in the Heisenberg picture through the rest of the encoding circuit. We can use these two classical bits and consume one ebit to teleport an additional information qubit. This technique boosts the entanglement-assisted rate of this code from 1/2 to 3/4 because we can now encode three information qubits for every four uses of the noisy quantum communication channel.

7.3 Closing Remarks

The "free entanglement" approach of this chapter uses entanglement less efficiently than the protocol in the previous chapter. It does not require expanding a set of generators and therefore does not require a heuristic convergence argument as do the codes of the previous chapter. The free entanglement method always works and results in encoding and decoding circuits that act on smaller numbers of qubits than the circuits in the previous chapter.

Chapter 8

Unified Quantum Convolutional Coding

"Never chase after a bus, a girl, or a unifying theory.
There'll always be another one coming along."
—John Archibald Wheeler (1911–2008)

In this chapter, we design a framework for "grandfather" quantum convolutional codes. Our grandfather codes are useful for the simultaneous transmission of classical and quantum information. Rather than using block codes for this purpose, we design quantum convolutional codes.* Our technique incorporates many of the known techniques for quantum coding: subsystem codes [27, 29], entanglement-assisted codes [34], convolutional codes [44, 45, 1], and classical-quantum mixed coding [63, 85, 84]. The goal of our technique is to provide a formalism for designing codes that approach the optimal triple trade-off rates in the grandfather resource inequality in (1.5).

We structure this chapter as follows. Section 8.1 details our "grandfather" quantum convolutional codes. We explicitly show how to encode a stream of classical-quantum information using finite-depth operations and discuss the error-correcting properties of our codes. We end with an example of a grandfather quantum convolutional code. We discuss which errors the code corrects actively and others that it corrects passively.

*Kremsky, Hsieh, and Brun address the formulation of grandfather block codes in Ref. [84].

8.1 Grandfather Quantum Convolutional Codes

We detail the stabilizer formalism for our grandfather quantum convolutional codes and describe how they operate. This formalism is a significant extension of the entanglement-assisted formalism.

An $[n, k, l; r, c]$ grandfather quantum convolutional code encodes k information qubits and l information classical bits with the help of c ebits, $a = n - k - l - c - r$ ancilla qubits, and r gauge qubits. Each input frame includes the following:

(i). Alice's half of c ebits in the state $|\Phi^+\rangle$.

(ii). $a = n - k - c - l - r$ ancilla qubits in the state $|0\rangle$.

(iii). r gauge qubits (which can be in any arbitrary state σ).

(iv). l classical information bits $x^1 \cdots x^l$, given by a computational basis state $|x\rangle = X^{x^1} \otimes \cdots \otimes X^{x^l} |0\rangle^{\otimes l}$.

(v). k information qubits in a state $|\psi\rangle$.[†]

The left side of Figure **??** shows an example initial qubit stream before an encoding circuit operates on it.

The stabilizer matrix $S_0(D)$ for the initial qubit stream is as follows:

$$S_0(D) = \left[\begin{array}{cccccc|cccccc} I & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & I & I & 0 & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right], \tag{8.1}$$

where all identity matrices in the first two sets of rows are $c \times c$, the identity matrix in the last row is $a \times a$, the three columns of all zeros in both the "Z" and "X" matrices are respectively $(a + 2c) \times r$, $(a + 2c) \times l$, and $(a + 2c) \times k$. The first two sets of rows stabilize a set of c ebits and the last set of rows stabilize a set of a ancilla qubits. The first c columns of both the "Z" and "X" matrix correspond to halves of ebits that Bob possesses and the last n columns in both matrices correspond to the qubits that Alice possesses.

[†]This statement is not entirely true because the information qubits can be entangled across multiple frames, or with an external system, but we use it to illustrate the idea.

Different generators for the grandfather code are important in active error correction, in passive error correction, and for the identification of the l classical information bits. We first write the unencoded generators that act on the initial qubit stream. The first subgroup of generators is the entanglement subgroup $\mathcal{S}_{E,0}$ with the following generators:

$$S_{E,0}(D) = \left[\begin{array}{ccccc|ccccc} I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 \end{array}\right]. \tag{8.2}$$

The above generators are equivalent to the first two sets of rows in (8.1) acting on Alice's n qubits. The next subgroup is the isotropic subgroup $\mathcal{S}_{I,0}$ with the following generators:

$$S_{I,0}(D) = \left[\begin{array}{ccccc|ccccc} 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]. \tag{8.3}$$

The above generators are equivalent to the last set of rows in (8.1) acting on Alice's n qubits. The encoded versions of both of the above two matrices are important in the active correction of errors. The next subgroup is the gauge subgroup $\mathcal{S}_{G,0}$ whose generators are as follows:

$$S_{G,0}(D) = \left[\begin{array}{ccccc|ccccc} 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0 & 0 \end{array}\right]. \tag{8.4}$$

The generators in $\mathcal{S}_{G,0}$ correspond to quantum operations that have no effect on the encoded quantum information and therefore represent a set of errors to which the code is immune. The last subgroup is the classical subgroup $\mathcal{S}_{C,0}$ with generators

$$S_{C,0}(D) = \left[\begin{array}{ccccc|ccccc} 0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]. \tag{8.5}$$

The grandfather code passively corrects errors corresponding to the encoded version of the above generators because the initial qubit stream is immune to the action of operators in $\mathcal{S}_{C,0}$ (up to a global phase). Alice could measure the generators in $\mathcal{S}_{C,0}$ to determine the classical information in each frame. Unlike quantum information, it is possible to measure classical information without disturbing it.

Alice performs an encoding circuit with finite-depth operations to encode her stream of qubits before sending them over the noisy quantum channel. The encoding circuit transforms the initial stabilizer $S_0(D)$ to the encoded stabilizer $S(D)$ as follows:

$$S(D) = \left[\begin{array}{cc|cc} I & Z_{E1}(D) & 0 & X_{E1}(D) \\ 0 & Z_{E2}(D) & I & X_{E2}(D) \\ 0 & Z_I(D) & 0 & X_I(D) \end{array}\right], \tag{8.6}$$

where $Z_{E1}(D)$, $X_{E1}(D)$, $Z_{E2}(D)$ and $X_{E2}(D)$ are $c \times n$-dimensional, $Z_I(D)$ and $X_I(D)$ are $a \times n$-dimensional. The encoding circuit affects only the rightmost n entries in both the "Z" and "X" matrix of $S_0(D)$ because these are the qubits in Alice's possession. It transforms $S_{E,0}(D)$, $S_{I,0}(D)$, $S_{G,0}(D)$, and $S_{C,0}(D)$ as follows:

$$S_E(D) = \left[\begin{array}{c|c} Z_{E1}(D) & X_{E1}(D) \\ Z_{E2}(D) & X_{E2}(D) \end{array}\right], \quad S_I(D) = \left[\begin{array}{c|c} Z_I(D) & X_I(D) \end{array}\right], \tag{8.7}$$

$$S_G(D) = \left[\begin{array}{c|c} Z_{G1}(D) & X_{G1}(D) \\ Z_{G2}(D) & X_{G2}(D) \end{array}\right], \quad S_C(D) = \left[\begin{array}{c|c} Z_C(D) & X_C(D) \end{array}\right], \tag{8.8}$$

where $Z_{G1}(D)$, $X_{G1}(D)$, $Z_{G2}(D)$ and $X_{G2}(D)$ are $r \times n$-dimensional and $Z_C(D)$ and $X_C(D)$ are $l \times n$-dimensional. The above polynomial matrices have the same commutation relations as their corresponding unencoded polynomial matrices in (8.2-8.5) and respectively generate the entanglement subgroup $\mathcal{S}_E$, the isotropic subgroup $\mathcal{S}_I$, the gauge subgroup $\mathcal{S}_G$, and the classical subgroup $\mathcal{S}_C$.

A grandfather quantum convolutional code operates as follows.

(i). Alice begins with an initial qubit stream as above. She performs the finite-depth encoding operations corresponding to a specific grandfather quantum convolutional code.

(ii). She sends the encoded qubits online over the noisy quantum communication channel. The code passively protects against errors in $\langle \mathcal{S}_I, \mathcal{S}_G, \mathcal{S}_C \rangle$.

(iii). Bob combines the received qubits with his half of the ebits in each frame. He obtains the error syndrome by measuring the generators in (8.6). He processes these syndrome bits with

a classical error estimation algorithm to diagnose errors and applies recovery operations to reverse the errors.

(iv). He then performs the inverse of the encoding circuit to recover the initial qubit stream with the information qubits and the classical information bits. He recovers the classical information bits either by measuring the generators in $\mathcal{S}_C$ before decoding or the generators in $\mathcal{S}_{C,0}$ after decoding.

A grandfather quantum convolutional code corrects errors in a Pauli error set $\mathcal{E}$ that obey one of the following conditions $\forall E_a, E_b \in \mathcal{E}$:

$$\exists\ g \in \langle \mathcal{S}_I, \mathcal{S}_E \rangle : \left\{ g, E_a^\dagger E_b \right\} = 0 \quad \text{or} \quad E_a^\dagger E_b \in \langle \mathcal{S}_I, \mathcal{S}_G, \mathcal{S}_C \rangle \,.$$

It corrects errors that anticommute with generators in $\langle \mathcal{S}_I, \mathcal{S}_E \rangle$ by employing a classical error estimation algorithm. The code passively protects against errors in the group $\langle \mathcal{S}_I, \mathcal{S}_G, \mathcal{S}_C \rangle$.

Our scheme for quantum convolutional coding incorporates many of the known techniques for quantum error correction. It can take full advantage of the benefits of these different techniques.

8.2 Example

We present an example of a grandfather quantum convolutional code in this section. The code protects one information qubit and classical bit with the help of an ebit, an ancilla qubit, and a gauge qubit. The first frame of input qubits has the state

$$\rho_0 = \left|\Phi^+\right\rangle \left\langle\Phi^+\right| \otimes |0\rangle \langle 0| \otimes \sigma_0 \otimes |x_0\rangle \langle x_0| \otimes |\psi_0\rangle \langle \psi_0| \,, \tag{8.9}$$

where $|\Phi^+\rangle$ is the ebit, $|0\rangle$ is the ancilla qubit, σ_0 is an arbitrary state for the gauge qubit, $|x_0\rangle$ is a classical bit represented by state $|0\rangle$ or $|1\rangle$, and $|\psi_0\rangle$ is one information qubit equal to $\alpha_0 |0\rangle + \beta_0 |1\rangle$. The states of the other input frames have a similar form though recall that information qubits can be entangled across multiple frames.

The initial stabilizer for the code is as follows:

$$S_0(D) = \left[\begin{array}{cccccc|cccccc} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right].$$

The first two rows stabilize the ebit shared between Alice and Bob. Bob possesses the half of the ebit in column one and Alice possesses the half of the ebit in column two in both the left and right matrix. The third row stabilizes the ancilla qubit. We name Alice's qubits one through five (they are actually two through six in the above matrix from the left to the right).

The generators for the initial entanglement subgroup $\mathcal{S}_{E,0}$, isotropic subgroup $\mathcal{S}_{I,0}$, gauge subgroup $\mathcal{S}_{G,0}$, and classical subgroup $\mathcal{S}_{C,0}$ are respectively as follows:

$$S_{E,0}(D) = \left[\begin{array}{ccccc|ccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{array}\right],$$

$$S_{I,0}(D) = \left[\begin{array}{ccccc|ccccc} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right],$$

$$S_{G,0}(D) = \left[\begin{array}{ccccc|ccccc} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right],$$

$$S_{C,0}(D) = \left[\begin{array}{ccccc|ccccc} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right].$$

The sender performs the following finite-depth operations (order is from left to right and top to bottom):

$$H(2)\ C(2,3,D)\ C(2,4,1+D)\ C(2,5,D)\ H(3,4,5)$$
$$C(2,3,D)\ C(2,5,D)\ H(2)\ C(1,2,D)\ C(1,4,1+D)$$
$$C(1,5,1+D)\ H(1,2,3,4,5)\ C(1,3,D)\ C(1,4,1+D)$$
$$C(1,5,1+D)\ S(1,4).$$

where the notation for the above encoding operations was established in Section 4.5. The initial stabilizer matrix $S_0(D)$ transforms to $S(D)$ under these encoding operations, where

$$S(D) = \left[\begin{array}{cccccc|cccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1+D & 0 & D & 1 & 1+D \\ 0 & 1+D & D & 0 & 1 & 1+D & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & D & D & D & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right]. \tag{8.10}$$

The generators for the different subgroups transform respectively as follows:

$$S_E(D) = \left[\begin{array}{ccccc|ccccc} 0 & 0 & 0 & 0 & 0 & 1+D & 0 & D & 1 & 1+D \\ 1+D & D & 0 & 1 & 1+D & 0 & 0 & 0 & 0 & 0 \end{array}\right],$$

$$S_I(D) = \left[\begin{array}{ccccc|ccccc} 0 & 0 & D & D & D & 0 & 1 & 0 & 1 & 1 \end{array}\right],$$

$$S_G(D) = \left[\begin{array}{ccccc|ccccc} 0 & \frac{1}{D} & 1 & \frac{1}{D} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{D} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right],$$

$$S_C(D) = \left[\begin{array}{ccccc|ccccc} 1 & 1+D^{-1} & 0 & 1+D^{-1} & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right].$$

The code actively protects against an arbitrary single-qubit error in every other frame. One can check that the syndromes of the stabilizer in $S(D)$ satisfy this property. Consider the Pauli generators corresponding to the generators in the entanglement subgroup and the isotropic subgroup:

$$\cdots \left|\begin{array}{ccccc|ccccc} X & I & I & X & X & X & I & X & I & X \\ Z & I & I & Z & Z & Z & Z & I & I & Z \\ I & X & I & X & X & I & I & Z & Z & Z \end{array}\right| \cdots, \tag{8.11}$$

where all other entries in the left and right directions are tensor products of the identity. We can use a table-lookup syndrome-based algorithm to determine the error-correcting capability of the code. The method is similar to the technique originally outlined in detail in Ref. [1]. The syndrome vector s consists of six bits where $s = s_1 \cdots s_6$. The first bit s_1 is one if the error anticommutes with the operator $XIIXX$ in the first part of the first generator above and zero otherwise. The second bit s_2 is one if the error anticommutes with the operator $XIXIX$ in the delayed part of the first generator above and zero otherwise. The third through sixth bits follow

Error	Syndrome	Error	Syndrome	Error	Syndrome
X_1	001100	X_3	000001	X_5	001101
Y_1	111100	Y_3	010001	Y_5	111111
Z_1	110000	Z_3	010000	Z_5	110010
X_2	000100	X_4	001001		
Y_2	000110	Y_4	101011		
Z_2	000010	Z_4	100010		

Table 8.1: A list of possible single-qubit errors in a particular frame and the corresponding syndrome vector. The syndrome corresponding to any single-qubit error is unique. The code therefore corrects an arbitrary single-qubit error in every other frame.

a similar pattern for the second and third generators above. Table 8.1 lists all single-qubit errors over five qubits and their corresponding syndromes. The code corrects an arbitrary single-qubit error in every other frame using this algorithm because the syndromes are all unique. A syndrome-based Viterbi algorithm might achieve better performance than the simple syndrome table-lookup algorithm outlined above.

This code also has passive protection against errors in $\langle \mathcal{S}_I, \mathcal{S}_G, \mathcal{S}_C \rangle$. The Pauli form of the errors in this group span over three frames and are as follows:

$$\cdots \left| \begin{array}{c|c|c} IIIII & IXIXX & IIZZZ \\ IZIZI & IIZII & IIIII \\ IZIII & IIXII & IIIII \\ IZIZI & ZZIZI & IIIII \end{array} \right| \cdots \tag{8.12}$$

The smallest weight errors in this group have weight two and three. The code passively corrects the above errors or any product of them or any five-qubit shift of them.

There is a trade-off between passive error correction and the ability to encode quantum information as discussed in Ref. [31]. One can encode more quantum information by dropping the gauge group and instead encoding extra information qubits. The gauge generators then become logical X and Z operators for the extra encoded qubits. One can also turn classical bits into information qubits by dropping the generators in the classical subgroup. These generators then

become logical Z operators for the extra encoded qubits. By making the above replacements, the code loses some of its ability to correct passively, but we gain a higher quantum information rate. On the other hand, if we replace a gauge qubit with an ancilla qubit or an ebit, we gain the ability to correct extra errors. The trade-off now is that replacing gauge qubits with ebits or ancillas enhances the active error-correcting capability of the code, but increases the overall complexity of error correction.

8.3 Closing Remarks

We have presented a framework and a representative example for grandfather quantum convolutional codes. We have explicitly shown how these codes operate, and how to encode and decode a classical-quantum information stream by using ebits, ancilla qubits, and gauge qubits for quantum redundancy. The ultimate goal for this theory is to find quantum convolutional codes that might play an integral part in larger quantum codes that approach the grandfather capacity [64]. One useful line of investigation may be to combine this theory with the recent quantum turbo-coding theory [58].

Chapter 9

Convolutional Entanglement Distillation

A chap at the "Entanglement Distillery,"
Was drunk so they gave him the pillory,
They'd foul dirty ebits,
He said, "Convolutional circuits!"
So they augmented the quantum artillery.

The goal of entanglement distillation resembles the goal of quantum error correction [65, 66]. An entanglement distillation protocol extracts noiseless, maximally-entangled ebits from a larger set of noisy ebits. A sender and receiver can use these noiseless ebits as a resource for several quantum communication protocols [13, 3].

Bennett et al. showed that a strong connection exists between quantum error-correcting codes and entanglement distillation and demonstrated a method for converting an arbitrary quantum error-correcting code into a one-way entanglement distillation protocol [66]. A one-way entanglement distillation protocol utilizes one-way classical communication between sender and receiver to carry out the distillation procedure. Shor and Preskill improved upon Bennett et al.'s method by avoiding the use of ancilla qubits and gave a simpler method for converting an arbitrary CSS quantum error-correcting code into an entanglement distillation protocol [86]. Nielsen and Chuang showed how to convert a stabilizer quantum error-correcting code into a stabilizer entanglement distillation protocol [32]. Luo and Devetak then incorporated shared entanglement to demonstrate

how to convert an entanglement-assisted stabilizer code into an entanglement-assisted entanglement distillation protocol [76]. All of the above constructions exploit the relationship between quantum error correction and entanglement distillation—we further exploit the connection in this chapter by forming a *convolutional* entanglement distillation protocol.

In this last chapter, our main contribution is a theory of convolutional entanglement distillation. Our theory allows us to import the entirety of classical convolutional coding theory for use in entanglement distillation. The task of finding a good convolutional entanglement distillation protocol now becomes the well-established task of finding a good classical convolutional code.

We begin in Section 9.3 by showing how to construct a convolutional entanglement distillation protocol from an arbitrary quantum convolutional code. We translate earlier protocols [86, 32] for entanglement distillation of a block of noisy ebits to the convolutional setting. A convolutional entanglement distillation protocol has the benefit of distilling entanglement "online." This online property is useful because the sender and receiver can distill entanglement "on the fly" as they obtain more noisy ebits. This translation from a quantum convolutional code to an entanglement distillation protocol is useful because it paves the way for our major contribution.

Our major advance is a method for constructing a convolutional entanglement distillation protocol when the sender and receiver initially share some noiseless ebits. As stated previously, prior quantum convolutional work requires the code to satisfy the restrictive self-orthogonality constraint, and authors performed specialized searches for classical convolutional codes that meet this constraint [43, 44, 48, 1]. We lift this constraint by allowing shared noiseless entanglement. The benefit of convolutional entanglement distillation with entanglement assistance is that we can import an *arbitrary* classical binary or quaternary convolutional code for use in a convolutional entanglement distillation protocol. The error-correcting properties for the convolutional entanglement distillation protocol follow directly from the properties of the imported classical code. Thus we can apply the decades of research on classical convolutional coding theory with many of the benefits of the convolutional structure carrying over to the quantum domain.

We organize this chapter as follows. We review stabilizer entanglement distillation in Section 9.1 and entanglement-assisted entanglement distillation in Section 9.2. In Section 9.3, we show how to convert an arbitrary quantum convolutional code into a convolutional entanglement distillation protocol. In Section 9.4, we provide several methods and examples for constructing convolutional

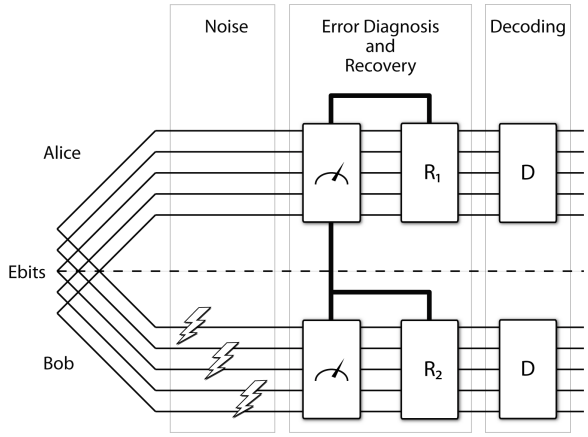

Figure 9.1: An example of a block entanglement distillation protocol. A sender creates a set of noisy ebits by sending half of a set of Bell states through a noisy quantum channel. Both sender and receiver perform multi-qubit measurements to diagnose channel error. The sender transmits her measurement results to the receiver over a classical communication channel. Both perform recovery and decoding operations to obtain a set of noiseless ebits.

entanglement distillation protocols where two parties possess a few initial noiseless ebits. These initial noiseless ebits act as a catalyst for the convolutional distillation protocol. The constructions in Section 9.4 make it possible to import an arbitrary classical binary or quaternary convolutional code for use in convolutional entanglement distillation.

9.1 Stabilizer Entanglement Distillation without Entanglement Assistance

The purpose of an $[n, k]$ entanglement distillation protocol is to distill k pure ebits from n noisy ebits where $0 \leq k \leq n$ [65, 66]. The yield of such a protocol is k/n. Two parties can then use the noiseless ebits for quantum communication protocols. Figure 9.1 illustrates the operation of a block entanglement distillation protocol.

The two parties establish a set of shared noisy ebits in the following way. The sender Alice first prepares n Bell states $|\Phi^+\rangle^{\otimes n}$ locally. She sends the second qubit of each pair over a noisy quantum channel to a receiver Bob. Let $|\Phi_n^+\rangle$ be the state $|\Phi^+\rangle^{\otimes n}$ rearranged so that all of Alice's qubits are on the left and all of Bob's qubits are on the right. The noisy channel applies a Pauli

error in the error set $\mathcal{E} \subset \Pi^n$ to the set of n qubits sent over the channel. The sender and receiver then share a set of n noisy ebits of the form $(\mathbf{I} \otimes \mathbf{A}) \left|\Phi_n^+\right\rangle$ where the identity $\mathbf{I}$ acts on Alice's qubits and $\mathbf{A}$ is some Pauli operator in $\mathcal{E}$ acting on Bob's qubits.

A one-way stabilizer entanglement distillation protocol uses a stabilizer code for the distillation procedure. Figure 9.1 highlights the main features of a stabilizer entanglement distillation protocol. Suppose the stabilizer $\mathcal{S}$ for an $[n, k]$ quantum error-correcting code has generators $g_1, \ldots, g_{n-k}$. The distillation procedure begins with Alice measuring the $n - k$ generators in $\mathcal{S}$. Let $\{\mathbf{P}_i\}$ be the set of the 2^{n-k} projectors that project onto the 2^{n-k} orthogonal subspaces corresponding to the generators in $\mathcal{S}$. The measurement projects $\left|\Phi_n^+\right\rangle$ randomly onto one of the i subspaces. Each $\mathbf{P}_i$ commutes with the noisy operator $\mathbf{A}$ on Bob's side so that

$$(\mathbf{P}_i \otimes \mathbf{I}) (\mathbf{I} \otimes \mathbf{A}) \left|\Phi_n^+\right\rangle = (\mathbf{I} \otimes \mathbf{A}) (\mathbf{P}_i \otimes \mathbf{I}) \left|\Phi_n^+\right\rangle . \tag{9.1}$$

The following important "Bell-state matrix identity" holds for an arbitrary matrix $\mathbf{M}$:

$$(\mathbf{M} \otimes \mathbf{I}) \left|\Phi_n^+\right\rangle = \left(\mathbf{I} \otimes \mathbf{M}^T\right) \left|\Phi_n^+\right\rangle . \tag{9.2}$$

Then (9.1) is equal to the following:

$$(\mathbf{I} \otimes \mathbf{A}) (\mathbf{P}_i \otimes \mathbf{I}) \left|\Phi_n^+\right\rangle = (\mathbf{I} \otimes \mathbf{A}) \left(\mathbf{P}_i^2 \otimes \mathbf{I}\right) \left|\Phi_n^+\right\rangle = (\mathbf{I} \otimes \mathbf{A}) \left(\mathbf{P}_i \otimes \mathbf{P}_i^T\right) \left|\Phi_n^+\right\rangle .$$

Therefore each of Alice's projectors $\mathbf{P}_i$ projects Bob's qubits onto a subspace $\mathbf{P}_i^T$ corresponding to Alice's projected subspace $\mathbf{P}_i$. This operation is one of the "weird" properties of entanglement because it projects the set of noisy ebits onto the codespace effectively "before" the noise acts on it. Alice restores her qubits to the simultaneous $+1$-eigenspace of the generators in $\mathcal{S}$. She sends her measurement results to Bob. Bob measures the generators in $\mathcal{S}$. Bob combines his measurements with Alice's to determine a syndrome for the error. He performs a recovery operation on his qubits to reverse the error. He restores his qubits to the simultaneous $+1$-eigenspace of the generators in $\mathcal{S}$. Alice and Bob both perform the decoding unitary corresponding to stabilizer $\mathcal{S}$ to convert their k logical ebits to k physical ebits.

9.2 Stabilizer Entanglement Distillation with Entanglement Assistance

Luo and Devetak provided a straightforward extension of the above protocol [76]. Their method converts an entanglement-assisted stabilizer code into an entanglement-assisted entanglement distillation protocol.

Luo and Devetak form an entanglement distillation protocol that has entanglement assistance from a few noiseless ebits. The crucial assumption for an entanglement-assisted entanglement distillation protocol is that Alice and Bob possess c noiseless ebits in addition to their n noisy ebits. The total state of the noisy and noiseless ebits is

$$(\mathbf{I}^A \otimes (\mathbf{A} \otimes \mathbf{I})^B) \left|\Phi^+_{n+c}\right\rangle \tag{9.3}$$

where $\mathbf{I}^A$ is the $2^{n+c} \times 2^{n+c}$ identity matrix acting on Alice's qubits and the noisy Pauli operator $(\mathbf{A} \otimes \mathbf{I})^B$ affects Bob's first n qubits only. Thus the last c ebits are noiseless, and Alice and Bob have to correct for errors on the first n ebits only.

The protocol proceeds exactly as outlined in the previous section. The only difference is that Alice and Bob measure the generators in an entanglement-assisted stabilizer code. Each generator spans over $n + c$ qubits where the last c qubits are noiseless.

We comment on the yield of this entanglement-assisted entanglement distillation protocol. An entanglement-assisted code has $n - k$ generators that each have $n + c$ Pauli entries. These parameters imply that the entanglement distillation protocol produces $k+c$ ebits. But the protocol consumes c initial noiseless ebits as a catalyst for distillation. Therefore the yield of this protocol is k/n.

In Section 9.4, we exploit this same idea of using a few noiseless ebits as a catalyst for distillation. The idea is similar in spirit to that developed in this section, but the mathematics and construction are different because we perform distillation in a convolutional manner.

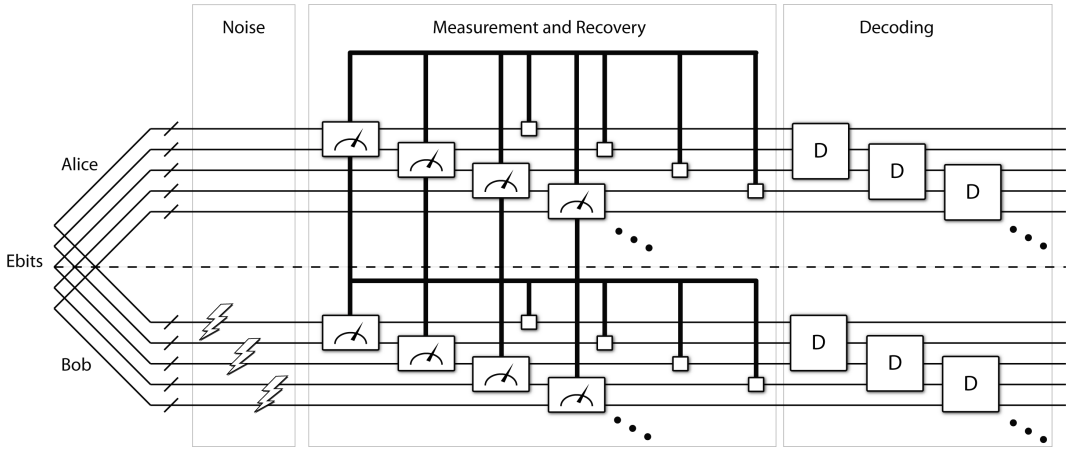

Figure 9.2: An example of a convolutional entanglement distillation protocol taken from the quantum convolutional code in Ref. [1]. The code in Ref. [1] has rate 1/3 and can correct for single-qubit errors in every other frame. Alice and Bob first measure the operators in the stabilizer for the quantum convolutional code. Alice performs conditional unitaries on her qubits to restore them to the +1 eigenspace of the stabilizer code. Alice forwards her measurement results to Bob. Bob performs a maximum-likelihood decoding procedure such as Viterbi decoding [2] to determine the qubit errors. He corrects for these errors. He restores his qubits to the +1 eigenspace of the stabilizer code. Alice and Bob both perform online decoding to obtain ebits with yield 1/3.

9.3 Convolutional Entanglement Distillation without Entanglement Assistance

We now show how to convert an arbitrary quantum convolutional code into a convolutional entanglement distillation protocol. Figure 9.2 illustrates an example of a yield-1/3 convolutional entanglement distillation protocol. The protocol has the same benefits as a quantum convolutional code: an online decoder with less decoding complexity than a block protocol, good error-correcting properties, and higher ebit yield than a block protocol. The protocol we develop in this section is useful for our major contribution presented in the next section.

We can think of our protocol in two ways. Our protocol applies when a sender Alice and a receiver Bob possess a countably infinite number of noisy ebits. Our protocol also applies as an online protocol when Alice and Bob begin with a finite number of noisy ebits and establish more as time passes. The countably infinite and online protocols are equivalent. We would actually implement the entanglement distillation protocol in the online manner, but we formulate the

forthcoming mathematics with the countably infinite description. Each step in the protocol does not need to wait for the completion of its preceding step if Alice and Bob employ the protocol online.

The protocol begins with Alice and Bob establishing a set of noisy ebits. Alice prepares a countably infinite number of Bell states $|\Phi^+\rangle$ locally. She sends one half of each Bell state through a noisy quantum channel. Alice and Bob then possess a state ρ^{AB} that is a countably infinite number of noisy ebits ρ_i^{AB} where

$$\rho^{AB} = \bigotimes_{i=1}^{\infty} \rho_i^{AB}. \tag{9.4}$$

The state ρ^{AB} is equivalent to the following ensemble

$$\left\{ p_i, \left|\Phi^+\right\rangle_i^{AB} \right\}. \tag{9.5}$$

In the above, p_i is the probability that the state is $|\Phi^+\rangle_i^{AB}$, where

$$\left|\Phi^+\right\rangle_i^{AB} \equiv (\mathbf{I} \otimes \mathbf{A}_i) \left|\Phi_\infty^+\right\rangle^{AB}, \tag{9.6}$$

and $|\Phi_\infty^+\rangle^{AB}$ is the state $\left(|\Phi^+\rangle^{AB}\right)^{\otimes\infty}$ rearranged so that all of Alice's qubits are on the left and all of Bob's are on the right. $\mathbf{A}_i \in \Pi^{\mathbb{Z}^+}$ is a Pauli sequence of errors acting on Bob's side. These errors result from the noisy quantum channel. $\mathbf{I}$ is a sequence of identity matrices acting on Alice's side indicating that the noisy channel does not affect her qubits. Alice and Bob need to correct for a particular error set in order to distill noiseless ebits.

Alice and Bob employ the following strategy to distill noiseless ebits. Alice measures the $n-k$ generators in the basic set $\mathcal{G}_0$. The measurement operation projects the first $n(\nu+1)$ ebits (ν is the constraint length) randomly onto one of 2^{n-k} orthogonal subspaces. Alice places the measurement outcomes in an $(n-k)$-dimensional classical bit vector $\mathbf{a}_0$. She restores her half of the noisy ebits to the simultaneous $+1$-eigenspace of the generators in $\mathcal{G}_0$ if $\mathbf{a}_0$ differs from the all-zero vector. She sends $\mathbf{a}_0$ to Bob over a classical communication channel. Bob measures the generators in $\mathcal{G}_0$ and stores the measurement outcomes in a classical bit vector $\mathbf{b}_0$. Bob compares $\mathbf{b}_0$ to $\mathbf{a}_0$ by calculating an error vector $\mathbf{e}_0 = \mathbf{a}_0 \oplus \mathbf{b}_0$. He corrects for any errors that $\mathbf{e}_0$ can identify. He may have to wait to receive later error vectors before determining the full error syndrome. He

restores his half of the noisy ebits to the simultaneous +1-eigenspace of the generators in $\mathcal{G}_0$ if the bit vector $\mathbf{b}_0$ indicates that his logical ebits are not in the +1-space. Alice and Bob repeat the above procedure for all shifts $D(\mathcal{G}_0)$, $D^2(\mathcal{G}_0)$, ... of the basic generators in $\mathcal{G}_0$. Bob obtains a set $\mathcal{E}$ of classical error vectors $\mathbf{e}_i$: $\mathcal{E} = \{\mathbf{e}_i : i \in \mathbb{Z}^+\}$. Bob uses a maximum-likelihood decoding technique such as Viterbi decoding [2] or a table-lookup on the error set $\mathcal{E}$ to determine which errors occur. This error determination process is a purely classical computation. He reverses the estimated errors after determining the syndrome.

The states that Alice and Bob possess after the above procedure are encoded logical ebits. They can extract physical ebits from these logical ebits by each performing the online decoding circuit for the code $\mathcal{G}$. The algorithm outlined in Ref. [45] gives a method for determining the online decoding circuit.

Example 9.3.1. We use the rate-1/3 quantum convolutional code in Example 4.1.1 to produce a yield-1/3 convolutional entanglement distillation protocol. Alice measures the generators in the stabilizer in (4.8) for every noisy ebit she shares with Bob. Alice communicates the result of her measurement of the first two generators to Bob. Alice restores the qubits on her side to be in the simultaneous +1-eigenspace of the first two generators. Bob measures the same first two generators. Alice measures the next two generators, communicates her results, etc. Bob compares his results to Alice's to determine the error bit vectors. Bob performs Viterbi decoding on the measurement results and corrects for errors. He rotates his states to the simultaneous +1-eigenspace of the generators. Alice and Bob perform the above procedure in an online manner according to Figure 9.2. Alice and Bob can decode the first six qubits after measuring the second two generators. They can decode because there is no overlap between the first two generators and any two generators after the second two generators. They use the circuit from [45] in reverse order to decode physical ebits from logical ebits. They distill ebits with yield 1/3 by using this convolutional entanglement distillation protocol. The ebit yield of 1/3 follows directly from the code rate of 1/3.

9.4 Convolutional Entanglement Distillation with Entanglement Assistance

The convolutional entanglement distillation protocol that we develop in this section operates identically to the one developed in the previous section. The measurements, classical communication, and recovery and decoding operations proceed exactly as Figure 9.2 indicates.

The difference between the protocol in this section and the previous one is that we now assume the sender and receiver share a few initial noiseless ebits. They use these initial ebits as a catalyst to get the protocol started. The sender and receiver require noiseless ebits for each round of the convolutional entanglement distillation protocol. They can use the noiseless ebits generated by earlier rounds for consumption in later rounds. It is possible to distill noiseless ebits in this way by catalyzing the process with a few noiseless ebits. The protocol we develop in this section is a more powerful generalization of the previous section's protocol.

The construction in this section allows sender and receiver to use an arbitrary set of Paulis for the distillation protocol. The set does not necessarily have to be a commuting set of Paulis.

The implication of the construction in this section is that we can import an arbitrary binary or quaternary classical convolutional code for use as a quantum convolutional code. We explicitly give some examples to highlight the technique for importing. The error-correcting properties and yield translate directly from the properties of the classical convolutional code. Thus the problem of finding a good convolutional entanglement distillation protocol reduces to that of finding a good classical convolutional code.

9.4.1 Yield (n-1)/n Convolutional Entanglement Distillation

We present our first method for constructing a convolutional entanglement distillation protocol that uses entanglement assistance. The shifted symplectic product from Section 4.3 is a crucial component of our formulation.

Suppose Alice and Bob use one generator $\mathbf{N}(\mathbf{u}(D))$ for an entanglement distillation protocol where

$$\mathbf{u}(D) = [\mathbf{z}(D)|\mathbf{x}(D)] = \left[\begin{array}{ccc|ccc} z_1(D) & \cdots & z_n(D) & x_1(D) & \cdots & x_n(D) \end{array} \right].$$

and where $z_1(D)$, ..., $z_n(D)$, $x_1(D)$, ..., $x_n(D)$ are binary polynomials. We do not impose a commuting constraint on generator $\mathbf{N}(\mathbf{u}(D))$. Alice and Bob choose generator $\mathbf{N}(\mathbf{u}(D))$ solely for its error-correcting capability.

The shifted symplectic product helps to produce a commuting generator from a noncommuting one. The shifted symplectic product of $\mathbf{u}(D)$ is

$$(\mathbf{u} \odot \mathbf{u})(D) = \sum_{i \in \mathbb{Z}} (\mathbf{u} \odot \mathbf{u})_i \ D^i. \tag{9.7}$$

The coefficient $(\mathbf{u} \odot \mathbf{u})_0$ for zero shifts is equal to zero because every tensor product of Pauli operators commutes with itself:

$$(\mathbf{u} \odot \mathbf{u})_0 = 0. \tag{9.8}$$

Recall that $\mathbf{u}(D)$ is self-time-reversal symmetric (4.32). We adopt the following notation for a polynomial that includes the positive-index or negative-index coefficients of the shifted symplectic product $(\mathbf{u} \odot \mathbf{u})(D)$:

$$(\mathbf{u} \odot \mathbf{u})(D)^+ = \sum_{i \in \mathbb{Z}^+} (\mathbf{u} \odot \mathbf{u})_i \ D^i, \quad (\mathbf{u} \odot \mathbf{u})(D)^- = \sum_{i \in \mathbb{Z}^-} (\mathbf{u} \odot \mathbf{u})_i \ D^i. \tag{9.9}$$

The following identity holds:

$$(\mathbf{u} \odot \mathbf{u})(D)^+ = (\mathbf{u} \odot \mathbf{u})\left(D^{-1}\right)^-. \tag{9.10}$$

Consider the following vector of polynomials:

$$\mathbf{a}(D) = \left[\ (\mathbf{u} \odot \mathbf{u})(D)^+ \ \middle| \ 1 \ \right]. \tag{9.11}$$

Its relations under the shifted symplectic product are the same as $\mathbf{u}(D)$:

$$(\mathbf{a} \odot \mathbf{a})(D) = (\mathbf{u} \odot \mathbf{u})(D)^+ + (\mathbf{u} \odot \mathbf{u})(D)^- = (\mathbf{u} \odot \mathbf{u})(D).$$

X_1	Z_1	Y_1	X_2	Z_2	Y_2
1	0	1	1	0	1
0	0	0	0	1	1
0	1	1	1	0	1
1	0	1	0	0	0

Table 9.1: The convolutional entanglement distillation protocol for Example 9.4.1 corrects for a single-qubit error in every fourth frame. Here we list the syndromes corresponding to errors X_1, Y_1, and Z_1 on the first qubit and to errors X_2, Y_2, and Z_2 on the second qubit. The syndromes are unique so that the receiver can identify which error occurs.

The vector $\mathbf{a}(D)$ provides a straightforward way to make $\mathbf{N}(\mathbf{u}(D))$ commute with all of its shifts. We augment $\mathbf{u}(D)$ with $\mathbf{a}(D)$. The augmented generator $\mathbf{u}'(D)$ is as follows:

$$\mathbf{u}'(D) = \left[\begin{array}{cc|cc} \mathbf{z}(D) & (\mathbf{u} \odot \mathbf{u})(D)^+ & \mathbf{x}(D) & 1 \end{array} \right]. \tag{9.12}$$

The augmented generator $\mathbf{u}'(D)$ has vanishing symplectic product, $(\mathbf{u}' \odot \mathbf{u}')(D) = 0$, because the shifted symplectic product of $\mathbf{a}(D)$ nulls the shifted symplectic product of $\mathbf{u}(D)$. The augmented generator $\mathbf{N}(\mathbf{u}'(D))$ commutes with itself for every shift and is therefore useful for convolutional entanglement distillation as outlined in Section 9.3.

We can construct an entanglement distillation protocol using an augmented generator of this form. The first n Pauli entries for every frame of generator $\mathbf{N}(\mathbf{u}'(D))$ correct errors. Entry $n+1$ for every frame of $\mathbf{N}(\mathbf{u}'(D))$ makes $\mathbf{N}(\mathbf{u}'(D))$ commute with every one of its shifts. The error-correcting properties of the code do not include errors on the last (extra) ebit of each frame; therefore, this ebit must be noiseless. It is necessary to catalyze the distillation procedure with $n\nu$ noiseless ebits where n is the frame size and ν is the constraint length. The distillation protocol requires this particular amount because it does not correct errors and generate noiseless ebits until it has finished processing the first basic set of generators and $\nu - 1$ of its shifts. Later frames can use the noiseless ebits generated from previous frames. Therefore these initial noiseless ebits are negligible when calculating the yield. This construction allows us to exploit the error-correcting properties of an arbitrary set of Pauli matrices for a convolutional entanglement distillation protocol.

We discuss the yield of such a protocol in more detail. Our construction employs one generator with $n+1$ qubits per frame. The protocol generates n noiseless ebits for every frame. But it also consumes a noiseless ebit for every frame. Every frame thus produces a net of $n-1$ noiseless ebits, and the yield of the protocol is $(n-1)/n$.

This yield of $(n-1)/n$ is superior to the yield of an entanglement distillation protocol taken from the quantum convolutional codes of Forney et al. [1]. Our construction should also give entanglement distillation protocols with superior error-correcting properties because we have no self-orthogonality constraint on the Paulis in the stabilizer.

It is possible to construct an online decoding circuit for the generator $\mathbf{u}'(D)$ by the methods given in [45]. A circuit satisfies the noncatastrophic property if the polynomial entries of all of the code generators have a greatest common divisor that is a power of the delay operator D [45]. The online decoding circuit for this construction obeys the noncatastrophicity property because the augmented generator $\mathbf{u}'(D)$ contains 1 as one of its entries.

Example 9.4.1. Suppose we have the following generator

$$\mathbf{N}(\mathbf{u}(D)) = (\cdots|II|ZZ|IX|XZ|ZI|II|\cdots),$$

where

$$\mathbf{u}(D) = \left[\begin{array}{cc|cc} 1+D^3 & 1+D^2 & D^2 & D \end{array}\right].$$

The above generator corrects for an arbitrary single-qubit error in every fourth frame. Table 9.4.1 lists the unique syndromes for errors in a single frame. The generator anticommutes with a shift of itself by one or two to the left or right. The shifted symplectic product confirms these commutation relations:

$$(\mathbf{u} \odot \mathbf{u})(D) = D + D^2 + D^{-1} + D^{-2}.$$

Let us follow the prescription in (9.12) for augmenting generator $\mathbf{N}(\mathbf{u}(D))$. The following polynomial

$$\mathbf{a}(D) = \left[\begin{array}{c|c} (\mathbf{u} \odot \mathbf{u})(D)^+ & 1 \end{array}\right] = \left[\begin{array}{c|c} D + D^2 & 1 \end{array}\right],$$

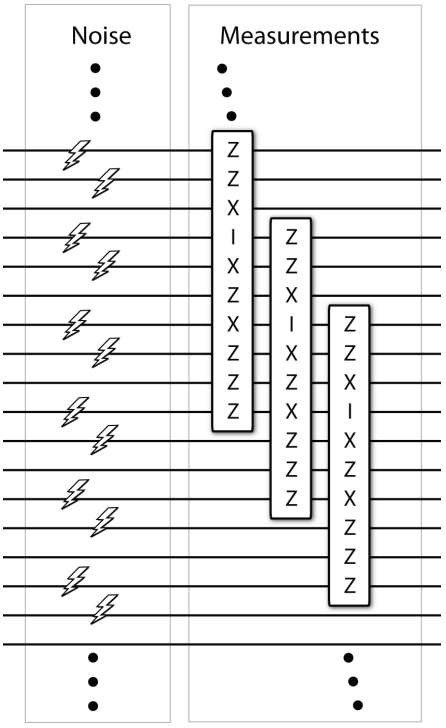

Figure 9.3: The above figure illustrates Bob's side of the convolutional entanglement distillation protocol that uses entanglement assistance. The noise affects the first and second of every three ebits that Bob shares with Alice. Every third ebit that Alice and Bob share are noiseless. The measurements correspond to those in Example 9.4.1.

has the same commutation relations as $\mathbf{u}(D)$:

$$(\mathbf{a} \odot \mathbf{a})(D) = (\mathbf{u} \odot \mathbf{u})(D). \tag{9.13}$$

We augment $\mathbf{u}(D)$ as follows:

$$\mathbf{u}'(D) = \left[\begin{array}{ccc|ccc} 1+D^3 & 1+D^2 & D+D^2 & D^2 & D & 1 \end{array}\right].$$

The overall generator now looks as follows in the Pauli representation:

$$\mathbf{N}(\mathbf{u}'(D)) = (\cdots|III|ZZX|IXZ|XZZ|ZII|III|\cdots).$$

The yield of a protocol using the above construction is 1/2. Figure 9.3 illustrates Bob's side of the protocol. It shows which of Bob's half of the ebits are noisy and noiseless, and it gives the measurements that Bob performs.

9.4.2 Yield (n-m)/n Convolutional Entanglement Distillation

The construction in the above section uses only one generator for distillation. We generalize the above construction to a code with an arbitrary number of generators. We give an example that illustrates how to convert an arbitrary classical quaternary convolutional code into a convolutional entanglement distillation protocol.

Suppose we have the following m generators

$$\{\mathbf{N}(\mathbf{u}_i(D)) : 1 \leq i \leq m\},$$

where

$$\left[\begin{array}{c} \mathbf{u}_1(D) \\ \mathbf{u}_2(D) \\ \vdots \\ \mathbf{u}_m(D) \end{array}\right] = \left[\begin{array}{c|c} \mathbf{z}_1(D) & \mathbf{x}_1(D) \\ \mathbf{z}_2(D) & \mathbf{x}_2(D) \\ \vdots & \vdots \\ \mathbf{z}_m(D) & \mathbf{x}_m(D) \end{array}\right]. \tag{9.14}$$

We make no assumption about the commutation relations of the above generators. We choose them solely for their error-correcting properties.

We again utilize the shifted symplectic product to design a convolutional entanglement distillation protocol with multiple generators. Let us adopt the following shorthand for the auto and cross shifted symplectic products of generators $\mathbf{u}_1(D), \ldots, \mathbf{u}_m(D)$:

$$\mathbf{u}_i^+ \equiv (\mathbf{u}_i \odot \mathbf{u}_i)(D)^+, \tag{9.15}$$

$$\mathbf{u}_{i,j} \equiv (\mathbf{u}_i \odot \mathbf{u}_j)(D). \tag{9.16}$$

Consider the following matrix:

$$\begin{bmatrix} \mathbf{a}_1(D) \\ \mathbf{a}_2(D) \\ \vdots \\ \mathbf{a}_m(D) \end{bmatrix} = \left[\begin{array}{cccc|c} \mathbf{u}_1^+ & \mathbf{u}_{1,2} & \cdots & \mathbf{u}_{1,m} & \\ 0 & \mathbf{u}_2^+ & \cdots & \mathbf{u}_{2,m} & \\ \vdots & & \ddots & \vdots & \mathbf{I}_{m\times m} \\ 0 & \cdots & 0 & \mathbf{u}_m^+ & \end{array}\right]. \tag{9.17}$$

The symplectic relations of the entries $\mathbf{a}_i(D)$ are the same as the original $\mathbf{u}_i(D)$:

$$(\mathbf{a}_i \odot \mathbf{a}_j)(D) = (\mathbf{u}_i \odot \mathbf{u}_j)(D) \qquad \forall i, j \in \{1, \ldots, m\},$$

or equivalently, if $\Omega_{\mathbf{a}}(D)$ is the shifted symplectic product matrix for the generators in (9.17) and $\Omega_{\mathbf{u}}(D)$ is the shifted symplectic product matrix for the "$\mathbf{u}$" generators, then

$$\Omega_{\mathbf{a}}(D) = \Omega_{\mathbf{u}}(D).$$

We mention that the following matrix also has the same symplectic relations:

$$\left[\begin{array}{cccc|c} \mathbf{u}_1^+ & 0 & \cdots & 0 & \\ \mathbf{u}_{2,1} & \mathbf{u}_2^+ & \cdots & \vdots & \\ \vdots & & \ddots & 0 & \mathbf{I}_{m\times m} \\ \mathbf{u}_{m,1} & \mathbf{u}_{m,2} & \cdots & \mathbf{u}_m^+ & \end{array}\right]. \tag{9.18}$$

Let us rewrite (9.17) as follows:

$$\left[\begin{array}{c} \mathbf{a}_1(D) \\ \mathbf{a}_2(D) \\ \vdots \\ \mathbf{a}_m(D) \end{array}\right] = \left[\begin{array}{c|c} \mathbf{z}'_1(D) & \mathbf{x}'_1(D) \\ \mathbf{z}'_2(D) & \mathbf{x}'_2(D) \\ \vdots & \vdots \\ \mathbf{z}'_m(D) & \mathbf{x}'_m(D) \end{array}\right]. \tag{9.19}$$

The above matrix provides a straightforward way to make the original generators commute with all of their shifts. We augment the generators in (9.14) by the generators $\mathbf{a}_i(D)$ to get the following $m \times 2(n+m)$ matrix:

$$\mathbf{U}'(D) = \left[\begin{array}{c|c} \mathbf{Z}(D) & \mathbf{X}(D) \end{array}\right] = \left[\begin{array}{cc|cc} \mathbf{z}_1(D) & \mathbf{z}'_1(D) & \mathbf{x}_1(D) & \mathbf{x}'_1(D) \\ \mathbf{z}_2(D) & \mathbf{z}'_2(D) & \mathbf{x}_2(D) & \mathbf{x}'_2(D) \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{z}_m(D) & \mathbf{z}'_m(D) & \mathbf{x}_m(D) & \mathbf{x}'_m(D) \end{array}\right].$$

Every row of the augmented matrix $\mathbf{U}'(D)$ has vanishing symplectic product with itself and any other row. This condition is equivalent to the following matrix condition for shifted symplectic orthogonality [44]:

$$\mathbf{Z}(D)\,\mathbf{X}^T\left(D^{-1}\right) - \mathbf{X}(D)\,\mathbf{Z}^T\left(D^{-1}\right) = 0. \tag{9.20}$$

The construction gives a commuting set of generators for arbitrary shifts and thus forms a valid stabilizer.

We can readily develop a convolutional entanglement distillation protocol using the above formulation. The generators in the augmented matrix $\mathbf{U}'(D)$ correct for errors on the first n ebits. The last m ebits are noiseless ebits that help to obtain a commuting stabilizer. It is necessary to catalyze the distillation protocol with $(n+m)\,\nu$ noiseless ebits. Later frames can use the noiseless ebits generated from previous frames. These initial noiseless ebits are negligible when calculating the yield.

We comment more on the yield of the protocol. The protocol requires a set of m generators with $n+m$ Pauli entries. It generates n ebits for every frame. But it consumes m noiseless ebits per frame. The net yield of a protocol using the above construction is thus $(n-m)/n$.

The key benefit of the above construction is that we can use an arbitrary set of Paulis for distilling noiseless ebits. This arbitrariness in the Paulis implies that we can import an arbitrary classical convolutional binary or quaternary code for use in a convolutional entanglement distillation protocol.

It is again straightforward to develop a noncatastrophic decoding circuit using previous techniques [45]. Every augmented generator in $\mathbf{U}'(D)$ has "1" as an entry so that it satisfies the property required for noncatastrophicity.

Example 9.4.2. We begin with a classical quaternary convolutional code with entries from $GF(4)$:

$$(\cdots|0000|1\bar{\omega}10|1101|0000|\cdots). \tag{9.21}$$

The above code is a convolutional version of the classical quaternary block code from Ref. [34]. We multiply the above generator by $\bar{\omega}$ and ω as prescribed in Refs. [19, 1] and use the map in (3.38) to obtain the following Pauli generators

$$\begin{aligned}\mathbf{N}(\mathbf{u}_1(D)) &= (\cdots|IIII|ZXZI|ZZIZ|IIII|\cdots),\\ \mathbf{N}(\mathbf{u}_2(D)) &= (\cdots|IIII|XYXI|XXIX|IIII|\cdots).\end{aligned} \tag{9.22}$$

We determine binary polynomials corresponding to the above Pauli generators:

$$\left[\begin{array}{c}\mathbf{u}_1(D)\\ \mathbf{u}_2(D)\end{array}\right] = \left[\begin{array}{cccc|cccc} 1+D & D & 1 & D & 0 & 1 & 0 & 0\\ 0 & 1 & 0 & 0 & 1+D & 1+D & 1 & D\end{array}\right]. \tag{9.23}$$

The first generator anticommutes with itself shifted by one to the left or right, the second generator anticommutes with itself shifted by one to the left or right, and the first generator anticommutes with the second shifted by one to the left. The following shifted symplectic products confirm the above commutation relations:

$$(\mathbf{u}_1 \odot \mathbf{u}_1)(D) = D^{-1}+D, \quad (\mathbf{u}_2 \odot \mathbf{u}_2)(D) = D^{-1}+D, \quad (\mathbf{u}_1 \odot \mathbf{u}_2)(D) = D^{-1}. \tag{9.24}$$

Consider the following two generators:

$$\left[\begin{array}{c} \mathbf{a}_1(D) \\ \mathbf{a}_2(D) \end{array}\right] = \left[\begin{array}{cc|cc} D & 0 & 1 & 0 \\ D & D & 0 & 1 \end{array}\right]. \tag{9.25}$$

Their relations under the shifted symplectic product are the same as those in (9.24).

$$(\mathbf{a}_1 \odot \mathbf{a}_1)(D) = (\mathbf{u}_1 \odot \mathbf{u}_1)(D), \quad (\mathbf{a}_2 \odot \mathbf{a}_2)(D) = (\mathbf{u}_2 \odot \mathbf{u}_2)(D), \quad (\mathbf{a}_1 \odot \mathbf{a}_2)(D) = (\mathbf{u}_1 \odot \mathbf{u}_2)(D).$$

We augment the generators $\mathbf{u}_1(D)$ and $\mathbf{u}_2(D)$ to generators $\mathbf{u}'_1(D)$ and $\mathbf{u}'_2(D)$ respectively as follows. The augmented matrix $\mathbf{U}'(D)$ is

$$\mathbf{U}'(D) = \left[\begin{array}{cccccc|cccccc} 1+D & D & 1 & D & D & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & D & D & 1+D & 1+D & 1 & D & 0 & 1 \end{array}\right]. \tag{9.26}$$

The first row of $\mathbf{U}'(D)$ is generator $\mathbf{u}'_1(D)$ and the second row is $\mathbf{u}'_2(D)$. The augmented generators have the following Pauli representation.

$$\begin{aligned} \mathbf{N}(\mathbf{u}'_1(D)) &= (\cdots|IIIIII|ZXZIXI|ZZIZZI|IIIIII|\cdots), \\ \mathbf{N}(\mathbf{u}'_2(D)) &= (\cdots|IIIIII|XYXIIX|XXIXZZ|IIIIII|\cdots). \end{aligned} \tag{9.27}$$

The original block code from Ref. [34] corrects for an arbitrary single-qubit error. The above entanglement distillation protocol corrects for a single-qubit error in eight qubits—two frames. This error-correcting capability follows from the capability of the block code. The yield of a protocol using the above stabilizer is again 1/2.

9.4.3 CSS-Like Construction for Convolutional Entanglement Distillation

We finally present a construction that allows us to import two arbitrary binary classical codes for use in a convolutional entanglement distillation protocol. The construction is similar to a CSS code because one code corrects for bit flips and the other corrects for phase flips.

We could simply use the technique from the previous section to construct a convolutional entanglement-distillation protocol. We could represent both classical codes as codes over $GF(4)$. We could multiply the bit-flip code by ω and the phase-flip code by $\bar{\omega}$ and use the map in (3.38) from $GF(4)$ to the Paulis. We could then use the above method for augmentation and obtain a valid quantum code for entanglement distillation. But there is a better method that exploits the structure of a CSS code to minimize the number of initial catalytic noiseless ebits.

Our algorithm below uses a Gram-Schmidt like orthogonalization procedure to minimize the number of initial noiseless ebits. The procedure is similar to the algorithm in [35] with some key differences.

Suppose we have m generators $\{\mathbf{N}(\mathbf{w}_i(D)) : 1 \leq i \leq m\}$ where

$$\left[\begin{array}{c} \mathbf{w}_1(D) \\ \vdots \\ \mathbf{w}_p(D) \\ \mathbf{w}_{p+1}(D) \\ \vdots \\ \mathbf{w}_m(D) \end{array}\right] = \left[\begin{array}{c|c} \mathbf{z}_1(D) & \mathbf{0} \\ \vdots & \vdots \\ \mathbf{z}_p(D) & \mathbf{0} \\ \mathbf{0} & \mathbf{x}_1(D) \\ \vdots & \vdots \\ \mathbf{0} & \mathbf{x}_{m-p}(D) \end{array}\right]. \tag{9.28}$$

and each vector $\mathbf{w}_i(D)$ has length $2n$. The above matrix could come from two binary classical codes. The vectors $\mathbf{z}_1(D),\ldots,\mathbf{z}_p(D)$ could come from one code, and the vectors $\mathbf{x}_1(D),\ldots,\mathbf{x}_{m-p}(D)$ could come from another code. The following orthogonality relations hold for the above vectors:

$$\forall \;\; 1 \leq i,j \leq p : (\mathbf{w}_i \odot \mathbf{w}_j)(D) = 0, \tag{9.29}$$

$$\forall \;\; p+1 \leq i',j' \leq m : (\mathbf{w}_{i'} \odot \mathbf{w}_{j'})(D) = 0. \tag{9.30}$$

We exploit the above orthogonality relations in the algorithm below.

We can perform a Gram-Schmidt process on the above set of vectors. This process orthogonalizes the vectors with respect to the shifted symplectic product. The procedure does not change the error-correcting properties of the original codes because all operations are linear.

The algorithm breaks the set of vectors above into pairs. Each pair consists of two vectors which are symplectically nonorthogonal to each other, but which are symplectically orthogonal

to all other pairs. Any remaining vectors that are symplectically orthogonal to all other vectors are collected into a separate set, which we call the set of isotropic vectors. This idea is similar to the decomposition of a vector space into an isotropic and symplectic part. We cannot label the decomposition as such because the shifted symplectic product is not a true symplectic product.

We detail the initialization of the algorithm. Set parameters $i = 0$, $c = 0$, $l = 0$. The index i labels the total number of vectors processed, c gives the number of pairs, and l labels the number of vectors with no partner. Initialize sets $\mathcal{U}$ and $\mathcal{V}$ to be null: $\mathcal{U} = \mathcal{V} = \emptyset$. $\mathcal{U}$ keeps track of the pairs and $\mathcal{V}$ keeps track of the vectors with no partner.

The algorithm proceeds as follows. While $i \leq m$, let $j \geq 2c + l + 2$ be the smallest index for a $\mathbf{w}_j(D)$ for which $(\mathbf{w}_{2c+l+1} \odot \mathbf{w}_j)(D) \neq 0$. Increment l and i by one, add i to $\mathcal{V}$, and proceed to the next round if no such pair exists. Otherwise, swap $\mathbf{w}_j(D)$ with $\mathbf{w}_{2c+l+2}(D)$. For $r \in \{2c + l + 3, \ldots, m\}$, perform

$$\mathbf{w}_r(D) = (\mathbf{w}_{2c+l+2} \odot \mathbf{w}_{2c+l+1})(D)\,\mathbf{w}_r(D) + (\mathbf{w}_r \odot \mathbf{w}_{2c+l+2})\left(D^{-1}\right)\mathbf{w}_{2c+l+1}(D),$$

if $\mathbf{w}_r(D)$ has a purely z component. Perform

$$\mathbf{w}_r(D) = (\mathbf{w}_{2c+l+1} \odot \mathbf{w}_{2c+l+2})(D)\,\mathbf{w}_r(D) + (\mathbf{w}_r \odot \mathbf{w}_{2c+l+1})\left(D^{-1}\right)\mathbf{w}_{2c+l+2}(D),$$

if $\mathbf{w}_r(D)$ has a purely x component. Divide every element in $\mathbf{w}_r(D)$ by the greatest common factor if the GCF is not equal to one. Then

$$(\mathbf{w}_r \odot \mathbf{w}_{2c+l+1})(D) = (\mathbf{w}_r \odot \mathbf{w}_{2c+l+2})(D) = 0. \tag{9.31}$$

Increment c by one, increment i by one, add i to $\mathcal{U}$, and increment i by one. Proceed to the next round.

We now give the method for augmenting the above generators so that they form a commuting stabilizer. At the end of the algorithm, the sets $\mathcal{U}$ and $\mathcal{V}$ have the following sizes: $|\mathcal{U}| = c$ and $|\mathcal{V}| = l$. Let us relabel the vectors $\mathbf{w}_i(D)$ for all $1 \leq i \leq 2c + l$. We relabel all pairs: call the first $\mathbf{u}_i(D)$ and call its partner $\mathbf{v}_i(D)$ for all $1 \leq i \leq c$. Call any vector without a partner $\mathbf{u}_{c+i}(D)$ for

all $1 \leq i \leq l$. The relabeled vectors have the following shifted symplectic product relations after the Gram-Schmidt procedure:

$$\begin{aligned}
(\mathbf{u}_i \odot \mathbf{v}_j)(D) &= f_i(D)\,\delta_{ij} \quad \forall \;\; i,j \in \{1,\ldots,c\}, \\
(\mathbf{u}_i \odot \mathbf{u}_j)(D) &= 0 \qquad\qquad \forall \;\; i,j \in \{1,\ldots,l\}, \\
(\mathbf{v}_i \odot \mathbf{v}_j)(D) &= 0 \qquad\qquad \forall \;\; i,j \in \{1,\ldots,c\},
\end{aligned} \tag{9.32}$$

where $f_i(D)$ is an arbitrary polynomial. Let us arrange the above generators in a matrix as follows:

$$\left[\mathbf{u}_1^T(D) \quad \cdots \quad \mathbf{u}_c^T(D) \quad \mathbf{v}_1^T(D) \quad \cdots \quad \mathbf{v}_c^T(D) \quad \mathbf{u}_{c+1}^T(D) \quad \cdots \quad \mathbf{u}_{c+l}^T(D)\right]^T. \tag{9.33}$$

We augment the above generators with the following matrix so that all vectors are orthogonal to each other:

$$\left[\begin{array}{cccc|c}
f_1(D) & 0 & \cdots & 0 & \mathbf{0}_{1\times c} \\
0 & f_2(D) & & \vdots & \mathbf{0}_{1\times c} \\
\vdots & & \ddots & 0 & \vdots \\
0 & \cdots & 0 & f_c(D) & \mathbf{0}_{1\times c} \\
\mathbf{0}_{c\times 1} & \mathbf{0}_{c\times 1} & \cdots & \mathbf{0}_{c\times 1} & \mathbf{I}_{c\times c} \\
\mathbf{0}_{l\times 1} & \mathbf{0}_{l\times 1} & \cdots & \mathbf{0}_{l\times 1} & \mathbf{0}_{l\times c}
\end{array}\right]. \tag{9.34}$$

The yield of a protocol using the above construction is $(n-m)/n$. Suppose we use an $[n, k_1]$ classical binary convolutional code for the bit flips and an $[n, k_2]$ classical binary convolutional code for the phase flips. Then the convolutional entanglement distillation protocol has yield $(k_1 + k_2 - n)/n$.

Example 9.4.3. Consider a binary classical convolutional code with the following parity check matrix:

$$\left[1+D \quad D \quad 1\right]. \tag{9.35}$$

We can use the above parity check matrix to correct both bit and phase flip errors in an entanglement distillation protocol. Our initial quantum parity check matrix is

$$\left[\begin{array}{ccc|ccc} 1+D & D & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1+D & D & 1 \end{array}\right]. \tag{9.36}$$

The shifted symplectic product for the first and second row is $D^{-1}+D$. We therefore augment the above matrix as follows:

$$\left[\begin{array}{cccc|cccc} 1+D & D & 1 & D^{-1}+D & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1+D & D & 1 & 1 \end{array}\right]. \tag{9.37}$$

The above matrix gives a valid stabilizer for use in an entanglement distillation protocol. The yield of a protocol using the above stabilizer is 1/3.

9.5 Closing Remarks

We constructed a theory of convolutional entanglement distillation. The entanglement-assisted protocol assumes that the sender and receiver have some noiseless ebits to use as a catalyst for distilling more ebits. These protocols have the benefit of lifting the self-orthogonality constraint. Thus we are able to import an arbitrary classical convolutional code for use in a convolutional entanglement distillation protocol. The error-correcting properties and rate of the classical code translate to the quantum case. Brun, Devetak, and Hsieh first constructed the method for importing an arbitrary classical block code in their work on entanglement-assisted codes [35, 34]. Our theory of convolutional entanglement distillation paves the way for exploring protocols that approach the optimal distillable entanglement by using the well-established theory of classical convolutional coding.

Convolutional entanglement distillation protocols also hold some key advantages over block entanglement distillation protocols. They have a higher yield of ebits, lower decoding complexity, and are an online protocol that a sender and receiver can employ as they acquire more noisy ebits.

We suggest that convolutional entanglement distillation protocols may bear some advantages for distillation of a secret key because of the strong connection between distillation and privacy

[86]. We are currently investigating whether convolutional entanglement distillation protocols can improve the secret key rate for quantum key distribution.

Chapter 10

Conclusion

Fly Photon! Trap Ion! Stay Spin!
Which of you we'll implement in?
For quantum coherence
Demands perseverance,
We'll try and God only knows when.

Quantum error correction theory plays a fundamental role in quantum computing and communication. Without error-correcting protocols, quantum computing and communication devices will fall prey to the hands of decoherence. It is crucial for theorists to continue developing techniques to protect quantum information because a fundamental discovery in the theory might bring quantum computing closer to reality.

We have augmented the theory of quantum error correction by contributing a theory of entanglement-assisted quantum convolutional coding. With the ability to import arbitrary classical convolutional codes, we can now construct quantum convolutional codes that inherit the desirable characteristics of their ancestral classical convolutional codes. Our entanglement-assisted quantum convolutional coding theory should be useful for future quantum communication engineers if they would like to have codes with a good performance/complexity trade-off.

We have said and again stress that the next important line of investigation is to combine the quantum turbo coding theory [58] with this theory. Convolutional codes form the constituent codes of a quantum turbo code and it would be interesting to investigate if we can enhance performance

with entanglement-assisted codes as the constituent codes. This investigation might produce quantum codes that come close to achieving the entanglement-assisted or "father" capacity.

There are also other avenues to pursue—these avenues include any scenario where entanglement assistance might help. It is useful to inspect the results of quantum Shannon theory to determine whether we can construct a coding scenario that fits with the constructions in the asymptotic scenario. We have conducted little analysis of the performance of our codes beyond stating that they inherit the properties of the imported classical codes. It would be interesting to observe the performance of these codes in a realistic noisy quantum channel when using syndrome-based Viterbi processing for correction of quantum error.

We have seen much creativity in the field of quantum error correction in the past decade because of the many strange resources available in quantum theory. Experimentalists are increasingly using an array of quantum error correction techniques with the goal of bringing us closer to having qubits with good quantum coherence. One can only imagine what resources future quantum coding theorists will exploit to protect their valuable quantum information.

BIBLIOGRAPHY

[1] G. David Forney, Markus Grassl, and Saikat Guha. Convolutional and tail-biting quantum error-correcting codes. *IEEE Transactions on Information Theory*, 53:865–880, 2007.

[2] Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269, 1967.

[3] 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, Mar 1993.

[4] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, Los Alamitos, CA, 1994. IEEE Computer Society Press.

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

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

[7] Paul Benioff. The computer as a physical system: A macroscopic quantum mechanical hamiltonian model of computers as represented by turing machines. *Journal of Statistical Physics*, 22(5):563–591, May 1980.

[8] Yuri Manin. Computable and uncomputable. *Sovetskoye Radio*, 1980.

[9] Seth Lloyd. Universal quantum simulators. *Science*, 273(5278):1073–1078, 1996.

[10] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC) (arXiv:quant-ph/9605043)*, pages 212–219, 1996.

[11] Lov K. Grover. Quantum mechanics helps in searching for a needle in a haystack. *Physical Review Letters*, 79(2):325–328, July 1997.

[12] Charles H. Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. In *Proceedings of IEEE International Conference on Computers Systems and Signal Processing*, pages 175–179, Bangalore, India, December 1984.

[13] Charles H. Bennett and Stephen J. Wiesner. Communication via one- and two-particle operators on einstein-podolsky-rosen states. *Physical Review Letters*, 69(20):2881–2884, Nov 1992.

[14] Rolf Landauer. Uncertainty principle and minimal energy dissipation in the computer. *International Journal of Theoretical Physics*, 21:283–297, 1982.

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

[16] Peter W. Shor. Scheme for reducing decoherence in quantum computer memory. *Physical Review A*, 52(4):R2493–R2496, Oct 1995.

[17] Daniel Gottesman. *Stabilizer Codes and Quantum Error Correction*. PhD thesis, California Institute of Technology (arXiv:quant-ph/9705052), 1997.

[18] A. Robert Calderbank, Eric M. Rains, Peter W. Shor, and N. J. A. Sloane. Quantum error correction and orthogonal geometry. *Physical Review Letters*, 78(3):405–408, Jan 1997.

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

[20] N. David Mermin. *Quantum Computer Science*. Cambridge University Press, 2007.

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

[22] Andrew M. Steane. Error correcting codes in quantum theory. *Physical Review Letters*, 77(5):793–797, Jul 1996.

[23] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North Holland, 1983.

[24] Paolo Zanardi and Mario Rasetti. Noiseless quantum codes. *Physical Review Letters*, 79(17):3306–3309, Oct 1997.

[25] Paolo Zanardi and Mario Rasetti. Error avoiding quantum codes. *Modern Physics Letters B*, 11:1085–1093, 1997.

[26] Daniel A. Lidar, Isaac L. Chuang, and K. Birgitta Whaley. Decoherence-free subspaces for quantum computation. *Physical Review Letters*, 81(12):2594–2597, Sep 1998.

[27] David Kribs, Raymond Laflamme, and David Poulin. Unified and generalized approach to quantum error correction. *Physical Review Letters*, 94(18):180501, 2005.

[28] David W. Kribs, Raymond Laflamme, David Poulin, and Maia Lesosky. Operator quantum error correction. *Quantum Information & Computation*, 6:383–399, 2006.

[29] David Poulin. Stabilizer formalism for operator quantum error correction. *Physical Review Letters*, 95(23):230504, 2005.

[30] Todd A. Brun, Igor Devetak, and Min-Hsiu Hsieh. General entanglement-assisted quantum error-correcting codes. In *Proceedings of the IEEE International Symposium on Information Theory*, June 2007.

[31] Min-Hsiu Hsieh, Igor Devetak, and Todd Brun. General entanglement-assisted quantum error-correcting codes. *Physical Review A*, 76(6):062313, 2007.

[32] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[33] Garry Bowen. Entanglement required in achieving entanglement-assisted channel capacities. *Physical Review A*, 66(5):052313, Nov 2002.

[34] Todd A. Brun, Igor Devetak, and Min-Hsiu Hsieh. Correcting quantum errors with entanglement. *Science*, 314(5798):436–439, October 2006.

[35] Todd A. Brun, Igor Devetak, and Min-Hsiu Hsieh. Catalytic quantum error correction. *arXiv:quant-ph/0608027v2*, August 2006.

[36] Daniel Gottesman. Solution set #9. http://www.perimeterinstitute.ca/personal/dgottesman/QECC2007/Sols9.pdf.

[37] Seth Lloyd. Capacity of the noisy quantum channel. *Physical Review A*, 55(3):1613–1622, Mar 1997.

[38] Peter W. Shor. The quantum channel capacity and coherent information. In *Lecture Notes, MSRI Workshop on Quantum Computation*, 2002.

[39] Igor Devetak. The private classical capacity and quantum capacity of a quantum channel. *IEEE Transactions on Information Theory*, 51:44–55, January 2005.

[40] Charles H. Bennett, Peter W. Shor, John A. Smolin, and Ashish V. Thapliyal. Entanglement-assisted classical capacity of noisy quantum channels. *Physical Review Letters*, 83(15):3081–3084, Oct 1999.

[41] Charles H. Bennett, Peter W. Shor, John A. Smolin, and Ashish V. Thapliyal. Entanglement-assisted capacity of a quantum channel and the reverse shannon theorem. *IEEE Transactions on Information Theory*, 48:2637–2655, 2002.

[42] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 1991.

[43] Harold Ollivier and Jean-Pierre Tillich. Description of a quantum convolutional code. *Physical Review Letters*, 91(17):177902, Oct 2003.

[44] Harold Ollivier and Jean-Pierre Tillich. Quantum convolutional codes: fundamentals. *arXiv:quant-ph/0401134*, 2004.

[45] Markus Grassl and Martin Rötteler. Noncatastrophic encoders and encoder inverses for quantum convolutional codes. In *Proceedings of the IEEE International Symposium on Information Theory (quant-ph/0602129)*, 2006.

[46] Markus Grassl and Martin Rötteler. Quantum convolutional codes: Encoders and structural properties. In *Proceedings of the Forty-Fourth Annual Allerton Conference*, 2006.

[47] Markus Grassl and Martin Rötteler. Constructions of quantum convolutional codes. In *Proceedings of the IEEE International Symposium on Information Theory (arXiv:quant-ph/0703182)*, 2007.

[48] G. David Forney and Saikat Guha. Simple rate-1/3 convolutional and tail-biting quantum error-correcting codes. In *Proceedings of the IEEE International Symposium on Information Theory (arXiv:quant-ph/0501099)*, 2005.

[49] Salah A. Aly, Markus Grassl, Andreas Klappenecker, Martin Rötteler, and Pradeep Kiran Sarvepalli. Quantum convolutional bch codes. In *Proceedings of the 10th Canadian Workshop on Information Theory (arXiv:quant-ph/0703113)*, pages 180–183, 2007.

[50] Salah A. Aly, Andreas Klappenecker, and Pradeep Kiran Sarvepalli. Quantum convolutional codes derived from reed-solomon and reed-muller codes. *arXiv:quant-ph/0701037*, 2007.

[51] Mark M. Wilde, Hari Krovi, and Todd A. Brun. Convolutional entanglement distillation. *arXiv:0708.3699*, 2007.

[52] Mark M. Wilde and Todd A. Brun. Entanglement-assisted quantum convolutional coding. *arXiv:0712.2223*, 2007.

[53] Mark M. Wilde and Todd A. Brun. Unified quantum convolutional coding. In *Proceedings of the IEEE International Symposium on Information Theory (arXiv:0801.0821)*, July 2008.

[54] Patrick Hayden, Michał Horodecki, Andreas Winter, and Jon Yard. A decoupling approach to the quantum capacity. *Open Systems & Information Dynamics*, 15:7–19, March 2008.

[55] Rochus Klesse. A random coding based proof for the quantum coding theorem. *Open Systems & Information Dynamics*, 15:21–45, March 2008.

[56] Michał Horodecki, Seth Lloyd, and Andreas Winter. Quantum coding theorem from privacy and distinguishability. *Open Systems & Information Dynamics*, 15:47–69, March 2008.

[57] Patrick Hayden, Peter W. Shor, and Andreas Winter. Random quantum codes from gaussian ensembles and an uncertainty relation. *Open Systems & Information Dynamics*, 15:71–89, March 2008.

[58] David Poulin, Jean-Pierre Tillich, and Harold Ollivier. Quantum serial turbo-codes. *arXiv:0712.2888*, 2007.

[59] H. F. Chau. Quantum convolutional error-correcting codes. *Physical Review A*, 58(2):905–909, Aug 1998.

[60] H. F. Chau. Good quantum-convolutional error-correction codes and their decoding algorithm exist. *Physical Review A*, 60(3):1966–1974, Sep 1999.

[61] Igor Devetak, Aram W. Harrow, and Andreas Winter. A resource framework for quantum shannon theory. *arXiv:quant-ph/0512015*, 2005.

[62] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.

[63] Igor Devetak and Peter W. Shor. The capacity of a quantum channel for simultaneous transmission of classical and quantum information. *Communications in Mathematical Physics*, 256:287–303, 2005.

[64] Igor Devetak, Patrick Hayden, Debbie Leung, and Peter Shor. Triple trade-offs in quantum shannon theory. *In preparation*, 2008.

[65] 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):722–725, Jan 1996.

[66] Charles H. Bennett, David P. DiVincenzo, John A. Smolin, and William K. Wootters. Mixed-state entanglement and quantum error correction. *Physical Review A*, 54(5):3824–3851, Nov 1996.

[67] Mark M. Wilde and Todd A. Brun. Optimal entanglement formulas for entanglement-assisted quantum coding. *Physical Review A*, 77:064302, 2008.

[68] Mark M. Wilde and Todd A. Brun. Protecting quantum information with entanglement and noisy optical modes. *In preparation*, 2008.

[69] Mark M. Wilde and Todd A. Brun. Quantum convolutional coding with free entanglement. *In preparation*, 2008.

[70] Mark M. Wilde, Hari Krovi, and Todd A. Brun. Entanglement-assisted quantum error correction with linear optics. *Physical Review A*, 76:052308, 2007.

[71] Daniel Gottesman. Class of quantum error-correcting codes saturating the quantum hamming bound. *Physical Review A*, 54(3):1862–1868, Sep 1996.

[72] Marc Hein, W. Dür, Jens Eisert, Robert Raussendorf, Maarten Van den Nest, and Hans J. Briegel. Entanglement in graph states and its applications. *Proceedings of the International School of Physics "Enrico Fermi" on "Quantum Computers, Algorithms and Chaos" (arXiv:quant-ph/0602096)*, July 2005.

[73] Robert Raussendorf and Hans J. Briegel. A one-way quantum computer. *Physical Review Letters*, 86(22):5188–5191, 2001.

[74] David Fattal, Toby S. Cubitt, Yoshihisa Yamamoto, Sergey Bravyi, and Isaac L. Chuang. Entanglement in the stabilizer formalism. *arXiv:quant-ph/0406168*, 2004.

[75] Raymond Laflamme, Cesar Miquel, Juan Pablo Paz, and Wojciech Hubert Zurek. Perfect quantum error correcting code. *Physical Review Letters*, 77(1):198–201, Jul 1996.

[76] Zhicheng Luo and Igor Devetak. Efficiently implementable codes for quantum key expansion. *Physical Review A*, 75(1):010303, 2007.

[77] Ana Cannas da Silva. *Lectures on Symplectic Geometry*. Springer, 2001.

[78] Bilal Shaw, Mark M. Wilde, Ognyan Oreshkov, Isaac Kremsky, and Daniel Lidar. Encoding one logical qubit into six physical qubits. *arXiv:0803.1495*, 2008.

[79] Min-Hsiu Hsieh, Igor Devetak, and Todd A. Brun. Quantum quasi-cyclic low-density parity-check codes. In *Proceedings of the Asian Conference on Quantum Information Science*, pages 101–102, 2007.

[80] Rolf Johannesson and Kamil Sh. Zigangirov. *Fundamentals of Convolutional Coding*. Wiley-IEEE Press, 1999.

[81] Harold Ollivier and Jean-Pierre Tillich. Trellises for stabilizer codes: Definition and uses. *Physical Review A*, 74:032304, 2006.

[82] Aram Harrow. Coherent communication of classical messages. *Physical Review Letters*, 92:097902, March 2004.

[83] Mark M. Wilde, Hari Krovi, and Todd A. Brun. Coherent communication with continuous quantum variables. *Physical Review A*, 75(6):060303(R), 2007.

[84] Isaac Kremsky, Min-Hsiu Hsieh, and Todd A. Brun. Classical enhancement of quantum error-correcting codes. *Accepted for publication in Physical Review A (arXiv:0802.2414)*, 2008.

[85] Cédric Bény, Achim Kempf, and David W. Kribs. Generalization of quantum error correction via the heisenberg picture. *Physical Review Letters*, 98(10):100502, 2007.

[86] Peter W. Shor and John Preskill. Simple proof of security of the bb84 quantum key distribution protocol. *Physical Review Letters*, 85(2):441–444, Jul 2000.