Quantum Inf Process
DOI 10.1007/s11128-010-0179-9

# Quantum convolutional coding with shared entanglement: general structure

**Mark M. Wilde · Todd A. Brun**

Received: 10 February 2010 / Accepted: 22 April 2010


**Abstract** We present a general theory of entanglement-assisted quantum convolutional coding. The codes have a convolutional or memory structure, they assume that the sender and receiver share noiseless entanglement prior to quantum communication, and they are not restricted to possess the Calderbank–Shor–Steane structure as in previous work. We provide two significant advances for quantum convolutional coding theory. We first show how to "expand" a given set of quantum convolutional generators. This expansion step acts as a preprocessor for a polynomial symplectic Gram–Schmidt orthogonalization procedure that simplifies the commutation relations of the expanded generators to be the same as those of entangled Bell states (ebits) and ancilla qubits. The above two steps produce a set of generators with equivalent error-correcting properties to those of the original generators. We then demonstrate how to perform online encoding and decoding for a stream of information qubits, halves of ebits, and ancilla qubits. The upshot of our theory is that the quantum code designer can engineer quantum convolutional codes with desirable error-correcting properties without having to worry about the commutation relations of these generators.

**Keywords** Quantum convolutional codes · Entanglement-assisted quantum convolutional codes · Quantum information theory · Entanglement-assisted quantum codes

M. M. Wilde (✉)
School of Computer Science, McGill University, Montreal, QC, Canada
e-mail: mark.wilde@mcgill.ca; mwilde@gmail.com

T. A. Brun
Center for Quantum Information Science and Technology and the Communication Sciences Institute of the Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089, USA
e-mail: tbrun@usc.edu

Published online: 08 June 2010

## 1 Introduction

The theory of quantum convolutional coding has recently emerged as a powerful means for protecting a stream of quantum information from the negative effects of a noisy quantum communication channel [1–12]. Quantum convolutional coding theory combined with other coding techniques may be 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 [13–19]. Poulin et al. have recently incorporated some of this well-developed theory of quantum convolutional coding into a theory of quantum serial-turbo coding [20] with the goal of designing quantum codes that come close to achieving capacity.

Quantum convolutional codes have numerous benefits. The periodic structure of their encoding and decoding circuits ensures a low complexity for encoding and decoding while also providing higher performance than a block code with equivalent encoding complexity [7]. 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.

A seminal paper on quantum error correction showed how to produce quantum error-correcting codes from classical linear binary or quaternary block codes [21]. Importing classical binary codes to construct quantum codes is now known as the Calderbank –Shor–Steane (CSS) construction [22]. Importing classical codes is useful because quantum code designers can utilize classical block codes with high performance to construct quantum codes with high performance. The only problem with this technique is that the imported classical block codes have to satisfy a restrictive dual-containing condition so that the operators in the resulting quantum code commute with one another.

Brun, Devetak, and Hsieh alleviated the dual-containing constraint by assuming that the sender and receiver share noiseless entanglement in the entanglement-assisted stabilizer formalism [23,24]. The entanglement-assisted stabilizer formalism allows quantum code designers to produce a quantum code from an arbitrary classical binary or quaternary block code by incorporating shared entanglement. In a different context, 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 [7]; 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). The current authors followed these two developments by detailing entanglement-assisted quantum convolutional codes [11] that have the CSS structure. We demonstrated in Ref. [11] how to produce an entanglement-assisted quantum convolutional code from two *arbitrary* classical binary convolutional codes—these imported codes do not need to satisfy a dual-containing constraint. The benefit of all these entanglement-assisted techniques is that we can use high-performance classical codes for correction of quantum errors.

In this paper, we show how to encode and decode an entanglement-assisted quantum convolutional code that does not necessarily have the CSS structure. The

methods in this work represent a significant extension of our previous work on CSS entanglement-assisted quantum convolutional codes [11]. 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 [24] 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 Refs. [3,11,25]. The encoding circuits incorporate both finite-depth and infinite-depth [11] operations, and the decoding circuits incorporate finite-depth operations only. One benefit of the techniques developed in this paper 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 with desirable error-correcting properties without having to search for codes that satisfy a commutativity constraint.

An important practical issue in the design of entanglement-assisted quantum convolutional codes is the characterization of performance in terms of some parameter like code distance or distance spectrum [20]. The focus of this paper is on the construction and design of encoding and decoding circuits for non-CSS entanglement-assisted quantum convolutional codes—not on the performance of these codes. We take this approach because entanglement-assisted codes produced from classical codes inherit distance and error-correcting properties from the parent classical codes. If the entanglement-assisted code does not come from a classical code, then one can analyze distance of the code in the standard way as the minimum weight element of the normalizer modulo the stabilizer [22] or with the distance spectrum parameter of Poulin et al. [20], but an analysis of this sort is beyond the scope of the present paper.

We structure our work as follows. Section 2 reviews the general definition of a quantum convolutional code and the matrix of binary polynomials that represents it. It also discusses the shifted symplectic product that captures the commutation relations of Pauli sequences. Section 3 reviews the theory of entanglement-assisted quantum block coding. In Sect. 4, 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 Sect. 5 how to compute the symplectic Gram–Schmidt orthogonalization procedure for the example in Sect. 4 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 gives the algorithms for computing the encoding and decoding circuits for an arbitrary entanglement-assisted quantum convolutional code. We present a detailed example in Sect. 7 that illustrates all of the procedures in this article. We finish with a discussion of the practical issues involved

in using these entanglement-assisted quantum convolutional codes and present some concluding remarks.

## 2 Review of quantum convolutional codes

We first review the theory of quantum convolutional coding. The reader can find more detailed reviews in Refs. [7,10,11]. A quantum convolutional code admits a representation in terms of Pauli sequences—infinite tensor products of Pauli matrices. We demonstrate how to map these Pauli sequences to a finite-dimensional binary polynomial matrix and call this mapping the Pauli-to-binary (P2B) isomorphism. It is more straightforward to work with binary polynomial matrices rather than Pauli sequences because we can employ linear-algebraic techniques.

We state some basic definitions and notation before proceeding with the definition of a quantum convolutional code. A Pauli sequence $\mathbf{A}$ is a countably infinite tensor product of Pauli matrices:

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

where each $A_i$ is a matrix in the Pauli group $\Pi = \{I, X, Y, Z\}$. Let $\Pi^{\mathbb{Z}^+}$ denote the set of all Pauli sequences. A finite-weight Pauli sequence is one in which the operators $X$, $Y$, and $Z$ act on a finite number of qubits and the identity operator acts on the countably infinite remaining qubits. An infinite-weight sequence is one that is not finite-weight.

**Definition 1** A rate-$k/n$ quantum convolutional code admits a representation by a basic set $\mathcal{G}_0$ of $n-k$ generators and all of their $n$-qubit shifts. For a quantum convolutional code, the generators commute with each other and all of the $n$-qubit shifts of themselves and the other generators. This commutativity requirement is necessary for the same reason that standard stabilizer codes require it [22]. The generators in the basic set $\mathcal{G}_0$ are as follows:

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

A frame of the code consists of $n$ qubits so that the frame size is $n$. 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.

We do not discuss the detailed operation of a quantum convolutional code in this article. Please consult Refs. [7,10,11] for a discussion of a quantum convolutional code's operation and see Figure 3 of Ref. [10] for a depiction of its operation. The current article addresses the issues of quantum convolutional code design.

We 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 [7]. 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:

$$\Pi^{\mathbb{Z}} = \left\{ \prod_{i \in \mathbb{Z}} D^i \left( A_i \right) : A_i \in \Pi^n \right\}, \tag{1}$$

where $\Pi^n$ is the Pauli group over $n$ qubits. 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:

$$\left[ \Pi^{\mathbb{Z}} \right] = \left\{ \beta \mathbf{A} \mid \mathbf{A} \in \Pi^{\mathbb{Z}}, \beta \in \mathbb{C}, |\beta| = 1 \right\}. \tag{2}$$

We develop a relation between binary polynomials and Pauli sequences that is useful for representing the shifting nature of quantum convolutional codes. We call this map the Pauli-to-binary (P2B) isomorphism. 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 $n$ [7]. Let the delay transform $D$ shift a Pauli sequence to the right by an arbitrary integer $n$. Consider a $2n$-dimensional vector $\mathbf{u}\left( D \right)$ of binary polynomials where $\mathbf{u}\left( D \right) \in \left( \mathbb{Z}_2 \left( D \right) \right)^{2n}$. Let us write $\mathbf{u}\left( D \right)$ as follows:

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

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

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

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

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

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

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

where $u_i \left( D \right) = \left( z_i \left( D \right) | x_i \left( D \right) \right)$. Suppose $\mathbf{v}\left( D \right) = \left( \mathbf{z}' \left( D \right) | \mathbf{x}' \left( D \right) \right)$, where $\mathbf{v}\left( D \right) \in \left( \mathbb{Z}_2 \left( D \right) \right)^{2n}$. The map $\mathbf{N}$ induces an isomorphism

$$[\mathbf{N}] : \left( \mathbb{Z}_2 \left( D \right) \right)^{2n} \rightarrow \left[ \Pi^{\mathbb{Z}} \right]$$

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

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

We illustrate the P2B isomorphism by means of an example. Consider the following generator for a quantum convolutional code:

$$\cdots \left| III \right| XXX \left| XZY \right| III \left| \cdots \right.$$

We obtained the above generator by modifying generators for a code from Ref. [7]. The vertical bars indicate that the frame size is three qubits and every three-qubit shift of the above generator yields another stabilizer generator for the code. The above generator forms a valid quantum convolutional code because it commutes with all of its three-qubit shifts. For now, we are not concerned with the error-correcting properties of the above generator—we are merely interested in illustrating the principle of the P2B isomorphism.

The above generator has the following representation as a binary polynomial matrix:

$$\left[\, 0 \;\; D \;\; D \,\middle|\, 1 + D \;\; 1 \;\; 1 + D \,\right].$$

The matrix on the right is the "X" matrix and the matrix on the left is the "Z" matrix. The vertical bar in the above matrix serves to distinguish between the "X" and "Z" matrices. Each "X" operator in the first frame of the code gives a "1" entry in the "X" matrix. There are no "1" entries in the "Z" matrix because there are no "Z" operators in the first frame of the code. Each "X" and "Z" operator in the second frame of the code gives a "$D$" entry in the respective "X" and "Z" matrices. The "Y" entry in the second frame contributes a "$D$ " entry to both the "X" and "Z" matrix.

In general, we represent a rate-$k/n$ quantum convolutional code with an $(n-k) \times 2n$-dimensional quantum check matrix $H(D)$ whose entries are binary polynomials where

$$H(D) = \left[\, Z(D) \,\middle|\, X(D) \,\right].$$

The shifted symplectic product $\odot$ captures the commutations relations of any two generators [10,11]. Suppose that $h_1(D) = \left[\, z_1(D) \,\middle|\, x_1(D) \,\right]$ and $h_2(D) = \left[\, z_2(D) \,\middle|\, x_2(D) \,\right]$ are the first and second respective rows of check matrix $H(D)$. The shifted symplectic product $(h_1 \odot h_2)(D)$ is a polynomial equal to the following expression:

$$(h_1 \odot h_2)(D) = z_1(D) \cdot x_2\left(D^{-1}\right) + x_1(D) \cdot z_2\left(D^{-1}\right),$$

where "$\cdot$" represents the standard inner product between two vectors and addition is binary. Please note that we have slightly changed the convention for the shifted symplectic product from that used in Ref's. [10,11] in order to agree with the convention

in Ref's. [3,5]. The shifted symplectic products $(h_1 \odot h_1)(D)$, $(h_2 \odot h_2)(D)$, and $(h_1 \odot h_2)(D)$ are equal to zero for in the matrix $H(D)$ because these generators are part of a valid quantum convolutional code and therefore commute with each other and with all shifts of themselves and each other. For a general set of generators, this condition does not necessarily have to hold. We spend much of our effort in this article developing techniques to modify the commutation relations of a general set of generators.

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 as follows:

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

where $\odot$ denotes the shifted symplectic product. The matrix $\Omega(D)$ obeys the symmetry: $\Omega(D) = \Omega^T\left(D^{-1}\right)$. 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 necessarily equal to the null matrix.

We comment briefly on row operations. We can add one row of the check matrix $H(D)$ to another row in it without changing the error-correcting properties of the code. This property holds because the code is additive. We can even premultiply check matrix $H(D)$ by an arbitrary matrix $R(D)$ with rational polynomial entries without changing the error-correcting properties of the code. This result follows from the classical theory of convolutional coding [26]. Let $H'(D)$ denote the resulting check matrix 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

$$\Omega'(D) = R(D)\,\Omega(D)\,R^T\left(D^{-1}\right). \tag{3}$$

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.

## 3 Review of entanglement-assisted quantum block codes

We briefly review the theory of entanglement-assisted quantum coding for block quantum codes. The entanglement-assisted stabilizer formalism subsumed the stabilizer formalism [21,27] by assuming that the sender and receiver in a quantum communication protocol can share entanglement [23,24]. This technique of incorporating shared entanglement allows the sender and receiver to import arbitrary classical codes for use in quantum error correction.

Consider the ebit or Bell state $|\Phi^+\rangle^{AB}$ shared between the sender Alice $A$ and the receiver Bob $B$:

$$|\Phi^+\rangle^{AB} = \frac{|0\rangle^A |0\rangle^B + |1\rangle^A |1\rangle^B}{\sqrt{2}}.$$

Two operators that stabilize this 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 consider that 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 in a code that anticommute. We can resolve the anticommutativity in the two generators by adding an extra $X$ or $Z$ operator to the generators on the receiver's side, at the cost of an ebit of entanglement. We can find an encoding circuit starting from our unencoded information qubits and one ebit to encode the generators for the code. Detailed algorithms exist to find an encoding circuit—please see Refs. [10,23,24,28]. These algorithms can also handle a set of generators that have more complicated commutation relations. It turns out that these algorithms use the optimal number of ebits for a given set of generators [29].

Let us write the local operators $X^A$ and $Z^A$ as respective symplectic vectors $g_A$ and $h_A$:

$$g_A = \left[\, 0 \,|\, 1 \,\right], \qquad h_A = \left[\, 1 \,|\, 0 \,\right].$$

The symplectic product matrix $\Omega$ of these two symplectic vectors is as follows:

$$\Omega = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{4}$$

The above symplectic product matrix is so special for the purposes of this article 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 that the sender possesses.

A general set of generators for a quantum block code can have complicated commutation relations, but there exists a symplectic Gram–Schmidt orthogonalization procedure that simplifies the commutation relations [10,23,24,28]. Specifically, the algorithm performs row operations that do not affect the code's error-correcting properties and thus gives a set of generators that form an equivalent code. For a given set of $n-k$ generators, the commutation relations for the generators resulting from the algorithm have a special form. Their symplectic product matrix $\Omega$ has the standard form:

$$\Omega = \bigoplus_{i=1}^{c} J \oplus \bigoplus_{j=1}^{a} [0]\,, \tag{5}$$

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 $n-k$ generators are equivalent to those of $c$ halves of ebits and $a$ ancilla qubits where $a = n-k-2c$. It is straightforward to find an encoding circuit that encodes $k+c$ information qubits, $a$ ancilla qubits, and $c$ halves of ebits once we have reduced the commutation relations of the generators to be in the above standard form.

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 given above. The difference between the current techniques and the former techniques [10,23,24,28] is that the current techniques operate on generators that have a convolutional form rather than a block form. The next section shows how to expand a set of quantum convolutional generators to simplify their commutation relations. The section following the next 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.

## 4 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.

### 4.1 Example of the expansion

Let us first suppose that we have one convolutional generator:

$$\cdots \left| I \right| X \left| Z \right| I \left| \cdots \right. \tag{6}$$

This generator does not yet represent a valid quantum convolutional code because it anticommutes with a shift of itself by one qubit to the left or to the right. 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 six physical qubits with six total generators. The generators are as follows:

$$\left[\begin{array}{c|c|c|c|c|c} X & Z & I & I & I & I \\ I & X & Z & I & I & I \\ I & I & X & Z & I & I \\ I & I & I & X & Z & I \\ I & I & I & I & X & Z \\ I & I & I & I & I & X \end{array}\right]. \tag{7}$$

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:

$$\left[\begin{array}{cc|cc|cc} X & Z & I & I & I & I \\ I & X & Z & I & I & I \\ I & I & X & Z & I & I \\ I & I & I & X & Z & I \\ I & I & I & I & X & Z \\ I & I & I & I & I & X \end{array}\right]. \tag{8}$$

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:

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

The above two generators with frame size two have equivalent error-correcting properties to the original generator in (6) 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) 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:

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

The representation of the original generator in (6) as a quantum check matrix in the polynomial formalism is as follows:

$$g(D) = \left[\, D \,\middle|\, 1 \,\right]. \tag{9}$$

The two-expanded check matrix has the following polynomial representation:

$$G_2(D) = \left[\begin{array}{cc|cc} 0 & 1 & 1 & 0 \\ D & 0 & 0 & 1 \end{array}\right], \tag{10}$$

and the three-expanded check matrix has the following polynomial representation:

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

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

$$G_2'(D) = \begin{bmatrix} 1 \\ D \end{bmatrix} [g(D)] \begin{bmatrix} 1 & D^{-1} & 0 & 0 \\ 0 & 0 & 1 & D^{-1} \end{bmatrix}.$$

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 $D$:

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

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

$$G_3(D) = \begin{bmatrix} 1 \\ D \\ D^2 \end{bmatrix} [g(D)] \begin{bmatrix} 1 & \frac{1}{D} & \frac{1}{D^2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \frac{1}{D} & \frac{1}{D^2} \end{bmatrix}.$$

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.

### 4.2 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

$$H(D) = \left[ Z(D) \,\middle|\, X(D) \right].$$

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:

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

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. One can verify that the above property holds by carrying out the above operations:

$$\begin{aligned} H_l(D) &= \left\lfloor R_l\left(D^{1/l}\right) H\left(D^{1/l}\right) C_l\left(D^{1/l}\right) \right\rfloor \\ &= \left\lfloor \left[\begin{array}{c} \mathbf{D}^0 \\ \mathbf{D}^{1/l} \\ \vdots \\ \mathbf{D}^{(l-1)/l} \end{array}\right] \left[\begin{array}{c|c} Z\left(D^{1/l}\right) & X\left(D^{1/l}\right) \end{array}\right] \left[\begin{array}{ccccccc} \mathbf{D}^0 & \cdots & \mathbf{D}^{-(l-1)/l} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \mathbf{D}^0 & \cdots & \mathbf{D}^{-(l-1)/l} \end{array}\right] \right\rfloor. \end{aligned}$$

After carrying out this last operation, the "Z" matrix expands as follows

$$\left[\begin{array}{ccc} \left\lfloor Z\left(D^{1/l}\right) \right\rfloor & \left\lfloor Z\left(D^{1/l}\right) \mathbf{D}^{-1/l} \right\rfloor & \cdots & \left\lfloor Z\left(D^{1/l}\right) \mathbf{D}^{-(l-1)/l} \right\rfloor \\ \left\lfloor \mathbf{D}^{1/l} Z\left(D^{1/l}\right) \right\rfloor & \left\lfloor \mathbf{D}^{1/l} Z\left(D^{1/l}\right) \mathbf{D}^{-1/l} \right\rfloor & \cdots & \left\lfloor \mathbf{D}^{1/l} Z\left(D^{1/l}\right) \mathbf{D}^{-(l-1)/l} \right\rfloor \\ \vdots & \vdots & \ddots & \vdots \\ \left\lfloor \mathbf{D}^{(l-1)/l} Z\left(D^{1/l}\right) \right\rfloor & \left\lfloor \mathbf{D}^{(l-1)/l} Z\left(D^{1/l}\right) \mathbf{D}^{-1/l} \right\rfloor & \cdots & \left\lfloor \mathbf{D}^{(l-1)/l} Z\left(D^{1/l}\right) \mathbf{D}^{-(l-1)/l} \right\rfloor \end{array}\right],$$

and the "X" matrix expands similarly. The technique has the desired effect of expanding each matrix by a factor of $l$ because the matrix is $l$ times the original size and because the fractional powers and flooring operations pick out the terms that should appear in the $l$-expanded matrix.

## 5 Polynomial symplectic Gram–Schmidt 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 the technique for an example 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.

## 5.1 Example of the polynomial symplectic Gram–Schmidt orthogonalization procedure

### *5.1.1 Pauli picture*

Let us consider again our example from the previous section. Specifically, consider the respective expressions in (6) and (8) 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 [22] . Therefore, we can multiply the sixth generator in (8) to the fourth. We can also multiply the sixth and the fourth to the second to yield the following equivalent code:

$$\left[\begin{array}{cc|cc|cc} X & Z & I & I & I & I \\ I & X & Z & X & Z & X \\ I & I & X & Z & I & I \\ I & I & I & X & Z & X \\ I & I & I & I & X & Z \\ I & I & I & I & I & X \end{array}\right]. \tag{11}$$

We have manipulated the two-expanded matrix rather than the code in (7) because the commutation relations of the above code are equivalent to the commutation relations of the following operators:

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

We can use three ebits to encode the set of generators in (11) 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:

$$\left[\begin{array}{ccc|ccc|ccc} Z & X & Z & I & I & I & I & I & I \\ X & I & X & I & Z & X & I & Z & X \\ I & I & I & Z & X & Z & I & I & I \\ I & I & I & X & I & X & I & Z & X \\ I & I & I & I & I & I & Z & X & Z \\ I & I & I & I & I & I & X & I & X \end{array}\right].$$

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:

$$\left[\begin{array}{ccc|ccc|ccc} Z & X & Z & I & I & I & I & I & I \\ X & I & X & X & Z & I & I & I & I \\ I & I & I & Z & X & Z & I & I & I \\ I & I & I & X & I & X & X & Z & I \\ I & I & I & I & I & I & Z & X & Z \\ I & I & I & I & I & I & X & I & X \end{array}\right].$$

One can check that the operators corresponding to the second two qubits of each frame are equivalent to the desired generators in (8).

### *5.1.2 Polynomial picture*

Let us consider the convolutional generator in (6). 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 (10) 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 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:

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

The generators in the above check matrix correspond to Pauli sequences with finite weight by the P2B isomorphism. Specifically, the above two generators are equivalent to the following two Pauli sequences and all of their three-qubit shifts:

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

Bob can measure these operators because they have finite weight.

### *5.1.3 Shifted symplectic product matrix picture*

We can more easily determine the row operations that simplify the commutation relations by looking only at the shifted symplectic product matrix. We would like to perform row operations on the check matrix to reduce its corresponding shifted symplectic product matrix to the standard form in (5), 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 (9) 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 $2 \times 2$ matrix $J$ in (4) 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 (10) to determine if we can reduce it with row operations to the standard form in (5). The shifted symplectic product matrix $\Omega_{G_2}(D)$ for the two-expanded check matrix $G_2(D)$ is as follows:

$$\Omega_{G_2}(D) = \left[\begin{array}{cc} 0 & 1+D^{-1} \\ 1+D & 0 \end{array}\right].$$

We can represent 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 (3). 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 and decoding circuits, beginning from $k$ information qubits, $c$ ebits, and $a$ ancilla qubits per frame. Before that, however, we present a procedure to reduce the commutation relations of any check matrix to those of ebits and ancilla qubits.

## 5.2 The polynomial symplectic Gram–Schmidt orthogonalization procedure for general codes

We detail a polynomial version of the symplectic Gram–Schmidt orthogonalization procedure in this section. It is a generalization 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.

### *5.2.1 The shifted symplectic product matrix for an l-expanded code*

Suppose the shifted symplectic product matrix $\Omega(D)$ of a given check matrix $H(D)$ is as follows:

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

**Lemma 1** *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

$$C_l'\left(D^{1/l}\right) \equiv \left[\mathbf{D}^0\ \mathbf{D}^{-1}\ \cdots\ \mathbf{D}^{-(l-1)}\right],$$

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 $g(D)$:

$$\left\lfloor f\left(D^{1/l}\right) g\left(D^{1/l}\right)\right\rfloor = \sum_{i=0}^{l-1}\left\lfloor D^{-i/l} f\left(D^{1/l}\right)\right\rfloor\left\lfloor D^{i/l} g\left(D^{1/l}\right)\right\rfloor. \tag{12}$$

Now consider the product $X_l(D)\, Z_l^T\left(D^{-1}\right)$:

$$\begin{aligned}
&X_l(D)\, Z_l^T\left(D^{-1}\right)\\
&= \left\lfloor R_l\left(D^{1/l}\right) X\left(D^{1/l}\right) C_l'\left(D^{1/l}\right)\right\rfloor\left(\left\lfloor R_l\left(D^{1/l}\right) Z\left(D^{1/l}\right) C_l'\left(D^{1/l}\right)\right\rfloor\right)^T\Big|_{D=D^{-1}}\\
&= \left\lfloor R_l\left(D^{1/l}\right) X\left(D^{1/l}\right) C_l'\left(D^{1/l}\right)\right\rfloor\left\lfloor C_l'^T\left(D^{-1/l}\right) Z^T\left(D^{-1/l}\right) R_l^T\left(D^{-1/l}\right)\right\rfloor\\
&= \sum_{i=0}^{l-1}\left\lfloor D^{-i/l} R_l\left(D^{1/l}\right) X\left(D^{1/l}\right)\right\rfloor\left\lfloor D^{i/l} Z^T\left(D^{-1/l}\right) R_l^T\left(D^{-1/l}\right)\right\rfloor\\
&= \left\lfloor R_l\left(D^{1/l}\right) X\left(D^{1/l}\right) Z^T\left(D^{-1/l}\right) R_l^T\left(D^{-1/l}\right)\right\rfloor,
\end{aligned}$$

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 (12). Our final step is to use the invariance of the flooring operation with respect to matrix addition:

$$\begin{aligned}
\Omega_l(D) &= X_l(D) Z_l^T\left(D^{-1}\right) + Z_l(D) X_l^T\left(D^{-1}\right), \\
&= \left\lfloor R_l\left(D^{1/l}\right) X\left(D^{1/l}\right) Z^T\left(D^{-1/l}\right) R_l^T\left(D^{-1/l}\right)\right\rfloor \\
&\quad + \left\lfloor R_l\left(D^{1/l}\right) Z\left(D^{1/l}\right) X^T\left(D^{-1/l}\right) R_l^T\left(D^{-1/l}\right)\right\rfloor \\
&= \left\lfloor \begin{array}{c} R_l\left(D^{1/l}\right) X\left(D^{1/l}\right) Z^T\left(D^{-1/l}\right) R_l^T\left(D^{-1/l}\right) \\ + R_l\left(D^{1/l}\right) Z\left(D^{1/l}\right) X^T\left(D^{-1/l}\right) R_l^T\left(D^{-1/l}\right) \end{array}\right\rfloor \\
&= \left\lfloor R_l\left(D^{1/l}\right) \Omega\left(D^{1/l}\right) R_l^T\left(D^{-1/l}\right)\right\rfloor.
\end{aligned}$$

*5.2.2 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 (5).

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:

1. There is a generator $h_i(D)$ such that $\left(h_i \odot h_j\right)(D) = 0$ 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:

$$\begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & h_{2,2} & \cdots & h_{2,r} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & h_{r,2} & \cdots & h_{r,r} \end{bmatrix} = [0] \oplus \begin{bmatrix} h_{2,2} & \cdots & h_{2,r} \\ \vdots & \ddots & \vdots \\ h_{r,2} & \cdots & h_{r,r} \end{bmatrix},$$

where [0] is the one-element zero matrix and we use the shorthand $h_{i,j} = \left(h_i \odot h_j\right)(D)$. We remove generator $h_1(D)$ from check matrix $H(D)$ and continue to step two below for the remaining generators in the matrix.

2. There are two generators $h_i(D)$ and $h_j(D)$ for which $\left(h_i \odot h_j\right)(D) = D^m$ for some integer $m$ and $(h_i \odot h_i)(D) = \left(h_j \odot h_j\right)(D) = 0$. In this case these generators correspond exactly to 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 $\left(h_i \odot h_j\right)(D) = 1$. 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:

$$h_i'(D) \equiv h_i(D) + (h_i \odot h_2)(D) \cdot h_1(D) + (h_i \odot h_1)(D) \cdot h_2(D) \quad \text{for all } i \in \{3, \ldots, r\}.$$

The shifted symplectic product matrix becomes as follows under these row operations:

$$\begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & h_{3,3}' & \cdots & h_{3,r}' \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & h_{r,3}' & \cdots & h_{r,r}' \end{bmatrix} = J \oplus \begin{bmatrix} h_{3,3}' & \cdots & h_{3,r}' \\ \vdots & \ddots & \vdots \\ h_{r,3}' & \cdots & h_{r,r}' \end{bmatrix}.$$

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 $h_3'(D), \ldots, h_r'(D)$. 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.

3. There are two generators $h_i(D)$ and $h_j(D)$ for which $(h_i \odot h_i)(D) = \left(h_j \odot h_j\right)(D) = 0$ but $\left(h_i \odot h_j\right)(D) \neq D^m$ for some $m$ and $\left(h_i \odot h_j\right)(D) \neq 0$. Multiply generator $h_j(D)$ by $1/\left(h_j \odot h_i\right)(D)$. Generator $h_j(D)$ 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: $(h_i \odot h_i)(D) = \left(h_j \odot h_j\right)(D) = 0$ 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.

4. 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. [23,24] 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 Encoding and decoding entanglement-assisted quantum convolutional codes

This section proves the main theorem of this paper. 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 (5). 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. The finite-depth property of the operations in the decoding circuit guarantees that catastrophic error propagation does not occur when decoding the encoded qubit stream. We do not review finite-depth or infinite-depth operations here but instead refer the reader to our previous article [11] that has a detailed discussion of these operations.

**Theorem 1** *Suppose we have a set of quantum convolutional generators in the* $n - k \times 2n$*-dimensional matrix* $H(D)$ *where*

$$H(D) = \left[ \, Z(D) \, \middle| \, X(D) \, \right].$$

*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 the 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* (5) *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* [11] *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. [4,3] 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:

$$\left[\begin{array}{cc|cc} Z''(D) & Z'(D) & 0 & X'(D) \\ \Gamma(D) & 0 & 0 & 0 \end{array}\right], \tag{13}$$

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 [3]. 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 (5) 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:

$$H'(D) = \left[\, Z(D) \,\middle|\, X(D) \,\right],$$

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

$$\begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix}, \tag{14}$$

where each identity and null matrix in the above matrix is $c \times c$-dimensional. We can employ the algorithm from Ref. [4] 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 (14) 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 (14)—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 (14). Perform Hadamard gates on the first $c$ qubits. The check matrix becomes

$$\left[\begin{array}{cc|cc} I & 0 & 0 & 0 \\ X_1'(D) & Z_2'(D) & I & X_2'(D) \end{array}\right]. \tag{15}$$

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:

$$\left[\begin{array}{ccc|ccc} I & I & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I & I & 0 \end{array}\right], \tag{16}$$

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" matrix [11]:

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

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 [11] 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 (15) 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:

$$\left[\begin{array}{ccc|ccc} \Gamma\left(D^{-1}\right) & I & 0 & 0 & 0 & 0 \\ 0 & A(D) & Z'_{2,N}(D) & I & \Gamma(D) & X'_{2,N}(D) \end{array}\right]. \tag{17}$$

We are done with the encoding algorithm. Alice begins with a set of ebits and performs the encoding operations detailed in (16–17) and then performs the finite-depth operations detailed in (13–15) 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

$$\left[\begin{array}{cc|ccc} I & \Gamma^{-1}\left(D^{-1}\right) & 0 & 0 & 0 \\ B(D) & A(D) & 0 & I & \Gamma(D) \; 0 \end{array}\right],$$

and the information-qubit matrix becomes

$$\left[\begin{array}{cc|ccc} Z''_{2,N}(D) & Z''_{2,N}(D)\,\Gamma^{-1}\left(D^{-1}\right) & I & 0 & 0 & 0 \\ X''_{2,N}(D) & X''_{2,N}(D)\,\Gamma^{-1}\left(D^{-1}\right) & 0 & 0 & 0 & I \end{array}\right].$$

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 (13–15). We have now finished the algorithm for the decoding circuit because the logical operators for the information qubits appear in their original form.

### 6.1 Discussion

Similar practical issues arise in these circuits as we discussed previously in Ref. [11]. Encoding circuits with infinite-depth operations 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 Ref. [11]. Some of the codes may have a simpler form in which the encoding circuit requires finite-depth operations only. Reference [12] gives an example of a code with this structure. These codes fall within the first class of codes discussed in Ref. [11] 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.

### 6.2 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 [11] where $n$ and $k$ are the parameters from the imported classical code. The rate should get a significant boost from entanglement—it 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:

$$\tilde{H}(D) = \gamma\left(\begin{bmatrix} \omega H(D) \\ \bar{\omega} H(D) \end{bmatrix}\right),$$

where $\gamma$ denotes the isomorphism between elements of $GF(4)$ and symplectic binary vectors that represent Pauli matrices. Specifically, $\gamma^{-1}(h) = \omega h_x + \bar{\omega} h_z$ where $h$ is a symplectic binary vector and $h_x$ and $h_z$ denote its "X" and "Z" parts, respectively. We use this construction for the example in the next section.

## 7 Example

We take the convolutional generators from Ref. [10] as our example. Reference [23] originally used these generators in a block code. Consider the following generator of a check matrix for a classical convolutional code over GF(4):

$$(\cdots|0000|1\bar{\omega}10|1101|0000|\cdots). \tag{18}$$

The generators of the generator matrix for this code are as follows:

$$(\cdots|0000|1011|0000|0000|\cdots), \tag{19}$$
$$(\cdots|0000|1001|1010|0000|\cdots), \tag{20}$$
$$(\cdots|0000|01\bar{\omega}1|0000|0000|\cdots), \tag{21}$$

because they are all in the null space of the above check matrix generator. The distance of this classical convolutional code is three because the first generator of the above three has the minimal weight of three. The entanglement-assisted quantum convolutional code we produce below inherits this distance.

We produce two quantum convolutional generators by mutliplying the above generator by $\omega$ and $\bar{\omega}$ and applying the following map:

| $GF(4)$ | $\Pi$ |
|---|---|
| 0 | $I$ |
| $\omega$ | $X$ |
| 1 | $Y$ |
| $\bar{\omega}$ | $Z$ |

, (22)

where $\Pi$ denotes the group of Pauli matrices for one qubit. The resulting quantum convolutional generators are as follows:

$$\begin{aligned}&(\cdots|IIII|ZXZI|ZZIZ|IIII|\cdots),\\&(\cdots|IIII|XYXI|XXIX|IIII|\cdots).\end{aligned}\tag{23}$$

It is not a CSS code because the original classical generator has support on the element $\bar{\omega}$. 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 "Z" matrix $Z(D)$ of the two-expanded check matrix is as follows:

$$Z(D)=\begin{bmatrix}1 & 0 & 1 & 0 & 1 & 1 & 0 & 1\\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\ D & D & 0 & D & 1 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\end{bmatrix},$$

and the "X" matrix $X(D)$ of the two-expanded check matrix is as follows:

$$X(D)=\begin{bmatrix}0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1\\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\ D & D & 0 & D & 1 & 1 & 1 & 0\end{bmatrix}.$$

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:

$$H_2(D) = \left[ \, Z_2(D) \, \middle| \, X_2(D) \, \right], \tag{24}$$

where

$$Z_2^T(D) = \begin{bmatrix} 0 & D & 1 & \frac{D^2+D}{D^2+D+1} \\ 1 & D & \frac{1}{D}+1 & \frac{D^2+D}{D^2+D+1} \\ 0 & 0 & 1 & 0 \\ 0 & D & 0 & \frac{D^2+D}{D^2+D+1} \\ 0 & 1 & 1 & \frac{D+1}{D^2+D+1} \\ 0 & 0 & 1 & \frac{1}{D^2+D+1} \\ 0 & 1 & 0 & \frac{D+1}{D^2+D+1} \\ 0 & 0 & 1 & 0 \end{bmatrix}, \qquad X_2^T(D) = \begin{bmatrix} 1 & 0 & 1+D^{-1} & \frac{D}{D^2+D+1} \\ 1 & 0 & \frac{1}{D} & \frac{D}{D^2+D+1} \\ 1 & 0 & 1+D^{-1} & 0 \\ 0 & 0 & 0 & \frac{D}{D^2+D+1} \\ 1 & 0 & 1+D^{-1} & \frac{1}{D^2+D+1} \\ 1 & 1 & 1+D^{-1} & \frac{D}{D^2+D+1} \\ 0 & 0 & 0 & \frac{1}{D^2+D+1} \\ 1 & 0 & 1+D^{-1} & 0 \end{bmatrix}.$$

(We give the transposition of the above two matrices because they otherwise would not fit in the space above.) 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 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:

$$H_0\left(D\right) = \left[\begin{array}{c|c} Z_0\left(D\right) & X_0\left(D\right) \end{array}\right],$$

where

$$Z_0\left(D\right) = \left[\begin{array}{cccccccccc} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right], \qquad X_0\left(D\right) = \left[\begin{array}{cccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right].$$

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):

$$\begin{aligned} & C\left(1, 4, D + D^2\right) C\left(1, 5, 1 + D^2\right) C\left(1, 6, 1\right) C\left(1, 7, 1 + D\right) \\ & C\left(2, 4, D\right) C\left(2, 5, 1 + D\right) C\left(2, 6, 1\right) \\ & H\left(3, \ldots, 8\right) C\left(1, 4, D + D^2 + D^4\right) C\left(1, 5, D^2\right) \\ & C\left(1, 6, 1 + D\right) C\left(1, 7, D^2\right) C\left(2, 4, D + D^2\right) \\ & C\left(2, 5, 1 + D\right) C\left(2, 6, 1\right) C\left(2, 7, 1 + D\right), \end{aligned}$$

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 $1/f\left(D\right)$. Alice performs the following infinite-depth operations:

$$\begin{aligned} & H\left(1, 2\right) C\left(1, \frac{1}{1 + D^{-1} + D^{-2}}\right) \\ & C\left(2, \frac{1}{1 + D^{-1} + D^{-2}}\right) H\left(1, 2\right). \end{aligned}$$

She then finishes the encoding circuit with the following finite-depth operations:

$$\begin{aligned} & H\left(1, 2\right) C\left(2, 3, 1\right) C\left(2, 5, 1\right) C\left(2, 6, 1\right) C\left(2, 8, 1\right) \\ & P\left(2\right) H\left(3, \ldots, 8\right) S\left(2, 3\right) \\ & C\left(1, 2, 1\right) C\left(1, 3, 1\right) C\left(1, 5, 1\right) C\left(1, 6, 1\right) C\left(1, 8, 1\right) P\left(2\right). \end{aligned} \tag{25}$$

The code she encodes has equivalent error-correcting properties to the check matrix in (24).

Bob performs the following operations in the decoding circuit. He first performs the operations in (25) in reverse order. He then performs the following finite-depth operations:

$$
\begin{aligned}
&C\left(B1, 4, D + D^2 + D^4\right) C\left(B1, 5, D^2\right) \\
&C\left(B1, 6, 1 + D\right) C\left(B1, 7, D^2\right) C\left(B2, 4, D + D^2\right) \\
&C\left(B2, 5, 1 + D\right) C\left(B2, 6, 1\right) C\left(B2, 7, 1 + D\right) \\
&H\left(3, \ldots, 8\right) \\
&C\left(B1, 4, D + D^2\right) C\left(B1, 5, 1 + D^2\right) C\left(B1, 6, 1\right) \\
&C\left(B1, 7, 1 + D\right) C\left(B2, 4, D\right) \\
&C\left(B2, 5, 1 + D\right) C\left(B2, 6, 1\right).
\end{aligned}
$$

The information qubits then appear at the output of this online decoding circuit.

## 8 Conclusion and current work

There are several differences between the methods used for general, non-CSS codes discussed in this article and the CSS codes used in our previous article [11]. 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 Sect. 4 in order to determine how to use ebits efficiently for codes in this article. There was also no need for an explicit Gram–Schmidt orthogonalization procedure in Ref. [11]. The Smith algorithm implicitly produced the row operations necessary for symplectic orthogonalization.

We do have some methods that do not require expansion of a check matrix or an explicit Gram–Schmidt procedure [31], but these methods do not make efficient use of entanglement and have a lower rate of noiseless qubit channel simulation and higher rate of entanglement consumption than the codes discussed in this article. 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 [30]. These other codes are "grandfather" codes in the sense of Refs. [12,32] because they consume entanglement to send both quantum and classical information.

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 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 mentioned in the previous paragraph 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 article 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 conjecture about optimal entanglement use in general entanglement-assisted quantum convolutional codes from Ref. [29] also remains an open question. The conjecture is that the optimal number of ebits required per frame is equal to the following expression:

$$\operatorname{rank}\left(Z\left(D\right)X^{T}\left(D^{-1}\right)+X\left(D\right)Z^{T}\left(D^{-1}\right)\right)/2,$$

where $Z\left(D\right)$ and $X\left(D\right)$ are the respective "Z" and "X" matrices for a given set of quantum convolutional generators. It is clear that this formula holds after we have expanded the original set of generators. The proof technique follows from the proof technique outlined in Ref. [29]. But we are not sure how to apply this formula to an initial set of unexpanded generators.

The techniques developed in this article represent a useful way for encoding quantum information. The next step should be to combine the theory in this article with Poulin et al.'s recent theory of quantum serial-turbo coding [20].

**Acknowledgments** M.M.W. acknowledges support from NSF Grant No. CCF-0545845, and T.A.B. acknowledges support from NSF Grant No. CCF-0448658 and No. CCF-0830801.

## References

1. Ollivier, H., Tillich, J.-P.: Description of a quantum convolutional code. Phys. Rev. Lett. **91**(17), 177902 (2003)
2. Ollivier, H., Tillich, J.-P.: Quantum convolutional codes: fundamentals. arXiv:quant-ph/0401134 (2004)
3. Grassl, M., Rötteler, M.: Noncatastrophic encoders and encoder inverses for quantum convolutional codes. In: IEEE International Symposium on Information Theory (quant-ph/0602129) (2006)
4. Grassl, M., Rötteler, M.: Quantum convolutional codes: encoders and structural properties. In: Proceedings of the Forty-Fourth Annual Allerton Conference, pp. 510–519 (2006)
5. Grassl, M., Rötteler, M.: Constructions of quantum convolutional codes. In: Proceedings of the IEEE International Symposium on Information Theory, pp. 816–820 (2007)
6. Forney, G.D., Guha, S.: Simple rate-1/3 convolutional and tail-biting quantum error-correcting codes. In: IEEE International Symposium on Information Theory (arXiv:quant-ph/0501099) (2005)
7. Forney, G.D., Grassl, M., Guha, S.: Convolutional and tail-biting quantum error-correcting codes. IEEE Trans. Inf. Theory **53**, 865–880 (2007)
8. Aly, S.A., Grassl, M., Klappenecker, A., Roetteler, M., Sarvepalli, P.K.: Quantum convolutional BCH codes. In: 10th Canadian Workshop on Information Theory (arXiv:quant-ph/0703113), pp. 180–183 (2007)
9. Aly, S.A., Klappenecker, A., Sarvepalli, P.K.: Quantum convolutional codes derived from Reed-Solomon and Reed-Muller codes. arXiv:quant-ph/0701037 (2007)

10. Wilde, M.M., Krovi, H., Brun, T.A.: Convolutional entanglement distillation. To appear in the International Symposium on Information Theory, arXiv:0708.3699, Austin, Texas, USA June (2010)
11. Wilde, M.M., Brun, T.A.: Entanglement-assisted quantum convolutional coding. Phys. Rev. A **81**, 042333 (2010)
12. Wilde, M.M., Brun, T.A.: Unified quantum convolutional coding. In: Proceedings of the IEEE International Symposium on Information Theory (arXiv:0801.0821), pp. 359–363. Toronto, ON, Canada, July 2008
13. Lloyd, S.: Capacity of the noisy quantum channel. Phys. Rev. A **55**(3), 1613–1622 (1997)
14. Shor, P.W.: The quantum channel capacity and coherent information. In: Lecture Notes, MSRI Workshop on Quantum Computation (2002)
15. Devetak, I.: The private classical capacity and quantum capacity of a quantum channel. IEEE Trans. Inf. Theory **51**, 44–55 (2005)
16. Hayden, P., Horodecki, M., Winter, A., Yard, J.: A decoupling approach to the quantum capacity. Open Syst. Inf. Dyn. **15**, 7–19 (2008)
17. Klesse, R.: A random coding based proof for the quantum coding theorem. Open Syst. Inf. Dyn. **15**, 21–45 (2008)
18. Horodecki, M., Lloyd, S., Winter, A.: Quantum coding theorem from privacy and distinguishability. Open Syst. Inf. Dyn. **15**, 47–69 (2008)
19. Hayden, P., Shor, P.W., Winter, A.: Random quantum codes from Gaussian ensembles and an uncertainty relation. Open Syst. Inf. Dyn. **15**, 71–89 (2008)
20. Poulin, D., Tillich, J.-P., Ollivier, H.: Quantum serial turbo-codes. IEEE Trans. Inf. Theory **55**(6), 2776–2798 (2009)
21. Calderbank, A.R., Rains, E.M., Shor, P.W., Sloane, N.J.A.: Quantum error correction via codes over GF(4). IEEE Trans. Inf. Theory **44**, 1369–1387 (1998)
22. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information. Cambridge University Press, Cambridge (2000)
23. Brun, T.A., Devetak, I., Hsieh, M.-H.: Correcting quantum errors with entanglement. Science **314**(5798), 436–439 (2006)
24. Brun, T.A., Devetak, I., Hsieh, M.-H.: Catalytic quantum error correction. arXiv:quant-ph/0608027, August (2006)
25. Grassl, M.: Convolutional and block quantum error-correcting codes. In: IEEE Information Theory Workshop, pp. 144–148, Chengdu, October (2006)
26. Johannesson, R., Zigangirov, K.Sh.: Fundamentals of Convolutional Coding. Wiley-IEEE Press, (1999)
27. Gottesman, D.: Stabilizer Codes and Quantum Error Correction. PhD thesis, California Institute of Technology (1997)
28. Shaw, B., Wilde, M.M., Oreshkov, O., Kremsky, I., Lidar, D.: Encoding one logical qubit into six physical qubits. Phys. Rev. A **78**, 012337 (2008)
29. Wilde, M.M., Brun, T.A.: Optimal entanglement formulas for entanglement-assisted quantum coding. Phys. Rev. A **77**, 064302 (2008)
30. Bennett, C.H., Wiesner, S.J.: Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states. Phys. Rev. Lett. **69**(20), 2881–2884 (1992)
31. Wilde, M.M., Brun, T.A.: Extra shared entanglement reduces memory demand in quantum convolutional coding. Phys. Rev. A **79**(3), 032313 (2009)
32. Kremsky, I., Hsieh, M.-H., Brun, T.A.: Classical enhancement of quantum-error-correcting codes. Phys. Rev. A **78**(1), 012341 (2008)