Quantum Serial Turbo Codes

David Poulin, Jean-Pierre Tillich, *Member, IEEE*, and Harold Ollivier

***Abstract*—In this paper, we present a theory of quantum serial turbo codes, describe their iterative decoding algorithm, and study their performances numerically on a depolarization channel. Our construction offers several advantages over quantum low-density parity-check (LDPC) codes. First, the Tanner graph used for decoding is free of 4-cycles that deteriorate the performances of iterative decoding. Second, the iterative decoder makes explicit use of the code's degeneracy. Finally, there is complete freedom in the code design in terms of length, rate, memory size, and interleaver choice. We define a quantum analogue of a state diagram that provides an efficient way to verify the properties of a quantum convolutional code, and in particular, its recursiveness and the presence of catastrophic error propagation. We prove that all recursive quantum convolutional encoders have catastrophic error propagation. In our constructions, the convolutional codes have thus been chosen to be noncatastrophic and nonrecursive. While the resulting families of turbo codes have bounded minimum distance, from a pragmatic point of view, the effective minimum distances of the codes that we have simulated are large enough not to degrade the iterative decoding performance up to reasonable word error rates and block sizes. With well-chosen constituent convolutional codes, we observe an important reduction of the word error rate as the code length increases.**

***Index Terms*—Belief propagation, convolutional codes, iterative decoding, quantum error correction, turbo codes.**

I. Introduction

FOR 50 years that followed Shannon's landmark paper [39] on information theory, the primary goal of the field of coding theory was the design of practical coding schemes that could come arbitrarily close to the channel capacity. Random codes were used by Shannon to prove the existence of codes approaching the capacity—in fact, he proved that the overwhelming majority of codes are good in this sense. For symmetric channels, this can even be achieved by linear codes. Unfortunately, decoding a linear code is an NP-hard problem [5], so they have no practical relevance. Making the decoding problem tractable thus requires the use of codes with even more structure.

The first few decades were dominated by algebraic coding theory. Codes such as Reed–Solomon codes [38] and Bose–Chaudhuri–Hocquenghem codes [7], [21] use the algebraic structure of finite fields to design codes with large minimal distances that have efficient minimal distance decoders. The most satisfying compromise today is instead obtained from families of codes (sometimes referred to as "probabilistic codes") with some element of randomness but sufficiently structured to be suitable for iterative decoding. They display good performances for a large class of error models with a decoding algorithm of reasonable complexity. The most prominent families of probabilistic codes are Gallager's low-density parity-check (LDPC) codes [16] and turbo codes [6]. They are all decoded by a belief propagation algorithm which, albeit suboptimal, has been shown to have astonishing performance even at rates very close to the channel capacity. Moreover, the randomness involved in the code design can facilitate the analysis of their average performance. Indeed, probabilistic codes are in many aspects related to quench-disordered physical systems, so standard statistical physics tools can be called into play [29], [46].

Quantum information and quantum error correction [4], [17], [23], [41], [44] are much younger theories and differ from their classical cousins in many aspects. For instance, there exists a quantum analogue of the Shannon channel capacity called the quantum channel capacity [12], [26], [40], which sets the maximum rate at which quantum information can be sent over a noisy quantum channel. Contrarily to the classical case, we do not know how to efficiently compute its value for channels of practical significance, except for quite peculiar channels such as the quantum erasure channel where it is equal to one minus twice the erasure probability [3]. For the depolarizing channel—the quantum generalization of the binary symmetric channel—random codes do not achieve the optimal transmission rate in general. Instead, they provide a lower bound on the channel capacity, often referred to as the hashing bound. In fact, coding schemes have been designed to reliably transmit information on a depolarization channel in a noise regime where the hashing bound is zero [14], [42].

The stabilizer formalism [17] is a powerful method in which a quantum code on n qubits can be seen as classical linear codes on $2n$ bits, but with a parity-check matrix whose rows are orthogonal relative to a symplectic inner product. Moreover, a special class of stabilizer codes, called CSS codes after their inventors [8], [43], can turn any pair of dual classical linear code into a quantum code with related properties. The stabilizer formalism and the CSS construction allow to import a great deal

Manuscript received March 04, 2008; revised February 02, 2009. Current version published May 20, 2009. The work of D. Poulin was supported in part by the Gordon and Betty Moore Foundation through Caltech's Center for the Physics of Information, by the National Science Foundation under Grant PHY-0456720, and by the Natural Sciences and Engineering Research Council of Canada. The material in this paper was presented in part at the IEEE International Symposium on Information Theory, Toronto, ON, Canada, July 2008.

D. Poulin was with the Center for the Physics of Information, California Institute of Technology, Pasadena, CA 91125 USA. He is now with the Département de Physique, Université de Sherbrooke, Sherbrooke, QC J1K 2R1 Canada (e-mail: david.poulin@usherbrooke.ca).

J.-P. Tillich is with the Researcher at the Institut de Recherche en Informatique et Automatique (INRIA), Rocquencourt, Le Chesnay F-78153, France (e-mail: Jean-Pierre.Tillich@inria.fr).

H. Ollivier was with the Perimeter Institute for Theoretical Physics, Waterloo, ON N2J 2W9 Canada. He is now with the Institut Louis Bachelier, 75002 Paris, France and the Fondation du Risque (e-mail: harold.olliver@polytechnique.org).

Communicated by A. Winter, Associate Editor for Quantum Information Theory.

Digital Object Identifier 10.1109/TIT.2009.2018339

of knowledge directly from the classical theory, and one may hope to use them to leverage the power of probabilistic coding to the quantum domain. In particular, one may expect that, as in the classical case, quantum analogues of LDPC codes or turbo codes could perform under iterative decoding as well as random quantum codes, i.e., that they could come arbitrarily close to the hashing bound.

For this purpose, it is also necessary to design a good iterative decoding algorithm for quantum codes. For a special class of noise models considered here—namely, Pauli noise models—it turns out that a version of the classical belief propagation algorithm can be applied. For CSS codes, in particular, each code in the pair of dual codes can be decoded independently as a classical code. However, this is done at the cost of neglecting some correlations between errors that impact the coding scheme's performances. For some class of stabilizer codes, the classical belief propagation can be improved to exploit the coset structure of degenerate errors which improve the code's performances. This is the case for concatenated block codes [35] and the turbo codes we consider here, but we do not know how to exploit this feature for LDPC codes, for instance. Finally, a quantum belief propagation algorithm was recently proposed [25] to enable iterative decoding of more general (non-Pauli) noise models. As in the classical case, quantum belief propagation also ties in with statistical physics [20], [24], [25], [36].

We emphasize that a fast decoding algorithm is crucial in quantum information theory. In the classical setting, when error correction codes are used for communication over a noisy channel, the decoding time translates directly into communication delays. This has been the driving motivation to devise fast decoding schemes, and is likely to be important in the quantum setting as well. However, there is an important additional motivation for efficient decoding in the quantum setting. Quantum computation is likely to require active stabilization. The decoding time thus translates into computation delays, and most importantly, in error suppression delays. If errors accumulate faster than they can be identified, quantum computation may well become infeasible: fast decoding is an essential ingredient to fault-tolerant computation (see, however, [13]).

The first attempts at obtaining quantum analogues of LDPC codes [9], [19], [28] have not yielded results as spectacular as their classical counterpart. This is due to several reasons. First, there are issues with the code design. Due to the orthogonality constraints imposed on the parity-check matrix, it is much harder to construct quantum LDPC codes than classical ones. In particular, constructing the code at random will certainly not do. The CSS construction is of no help since random sparse classical codes do not have sparse duals. In fact, it is still unknown whether there exist families of quantum LDPC codes with nonvanishing rate and unbounded minimum distance. Moreover, all known constructions seem to suffer from a poor minimum distances for reasons which are not always fully understood. Second, there are issues with the decoder. The Tanner graph associated to a quantum LDPC code necessarily contains many 4-cycles, which are well known for their negative effect on the performances of iterative decoding. Moreover, quantum LDPC codes are by definition highly degenerate but their decoder does not exploit this property: rather it is impaired by it [37].

On the other hand, generalizing turbo codes to the quantum setting first requires a quantum analogue of convolutional codes. These have been introduced in [10], [11], [31], and [32] and followed by further investigations [1], [15], [18]. Quantum turbo codes can be obtained from the interleaved serial concatenation of convolutional codes. This idea was first introduced in [33]. There, it was shown that, on memoryless Pauli channels, quantum turbo codes can be decoded similarly to classical serial turbo codes. One of the motivations behind this work was to overcome some of the problems faced by quantum LDPC codes. For instance, graphical representations of serial quantum turbo codes do not necessarily contain 4-cycles. Moreover, there is complete freedom in the code parameters. Both of these points are related to the fact that there are basically no restrictions on the choice of the interleaver used in the concatenation. Another advantage over LDPC codes is that the decoder makes explicit use of the coset structure associated to degenerate errors.

Despite these features, the iterative decoding performance of the turbo code considered in [33] was quite poor, much poorer in fact than the results obtained from quantum LDPC codes. The purpose of this paper is to discuss in length several issues omitted in [33], to provide a detailed description of the decoding algorithm, to suggest much better turbo codes than the one proposed in [33], and, most importantly, to address the issue of catastrophic error propagation for recursive quantum convolutional encoders.

Noncatastrophic and recursive convolutional encoders are responsible for the great success of parallel and serial classical turbo codes. In a serial concatenation scheme, an inner convolutional code that is recursive yields turbo-code families with unbounded minimum distance [22], while noncatastrophic error propagation is necessary for iterative decoding convergence. The last point can be circumvented in several ways (by doping, for instance; see [45]) and some of these tricks can be adapted to the quantum setting, but are beyond the scope of this paper.

The proof [22] that serial turbo codes have unbounded minimal distance carries almost verbatim to the quantum setting. Thus, it is possible to design quantum turbo codes with polynomially large minimal distances. However, we will demonstrate that all recursive quantum convolutional encoders have catastrophic error propagation. This phenomenon is related to the orthogonality constraints which appear in the quantum setting and to the fact that quantum codes are in a sense coset codes. As a consequence, such encoders are not suitable for (standard) serial turbo-codes schemes.

In our constructions, the convolutional codes are, therefore, chosen to be noncatastrophic and nonrecursive, so there is no guarantee that the resulting families of turbo codes have a minimum distance which grows with the number of encoded qubits. Despite these limitations, we provide strong numerical evidence that their error probability decreases as we increase the block size at a fixed rate—and this is up to rather large block sizes. In other words, from a pragmatic point of view, the minimum distances of the codes that we have simulated are large enough not to degrade the iterative decoding performance up to moderate word error rates $(10^{-3} - 10^{-5})$ and block sizes $(10^2 - 10^4)$.

The style of our presentation is motivated by the intention to accommodate a readership familiar with either classical turbo codes or quantum information science. This unavoidably implies some redundancy and the expert reader may want to skip some sections, or perhaps glimpse at them to pick up the notation. In particular, the necessary background from classical coding theory and convolutional codes is presented in Section II using the circuit language of quantum information science. This framework is somewhat unconventional: block codes are defined using reversible matrices rather than parity-check or generating matrices, and convolutional codes are defined via a reversible seed transformation instead of a linear filter built from shift registers and feedback lines; yet it requires little departure from standard presentations. The benefit is a very smooth transition between classical codes and quantum codes, which are the subject of Section III. Whenever possible, the definitions used in the quantum setting directly mirror those established in the classical setting. The other benefit of this framework is that it permits to generate all quantum convolutional codes straightforwardly without being hassled by the orthogonality constraint. In fact, the codes we describe are, in general, not of the CSS class.

Section IV uses the circuit representation to define quantum convolutional codes and their associated state diagram. The state diagram is an important tool to understand the properties of a convolutional code. In particular, the detailed analysis of the state diagram of recursive convolutional encoders performed in Section IV-E will lead to the conclusion that they all have catastrophic error propagation. Section V is a detailed presentation of the iterative decoding procedure used for quantum turbo codes. Finally, our numerical results on the codes' word error rate and spectral properties are presented in Section VI.

II. Classical Preliminaries

The main purpose of this section is to introduce a circuit representation of convolutional encoders which simplifies the generalization of several crucial notions to the quantum setting. For instance, it allows to define in a straightforward way a state diagram for the quantum analogue of a convolutional code which arises naturally from this circuit representation. This state diagram will be particularly helpful for defining and studying fundamental issues related to turbo codes such as recursiveness and noncatastrophicity of the constituent convolutional encoders. The circuit representation is also particularly well suited to present the decoding algorithm of quantum convolutional codes.

A. Linear Block Codes

A classical binary linear code C of dimension k and length n can be specified by a full-rank $(n-k)\times n$ parity-check matrix H over $\mathbb{F}_2$

$$C=\{\overline{c}\mid H\overline{c}^T=0\}. \tag{1}$$

Alternatively, the code can be specified by fixing the encoding of each information word $c\in\mathbb{F}_2^k$ through a linear mapping $c\mapsto\overline{c}=cG$ for some full-rank $k\times n$ generator matrix G over $\mathbb{F}_2$ that satisfies $GH^T=0$. Since G has rank k, there exists an $n\times k$ matrix over $\mathbb{F}_2$ that we denote by a slight abuse of notation by G^{-1} satisfying $GG^{-1}=\mathbb{1}_k$ where for any integer k, $\mathbb{1}_k$ denotes the $k\times k$ identity matrix. Similarly, since H has rank $n-k$, there exists an $n\times(n-k)$ matrix H^{-1} over $\mathbb{F}_2$ satisfying $HH^{-1}=\mathbb{1}_{n-k}$.

Lemma 1: The right inverses H^{-1} and G^{-1} can always be chosen such that $(H^{-1})^TG^{-1}=0$.

Proof: Let $B=(H^{-1})^TG^{-1}$. The substitution $H^{-1}\rightarrow H^{-1}+G^TB^T$ preserves the property $HH^{-1}=\mathbb{1}$ and fulfills the desired requirement. □

We will henceforth assume that the right inverses H^{-1} and G^{-1} are chosen to fulfill the condition of Lemma 1.

To study the analogy between classical linear binary codes and stabilizer codes, we view a rate $\frac{k}{n}$ classical linear code and its encoding in a slightly unconventional fashion. We specify the encoding by an $n\times n$ invertible *encoding* matrix V over $\mathbb{F}_2$. The code space is defined as

$$C=\{\overline{c}=(c:0_{n-k})V\mid c\in\mathbb{F}_2^k\} \tag{2}$$

where we use the following notation.

Notation 1: For an n-tuple $a\in\mathcal{A}^n$ and an m-tuple $b\in\mathcal{A}^m$ over some alphabet $\mathcal{A}$, we denote by $a:b$ the $(n+m)$-tuple formed by the concatenation of a followed by b.

Given the generator matrix G and parity-check matrix H of a code, the encoding matrix V can be fixed to

$$V=\begin{pmatrix}G\\(H^{-1})^T\end{pmatrix}. \tag{3}$$

This matrix is invertible

$$V^{-1}=\left(G^{-1},H^T\right) \tag{4}$$

and satisfies $VV^{-1}=\mathbb{1}_n$ following Lemma 1 . Clearly, the encoding matrix $V:\mathbb{F}_2^n\rightarrow\mathbb{F}_2^2$ specifies both the code space and the encoding. The output $b=aV$ of the encoding matrix V is in the code space if and only if the input is of the form $a=(c:0_{n-k})$, where $c\in\mathbb{F}_2^k$. This follows from the equalities $aV=cG=\overline{c}\in C$ and $(c:s)VH^T=s$.

The encoding matrix also specifies the syndrome associated to each error. When transmitted on a bit-flip channel, a codeword $\overline{c}$ will result in the message $m=\overline{c}+p$ for some $p\in\mathbb{F}_2^n$. The error p can be decomposed into an error syndrome $s\in\mathbb{F}_2^{n-k}$ and a logical error $l\in\mathbb{F}_2^k$ as $pV^{-1}=(l:s)$. This is conveniently represented by the circuit diagram shown at Fig. 1, in which time flows from left to right. In such diagrams, the inverse V^{-1} is obtained by reading the circuit from right to left, running time backwards. This circuit representation is at the core of our construction of quantum turbo codes; it greatly simplifies all definitions and analyses.

A probability distribution $\mathbf{P}(p)$ on the error p incurred during transmission induces a probability distribution on logical transformation and syndromes

$$\mathbf{P}(l,s)=\mathbf{P}(p)\Big|_{p=(l:s)V^{-1}}. \tag{5}$$

We call $\mathbf{P}(l,s)$ the *pullback* of the probability $\mathbf{P}(p)$ through the gate V. Maximum-likelihood decoding $l_{\mathrm{ML}}:\mathbb{F}_2^{n-k}\rightarrow\mathbb{F}_2^k$

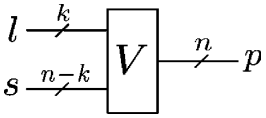

Fig. 1. Circuit representation of encoder $(l : s)V = p$. Slashed wires with integer superscript j indicate a j-bit input/output. The l-bit input are called the logical bits, the other $(n-k)$-bit input are called syndrome or stabilizer bits, and the n-bit output are the physical bits. The string $p \in \mathbb{F}_2^n$ is a codeword if and only if $s = 0_{n-k}$.

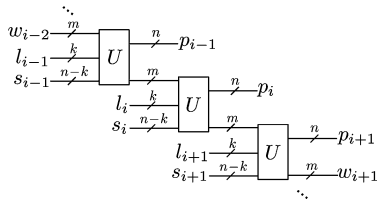

Fig. 2. Circuit diagram of a convolutional encoder with seed transformation U.

consists in identifying the most likely logical transformation l given the syndrome s

$$l_{\mathrm{ML}}(s) = \mathrm{argmax}_{\boldsymbol{l}} \mathbf{P}(l|s) \tag{6}$$

where the conditional probability is defined the usual way

$$\mathbf{P}(l|s) = \frac{\mathbf{P}(l,s)}{\sum_{l'} \mathbf{P}(l',s)}. \tag{7}$$

Similarly, we can define the bitwise maximum-likelihood decoder $l^i_{\mathrm{ML}} : \mathbb{F}_2^{n-k} \to \mathbb{F}_2$, which performs a local optimization on each logical bit

$$l^i_{\mathrm{ML}}(s) = \mathrm{argmax}_{l^i} \mathbf{P}(l^i|s) \tag{8}$$

where the marginal conditional probability is defined the usual way

$$\mathbf{P}(l|s) = \sum_{l^1,\ldots l^{i-1},l^{i+1},\ldots l^k} \mathbf{P}(l^1,\ldots l^k|s). \tag{9}$$

B. Convolutional Codes

We define now a convolutional code as a linear code whose encoder V has the form shown at Fig. 2. The circuit is built from repeated uses of a linear invertible seed transformation $U : \mathbb{F}_2^{n+m} \to \mathbb{F}_2^{n+m}$ shifted by n bits. In this circuit, particular attention must be paid to the order of the inputs as they alternate between syndrome bits and logical bits. The total number of identical repetition is called the duration of the code and is denoted N. The m bits that connect gates from consecutive "time slices" are called memory bits. The encoding is initialized by setting the first m memory bits to $w_0 = 0_m$. There are several ways to terminate the encoding, but we here focus on a padding technique. This simply consists in setting the k logical bits of the last t time slices $i = N+1, N+2, \ldots N+t$ equal to $l_i = 0_k$, where t is a free parameter independent of N. The rate of the code is thus $k/n + O(1/N)$.

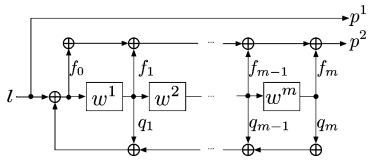

Fig. 3. Representation of convolutional encoder as a linear filter. The labels f and q take values 0 and 1 and indicate, respectively, the absence or presence of the associated wire. Although linear, this transformation is not invertible.

Note that in this diagram, we use a *subscript* to denote the different elements of a stream. For instance, p_i denotes the n-bit output string at time i. The jth bits of p_i would be denoted by a *subscript* as p_i^j, or simply p^j when the particular time i is clear from context. This convention will be used throughout this paper.

This definition of convolutional code differs at first sight from the usual one based on linear filters built from shift register and feedback lines. An example of a linear filter for a rate $\frac{1}{2}$ (systematic and recursive) convolutional encoder is shown in Fig. 3. Another common description of this encoder would be in terms of its rational transfer function which related the D-transform of the output $p(D)$ to that of the input $l(D)$. Remember that the D-transform of a bit stream $x_1 : x_2 : x_3 : \cdots$ is given by $x(D) = \sum_i x_i D^i$. For the code of Fig. 3, the output's D-transforms are

$$p^1(D) = l(D) \tag{10}$$

$$p^2(D) = \frac{f_0 + f_1 D + \cdots + f_m D^m}{1 + q_1 D + \cdots + q_m D^m} l(D) \tag{11}$$

where the inverse is the Laurent series defined by long division. The code can also be specified by the recursion relation

$$\begin{aligned} w_i^j &= w_{i-1}^{j-1}, \qquad \text{for } j > 1 \\ w_i^1 &= l_i + \sum_{j=1}^m q_j w_{i-1}^j \\ p_i^2 &= f_0 \left(\sum_{j=1}^m q_j w_{i-1}^j + l_i \right) + \sum_{j=1}^m f_j w_{i-1}^j \\ &= f_0 l_i + \sum_{j=1}^m (f_j + f_0 q_j) w_{i-1}^j. \end{aligned}$$

These definitions are in fact equivalent to the circuit of Fig. 2 with the seed transformation U specified by Fig. 4. Note that we can assume without lost of generality that $f_m = 1$ or $q_m = 1$ (or both), and these two cases lead to different seed transformations. The generalization to arbitrary linear filters is straightforward. In terms of matrices, the seed transformation associated to this

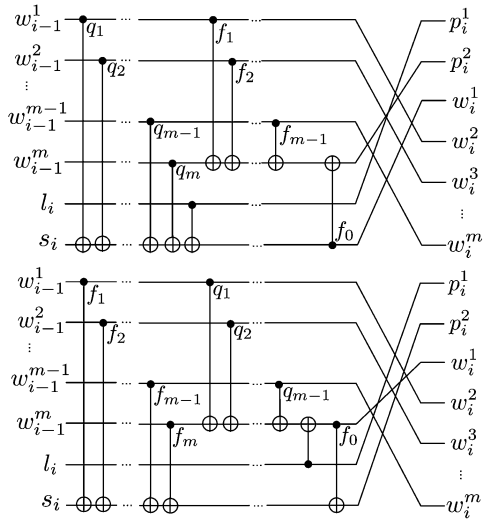

Fig. 4. Seed transformation circuit for convolutional code of Fig. 3. Top: Case $f_m = 1$. Bottom: Case $q_m = 1$. The labels f and q take values 0 and 1 and indicate, respectively, the absence or presence of the associated gate. Both circuits are entirely built from controlled nots, and are, therefore, invertible. As its name indicates, the controlled not acts by negating the target bit $\oplus$ if and only if the control bit $\bullet$ is in state 1.

convolutional code encodes the relation $(p_i : w_i) = (w_{i-1} : l_i : s_i)U$ with U given by

$$U = \begin{pmatrix} \overbrace{\mu_{\mathrm{P}}}^{n} & \overbrace{\mu_{\mathrm{M}}}^{m} \\ \Lambda_{\mathrm{P}} & \Lambda_{\mathrm{M}} \\ \Sigma_{\mathrm{P}} & \Sigma_{\mathrm{M}} \end{pmatrix} \begin{matrix} \}m \\ \}k \\ \}n-k \end{matrix} \tag{12}$$

where

$$\mu_{\mathrm{P}} = \begin{pmatrix} 0 & f_1 + f_0 q_1 \\ \vdots & \vdots \\ 0 & f_m + f_0 q_m \end{pmatrix} \quad \mu_{\mathrm{M}} = \begin{pmatrix} q_1 & & \\ q_2 & & \mathbb{1}_{m-1} \\ \vdots & & \\ q_m & 0 \quad 0 & 0 \end{pmatrix}$$

$\Lambda_{\mathrm{P}} = (1, f_0)$, and $\Lambda_{\mathrm{M}} = (1\ 0_{m-1})$. The two other components depend on whether $f_m = 1$ or $q_m = 1$. In the former case, $\Sigma_{\mathrm{P}} = (0, f_0)$ and $\Sigma_{\mathrm{M}} = (1\ 0_{m-1})$, while in the latter case, $\Sigma_{\mathrm{P}} = (0, 1)$ and $\Sigma_{\mathrm{M}} = (0_m)$.

Not only does the circuit of Fig. 2 produce the same encoding as the linear filter of Fig. 3, but also it has the same memory states. More precisely, the value contained in the jth shift register at time i in Fig. 3 is equal to the value of the jth memory bit between gate i and $i+1$ on Fig. 2. This is important because it allows to define the state diagram (see Section IV-B) directly from the circuit diagram Fig. 4.

Of particular interest are systematic recursive encoders that are defined as follows.

Definition 1 (Systematic Encoder): An encoder is *systematic* when the input stream is a substream of the output stream.

Definition 2 (Recursive Encoder): A convolutional encoder is *recursive* when its rational transfer function involves genuine Laurent series (as opposed to simple polynomials).

Systematic encoders copy the input stream in clear in one of the output stream. Typically, they have transfer functions of the form $p^j(D) = l^j(D)$ for $j = 1, \ldots, k$ and arbitrary $p^j(D)$ for $j > k$, so p_i^j is a copy of l_i^j. The systematic character of the code considered in the above example is most easily seen from Fig. 3: p^1 is a copy of the input l. Systematic encoders are used to avoid catastrophic error propagation. This term will be defined formally in the quantum setting, but it essentially means that an error affecting a finite number of physical bits is mapped to a logical transformation on an infinite number of logical bits by the encoder inverse. Catastrophic encoders cannot be used directly in standard turbo-code schemes. The problem is that the first iteration of iterative decoding does not provide information on the logical bits. This is due to the fact that as the length of the convolutional encoder tends to infinity and in the absence of prior information about the value of the logical bits, the logical bit error rate after decoding tends to $\frac{1}{2}$.

A recursive encoder has an infinite impulsive response: on input l of Hamming weight 1, it creates an output of infinite weight for a code of infinite duration N. Recursiveness is also related to the presence of feedback in the encoding circuit, which is easily understood from the linear filter of Fig. 3. Except when the polynomial $\sum q_i D^i$ factors $\sum f_i D^i$, an encoder with feedback will be recursive. It is essential to use as constituent recursive convolutional codes in classical turbo-code schemes to obtain families of turbo codes of unbounded minimum distance and with performances which improve with the block size.

III. Quantum Mechanics and Quantum Codes

In this section, we review some basic notions of quantum mechanics, the stabilizer formalism, and the decoding problem for quantum codes. In Section III-B, stabilizer codes are defined the usual way, as subspaces of the Hilbert space stabilized by an Abelian subgroup of the Pauli group. We detail in Section III-C how these codes are decoded. Even if a stabilizer code is a continuous space, it can be defined and studied by using only discrete objects (parity-check matrix, encoding matrix, syndrome) which are quite close to classical linear codes. We discuss in Section III-D the relations between such quantum codes and classical linear codes but also highlight the crucial distinctions between them. Particular emphasis is put on the role of the encoder because it is a crucial ingredient for our definition of quantum turbo codes. The encoder also provides an intuitive picture for the logical cosets, which are an important distinction between classical codes and quantum stabilizer codes.

A. Qubits and the Pauli Group

A *qubit* is a physical system whose state is described by a unit-length vector in a two-dimensional Hilbert space. The two vectors of a given orthonormal basis are conventionally denoted by $|0\rangle$ and $|1\rangle$. We identify the Hilbert space with $\mathbb{C}^2$ in the usual way with the help of such a basis. The state of a system comprising n qubits is an unit-length vector in the tensor product of n two-dimensional Hilbert spaces. It is a space of dimension 2^n which can be identified with $(\mathbb{C}^2)^{\otimes n} \simeq \mathbb{C}^{2^n}$. It has a basis given by all tensor products of the form $|x_1\rangle \otimes \cdots \otimes |x_n\rangle$, where the $x_i \in \{0, 1\}$ and the inner product between two basis elements

$|x_1\rangle \otimes \cdots \otimes |x_n\rangle$ and $|y_1\rangle \otimes \cdots \otimes |y_n\rangle$ is the product of the inner products of $|x_i\rangle$ with the corresponding $|y_i\rangle$. In other words, this basis is orthonormal. It will be convenient to use the following notation.

Notation 2:

$$|0_n\rangle \triangleq \underbrace{|0\rangle \otimes \cdots \otimes |0\rangle}_{n \text{ times}}.$$

The error model we consider in this paper is a Pauli-memoryless channel which is defined with the help of the three Pauli matrices

$$\mathcal{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \mathcal{Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad \mathcal{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

These matrices anticommute with each other and satisfy the following multiplication table:

$\times$	$\mathcal{X}$	$\mathcal{Y}$	$\mathcal{Z}$
$\mathcal{X}$	$\mathcal{I}$	$i\mathcal{Z}$	$-i\mathcal{Y}$
$\mathcal{Y}$	$-i\mathcal{Z}$	$\mathcal{I}$	$i\mathcal{X}$
$\mathcal{Z}$	$i\mathcal{Y}$	$-i\mathcal{X}$	$\mathcal{I}$

where $\mathcal{I}$ denotes the 2×2 identity matrix. The action of these operators on the state of a qubit is obtained by right multiplication $|\psi\rangle \rightarrow \mathcal{P}|\psi\rangle$, with $|\psi\rangle$ viewed as an element of $\mathbb{C}^2$.

These matrices generate the Pauli group $\mathcal{G}_1$ which is readily seen to be the set

$$\{\pm\mathcal{I}, \pm i\mathcal{I}, \pm\mathcal{X}, \pm i\mathcal{X}, \pm\mathcal{Y}, \pm i\mathcal{Y}, \pm\mathcal{Z}, \pm i\mathcal{Z}\}.$$

They also form all the errors which may affect one qubit in our error model. If we have an n-qubit system, then the errors which may affect it belong to the Pauli group $\mathcal{G}_n$ over n qubits which is defined by

$$\begin{aligned}\mathcal{G}_n &= \mathcal{G}_1^{\otimes n} \\ &= \{\epsilon \mathcal{P}_1 \otimes \cdots \otimes \mathcal{P}_n | \epsilon \in \{\pm 1, \pm i\}, \mathcal{P}_i \in \{\mathcal{I}, \mathcal{X}, \mathcal{Y}, \mathcal{Z}\}\}.\end{aligned}$$

This group is generated by i and the set of $\mathcal{X}_i$'s and $\mathcal{Z}_i$'s for $i = 1, 2, \ldots, n$, which are defined as follows.

Notation 3:

$$\mathcal{X}_i \triangleq \overbrace{\mathcal{I} \otimes \cdots \otimes \mathcal{I}}^{i-1 \text{ times}} \otimes \mathcal{X} \otimes \overbrace{\mathcal{I} \otimes \cdots \otimes \mathcal{I}}^{n-i \text{ times}}$$
$$\mathcal{Z}_i \triangleq \overbrace{\mathcal{I} \otimes \cdots \otimes \mathcal{I}}^{i-1 \text{ times}} \otimes \mathcal{Z} \otimes \overbrace{\mathcal{I} \otimes \cdots \otimes \mathcal{I}}^{n-i \text{ times}}.$$

In quantum mechanics, two states are physically indistinguishable if they differ by a multiplicative constant. This motivates the definition of another group of errors, called the effective Pauli group, obtained by taking the quotient of $\mathcal{G}_n$ by $\{\pm\mathcal{I}, \pm i\mathcal{I}\}$.

Definition 3 (Effective Pauli Group): The effective Pauli group G_n on n qubits is the set of equivalence classes $[\mathcal{P}]$ for $\mathcal{P}$ in $\mathcal{G}_n$, where the equivalence class $[\mathcal{P}]$ is the set of elements of $\mathcal{G}_n$ which differ from $\mathcal{P}$ by a multiplicative constant. We will also use the notation $I \triangleq [\mathcal{I}]$, $X \triangleq [\mathcal{X}]$, $Y \triangleq [\mathcal{Y}]$, $Z \triangleq [\mathcal{Z}]$, and $X_i = [\mathcal{X}_i]$, $Z_i = [\mathcal{Z}_i]$.

All the effective Pauli groups G_n are Abelian. $(G_1, +)$ is isomorphic to $(\mathbb{F}_2 \times \mathbb{F}_2, +)$ where the group operation of G_1 corresponds to bitwise addition over $\mathbb{F}_2 \times \mathbb{F}_2$. As a consequence, effective Pauli operators can be represented by binary couples. We will henceforth make use of the following representation:

$$I \leftrightarrow (0,0) \tag{13}$$
$$X \leftrightarrow (1,0) \tag{14}$$
$$Y \leftrightarrow (1,1) \tag{15}$$
$$Z \leftrightarrow (0,1). \tag{16}$$

Note that $G_n \cong G_1^n$ and we will either view, depending on the context, an element $P \in G_n$ as an n-tuple $(P^i)_{i=1}^n$ with entries in G_1 or as $2n$-tuple with entries in $\mathbb{F}_2$ obtained by replacing each P_i by its corresponding binary representation. G_n is generated by the X_i and Z_i, and we introduce the following notation.

Notation 4: For P in G_n, we denote by P^x and P^z the only elements of G_n satisfying:

1) $P = P^x + P^z$;
2) $P^x \in \{I, X\}^n$, $P^z \in \{I, Z\}^n$.

An important property of $\mathcal{G}_n$ is that any pair of elements $\mathcal{P}, \mathcal{Q}$ either commutes or anticommutes. This leads to the definition of an inner product "$\star$" for elements $P = (P_i)_{1 \leq i \leq n}$ and $Q = (Q_i)_{1 \leq i \leq n}$ of G_n such that $P \star Q = \sum_{i=1}^n P_i \star Q_i \bmod 2$. Here, $P_i \star Q_i = 1$ if $P_i \neq Q_i$, $P_i \neq I$, and $Q_i \neq I$; and $P_i \star Q_i = 0$, otherwise.

Fact 1: $\mathcal{P}, \mathcal{Q} \in \mathcal{G}_n$: commute if and only if $[\mathcal{P}] \star [\mathcal{Q}] = 0$.

This product can also be defined with the help of the following matrix which will appear again later in the definition of symplectic matrices.

Notation 5:

$$\Lambda_n \triangleq \mathbb{1}_n \otimes \mathcal{X}.$$

By viewing now elements of G_n as binary $2n$ -tuples we have the following.

Definition 4 (Inner Product): Define the inner product $\star : G_n \times G_n \rightarrow \mathbb{F}_2$ by $P \star Q = P \Lambda_n Q^T$.

G_n is an $\mathbb{F}_2$-vector space and we use the $\star$ inner product to define the orthogonal space of a subspace of G_n as follows.

Definition 5 (Orthogonal Subspace): Let V be a subset of G_n. We define $V^\perp$ by

$$V^\perp \triangleq \{P \in G_n : P \star Q = 0, \text{ for every } Q \in V\}.$$

$V^\perp$ is always a subspace of G_n and if the space spanned by V is of dimension t, then $V^\perp$ is of dimension $2n - t$.

From the fact that two states are indistinguishable if they differ by a multiplicative constant, a Pauli error may only be specified by its effective Pauli group equivalence to which it belongs. A very important quantum error model is the *depolarizing*

channel. It is in a sense the quantum analogue of the binary symmetric channel.

Definition 6 (Depolarizing Channel): The depolarizing channel on n qubits of error probability p is an error model where all the errors which occur belong to G_n and the probability that a particular element P is chosen is equal to $(1-p)^{n-\mathrm{w}(P)}\left(\frac{p}{3}\right)^{\mathrm{w}(P)}$ where the weight $\mathrm{w}(P)$ of a Pauli error is given by the following.

Notation 6: $\mathrm{w}(P)$ is the number of coordinates of P which differ from I.

In other words, the coordinates of the error are chosen independently: there is no error on a given coordinate with probability $1-p$ and there is an error on it of type X, Y, or Z, each with probability $\frac{p}{3}$.

B. Stabilizer Codes: Hilbert Space Perspective

A quantum error correction code protecting a system of k qubits by embedding them in a larger system of n qubits is a 2^k-dimensional subspace $\mathcal{C}$ of $(\mathbb{C}^2)^{\otimes n}$. We say that it is a quantum code of *length* n and *rate* $\frac{k}{n}$. It can be specified by a unitary transformation $\mathcal{V}:\mathbb{C}^{2^n}\rightarrow\mathbb{C}^{2^n}$

$$\mathcal{C}=\left\{|\overline{\psi}\rangle=\mathcal{V}(|\psi\rangle\otimes|0_{n-k}\rangle)\mid|\psi\rangle\in\mathbb{C}^{2^k}\right\}. \tag{17}$$

This definition directly reflects (2). As in the classical case, the matrix $\mathcal{V}$ specifies not only the code but also the encoding, that is, the particular embedding $(\mathbb{C}^2)^{\otimes k}\rightarrow(\mathbb{C}^2)^{\otimes n}$. An importance distinction, however, is that in the quantum case, the dimension of the matrix $\mathcal{V}$ is exponential in the number of qubits n. To obtain an efficiently specifiable code, we choose $\mathcal{V}$ from a subgroup of the unitary group over $(\mathbb{C}^2)^{\otimes n}$ called the Clifford group. In fact, not only are Clifford transformations over n qubits efficiently specifiable, but also they can be implemented efficiently by a quantum circuit involving only $O(n^2)$ elementary quantum gates on 1 and 2 qubits (see [30, Th. 10.6], for instance).

Definition 7 (Clifford Transformation and Clifford Group): A *Clifford transformation* over n qubits is a unitary transform $\mathcal{V}$ over $(\mathbb{C}^2)^{\otimes n}$, which leaves the Pauli group over n qubits globally invariant by conjugation

$$\mathcal{V}\mathcal{G}_n\mathcal{V}^\dagger=\mathcal{G}_n.$$

The set of Clifford transformations is a group and is called the *Clifford group* over n qubits.

This definition naturally leads to the action of the Clifford group on elements of the Pauli group.

Definition 8 (Action of Clifford Transformation on Pauli): A Clifford transformation $\mathcal{V}$ acts on the Pauli group as

$$\begin{aligned}\mathcal{G}_n&\rightarrow\mathcal{G}_n\\ \mathcal{P}&\mapsto\mathcal{P}'=\mathcal{V}\mathcal{P}\mathcal{V}^\dagger.\end{aligned}$$

It also acts on the effective Pauli group by the mapping $[\mathcal{P}]\mapsto[\mathcal{P}']$.

The last mapping is $\mathbb{F}_2$-linear and there is a square binary matrix V of size $2n$ which is such that

$$[\mathcal{V}\mathcal{P}\mathcal{V}^\dagger]=[\mathcal{P}]V.$$

This matrix will be called the *encoding matrix*.

Definition 9 (Encoding Matrix): The encoding matrix V associated to an encoding operation $\mathcal{V}$, which is a Clifford transformation over n qubits, is the binary matrix V of size $2n\times 2n$ such that for any $\mathcal{P}\in\mathcal{G}_n$, we have

$$[\mathcal{V}\mathcal{P}\mathcal{V}^\dagger]=[\mathcal{P}]V.$$

Clearly then, a Clifford transformation on n qubits can be specified by its associated encoding matrix V on $\mathbb{F}_2^{2n}$ together with a collection of $2n$ phases. This shows that Clifford transformations are efficiently specifiable as claimed. It can readily be verified that the rows of V, denoted V_i $i=1,2,\ldots,2n$, are equal to

$$V_{2i-1}=[\mathcal{V}\mathcal{X}_i\mathcal{V}^\dagger]=X_iV \tag{18}$$

$$V_{2i}=[\mathcal{V}\mathcal{Z}_i\mathcal{V}^\dagger]=Z_iV. \tag{19}$$

Since conjugation by a unitary matrix $\mathcal{V}$ does not change the commutation relations, the above equations imply that the encoding matrix is a symplectic matrix, whose definition is recalled below.

Definition 10 (Symplectic Transformation): A n-qubit *symplectic transformation* is a $2n\times 2n$ matrix U over $\mathbb{F}_2$ that satisfies

$$U\Lambda_nU^T=\Lambda_n.$$

By definition, symplectic transformations are invertible and preserve the inner product $\star$ between n-qubit Pauli group elements. Conversely, all symplectic matrices always correspond to a (nonunique) Clifford transformation. A stabilizer code is thus a quantum code specified by (17), but with $\mathcal{V}$ in the Clifford group. The code $\mathcal{C}$ (but not the encoding) can equivalently be specified with $(n-k)$-independent mutually commuting elements of $\mathcal{G}_n$ of order 2 as follows.

Definition 11 (Stabilizer Code): The stabilizer code $\mathcal{C}$ associated to the stabilizer set $\{\mathcal{H}_i, i=1\ldots n-k\}$, where the $\mathcal{H}_i$'s are independent mutually commuting elements of $\mathcal{G}_n$ of order 2 and different from -1, is the subspace of $(\mathbb{C}^2)^{\otimes n}$ of elements stabilized by the $\mathcal{H}_i$'s, that is

$$\mathcal{C}=\left\{|\overline{\psi}\rangle\mid\mathcal{H}_i|\overline{\psi}\rangle=|\overline{\psi}\rangle,1\leq i\leq n-k\right\}. \tag{20}$$

This is the usual definition of stabilizer codes. The $\mathcal{H}_i$ play a role analogous to the rows of the parity-check matrix of a classical linear code, and this connection will be formalized in Section III-D. To see the equivalence between this definition and (17), set $\mathcal{H}_i=\mathcal{V}Z_{k+i}\mathcal{V}^\dagger$. These operators are independent and of order 2 since they are conjugate to the $\mathcal{Z}_i$, which are

independent and of order 2. Now, consider a $|\overline{\psi}\rangle \in \mathcal{C}$ as defined in (17). For all $\mathcal{H}_i$, we have

$$\mathcal{H}_i|\overline{\psi}\rangle = \mathcal{V}\mathcal{Z}_{i+k}\mathcal{V}^\dagger\mathcal{V}(|\psi\rangle \otimes |0_{n-k}\rangle) \quad (21)$$
$$= \mathcal{V}(|\psi\rangle \otimes \mathcal{Z}_i|0_{n-k}\rangle) = |\overline{\psi}\rangle \quad (22)$$

where we used the fact that $\mathcal{Z}|0\rangle = |0\rangle$. Hence, $|\overline{\psi}\rangle$ satisfies the condition of Definition 11. Conversely, for any state $|\overline{\psi}\rangle \in \mathcal{C}$ according to Definition 11, we have

$$\mathcal{Z}_{k+i}\mathcal{V}^\dagger|\overline{\psi}\rangle = \mathcal{V}^\dagger\mathcal{H}_i|\overline{\psi}\rangle \quad (23)$$
$$= \mathcal{V}^\dagger|\overline{\psi}\rangle \quad (24)$$

which implies that the $(k+i)$th qubit of $\mathcal{V}^\dagger|\overline{\psi}\rangle$ must be in state $|0\rangle$. Since this holds for all $i = 1, 2, \ldots, n-k$, we conclude that the two definitions are equivalent. This equivalence has the following consequence.

Fact 2: A stabilizer code of length n associated to $(n-k)$-independent generators $\mathcal{H}_i$ is of dimension 2^k.

Since $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ are all of order 2, all the generators of order 2 in $\mathcal{G}_n$ are of the form $\pm\mathcal{P}$ where $\mathcal{P}$ is a tensor product of n matrices all chosen among the set $\{\mathcal{I}, \mathcal{X}, \mathcal{Y}, \mathcal{Z}\}$. Thus, we can specify the generators $\mathcal{H}_i$ of the stabilizer code by giving only the associated effective Pauli group elements together with a sign for each generator. Changing the sign of a stabilizer generator changes the code, but not its properties.[1] More precisely, the set of Pauli errors which can be corrected by such a code does not depend on the signs which have been chosen. Hence, we can specify a family of "equivalent" codes by specifying instead of the $\mathcal{H}_j$'s the set of $H_j \triangleq [\mathcal{H}_j] = VZ_{j+k}$. It is important to note that these elements have to be orthogonal: the fact that the $\mathcal{H}_i$'s commute translates into the orthogonality condition $H_i \star H_j = 0$. Thus, the H_i span a linear space called the stabilizer space, which we denote $C(I)$ for reasons that will become apparent later.

Thus, in analogy with classical linear codes, a stabilizer code (or more precisely an equivalent class thereof) can be efficiently specified by an encoding matrix V on $\mathbb{F}_2^{2n}$. This matrix also provides an efficient description of the encoding up to a set of phases. There is another analogy with a classical encoding matrix that will be crucial for our definition of quantum turbo codes. Assume that we concatenate two stabilizer codes and that these codes are encoded by Clifford transformations. The result of the concatenation is also a stabilizer code (because Clifford transformations form a group) and the resulting encoding matrix is just the product of the two encoding matrices of each constituent code. This reflects the fact that the encoding matrices provide a representation of the Clifford group.

Fact 3: Let $\mathcal{V}_1$ and $\mathcal{V}_2$ be two Clifford transformations over n qubits with encoding matrices V_1 and V_2, respectively. Then, $\mathcal{V}_2\mathcal{V}_1$ is a Clifford transformation with encoding matrix V_1V_2.

[1]This is strictly true for Pauli channels which are considered here. For a general noise model, error correcting properties may actually depend on the sign of the stabilizer generators.

Proof: Consider the Clifford transformation $\mathcal{V} \triangleq \mathcal{V}_2\mathcal{V}_1$. It suffices to verify the statement on a generating set of the Pauli group

$$\left[\mathcal{V}\mathcal{X}_i\mathcal{V}^\dagger\right] = \left[\mathcal{V}_2\mathcal{V}_1\mathcal{X}_i\mathcal{V}_1^\dagger\mathcal{V}_2^\dagger\right]$$
$$= \left[\mathcal{V}_1\mathcal{X}_i\mathcal{V}_1^\dagger\right]V_2$$
$$= X_iV_1V_2. \quad (25)$$

Equation (25) uses the fact that $\mathcal{V}_1\mathcal{X}_i\mathcal{V}_1^\dagger$ belongs to $\mathcal{G}_n$. The same kind of result holds for the $\mathcal{Z}_i$'s and this completes the proof. □

C. Decoding

When transmitted on a Pauli channel, an encoded state $|\overline{\psi}\rangle = \mathcal{V}(|\psi\rangle \otimes |0_{n-k}\rangle)$ (where $|\psi\rangle$ belongs to $(\mathbb{C}^2)^{\otimes k}$) will result in a state $\mathcal{P}|\overline{\psi}\rangle$ for some $\mathcal{P} \in \mathcal{G}_n$. Upon inverting the encoding, we obtain the state

$$\mathcal{V}^\dagger\mathcal{P}|\overline{\psi}\rangle = \mathcal{V}^\dagger\mathcal{P}\mathcal{V}(|\psi\rangle \otimes |0_{n-k}\rangle)$$
$$= (\mathcal{L}|\psi\rangle) \otimes (\mathcal{S}|0_{n-k}\rangle)$$

where $\mathcal{L}$ belongs to $\mathcal{G}_k$ and $\mathcal{S} = \alpha\mathcal{S}_1 \otimes \cdots \otimes \mathcal{S}_{n-k}$ belongs to $\mathcal{G}_{n-k}$ (and the $\mathcal{S}_i$'s to $\{\mathcal{I}, \mathcal{X}, \mathcal{Y}, \mathcal{Z}\}$). Notice that $\mathcal{S}|0_{n-k}\rangle$ is equal to $\epsilon|s_1\rangle \otimes \cdots \otimes |s_{n-k}\rangle$ where $\epsilon \in \{\pm 1, \pm i\}$ and

$$s_i = 0, \qquad \text{if } \mathcal{S}_i \in \{\mathcal{I},\ \mathcal{Z}\} \quad (26)$$
$$s_i = 1, \qquad \text{otherwise.} \quad (27)$$

Measuring the $n-k$ last qubits reveals $s_1 \cdots s_{n-k}$ which is the analogue of a classical syndrome. This motivates the following definition.

Definition 12 (Error Syndrome): The syndrome $s(\mathcal{P})$ associated to a Pauli error $\mathcal{P}$ is the binary vector $(s_i)_{1 \le i \le n-k}$ defined by (26) and (27).

Note that the syndrome $s(P)$ can be obtained from the H_i's (which are defined as in Section III-B by $H_i = \left[\mathcal{V}\mathcal{Z}_{k+i}\mathcal{V}^\dagger\right] = Z_{k+i}V$) by the following.

Proposition 1:

$$s(\mathcal{P}) = ([\mathcal{P}] \star H_i)_{1 \le i \le n-k}.$$

Proof: $s_i(\mathcal{P})$ is equal to $[\mathcal{P}]V^{-1} \star Z_{i+k}$ by definition. Since symplectic transformations preserve the symplectic inner product we deduce that $s_i(\mathcal{P}) = ([\mathcal{P}]V^{-1}) \star Z_{i+k} = [\mathcal{P}] \star Z_{i+k}V = [\mathcal{P}] \star H_i$. □

This proposition motivates the following definition of a parity-check matrix of a stabilizer code

Definition 13 (Parity-Check Matrix): The parity-check matrix H of a quantum code with stabilizer set $\{H_1, \ldots, H_{n-k}\}$ is the binary matrix of size $(n-k) \times 2n$ with rows $H_1, \ldots, H_{n-k}$.

The calculation of the syndrome depends only on the effective Pauli error $P = [\mathcal{P}]$. As we did for classical errors in Section II-A, it will be convenient to decompose the error as $PV^{-1} = (L : S)$, with $L \in G_k$ and $S \in G_{n-k}$. As in the

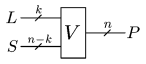

Fig. 5. Circuit representation of encoder $(L : S)V = P$. The operator $P \in G_n$ is a codeword (has trivial syndrome) if and only if $S \in \{I, Z\}^{n-k}$.

classical case, this is conveniently represented by the circuit diagram of Fig. 5. At this point, however, the analogy with the classical case partially breaks down. As described in Section II-A, in the classical setting, a bit-flip error p can be decomposed as $pV^{-1} = (l : s)$. In that case, s is the error syndrome and is, therefore, known. Decoding then consists in identifying the most likely l given knowledge of s. In the quantum case, however, S is only partially determined by the error syndrome $s(P)$. Indeed, we can decompose S as $S = S^x + S^z$ (cf., Notation 4), and notice that from (26) and (27), $s(P)$ reveals only S^x. More precisely, we have the following relation for the ith component S_i^x of S^x:

$$\begin{aligned} S_i^x &= X, \qquad \text{if } s_i = 1 \\ S_i^x &= I, \qquad \text{otherwise.} \end{aligned}$$

Hence, two physical errors $P = (L : S^x + S^z)V$ and $P' = (L : S^x + S'^z)V = P + (I_k : S^z + S'^z)V$ have the same error syndrome[2] S^x, so they cannot be distinguished. However, they also yield the same logical transformation L, so they can be corrected by the same operation (namely, applying $L = L^{-1}$ again). Therefore, they *cannot* and *need not* be distinguished by the error syndrome: such errors are called degenerate. This reflects the fact that all errors of the form $P = (I_k : S^z)V$ (with $S^z \in \{I, Z\}^{n-k}$) have zero syndrome but do not need to be corrected. We denote such kind of errors by the following.

Definition 14 (Harmless Undetected Errors): The set of errors P of the form $P = (I_k : S^z)V$ where S^z ranges over $\{I, Z\}^{n-k}$ is called the set of harmless undetected errors.

All the other errors of zero syndrome (and which are, therefore, undetected) have a nontrivial action on the k first qubits after inverting the encoding transformation. This motivates the following definition.

Definition 15 (Harmful Undetected Errors): The set of errors P of the form $P = (L : S^z)V$, where S^z ranges over $\{I, Z\}^{n-k}$ and L is different from I_k, is called the set of harmless undetected errors.

Note that the set of errors of the form $(I_k : S^z)V$ with S^z in $\{I, Z\}^{n-k}$ is also the subgroup spanned by the rows V_{2i} for $i \in \{k+1, \ldots, n\}$, or what is the same, the subgroup spanned by the $H_i \triangleq Z_{i+k}V$ for $i \in \{1, \ldots, n-k\}$. In other words, we have Proposition 2.

Proposition 2: The set of harmless undetected errors is equal to $C(I)$.

This fact that there are errors which do no need to be corrected has an important consequence. Contrarily to the classical setting where the most likely error satisfying the measured syndrome is sought, in the quantum case, we look for the most likely coset of $C(I)$ satisfying the measured syndrome. Such a coset is the set of errors of the following form.

Definition 16 (Logical Coset): Given an encoding matrix V, the *logical coset* $C(L, S^x)$ associated to the logical transformation $L \in G_k$ and to the syndrome S^x (belonging to $\{I, X\}^{n-k}$) is defined as

$$\begin{aligned} C(L, S^x) &= \{P = (L : S^z + S^x)V \mid S^z \in \{I, Z\}^{n-k}\} \\ &= (L : S^x)V + C(I). \end{aligned}$$

When $S^x = I_{n-k}$, we simply write $C(L)$ instead of $C(L, I_{n-k})$.

What replaces the classical probability that a given information sequence has been sent given a measured syndrome is in the quantum case the probability $\mathbf{P}(L|S^x)$ that applying the transformation $L^{-1} = L$ to the k first qubits after performing the inverse of the encoding operation corrects the error on these qubits. It corresponds to the probability that the error belongs to the coset $C(L, S^x)$, which is, therefore, equal to

$$\mathbf{P}(L|S^x) = \frac{\mathbf{P}(L, S^x)}{\sum_{L'} \mathbf{P}(L', S^x)} \tag{28}$$

where the probability $\mathbf{P}(L, S^x)$ is the pullback of $\mathbf{P}(P)$ through the encoding matrix

$$\mathbf{P}(L, S^x) = \sum_{S^z \in \{I,Z\}^{n-k}} \mathbf{P}(P)\Big|_{P=(L:S^x+S^z)V^{-1}}. \tag{29}$$

Similarly to the classical setting, maximum-likelihood decoding consists in identifying the most likely logical transformation L given the syndrome S^x. More formally, we have the following definition.

Definition 17 (Maximum-Likelihood Decoder): The maximum-likelihood decoder $L_{\mathrm{ML}} : \{I, X\}^{n-k} \to G_k$ is defined by

$$L_{\mathrm{ML}}(S^x) = \mathrm{argmax}_L \mathbf{P}(L|S^x). \tag{30}$$

The classical maximum *a posteriori* (MAP) decoding (or bitwise decoding) has also a quantum analogue.

Definition 18 (Qubitwise Maximum-Likelihood Decoder): The qubitwise maximum-likelihood decoder $L^i_{\mathrm{ML}} : \{I, X\}^{n-k} \to G_1$ is defined by

$$L^i_{\mathrm{ML}}(S^x) = \mathrm{argmax}_{L^i} \mathbf{P}(L^i|S^x) \tag{31}$$

where the marginal conditional probability is defined the usual way

$$\mathbf{P}(L^i|S^x) = \sum_{L^1, \ldots L^{i-1}, L^{i+1}, \ldots L^k} \mathbf{P}(L^1, \ldots L^k|S^x). \tag{32}$$

Equation (29) differs from its classical analogue (5) by a summation over S^z, which reflects the coset structure of the code.

[2]By a slight abuse of terminology, we use the one-to-one correspondence between s and S^x to refer to both quantities as the error syndrome.

Aside from this distinction, the maximum-likelihood decoders are defined as in the classical case.

D. Comparison Between Stabilizer Codes and Classical Linear Codes

One of the main advantage of the stabilizer formalism is that it allows to discretize a seemingly continuous problem by studying the effect of Pauli errors (which are discrete) on the continuous code subspace. By classifying these errors, discrete quantities such as error syndromes or parity-check matrices arise naturally. In other words, stabilizer codes share many analogies with classical linear codes, but there are also some fundamental differences. Let us summarize these analogies and differences here. We assume in what follows that the relevant quantum quantities are defined for a stabilizer code $\mathcal{C}$ of length n and rate $\frac{k}{n}$.

Syndrome and parity-check matrix. The parity-check matrix H is a binary matrix of size $(n-k) \times 2n$. It differs from a classical parity-check matrix in two respects.

1) Its rows H_i must be orthogonal with respect to the $\star$-product.
2) The syndrome $s(P)$ of a Pauli error P in G_n is defined with the help of the $\star$-product (rather than by matrix multiplication): $s(P) = (H_i \star P)_{1 \leq i \leq n-k}$.

Encoding matrix. It is a binary matrix V of size $2n \times 2n$ and must be a symplectic matrix (and any symplectic matrix is the encoding matrix of a certain stabilizer code). Because it is a symplectic matrix $V^{-1} = \Lambda_n V^T \Lambda_n$, it plays a role analogous to both the classical encoding matrix and its inverse. As the classical encoding matrix (3), it contains a generator matrix as a submatrix. As the inverse of the classical encoding matrix (4), it also contains a parity check matrix as a submatrix. The parity check matrix is formed of rows $V_{2(k+1)}, V_{2(k+2)}, \ldots, V_{2n}$ while the generating matrix consists of rows $V_1, V_2, \ldots, V_{2k}$. The remaining rows $V_{2k+1}, V_{2(k+1)+1}, \ldots, V_{2n-1}$ are sometimes referred to as "pure errors" [34]. Indeed, taking the rows of V as generators of G_n, the syndrome associated to an element of G_n depends only on its pure error component. Hence, their classical analogue is the matrix $(H^{-1})^T$ appearing in the classical encoding matrix (4).

The encoding matrix V is associated to a (continuous) unitary encoding transformation $\mathcal{V}$. As in the classical case, the natural decoding process consists in inverting $\mathcal{V}$ and measuring the last $n-k$ qubits, which yields a syndrome that is associated to a parity check matrix.

Code. We may define the discrete stabilizer code as in the classical setting as the set of errors with zero syndrome, as shown in the following.

Definition 19 (Discrete Stabilizer Code): The discrete stabilizer code C associated to the stabilizer set $\{H_i, i = 1 \ldots n-k\}$, where the H_i's are independent mutually orthogonal elements of G_n, is the subspace of G_n orthogonal to the H_i, that is

$$C = \{P \in G_n \mid H_i \star P = 0, 1 \leq i \leq n-k\} \qquad (33)$$

or more succinctly, $C = C(I)^{\perp}$.

Codewords. There is an important difference between the classical setting and the quantum setting here. Since all elements of a coset of $C(I)$ have the same effect on $\mathcal{C}$, we make no distinction between the elements of such cosets. Therefore, the codewords in the quantum setting are grouped in cosets of $C(I)$. Note that all elements of the coset $C(I)$ are the analogue of the zero codeword. With the notation introduced in Section III-C, we have

$$C = \bigcup_{L \in G_k} C(L). \qquad (34)$$

Minimum distance. In the classical setting, the minimum distance of a linear code is the smallest Hamming weight of a nonzero codeword. This definition carries over to the quantum setting with the coset $C(I)$ playing the role of the zero codeword. Thus, the minimal distance of a code is the minimum weight $w(P)$ of an element P of $C - C(I)$. With this definition of the minimum distance d, it is straightforward to check that the number of errors which are corrected by a decoder that outputs the coset $C(L, S^x)$ containing the element P of lowest weight and satisfying the syndrome S^x is equal to $\lfloor \frac{d-1}{2} \rfloor$.

Information symbols. There is in the quantum setting a natural notion of information sequence corresponding to a Pauli error P, which consists in taking the element L in G_k such that there exists an S in G_{n-k} for which $(L:S)V = P$.

IV. Quantum Turbo Codes

In this section, we describe quantum turbo codes obtained from interleaved serial concatenation of quantum convolutional codes. This first requires the definition of quantum convolutional codes. We will define them through their circuit representation as in [31] rather than through their parity-check matrix as in [1], [15], and [18]: this allows to define in a natural way the state diagram and is also quite helpful for describing the decoding algorithm.

A. Quantum Convolutional Codes

A quantum convolutional encoder can be defined quite succinctly as a stabilizer code with encoding matrix V given by the circuit diagram of Fig. 6. The circuit is built from repeated uses of the seed transformation U shifted by n qubits. In this circuit, particular attention must be paid to the order of the inputs as they alternate between stabilizer qubits and logical qubits. This is a slight deviation from the convention established in the previous section, and it is convenient to introduce the following notation to label the different qubits appearing in the encoding matrix of a quantum stabilizer code.

Definition 20: The positions corresponding to L are called the *logical positions* and the positions corresponding to S are called the *syndrome positions*.

The total number of identical repetition of the seed transformation U is called the duration of the code and is denoted N. The m qubits that connect gates from consecutive time slices are called memory qubits. The encoding is initialized by setting the first m memory qubits in the $|0_m\rangle$ state. To terminate the encoding, set the k information qubits of the last t time slices in the $|0_k\rangle$ state, where t is a free parameter independent of N.

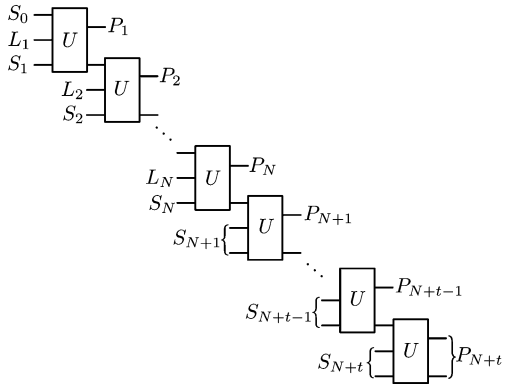

Fig. 6. Circuit diagram of a quantum convolutional encoder with seed transformation U. The superscripts indicating the number of qubits per wire are omitted for clarity, and can be found in Fig. 7.

The rate of the code is thus $kN/(n(N+t)+m)$ which is of the form $k/n + O(1/N)$ for fixed t.

Formally, a quantum convolutional code can be defined as follows.

Definition 21 (Quantum Convolutional Encoder): Let n, k, m, and t be integers defining the parameters of the code, and N the duration of the encoding. Let U be an $(n+m)$-qubit symplectic matrix called the seed transformation. The encoding matrix V of the quantum convolutional encoder is a symplectic matrix over $m + n(N+t)$ qubits given by

$$\begin{aligned} V &= U_{[1\ldots n+m]} U_{[n+1\ldots 2n+m]} \cdots U_{[(N+t-1)n+1\ldots(N+t)n+m]} \\ &= \prod_{i=1}^{N+t} U_{[(i-1)n+1\ldots in+m]} \end{aligned}$$

where $[a\ldots b]$ stands for the integer interval $\{a, a+1, \ldots, b\}$ and where $U_{[(i-1)n+1\ldots in+m]}$ acts on an element $(P_1, \ldots, P_{m+n(N+t)}) \in G_{m+n(N+t)}$ such that its image $(P'_1, \ldots, P'_{m+n(N+t)})$ satisfies $(P'_{(i-1)n+1}, \ldots, P'_{in+m}) = (P_{(i-1)n+1}, \ldots, P_{in+m})U$ and all other P_i are given by $P'_i = P_i$. The syndrome symbols correspond to the positions belonging to $[1\ldots m] \cup \bigcup_{i\in[1\ldots N]}[(i-1)n+m+k+1\ldots in+m] \cup \bigcup_{i\in[N+1\ldots(N+t)]}[(i-1)n+m\ldots in+m]$.

It will be convenient to decompose an element P in $G_{n(N+t)+m}$ as $P = (P_1 : P_2 : \cdots : P_{N+t})$ where the P_i's belong to G_n for i in $\{1, 2, \ldots, N+t-1\}$ and P_{N+t}'s belongs to G_{n+m}. This decomposition directly reflects the structure of the output wires appearing on the right-hand side of the circuit diagram of Fig. 6.

Similarly, we will decompose the Pauli-stream obtained by applying the inverse encoder to P as

$$(S_0 : L_1 : S_1 : \cdots : L_N : S_N : S_{N+1} : \cdots : S_{N+t}) \triangleq PV^{-1},$$

where S_0 belongs to G_m, the L_i's all belong to G_k, the S_i's belong to G_{n-k} for i in $\{1, \ldots, N\}$, and the S_{N+j}'s belong to G_n for j in $\{1, \ldots, t\}$. This decomposition directly reflects the structure of the input wires appearing on the left-hand side of the circuit diagram of Fig. 6.

While the P_j's are related to the L_j and S_j via a matrix V of dimension $2(N+t)n + 2m$, the convoluted structure of V can be exploited to recursively compute this transformation without the need to manipulate objects of size increasing with N. This requires the introduction of auxiliary memory variables $M_j \in G_m$. The recursion is initialized by setting

$$(M_{N+t-1} : S_{N+t}) \triangleq P_{N+t}U^{-1}. \tag{35}$$

The S_j for $i \in \{N+1, \ldots, N+t-1\}$ are obtained by recursion on i

$$(M_{i-1} : S_i) \triangleq (P_i : M_i)U^{-1} \tag{36}$$

and the M_{i-1}, L_i, S_i for i in $\{1, \ldots, N\}$ are obtained from the recursion

$$(M_{i-1} : L_i : S_i) \triangleq (P_i : M_i)U^{-1}. \tag{37}$$

Finally, set

$$S_0 = M_0. \tag{38}$$

Clifford transformation U on $n+m$ qubits can be used as a seed transformation and defines a convolutional code. It will be useful to decompose U into blocks of various sizes

$$U = \begin{pmatrix} \overbrace{\mu_{\mathrm{P}}}^{2n} & \overbrace{\mu_{\mathrm{M}}}^{2m} \\ \Lambda_{\mathrm{P}} & \Lambda_{\mathrm{M}} \\ \Omega_{\mathrm{P}} & \Omega_{\mathrm{M}} \end{pmatrix} \begin{matrix} \}2m \\ \}2k \\ \}2(n-k) \end{matrix}. \tag{39}$$

Just as in the classical case, this definition of quantum convolutional code can easily be seen to be equivalent to the ones that have previously appeared in the literature [1], [15], [18]. In particular, the D-transform associated to the code can easily be obtained from the submatrices of V appearing in (39). However, these concepts will not be important for our analysis.

Our definition of convolutional code is stated in terms of their encoding matrix V. From this perspective, convolutional codes are ordinary, albeit very large, stabilizer codes. However, there are important aspects of convolutional codes that distinguish them from generic stabilizer codes.

As mentioned in Section III-B, stabilizer codes have, in general, encoding circuits using a number of elements proportional to the square of the number of physical qubits. Convolutional codes have by definition circuit complexity that scales *linearly* with N for fixed m: each application of the seed transformation U requires a constant number of gates, and this transformation is repeated $N+t$ times.

The most important distinction, however, has to do with the decoding complexity. The maximum-likelihood decoder of a stabilizer code consists in an optimization over the logical cosets, of which there are 4^K where K denotes the number of encoded qubits. Without any additional structure on V, maximum-likelihood decoding is an NP-hard problem [5]. Quantum convolutional codes on the other hand have decoding complexity that scales *linearly* with K. The algorithm that accomplishes this task will be described in details in Section V.

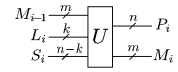

Fig. 7. Seed transformation circuit.

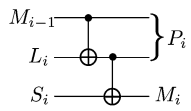

Fig. 8. Seed transformation for an $n = 1$, $k = 1$, and $m = 1$ quantum convolutional code. It corresponds to a unitary transform which maps $|a\rangle \otimes |b\rangle \otimes |c\rangle$ to $|a\rangle \otimes |a+b\rangle|a+b+c\rangle$ for $a, b, c \in \{0, 1\}$. Therefore, the seed transformation U acts as follows on the Z_i and X_i: $Z_1U = (Z, I, I)U = (Z, I, I)$, $X_1U = (X, X, X)$, $Z_2U = (Z, Z, I)$, $X_2U = (I, X, X)$, $Z_3U = (I, Z, Z)$, $X_3U = (I, I, X)$.

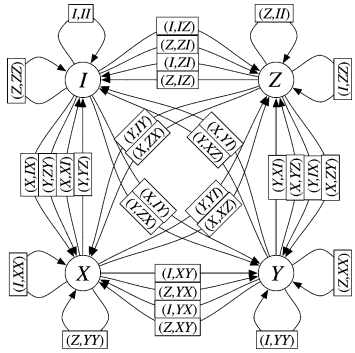

Fig. 9. State diagram for the seed transformation shown in Fig. 8.

B. State Diagram

We will now define some properties of convolutional codes that will play important roles in the analysis of the performance of turbo codes. Most of these definitions rely on the state diagram of a convolutional code, which is defined similarly as in the classical case.

Definition 22 (State Diagram): The state diagram of an encoder with seed transformation U and parameters (n, k, m) is a directed multigraph with 4^m vertices called memory states, each labeled by an $M \in G_m$. Two vertices M and M' are linked by an edge $M \to M'$ with label (L, P) if and only if there exists $L \in G_k$, $P \in G_n$, and a $S^z \in \{I, Z\}^{n-k}$ such that

$$P : M' = (M : L : S^z)U. \tag{40}$$

The labels L and P are referred to as the logical label and the physical label of the edge, respectively.

Thus, the state diagram represents partial information about the transformation $(M : L : S^z) \to (P : M')$ generated by the seed transformation U, partial information because all information about S^z is discarded. Note that $S^z \in \{I, Z\}^{n-k}$, so the state diagram only contains information about the streams of Pauli operators that remain in the set of codewords C. The restriction on the S^z input can be lifted if we instead consider the effective seed transformation

$$U_{\text{eff}} \triangleq \begin{pmatrix} \overbrace{\mu_{\text{P}}}^{2n} & \overbrace{\mu_{\text{M}}}^{2m} \\ \Lambda_{\text{P}} & \Lambda_{\text{M}} \\ \Sigma_{\text{P}} & \Sigma_{\text{M}} \end{pmatrix} \begin{matrix} \}2m \\ \}2k \\ \}n-k \end{matrix} \tag{41}$$

where the matrix $[\Sigma_{\text{P}} : \Sigma_{\text{M}}]$ is obtained by removing every second row from the matrix $[\Omega_{\text{P}} : \Omega_{\text{M}}]$ (i.e., the rows which represent the action on the $\mathcal{X}_i$). This definition will be convenient for later analysis.

The state diagram of the seed transformation represented in Fig. 8 is shown in Fig. 9. For instance, the self-loop at I labeled (I, II) represents the trivial fact that $(I : I : I)U = (II : I)$. The edge from Y to I labeled (Y, XZ) represents the transformation $(Y : Y : I)U = (XZ : I)$, and so on.

The state diagram is crucial for analyzing the properties of the associated code, and also for defining some of its essential features. Here, we give some definitions based on the state diagram that will be important in our analysis.

Definition 23 (Path): A path in the state diagram is a sequence of vertices $M_1, M_2, \ldots$ such that $M_i \to M_{i+1}$ belongs to the state diagram.

Each element of C is naturally associated to a path in the state diagram, which corresponds to the memory states visited upon its encoding. The physical and logical weight of a codeword can be obtained by adding the corresponding weights of the edges in the path associated to the codeword. More generally, we will refer to the weight of a path as the sum of the weight of its edges.

Definition 24 (Zero-Physical-Weight Cycle): A zero-physical-weight cycle is a closed path in the state diagram that uses only edges with zero physical weight.

In the state diagram of Fig. 9, for example, there are two zero-physical-weight cycles corresponding to the transformations $(I : I : I)U = (II : I)$ and $(Z : Z : Z)U = (II : Z)$.

Definition 25 (Noncatastrophic Encoder): An encoder C is *noncatastrophic* if and only if the only cycles in its state diagram with physical weight 0 have logical weight 0.

We see, for instance, that the state diagram of our running example is catastrophic due to the presence of the self-loop with label (Z, II) at state Z: this cycle has physical weight 0 and logical weight 1. To understand the consequences of a catastrophic seed transformation, consider the act of inverting the encoding transformation of the associated convolutional encoder. This is done by running the circuit of Fig. 6 backwards. Suppose that a single Y error affected the transmitted qubits. More specifically, at time i, $1 \leq i \leq N$, there is a Y on the lower physical wire of the seed transformation of Fig. 8 (i.e., $P_i^2 = Y$) and everything else is I. Since $(IY : I)U^{-1} = (Z : Y : X)$, this will result in a Z in the memory qubit M_{i-1}, a Y in the logical qubit L_i, and an X in the stabilizer qubit S_i. The $S_i = X$ triggers a nontrivial syndrome, which signals the presence of an error. Moreover, because of the self-loop at $M = Z$ that has nonzero logical weight but zero physical weight, this error will continue to propagate without triggering additional syndrome bits, while creating Z's in L_{i-1} and M_{i-2}, and in L_{i-2} and M_{i-3}, and so on. Thus, an error of finite physical weight results in an error of unbounded logical weight, and a finite syndrome. This is the essence of catastrophic error propagation.

Catastrophic encoders may have large minimal distances, but perform poorly under iterative decoding. All the codes we have

considered in our numerical simulations were noncatastrophic. In fact, they even satisfied the following stronger condition.

Definition 26 (Completely Noncatastrophic Code): A completely noncatastrophic code is such that the only loop in its state diagram with physical weight zero is the self-loop at I_m.

In the classical setting, noncatastrophicity is insured, for instance, by the use of systematic encoders. For such encoders, the logical string c is contained as a substring of the encoded string $\overline{c} = cV$. Systematic quantum encoding can be obtained by setting the first k columns of $\Lambda_{\mathrm{P}} = \mathbb{1}_k$ and the first k columns of $\Sigma_{\mathrm{P}} = \mu_{\mathrm{P}} = 0$ [cf., (41)]. However, this would imply that the stabilizers act trivially on the first k output qubits, resulting in a minimal distance equal to 1. We conclude that it is not possible to design a systematic quantum encoder with minimal distance greater than 1. Thus, noncatastrophicity is a condition that needs to be built in by hand. Fortunately, it can be efficiently verified directly on the state diagram and we have made great use of this fact.

In the classical setting, turbo codes can be designed with a minimal distance that grows polynomially with N when the inner code is recursive. Recall that recursive means that the encoder has an infinite impulsive response: when a single 1 is input at any logical wire of the encoding circuit Fig. 2 and every other input is 0, the resulting output has infinite weight for a code of infinite duration. This definition can be generalized to the quantum setting.

Definition 27 (Quasi-Recursive Encoder): Consider executing the encoding circuit of Fig. 6 on an input containing a single nonidentity Pauli operators on a logical wire, with all other inputs set to I. The corresponding encoder V is *quasi-recursive* when the resulting output has infinite weight when the code has infinite duration N.

However, it can be verified that this notion of recursiveness is too weak to derive a good lower bound on the minimal distance of turbo codes. This departure from the theory of classical codes stems from the fact that quantum codes are coset codes. As in the classical case, the proper definition of a recursive encoder demands that it generates an infinite impulsive response. The novelty comes from the fact that this must be true for every elements in the coset associated to the impulsive logical input: not only must the encoded version of X_i, Y_i, and Z_i have weight growing with the duration of the code N, but also so must every elements of $C(X_i), C(Y_i)$, and $C(Z_i)$. Formally, we can define recursive quantum convolutional encoders in the following two steps.

Definition 28 (Admissible Path): A path in the state diagram is *admissible* if and only if its first edge is not part of a zero physical-weight cycle.

Definition 29 (Recursive Encoder): A recursive encoder is such that any admissible path with logical weight 1 starting from a vertex belonging to a zero physical-weight loop does not contain a zero physical-weight loop.

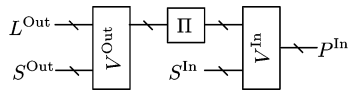

Fig. 10. Circuit diagram for a turbo encoder.

Once again, this property can be directly and efficiently tested given the seed transformation of the convolutional code by constructing its state diagram.

C. Interleaved Serial Concatenation

Quantum turbo codes are obtained from a particular form of interleaved concatenation of quantum convolutional codes. Interleaving is slightly more complex in the quantum setting since in addition to permuting the qubits it is also possible to perform a Clifford transformation on each qubit which amounts to permute X, Y, and Z. More precisely, we have the following.

Definition 30 (Quantum Interleaver): A *quantum interleaver* Π of size N is an N-qubit symplectic transformation composed of a permutation π of the N qubit registers and a tensor product of single-qubit symplectic transformation. It acts as follows by multiplication on the right on G_N:

$$(P_1, \ldots, P_N) \mapsto (P_{\pi(1)}K_1, \ldots, P_{\pi(N)}K_N)$$

where $K_1, \ldots, K_N$ are some fixed symplectic matrices acting on G_1.

It follows that interleavers preserve the weight of N-Pauli streams. An interleaved serial concatenation of two quantum encoders has three basic components:

1) an *outer code* encoding k^{Out} qubits by embedding them in a register of n^{Out} qubits, with encoder V^{Out};
2) an *inner code* encoding k^{In} qubits by embedding them in a register of n^{In} qubits, with encoder V^{Out} and which is such that $k^{\mathrm{In}} = n^{\mathrm{Out}}$;
3) a *quantum interleaver* Π of size $N = n^{\mathrm{Out}} = k^{\mathrm{In}}$.

The resulting encoding matrix of the interleaved concatenated code is a symplectic matrix V acting on $G_{n^{\mathrm{In}}}$ such that

$$V = V'^{\mathrm{Out}}\Pi' V^{\mathrm{In}}$$

with the action of V'^{Out} and Π' on $G_{n^{\mathrm{In}}}$ being defined by

$$(L : S^{\mathrm{Out}} : S^{\mathrm{In}})V'^{\mathrm{Out}} = ((L : S^{\mathrm{Out}})V^{\mathrm{Out}} : S^{\mathrm{In}}) \qquad (42)$$

for $(L : S^{\mathrm{Out}} : S^{\mathrm{In}}) \in G_{k^{\mathrm{Out}}} \times G_{n^{\mathrm{Out}}-k^{\mathrm{Out}}} \times G_{n^{\mathrm{In}}-k^{\mathrm{In}}}$, and

$$(L' : S^{\mathrm{In}})\Pi' = (L'\Pi : S^{\mathrm{In}}) \qquad (43)$$

for $L' \in G_{n^{\mathrm{Out}}}$. These relations are summarized in Fig. 10.

The rate of the concatenated code is equal to $\frac{k^{\mathrm{Out}}}{n^{\mathrm{In}}} = \frac{k^{\mathrm{Out}}}{n^{\mathrm{Out}}}\frac{k^{\mathrm{In}}}{n^{\mathrm{In}}}$, which is the product of the rates of the inner code and the outer code.

A serial quantum turbo code is obtained from this interleaved concatenation scheme by choosing V^{Out} and V^{In} as quantum convolutional encoders.

D. Figure of Merit

There are a number of not equivalent ways of characterizing the performance of a code. We might use the minimum distance, but a quantity that is more informative is the weight enumerator, which counts the number of undetected harmful errors of each weight. For a convolutional code, however, as the number K of encoded qubits tends to infinity, the weight enumerator will be infinite. Indeed, because of the translational invariance of the encoding circuit, finite error patterns come in an infinite number of copies obtained by translation. Instead, we can consider the *distance spectrum* of a noncatastrophic encoder, which is defined as follows.

Definition 31 (Distance Spectrum): The *distance spectrum* $(F(w))_{w\geq 0}$ of a noncatastrophic convolutional encoder is a sequence for which $F(w)$ is the number of admissible paths in the state diagram starting and ending in memory states that are part of zero-weight cycles, and with physical weight w and logical weight greater than 0.

Another relevant quantity is the distance spectrum for logical-weight-one elements of C, which is defined similarly.

Definition 32 (Logical-Weight-One Distance Spectrum): The distance spectrum for logical-weight-one codewords $F_1(w)$ of a noncatastrophic convolutional encoder is the number of admissible paths in the state diagram starting and ending in memory states that are part of zero-weight cycles, and with physical weight w and logical weight 1.

It can easily be seen that the minimum distance of a turbo code obtained from the concatenation of two convolutional codes is no greater than $d_*^{\rm Out} * d_1^{\rm In}$ where $d_1 = \min_w\{F_1(w) > 0\}$ and the free minimal distance is $d_* = \min_w\{F(w) > 0\}$. The free distance is defined similarly to the classical case by the smallest weight of a harmful undetected error in the convolutional code with infinite time duration. It is so-to-speak a kind of typical minimal distance for convolutional codes, ignoring finite-size effects. To maximize the minimum distance of the turbo code, we must use outer codes with large free distances d_* and inner encoders with large value of d_1. Recursive encoders, for instance, have d_1 proportional to N, and therefore, serve as ideal inner codes. However, it happens that we cannot use recursive encoders as inner codes as we will see in Section IV-E. Hence, a good rule of thumb is to use inner encoders that minimize the value of $F_1(w)$ at small w, and similarly use an outer code which minimizes the value of $F(w)$ at small w. These will result in a turbo code with a distance spectrum that is small at low distances.

Finally, given an error model, the word error rate (WER) and qubit error rate (QER) provide a good operational figure of merit. The QER is the probability that an individual logical qubit is incorrectly decoded. In other words, the QER represents the fraction of logical qubits that have errors after the decoding. The WER is the probability that at least one qubit in the block is incorrectly decoded. We expect, in general, QER $\ll$ WER. The WER is thus a much more strenuous figure of merit than the QER. For instance, if N qubits are encoded in N/k block codes for some constant k, then as N increases, the WER approaches 1 exponentially while the QER remains constant. As we will see, turbo codes have a completely different behavior. In general, we will be interested in the WER averaged over the choice of interleaver Π.

E. Recursive Convolutional Encoders Are Catastrophic

In the classical setting, noncatastrophic and recursive convolutional encoders are of particular interest. When used as the inner encoder of a concatenated coding scheme, the resulting code has a minimal distance that grows polynomially with their length and offer good iterative decoding performances. More precisely, random serial turbo codes have a minimum distance which is typically of order $N^{\frac{d_*^{\rm Out}-2}{d_*^{\rm Out}}}$ when the inner encoder is recursive, where N is the length of the concatenated code and $d_*^{\rm Out}$ the free distance of the outer code [22]. That the encoder be noncatastrophic is important to obtain good iterative decoding performances.

This result and its proof would carry over the quantum setting almost verbatim with our definition of recursive encoders. The quantum case is slightly more subtle due to the coset structure of the code. Unfortunately, such encoders do not exist.

Theorem 1: Quantum convolutional recursive encoders are catastrophic.

This result is perhaps surprising since the notions of catastrophic and recursive are quite distinct in the classical setting. Nonetheless, the stringent symplectic constraints imposed to the seed transformation U give rise to a conflicting relation between them. The proof of Theorem 1 is rather involved. Here, we present its main steps and leave the details to the Appendix.

The proof involves manipulation of the rows of the effective encoding matrix (41), and for that reason, it is more appropriate to view effective Pauli operators as elements of $\mathbb{F}_2^{2n}$. The proof proceeds directly by demonstrating that the state diagram of any recursive convolutional encoder contains a directed cycle with zero physical weight and nonzero logical weight. We first need a characterization of the memory states M that can be part of a zero physical weight cycle. We break this into three steps. First, we characterize the set of states that are the endpoint of edges in the state diagram with zero physical-weight edges. In other words, we want to find all possible values for the memory element M' in $\mathbb{F}_2^{2m}$ such that there exist $M \in \mathbb{F}_2^{2m}$, $S \in \mathbb{F}_2^{n-k}$, and $L \in \mathbb{F}_2^{2k}$ such that $(M : L : S)U_{\rm eff} = (\mathbf{0}_{2n} : M')$.

Lemma 2: Given a seed transformation U, let $\mathfrak{S}$ be the subspace of $\mathbb{F}_2^{2m}$ spanned by the rows of $\Sigma_{\rm M}$. The set of endpoints of edges with zero physical label is equal to $\mathfrak{S}^\perp$, and conversely, any M in $\mathfrak{S}^\perp$ is the end vertex in the state diagram of exactly one edge of zero physical weight.

If the state diagram contains a zero physical-weight cycle, it is therefore necessarily supported on the subset of vertices $\mathfrak{S}^\perp$. However, edges of zero physical weight with endpoints vertices in $\mathfrak{S}^\perp$ may originate from vertices outside $\mathfrak{S}^\perp$. Such edges are not part of zero-physical-weight cycles. The next step is thus to characterize the set of endpoints of edges with zero physical label and starting point in $\mathfrak{S}^\perp$. Since in the absence

of other inputs each time interval modifies the memory state by $M \to M\mu_{\mathrm{M}}$, we intuitively expect this set to be $\mathfrak{S}_0^{\perp}$, where $\mathfrak{S}_0$ is the smallest subspace containing $\mathfrak{S}$ and stable by μ_{M}. This is confirmed by the following lemma.

Lemma 3: Given a seed transformation U, let

$$\mathfrak{S}_0 \triangleq \sum_{i=0}^{\infty} \mathfrak{S}\mu_{\mathrm{M}}^i. \tag{44}$$

For any element M' of $\mathfrak{S}_0^{\perp}$, there exists a unique element M in $\mathfrak{S}_0^{\perp}$, such that there is an edge of physical weight 0 from M to M'.

This lemma narrows down the set of vertices in the state diagram that can support zero physical-weight cycles. In particular, we can define a subgraph of the state diagram obtained from the vertex set $\mathfrak{S}_0^{\perp}$ and directed edges with trivial physical labels. This subgraph is guaranteed to have constant in-degree 1 for all its vertices, but some of its vertices may have no outgoing edges. These would definitely not be part of a cycle. To ensure that all vertices in the subgraph have a positive number of outgoing edges, we must once more restrict its set of vertices. The (left) nullspace of μ_{M}^i's will play a fundamental role.

Notation 7: Let μ be a linear mapping from $\mathbb{F}_2^{2m}$ to itself. We denote by $\mathrm{Null}(\mu)$ the (left) nullspace of μ, that is

$$\mathrm{Null}(\mu) = \left\{ M \in \mathbb{F}_2^{2m} | M\mu = \mathbf{0}_{2m} \right\}.$$

Notation 8: Let $\mathfrak{N}_0 \triangleq \sum_{i=1}^{\infty} \mathrm{Null}(\mu_{\mathrm{M}}^i)$ and $\mathfrak{V}_0 = \mathfrak{S}_0 + \mathfrak{N}_0$. Let $\mathfrak{G}$ be a subgraph of the state diagram obtained from the vertex set $\mathfrak{V}_0^{\perp}$ and edges with trivial physical label. This graph is called the *kernel graph* of the quantum convolutional code with seed transformation U.

By replacing the vertex set $\mathfrak{S}_0^{\perp}$ by $\mathfrak{V}_0^{\perp}$, our goal was to eliminate any vertex with no outgoing edge. This turns out to be successful as shown by the following lemma.

Lemma 4: The kernel graph has constant in-degree 1 and positive out-degree for any vertex.

Thus, any cycle with zero physical weight must be supported on the kernel graph of the seed transformation. The next step in order to prove Theorem 1 is to demonstrate that when the quantum convolutional encoder is recursive, its corresponding kernel graph $\mathfrak{G}$ does not only consist of the single zero vertex with a self-loop attached to it, corresponding to the trivial relation $(\mathbf{0}_{2m} : \mathbf{0}_{2k} : \mathbf{0}_{n-k})U_{\mathrm{eff}} = (\mathbf{0}_{2n} : \mathbf{0}_{2m})$.

Lemma 5: The kernel graph of a recursive quantum convolutional encoder has strictly more than one vertex.

This result is an essential distinction between the quantum case and the classical case. In the classical case, when the memory state is nonzero, it is always possible to create a nonzero physical output, for instance, by copying the state of the memory at the output. But this is not possible quantum mechanically.

Before proving that $\mathfrak{G}$ contains a cycle with nonzero logical weight, we will first prove that it contains at least one edge with nonzero logical weight. For this purpose, let us characterize the subset of edges with zero physical weight and zero logical weight.

Lemma 6: Given a seed transformation U, let $\mathfrak{L}$ be the subspace of $\mathbb{F}_2^{2m}$ spanned by the rows of Λ_{M} and Σ_{M}. The set of endpoints of edges of zero physical and logical weight is equal to $\mathfrak{L}^{\perp}$.

This result and its proof are structurally similar to Lemma 2, except that $\mathfrak{S}$ has been replaced by $\mathfrak{L}$. From this, we conclude as follows.

Lemma 7: The kernel graph of a recursive quantum convolutional encoder contains an edge with nonzero logical weight.

Armed with this result, we are now in a position to prove the main result of this section.

Proof (of Theorem 1): Consider a recursive quantum convolutional encoder and its associated kernel graph. By Lemma 7, this graph has at least one edge with nonzero logical weight. Let us say that it goes from M_0 to M_1. From Lemma 4, we can follow a directed path of arbitrary length l with (M_0, M_1) as starting edge

$$M_0 \to M_1 \to \cdots \to M_{t-1} \to M_t.$$

If the length of the path is greater than the number of vertices of the graph, it must contain at least twice the same vertex. Moreover, M_0 must be part of this cycle. Otherwise, we would have a path of the form $M_0 \to M_1 \to \cdots M_j \to M_{j+1} \to \cdots \to M_l = M_j$ with $j > 0$. In this case, M_j would have in-degree 2, which is impossible. In other words, there is a directed cycle in the state diagram with zero physical weight and nonzero logical weight. The corresponding convolutional encoder is, therefore, catastrophic. □

V. Decoding

This section describes the decoding procedure for turbo codes operated on memoryless Pauli channels. With an n-qubit memoryless Pauli channel, errors are elements of G_n distributed according to a product distribution $\mathbf{P}(P_1 : P_2 : \cdots : P_n) = f_1(P_1)f_2(P_2)\ldots f_n(P_n)$. The depolarizing channel described in Section III is a particular example of such a channel where all f_j are equal. We note that our algorithm can be extended to non-Pauli errors using the belief propagation algorithm of [25], but leave this generalization for a future paper. The decoding algorithm we present is an adaptation to the quantum setting of the usual soft-input–soft-output (SISO) algorithm used to decode serial turbo codes (see [2]). It differs from the classical version in several points.

1) As explained in Section III-C, for decoding a quantum code, we do not consider the state of the qubits directly (which belong to a continuous space and which cannot be measured without being disturbed) but instead consider the Pauli error (which is discrete) that has affected the quantum state. Decoding consists in inferring the transformation that has affected the state rather than inferring what the state should be.

2) Decoding a quantum code is related to classical "syndrome decoding" (see [27, Ch. 47]) with the caveat that errors differing by a combination of the rows of the parity-check matrix act identically on the codewords. Thus, maximum-likelihood decoding consists in identifying the most likely error coset given the syndrome. The coset with largest probability can differ from the one containing the most likely Pauli error.
3) We cannot assume as in the classical case that the SISO decoder of the convolutional quantum code starts at the zero state and ends at the zero state. This is related to the fact that the memory is described in terms of the Pauli error that has affected the qubits rather than reflecting a property of the encoded state. Instead, we perform a measurement which reveals partial information (the X component) about the first memory element.

Let us now describe how each constituent convolutional code is decoded with a SISO decoder.

A. *Decoding of Convolutional Codes*

As stated in Definition 18, qubitwise maximum likelihood consists in finding the logical operator L_i that maximizes the marginal conditional probability $\mathbf{P}(L_i|S^x)$. We call the algorithm that computes this probability, but without returning the L_i that optimizes it, a SISO decoder. The purpose of this section is to explain how such a decoder can be implemented efficiently for quantum convolutional codes.

We choose to base our presentation solely on the circuit description of the code. Our algorithm is essentially equivalent to a sum–product algorithm operated on the trellis of the code [33]. However, the novelties of quantum codes listed above require some crucial modifications of the trellis-based decoding. We find that these complication are greatly alleviated when decoding is formulated directly in terms of the circuit.

Since the distinction between trellis-based and circuit-based decoding is technical rather than conceptual, we will present the procedure in details and omit its derivation from first principles. As usual, when operated on a memoryless Pauli channel, the whole procedure is nothing but Bayesian updating of probabilities.

Consider a quantum convolutional code with parameters (n, k, m, t), seed transformation U, and duration N as shown in Fig. 6. We use the same notation as in Section IV-A and denote by V the associated encoding matrix. Let us recall that it maps $G_{n(N+t)+m}$ to itself. As in Section IV-A, we decompose an element P in $G_{n(N+t)+m}$ (i.e., an error on the channel) as $P = (P_1 : P_2 : \cdots : P_{N+t})$ where the P_i's belong to G_n for i in $\{1, 2, \ldots, N + t - 1\}$ and P_{N+t} belongs to G_{n+m}. It will be convenient to denote the coordinates of each P_i by P_i^j, i.e., $P_i = \left(P_i^1 : P_i^2 : \cdots : P_i^n\right)$ where the P_i^j's belong to G_1.

Similarly, we will decompose the Pauli-stream obtained by applying the inverse encoder to P as

$$(S_0 : L_1 : S_1 : \cdots : L_N : S_N : S_{N+1} : \cdots : S_{N+t}) \triangleq PV^{-1}$$

where S_0 belongs to G_m, the L_i's all belong to G_k, the S_i's belong to G_{n-k} for i in $\{1, \ldots, N\}$, and the S_{N+j}'s belong to G_n for j in $\{1, \ldots, t\}$. As explained in Section IV-A, the L_i and S_i can be obtained from the P_i via a recursion relation (35)–(37), which uses auxiliary memory variables M_i. This recursive procedure can be understood intuitively from the circuit diagram of Fig. 6. It simply consists in propagating the effective Pauli operator P from the right-hand side to the left-hand side of the circuit. This can be done in $N + t$ steps, each step passing through a single seed transformation U, and the memory variables M_j simply represent the operators acting on the memory qubit between two consecutive seed transformations. The decoding algorithm actually follows the same logic. As explained in Section III-C, the probability on L and S is obtained from the pullback of $\mathbf{P}(P)$ through the encoder V [cf., (29)]. For a convolutional code, this pullback can be decomposed into elementary steps, each step passing through a single seed transformation U and computing intermediate probabilities on the memory variables.

In addition to the procedure just outlined, the decoder must also update the probability $\mathbf{P}(L)$ obtained from the pullback of $\mathbf{P}(P)$ conditioned on the value of the observed syndrome. This operation is slightly more subtle, and requires not only the pullback of probabilities through the circuit, but also their push-forward (propagating from the left-hand side to the right-hand side of the circuit). For that reason, the decoding algorithm presented in Algorithm 1 will consist of three steps, a backward pass (Algorithm 2), a forward pass (Algorithm 3), and a local update (Algorithm 4). As indicated by their names, these, respectively, perform a pullback of probabilities, a push-forward of probabilities, and finally, an operation that combines these two probabilities into the final result.

Algorithm 1: The SISO algorithm for quantum convolutional codes

INPUTS:

$\mathbf{P}(P_i^j)$ for $i \in [N + t]$, $j \in [n]$, (and $j \in [n + m]$ when $i = N + t$) From physical noise model

$\mathbf{P}(L_i^j)$ for $i \in [N]$, $j \in [k]$ From turbo decoder

S^x From syndrome measurement

OUTPUTS:

$\mathbf{P}(P_i^j|S^x)$ for $i \in [N + t]$, $j \in [n]$ (and $j \in [n + m]$ when $i = N + t$),

$\mathbf{P}(L_i^j|S^x)$ for $i \in [N]$, $j \in [k]$

ALGORITHM:

backward pass

forward pass

local update

Algorithm 2: Backward pass

INPUTS:

Same as SISO algorithm

OUTPUTS:

$\mathbf{P}(M_i|S^x_{>i})$ for $i \in [N+t]$.

ALGORITHM:

{ Initialization: $\mathbf{P}(M_{n+t})$ is given directly by the physical noise model. }

for all $\gamma \in G_m$ **do**

$\mathbf{P}(M_{n+t} = \gamma) \leftarrow \Pi_{j=1}^{m} \mathbf{P}(P_{N+t}^{n+j} = \gamma^j)$

end for

{ Recursion: first t steps }

for $i = N+t-1$ to $N+1$ **do**

$$\mathbf{P}(M_i|S^x_{>i}) \propto \sum_{\sigma \in G_n : \sigma^x = S_i^x} \Big[\mathbf{P}(P_{i+1} = (M_i : \sigma) U_P) \times \mathbf{P}(M_{i+1} = (M_i : \sigma) U_M | S^x_{>i+1}) \Big]$$

end for

{ Recursion: last N steps }

for $i = N$ to 1 **do**

$$\mathbf{P}(M_i|S^x_{>i}) \propto \sum_{\substack{\lambda \in G_k \\ \sigma \in G_{n-k} : \sigma^x = S_i^x}} \Big[\mathbf{P}(L_i = \lambda) \mathbf{P}(P_{i+1} = (M_i : \lambda : \sigma) U_P) \times \mathbf{P}(M_{i+1} = (M_i : \lambda : \sigma) U_M | S^x_{>i+1}) \Big]$$

end for

Algorithm 3: Forward pass

INPUTS:

Same as SISO algorithm

OUTPUTS:

$\mathbf{P}(M_i|S^x_{\le i})$ for $i \in \{0, \ldots, N+t-1\}$.

ALGORITHM:

{ Initialization: }

for all $\gamma \in G_m$ **do**

if $\gamma^x = S_0^x$ **then**

$\mathbf{P}(M_0 = \gamma | S_0^x) \leftarrow \frac{1}{2^m}$

else

$\mathbf{P}(M_0 = \gamma | S_0^x) \leftarrow 0$

end if

end for { Recursion: }

for $i = 1$ to $N+t+1$ **do**

$$\mathbf{P}(M_i|S^x_{\le i}) \propto \sum_{\substack{\mu \in G_m, \lambda \in G_k \\ \sigma \in G_{n-k} : \sigma^x = S_i^x \\ M_i = (\mu : \lambda : \sigma) U_M}} \Big[\mathbf{P}(L_i = \lambda) \mathbf{P}(P_i = (\mu : \lambda : \sigma) U_P) \times \mathbf{P}\left(M_{i-1} = \mu | S^x_{\le i-1}\right) \Big]$$

end for

Algorithm 4: Local update

INPUTS:

Same as SISO algorithm

$\mathbf{P}(M_i|S^x_{>i})$ for $i \in [N+t]$ From backward pass

$\mathbf{P}(M_i|S^x_{\le i})$ for $i \in \{0, \ldots, N+t-1\}$ From forward pass

OUTPUTS:

$\mathbf{P}(P_i^j|S^x)$ for $i \in [N+t]$, $j \in [n]$ (and $j \in [n+m]$ for $i = N+t$)

$\mathbf{P}(L_i^j|S^x)$ for $i \in [N]$ and $j \in [k]$

ALGORITHM:

for $i = 1$ to $N+t$ **do**

$$\mathbf{P}(L_i|S^x) \propto \sum_{\substack{\mu \in G_m \\ \sigma \in G_{n-k} : \sigma^x = S_i^x}} \Big[\mathbf{P}(L_i) \mathbf{P}\left(M_{i-1} = \mu | S^x_{\le i-1}\right) \times \mathbf{P}\left(P_i = (\mu : L_i : \sigma) U_P\right) \times \mathbf{P}\left(M_i = (\mu : L_i : \sigma) U_m | S^x_{>i}\right) \Big]$$

$$\mathbf{P}(P_i|S^x) \propto \sum_{\substack{\mu \in G_m, \lambda \in G_k \\ \sigma \in G_{n-k} : \sigma^x = S_i^x \\ P_i = (\mu : \lambda : \sigma) U_P}} \Big[\mathbf{P}(P_i) \mathbf{P}(L_i = \lambda) \mathbf{P}(M_{i-1} = \mu | S^x_{\le i-1}) \times \mathbf{P}\left(M_i = (\mu : \lambda : \sigma) U_M | S^x_{>i}\right) \Big]$$

end for

{ Marginalization: }

Compute $\mathbf{P}(L_i^j|S^x)$ from $\mathbf{P}(L_i|S^x)$

Compute $\mathbf{P}(P_i^j|S^x)$ from $\mathbf{P}(P_i|S^x)$

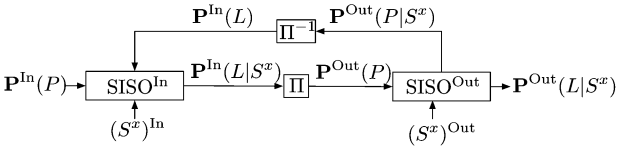

Fig. 11. Information flow in the iterative turbo decoding procedure.

Our description of these algorithms makes use of the following notation:

$$S \triangleq (S_i)_{0\le i\le N+t} \tag{45}$$
$$S_{\le i} \triangleq (S_j)_{0\le j\le i} \tag{46}$$
$$S_{>i} \triangleq (S_j)_{i<j\le N+t} \tag{47}$$

and we denote by U_P the binary matrix formed by the $2n$ first columns of U and by U_M the binary matrix formed by the $2m$ last columns of U. This means that

$$P_i = (M_{i-1} : L_i : S_i)U_P$$
$$M_i = (M_{i-1} : L_i : S_i)U_M$$

where the M_i are defined from (35)–(37). The notation $\mathbf{P}(M_i) \propto \cdots$ means that entries of the vector $((\mathbf{P}(M_i = \mu))_{\mu\in G_m}$ are proportional to the corresponding right-hand side term, the proportionality factor being given by normalization. Finally, for any integer n, we denote $[n] \triangleq \{1, 2, \ldots, n\}$.

B. Turbo Decoder

A turbo code is built from the interleaved serial concatenation of two convolutional codes. The decoding of such a code uses the SISO decoder of its constituent convolutional codes in an iterative way that is schematically illustrated in Fig. 11.

The inner code is first decoded as described above but without any information on the logical random variables: $\mathbf{P}^{\mathrm{In}}(L_i^j)$ is the uniform distribution. The distribution $\mathbf{P}^{\mathrm{In}}(P_i^j)$ is given directly by the channel model. The only output which is used in the following step is the output distribution on the logical variables given the syndrome measured on the inner code: $\mathbf{P}^{\mathrm{In}}(L_i^j|S^x)$ (S^x really refers to the part of the syndrome measured for the inner code and not to the whole syndrome, but we do not attach a "In" to it to avoid cumbersome notation).

Then, the outer code is decoded with the SISO algorithm, using as input distribution for the logical variables, as in the previous case, the uniform distribution. The input distribution of the physical variables $\mathbf{P}^{\mathrm{Out}}(P_i^j)$ is deduced from the logical output distribution of the inner decoder

$$\mathbf{P}^{\mathrm{Out}}(P_{i^\pi}^{j^\pi} = \gamma) = \mathbf{P}^{\mathrm{In}}(L_i^j K_i^j = \gamma|S^x)$$

where i^π and j^π are such that $(i^\pi, j^\pi) = \pi(i, j)$, and the K_i^j are the single-qubit symplectic transformations that appear in the quantum interleaver Π. This yields the output distributions $\mathbf{P}^{\mathrm{Out}}(P_i^j|S^x)$ and $\mathbf{P}(L_i^j|S^x)$ (again, S^x only refers to the part of the syndrome attached to the outer code). This step is terminated by estimating the most likely error coset $\hat{L}$, setting

$$\hat{L}_i^j = \mathrm{argmax}_\gamma \left\{\mathbf{P}^{\mathrm{Out}}(L_i^j = \gamma|S^x)\right\}.$$

To iterate this procedure, use the output probability $\mathbf{P}^{\mathrm{Out}}(P_i^j|S^x)$ as information on the logical variables of the inner code: in other words, set as input distribution for inner SISO decoding

$$\mathbf{P}^{\mathrm{In}}(L_i^j = \gamma) = \mathbf{P}^{\mathrm{Out}}(P_{i^\pi}^{j^\pi} = \gamma K_{j^\pi}^{i^\pi}|S^x)$$

and the distribution of the physical variables is set by the physical channel as before. This is represented by the feedback loop in Fig. 11 where information from the outer decoder is returned to the inner decoder.

This procedure can be repeated an arbitrary number of times, with each iteration yielding an estimate of the maximum-likelihood decoder of the outer code. The iterations can be halted after a fixed number of rounds, or when the estimate does not vary from one iteration to the next. Although the decoding scheme is exact for both constituent codes, the overall turbo decoding is suboptimal. The reason for this is that although $\mathbf{P}^{\mathrm{In}}(P)$ is memoryless, the induced channel $\mathbf{P}^{\mathrm{Out}}(P) =$ on the outer code obtained from $\mathbf{P}^{\mathrm{Out}}(P_{i^\pi}^{j^\pi}) = \mathbf{P}^{\mathrm{In}}(L_i^j K_i^j|S^x)$ is not. The decoder ignores this fact and only uses the marginals $\mathbf{P}^{\mathrm{Out}}(P_i^j)$ of $\mathbf{P}^{\mathrm{Out}}(P)$. This is the price to pay for an efficient decoding algorithm.

VI. Results

The convolutional codes we used for our construction of turbo codes are for the most part generated at random. That is, we first generate a random seed transformation U of desired dimensions. Using its state diagram, we then test whether the corresponding encoder is catastrophic, and if so we reject it and start over. Noncatastrophicity is the only criterion that we systematically imposed.

As a first sieve among the randomly generated noncatastrophic seed transformations, we can study their distance spectrums and make some heuristic test based on it. Examples of good seed transformations obtained from this procedure are $U_{(3,1,3)} = \{2085,926,2053,1434,910,3943,1484,2881,3212,2250,68,331\}$, $U_{(3,1,4)} = \{13159,10335,13127,6554,10319,14441,10625,5835,832,13893,11916,11329,8204,5570\}$, and $U_{(2,1,4)} = \{610,3323,760,1591,2500,942,2290,794,1535,2202,2859,809\}$, where the binary symplectic encoding matrix is specified by its list of rows and each row is given by the integer corresponding to the binary entry. The subscripts on the encoders specify its parameters (n, k, m). Hence, the first two codes have rate $\frac{1}{3}$ but differ by the size of their memory. The third code has a higher rate of $\frac{1}{2}$. The first few values of the distance spectrum of logical-weight-one codewords for these quantum convolutional code are given in Table I, while the distance spectrum of all codewords are listed in Table II.

Based on those values, we conclude that the turbo codes obtained from concatenation of a code using seed transformation

TABLE I
DISTANCE SPECTRUM $F_1(w)$ OF LOGICAL-WEIGHT-ONE CODEWORDS

w	$U_{(3,1,3)}$	$U_{(3,1,4)}$	$U_{(2,1,4)}$
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	2	0	0
7	4	3	0
8	8	0	2
9	16	7	0
10	35	0	3
11	70	34	2
12	143	0	0
13	295	156	2
14	634	0	10
15	1 362	586	12
16	2 802	0	37
17	5 714	2 827	38
18	11 526	0	121
19	23 674	11 430	86
20	48 817	0	280

TABLE II
DISTANCE SPECTRUM $F(w)$

w	$U_{(3,1,4)}$	$U_{(3,1,4)}$	$U_{(2,1,4)}$
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0
4	1	0	0
5	11	0	6
6	47	11	82
7	265	70	442
8	1 275	324	3 379
9	6 397	1 596	24 074
10	31 785	7 773	174 997
11	160 311	40 971	1 253 748
12	801 232	206 959	9 033 087

$U_{(3,1,4)}$ with themselves have a minimal distance no greater than $6 \times 4 = 24$. Similarly, the codes obtained by the concatenation of $U_{(3,1,4)}$ with themselves have minimal distance no greater than $7 \times 6 = 42$, and the code obtained from the concatenation of $U_{(2,1,4)}$ with itself has $8 \times 6 = 48$.

These are upper bounds on the minimal distance and do not translate directly into the performance of the code. On the one hand, the actual minimal distance of a turbo code depends on the interleaver, which we chose completely at random. In all cases, there are most likely lower weight codewords than the estimate provided by these lower bounds, but those are atypical. On the other hand, the codes are not decoded with a minimum distance decoder, so even a true large minimal distance does not imply low WER.

The WER of a quantum turbo code on a depolarization channel can be estimated using Monte Carlo methods. An error $P \in G_N$ is generated randomly according to the channel model probability distribution. The syndrome associated to this error is evaluated, and based on its value, the decoding algorithm (see Section V) is executed. The decoding algorithm outputs an error estimate P'. If $P - P' \in C(I)$, the decoding is accepted, otherwise it is rejected. In other words, the decoding is accepted only if all K encoded qubits are correctly recovered. The WER is then the fraction of rejected decodings.

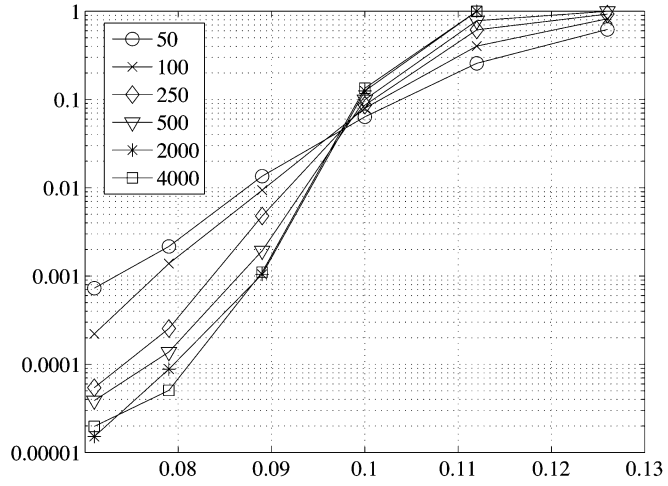

Fig. 12. WER versus depolarizing probability p for the quantum turbo code obtained from the concatenation of the convolutional code with seed transformation $U_{(3,1,3)}$ with itself, for different number of encoded qubits K. Each constituent convolutional code has $m = 3$ qubits of memory and has rate $\frac{1}{3}$, so the rate of the turbo code is $\frac{1}{9}$.

The WERs as a function of the depolarizing probability p are shown for a selection of codes in Figs. 12–14. Perhaps the most striking feature of those curves is the existence of a pseudothreshold value of p below which the WER *decreases* as the number of encoded qubits increases. Since the codes have a bounded minimal distance, this is not a true threshold in the sense that as we keep increasing the number of encoded qubits, the WER should start to increase. However, we see that for modest sizes K of up to 4000, this effect is not observed. We do see, however, that the improvement appears to be saturating around these values. The pseudothreshold is particularly clear for the seed transformation $U_{(3,1,3)}$, where it is approximately 0.098, and for the seed transformation $U_{(2,1,4)}$, where it is approximately 0.067. Its value for the seed transformation $U_{(3,1,4)}$ is not as clear, but seams to be between 0.95 and 0.11.

These values should be compared with the hashing bound, whose value is approximately 0.16024 for a rate $\frac{1}{9}$ code and 0.12689 for rate $\frac{1}{4}$. We can also compare with the results obtained from LDPC codes in [28, Fig. 10], by evaluating the depolarizing probability p at which the WER drops below 10^{-4}. For a rate $\frac{1}{4}$, this threshold was achieved at $p_{th} \approx 0.033$ (note the convention $f_m = \frac{2}{3}p$) for LDPC codes while the turbo code shown in Fig. 14 has $p_{th} \approx 0.048$. It should also be noted that this improved threshold is achieved with a smaller block size than that used for the LDPC in [28]; a larger block should further improve this result.

As expected, changing the rate of the code directly affects the value of the pseudothreshold. This is seen by comparing either Figs. 12 or 13 to Fig. 14. The effect of the memory size is however less obvious. Comparing Figs. 12 and 13, it appears that the effect of a larger memory is to sharpen the slope of the WER profile below the pseudothreshold for fixed K. In other words, the main impact of the memory size is not in the value of the pseudothreshold, but rather in the effectiveness of the error suppression below that threshold. This conclusion is somewhat

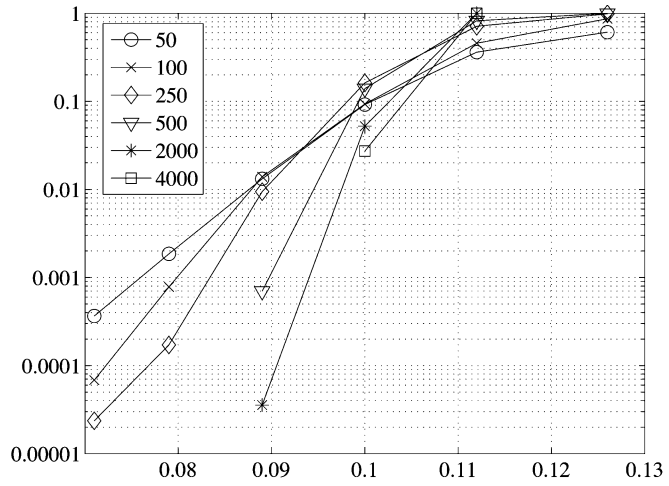

Fig. 13. WER versus depolarizing probability p for the quantum turbo code obtained from the concatenation of the convolutional code with seed transformation $U_{(3,1,4)}$ with itself, for different number of encoded qubits K. Each constituent convolutional code has $m = 4$ qubits of memory and has rate $\frac{1}{3}$, so the rate of the turbo code is $\frac{1}{9}$.

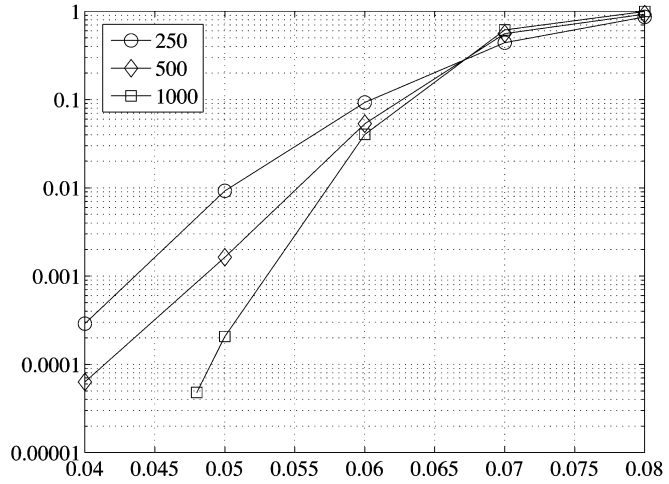

Fig. 14. WER versus depolarizing probability p for the quantum turbo code obtained from the concatenation of the convolutional code with seed transformation $U_{(2,1,4)}$ with itself, for different number of encoded qubits K. Each constituent convolutional code has $m = 4$ qubits of memory and has rate $\frac{1}{2}$, so the rate of the turbo code is $\frac{1}{4}$.

supported by Fig. 15 where the WER is plotted for a variety of memory configurations. In all cases, the slope of the WER increases with the memory size.

VII. Conclusion

In this paper, we have presented a detailed theory of quantum serial turbo codes based on the interleaved serial concatenation of quantum convolutional codes. The description and the analysis of these codes were greatly simplified by the use of a circuit representation of the encoder. In particular, this representation provides a simple definition of the state diagram associated to a quantum convolutional code, and enables a simple and intuitive derivation of their efficient decoding algorithm.

By a detailed analysis of the state diagram, we have shown that all recursive convolutional encoders have catastrophic

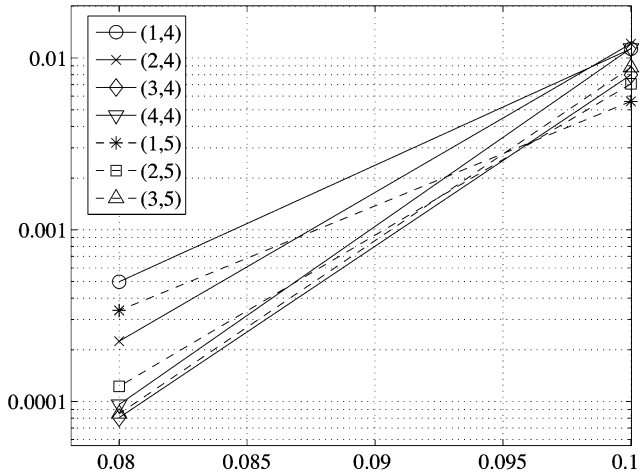

Fig. 15. WER versus depolarizing probability p for a quantum turbo code encoding $K = 100$ qubits and rate $\frac{1}{9}$ with different memory configurations $(m^{\mathrm{In}}, m^{\mathrm{Out}})$.

error propagation. Recursive convolutional encoders can be constructed and yield serial turbo codes with polynomial minimal distances. However, they offer extremely poor iterative decoding performances due to their unavoidable catastrophic error propagation. The encoders we have used in our constructions are thus chosen to be noncatastrophic and nonrecursive. While the resulting codes have bounded minimal distance, we have found that they offer good iterative decoding performances over a range of block sizes and word error rates that are of practical interest.

Compared to quantum LDPC codes, quantum turbo codes offer several advantages. On the one hand, there is complete freedom in the code design in terms of length, rate, memory size, and interleaver choice. The freedom in the interleaver is crucial since it is the source of the randomness that is responsible for the success of these codes. On the other hand, the graphical representation of turbo codes is free of 4-cycles that deteriorate the performances of iterative decoding. Finally, the iterative decoder makes explicit use of the code's degeneracy. This feature is important because turbo codes, like LDPC codes, have low-weight stabilizers and are hence greatly degenerated.

In future work, we hope to surmount the obstacle of catastrophic error propagation. A concrete avenue is the generalized stabilizer formalism of operator quantum error correction [34], which could circumvent the conclusions of our theorem established in the context of subspace stabilizer codes. Doping [45] is an other possibility that we will investigate.

Appendix

To prove Lemma 2, we first establish some simple facts.

Fact 4: The subspace of $\mathbb{F}_2^{2n+2m}$ orthogonal to all the rows of U_{eff} is the space spanned by the rows of its submatrix $[\Sigma_{\mathrm{P}} : \Sigma_{\mathrm{M}}]$. Similarly, the subspace of $\mathbb{F}_2^{2n+2m}$ orthogonal to all the rows of $\begin{bmatrix} \mu_{\mathrm{P}} & \mu_{\mathrm{M}} \\ \Sigma_{\mathrm{P}} & \Sigma_{\mathrm{M}} \end{bmatrix}$ is the space spanned by the rows of $\begin{bmatrix} \Lambda_{\mathrm{P}} & \Lambda_{\mathrm{M}} \\ \Sigma_{\mathrm{P}} & \Sigma_{\mathrm{M}} \end{bmatrix}$.

Proof: The subspace V of $\mathbb{F}_2^{2n+2m}$ orthogonal to all the rows of U_{eff} is of dimension $2n+2m-(2m+2k+n-k) = n-k$. We observe now that the rows of $[\Sigma_{\mathrm{P}} : \Sigma_{\mathrm{M}}]$ are all independent and all orthogonal to the rows of U_{eff}. They form, therefore, a basis of V. This finishes the proof of the first statement. The second one is obtained by similar arguments. □

Proof (of Lemma 2): Let $M' \in \mathbb{F}_2^{2m}$ be such that there exist $M \in \mathbb{F}_2^{2m}$, $S \in \mathbb{F}_2^{n-k}$, and $L \in \mathbb{F}_2^{2k}$ such that $(M : L : S)U_{\text{eff}} = (\mathbf{0}_{2n} : M')$. Notice now that $(\mathbf{0}_{2n} : M')$ is spanned by the rows of U_{eff} and is, therefore, orthogonal to all the rows of the matrix $[\Sigma_{\mathrm{P}} : \Sigma_{\mathrm{M}}]$. This implies that M' belongs to $\mathfrak{S}^{\perp}$. Conversely, any row vector of the form $(\mathbf{0}_{2n} : M')$ with M' belonging to $\mathfrak{S}^{\perp}$ is orthogonal to all the rows of $[\Sigma_{\mathrm{P}} : \Sigma_{\mathrm{M}}]$ and is, therefore, spanned by the rows of U_{eff}. This implies that there exist $M \in \mathbb{F}_2^{2m}$, $S \in \mathbb{F}_2^{n-k}$, and $L \in \mathbb{F}_2^{2k}$ such that $(M : L : S)U_{\text{eff}} = (\mathbf{0}_{2n} : M')$. Furthermore, it can be noticed from the fact that the rows of U_{eff} are independent, that if such an $(M : L : S)$ exists, it is unique. □

The proof of Lemma 3 requires a straightforward Fact and a Lemma.

Fact 5: For any $M, M' \in \mathbb{F}_2^{2m}$, we have

$$(M\mu_{\mathrm{P}} : M\mu_{\mathrm{M}}) \star (M'\mu_{\mathrm{P}} : M'\mu_{\mathrm{M}}) = M \star M'.$$

Proof: This is a straightforward consequence of the orthogonality relations satisfied by the first $2m$ rows of U. □

Lemma 8: Let $T \in \mathbb{F}_2^{2m}$ and let M' be such that there exist $M \in \mathbb{F}_2^{2m}$, $S \in \mathbb{F}_2^{n-k}$, and $L \in \mathbb{F}_2^{2k}$ such that $(M : L : S)U_{\text{eff}} = (\mathbf{0}_{2n} : M')$. We have

$$M' \star T\mu_{\mathrm{M}} = M \star T. \tag{48}$$

Proof: We observe that

$$\begin{aligned} M' \star T\mu_{\mathrm{M}} &= (\mathbf{0}_{2n} : M') \star (T\mu_{\mathrm{P}} : T\mu_{\mathrm{M}}) \\ &= (M\mu_{\mathrm{P}} + L\Lambda_{\mathrm{P}} + S\Sigma_{\mathrm{P}} \\ &\qquad : M\mu_{\mathrm{M}} + L\Lambda_{\mathrm{M}} + S\Sigma_{\mathrm{M}}) \star (T\mu_{\mathrm{P}} : T\mu_{\mathrm{M}}) \\ &= (M\mu_{\mathrm{P}} : M\mu_{\mathrm{M}}) \star (T\mu_{\mathrm{P}} : T\mu_{\mathrm{M}}) \end{aligned}$$

where the last equation follows from the fact that any row of $[\Lambda_{\mathrm{P}} : \Lambda_{\mathrm{M}}]$ or $[\Sigma_{\mathrm{P}} : \Sigma_{\mathrm{M}}]$ is orthogonal to all the rows of $[\mu_{\mathrm{P}} : \mu_{\mathrm{M}}]$. From this, we conclude that

$$M' \star T\mu_{\mathrm{M}} = M \star T. \qquad \square$$

Proof (of Lemma 3): Since $M' \in \mathfrak{S}^{\perp}$, there exist by Lemma 2, $M \in \mathbb{F}_2^{2m}$, $S \in \mathbb{F}_2^{n-k}$, and $L \in \mathbb{F}_2^{2k}$ such that $(M : L : S)U_{\text{eff}} = (\mathbf{0}_{2n} : M')$. Let $T \in \mathfrak{S}_0$. Using Lemma 8, we obtain

$$M' \star T\mu_{\mathrm{M}} = M \star T.$$

Notice now that $M' \star T\mu_{\mathrm{M}} = 0$ since $T\mu_{\mathrm{M}} \in \mathfrak{S}_0$. From this, $M \star T = 0$. This shows that M belongs to $\mathfrak{S}_0^{\perp}$ as well. The unicity of M is a consequence of Lemma 2. □

The following lemma is used in the proof of Lemma 4.

Lemma 9: Let μ be a linear mapping from $\mathbb{F}_2^{2m}$ to itself. Let V be a subspace of $\mathbb{F}_2^{2m}$ such that $\mu(V) \subset V$ and which contains the null space of any positive power of μ. Then, for any M in $V^{\perp}$, there exists M' in $V^{\perp}$ such that for any T in $\mathbb{F}_2^{2m}$

$$M \star T = M' \star T\mu.$$

Proof: We are first going to prove this statement in the case

$$V = \bigcup_{t=1}^{\infty} \mathrm{Null}(\mu^t).$$

This is a subspace of $\mathbb{F}_2^{2m}$ since the $\mathrm{Null}(\mu^t)$'s are nested sets

$$\mathrm{Null}(\mu) \subset \mathrm{Null}(\mu^2) \subset \cdots \subset \mathrm{Null}(\mu^t) \subset \cdots.$$

Let us consider the space $\mathrm{Im}(\mu^t)$ generated by the rows of μ^t. Since $\mathbb{F}_2^{2m} \supset \mathrm{Im}(\mu) \supset \mathrm{Im}(\mu^2) \supset \cdots$, there must exist a positive t such that $\mathrm{Im}(\mu^t)) = \mathrm{Im}(\mu^{t+1})$. In this case, $\mu(\mathrm{Im}(\mu^t)) = \mathrm{Im}(\mu^t)$. This implies that the restriction of μ to $\mathrm{Im}(\mu^t)$ is a one-to-one mapping and that $\mathrm{Null}(\mu^t) \cap \mathrm{Im}(\mu^t) = \{\mathbf{0}_{2m}\}$. Since $\dim(\mathrm{Null}(\mu^t)) + \dim(\mathrm{Im}(\mu^t)) = 2m$, we can form a basis $(T_1, \ldots T_l, T_{l+1}, \ldots, T_{2m})$ of $\mathbb{F}_2^{2m}$ such that $(T_1, \ldots, T_l)$ spans $\mathrm{Null}(\mu^t)$ and $(T_{l+1}, \ldots, T_{2m})$ spans $\mathrm{Im}(\mu^t)$. Moreover, all the $\mathrm{Null}(\mu^v)$'s are equal for v greater than or equal to t. This follows directly from the fact that the $\mathrm{Im}(\mu^v)$'s are all equal in this case. This can be checked by using the relations $\dim(\mathrm{Null}(\mu^v)) + \dim(\mathrm{Im}(\mu^v)) = 2m$. From these equalities, we deduce that $\dim(\mathrm{Null}(\mu^t)) = \dim(\mathrm{Null}(\mu^{t+1})) = \cdots = \dim(\mathrm{Null}(\mu^v)) = \cdots$. The $\mathrm{Null}(\mu^v)$'s are nested sets, and therefore, $\mathrm{Null}(\mu^t) = \mathrm{Null}(\mu^{t+1}) = \cdots = \mathrm{Null}(\mu^v) = \cdots$. This implies that $V = \mathrm{Null}(\mu^t)$. We define U_i for i in $\{l+1, \ldots, 2m\}$ as the unique element in $\mathbb{F}_2^{2m}$ such that $U_i\mu = T_i$. There exists a unique M' such that

$$M' \star T_i = 0, \qquad \text{for } i \in \{1, \ldots, l\} \tag{49}$$

$$M' \star T_i = M \star U_i, \qquad \text{for } i \in \{l+1, \ldots, 2m\}. \tag{50}$$

This M' belongs to $V^{\perp}$ by (49). Note now that we have defined M' in such a way that $M \star T$ coincides with $M' \star T\mu$ over the basis $(T_1, \ldots, T_l, U_{l+1}, \ldots, U_{2m})$. Therefore, by linearity of the $\star$ product, we have $M \star T = M' \star T\mu$ for all T in $\mathbb{F}_2^{2m}$.

The general case is a direct consequence of this particular case. We define M' similarly by (49) and (50) and it is readily checked that M' belongs to $V^{\perp}$. □

Proof (of Lemma 4): We know from Lemma 3 that for any element M' in $\mathfrak{S}_0^{\perp}$, there exists a unique M in $\mathfrak{S}_0^{\perp}$ such that there is an edge of zero physical weight in the state diagram which goes from M to M'. To prove that the kernel graph has constant in-degree 1, we just have to prove that when M' belongs to the subset $\mathfrak{V}_0^{\perp}$ of $\mathfrak{S}_0^{\perp}$ the corresponding M also belongs to this subset. Since for any $T \in \mathfrak{N}_0$, we have $M' \star T = 0$ and since $\mathfrak{N}_0$ is stable by applying μ_{M} to the left, we obtain for such a T, $M \star T = M' \star T\mu_{\mathrm{M}} = 0$. This shows that M also belongs to $\mathfrak{N}_0^{\perp}$ which shows that M belongs to $\mathfrak{V}_0^{\perp}$.

On the other hand, by applying Lemma 9 with $V = \mathfrak{V}_0$, we know that for any vertex M of the kernel graph, there is an M' belonging also to $\mathfrak{V}_0^{\perp}$ such that for any T in $\mathbb{F}_2^{2m}$

$$M \star T = M' \star T\mu_{\mathrm{M}}.$$

Note that given such an M' there is a unique M which satisfies the aforementioned equality for all T. Therefore, M is necessarily the starting vertex of the unique directed edge of physical weight 0 having as endpoint M'. □

Proof (of Lemma 5): We just have to prove that the set $\mathfrak{V}_0$ is not equal to the whole space $\mathbb{F}_2^{2m}$. We proceed by contradiction. Assume that $\mathfrak{V}_0 = \mathbb{F}_2^{2m}$. Notice now that there exists a finite number t such that

$$\mathfrak{V}_0 = \mathrm{Null}\left(\mu_{\mathrm{M}}^t\right) + \sum_{i=0}^{t} \mathfrak{S}\mu_{\mathrm{M}}^i.$$

For such a t, any M in $\mathbb{F}_2^{2m}$ can be expressed as a sum $M = N + \sum_{i=0}^{t} T_i\mu_{\mathrm{M}}^i$, where N is in $\mathrm{Null}(\mu_{\mathrm{M}}^t)$ and the T_i's all belong to $\mathfrak{S}$, i.e., they are of the form $T_i = S_i\Sigma_{\mathrm{M}}$ for some $S_i \in \mathbb{F}_2^{n-k}$. Consider now a finite path starting at the origin with logical weight 1 and nonzero physical weight. We denote by M its endpoint (which is viewed as an element in $\mathbb{F}_2^{2m}$). We decompose $M\mu_{\mathrm{M}}^{t+1}$ as explained before

$$M\mu_{\mathrm{M}}^{t+1} = N + \sum_{i=0}^{t} S_{t-i}\Sigma_{\mathrm{M}}\mu_{\mathrm{M}}^i$$

where the S_i's belong to $\mathbb{F}_2^k$. The path of length t which starts at M and which corresponds to the sequence of pairs of logical transformations/stabilizer transformations $(\mathbf{0}_{2k} : S_0) \rightarrow (\mathbf{0}_{2k} : S_1) \rightarrow \cdots \rightarrow (\mathbf{0}_{2k} : S_t)$ will go from point M to

$$M\mu_{\mathrm{M}}^{t+1} + \sum_{i=0}^{t} S_{t-i}\Sigma_{\mathrm{M}}\mu_{\mathrm{M}}^i = N.$$

By extending this path by feeding in t zero transformations $(\mathbf{0}_{2k} : \mathbf{0}_{n-k})$, we go from vertex N to $\mu^t(N)$, which is equal to $\mathbf{0}_{2m}$ by definition. This path may then continue by feeding in additional zero transformations and will stay at the zero vertex forever. This contradicts the fact that the quantum code is recursive. □

Proof (of Lemma 6): Let M' be an element of $\mathbb{F}_2^{2m}$ for which there exist $M \in \mathbb{F}_2^{2m}$ and $S \in \mathbb{F}_2^{n-k}$, such that $(M : \mathbf{0}^{2k} : S)U_{\mathrm{eff}} = (\mathbf{0}_{2n} : M')$. $(\mathbf{0}_{2n} : M')$ is spanned by the rows of $[\mu_{\mathrm{P}} : \mu_{\mathrm{M}}]$ and $[\Sigma_{\mathrm{P}} : \Sigma_{\mathrm{M}}]$. By Fact 5, this implies that $(\mathbf{0}_{2n} : M')$ is orthogonal to all the rows of the matrices $[\Lambda_{\mathrm{P}} : \Lambda_{\mathrm{M}}]$ and $[\Sigma_{\mathrm{P}} : \Sigma_{\mathrm{M}}]$. Hence, M' should belong to $\mathfrak{L}^{\perp}$. On the other hand, any $(\mathbf{0}_{2n} : M')$ for which M' belongs to $\mathfrak{L}^{\perp}$ is orthogonal to all the rows of $[\Lambda_{\mathrm{P}} : \Lambda_{\mathrm{M}}]$ and $[\Sigma_{\mathrm{P}} : \Sigma_{\mathrm{M}}]$ and is, therefore, spanned by the rows of $[\mu_{\mathrm{P}} : \mu_{\mathrm{M}}]$ and $[\Sigma_{\mathrm{P}} : \Sigma_{\mathrm{M}}]$. □

Proof (of Lemma 7): This amounts to proving that there exists a vertex in the kernel graph which does not belong to $\mathfrak{L}^{\perp}$. The set of vertices of the kernel graph is $\mathfrak{V}_0^{\perp}$. Therefore, we need to find an element of $\mathfrak{L}$ that is not in $\mathfrak{V}_0$. In particular, we would be done if there existed a row of Λ_{M} which did not belong to $\mathfrak{V}_0$.

Assume the opposite. Let t be the integer such that $\mathfrak{V}_0 = \mathrm{Null}(\mu_{\mathrm{M}}^t) + \sum_{i=0}^{t} \mathfrak{S}\mu_{\mathrm{M}}^i$. Then, for every L in $\mathbb{F}_2^{2m}$ of weight 1 and any integer k, there exists $S_0, S_1, \ldots, S_t$ in $\mathbb{F}_2^{n-k}$ and an N in $\mathrm{Null}(\mu_{\mathrm{M}}^t)$ such that

$$L\Lambda_{\mathrm{M}}\mu_{\mathrm{M}}^k = N + \sum_{i=0}^{t} S_{t-i}\Sigma_{\mathrm{M}}\mu_{\mathrm{M}}^i.$$

Consider a finite path of nonzero physical weight and logical weight 1 starting at the origin and ending at a vertex M. Assume that this path corresponds to the sequence of pairs of logical/stabilizer inputs

$$(\mathbf{0}_{2k} : S_0) \rightarrow (\mathbf{0}_{2k} : S_1) \rightarrow \cdots \rightarrow (\mathbf{0}_{2k} : S_{i-1}) \rightarrow (L : S_i) \rightarrow (\mathbf{0}_{2k} : S_{i+1}) \rightarrow \cdots \rightarrow (\mathbf{0}_{2k} : S_u)$$

(i.e., the only time where the logical transformation is nonzero is at time i and is equal to L which is assumed to be of weight 1). The final memory state would then be

$$M = L\Lambda_{\mathrm{M}}\mu_{\mathrm{M}}^{u-i} + \sum_{i=0}^{u} S_{u-i}\Sigma_{\mathrm{M}}\mu_{\mathrm{M}}^i. \tag{51}$$

Since, by assumption, the rows of Λ_{M} are in $\mathfrak{V}_0$, there exists $S_0', \ldots, S_t'$ in $\mathbb{F}_2^{n-k}$ and N' in $\mathrm{Null}(\mu_{\mathrm{M}}^t)$ such that

$$L\Lambda_{\mathrm{M}}\mu_{\mathrm{M}}^{u+t+1-i} + \sum_{i=0}^{u} S_{u-i}\Sigma_{\mathrm{M}}\mu_{\mathrm{M}}^{i+t+1} = N' + \sum_{i=0}^{t} S'_{t-i}\Sigma_{\mathrm{M}}\mu_{\mathrm{M}}^i. \tag{52}$$

Thus, if we extend the path by the sequence of inputs

$$(\mathbf{0}_{2k} : S_0') \rightarrow (\mathbf{0}_{2k} : S_1') \rightarrow \cdots \rightarrow (\mathbf{0}_{2k} : S_t')$$

we arrive at the vertex M' which satisfies

$$\begin{aligned} M' &= M\mu_{\mathrm{M}}^{t+1} + \sum_{i=0}^{t} S'_{t-i}\Sigma_{\mathrm{M}}\mu_{\mathrm{M}}^i \\ &= L\Lambda_{\mathrm{M}}\mu_{\mathrm{M}}^{u+t+1-i} + \sum_{i=0}^{u} S_{u-i}\Sigma_{\mathrm{M}}\mu_{\mathrm{M}}^{i+t+1} + \sum_{i=0}^{t} S'_{t-i}\Sigma_{\mathrm{M}}\mu_{\mathrm{M}}^i \\ &= N'. \end{aligned}$$

Extending this whole sequence by adding t zero transformations $(\mathbf{0}_{2k} : \mathbf{0}_{n-k})$ will bring this path back to the origin since N' is in the kernel of μ_{M}^t. Once at the origin, then encoder can remain in that state forever without any additional physical output. This implies that the code is nonrecursive, and completes the proof. □

REFERENCES

[1] S. A. Aly, A. Klappenecker, and P. K. Sarvepalli, "On quantum and classical BCH codes," *IEEE Trans. Inf. Theory*, vol. 53, no. 3, pp. 1183–1188, Mar. 2007.

[2] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: performance analysis, design, and iterative decoding," *IEEE Trans. Inf. Theory*, vol. 44, no. 3, pp. 909–926, May 1998.

[3] C. H. Bennett, D. P. DiVincenzo, and J. A. Smolin, "Capacities of quantum erasure channels," *Phys. Rev. Lett.*, vol. 78, pp. 3217–3220, 1997.
[4] C. H. Bennett, D. P. DiVincenzo, J. A. Smolin, and W. K. Wootters, "Mixed state entanglement and quantum error-correcting codes," *Phys. Rev. A, Gen. Phys.*, vol. 54, pp. 3824–3851, 1996.
[5] E. R. Berlekamp, R. J. McEliece, and H. v. Tilborg, "On the inherent intractability of certain coding problems," *IEEE Trans. Inf. Theory*, vol. IT-24, no. 3, pp. 384–386, May 1978.
[6] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding," in *Proc. Int. Conf. Commun.*, Genève, Switzerland, May 1993, pp. 1064–1070.
[7] R. C. Bose and D. K. Ray-Chaudhuri, "On a class of error-correcting binary group codes," *Inf. Control*, vol. 3, pp. 68–79, 1960.
[8] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Phys. Rev. A, Gen. Phys.*, vol. 54, pp. 1098–1105, 1996.
[9] T. Camara, H. Ollivier, and J.-P. Tillich, "A class of quantum LDPC codes: Construction and performances under iterative decoding," in *Proc. Int. Symp. Inf. Theory*, Nice, France, Jun. 2007, pp. 811–815.
[10] H. F. Chau, "Good quantum convolutional error correction codes and their decoding algorithm exist," 1998 [Online]. Available: quant-ph/9806032
[11] H. F. Chau, "Quantum convolutional correcting codes," *Phys. Rev. A*, vol. 58, pp. 905–909, 1998.
[12] I. Devetak, "The private classical capacity and quantum capacity of a quantum channel," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 44–55, Jan. 2005.
[13] D. P. DiVincenzo and P. Aliferis, "Effective fault-tolerant quantum computation with slow measurements," *Phys. Rev. Lett.*, vol. 98, p. 020501, 2007.
[14] D. P. DiVincenzo, P. W. Shor, and J. A. Smolin, "Quantum-channel capacity of very noisy channels," *Phys. Rev. A, Gen. Phys.*, vol. 57, pp. 830–839, 1998.
[15] J. 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.
[16] R. G. Gallager, *Low Density Parity Check Codes*. Cambridge, MA: MIT Press, 1963.
[17] D. Gottesman, "Stabilizer codes and quantum error correction," Ph.D. dissertation, Phys., Math, Astronom. Dept., California Inst. Technol., Pasadena, CA, 1997.
[18] M. Grassl and M. Rötteler, "Non-catastrophic encoders and encoder inverses for quantum convolutional codes," in *Proc. IEEE Int. Symp. Inf. Theory*, 2006, pp. 1109–1113.
[19] M. Hagiwara and H. Imai, "Quantum quasi-cyclic LDPC codes," 2007 [Online]. Available: quant-ph0701020
[20] M. Hastings, "Quantum belief propagation: An algorithm for thermal quantum systems," *Phys. Rev. B, Condens. Matter*, vol. 76, 2007, 201102.
[21] A. Hocquenghem, "Codes correcteurs d'erreurs," *Chiffres*, vol. 2, pp. 147–156, 1959.
[22] N. Kahale and R. Urbanke, "On the minimum distance of parallel and serially concatenated codes," in *Proc. IEEE Int. Symp. Inf. Theory*, 1998, p. 31.
[23] E. Knill and R. Laflamme, "Theory of quantum error-correcting codes," *Phys. Rev. A, Gen. Phys.*, vol. 55, pp. 900–911, 1997.
[24] C. Laumann, A. Scardicchio, and S. Sondhi, "Cavity method for quantum spin glasses on the Bethe lattice," 2007 [Online]. Available: arXiv:0706.4391
[25] M. Leifer and D. Poulin, "Quantum graphical models and belief propagation," *Ann. Phys.*, vol. 323, pp. 1899–1946, 2007.
[26] S. Lloyd, "Capacity of the noisy quantum channel," *Phys. Rev. A, Gen. Phys.*, vol. 55, pp. 1613–1622, 1997.
[27] D. J. C. MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2003.
[28] D. J. C. MacKay, G. Mitchison, and P. L. McFadden, "Sparse graph codes for quantum error-correction," *IEEE Trans. Inf. Theory*, vol. 50, no. 10, pp. 2315–2330, Oct. 2004.
[29] M. Mézard and A. Montanari, *Constraint Satisfaction Networks in Physics and Computation*. Oxford, U.K.: Clarendon, 2007.
[30] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge, U.K.: Cambridge Univ. Press, 2000.
[31] H. Ollivier and J.-P. Tillich, "Description of a quantum convolutional code," *Phys. Rev. Lett.*, vol. 91, 2003, 177902.
[32] H. Ollivier and J.-P. Tillich, "Quantum convolutional codes: Fundamentals," 2004 [Online]. Available: quant-ph/0401134
[33] H. Ollivier and J.-P. Tillich, "Trellises for stabilizer codes: definition and uses," *Phys. Rev. A, Gen. Phys.*, vol. 74, 2006, 032304.
[34] D. Poulin, "Stabilizer formalism for operator quantum error correction," *Phys. Rev. Lett.*, vol. 95, 2005, 230504.
[35] D. Poulin, "Optimal and efficient decoding of concatenated quantum block codes," *Phys. Rev. A, Gen. Phys.*, vol. 74, 2006, 052333.
[36] D. Poulin and E. Bilgin, "Belief propagation algorithm for computing correlation functions in finite-temperature quantum many-body systems on loopy graphs," *Phys. Rev. A, Gen. Phys.*, vol. 77, 2008, 52318.
[37] D. Poulin and Y.-J. Chung, "On the iterative decoding of sparse quantum codes," *Quant. Inf. Comput.*, vol. 8, pp. 986–1000, 2008.
[38] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. SIAM*, vol. 8, pp. 300–304, 1960.
[39] C. E. Shannon, "A mathematical theory of communication," *Bell J. Syst. Tech.*, vol. 27, pp. 379–423, 1948.
[40] P. Shor, in *Proc. MSRI Workshop Quant. Comput.*, 2002 [Online]. Available: http://www.msri.org/publications
[41] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Phys. Rev. A, Gen. Phys.*, vol. 52, pp. 2493–2496, 1995.
[42] G. Smith and J. A. Smolin, "Degenerate coding for Pauli channels," 2006 [Online]. Available: quant-ph/0604107
[43] A. M. Steane, "Error correcting codes in quantum theory," *Phys. Rev. Lett.*, vol. 77, pp. 793–797, 1996.
[44] A. M. Steane, "Simple quantum error correcting codes," *Phys. Rev. A, Gen. Phys.*, vol. 54, pp. 4741–4751, 1996.
[45] S. ten Brink, "Designing iterative decoding schemes with the extrinsic information transfer chart," *AEU Int. J. Electron. Commun.*, vol. 54, pp. 389–398, 2000.
[46] J. S. Yedidia, "An idiosyncratic journey beyond mean field theory," in *Advanced Mean Field Methods: Theory and Practice*. Cambridge, MA: MIT Press, 2001, p. 21.

David Poulin received the Ph.D. degree in physics from the University of Waterloo, Waterloo, ON, Canada, in 2004.

He has been a Postdoctoral Fellow at The University of Queensland, Brisbane, Australia, in 2005 and at California Institute of Technology during 2006–2008. He joined the Physics Department of the Université de Sherbrooke in 2008 where he is currently an Assistant Professor. His research interests include the theory of quantum error correction, quantum algorithms, and numerical methods for the simulation of quantum many-body systems.

Jean-Pierre Tillich (M'06) was born in Mulhouse, France, in 1966. He received the Engineer degree from École des Mines de Paris, Paris, France, in 1989 and the Ph.D. degree in computer science from École Nationale Supérieure des Télécommunications (ENST), Paris, France, in 1994.

From 1997 to 2003, he was an Assistant Professor at the University Paris XI, Paris, France. He is now a Researcher at the Institut de Recherche en Informatique et Automatique (INRIA), Rocquencourt, Le Chesnay, France. His research interests include classical and quantum coding theory, cryptography, and graph theory.

Harold Ollivier received the Ph.D. degree from École Polytechnique in 2004 for his work on quantum foundations, decoherence, and error correction.

He was a Postdoctoral Fellow at the Perimeter Institute for Theoretical Physics, Waterloo, ON, Canada, until 2006, where he further developed new quantum error correction schemes. He later joined the French Ministry for Finance and Economy where he was in charge of venture capital policies. He now manages the Institut Louis Bachelier, Paris, France and the Fondation du Risque, two nonprofit companies dedicated to funding academic research in finance, insurance, and risk management.