Minimal-Memory, Noncatastrophic, Polynomial-Depth Quantum Convolutional Encoders

Monireh Houshmand, Saied Hosseini-Khayat, and Mark M. Wilde, *Member, IEEE*

Abstract—Quantum convolutional coding is a technique for encoding a stream of quantum information before transmitting it over a noisy quantum channel. Two important goals in the design of quantum convolutional encoders are to minimize the memory required by them and to avoid the catastrophic propagation of errors. In a previous paper, we determined minimal-memory, noncatastrophic, polynomial-depth encoders for a few exemplary quantum convolutional codes. In this paper, we elucidate a general technique for finding an encoder of an arbitrary quantum convolutional code such that the encoder possesses these desirable properties. We also provide an elementary proof that these encoders are nonrecursive. Finally, we apply our technique to many quantum convolutional codes from the literature.

Index Terms—Catastrophicity, memory commutativity matrix, minimal memory, quantum convolutional codes.

I. INTRODUCTION

A quantum convolutional code is a particular type of quantum error-correcting code [2]–[4] that is well suited for the regime of quantum communication [5]–[7]. In this regime, we assume that a sender and receiver have free access to local, noiseless quantum computers, and the only source of noise is due to a quantum communication channel connecting the sender to the receiver. The advantage of the convolutional approach to quantum error correction is that the repeated application of the same unitary operation encodes a stream of quantum information, and the complexity of the decoding algorithm is linear in the length of the qubit stream [8]. Many researchers have generated a notable literature on this topic, addressing various issues such as code constructions [9], [10], encoders and decoders [7], [11], [12], and alternate paradigms with entanglement assistance [13]–[15] or with gauge qubits and classical bits [16]. Perhaps more importantly for the quantum communication paradigm, quantum convolutional codes are the constituents of a quantum serial turbo code [8], and these codes are among the highest performing codes in both the standard [8] and entanglement-assisted settings [17].

One of the most important parameters for a quantum convolutional encoder is the size of its memory, defined as the number of qubits that are fed from its output into the next round of encoding. A quantum convolutional encoder with a large memory is generally more difficult to implement because it requires coherent control of a large number of qubits. Furthermore, the complexity of the decoding algorithm for a quantum convolutional code is linear in the length of the qubit stream, but it is exponential in the size of the memory [8]. The decoding algorithm will thus have more delay for a larger memory, and this could potentially lead to further errors in the more practical setting where there is local noise at the receiving end. Therefore, an interesting and legitimate question is to determine the minimal number of memory qubits required to implement a given quantum convolutional code.

Another property that any good quantum convolutional *decoder* should possess is noncatastrophicity. As the name suggests, the consequences of decoding with a catastrophic decoder are disastrous—it can propagate some uncorrected errors infinitely throughout the decoded information qubit stream and the receiver will not know that this is happening. We should clarify that catastrophicity is a property of the decoder because the only errors that occur in the communication paradigm are those due to the channel, and thus, the decoder (and not the encoder) has the potential to propagate uncorrected errors. We could say just as well that catastrophicity is a property of an encoder if the decoder is the exact inverse of the encoder (as is the case in [8]). Either way, since the property of noncatastrophicity is essential and having a minimal memory is highly desirable, we should demand for our encoders and decoders to be both minimal-memory and noncatastrophic.

The minimal-memory/noncatastrophic question is essentially understood for the case of irreversible encoders for classical convolutional codes [18], [19] by making use of ideas in linear system theory. These results at the surface do not appear to address the case of reversible classical encoders, which would be more relevant for answering the minimal-memory/noncatastrophic question in the quantum case.

In [7], [11], and [12], Grassl and Rötteler proposed an algorithm to construct noncatastrophic quantum circuits for encoding quantum convolutional codes. Their encoders there do not have a convolutional structure, and their work did not address how much quantum memory their encoders would require for implementation. In follow-up work, we found a minimal-memory realization of a Grassl–Rötteler encoder by performing a longest path search through a "commutativity graph" that corresponds to the encoder [20], [21]. Our approach was generally suboptimal because there exist many encoders for a given convolutional code—starting from a Grassl–Rötteler encoder and finding the minimal-memory representation for it does not necessarily lead to a minimal-memory encoder among all possible representations of the code. Also, the complexity of the

Grassl–Rötteler algorithm for computing an encoding circuit could be exponential in general, resulting in an encoding circuit with exponential depth [7], [11].

The purpose of this paper is to elucidate the technique of [1] in full detail. The encoders resulting from our technique are convolutional and possess the aforementioned desirable properties simultaneously—they are minimal-memory, noncatastrophic, and have an $O(n^2)$ depth, where n is the frame size of code. In addition, we prove that the resulting encoders are nonrecursive. Poulin *et al.* [8] already proved that all noncatastrophic encoders are nonrecursive, but our proof of this fact for the encoders studied here is arguably much simpler than the proof of Theorem 1 in [8]. Interestingly, the essence of our technique for determining an encoder is commutation relations, which often are lurking behind many fundamental questions in quantum information theory. The commutation relations that are relevant for our technique are those for the Pauli operators acting on the memory qubits. An upshot of our technique for minimizing memory is that it is similar to one in [22] and [23] for finding the minimal number of entangled bits required in an entanglement-assisted quantum error-correcting code [24]. This result is perhaps unsurprising in hindsight, given that an encoder generally entangles information qubits and ancilla qubits with the memory qubits before sending encoded qubits out over the channel.

This paper is organized as follows. For the sake of completeness, we begin by reviewing the definition of a quantum convolutional code. We then review our technique from [1] for determining a quantum convolutional encoder for a given set of stabilizer generators, and we prove a theorem concerning the consistency of these generators with commutation relations of the encoder. Section III-B introduces the idea of a memory commutativity matrix that is rooted in ideas from [1]. Section V reviews the state diagram for a quantum convolutional encoder [8], [19], [25], [26], and the section after it reviews catastrophicity. All of the above sections feature a "running example" that is helpful in illustrating the main concepts. Section V-A details our main results, which are sufficient conditions for any quantum convolutional encoder to be both minimal-memory and noncatastrophic. These sufficient conditions apply to the memory commutativity matrix of the quantum convolutional encoder. Section VI then proves that the encoders studied in Section V-A are nonrecursive. Finally, we conclude in Section VII with a summary and a list of open questions, and the appendix of [27] gives many examples of quantum convolutional codes from [6] and [12] for which we can find minimal-memory, noncatastrophic encoders.

II. QUANTUM CONVOLUTIONAL CODES

In this section, we recall some standard facts and then review the definition of a quantum convolutional code. A Pauli sequence is a countably infinite tensor product of Pauli matrices

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

where each operator A_i in the sequence is an element of the Pauli group $\Pi \equiv \{I, X, Y, Z\}$. Let $\Pi^{\mathbb{Z}^+}$ denote the set of all Pauli sequences. A Pauli sequence is finite-weight if only finitely many operators A_i in the sequence are equal to X, Y, or Z, and it is an infinite-weight sequence otherwise.
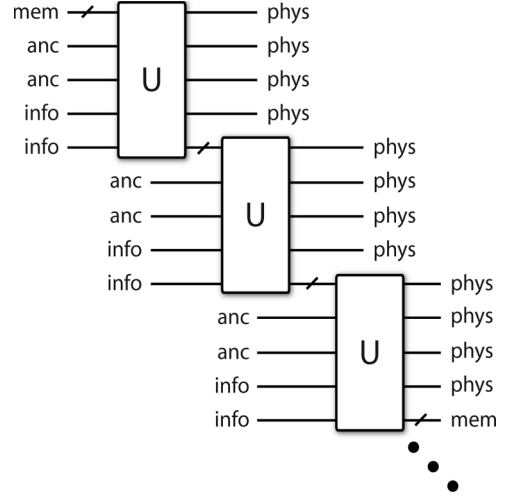

Fig. 1. Encoder U for a quantum convolutional code that has four physical qubits for every two information qubits. The encoder U acts on m memory qubits, two ancilla qubits, and two information qubits to produce four output physical qubits to be sent over the channel and m output memory qubits to be fed into the next round of encoding.

Definition 1 (Quantum Convolutional Code): A rate-k/n quantum convolutional code admits a representation with a basic set $\mathcal{G}_0$ of $n - k$ generators and all of their n-qubit shifts

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

In order to form a quantum convolutional code, these generators should commute with themselves and all of the n-qubit shifts of themselves and the other generators.

Equivalently, a rate-k/n quantum convolutional code is specified by $n - k$ generators $h_1, h_2, \ldots, h_{n-k}$, where

$$
\begin{array}{llllll}
h_1 & = & h_{1,1} & | & h_{1,2} & | \ldots | & h_{1,l_1} \\
h_2 & = & h_{2,1} & | & h_{2,2} & | \ldots | & h_{2,l_2} \\
\vdots & & \vdots & \vdots & \vdots & \vdots & \vdots \\
h_{n-k} & = & h_{n-k,1} & | & h_{n-k,2} & | \cdots | & h_{n-k,l_{n-k}}.
\end{array}
\tag{1}
$$

Each entry $h_{i,j}$ is an n-qubit Pauli operator and l_i is the degree of generator h_i (in general, the degrees l_i can be different from each other). We obtain the other generators of the code by shifting the above generators to the right by multiples of n qubits. (In the above, note that the entries $h_{1,l_1}, h_{2,l_2}, \ldots, h_{n-k,l_{n-k}}$ are not required to be in the same column, but we have written it in the above way for convenience.)

We select the first quantum convolutional code from [12, Fig. 1] as our running example for this paper. This code has the following two generators:

$$
\begin{array}{llllll}
h_1 & = & XXXX & | & XXIX & | & IXII & | & IIXX \\
h_2 & = & ZZZZ & | & ZZIZ & | & IZII & | & IIZZ
\end{array}
\tag{2}
$$

with $n = 4$ and $n - k = 2$, implying that the code encodes $k = 2$ information qubits for every four physical qubits. Observe that the above generators commute with each other and with the generators resulting from all possible four-qubit shifts of the above generators.

III. PROPOSED ENCODING ALGORITHM

Fig. 1 depicts an example of an encoder for a quantum convolutional code. The encoder depicted there can encode our

running example in (2) that has four physical qubits for every two information qubits. More generally, a convolutional encoder acts on some number m of memory qubits, $n - k$ ancilla qubits, and k information qubits, and it produces n output physical qubits and m output memory qubits to be fed into the next round of encoding.

For our example in (2), the unencoded qubit stream might have the following form:

$$|0\rangle \, |0\rangle \, |\phi_1\rangle \, |\phi_2\rangle \, |0\rangle \, |0\rangle \, |\phi_3\rangle \, |\phi_4\rangle \cdots \qquad (3)$$

so that an ancilla qubit appears as every first and second qubit and an information qubit appears as every third and fourth qubit (generally, these information qubits can be entangled with each other and even with an inaccessible reference system, but we write them as product states for simplicity). A particular set of stabilizer generators for the unencoded qubit stream in (3) is as follows (along with all of their four-qubit shifts):

$$\begin{array}{c|c|c|c} ZIII & IIII & IIII & IIII \\ IZII & IIII & IIII & IIII \end{array} \qquad (4)$$

so that the states in (3) are in the simultaneous $+1$-eigenspace of the above operators and all of their four-qubit shifts.

The objective of the convolutional encoder is to transform these "unencoded" Pauli Z operators to the encoded stabilizer generators in (2). That is, it should be some Clifford transformations[1] of the following form:

Mem.	Anc.		Info.		Phys.				Mem.
$I^{\otimes m}$	Z	I	I	I	X	X	X	X	$g_{1,1}$
$g_{1,1}$	I	I	I	I	X	X	I	X	$g_{1,2}$
$g_{1,2}$	I	I	I	I	I	X	I	I	$g_{1,3}$
$g_{1,3}$	I	I	I	I	I	I	X	X	$I^{\otimes m}$
$I^{\otimes m}$	I	Z	I	I	Z	Z	Z	Z	$g_{2,1}$
$g_{2,1}$	I	I	I	I	Z	Z	I	Z	$g_{2,2}$
$g_{2,2}$	I	I	I	I	I	Z	I	I	$g_{2,3}$
$g_{2,3}$	I	I	I	I	I	I	Z	Z	$I^{\otimes m}$

$$\qquad (5)$$

where, as a visual aid, we have separated the memory qubits, ancilla qubits, and information qubits at the input with a vertical bar and we have done the same for the physical qubits and memory qubits at the output. A horizontal bar separates the entries of the encoder needed to encode the first generator from the entries needed to encode the second generator. Each $g_{i,j}$ is a Pauli operator acting on some number m of memory qubits—these operators should be consistent with the input–output commutation relations of the encoder (more on this later). We stress that the above input–output relations only partially specify the encoder such that it produces a code with the stabilizer generators in (2), and there is still a fair amount of freedom remaining in the encoding.

In the general case, a convolutional encoder should transform a unencoded Pauli Z operator acting on the ith ancilla qubit to the ith stabilizer generator h_i in (1). The first application of the encoder U results in an intermediate, unspecified Pauli operator $g_{i,1}$ acting on the m output memory qubits. The second application of the encoder U results in an intermediate, unspecified

---

[1]A Clifford transformation is a unitary operator that preserves the Pauli group under unitary conjugation.

Pauli operator $g_{i,2}$ acting on the m output memory qubits and so on. The shift invariance of the overall encoding guarantees that shifts of the unencoded Z Pauli operators transform to appropriate shifts of the generators. A convolutional encoder for the code should perform the following transformation:

Mem.	Anc.	Info.	Phys.	Mem.
$I^{\otimes m}$	Z_1	$I^{\otimes k}$	$h_{1,1}$	$g_{1,1}$
$g_{1,1}$	$I^{\otimes n-k}$	$I^{\otimes k}$	$h_{1,2}$	$g_{1,2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
g_{1,l_1-2}	$I^{\otimes n-k}$	$I^{\otimes k}$	h_{1,l_1-1}	g_{1,l_1-1}
g_{1,l_1-1}	$I^{\otimes n-k}$	$I^{\otimes k}$	h_{1,l_1}	$I^{\otimes m}$
$I^{\otimes m}$	Z_2	$I^{\otimes k}$	$h_{2,1}$	$g_{2,1}$
$g_{2,1}$	$I^{\otimes n-k}$	$I^{\otimes k}$	$h_{2,2}$	$g_{2,2}$
$\vdots$	$\vdots$	$\vdots$ $\rightarrow$	$\vdots$	$\vdots$
g_{2,l_2-2}	$I^{\otimes n-k}$	$I^{\otimes k}$	h_{2,l_2-1}	g_{2,l_2-1}
g_{2,l_2-1}	$I^{\otimes n-k}$	$I^{\otimes k}$	h_{2,l_2}	$I^{\otimes m}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$I^{\otimes m}$	Z_s	$I^{\otimes k}$	$h_{s,1}$	$g_{s,1}$
$g_{s,1}$	$I^{\otimes n-k}$	$I^{\otimes k}$	$h_{s,2}$	$g_{s,2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
g_{s,l_s-2}	$I^{\otimes n-k}$	$I^{\otimes k}$	h_{s,l_s-1}	g_{s,l_s-1}
g_{s,l_s-1}	$I^{\otimes n-k}$	$I^{\otimes k}$	h_{s,l_s}	$I^{\otimes m}$

$$\qquad (6)$$

where m is some unspecified number of memory qubits, k is the number of information qubits, $n - k$ is the number of ancilla qubits, and we make the abbreviation $s \equiv n - k$. Again, the above transformation only partially specifies the encoding. Also, note that it is not necessary for the $g_{i,j}$ operators to be independent of one another and we address this point later on.

A. Consistency of Commutation Relations

A fundamental property of any valid Clifford transformation is that it preserves commutation relations. That is, the input commutation relations should be consistent with the output commutation relations. So, for all $1 \le i \le n - k$ and $1 \le j \le l_i - 1$, the entries $g_{i,j}$ are m-qubit Pauli operators that are unspecified above, but they should be chosen in such a way that the input–output commutation relations are consistent. That this consistency is possible follows from the fact that the stabilizer generators in (1) form a valid quantum convolutional code according to Definition 1, and it is the content of our first theorem.

Theorem 2 (Consistency of Commutation Relations): Suppose the stabilizer generators in (1) form a valid quantum convolutional code. Then, there exists a set of Pauli operators $g_{i,j}$ for $1 \le i \le n - k$ and $1 \le j \le l_i$ such that the commutation relations on the LHS of (6) are consistent with those on the RHS of (6).

Proof: Let $g_{i,j} \odot g_{k,l}$ be a function that equals one if $g_{i,j}$ and $g_{k,l}$ anticommute and zero if they commute. By inspecting the transformation in (6), several commutation relations should be satisfied. First, for all $i, i' \in \{1, 2, \ldots, n - k\}$ and for all $j' \in \{1, \ldots, l_{i'} - 1\}$

$$g_{i,1} \odot g_{i',j'} = h_{i,1} \odot h_{i',j'}$$

because the first row of each block on the LHS of (6) commutes with all other rows, and for consistency, the RHS of the corresponding rows should commute as well. Next, for all i, $i' \in \{1, 2, \ldots, n-k\}$, $j \in \{1, \ldots, l_i - 2\}$, and $j' \in \{1, \ldots, l_{i'} - 2\}$

$$g_{i,j} \odot g_{i',j'} = (h_{i,j+1} \odot h_{i',j'+1}) + (g_{i,j+1} \odot g_{i',j'+1})$$

because the commutation relations between any of the second to second-to-last rows in the same or different blocks on the LHS of (6) should be consistent with those of the corresponding rows on the RHS. Finally, for all i, $i' \in \{1, 2, \ldots, n-k\}$ and $j \in \{1, \ldots, l_i - 1\}$

$$g_{i,j} \odot g_{i',l_{i'}-1} = h_{i,j+1} \odot h_{i',l_{i'}}$$

because the commutation relations between the last row of each block and any other row on the LHS of (6) should be consistent with those of the corresponding rows on the RHS.

If we start from the first row of any block in (6), a forward commutativity propagation imposes the following equality (without loss of generality (WLOG) suppose $j \geq j'$)

$$g_{i,j} \odot g_{i',j'} = \sum_{k=1}^{\min\{(l_i-j),(l_{i'}-j')\}} h_{i,j+k} \odot h_{i',j'+k} \quad (7)$$

and if we start from the last row of any block in (6), a backward commutativity propagation imposes the following equality:

$$g_{i,j} \odot g_{i',j'} = \sum_{k=0}^{j'-1} h_{i,j-k} \odot h_{i',j'-k}. \quad (8)$$

By adding the RHS of (7) and (8), we obtain the following equality:

$$\sum_{k=1}^{\min\{(l_i-j),(l_{i'}-j')\}} h_{i,j+k} \odot h_{i',j'+k} + \sum_{k=0}^{j'-1} h_{i,j-k} \odot h_{i',j'-k}$$
$$= \sum_{k=1}^{\min\{(l_i-j+j'),l_{i'}\}} h_{i,k+j-j'} \odot h_{i',k}$$
$$= \left(D^{j-j'} h_i \right) \odot h_{i'}$$

where we have introduced the delay operator D from [5] and [6]. Finally, due to the commutativity constraints for the generators of a valid quantum convolutional code, we obtain the following equality:

$$\sum_{k=1}^{\min\{(l_i-j+j'),l_{i'}\}} h_{i,k+j-j'} \odot h_{i',k} = \left(D^{j-j'} h_i \right) \odot h_{i'}$$
$$= 0.$$

Therefore, the RHS of equations in (7) and (8) are the same, and the different constraints imposed by the encoder on the commutation relations of $g_{i,j}$ and $g_{i',j'}$ are consistent. ■

The next section shows how to choose the operators $g_{i,j}$ for the memory qubits such that they are consistent while also acting on a minimal number of memory qubits.

B. Memory Commutativity Matrix

In our running example in (2) and (5), we did not specify how to choose the Pauli operators $g_{i,j}$ acting on the memory qubits. It would be ideal to choose them so that they are consistent with the input–output commutation relations of the transformation in (5), and also so that they act on a minimal number of memory qubits. In this way, we can determine a minimal-memory encoder for the particular stabilizer generators in (2).

As stated earlier, any valid Clifford transformation preserves commutation relations. That is, if two input Pauli operators commute, then the corresponding output Pauli operators should also commute (and similarly, two outputs should anticommute if their corresponding inputs anticommute). So, consider that the first two input rows in (5) commute. Then, the two output rows should commute as well, and in order for this to happen, $g_{1,1}$ and $g_{1,2}$ should commute because $XXXX$ and $XXIX$ commute. For a different case, observe that the first and fifth input rows commute, and for consistency, the first and fifth output rows should commute. Thus, $g_{1,1}$ and $g_{2,1}$ should commute because $XXXX$ and $ZZZZ$ already commute. We can continue in this manner and enumerate all of the commutation relations for the memory operators $g_{i,j}$ simply by ensuring that the input–output commutation relations in (5) are consistent:

$$[g_{1,1}, g_{1,2}] = [g_{1,1}, g_{1,3}] = [g_{1,1}, g_{2,1}]$$
$$= \{g_{1,1}, g_{2,2}\} = \{g_{1,1}, g_{2,3}\} = 0 \quad (9)$$
$$[g_{1,2}, g_{1,3}] = \{g_{1,2}, g_{2,1}\} = \{g_{1,2}, g_{2,2}\}$$
$$= [g_{1,2}, g_{2,3}] = 0 \quad (10)$$
$$\{g_{1,3}, g_{2,1}\} = [g_{1,3}, g_{2,2}] = [g_{1,3}, g_{2,3}] = 0 \quad (11)$$
$$[g_{2,1}, g_{2,2}] = [g_{2,1}, g_{2,3}] = 0 \quad (12)$$
$$[g_{2,2}, g_{2,3}] = 0 \quad (13)$$

where $[A, B] \equiv AB - BA$ is the commutator and $\{A, B\} \equiv AB + BA$ is the anticommutator. In determining some of the later commutation relations, we need to rely on earlier found ones.

Our objective now is to determine the minimal number of memory qubits on which the operators $g_{i,j}$ should act in order for the transformation in (5) to be consistent with the commutation relations in (9)–(13). To this end, it is helpful to write the above commutation relations as entries in a square binary-valued matrix Ω, that we refer to as the "memory commutativity matrix."

Definition 3 (Memory Commutativity Matrix): The memory commutativity matrix Ω corresponding to an encoder of the form in (6) for a set of stabilizer generators has its entries equal to

$$[\Omega]_{(i,j),(k,l)} \equiv g_{i,j} \odot g_{k,l} \quad (14)$$

where we think of the double indices (i, j) and (k, l) as single indices for the matrix elements of Ω, $g_{i,j}$ and $g_{k,l}$ are all of the Pauli operators in (6) acting on the memory qubits, and $g_{i,j} \odot g_{k,l}$ is a function that equals one if $g_{i,j}$ and $g_{k,l}$ anticommute and zero if they commute (implying that Ω is a symmetric matrix).

For our running example in (2), (5), and (9)–(13), the memory commutativity matrix Ω is equal to

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (15)$$

if we take the ordering $g_{1,1}$, $g_{1,2}$, $g_{1,3}$, $g_{2,1}$, $g_{2,2}$, $g_{2,3}$ and consider the commutation relations found in (9)–(13).

The memory commutativity matrix captures commutation relations between Pauli matrices, and our objective is to determine the minimal number of memory qubits on which the memory operators should act in order to be consistent with the above commutation relations. This leads us to our next theorem.

Theorem 4 (Minimal-Memory Encoder): For a given memory commutativity matrix Ω, the minimal number m of memory qubits needed for an encoder is equal to

$$m = \dim(\Omega) - \frac{1}{2}\text{rank}(\Omega).$$

Proof: To prove this theorem, we can exploit ideas from the theory of entanglement-assisted quantum error correction [24], after realizing that finding the minimal number of memory qubits on which the memory operators should act is related to finding the minimal number of ebits required in an entanglement-assisted quantum code. In particular, by the symplectic Gram–Schmidt procedure outlined in [24]–[23], there exists a sequence of full-rank matrices acting by conjugation on the memory commutativity matrix Ω that reduces it to the following standard form:

$$\Omega_0 \equiv \bigoplus_{k=1}^{c}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \oplus \bigoplus_{l=1}^{d}[0] \qquad (16)$$

such that $2c + d = \dim(\Omega)$ for some integers $c, d \geq 0$. Let G denote this sequence of operations. Observe that $\dim(\Omega) = \dim(\Omega_0)$ and $\text{rank}(\Omega) = \text{rank}(\Omega_0)$ because this sequence G of operations is full rank. Furthermore, it holds that $\text{rank}(\Omega_0) = 2c$ because the rank of a direct sum is the sum of the individual matrix ranks. Observe that the Pauli operators $X_1, Z_1, \ldots, X_c, Z_c$ and $Z_{c+1}, \ldots, Z_{c+d}$ acting on $c + d$ qubits have the same commutativity matrix as the standard form given in (16), and furthermore, these operators are minimal, in the sense that there is no set of operators acting on fewer than $c + d$ qubits that could satisfy the commutation relations in (16). We then perform the inverse G^{-1} on the operators $X_1, Z_1, \ldots, X_c, Z_c$ and $Z_{c+1}, \ldots, Z_{c+d}$, producing a set of memory operators $g_{i,j}$ that are consistent with the commutation relations in (14), ensuring that the encoder is valid, while acting on the minimal number of memory qubits possible. The resulting number m of memory qubits is then $m = c + d$, or equivalently

$$m = \dim(\Omega) - \frac{1}{2}\text{rank}(\Omega)$$

because $\dim(\Omega) = 2c + d$ and $\text{rank}(\Omega) = 2c$ [22], [23]. ∎

We can apply the above theorem to our running example in (2) and (5). The rank of the matrix in (15) is full (equal to six), implying that $c = 3$ and the minimal number of memory qubits to encode the generators in (2) is three qubits. Indeed, the standard form of the memory commutativity matrix is

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \oplus \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \oplus \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

A set of Pauli operators with commutation relations corresponding to this standard form is X_1, Z_1, X_2, Z_2, X_3, and Z_3. We can multiply these Pauli operators together to produce the generators $g_{1,1} = XIX$, $g_{1,2} = IIX$, $g_{1,3} = IZI$, $g_{2,1} = ZXZ$, $g_{2,2} = IIZ$, and $g_{2,3} = ZII$ with a commutativity matrix equivalent to that in (15). We can then use these generators as memory operators for the encoder in (5),

producing the following valid minimal-memory convolutional encoder for the stabilizer generators in (2):

Mem.	Anc.	Info.	Phys.	Mem.
III	ZI	II	$XXXX$	XIX
XIX	II	II	$XXIX$	IIX
IIX	II	II	$IXII$	IZI
IZI	II	II $\rightarrow$	$IIXX$	III
III	IZ	II	$ZZZZ$	ZXZ
ZXZ	II	II	$ZZIZ$	IIZ
IIZ	II	II	$IZII$	ZII
ZII	II	II	$IIZZ$	III

(17)

Once we have determined the transformation that the encoder should perform, there is an algorithm for determining an encoder with polynomial depth [28]. There are many encoders which implement the transformation in (17). In order to specify a particular encoder in full, one would need to "complete" the above transformation by determining six additional input–output relations that are independent of the other input–output relations, so that the resulting 14 input–output relations form a basis for the Pauli group acting on seven qubits.

IV. OTHER REPRESENTATIONS OF A CODE

We can find other representations of a quantum convolutional code by multiplying stabilizer generators together or by delaying some of them. In this section, we analyze the impact of these operations on the minimal memory requirements for encoders, and we propose an algorithm to find a minimal-memory encoder among all the representations of a given code.

A. Multiplication of Stabilizers

Suppose we obtain another set of stabilizer generators (say, S') for the same code specified in (1), by multiplying one stabilizer by another. WLOG, suppose that the first stabilizer generator h_1 is multiplied by second stabilizer generator h_2, and suppose that $l_1 > l_2$. As a result, only the rows of transformation corresponding to the second stabilizer generator for S' (the rows in the second block of the transformation) are different from the rows of the transformation corresponding to the original set of stabilizers in (1). In the following, we write the rows of the transformation corresponding to the first and second stabilizer for S':

Mem.	Anc.	Info.	Phys.	Mem.
$I^{\otimes m}$	Z_1	$I^{\otimes k}$	$h_{1,1}$	$g_{1,1}$
$g_{1,1}$	$I^{\otimes n-k}$	$I^{\otimes k}$	$h_{1,2}$	$g_{1,2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
g_{1,l_1-2}	$I^{\otimes n-k}$	$I^{\otimes k}$	h_{1,l_1-1}	g_{1,l_1-1}
g_{1,l_1-1}	$I^{\otimes n-k}$	$I^{\otimes k}$	h_{1,l_1}	$I^{\otimes m}$
$I^{\otimes m}$	Z_2	$I^{\otimes k}$	$h_{2,1} \times h_{1,1}$	$g_{2,1}$
$g_{2,1}$	$I^{\otimes n-k}$	$I^{\otimes k}$ $\rightarrow$	$h_{2,2} \times h_{1,2}$	$g_{2,2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
g_{2,l_2-1}	$I^{\otimes n-k}$	$I^{\otimes k}$	$h_{2,l_2} \times h_{1,l_2}$	g_{2,l_2}
g_{2,l_2}	$I^{\otimes n-k}$	$I^{\otimes k}$	h_{1,l_2+1}	g_{2,l_2+1}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
g_{2,l_1-2}	$I^{\otimes n-k}$	$I^{\otimes k}$	h_{1,l_1-1}	g_{1,l_1-1}
g_{2,l_1-1}	$I^{\otimes n-k}$	$I^{\otimes k}$	h_{1,l_1}	$I^{\otimes m}$

(18)

The RHS of the last row of the first and second block in (18) are the same. So we deduce that the memory states of g_{1,l_1-1} and g_{2,l_1-1} are the same as well. Thus, we can omit the last row of the second block and exchange g_{2,l_1-1} by g_{1,l_1-1} in the transformation. By proceeding in the transformation and omitting repetitive rows, it will turn into the following transformation:

$$
\begin{array}{c|c|c|c|c}
\text{Mem.} & \text{Anc.} & \text{Info.} & \text{Phys.} & \text{Mem.} \\
\hline\hline
I^{\otimes m} & Z_1 & I^{\otimes k} & h_{1,1} & g_{1,1} \\
g_{1,1} & I^{\otimes n-k} & I^{\otimes k} & h_{1,2} & g_{1,2} \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
g_{1,l_2-1} & I^{\otimes n-k} & I^{\otimes k} & h_{1,l_2} & g_{1,l_2} \\
g_{1,l_2} & I^{\otimes n-k} & I^{\otimes k} & h_{1,l_2+1} & g_{1,l_2+1} \\
\vdots & \vdots & \vdots & \to & \vdots & \vdots \\
g_{1,l_1-2} & I^{\otimes n-k} & I^{\otimes k} & h_{1,l_1-1} & g_{1,l_1-1} \\
g_{1,l_1-1} & I^{\otimes n-k} & I^{\otimes k} & h_{1,l_1} & I^{\otimes m} \\
\hline
I^{\otimes m} & Z_2 & I^{\otimes k} & h_{2,1}\times h_{1,1} & g_{2,1} \\
g_{2,1} & I^{\otimes n-k} & I^{\otimes k} & h_{2,2}\times h_{1,2} & g_{2,2} \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
g_{2,l_2-1} & I^{\otimes n-k} & I^{\otimes k} & h_{2,l_2}\times h_{1,l_2} & g_{1,l_2}
\end{array}
\tag{19}
$$

By multiplying the first row of the first block by the first row of the second block, the second row of the first block by the second row of the second block,…, and the l_2^{th} row of the first block by the l_2^{th} row of the second block in (19), we obtain the following transformation:

$$
\begin{array}{c|c|c|c|c}
\text{Mem.} & \text{Anc.} & \text{Info.} & \text{Phys.} & \text{Mem.} \\
\hline\hline
I^{\otimes m} & Z_1 & I^{\otimes k} & h_{1,1} & g_{1,1} \\
g_{1,1} & I^{\otimes n-k} & I^{\otimes k} & h_{1,2} & g_{1,2} \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
g_{1,l_2-1} & I^{\otimes n-k} & I^{\otimes k} & h_{1,l_2} & g_{1,l_2} \\
g_{1,l_2} & I^{\otimes n-k} & I^{\otimes k} & h_{1,l_2+1} & g_{1,l_2+1} \\
\vdots & \vdots & \vdots & \to & \vdots & \vdots \\
g_{1,l_1-2} & I^{\otimes n-k} & I^{\otimes k} & h_{1,l_1-1} & g_{1,l_1-1} \\
g_{1,l_1-1} & I^{\otimes n-k} & I^{\otimes k} & h_{1,l_1} & I^{\otimes m} \\
\hline
I^{\otimes m} & Z_2 & I^{\otimes k} & h_{2,1} & g_{1,1}\times g_{2,1} \\
g_{1,1}\times g_{2,1} & I^{\otimes n-k} & I^{\otimes k} & h_{2,2} & g_{1,2}\times g_{2,2} \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
g_{1,l_2-1}\times g_{2,l_2-1} & I^{\otimes n-k} & I^{\otimes k} & h_{2,l_2} & I^{\otimes m}
\end{array}
\tag{20}
$$

By comparing the second block of the above transformation with the second block of (6), it is clear that if we write the memory commutativity matrix for $\{\{g_{1,i}\}_{\{i=1,\ldots,l_i-1\}}, \{g_{1,i}\cdot g_{2,i}\}_{\{i=1,\ldots,l_i-1\}},\ldots, \{g_{n-k,i}\}_{\{i=1,\ldots,l_i-1\}}\}$ for the new set of stabilizers, the commutativity matrix will be the same as the one for the original set of stabilizers, and so the minimal amount of memory will not change. With a similar approach, we can
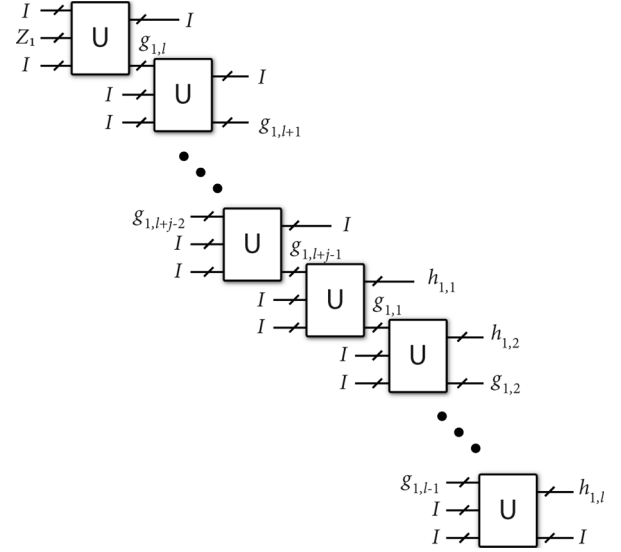

Fig. 2. If the first stabilizer h_1 is delayed by j frames, the encoder should transform the "unencoded" Pauli Z operator acting on the first ancilla qubit to the operator $D^j(h_1)$, as the above figure shows.

show that in the case that $l_1 \leq l_2$, the memory commutativity matrix will not change as well.

B. Delay of Stabilizers

Now suppose we obtain a different representation for the code by delaying one of the stabilizer generators. Suppose WLOG that the first stabilizer is delayed by j frames. Therefore, the encoder should transform the "unencoded" Pauli Z operator acting on the first ancilla qubit to the operator $D^j(h_1)$ (as Fig. 2 shows). Let $g_{1,l_1}, g_{1,l_1+1},\ldots, g_{1,l_1+j-1}$ denote the first j memory operators in the Fig. 2. Let Ω' denote the memory commutativity matrix for the new stabilizer set. Hence, the encoder should perform the transformation in (22). (The first block in (22) differs from the first block in (6) and the others are the same.)

Based on the transformation in (22), we see that all memory states in $\{g_{1,l_1+s}, s \in \{0,1,2,\ldots,j-1\}\}$ commute with all other memory stabilizers. Based on this fact, we see that for the other memory states, the commutativity relations in (7) still hold

$$
g_{i,j}\odot g_{i',j'} = \sum_{k=1}^{\min\{(l_i-j),(l_{i'}-j')\}} h_{i,j+k}\odot h_{i',j'+k}. \tag{21}
$$

Therefore, j rows and j columns corresponding to $\{g_{1,l_1},\ldots, g_{1,l_1+j-1}\}$ in the commutativity matrix (Ω') are all zero, and the other rows and columns are the same as the corresponding rows and columns in Ω (the commutativity matrix for the original set of generators). Hence, the rank of Ω' is the same as the rank of Ω, but its dimension is equal to $j + \dim(\Omega)$. Therefore, it requires j more memory qubits.

In summary, by multiplying stabilizer generators by each other, the amount of memory does not change, but by

delaying one of them by j frames, the required memory increases by j.

Mem.	Anc.	Info.	Phys.	Mem.
$I^{\otimes m}$	Z_1	$I^{\otimes k}$	$I^{\otimes n}$	g_{1,l_1}
g_{1,l_1}	$I^{\otimes n-k}$	$I^{\otimes k}$	$I^{\otimes n}$	g_{1,l_1+1}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
g_{1,l_1+j-2}	$I^{\otimes n-k}$	$I^{\otimes k}$	$I^{\otimes n}$	g_{1,l_1+j-1}
g_{1,l_1+j-1}	$I^{\otimes n-k}$	$I^{\otimes k}$	$h_{1,1}$	$g_{1,1}$
$g_{1,1}$	$I^{\otimes n-k}$	$I^{\otimes k}$	$h_{1,2}$	$g_{1,2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
g_{1,l_1-2}	$I^{\otimes n-k}$	$I^{\otimes k} \rightarrow$	h_{1,l_1-1}	g_{1,l_1-1}
g_{1,l_1-1}	$I^{\otimes n-k}$	$I^{\otimes k}$	h_{1,l_1}	$I^{\otimes m}$
$I^{\otimes m}$	Z_2	$I^{\otimes k}$	$h_{2,1}$	$g_{2,1}$
$g_{2,1}$	$I^{\otimes n-k}$	$I^{\otimes k}$	$h_{2,2}$	$g_{2,2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
g_{2,l_2-2}	$I^{\otimes n-k}$	$I^{\otimes k}$	h_{2,l_2-1}	g_{2,l_2-1}
g_{2,l_2-1}	$I^{\otimes n-k}$	$I^{\otimes k}$	h_{2,l_2}	$I^{\otimes m}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

$$. (22)$$

C. Shortening Algorithm

We can take advantage of the above observations to construct an algorithm that reduces the minimal memory requirements for a given quantum convolutional code. First suppose that the first block of each stabilizer generator which acts on the first n qubits, (i.e., $h_{i,1}$, $i \in \{1, 2, \ldots, n-k\}$) are all independent of each other. If we find another representation for the code by multiplying the stabilizers, as we proved, the amount of minimal memory will be the same, and if we find another representation for the code by delaying some of the stabilizers, the minimal required memory will be more. Now suppose that there is a dependence among the generators $h_{i,1}$. Suppose WLOG that $h_{i,1}$ is equal to $h_{j,1}$, so that by multiplying $h_{i,1}$ with $h_{j,1}$ the first block of jth stabilizer $h_{j,1}$ becomes equal to the identity. Therefore, by shifting it one frame to the left, the amount of minimal memory requirement will be decreased. Therefore, for finding the minimal-memory encoder among all representations of the same code, we should find a representation in which all the first blocks of the stabilizer generators are independent of each other. Also we should remove any dependence among the last blocks of the stabilizer generators as well in order to be confident that the memory states in (6) are independent of each other and our formula is valid. In the next section, we propose an algorithm, that we call the "shortening algorithm," to be confident that there is no dependence among the first blocks of the stabilizer generators and also among the last blocks of the generators. For a given set of stabilizers, first we should apply the shortening algorithm and then write the transformation in (6) for the output stabilizer generators of the algorithm to find the minimal-memory encoder among all representations of a code.

Algorithm 1 is the algorithm for shortening the stabilizers to be confident that we are finding the minimal memory requirements among all stabilizer representations of a given code. There is no dependence among the first blocks and also last blocks of output stabilizers of the algorithm. The function

Subset(S) returns all the subsets of S except for the empty subset. The complexity of the algorithm is exponential in $n - k$.

Algorithm 1 Algorithm for shortening generators

$l_{i,\min} \leftarrow$ minimum degree of h_i
for $i := 1$ to $n - k$ **do**
  $h_i \leftarrow h_i \times D^{-l_{i,min}}$
end for
DepFound $\leftarrow 1$
while DepFound $= 1$ **do**
  DepFound $\leftarrow 0$
  for $i := 1$ to $n - k$ **do**
    $m \leftarrow 1$
    for $j := 1$ to $n - k$ **do**
      if $i \neq j$ **AND** $l_j \leq l_i$ **then**
        $S_m \leftarrow h_{j,1}$
        $m + +$
      end if
    end for
    $\tilde{S} \leftarrow$ Subset(S)
    for $y = 1, y < 2^m - 1$ **do**
      if the product of members of $\tilde{S}_y$ is equal to $h_{i,1}$
      then
        for all $h_{g,1}$ in $\tilde{S}_y$ **do**
          $h_i \leftarrow h_i \times h_g$
        end for
        $h_i \leftarrow D^{-1} h_i$
        $l_i - -$
        DepFound $\leftarrow 1$
      end if
    end for
    CLEAR (S); CLEAR ($\tilde{S}$)
  end for
end while
DepFound $\leftarrow 1$
while DepFound $= 1$ **do**
  DepFound $\leftarrow 0$
  for $i := 1$ to $n - k$ **do**
    $m \leftarrow 1$
    for $j := 1$ to $n - k$ **do**
      if $i \neq j$ **AND** $l_j \leq l_i$ **then**
        $S_m \leftarrow h_{j,l_j}$
        $m + +$
      end if
    end for
    $\tilde{S} \leftarrow$ Subset(S)
    for $y = 1, y < 2^m - 1$ **do**
      if the product of members of $\tilde{S}_y$ is equal to h_{i,l_i}
      then
        for all h_{g,l_g} in $\tilde{S}_y$ **do**
          $h_i = h_i \times D^{l_i - l_g} h_g$
        end for
        $l_i - -$
        DepFound $\leftarrow 1$
      end if
    end for
    CLEAR (S); CLEAR ($\tilde{S}$)
  end for
end while

V. CATASTROPHICITY

Although the convolutional encoder in (17) has a minimal number of memory qubits, it may not necessarily be noncatastrophic (though, we show that it actually is noncatastrophic in Section V-A1). We should ensure that the encoder is noncatastrophic if the receiver decodes the encoded qubits with the inverse of the encoder and then exploits the decoding algorithm in [8] to correct for errors introduced by a noisy channel. As a prerequisite for noncatastrophicity, we need to review the notion of a state diagram for a quantum convolutional encoder.

The state diagram for a quantum convolutional encoder is the most important tool for analyzing properties such as its distance spectrum and for determining whether it is catastrophic [8]. It is similar to the state diagram for a classical encoder [19], [25], [26], with an important exception for the quantum case that incorporates the fact that the logical operators of a quantum code are unique up to multiplication by the stabilizer generators. The state diagram allows us to analyze the flow of the logical operators through the convolutional encoder.

Definition 5 (State Diagram): The state diagram for a quantum convolutional encoder is a directed multigraph with 4^m vertices that we can think of as "memory states," where m is the number of memory qubits in the encoder. Each memory state corresponds to an m-qubit Pauli operator M that acts on the memory qubits. We connect two vertices M and M' with a directed edge from M to M' and label this edge as (L, P) if the encoder takes the m-qubit Pauli operator M, an $(n - k)$-qubit Pauli operator $S^z \in \{I, Z\}^{n-k}$ acting on the $n - k$ ancilla qubits, and a k-qubit Pauli operator L acting on the information qubits, to an n-qubit Pauli operator P acting on the n physical qubits and an m-qubit Pauli operator M' acting on the m memory qubits

$$
\begin{array}{c|c|c|c|c|c}
\text{Mem.} & \text{Anc.} & \text{Info.} & \xrightarrow{\text{encoder}} & \text{Phys.} & \text{Mem.} \\
\hline
M & S^z & L & & P & M'
\end{array} \ .
$$

The labels L and P are the respective logical and physical labels of the edge.

Observe that the state diagram has 4^m vertices and 2^{2m+n+k} edges (there are 4^m memory states, 4^k logical transitions for L, and 2^{n-k} ancilla operators). This is the main reason that it is important to reduce the size of the encoder's memory—it is related to the complexity of the decoding algorithm.

We do not explicitly depict the state diagram for our running example because it would require $4^3 = 64$ vertices and $2^{2(3)+4+2} = 4096$ edges (though note that the entries in (17) and their combinations already give $2^8 = 256$ edges that should be part of the state diagram—we would need the full specification of the encoder for our running example in order to determine its state diagram). A simple example of an encoder that acts on one memory qubit, one ancilla qubit, and one information qubit is depicted in [8, Fig. 8]. Thus, its state diagram has only four vertices and 32 edges, and the encoder's state diagram is depicted in [8, Fig. 9].

We now review the definition of catastrophicity from [8],[2] which is based on the classical notion of catastrophicity from [25] and [26]. The essential idea behind catastrophic error propagation is that an error with finite weight, after being fed through the inverse of the encoder, could propagate infinitely throughout the decoded information qubit stream without triggering syndromes corresponding to these errors. The only way that this catastrophic error propagation can occur is if there is some cycle in the state diagram where all of the edges along the cycle have physical labels equal to the identity operator, while at least one of the edges has a logical label that is not equal to the identity. If such a cycle exists, it implies that the finite-weight channel error produces an infinite-weight information qubit error without triggering syndrome bits corresponding to this error (if it did trigger syndrome bits, this cycle would not be in the state diagram), and an iterative decoding algorithm such as that presented in [8] is not able to detect these errors. So, we can now state the definition of a catastrophic encoder.

Definition 6 (Catastrophic Encoder): A quantum convolutional encoder acting on memory qubits, information qubits, and ancilla qubits is catastrophic if there exists a cycle in its state diagram where all edges in the cycle have zero physical weight, but there is at least one edge in the cycle with nonzero logical weight.[3]

A. Toward a Minimal-Memory/Noncatastrophic Encoder

This section presents our main results that apply to the task of finding a minimal-memory, noncatastrophic encoder for an arbitrary set of stabilizer generators that form a valid quantum convolutional code. Our first theorem states a sufficient condition for a minimal-memory encoder to be noncatastrophic, and this theorem applies to our running example in (2) and (17).

1) Encoders With a Full-Rank Memory Commutativity Matrix:

Theorem 7: Suppose the memory commutativity matrix of a given set of stabilizer generators is full rank. Then, any minimal-memory encoder with a partial specification given by Theorem 4 is noncatastrophic.

---

[2]We should note that there have been previous (flawed) definitions of catastrophicity in the quantum convolutional coding literature. The first appearing in [29] is erroneous by the argument in [30]. Suppose that a convolutional encoder cyclically permutes the qubits in a frame upward so that the first qubit becomes the last, and suppose it then follows with a block encoding on the other qubits. This encoder cannot be arranged into the "pearl-necklace" form required by [29, Proposition 4.1], but nevertheless is obviously noncatastrophic because errors never propagate between logical qubits in different frames.

The definition of noncatastrophicity in [11] is also erroneous. It states that an encoder is noncatastrophic if it can be arranged into a circuit of finite depth. This definition excludes the class of recursive quantum convolutional encoders, which cannot be arranged into a circuit of finite depth. Now, it turns out from a detailed analysis that every recursive quantum convolutional encoder is catastrophic according to the definition in Definition 6 (see [8, Th. 1]), but this theorem does not apply to entanglement-assisted quantum convolutional encoders that can be both recursive and noncatastrophic [17]. Thus, in light of these latter developments, the definition of noncatastrophicity from [11] is flawed.

[3]Interestingly, catastrophicity in the quantum world is not only a property of the encoder, but it also depends on the resources on which the encoder acts [17]. For example, we can replace the ancilla qubit of the catastrophic encoder in [8, Fig. 8] with one system of an entangled bit, and the resulting encoder becomes noncatastrophic. This type of thing can never happen classically if the only kind of resource employed is a classical bit.

Proof: We need to consider an encoder of the general form in (6). Suppose for a contradiction that the minimal-memory encoder with m memory qubits is catastrophic. By Definition 6, this implies that there is some cycle through a set of memory states $\{m_1, \ldots, m_p\}$ of the following form (with zero physical weight but nonzero logical weight):

Mem.	Anc.	Info.	Phys.	Mem.	
m_1	s_1	l_1	$I^{\otimes n}$	m_2	
m_2	s_2	l_2 $\rightarrow$	$I^{\otimes n}$	m_3	(23)
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
m_p	s_p	l_p	$I^{\otimes n}$	m_1	

where $m_1, \ldots, m_p$ are arbitrary Pauli operators acting on the memory qubits, the operators $s_i \in \{I, Z\}^{\otimes(n-k)}$ act on the $n - k$ ancilla qubits, and the operators l_i are arbitrary k-qubit Pauli operators acting on the k information qubits (with at least one of them not equal to the identity operator). Observe that all of the output rows on the RHS of (23) commute with the last row in each of the $n - k$ blocks on the RHS of the transformation in (6). This observation implies that all of the rows on the LHS of (23) should commute with the corresponding rows on the LHS of the transformation in (6). Therefore, all operators $m_1, m_2, \ldots, m_p$ acting on the memory qubits commute with the memory operators g_{i,l_i-1} for all $i \in \{1, 2, \ldots, n - k\}$. Continuing, we now know that all of the rows on the RHS of (23) commute with the second-to-last row in each of the $n - k$ blocks on the RHS of the transformation in (17). This then implies that $m_1, \ldots, m_p$ commute with g_{i,l_i-2} for all $i \in \{1, 2, \ldots, n - k\}$ by the same reasoning above. Continuing in this manner up the rows of each of the $n - k$ blocks, we can show that the operators $m_1, m_2, \ldots, m_p$ commute with all of the memory operators $g_{i,j}$ for all $i \in \{1, 2, \ldots, n - k\}$ and $j \in \{1, 2, \ldots, l_i - 1\}$.

All of these commutativity constraints restrict the form of the operators $m_1, \ldots, m_p$ in the catastrophic cycle. By assumption, the rank of the memory commutativity matrix is full and equal to $2m$. This implies that there are $2m$ memory operators $g_{i,j}$ and they form a complete basis for the Pauli group on m qubits. It follows that each of the operators $m_1, \ldots, m_p$ is equal to the identity operator on m qubits because they are required to commute with all $g_{i,j}$ and the only operator that can do so is the m-qubit identity operator. So all of the entries in (23) are really just cycles of the form

Mem.	Anc.	Info.	Phys.	Mem.	
$I^{\otimes m}$	s_1	l_1	$I^{\otimes n}$	$I^{\otimes m}$	
$I^{\otimes m}$	s_2	l_2 $\rightarrow$	$I^{\otimes n}$	$I^{\otimes m}$	.
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
$I^{\otimes m}$	s_p	l_p	$I^{\otimes n}$	$I^{\otimes m}$	

The above input–output relations restrict $s_1, \ldots, s_p$, and $l_1, \ldots, l_p$ further—it is impossible for $s_1, \ldots, s_p$, and $l_1, \ldots, l_p$ to be any Pauli operator besides the identity operator. Otherwise, the encoder would not transform the entry on the LHS to the all identity operator. Thus, the only cycle of zero-physical weight in a minimal-memory encoder given by Theorem 4 that implements the transformation in (6) is the self-loop at the identity memory state with zero logical weight, which implies the encoder is noncatastrophic. ∎

We return to our running example from (2). We determined in (17) a partial specification of a minimal-memory encoder for these generators, and the above theorem states that any encoder that realizes this transformation is noncatastrophic as well. Indeed, we can study the proof technique above for this example. Suppose for a contradiction that a catastrophic cycle exists in the state diagram for the minimal-memory encoder in (17). Such a catastrophic cycle has the following form:

Mem.	Anc.		Info.		Phys.	Mem.
m_1	$s_{1,1}$	$s_{1,2}$	$l_{1,1}$	$l_{1,2}$	$I^{\otimes 4}$	m_2
m_2	$s_{2,1}$	$s_{2,2}$	$l_{2,1}$	$l_{2,2}$ $\rightarrow$	$I^{\otimes 4}$	m_3
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
m_p	$s_{p,1}$	$s_{p,2}$	$l_{p,1}$	$l_{p,2}$	$I^{\otimes 4}$	m_1

$$(24)$$

where $m_1, \ldots, m_p$ can be arbitrary Pauli operators acting on the three memory qubits, each $s_{i,j} \in \{I, Z\}$ acts on an ancilla qubit, and each $l_{i,j}$ is an arbitrary single-qubit Pauli operator acting on an information qubit (with at least one $l_{i,j}$ not equal to the identity operator). Observe that all of the output rows on the RHS of (24) commute with the fourth and eighth rows on the RHS of the transformation in (17). This observation implies that all of the rows on the LHS of (24) should commute with the fourth and eighth rows on the LHS of the transformation in (17). Therefore, all operators $m_1, m_2, \ldots, m_p$ acting on the memory qubits commute with IZI and ZII. Continuing, we now know that all of the rows on the RHS of (24) commute with the third and seventh rows of (17) because $[m_i, Z_2] = [m_i, Z_1] = 0$ for all $1 \leq i \leq p$. This then implies that $m_1, \ldots, m_p$ commute with IIX and IIZ by the same reasoning above. We can continue one last time to show that all $m_1, \ldots, m_p$ commute with XIX and ZXZ. Similar to the reasoning in the above theorem, all of these commutativity constraints restrict the form of the operators $m_1, \ldots, m_p$ in the catastrophic cycle. In fact, the only three-qubit operator that commutes with IZI, ZII, IIZ, IIZ, XIX, and ZXZ is the three-qubit identity operator because the aforementioned operators form a complete basis for the Pauli group on three qubits. Applying the same logic as at the end of the above proof then allows us to conclude that the encoder is noncatastrophic.

2) Encoders Without a Full-Rank Memory Commutativity Matrix and With an Empty Partial Null Space: Now suppose that the memory commutativity matrix of a given set of stabilizer generators is not full rank. As we explained in the proof of Theorem 7, the memory operators $m_1, \ldots, m_p$ of a catastrophic cycle in (23) commute with all memory operators $g_{i,j}$ in (6). Since the number of commutativity constraints is less than $2m$ in this case (where m is the number of qubits on which the memory operators act), there are other choices for the catastrophic memory operators $m_1, \ldots, m_p$ besides the m-qubit identity operator that are consistent with these constraints. This implies that some of the encoders implementing the transformation in (6) may be catastrophic. To illustrate this case, we choose the second code of [12, Fig. 1] as another running example. This code has the following stabilizer generators:

$h_1 = XXXX$	$XXII$	$IXIX$	$IIXX$	$XXXX$	
$h_2 = ZZZZ$	$ZZII$	$IZIZ$	$IIZZ$	$ZZZZ$	.

An encoding unitary for this code should be as follows:

Mem.	Anc.	Info.		Phys.	Mem.
$I^{\otimes m}$	$Z\ I$	$I\ I$		$X\ X\ X\ X$	$g_{1,1}$
$g_{1,1}$	$I\ I$	$I\ I$		$X\ X\ I\ I$	$g_{1,2}$
$g_{1,2}$	$I\ I$	$I\ I$		$I\ X\ I\ X$	$g_{1,3}$
$g_{1,3}$	$I\ I$	$I\ I$		$I\ I\ X\ X$	$g_{1,4}$
$g_{1,4}$	$I\ I$	$I\ I$	$\rightarrow$	$X\ X\ X\ X$	$I^{\otimes m}$
$I^{\otimes m}$	$I\ Z$	$I\ I$		$Z\ Z\ Z\ Z$	$g_{2,1}$
$g_{2,1}$	$I\ I$	$I\ I$		$Z\ Z\ I\ I$	$g_{2,2}$
$g_{2,2}$	$I\ I$	$I\ I$		$I\ Z\ I\ Z$	$g_{2,3}$
$g_{2,3}$	$I\ I$	$I\ I$		$I\ I\ Z\ Z$	$g_{2,4}$
$g_{2,4}$	$I\ I$	$I\ I$		$Z\ Z\ Z\ Z$	$I^{\otimes m}$

By inspecting the commutativity relations of the memory operators $g_{i,j}$ in the above transformation, the commutativity matrix is

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

with dimension equal to eight and rank equal to four. So, based on Theorem 4, the minimal number of required memory qubits is six. A set of memory operators which act on a minimal number of qubits is as follows:

$$g_{1,1} = ZIIIII, \quad g_{1,2} = IIXIII, \quad g_{1,3} = IIIZII,$$
$$g_{1,4} = IIIIZI, \quad g_{2,1} = IZIIII, \quad g_{2,2} = IIIXII,$$
$$g_{2,3} = IIZIII, \quad g_{2,4} = IIIIIZ.$$

Thus, the encoder implements the following transformation:

Mem.	Anc.	Info.	Phys.	Mem.
$IIIIII$	ZI	II	$XXXX$	$ZIIIII$
$ZIIIII$	II	II	$XXII$	$IIXIII$
$IIXIII$	II	II	$IXIX$	$IIIZII$
$IIIZII$	II	II	$IIXX$	$IIIIZI$
$IIIIZI$	II	II $\rightarrow$	$XXXX$	$IIIIII$
$IIIIII$	IZ	II	$ZZZZ$	$IZIIII$
$IZIIII$	II	II	$ZZII$	$IIIXII$
$IIIXII$	II	II	$IZIZ$	$IIZIII$
$IIZIII$	II	II	$IIZZ$	$IIIIIZ$
$IIIIIZ$	II	II	$ZZZZ$	$IIIIII$

(25)

When the commutativity matrix is not full rank, we should add some rows to the transformation in (6) to ensure that the encoder implementing the transformation is noncatastrophic. To fulfill this requirement, the first step is to find a set C of memory states that can be a part of catastrophic cycle (i.e., memory states which satisfy the commutativity relations mentioned in the proof of Theorem 7). In our running example, the memory operators in a catastrophic cycle should commute with $ZIIIII$, $IIXIII$, $IIIZII$, $IIIIZI$, $IZIIII$, $IIIXII$, $IIZIII$, and $IIIIIZ$. Thus, they must be an operator in the following set:

$$C = \{Z_1^{e_1} Z_2^{e_2} Z_5^{e_3} Z_6^{e_4} : e_1, e_2, e_3, e_4 \in \{0,1\}\}.$$

The next step is to search among the rows and their combinations in (6) to find a set S_1 whose members can potentially be a part of catastrophic cycle. Entries in S_1 have the following form:

Mem.	Anc.	Info.		Phys.	Mem.
M	S^z	L	$\rightarrow$	$I^{\otimes n}$	M'

where M and M' are both elements of the set C, the operator $S^z \in \{I, Z\}^{\otimes(n-k)}$ acts on $n - k$ ancilla qubits, and L is an arbitrary k-qubit Pauli operator acting on the information qubits.

In our running example in (25), members of the set S_1 are obtained by adding the first row to the fifth row, and the sixth row to the tenth row, giving the following input–output relations:

Mem.	Anc.	Info.		Phys.	Mem.
$IIIIZI$	ZI	II	$\rightarrow$	$IIII$	$ZIIIII$
$IIIIIZ$	IZ	II		$IIII$	$IZIIII$

(26)

Consider a set S_2 of rows with physical output equal to the four-qubit identity operator. Now we should add such a set of rows to the transformation so that the output memory operators of the members of S_1 and S_2 make a complete basis for the set C. This guarantees that the rows potentially part of catastrophic cycle are just an entry or a combination of entries of $S_1 \cup S_2$. So if we choose the memory states of the elements of S_2 such that the set $S_1 \cup S_2$ does not create a catastrophic cycle, we can ensure that any encoders performing the transformation with the added rows will be noncatastrophic. In our running example from (25) and (26), we just add two new rows (the rows after the line) as follows:

Mem.	Anc.	Info.	Phys.	Mem.
$IIIIZI$	ZI	II	$IIII$	$ZIIIII$
$IIIIIZ$	IZ	II $\rightarrow$	$IIII$	$IZIIII$
$IIIIII$	II	XI	$IIII$	$IIIIZI$
$IIIIII$	II	IX	$IIII$	$IIIIIZ$

(27)

All combinations of the entries in (27) are as follows:

Mem.	Anc.	Info.	Phys.	Mem.
$IIIIZI$	ZI	II	$IIII$	$ZIIIII$
$IIIIIZ$	IZ	II	$IIII$	$IZIIII$
$IIIIII$	II	XI	$IIII$	$IIIIZI$
$IIIIII$	II	IX	$IIII$	$IIIIIZ$
$IIIIZZ$	ZZ	II	$IIII$	$ZZIIII$
$IIIIZI$	ZI	XI	$IIII$	$ZIIIZI$
$IIIIZI$	ZI	IX $\rightarrow$	$IIII$	$ZIIIIZ$
$IIIIIZ$	IZ	XI	$IIII$	$IZIIZI$
$IIIIIZ$	IZ	IX	$IIII$	$IZIIIZ$
$IIIIII$	II	XX	$IIII$	$IIIIZZ$
$IIIIZZ$	ZZ	XI	$IIII$	$ZZIIZI$
$IIIIZZ$	ZZ	IX	$IIII$	$ZZIIIZ$
$IIIIZI$	ZI	XX	$IIII$	$ZIIIZZ$
$IIIIIZ$	IZ	XX	$IIII$	$IZIIZZ$
$IIIIZZ$	ZZ	XX	$IIII$	$ZZIIZZ$

(28)

By inspecting the rows in (28) it is clear that there is no catastrophic cycle.

Theorem 8 below generalizes the technique from the above example to give a straightforward way for adding rows when S_1 is an empty set.

Theorem 8: Suppose the memory commutativity matrix of a given set of stabilizer generators is not full rank, and suppose further that the set S_1 corresponding to the transformation is an empty set. Then adding rows in the following form to the transformation in (6) ensures that any encoder implementing the transformation is noncatastrophic:

$$\begin{array}{c|c|c|c|c} \underline{Mem.} & \underline{Anc.} & \underline{Info.} & \underline{Phys.} & \underline{Mem.} \\ I^{\otimes m} & I^{\otimes (n-k)} & X_1 & I^{\otimes n} & M_1 \\ \vdots & \vdots & \vdots & \to & \vdots & \vdots \\ I^{\otimes m} & I^{\otimes (n-k)} & X_a & I^{\otimes n} & M_a \end{array} \quad (29)$$

where X_i denotes the Pauli X operator acting on the ith information qubit and the operators $M_1, \ldots, M_a$ form a complete basis for the set C.

Proof: Suppose for a contradiction that the entries in (29) create a catastrophic cycle. Since all input memory operators in (29) are equal to the m-qubit identity operator, the output memory operator of the last row in a catastrophic cycle in (23) should be equal to the identity as well (so that the sequence of memory states forms a cycle). This implies that the last row of the catastrophic cycle is as follows:

$$\begin{array}{c|c|c|c|c} \underline{Mem.} & \underline{Anc.} & \underline{Info.} & \underline{Phys.} & \underline{Mem.} \\ m_p & S^z & L & \to & I^{\otimes n} & I^{\otimes k} \end{array}.$$

So m_p and consequently $m_{p-1}, \ldots, m_1$ are all equal to the m-qubit identity operator. Thus, all of the entries in (23) are really just cycles of the following form:

$$\begin{array}{c|c|c|c|c} \underline{Mem.} & \underline{Anc.} & \underline{Info.} & \underline{Phys.} & \underline{Mem.} \\ I^{\otimes m} & s_1 & l_1 & I^{\otimes n} & I^{\otimes m} \\ I^{\otimes m} & s_2 & l_2 & \to & I^{\otimes n} & I^{\otimes m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ I^{\otimes m} & s_p & l_p & I^{\otimes n} & I^{\otimes m} \end{array}.$$

The above input–output relations imply that $s_1, \ldots, s_p$ and $l_1, \ldots, l_p$ are identity operators (otherwise, it would not be possible to effect the above transformation). Thus, the only cycle of zero-physical weight is the self-loop at the identity memory state with zero logical weight, which implies there is no catastrophic cycle. ∎

VI. Nonrecursiveness

In this section, we demonstrate that the encoders from both Theorems 7 and 8 are nonrecursive. Recursiveness or lack thereof is a fundamental property of a quantum convolutional encoder as demonstrated in [8]. In [8], Poulin *et al.* proved that any noncatastrophic quantum convolutional encoder is already nonrecursive. Note that this situation is much different from classical convolutional encoders for which these two properties are not directly linked. In light of the results of Poulin *et al.*, it follows that our encoders from Theorems 7 and 8 are nonrecursive because they are already noncatastrophic. Nevertheless, we prove below that the encoders are nonrecursive because our proof technique is arguably much simpler than the proof of

Theorem 1 from [8]. Though, before proving these theorems, we briefly review the definition of recursiveness.

Definition 9 (Recursive Encoder): An admissible path is a path in the state diagram for which its first edge is not part of a zero physical-weight cycle. Consider any vertex belonging to a zero physical-weight loop and any admissible path beginning at this vertex that also has logical weight one. The encoder is recursive if all such paths do not contain a zero physical-weight loop.

We can gain some intuition behind the above definition by recalling the definition of a recursive classical convolutional encoder. In the classical case, an encoder is recursive if it has an infinite impulse response—that is, if it outputs an infinite-weight, periodic sequence in response to an input consisting of a single "one" followed by an infinite number of "zeros." Definition 9 above for the quantum case ensures that the response to a single Pauli operator (one of $\{X, Y, Z\}$) at a single logical input along with the identity operator at all other logical inputs leads to a periodic output sequence of Pauli operators with infinite weight. The definition above ensures that this is not only the case for the above sequence but also for one in which the ancilla qubit inputs can be chosen arbitrarily from $\{I, Z\}$. Thus, it is a much more stringent condition for a quantum convolutional encoder to be recursive.

We are now in a position to prove the main theorem of this section.

Theorem 10: The encoders from Theorems 7 and 8 are nonrecursive in addition to being noncatastrophic.

Proof: In order to prove that an encoder is nonrecursive, we just need to find a single logical-weight-one admissible path beginning and ending in the identity memory state. First consider that every memory state in (6) already has a zero-logical-weight path back to the identity memory state. (For example, for the entry $g_{1,1}$ in the second row, one would just need to input $I^{\otimes k}$ and $I^{\otimes n-k}$ for the logical inputs and ancillas, which in turn leads to state $g_{1,2}$. Continuing in this fashion leads to the state g_{1,l_1-1}, which finally leads to the identity memory state.)

Now consider the encoders from Theorem 7 and consider further the following transformation:

$$I^{\otimes m} \otimes I^{\otimes n-k} \otimes X_i \to h \otimes g$$

where h is some arbitrary n-qubit Pauli operator and g is some m-qubit Pauli operator. From the fact that the memory commutativity matrix is full rank, we know that it is possible to construct the memory state g by combining the memory states from (6) (say, for example, $g = g_{i_1,j_1} \cdot g_{i_2,j_2} \cdots g_{i_m,j_m}$). Furthermore, by inputting $I^{\otimes k}$ and $I^{\otimes n-k}$ for all subsequent logical and ancilla inputs, we can construct a path that is a combination of the paths taken by each of $g_{i_1,j_1}, g_{i_2,j_2}, \ldots, g_{i_m,j_m}$. Since all of these paths end up in the identity memory state, it follows that the combination of the paths also ends up in the identity memory state. So there is a logical-weight-one admissible path beginning and ending in the identity memory state. This concludes the proof for encoders from Theorem 7.

The proof for the encoders from Theorem 8 is similar to the above proof. First, let us consider the memory states that are

part of the set C. The rows in (29) added to the transformation are all weight-one logical edges from the identity memory state to a state in C because they have the following form:

$$I^{\otimes m} \otimes I^{\otimes n-k} \otimes X_i \rightarrow I^{\otimes n} \otimes M_i.$$

Since all of the memory states in (6) commute with the elements of C, we can combine some of these commuting states together to realize the memory state M_i. By the same argument as before, inputting $I^{\otimes k}$ and $I^{\otimes n-k}$ for all subsequent logical and ancilla inputs eventually leads back to the identity memory state because all of the individual paths lead back as seen in (6). This concludes the proof for encoders from Theorem 8. ∎

VII. CONCLUSION

We have presented an algorithm to find a minimal-memory, noncatastrophic, polynomial-depth encoder for a given set of stabilizer generators. Our algorithm first determines a transformation that the encoder should perform, without specifying the Pauli operators acting on the memory qubits. It then finds a set of Pauli operators which act on a minimal number of memory qubits and are consistent with the input–output commutation relations of the encoder. The number of minimal memory qubits depends on the dimension and the rank of the "memory commutativity matrix," which details the commutativity relations between the memory operators. Once the memory operators are determined, there is a polynomial-time algorithm to find the encoder which performs the transformation. We have also proved that any minimal-memory encoder with a full-rank memory commutativity matrix is noncatastrophic. However, when the memory commutativity matrix is not full-rank, we should add some rows to the transformation to ensure that the encoder is noncatastrophic. Theorem 8 includes an explicit way of adding rows to transformations that have an empty partial null space. We proved that the encoders from Theorems 7 and 8 are nonrecursive in addition to being noncatastrophic. Finally, the appendix of [27] contains details of our algorithm for many examples of quantum convolutional codes from [6] and [12].

Some open questions still remain. First, we are assuming a particular form for our encoders that they have to take the unencoded Pauli Z operators to the encoded stabilizer operators. Although this form for the encoder is natural, it might be the case that allowing for a different form could lead to encoders with smaller memory requirements. Another open problem is to find an explicit way of adding rows to any transformation without a full-rank memory commutativity matrix in order to ensure that the encoder is noncatastrophic. It is also an open problem to find minimal-memory, noncatastrophic encoders for subsystem convolutional codes [31], [16], entanglement-assisted quantum convolutional codes [13], and convolutional codes that send both classical and quantum information [16].

ACKNOWLEDGMENT

REFERENCES

[1] M. M. Wilde, M. Houshmand, and S. Hosseini-Khayat, "Examples of minimal-memory, non-catastrophic quantum convolutional encoders," presented at the IEEE Int. Symp. Inf. Theory, Saint-Petersburg, Russia, Aug. 2011, arXiv:1011.5535.

[2] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Phys. Rev. A*, vol. 52, no. 4, pp. R2493–R2496, Oct. 1995.

[3] D. Gottesman, "Stabilizer codes and quantum error correction," Ph.D. dissertation, California Inst. Technol., Pasadena, 1997.

[4] A. M. Steane, "Error correcting codes in quantum theory," *Phys. Rev. Lett.*, vol. 77, no. 5, pp. 793–797, Jul. 1996.

[5] H. Ollivier and J.-P. Tillich, "Description of a quantum convolutional code," *Phys. Rev. Lett.*, vol. 91, no. 17, pp. 177902-1–177902-4, Oct. 2003.

[6] G. D. Forney, M. Grassl, and S. Guha, "Convolutional and tail-biting quantum error-correcting codes," *IEEE Trans. Inf. Theory*, vol. 53, no. 3, pp. 865–880, Mar. 2007.

[7] M. Grassl and M. Rötteler, "Quantum convolutional codes: Encoders and structural properties," in *Proc. 44th Annu. Allerton Conf.*, Sep. 2006, pp. 510–519.

[8] D. Poulin, J.-P. Tillich, and H. Ollivier, "Quantum serial turbo-codes," *IEEE Trans. Inf. Theory*, vol. 55, no. 6, pp. 2776–2798, Jun. 2009.

[9] S. A. Aly, M. Grassl, A. Klappenecker, M. Roetteler, and P. K. Sarvepalli, "Quantum convolutional BCH codes," in *Proc. 10th Can. Workshop Inf. Theory*, 2007, pp. 180–183, arXiv:quant-ph/0703113.

[10] S. A. Aly, A. Klappenecker, and P. K. Sarvepalli, "Quantum convolutional codes derived from Reed-Solomon and Reed-Muller codes," in *Proc. Int. Symp. Inf. Theory*, Nice, France, Jun. 2007, pp. 821–825, arXiv:quant-ph/0701037.

[11] M. Grassl and M. Rötteler, "Noncatastrophic encoders and encoder inverses for quantum convolutional codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Seattle, WA, Jul. 2006, pp. 1109–1113, arXiv:quant-ph/0602129.

[12] M. Grassl and M. Rötteler, "Constructions of quantum convolutional codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Nice, France, Jun. 2007, pp. 816–820.

[13] M. M. Wilde and T. A. Brun, "Entanglement-assisted quantum convolutional coding," *Phys. Rev. A*, vol. 81, no. 4, p. 042333, Apr. 2010.

[14] M. M. Wilde and T. A. Brun, "Quantum convolutional coding with shared entanglement: General structure," *Quant. Inf. Process.*, vol. 9, no. 5, pp. 509–540, Oct. 2010.

[15] M. M. Wilde and T. A. Brun, "Extra shared entanglement reduces memory demand in quantum convolutional coding," *Phys. Rev. A*, vol. 79, no. 3, pp. 032313-1–032313-7, Mar. 2009.

[16] M. M. Wilde and T. A. Brun, "Unified quantum convolutional coding," presented at the IEEE Int. Symp. Inf. Theory, Toronto, ON, Canada, Jul. 2008, arXiv:0801.0821.

[17] M. M. Wilde and M.-H. Hsieh, "Entanglement boosts quantum turbo codes," presented at the IEEE Int. Symp. Inf. Theory, Saint-Petersburg, Russia, Aug. 2011, arXiv:1010.1256.

[18] G. D. Forney, "Convolutional codes I: Algebraic structure," *IEEE Trans. Inf. Theory*, vol. IT-16, no. 6, pp. 720–738, Nov. 1970.

[19] R. Johannesson and K. S. Zigangirov, *Fundamentals of Convolutional Coding*. New York: Wiley-IEEE Press, 1999.

[20] M. Houshmand, S. Hosseini-Khayat, and M. M. Wilde, "Minimal memory requirements for pearl necklace encoders of quantum convolutional codes," Nov. 2010 [Online]. Available: arXiv:1004.5179

[21] M. Houshmand and S. Hosseini-Khayat, "Minimal-memory realization of pearl-necklace encoders of general quantum convolutional codes," *Phys. Rev. A*, vol. 83, no. 2, pp. 022308-1–022308-10, Feb. 2011.

[22] M. M. Wilde and T. A. Brun, "Optimal entanglement formulas for entanglement-assisted quantum coding," *Phys. Rev. A*, vol. 77, pp. 064302–064302, 2008.

[23] M. M. Wilde, "Logical operators of quantum codes," *Phys. Rev. A*, vol. 79, no. 6, pp. 062322-1–062322-5, Jun. 2009.

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

[25] A. J. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Trans. Commun. Technol.*, vol. COM-19, no. 5, pp. 751–772, Oct. 1971.

[26] R. J. McEliece, *The Theory of Information and Coding*. Cambridge, U.K.: Cambridge Univ. Press, 2002.

[27] M. Houshmand, S. Hosseini-Khayat, and M. M. Wilde, Minimal-memory, non-catastrophic, polynomial-depth quantum convolutional encoders 2011, arXiv:1105.0649.

[28] S. Bravyi, D. Fattal, and D. Gottesman, "GHZ extraction yield for multipartite stabilizer states," *J. Math. Phys.*, vol. 47, no. 6, pp. 062106-1–062106-19, 2006.

[29] H. Ollivier and J.-P. Tillich, Quantum convolutional codes: Fundamentals 2004, arXiv:quant-ph/0401134.

[30] J. Gütschow, Representation of convolutional stabilizer codes as clifford memory channels Univ. Hannover, Hanover, Germany, Jun. 2010, Tech. Rep..

[31] D. Poulin, "Stabilizer formalism for operator quantum error correction," *Phys. Rev. Lett.*, vol. 95, no. 23, p. 230504, 2005.

Monireh Houshmand was born in Mashhad, Iran. She earned her B.S., M.S., and Ph.D. degrees in Electrical Engineering from Ferdowsi University of Mashhad, in 2005, 2007, and 2011, respectively. She is currently a faculty member of the Electrical Engineering Department at Imam Reza University of Mashhad. Her research interests include quantum error correction and quantum cryptography.

Saied Hosseini-Khayat was born in Tehran, Iran. He received the BS degree in electrical engineering from Shiraz University, Iran (1986), and the M.S. and Ph.D. degrees in electrical engineering from Washington University in St. Louis, USA (1991 and 1997). He held R&D positions at Globespan Semiconductor and Erlang Technologies. His expertise and interests lie in the area of digital system design for communications, networking and cryptography. At present, he is an associate professor of electrical engineering at Ferdowsi University of Mashhad, Iran.

Mark M. Wilde (M'99) was born in Metairie, Louisiana, USA. He received the B.S. degree in computer engineering from Texas A&M University, College Station, Texas, in 2002, the M.S. degree in electrical engineering from Tulane University, New Orleans, Louisiana, in 2004, and the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, California, in 2008.

Currently, he is a Postdoctoral Fellow at the School of Computer Science, McGill University, Montreal, QC, Canada and will start in August 2013 as an Assistant Professor in the Department of Physics and Astronomy and the Center for Computation and Technology at Louisiana State University. He has published over 60 articles and preprints in the area of quantum information processing and is the author of the text *Quantum Information Theory*, to be published by Cambridge University Press. His current research interests are in quantum Shannon theory and quantum error correction.

Dr. Wilde is a member of the American Physical Society and has been a reviewer for the IEEE TRANSACTIONS ON INFORMATION THEORY and the IEEE International Symposium on Information Theory.