Convolutional Entanglement Distillation

Mark M. Wilde, Hari Krovi, and Todd A. Brun

***Abstract*—We develop a theory of entanglement distillation that exploits a convolutional coding structure. We provide a method for converting an arbitrary classical binary or quaternary convolutional code into a convolutional entanglement distillation protocol. The yield and error-correcting properties of such a protocol depend respectively on the rate and error-correcting properties of the imported classical convolutional code. In a convolutional entanglement distillation protocol, two parties sharing noisy ebits can distill noiseless ebits "online" as they acquire more noisy ebits and this online protocol reduces decoding complexity.**

***Index Terms*—quantum convolutional codes, convolutional entanglement distillation, entanglement-assisted quantum codes**

I. Introduction

The theory of quantum error correction [1], [2] plays a prominent role in the practical realization and engineering of quantum computing and communication devices. 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 entanglement-assisted stabilizer formalism is a useful extension of the standard stabilizer theory of quantum error correction [3], [4]. This formalism includes entanglement as a resource that a sender and receiver can exploit. The benefit of including shared entanglement is that one can produce an entanglement-assisted quantum code from an arbitrary classical binary or quaternary code. Another benefit is that the performance of the original classical code determines the performance of the resulting quantum code.

The goal of entanglement distillation resembles the goal of quantum error correction [5], [6]. An entanglement distillation protocol (EDP) extracts noiseless, maximally-entangled ebits from a larger set of noisy ebits. 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 EDP [6]. Luo and Devetak incorporated shared entanglement to demonstrate how to convert an entanglement-assisted stabilizer code into an entanglement-assisted EDP [7].

Several authors have contributed to the theory of quantum convolutional codes [8], [9], [10]. Quantum convolutional codes are useful in a communication context where a sender has a stream of qubits to send to a receiver.

Todd A. Brun is with the Communication Sciences Institute of the Ming Hsieh Department of Electrical Engineering at the University of Southern California, Los Angeles, California 90089 USA. Mark M. Wilde and Hari Krovi were at the above location when conducting this research. Mark M. Wilde is now with the School of Computer Science, McGill University, Montreal, Quebec, Canada. Hari Krovi is now with the Department of Computer Science & Engineering, University of Connecticut, Storrs, CT, U.S.A. (E-mail: mark.wilde@alumni.usc.edu; krovi@engr.uconn.edu; tbrun@usc.edu)

In this paper, our main contribution is a theory of convolutional entanglement distillation. Our theory allows us to import arbitrary classical convolutional codes for use in entanglement distillation. The task of finding a convolutional entanglement distillation protocol (CEDP) now becomes the established task of finding a classical convolutional code.

We begin in Section II by showing how to construct a CEDP from an arbitrary quantum convolutional code. We translate earlier protocols for entanglement distillation of a block of noisy ebits to the convolutional setting. Our CEDP possesses several benefits—it distills entanglement "online." It outperforms a block EDP when considering the tradeoff between distillation yield and decoding complexity.

Another contribution of this article is a method for constructing a CEDP when the sender and receiver initially share some noiseless ebits. All prior quantum convolutional work requires the code to satisfy a restrictive self-orthogonality constraint [8], [9], [10]. We lift this constraint by allowing shared noiseless entanglement. The benefit of a CEDP with entanglement assistance is that we can produce a CEDP from an *arbitrary* classical binary or quaternary convolutional code. The error-correcting properties for the CEDP follow directly from the properties of the imported classical code. We mention that our entanglement-assisted CEDP is a particular way of implementing the breeding protocol [5], [6].

We organize our work as follows. In Section II, we show how to convert an arbitrary quantum convolutional code into a CEDP. Section III discusses the shifted symplectic product that is important for our entanglement-assisted CEDPs. In Section IV, we provide several methods and examples for constructing CEDPs where two parties possess a few initial noiseless ebits. These initial noiseless ebits act as a catalyst for the CEDP. The constructions in Section IV make it possible to import an arbitrary classical binary or quaternary convolutional code for use as a CEDP.

II. CEDP without Entanglement Assistance

We first show how to convert an arbitrary quantum convolutional code into a CEDP. 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 EDP in the online manner, but we formulate the forthcoming mathematics with the countably infinite description.

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 half of each Bell state

through a noisy quantum channel. Alice and Bob then possess the ensemble $\{p_i, |\Phi^+\rangle_i^{AB}\}$ where p_i is the probability that the state is $|\Phi^+\rangle_i^{AB}$, where

$$\left|\Phi^+\right\rangle_i^{AB} \equiv (\mathbf{I} \otimes \mathbf{A}_i) \left|\Phi_\infty^+\right\rangle^{AB}, \tag{1}$$

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 [10] of channel errors acting on Bob's side. The identity matrices acting on Alice's side indicate that the noisy channel does not affect her qubits.

Alice and Bob employ the following strategy to distill noiseless ebits. Alice measures the $n-k$ generators in the basic generator set $\mathcal{G}_0$ of a quantum convolutional code [10]. The measurement projects the first $n(\nu+1)$ ebits (ν is the constraint length [10]) 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 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 typically has to wait to receive later error vectors before performing corrective operations. Alice and Bob repeat the above procedure for all shifts $D(\mathcal{G}_0)$, $D^2(\mathcal{G}_0)$, ... of the basic generators in $\mathcal{G}_0$ where D is the delay operator for a quantum convolutional code [10]. 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 [11] or a table-lookup on the error set $\mathcal{E}$ to determine the most likely errors. This error determination process is a purely classical computation. As soon as Bob finishes correcting some errors, they can both use the bit vectors $\mathbf{a}_i$ for all $i \in \mathbb{Z}^+$ to restore their ebits to a set of logical ebits.[1] They can then extract physical ebits from these logical ebits by each performing the online decoding circuit corresponding to the original quantum convolutional code. The algorithm outlined in [12] gives a method for determining the online decoding circuit.

III. The Shifted Symplectic Product

We now consider the commutative properties of Pauli sequences. We develop some mathematics that leads to 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. All of our constructions following this preliminary section exploit the properties of the shifted symplectic product.

We first define the phase-free Pauli group $\left[\Pi^{\mathbb{Z}}\right]$ on a sequence of qubits. The delay transform D shifts a Pauli sequence to the right by n qubits [10]. 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 n-qubit shifts of arbitrary Pauli operators:

$$\Pi^{\mathbb{Z}} = \left\{ \prod_{i\in\mathbb{Z}} D^i(A_i) : A_i \in \Pi^n \right\}, \tag{2}$$

where Π^n is the Pauli group over n qubits. We remark that $D^i(A_i) = D^i(A_i \otimes I^{\otimes\infty})$. 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:

$$\left[\Pi^{\mathbb{Z}}\right] = \left\{\beta\mathbf{A} \mid \mathbf{A} \in \Pi^{\mathbb{Z}}, \beta \in \mathbb{C}, |\beta| = 1\right\}. \tag{3}$$

We develop a relation between binary polynomials and Pauli sequences that is useful for representing the shifting nature of quantum convolutional codes. A quantum convolutional code in general consists of generators with n qubits per frame. One can obtain the full set of generators of the quantum convolutional code by shifting the basic generator set by integer multiples of the frame size n [10]. Let the delay transform D shift a Pauli sequence to the right by an arbitrary 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:

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

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

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

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

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

where $u_i(D) = (z_i(D) \,|\, x_i(D))$. Suppose $\mathbf{v}(D) = (\mathbf{z}'(D) \,|\, \mathbf{x}'(D))$, where $\mathbf{v}(D) \in (\mathbb{Z}_2(D))^{2n}$. The map $\mathbf{N}$ induces an isomorphism $[\mathbf{N}] : (\mathbb{Z}_2(D))^{2n} \to \left[\Pi^{\mathbb{Z}}\right]$ because addition of binary polynomials is equivalent to multiplication of Pauli elements up to a global phase: $[\mathbf{N}(\mathbf{u}(D) + \mathbf{v}(D))] = [\mathbf{N}(\mathbf{u}(D))]\,[\mathbf{N}(\mathbf{v}(D))]$.

A commuting set comprising a basic set of finite-weight generators and all of their shifts specifies a quantum convolutional code [10]. How can we capture the commutation relations of a Pauli sequence and all of its shifts? The shifted symplectic product $\odot$, where

$$\odot : (\mathbb{Z}_2(D))^{2n} \times (\mathbb{Z}_2(D))^{2n} \to \mathbb{Z}_2(D), \tag{4}$$

does so in an elegant way. It maps vectors of binary polynomials to a finite-degree and finite-delay binary polynomial:

$$(\mathbf{u} \odot \mathbf{v})(D) = \mathbf{z}\left(D^{-1}\right) \cdot \mathbf{x}'(D) - \mathbf{x}\left(D^{-1}\right) \cdot \mathbf{z}'(D), \tag{5}$$

where $\cdot$ represents the standard inner product. The symplectic orthogonality condition originally given in Ref. [9] inspires

[1] A simple example of a logical ebit is as follows:

$$\left(|0_L\rangle^A |0_L\rangle^B + |1_L\rangle^A |1_L\rangle^B\right)/\sqrt{2}$$

where $|0_L\rangle$ and $|1_L\rangle$ are the respective logical '0' and logical '1' codewords for some quantum code that encodes one information qubit.

the definition for the shifted symplectic product. The shifted symplectic product is not a proper symplectic product because it fails to be "alternating." The alternating property requires that $(\mathbf{u} \odot \mathbf{v})(D) = -(\mathbf{v} \odot \mathbf{u})(D)$, but we find instead that the following holds:

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

Every vector $\mathbf{u}(D) \in \mathbb{Z}_2(D)^{2n}$ is self-time-reversal symmetric with respect to $\odot$ because addition and subtraction are the same over $\mathbb{Z}_2$:

$$(\mathbf{u} \odot \mathbf{u})(D) = (\mathbf{u} \odot \mathbf{u})\left(D^{-1}\right) \qquad \forall \mathbf{u}(D) \in \mathbb{Z}_2(D)^{2n}. \quad (6)$$

We employ the addition convention from now on and drop the minus signs. The shifted symplectic product for vectors of binary polynomials is a binary polynomial in D. We write its coefficients as follows:

$$(\mathbf{u} \odot \mathbf{v})(D) = \sum_{i \in \mathbb{Z}} (\mathbf{u} \odot \mathbf{v})_i \ D^i. \quad (7)$$

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:

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

IV. CEDP with Entanglement Assistance

The CEDP in this section operates identically to the one in Section II. The measurements, classical communication, and recovery and decoding operations proceed exactly as before.

The difference is that we now assume the sender and receiver share a few initial noiseless ebits that catalyze the distillation protocol. The sender and receiver require noiseless ebits for each round of the CEDP. They can use the noiseless ebits generated by earlier rounds for consumption in later rounds. We discuss practical concerns in the conclusion.

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 CEDP. We explicitly give an example that highlights 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 CEDP reduces to that of finding a good classical convolutional code.

A. A Yield (n-m)/n CEDP with Entanglement Assistance

Suppose we have m generators $\{\mathbf{N}(\mathbf{u}_i(D)) : 1 \leq i \leq m\}$, where

$$\left[\begin{array}{c} \mathbf{u}_1(D) \\ \mathbf{u}_2(D) \\ \vdots \\ \mathbf{u}_m(D) \end{array}\right] = \left[\begin{array}{c|c} \mathbf{z}_1(D) & \mathbf{x}_1(D) \\ \mathbf{z}_2(D) & \mathbf{x}_2(D) \\ \vdots & \vdots \\ \mathbf{z}_m(D) & \mathbf{x}_m(D) \end{array}\right]. \quad (8)$$

We make no assumption about the commutation relations of the above generators. We choose them solely for their error-correcting properties.

We utilize the shifted symplectic product to design a CEDP. Let us adopt the following shorthand for the auto and cross shifted symplectic product of generators $\mathbf{u}_1(D), \ldots, \mathbf{u}_m(D)$:

$$\mathbf{u}_i^+ \equiv (\mathbf{u}_i \odot \mathbf{u}_i)(D)^+, \qquad \mathbf{u}_{i,j} \equiv (\mathbf{u}_i \odot \mathbf{u}_j)(D), \quad (9)$$

where the notation $(\mathbf{u}_i \odot \mathbf{u}_i)(D)^+$ indicates that we extract the polynomial from $(\mathbf{u}_i \odot \mathbf{u}_i)(D)$ that includes only positive powers of D. Consider the following matrix:

$$\left[\begin{array}{c} \mathbf{a}_1(D) \\ \mathbf{a}_2(D) \\ \vdots \\ \mathbf{a}_m(D) \end{array}\right] = \left[\begin{array}{cccc|c} \mathbf{u}_1^+ & \mathbf{u}_{2,1} & \cdots & \mathbf{u}_{m,1} & \\ 0 & \mathbf{u}_2^+ & \cdots & \mathbf{u}_{m,2} & \\ \vdots & & \ddots & \vdots & \mathbf{I}_{m \times m} \\ 0 & \cdots & 0 & \mathbf{u}_m^+ & \end{array}\right]. \quad (10)$$

The symplectic relations of the entries $\mathbf{a}_i(D)$ are the same as the original $\mathbf{u}_i(D)$:

$$(\mathbf{a}_i \odot \mathbf{a}_j)(D) = (\mathbf{u}_i \odot \mathbf{u}_j)(D) \qquad \forall i, j \in \{1, \ldots, m\}.$$

We mention that the following matrix also has the same symplectic relations:

$$\left[\begin{array}{cccc|c} \mathbf{u}_1^+ & 0 & \cdots & 0 & \\ \mathbf{u}_{1,2} & \mathbf{u}_2^+ & \cdots & \vdots & \\ \vdots & & \ddots & 0 & \mathbf{I}_{m \times m} \\ \mathbf{u}_{1,m} & \mathbf{u}_{2,m} & \cdots & \mathbf{u}_m^+ & \end{array}\right]. \quad (11)$$

Let us rewrite (10) as follows:

$$\left[\begin{array}{c} \mathbf{a}_1(D) \\ \mathbf{a}_2(D) \\ \vdots \\ \mathbf{a}_m(D) \end{array}\right] = \left[\begin{array}{c|c} \mathbf{z}'_1(D) & \mathbf{x}'_1(D) \\ \mathbf{z}'_2(D) & \mathbf{x}'_2(D) \\ \vdots & \vdots \\ \mathbf{z}'_m(D) & \mathbf{x}'_m(D) \end{array}\right]. \quad (12)$$

The above matrix provides a straightforward way to make the original generators commute with all of their shifts. We augment the generators in (8) by the generators $\mathbf{a}_i(D)$ to get the following $m \times 2(n+m)$ matrix:

$$\mathbf{U}'(D) = \left[\begin{array}{c|c} \mathbf{Z}(D) & \mathbf{X}(D) \end{array}\right] = \left[\begin{array}{cc|cc} \mathbf{z}_1(D) & \mathbf{z}'_1(D) & \mathbf{x}_1(D) & \mathbf{x}'_1(D) \\ \mathbf{z}_2(D) & \mathbf{z}'_2(D) & \mathbf{x}_2(D) & \mathbf{x}'_2(D) \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{z}_m(D) & \mathbf{z}'_m(D) & \mathbf{x}_m(D) & \mathbf{x}'_m(D) \end{array}\right]. \quad (13)$$

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 [9]:

$$\mathbf{Z}\left(D^{-1}\right)\mathbf{X}(D)^T - \mathbf{X}\left(D^{-1}\right)\mathbf{Z}(D)^T = 0. \quad (14)$$

The construction gives a commuting set of generators for arbitrary shifts and thus forms a valid stabilizer.

The extra "primed" entries in the augmented matrix act on the noiseless ebits that Alice and Bob share while the "unprimed" entries act on the noisy ebits. Alice and Bob exploit the error-correcting properties of the unprimed entries, and the "primed" entries serve to resolve the anticommutativity inherent in the "unprimed" entries.

We can readily develop a CEDP using the above formulation. The generators in the augmented matrix $\mathbf{U}'(D)$ correct for errors on the first n ebits of each frame. The last m ebits of each frame are noiseless ebits that help to obtain a commuting stabilizer. It is necessary to catalyze the distillation protocol with a number of noiseless ebits exponential in $(n+m)\nu$ because of the complexity of Viterbi decoding [11]. Later frames can use the noiseless ebits generated from previous frames. These initial noiseless ebits are negligible when calculating the yield of the CEDP.

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 CEDP.

It is straightforward to develop a noncatastrophic decoding circuit using previous techniques [12]. Every augmented generator in $\mathbf{U}'(D)$ has 1 as an entry so that it satisfies the property required for noncatastrophicity.

Example 1: We begin with a classical quaternary convolutional code with entries from $\mathbb{F}_4$:

$$(\cdots|0000|1\bar{\omega}10|1101|0000|\cdots). \tag{15}$$

The above code is a convolutional version of the classical quaternary block code from Ref. [3]. We multiply the above generator by $\bar{\omega}$ and ω as prescribed in Refs. [2], [10] and use the following map,

$$\begin{array}{c|cccc} \Pi & I & X & Y & Z \\ \hline \mathbb{F}_4 & 0 & \omega & 1 & \bar{\omega} \end{array}, \tag{16}$$

to obtain the following Pauli generators

$$\begin{aligned} \mathbf{N}(\mathbf{u}_1(D)) &= (\cdots|IIII|ZXZI|ZZIZ|IIII|\cdots), \\ \mathbf{N}(\mathbf{u}_2(D)) &= (\cdots|IIII|XYXI|XXIX|IIII|\cdots). \end{aligned} \tag{17}$$

We determine binary polynomials corresponding to the above Pauli generators:

$$\left(\begin{array}{cccc|cccc} 1+D & D & 1 & D & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1+D & 1+D & 1 & D \end{array}\right), \tag{18}$$

where the first row is $\mathbf{u}_1(D)$ and the second is $\mathbf{u}_2(D)$. The shifted symplectic products give the commutation relations:

$$\begin{aligned} &(\mathbf{u}_1 \odot \mathbf{u}_1)(D) = D^{-1}+D, \quad (\mathbf{u}_2 \odot \mathbf{u}_2)(D) = D^{-1}+D, \\ &(\mathbf{u}_1 \odot \mathbf{u}_2)(D) = D. \end{aligned} \tag{19}$$

Consider the following two generators:

$$\left(\begin{array}{c} \mathbf{a}_1(D) \\ \mathbf{a}_2(D) \end{array}\right) = \left(\begin{array}{cc|cc} D & 0 & 1 & 0 \\ D & D & 0 & 1 \end{array}\right). \tag{20}$$

Their relations under the shifted symplectic product are the same as those in (19). We use the construction from (11) so that we have positive delay operators in the augmented matrix. 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 "Z matrix" is

$$\mathbf{Z}(D) = \begin{pmatrix} 1+D & D & 1 & D & D & 0 \\ 0 & 1 & 0 & 0 & D & D \end{pmatrix},$$

and the augmented "X matrix" is

$$\mathbf{X}(D) = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1+D & 1+D & 1 & D & 0 & 1 \end{pmatrix}.$$

The augmented matrix $\mathbf{U}'(D)$ is

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

The first row of $\mathbf{U}'(D)$ is generator $\mathbf{u}'_1(D)$ and the second row is $\mathbf{u}'_2(D)$. The original block code from Ref. [3] corrects for an arbitrary single-qubit error. The above CEDP 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.

Forney, Guha, Grassl (FGG) showed how to produce a rate-$\frac{n-2}{n}$ quantum convolutional code by importing a classical convolutional code over $\mathbb{F}_4$ that has one generator and is self-orthogonal with respect to the trace product [10]. It is then straightforward to convert an FGG code to a yield-$\frac{n-2}{n}$ CEDP with the method in Section II. With the construction in this section, it is possible to produce a yield-$\frac{n-2}{n}$ CEDP from a classical convolutional code over $\mathbb{F}_4$ that has one generator and no self-orthogonality constraint.

B. CSS-Like Construction for a CEDP

We finally present a construction that allows us to import two arbitrary binary classical codes for use in a CEDP. The construction is similar to a CSS code because one code corrects for bit flips and the other corrects for phase flips.

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 [4] with some key differences.

Suppose we have m generators $\{\mathbf{N}(\mathbf{w}_i(D)) : 1 \leq i \leq m\}$ where

$$\left[\begin{array}{c} \mathbf{w}_1(D) \\ \vdots \\ \mathbf{w}_p(D) \\ \mathbf{w}_{p+1}(D) \\ \vdots \\ \mathbf{w}_m(D) \end{array}\right] = \left[\begin{array}{c|c} \mathbf{z}_1(D) & \mathbf{0} \\ \vdots & \vdots \\ \mathbf{z}_p(D) & \mathbf{0} \\ \mathbf{0} & \mathbf{x}_1(D) \\ \vdots & \vdots \\ \mathbf{0} & \mathbf{x}_{m-p}(D) \end{array}\right].$$

and each vector $\mathbf{w}_i(D)$ has length $2n$. The vectors $\mathbf{z}_1(D)$, $\ldots$, $\mathbf{z}_p(D)$ could come from one classical binary code, and the vectors $\mathbf{x}_1(D)$, $\ldots$, $\mathbf{x}_{m-p}(D)$ could come from another classical binary code. The following orthogonality relations hold for the above vectors:

$$\begin{aligned} \forall \;\; 1 \leq i,j \leq p &: (\mathbf{w}_i \odot \mathbf{w}_j)(D) = 0, \\ \forall \;\; p+1 \leq i',j' \leq m &: (\mathbf{w}_{i'} \odot \mathbf{w}_{j'})(D) = 0. \end{aligned}$$

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 the index i to the set $\mathcal{V}$, 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

$$\begin{aligned}\mathbf{w}_r(D) \leftarrow (\mathbf{w}_{2c+l+2} \odot \mathbf{w}_{2c+l+1})(D)\,\mathbf{w}_r(D) \\ + (\mathbf{w}_r \odot \mathbf{w}_{2c+l+2})\left(D^{-1}\right)\mathbf{w}_{2c+l+1}(D).\end{aligned}$$

if $\mathbf{w}_r(D)$ has a purely z component. The symbol $\leftarrow$ indicates a reassignment. Perform

$$\begin{aligned}\mathbf{w}_r(D) \leftarrow (\mathbf{w}_{2c+l+1} \odot \mathbf{w}_{2c+l+2})(D)\,\mathbf{w}_r(D) \\ + (\mathbf{w}_r \odot \mathbf{w}_{2c+l+1})\left(D^{-1}\right)\mathbf{w}_{2c+l+2}(D).\end{aligned}$$

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

$$(\mathbf{w}_r \odot \mathbf{w}_{2c+l+1})(D) = (\mathbf{w}_r \odot \mathbf{w}_{2c+l+2})(D) = 0.$$

Increment c by one, increment i by one, add the index i to the set $\mathcal{U}$, 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:

$$\begin{aligned}(\mathbf{u}_i \odot \mathbf{v}_j)(D) &= f_i(D)\,\delta_{ij} \quad \forall \;\; i,j \in \{1,\ldots,c\}, \\ (\mathbf{u}_i \odot \mathbf{u}_j)(D) &= 0 \qquad\qquad \forall \;\; i,j \in \{1,\ldots,l\}, \\ (\mathbf{v}_i \odot \mathbf{v}_j)(D) &= 0 \qquad\qquad \forall \;\; i,j \in \{1,\ldots,c\},\end{aligned}$$

where $f_i(D)$ is an arbitrary polynomial. Let us arrange the above generators in a matrix as follows:

$$\begin{bmatrix}\mathbf{u}_1^T & \cdots & \mathbf{u}_c^T & \mathbf{v}_1^T & \cdots & \mathbf{v}_c^T & \mathbf{u}_{c+1}^T & \cdots & \mathbf{u}_{c+l}^T\end{bmatrix}^T,$$

where we suppress the D argument in the above vectors. We augment the above generators with the following matrix so that all vectors are orthogonal to each other:

$$\left[\begin{array}{cccc|c} f_1\left(D^{-1}\right) & 0 & \cdots & 0 & \mathbf{0}_{1\times c} \\ 0 & f_2\left(D^{-1}\right) & & \vdots & \mathbf{0}_{1\times c} \\ \vdots & & \ddots & 0 & \vdots \\ 0 & \cdots & 0 & f_c\left(D^{-1}\right) & \mathbf{0}_{1\times c} \\ \mathbf{0}_{c\times 1} & \mathbf{0}_{c\times 1} & \cdots & \mathbf{0}_{c\times 1} & \mathbf{I}_{c\times c} \\ \mathbf{0}_{l\times 1} & \mathbf{0}_{l\times 1} & \cdots & \mathbf{0}_{l\times 1} & \mathbf{0}_{l\times c} \end{array}\right].$$

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 CEDP has yield $(k_1 + k_2 - n)/n$.

V. Conclusion and Current Work

We have 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 CEDP. The error-correcting properties and rate of the classical code translate to the quantum case, implying that a good classical convolutional code leads to a good entanglement distillation protocol.

One major practical concern with an entanglement-assisted CEDP is the effect of residual errors. In practice, a code cannot correct all the errors from a given noisy channel. The performance of the CEDP will be adversely affected if the ebits that catalyze are not really noiseless. Our main purpose in this article is to describe the structure of CEDPs and we leave this question open for future study. But two simple solutions to this problem are either to use a better code or to perform further distillation steps before reusing distilled ebits as a catalyst if residual errors degrade the ebits from being truly noiseless.

The authors thank Igor Devetak and Zhicheng Luo for discussions, Saikat Guha for locating Jonsson's master's thesis, and the anonymous referees for constructive feedback. MMW acknowledges support from NSF Grant 0545845, the National Research Foundation & Ministry of Education, Singapore, and the Hearne Insitute for Theoretical Physics. HK and TAB acknowledge support from NSF Grant CCF-0448658.

References

[1] D. Gottesman, "Stabilizer codes and quantum error correction," Ph.D. dissertation, California Institute of Technology, 1997.

[2] A. Calderbank, E. Rains, P. Shor, and N. Sloane, "Quantum error correction via codes over GF(4)," *IEEE Trans. Inf. Theory*, vol. 44, pp. 1369–1387, 1998.

[3] T. A. Brun, I. Devetak, and M.-H. Hsieh, "Correcting quantum errors with entanglement," *Science*, vol. 314, no. 5798, pp. pp. 436 – 439, October 2006.

[4] ——, "Catalytic quantum error correction," *arXiv:quant-ph/0608027*, August 2006.

[5] C. H. Bennett, G. Brassard, S. Popescu, B. Schumacher, J. A. Smolin, and W. K. Wootters, "Purification of noisy entanglement and faithful teleportation via noisy channels," *Phys. Rev. Lett.*, vol. 76, no. 5, pp. 722–725, Jan 1996.

[6] C. H. Bennett, D. P. DiVincenzo, J. A. Smolin, and W. K. Wootters, "Mixed-state entanglement and quantum error correction," *Phys. Rev. A*, vol. 54, no. 5, pp. 3824–3851, Nov 1996.

[7] Z. Luo and I. Devetak, "Efficiently implementable codes for quantum key expansion," *Phys. Rev. A*, vol. 75, no. 1, p. 010303, 2007.

[8] H. Ollivier and J.-P. Tillich, "Description of a quantum convolutional code," *Phys. Rev. Lett.*, vol. 91, no. 17, p. 177902, Oct 2003.

[9] ——, "Quantum convolutional codes: fundamentals," *arXiv:quant-ph/0401134*, 2004.

[10] G. D. Forney, M. Grassl, and S. Guha, "Convolutional and tail-biting quantum error-correcting codes," *IEEE Trans. Inf. Theory*, vol. 53, pp. 865–880, 2007.

[11] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. 13, pp. 260–269, 1967.

[12] M. Grassl and M. Rötteler, "Noncatastrophic encoders and encoder inverses for quantum convolutional codes," in *IEEE Int. Symp. on Inf. Theory (quant-ph/0602129)*, 2006, pp. 1109–1113.