Convolutional and Tail-Biting
Quantum Error-Correcting Codes

G. David Forney, Jr.,[*] Markus Grassl,[†] and Saikat Guha[‡]

Abstract

Rate-$(n-2)/n$ unrestricted and CSS-type quantum convolutional codes with up to 4096 states and minimum distances up to 10 are constructed as stabilizer codes from classical self-orthogonal rate-$1/n$ $\mathbb{F}_4$-linear and binary linear convolutional codes, respectively. These codes generally have higher rate and less decoding complexity than comparable quantum block codes or previous quantum convolutional codes. Rate-$(n-2)/n$ block stabilizer codes with the same rate and error-correction capability and essentially the same decoding complexity are derived from these convolutional codes via tail-biting.

Index terms: Quantum error-correcting codes, CSS-type codes, quantum convolutional codes, quantum tail-biting codes.

I.   Introduction

Quantum error-correcting codes (QECCs) protect quantum states from unwanted perturbations, allowing the implementation of robust quantum computing and communication systems.

The first breakthrough in this field was Shor's demonstration in 1995 via a 9-qubit single-error-correcting code [28] that quantum error-correction was even possible, which was not obvious *a priori*. Shortly thereafter, a more efficient 7-qubit single-error-correcting code was found by Steane [29] and by Calderbank and Shor [5]. This code was merely the first of a class of quantum codes based on classical binary error-correcting codes, which we call *CSS-type* codes. Later in 1996, an even more efficient 5-qubit single-error-correcting code was found by Bennett *et al.* [2] and by Laflamme *et al.* [22].

Soon thereafter, a general theory of *stabilizer codes* was developed [6, 14, 24, 27], which includes the above codes as particular cases, and indeed essentially all QECCs developed to date. The stabilizer formalism, which we review below, has the virtue of reducing the QECC problem to pure mathematics, and thus allowing non-physicists to contribute to the field. In particular, it shows how to convert certain classical $\mathbb{F}_4$-linear and binary error-correcting codes to QECCs [7].

[*]Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139. E-mail: `forneyd@comcast.net`.

[†]Institut für Algorithmen und Kognitive Systeme, Universität Karlsruhe (TH), 76128 Karlsruhe, Germany. E-mail: `grassl@ira.uka.de`.

[‡]Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, MA 02139. E-mail: `saikat@mit.edu`.

In this paper, we systematically develop quantum convolutional codes (QCCs) using the same general principles. We focus on rate-$1/n$ codes, where we can exhibit some simple and attractive codes as examples, but our construction principles are general.

Practical classical communication systems have mostly used convolutional codes rather than block codes, because convolutional codes are generally superior in terms of their performance-complexity tradeoff. In quantum coding, it is still too early to say which characteristics of QECCs will turn out to be the most important. However, we find that quantum convolutional codes compare favorably with quantum block codes in the following ways:

- **Code rate.** In general, QCCs require fewer encoded qubits to protect the same number of information qubits than comparable block codes. For example, our rate-$1/3$ single-error-correcting QCCs are comparable to the 5-qubit and 7-qubit single-error-correcting block codes mentioned above, but have higher code rate.

- **Decoding complexity.**[1] In general, QCCs have simpler decoding algorithms. For example, we present extremely simple decoding algorithms for our single-error-correcting QCCs.

- **Performance.** In general, QCCs have a superior tradeoff between performance and complexity. For example, our rate-$1/3$ single-error-correcting QCCs have comparable error probability to the 5-qubit and 7-qubit block codes, even though they have higher rate and simpler decoders.

One possible drawback of QCCs is their lack of a natural block structure. Previous authors have proposed terminating QCCs to yield block codes with the same error-correction capability, at the cost of reduced rate (sometimes without recognizing that a terminated convolutional stabilizer code may not be a valid (*i.e.,* self-orthogonal) block stabilizer code; see Section III-C.) We propose instead to construct quantum *tail-biting* codes (QTBCs), which are block stabilizer codes that retain the same code rate, error correction capability, and decoding algorithms as the QCCs from which they are derived, provided that the block length is large enough. We exhibit families of rate-$1/n$ QTBCs with attractive performance-complexity tradeoffs.

Surprisingly, no one previously seems to have constructed QCCs that are clearly superior to quantum block codes by a straightforward extension of the stabilizer formalism. Chau [9, 10] proposed two "quantum convolutional codes:" the first involves a one-to-one convolutional sequence transformation followed by a quantum block code, has the same performance and complexity as the block code, and thus is arguably not really "convolutional;" the second generalizes Shor's concatenated construction to convolutional codes, and yields a single-error-correcting rate-$1/4$ QCC that corrects 1 error in every 8 qubits, comparable to a $[8, 2, 3]$ shortened quantum Hamming code (see Section II.E below). Ollivier and Tillich [25, 26] constructed a rate-$1/5$ single-error-correcting QCC using the stabilizer formalism, but unfortunately their example QCC does not improve on the comparable 5-qubit block code in either performance or complexity. However, they do address various gate-level implementation issues that we do not address in this paper. (For further results on quantum convolutional encoders, see [18].) More recently, Almeida and Palazzo [1] have proposed rate-$1/4$, -$1/3$ and -$2/4$ convolutional codes using a Shor-type concatenated construction; these codes appear to be much more complicated than ours to decode.

---

[1]In this paper, "complexity" will always mean decoding complexity; we do not consider complexity issues at the quantum gate level. Moreover, the decoding computation after the measurement of syndromes is entirely classical.

In Section II, we briefly review stabilizer codes, particularly $\mathbb{F}_4$-linear and CSS-type codes. In Section III, we show how to construct quantum convolutional and tail-biting codes from classical self-orthogonal $\mathbb{F}_4$-linear and binary convolutional codes. We give examples of simple rate-1/3 single-error-correcting $\mathbb{F}_4$-linear and CSS-type QCCs and QTBCS, and their decoding algorithms. In Section IV, we briefly summarize the algebraic structure theory of rate-$1/n$ linear shift-invariant convolutional codes and their orthogonal codes, and specify the relevant symmetries of these codes in the QECC context. In Section V, we tabulate rate-$1/n$ single-error-correcting codes and the corresponding tail-biting codes. In Section VI, we present the best rate-1/3 codes with state space sizes up to 2048 and minimum distances up to 10, with their corresponding tail-biting codes.

II.   Review of stabilizer codes

In this section we review the stabilizer formalism, originally developed by Calderbank *et al.* [6] and Gottesman [14], in order to fix nomenclature and notation. We focus especially on $\mathbb{F}_4$-linear stabilizer codes [7] and CSS-type codes [5, 29].

A.   Qubits and Pauli matrices

A *qubit* is a quantum system whose Hilbert space $\mathcal{H}$ is two-dimensional.

Given a basis for $\mathcal{H}$, a basic set of unitary Hermitian operators on $\mathcal{H}$ is the set $\Pi = \{I, X, Y, Z\}$ of *Pauli matrices*, defined by

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

The multiplication table of these matrices is evidently as follows:

$\times$	I	X	Y	Z
I	I	X	Y	Z
X	X	I	iZ	$-iY$
Y	Y	$-iZ$	I	iX
Z	Z	iY	$-iX$	I

Their commutation properties may therefore be summarized as follows: if A and B are two Pauli matrices, then $AB = BA$ if A or B is the identity or if $A = B$; otherwise $AB = -BA$.

The set Π is not a multiplicative group, because it is not closed under multiplication. However, let us consider instead the set $[\Pi] = \{[A] \mid A \in \Pi\}$ of equivalence classes of Pauli matrices defined by $[A] = \{\beta A \mid \beta \in \mathbb{C}, |\beta| = 1\}$; *i.e.*, B is equivalent to $A \in \Pi$ if $B = \beta A$, where β is a unit-magnitude complex number.[2] From the multiplication table of Π, multiplication in $[\Pi]$ is well defined and commutative, since $[A][B] = [B][A] = [AB] = [BA]$. The multiplication table of $[\Pi]$ is

$\times$	$[I]$	$[X]$	$[Y]$	$[Z]$
$[I]$	$[I]$	$[X]$	$[Y]$	$[Z]$
$[X]$	$[X]$	$[I]$	$[Z]$	$[Y]$
$[Y]$	$[Y]$	$[Z]$	$[I]$	$[X]$
$[Z]$	$[Z]$	$[Y]$	$[X]$	$[I]$

---

[2]Previous authors restrict β to $\{\pm 1, \pm i\}$, which suffices to make $[\Pi]$ a group; however, allowing β to range over all unit-magnitude complex numbers is more natural physically.

Thus $[\Pi]$ forms a commutative (abelian) multiplicative group, which we will call the *projective Pauli group.*

By inspection, the projective Pauli group $[\Pi]$ is isomorphic to the group $(\mathbb{Z}_2)^2 = \{00, 01, 10, 11\}$ of binary 2-tuples, whose addition table is

+	00	10	11	01
00	00	10	11	01
10	10	00	01	11
11	11	01	00	10
01	01	11	10	00

Alternatively, the projective Pauli group is isomorphic to the additive group of the quaternary field $\mathbb{F}_4 = \{0, 1, \omega, \overline{\omega}\}$, whose addition table is

+	0	ω	1	$\overline{\omega}$
0	0	ω	1	$\overline{\omega}$
ω	ω	0	$\overline{\omega}$	1
1	1	$\overline{\omega}$	0	ω
$\overline{\omega}$	$\overline{\omega}$	1	ω	0

In short, $0 + a = a$, $a + a = 0$ (so subtraction is the same as addition), and $1 + \omega + \overline{\omega} = 0$.

These tables have been arranged to suggest that the elements of $[\Pi]$, or their representatives in Π, may be labeled by elements of $(\mathbb{Z}_2)^2$ or of $\mathbb{F}_4$ according to the following correspondences:

Π	$(\mathbb{Z}_2)^2$	$\mathbb{F}_4$
I	00	0
X	10	ω
Y	11	1
Z	01	$\overline{\omega}$

Then the *label maps* $\ell : [\Pi] \to (\mathbb{Z}_2)^2$ and $L : [\Pi] \to \mathbb{F}_4$ that are defined by these correspondences are isomorphisms; *i.e.,*

$$\ell([A]) + \ell([B]) = \ell([AB]); \qquad L([A]) + L([B]) = L([AB]).$$

By a slight abuse of notation, we may apply the label maps ℓ and L to Π, or to any matrices in the equivalence classes $[A] \in [\Pi]$. By a further slight abuse of notation, we may pass between two-bit and quaternary labels via label maps $\ell : \mathbb{F}_4 \to (\mathbb{Z}_2)^2$ and $L : (\mathbb{Z}_2)^2 \to \mathbb{F}_4$.

The two label bits in $\ell(A)$, namely $\ell_1(A)$ and $\ell_2(A)$, represent a bit flip and a phase flip, respectively, since X (or any βX, $|\beta| = 1$) is a bit flip operator, Z is a phase flip operator, and $Y = iXZ$ is a combination of a bit flip and a phase flip.

Finally, we may use the quaternary labels to characterize the commutation properties of Pauli matrices. The *traces* of the elements $\{0, 1, \omega, \overline{\omega}\}$ of $\mathbb{F}_4$ are defined as $\{0, 0, 1, 1\}$, and their *conjugates* are defined as $\{0, 1, \overline{\omega}, \omega\}$. The *Hermitian inner product* of two elements $a, b \in \mathbb{F}_4$ is defined as $\langle a, b \rangle = a^\dagger b \in \mathbb{F}_4$, where "$\dagger$" denotes conjugation. The *trace inner product* is defined as $\mathrm{Tr}\, \langle a, b \rangle \in \mathbb{F}_2$. Thus the multiplication, Hermitian inner product, and trace inner product tables of $\mathbb{F}_4$ are

$\times$	0	1	ω	$\overline{\omega}$
0	0	0	0	0
1	0	1	ω	$\overline{\omega}$
ω	0	ω	$\overline{\omega}$	1
$\overline{\omega}$	0	$\overline{\omega}$	1	ω

$\langle,\rangle$	0	1	ω	$\overline{\omega}$
0	0	0	0	0
1	0	1	ω	$\overline{\omega}$
ω	0	$\overline{\omega}$	1	ω
$\overline{\omega}$	0	ω	$\overline{\omega}$	1

Tr $\langle,\rangle$	0	1	ω	$\overline{\omega}$
0	0	0	0	0
1	0	0	1	1
ω	0	1	0	1
$\overline{\omega}$	0	1	1	0

Comparing the trace inner product table of $\mathbb{F}_4$ with the multiplication table of Π, we see that two matrices $A, B \in \Pi$ commute if $\mathrm{Tr}\ \langle L(A), L(B) \rangle = 0$, and anticommute if $\mathrm{Tr}\ \langle L(A), L(B) \rangle = 1$. In other words,

$$AB = (-1)^{\mathrm{Tr}\ \langle L(A), L(B) \rangle} BA.$$

A corresponding inner product may be defined over $(\mathbb{Z}_2)^2$ (a "symplectic" or "twisted" inner product) such that a similar result is obtained; however, we will have no need for such a binary inner product in this paper.

B. Multi-qubit systems and Pauli n-tuples

An n-qubit system is a quantum system whose Hilbert space $\mathcal{H}$ is the tensor product of n two-dimensional spaces, and is thus 2^n-dimensional.

A Pauli n-tuple $\mathbf{A} = A_1 \otimes A_2 \otimes \cdots \otimes A_n$ is a tensor product of n Pauli matrices $A_i, 1 \leq i \leq n$, that act separately on each of the n qubits. The set of all 4^n Pauli n-tuples will be denoted by Π^n.

The product of two Pauli n-tuples is the componentwise product of its elements:

$$\mathbf{AB} = (A_1 B_1) \otimes (A_2 B_2) \otimes \cdots \otimes (A_n B_n).$$

Consequently the product is a Pauli n-tuple up to phase. If we again define equivalence classes of Pauli n-tuples up to phase by $[\mathbf{A}] = \{\beta \mathbf{A} \mid \beta \in \mathbb{C}, |\beta| = 1\}$, then we obtain a well-defined product

$$[\mathbf{A}][\mathbf{B}] = [A_1 B_1] \otimes [A_2 B_2] \otimes \cdots \otimes [A_n B_n] = [\mathbf{AB}].$$

Thus the set of 4^n equivalence classes of Pauli n-tuples is a commutative multiplicative group $[\Pi^n]$ which is isomorphic to $((\mathbb{Z}_2)^2)^n$. We may thus label the elements of $[\Pi^n]$ by binary $2n$-tuples in $((\mathbb{Z}_2)^2)^n$, or by quaternary n-tuples in $(\mathbb{F}_4)^n$, by extending the label maps ℓ and L of the previous subsection. The resulting label maps remain isomorphisms; *i.e.*,

$$\ell([\mathbf{A}]) + \ell([\mathbf{B}]) = \ell([\mathbf{AB}]); \qquad L([\mathbf{A}]) + L([\mathbf{B}]) = L([\mathbf{AB}]).$$

Finally, the Hermitian inner product over $(\mathbb{F}_4)^n$ is defined by the componentwise sum $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_i a_i^\dagger b_i$. The trace inner product over $(\mathbb{F}_4)^n$ is therefore also a componentwise sum:

$$\mathrm{Tr}\ \langle L(\mathbf{A}), L(\mathbf{B}) \rangle = \mathrm{Tr}\ \left(\sum_{i=1}^n L(A_i)^\dagger L(B_i) \right) = \sum_{i=1}^n \mathrm{Tr}\ L(A_i)^\dagger L(B_i).$$

Thus the single-qubit Pauli matrix commutation relation may be extended to Pauli n-tuples:

$$\mathbf{BA} = (-1)^{\mathrm{Tr}\ \langle L(\mathbf{A}), L(\mathbf{B}) \rangle} \mathbf{AB}.$$

C. Stabilizer and normalizer codes

Within the stabilizer formalism, an $[n, k]$ *stabilizer code* is defined by a set of $n - k$ *independent* (*i.e.,* none of the generators is equivalent to a product of the others, up to phase) commuting Pauli *n-tuples* $\mathcal{G} = \{\mathbf{G}_j, 1 \leq j \leq n - k\}$, where $0 \leq k \leq n$. The code subspace is the common eigenspace $\mathcal{S}$ of the generators $\mathbf{G}_j \in \mathcal{G}$ such that the eigenvalue of each $\mathbf{G}_j \in \mathcal{G}$ is $+1$; *i.e.*, $\mathcal{S}$ is the subspace of $\mathcal{H}$ that is stabilized by $\mathcal{G}$.

The *stabilizer group* $S \subset [\Pi^n]$ is defined as the set of all of equivalence classes in $[\Pi^n]$ of all 2^{n-k} products of the $n - k$ generators $\mathbf{G}_j \in \mathcal{G}$. By the independence condition, the equivalence classes of these 2^{n-k} products are distinct.

The *binary stabilizer label code* $\ell(S)$ is then the image of S under the binary label map. Since $\ell : S \rightarrow \ell(S)$ must be an isomorphism, $\ell(S)$ must be the classical $(2n, n - k)$ binary linear block code that is generated by the $n - k$ linearly independent generators $\ell(\mathbf{G}_j)$.

Similarly, the *quaternary stabilizer label code* $L(S)$ is the image of S under the quaternary label map. Evidently $L(S)$ is a group under addition that is isomorphic to $\ell(S) \cong (\mathbb{Z}_2)^{n-k}$.

In order that all generators $\mathbf{G}_j \in \mathcal{G}$ commute, the trace inner product of the quaternary labels of any two generators must be 0; *i.e.*, $L(S)$ must be self-orthogonal under the trace inner product. This holds if and only if the generators of $L(S)$ are self-orthogonal and mutually orthogonal under the trace inner product.

We will focus on the case in which $L(S)$ is actually $\mathbb{F}_4$-linear; *i.e.*, closed under multiplication by scalars in $\mathbb{F}_4$. $L(S)$ is then a classical $(n, (n - k)/2)$ linear block code over $\mathbb{F}_4$. This implies that the integer $n - k$ must be even, and that $L(S)$ is the set of all $\mathbb{F}_4$-linear combinations of $(n - k)/2$ independent generators $\mathbf{g}_i, 1 \leq i \leq (n - k)/2$.

Equivalently, $L(S)$ is the set of all binary linear combinations of the $n - k$ generators $\omega \mathbf{g}_i$ and $\overline{\omega} \mathbf{g}_i$, $1 \leq i \leq (n - k)/2$. This implies that the set $\mathcal{G}$ of generators $\mathbf{G}_j$ of S may be taken to be the $n - k$ inverse images in Π^n of this set of $(n - k)/2$ generator pairs under the inverse quaternary label map. (Note that when inverting label maps defined on Π^n, we take the inverse as the unique element of Π^n with the given label; *i.e.*, we fix the phase of the inverse image.)

Moreover, if $L(S)$ is $\mathbb{F}_4$-linear, then $L(S)$ must be self-orthogonal under the Hermitian inner product, not just the trace inner product, since Tr $\langle \alpha \mathbf{g}, \mathbf{h} \rangle = $ Tr $\alpha \langle \mathbf{g}, \mathbf{h} \rangle$ is equal to 0 for $\alpha = 1, \omega, \overline{\omega}$ if and only if $\langle \mathbf{g}, \mathbf{h} \rangle = 0$.

The *quaternary normalizer label code* is defined as the orthogonal code $L(S)^{\perp}$ to $L(S)$ with respect to the trace inner product; *i.e.*, the set of all elements of $(\mathbb{F}_4)^n$ whose trace inner product with all elements of $L(S)$ is zero. If $L(S)$ is $\mathbb{F}_4$-linear, then, by the same argument as above, the orthogonal code $L(S)^{\perp}$ under the trace inner product must be equal to the orthogonal code $L(S)^{\perp}$ under the Hermitian inner product. Thus $L(S)^{\perp}$ is a classical $(n, (n + k)/2)$ $\mathbb{F}_4$-linear block code.

The inverse image of $L(S)^{\perp}$ under the inverse quaternary label map is called the *projective normalizer group* $N(S) \subseteq [\Pi^n]$, or simply the "normalizer group," a commutative subgroup of $[\Pi^n]$ of size 2^{n+k}. In other words, $L(N(S)) = L(S)^{\perp}$, so the normalizer group is the set of equivalence classes of all Pauli n-tuples that commute with all elements of the stabilizer group S. Evidently S is a subgroup of $N(S)$, and the stabilizer label code $L(S)$ is a subcode of $L(N(S))$.

D. Quantum error correction

We now explain briefly how an $[n, k]$ stabilizer code may be used to encode the state of a k-qubit system into that of an n-qubit system, and then to correct a certain set Σ of error patterns.

Let $\mathcal{H}$ denote the 2^n-dimensional Hilbert space of the n-qubit system. The $n - k$ independent commuting Pauli n-tuple generators $\mathbf{G}_j$ of the stabilizer group S each have eigenvalues $\{\pm 1\}$, and have two corresponding orthogonal eigenspaces of dimension 2^{n-1}. It is straightforward to show that since the $n - k$ generators are independent and commuting, the stabilizer group S that they generate has a set of 2^{n-k} orthogonal eigenspaces, each of dimension 2^k, corresponding to the 2^{n-k} possible combinations of $\{\pm 1\}$ eigenvalues of the generators.

The 2^k-dimensional eigenspace for which all $n - k$ generator eigenvalues equal $+1$ (*i.e.,* the subspace stabilized by S) is defined as the code subspace $\mathcal{S} \subseteq \mathcal{H}$. For encoding, first the Hilbert space of a k-qubit quantum system is embedded into $\mathcal{H}$, *e.g.,* by initializing the remaining $n - k$ qubits to a fixed state. Then a unitary transformation maps this 2^k-dimensional subspace into $\mathcal{S}$.

The object of quantum error correction is to recover the encoded state $|\phi\rangle \in \mathcal{S}$ from a possibly perturbed state $|\phi'\rangle \in \mathcal{H}$. It turns out that it suffices to consider perturbations of the form $|\phi'\rangle = \mathbf{E}|\phi\rangle$ for an *error pattern* $\mathbf{E}$ which is a Pauli n-tuple, because all possible linear perturbations are linear combinations of Pauli n-tuples.

By the general principles of quantum mechanics, a measurement of an operator of a quantum system yields an eigenvalue of that operator, and projects the system state onto the corresponding eigenspace. A set of operators may be measured simultaneously if and only if they commute.

In decoding, we first measure simultaneously the $n - k$ commuting generators $\mathbf{G}_j$ of S. This yields an $(n - k)$-tuple of eigenvalues ± 1, which may be mapped to a binary $(n - k)$-tuple using the standard map $\{\pm 1\} \to \{0, 1\}$. The resulting binary $(n - k)$-tuple $\mathbf{s} = (s_1, \ldots, s_{n-k}) \in (\mathbb{Z}_2)^{n-k}$ is called the *syndrome sequence*. This "hard decision" in fact extracts all relevant information.

If the error pattern is $\mathbf{E}$, then measurement of $\mathbf{G}_j$ results in the eigenvalue $+1$ if $\mathbf{E}$ commutes with $\mathbf{G}_j$, or -1 if $\mathbf{E}$ anticommutes with $\mathbf{G}_j$, regardless of the original state $|\phi\rangle \in \mathcal{S}$. (*Proof*: If $\mathbf{E}$ commutes with $\mathbf{G}_j$, then $\mathbf{G}_j(\mathbf{E}|\phi\rangle) = \mathbf{E}\mathbf{G}_j|\phi\rangle = \mathbf{E}|\phi\rangle$; otherwise $\mathbf{G}_j(\mathbf{E}|\phi\rangle) = -\mathbf{E}\mathbf{G}_j|\phi\rangle = -\mathbf{E}|\phi\rangle$.)

Thus the syndrome bit s_j depends only on $\mathbf{E}$, and is given by $s_j(\mathbf{E}) = \mathrm{Tr}\,\langle L(\mathbf{E}), L(\mathbf{G}_j)\rangle$. The syndrome bit map $s_j : [\Pi^n] \to \mathbb{Z}_2$ is a homomorphism, by the bilinearity of the trace inner product.

The *syndrome map* $\mathbf{s} : [\Pi^n] \to (\mathbb{Z}_2)^{n-k}$ defined by $\mathbf{s}(\mathbf{E}) = \{s_j(\mathbf{E}), 1 \leq j \leq n - k\}$ is thus a homomorphism. Its kernel is the set of all equivalence classes of Pauli n-tuples that commute with all generators $\mathbf{G}_j$, which is precisely the normalizer group $N(S)$. It follows that each of the 2^{n-k} cosets of $N(S)$ in $[\Pi^n]$ maps to a distinct binary syndrome $(n - k)$-tuple $\mathbf{s}$.

In the case where $L(S)$ is $\mathbb{F}_4$-linear, we can compute an $\mathbb{F}_4$-syndrome equal to the Hermitian inner product $\langle L(\mathbf{E}), \mathbf{g}\rangle \in \mathbb{F}_4$ for any $\mathbf{g} \in L(S)$ as follows. By $\mathbb{F}_4$-linearity, both $\omega\mathbf{g}$ and $\overline{\omega}\mathbf{g}$ are also in $L(S)$. The corresponding pair of syndrome bits is

$$(\mathrm{Tr}\,\langle L(\mathbf{E}), \omega\mathbf{g}\rangle, \mathrm{Tr}\,\langle L(\mathbf{E}), \overline{\omega}\mathbf{g}\rangle) = (\mathrm{Tr}\,\overline{\omega}\langle L(\mathbf{E}), \mathbf{g}\rangle, \mathrm{Tr}\,\omega\langle L(\mathbf{E}), \mathbf{g}\rangle).$$

Now observe that $(\mathrm{Tr}\,\omega a, \mathrm{Tr}\,\overline{\omega} a) = \{(0,0), (1,1), (1,0), (0,1)\}$ for $a = \{0, 1, \omega, \overline{\omega}\}$; in other words, $(\mathrm{Tr}\,\omega a, \mathrm{Tr}\,\overline{\omega} a) = \ell(a), a \in \mathbb{F}_4$. Thus these two syndrome bits are the two bits of the binary label $\ell(\langle L(\mathbf{E}), \mathbf{g}\rangle)$, which identifies the $\mathbb{F}_4$-syndrome $\langle L(\mathbf{E}), \mathbf{g}\rangle$.

Thus we can measure the $\mathbb{F}_4$-syndromes $S_k(\mathbf{E}) = \langle L(\mathbf{E}), \mathbf{g}_k\rangle$ for each of the $(n - k)/2$ generators $\mathbf{g}_k$ of the quaternary stabilizer label code $L(S)$. The syndrome map then becomes a homomorphism $\mathbf{S} : [\Pi^n] \to (\mathbb{F}_4)^{(n-k)/2}$ with kernel $N(S)$. Again, each of the $4^{(n-k)/2}$ cosets of $N(S)$ in $[\Pi^n]$ maps to a unique $\mathbb{F}_4$-syndrome $(n - k)/2$-tuple $\mathbf{S}$.

The syndrome measurement projects the perturbed state $|\phi'\rangle$ onto a state $|\phi''\rangle$ in the 2^k-dimensional eigenspace $\mathcal{S}(\mathbf{s})$ that corresponds to the measured syndrome $\mathbf{s}$ (or $\mathbf{S}$). If the actual perturbation was a Pauli n-tuple $\mathbf{E}$, then $|\phi'\rangle = \mathbf{E}|\phi\rangle$ is an eigenvector of $\mathcal{S}(\mathbf{s})$, so $|\phi''\rangle = |\phi'\rangle$.

The next step in decoding is to find the most likely error pattern $\hat{\mathbf{E}}$ in the coset of $N(S)$ whose syndrome is $\mathbf{s}$ (or $\mathbf{S}$), called the coset leader. The most likely error pattern is assumed to be the one of lowest Hamming weight; *i.e.,* the Pauli n-tuple with the fewest non-identity components.

Finding the coset leader is an entirely classical computation. There are 2^{n-k} possible syndromes $\mathbf{s}$ (or $\mathbf{S}$), so if $n-k$ is not too large, then the coset leaders may be precomputed, and this step may be performed by a table lookup in a table with 2^{n-k} entries. In the QECC literature, the table lookup method is usually assumed, explicitly or implicitly.

Finally, given the coset leader $\hat{\mathbf{E}}$, the perturbed state $|\phi'\rangle$ is "corrected" to $\hat{\mathbf{E}}|\phi'\rangle = \hat{\mathbf{E}}\mathbf{E}|\phi\rangle$. Decoding is successful if the coset leader $\hat{\mathbf{E}}$ is merely in the same coset of S as the actual error pattern $\mathbf{E}$, so $\hat{\mathbf{E}}\mathbf{E} \in S$, because any error pattern $\hat{\mathbf{E}}\mathbf{E} \in S$ stabilizes any $|\phi\rangle \in \mathcal{S}$; *i.e.,* error patterns in S do not affect states in $\mathcal{S}$.

The set of 2^{n-k} coset leaders of the cosets of $N(S)$ (including $[\mathbf{I}]$) will be denoted by Σ. The set of correctable error patterns is then $[S\Sigma]$. If $[S\Sigma]$ contains all error patterns of Hamming weight t or less, then $\mathcal{S}$ is called a t-error-correcting code.

Since the Hamming distance function is a true metric, $\mathcal{S}$ will be t-error-correcting if the minimum Hamming distance d between cosets of S in $N(S)$ is greater than $2t$ (*i.e.,* if $d \geq 2t+1$), because then two error patterns $\mathbf{E}$ and $\hat{\mathbf{E}}$ of weight $\leq t$ cannot lie in the same coset of $N(S)$ unless they are in the same coset of S; *i.e.,* $\hat{\mathbf{E}}\mathbf{E}$ cannot be in $N(S)$ unless it is in S. By the group property, this minimum distance d is the minimum Hamming weight of any nonzero coset of S in $N(S)$; *i.e.,* d is the minimum Hamming weight in $N(S) \setminus S$.

An $[n, k]$ stabilizer code in which $N(S) \setminus S$ has minimum Hamming weight d is called an $[n, k, d]$ stabilizer code. We will consider only *nondegenerate* codes, in which the normalizer code $L(N(S))$ actually has minimum Hamming distance d; *i.e.,* $L(S)$ has a minimum distance of at least d.

E. Summary: stabilizer codes from $\mathbb{F}_4$-linear codes

In summary, to construct a nondegenerate $[n, k, d]$ stabilizer code with $n-k$ even, it suffices to find a classical self-orthogonal $(n, (n-k)/2)$ $\mathbb{F}_4$-linear block code $L(S)$ whose orthogonal $(n, (n+k)/2)$ code $L(S)^\perp$ under the Hermitian inner product has minimum Hamming distance $d^\perp = d$.

Example A (Five-qubit "quantum Hamming code"). In order to construct a single-error-correcting $[5, 1, 3]$ stabilizer code, we take the quaternary stabilizer label code $L(S)$ to be the classical $(5, 2, 4)$ self-orthogonal (doubly extended Reed-Solomon) code over $\mathbb{F}_4$ generated by

$$\begin{matrix} 0 & \overline{\omega} & \omega & \omega & \overline{\omega} \\ \overline{\omega} & 0 & \overline{\omega} & \omega & \omega \end{matrix},$$

The orthogonal code $L(N(S)) = L(S)^\perp$ under the Hermitian inner product is then the $(5, 3, 3)$ quaternary Hamming code.

A $[5, 1, 3]$ stabilizer code is single-error-correcting. Each of the 15 error patterns in Π^5 of Hamming weight 1 therefore lies in a distinct one of the 15 nonzero cosets of $N(S)$; *i.e.,* this code is a "perfect" single-error-correcting "quantum Hamming code." Decoding may be performed by a table lookup in a table that maps each of the 16 possible syndromes to the corresponding minimum-weight error pattern. □

Similarly, for all integers $m \geq 2$, there exist classical perfect $\mathbb{F}_4$-linear Hamming codes with parameters $(n = (4^m - 1)/3, k = n - m, d = 3)$ that contain their orthogonal (n, m) codes [7].[3]

---

[3]By Bonisoli's theorem [30], the (n, m) code is equidistant, with all nonzero codewords having weight 4^{m-1}.

These codes may be used to construct quantum $[n = (4^m − 1)/3, k = n − 2m, d = 3]$ Hamming codes that can be decoded using a lookup table with 4^m entries; *e.g.,* stabilizer codes with parameters $[5, 1, 3], [21, 15, 3], [85, 77, 3]$, and so forth.

F.   CSS-type stabilizer codes from $\mathbb{F}_2$-linear codes

The binary field $\mathbb{F}_2$ is a subfield of the quaternary field $\mathbb{F}_4$. Therefore the $(n − k)/2$ generators $\{\mathbf{g}_j, 1 \le j \le (n − k)/2\}$ of a classical $(n, (n − k)/2)$ binary linear code $\mathcal{B}$ may be taken as the generators of an $(n, (n − k)/2)$ $\mathbb{F}_4$-linear code $\mathcal{C}$. Since the Hermitian inner product of binary sequences is the ordinary binary inner product, $\mathcal{C}$ will be self-orthogonal if $\mathcal{B}$ is self-orthogonal.

As we have seen, $\mathcal{C}$ may be characterized as the set of all binary linear combinations of the $n − k$ generators $\{\omega\mathbf{g}_j, \overline{\omega}\mathbf{g}_j \mid 1 \le j \le (n − k)/2\}$. The two-bit labels $\ell(\omega\mathbf{g}_j)$ are nonzero only in bit flip bits, whereas the labels $\ell(\overline{\omega}\mathbf{g}_j)$ are nonzero only in phase flip bits. Therefore the binary code $\ell(\mathcal{C})$ may be characterized as two interleaved, independent binary codes: namely, the code $\mathcal{B}$ applied to the n bit flip bits, and the code $\mathcal{B}$ applied to the n phase flip bits, respectively. In short, $\ell(\mathcal{C})$ is a direct product code:

$$\ell(\mathcal{C}) = \mathcal{B} \times \mathcal{B}.$$

Similarly, the orthogonal code $\mathcal{C}^\perp$ to $\mathcal{C}$ is generated by $(n + k)/2$ generators of the orthogonal $(n, (n + k)/2)$ binary linear code $\mathcal{B}^\perp$, and the corresponding binary code is the direct product code $\ell(\mathcal{C}^\perp) = \mathcal{B}^\perp \times \mathcal{B}^\perp$. If the minimum distance of $\mathcal{B}^\perp$ is $d^\perp$, then the minimum distance of $\mathcal{C}^\perp$ is $d^\perp$.

More generally, Calderbank and Shor [5] and Steane [29] proposed codes of the form $\mathcal{B}_1 \times \mathcal{B}_2$, where the bit flip code $\mathcal{B}_1$ and the phase flip code $\mathcal{B}_2$ are possibly different orthogonal binary codes. We will consider only codes of the type $\mathcal{B} \times \mathcal{B}$, which we will call *CSS-type codes*.

In short, to construct a nondegenerate $[n, k, d]$ stabilizer code with $n − k$ even, it suffices to find a classical self-orthogonal $(n, (n − k)/2)$ binary linear block code $\mathcal{B}$ whose orthogonal $(n, (n + k)/2)$ code $\mathcal{B}^\perp$ has minimum Hamming distance $d^\perp = d$.

Example B (Seven-qubit Steane code) Consider the $(7, 3, 4)$ binary linear (dual Hamming) code $\mathcal{B}$ that is generated by the following generators:

$$\begin{array}{ccccccc} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{array}.$$

This code is evidently self-orthogonal. Its orthogonal code $\mathcal{B}^\perp$ is the $(7, 4, 3)$ binary Hamming code. Thus the resulting stabilizer code is a $[7, 1, 3]$ single-error-correcting code.   □

In general, CSS-type codes have poorer parameters $[n, k, d]$ than general $\mathbb{F}_4$-linear codes, because binary codes have poorer parameters than quaternary codes. However, they have the advantage that bit flip and phase flip errors may be decoded separately, as follows. Note that the syndrome bits may be written as

$$(\mathrm{Tr} \langle L(\mathbf{E}), \omega\mathbf{g}_j \rangle, \mathrm{Tr} \langle L(\mathbf{E}), \overline{\omega}\mathbf{g}_j \rangle) = (\langle \ell_2(\mathbf{E}), \mathbf{g}_j \rangle, \langle \ell_1(\mathbf{E}), \mathbf{g}_j \rangle),$$

because $\ell(\omega\mathbf{g}_j) = (\mathbf{g}_j, \mathbf{0})$ and $\ell(\overline{\omega}\mathbf{g}_j) = (\mathbf{0}, \mathbf{g}_j)$. In other words, the first syndrome bits in each pair form a set of $(n − k)/2$ syndromes for the phase error bits, and the second for the bit error bits. We can then decode each set of syndromes independently, using a decoder for the binary code $\mathcal{B}^\perp$.

If the Hamming weight of **E** is not greater than $\lfloor (d-1)/2 \rfloor$, then the Hamming weights of $\ell_1(\mathbf{E})$ and $\ell_2(\mathbf{E})$ both satisfy the same bound, so decoding will be successful. Indeed, two independent binary decodings of up to $\lfloor (d-1)/2 \rfloor$ bit errors will correct some higher-weight error patterns.

Example B (cont.) A quaternary decoder for the $(7,4,3)$ $\mathbb{F}_4$-linear code $\mathcal{C}^\perp$ requires calculation of three $\mathbb{F}_4$-syndromes and table lookup in a 64-entry table (for complete decoding; single-error-correction requires at least 21 entries). A binary decoder for the $(7,4,3)$ binary linear code $\mathcal{B}^\perp$ requires calculation of three binary syndromes and table lookup in an 8-entry table. Two binary decodings of $\mathcal{B}^\perp$ are thus arguably simpler than one quaternary decoding of $\mathcal{C}^\perp$. □

Similarly, for all integers $m \geq 3$, there exist classical perfect binary Hamming codes with parameters $(n = 2^m - 1, k = n - m, d = 3)$ that contain their orthogonal (n, m) codes.[4] These codes may be used to construct single-error-correcting quantum $[n = 2^m - 1, k = n - 2m, d = 3]$ stabilizer codes that can be decoded by using a lookup table with 2^m entries twice; *e.g.,* stabilizer codes with parameters $[7, 1, 3], [15, 7, 3], [31, 21, 3]$, and so forth.

In summary, while CSS-type codes based on binary codes may have poorer parameters $[n, k, d]$ than codes based on $\mathbb{F}_4$-linear codes, they may nevertheless have advantages in terms of decoding complexity. (The question of decoding complexity seems hardly to have been addressed previously in the QECC literature, with a few notable exceptions, *e.g.,* [3, 17, 23].) For example, the $[15, 7, 3]$ CSS-type code has comparable decoding complexity to the $[5, 1, 3]$ 5-qubit quantum Hamming code, as well as comparable performance, but has a higher code rate.

III. Quantum convolutional and tail-biting codes

In this section, using the stabilizer formalism, we show how to construct quantum convolutional and tail-biting codes from classical self-orthogonal $\mathbb{F}_4$-linear and binary convolutional codes.

A. Infinite-qubit systems

A quantum convolutional code will be defined (at least in principle) on a quantum system containing a countably infinite ordered sequence of qubits; *i.e.,* whose Hilbert space $\mathcal{H} = \cdots \otimes \mathcal{H}_{-1} \otimes \mathcal{H}_0 \otimes \mathcal{H}_1 \otimes \cdots$ is the tensor product of an infinite sequence of two-dimensional Hilbert spaces $\mathcal{H}_i, i \in \mathbb{Z}$, and thus is infinite-dimensional.

A Pauli sequence $\mathbf{A} = \bigotimes_{i \in \mathbb{Z}} A_i$ acting on $\mathcal{H}$ is an infinite tensor product of Pauli matrices $A_i, i \in \mathbb{Z}$. The set of all such infinite Pauli sequences may be denoted by $\Pi^{\mathbb{Z}}$. Again we may define the set $[\Pi^{\mathbb{Z}}] = \{\beta \mathbf{A} | \mathbf{A} \in \Pi^{\mathbb{Z}}, \beta \in \mathbb{C}, |\beta| = 1\}$ of equivalence classes of Pauli sequences up to phase. The label maps ℓ and L of Section II.A may be extended to maps from $[\Pi^{\mathbb{Z}}]$ to sequences in $((\mathbb{Z}_2)^2)^{\mathbb{Z}}$ and $(\mathbb{F}_4)^{\mathbb{Z}}$, respectively. Inner products and convolutions are well defined, provided that the corresponding sums involve only a finite number of nonzero terms.

B. Convolutional stabilizer codes

We will define a rate-k/n convolutional stabilizer code, $0 \leq k \leq n$, by a basic set of $n - k$ independent commuting Pauli sequences $\mathcal{G}_0 = \{\mathbf{G}_j, 1 \leq j \leq n - k\}$, where each generator $\mathbf{G}_j \in \mathcal{G}$ has finite support; *i.e.,* only finitely many Pauli matrices in $\mathbf{G}_j$ are not identity matrices.

---

[4]By Bonisoli's theorem, the (n, m) code is equidistant, with all nonzero codewords having weight 2^{m-1}.

The full generator set $\mathcal{G}$ will then be the set of all shifts of the k basic generators in $\mathcal{G}_0$ by integer multiples of n qubits. We require the generators in $\mathcal{G}$ to be independent and commuting. The code subspace $\mathcal{S}$ will again be defined to be the subspace of $\mathcal{H}$ that is stabilized by $\mathcal{G}$.

The stabilizer group S is again defined as the subgroup of $[\Pi^{\mathbb{Z}}]$ of all products of generators in $\mathcal{G}$. The binary and quaternary label codes $\ell(S)$ and $L(S)$ again denote the images of S under the binary and quaternary label maps, respectively.

In this paper we will only consider codes in which $L(S)$ is $\mathbb{F}_4$-linear. Then $n - k$ must be even, and $L(S)$ is a classical rate-$((n - k)/2)/n$ convolutional code over $\mathbb{F}_4$ generated by the set of all shifts by integer multiples of n symbols of some set of $(n - k)/2$ independent generators $\{\mathbf{g}_i, 1 \le i \le (n - k)/2\}$ with finite support. In this case the generators in $\mathcal{G}_0$ may be taken as the inverse images of the $n - k$ generators $\omega \mathbf{g}_i$ and $\overline{\omega} \mathbf{g}_i$.

Finally, in order that each generator in $\mathcal{G}_0$ commute with all shifts of all generators in $\mathcal{G}_0$, the $\mathbb{F}_4$-linear convolutional code $L(S)$ must be self-orthogonal under the Hermitian inner product; *i.e.*, a subcode of its orthogonal code $L(S)^\perp$, a classical $\mathbb{F}_4$-linear rate-$((n + k)/2)/n$ convolutional code.

As with block stabilizer codes, a convolutional stabilizer code will turn out to have minimum distance d if the orthogonal code $L(S)^\perp$ has minimum Hamming distance $d^\perp = d$.

We will focus on rate-$(n - 2)/n$ convolutional stabilizer codes, for which the rates of $L(S)$ and $L(S)^\perp$ are $1/n$ and $(n - 1)/n$, respectively. The generators of a rate-$1/n$ classical convolutional code $L(S)$ are the set of all shifts by an integer number of $(\mathbb{F}_4)^n$-blocks of a single finite-support sequence $\mathbf{g} = \{g_m \in (\mathbb{F}_4)^n, m \in \mathbb{Z}\}$.

Example 1 (rate-$1/3$, single-error-correcting, $\mathbb{F}_4$-linear convolutional stabilizer code). Consider the classical rate-$1/3$ $\mathbb{F}_4$-linear shift-invariant convolutional code $\mathcal{C} = L(S)$ that is generated by all shifts by integer multiples of 3 symbols of the generator sequence $\mathbf{g}_1 = (\ldots |000|111|1\omega\overline{\omega}|000|\ldots)$; *i.e.*, whose generators are:

$$
\begin{array}{c}
\cdots \\
\begin{array}{c|ccc|ccc|ccc|ccc|ccc|ccc|c}
\ldots & 0 & 0 & 0 & 1 & 1 & 1 & 1 & \omega & \overline{\omega} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots \\
\ldots & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & \omega & \overline{\omega} & 0 & 0 & 0 & 0 & 0 & 0 & \ldots \\
\ldots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & \omega & \overline{\omega} & 0 & 0 & 0 & \ldots
\end{array} \\
\cdots
\end{array}
$$

In D-transform notation (see Section IV), the basic generator is $\mathbf{g}_1(D) = (1 + D, 1 + \omega D, 1 + \overline{\omega} D)$, and the set of all generators is $\{D^\ell \mathbf{g}_1(D), \ell \in \mathbb{Z}\}$.

The corresponding stabilizer group S is then generated by sequences of Pauli matrices that correspond to multiples by ω and $\overline{\omega}$ of these generators; *i.e.*, the generators of S are the shifts by an integral number of 3-blocks of the two basic generators $(\ldots |III|XXX|XZY|III|\ldots)$ and $(\ldots |III|ZZZ|ZYX|III|\ldots)$.

It is easy to verify that $\mathbf{g}_1$ is orthogonal to itself and to any shift of itself under the Hermitian inner product, which suffices to show that all generators are orthogonal. Thus $\mathcal{C}$ is self-orthogonal; *i.e.*, all generators of S commute.

The orthogonal convolutional code $\mathcal{C}^\perp$ under the Hermitian inner product is the rate-$2/3$ $\mathbb{F}_4$-linear convolutional code that is generated by all shifts of $\mathbf{g}_1$ and $\mathbf{g}_2 = (\ldots |000|\overline{\omega}\omega 1|000|\ldots)$, whose D-transform is $\mathbf{g}_2(D) = (\overline{\omega}, \omega, 1)$. It is easy to verify that the minimum Hamming distance of $\mathcal{C}^\perp$ is $d^\perp = 3$, with the only weight-3 codewords being multiples of shifts of $\mathbf{g}_2$. The convolutional stabilizer code defined by $\mathcal{C}$ thus has minimum Hamming distance 3, so it is single-error-correcting.

$\square$

In principle, the convolutional code $\mathcal{C}$ of Example 1 has an infinite number of generators $\mathbf{g}_j$, covering an infinite number of 3-blocks. However, because the support of each generator is only two 3-blocks, the code constraints are localized; the code symbols in any block depend only on the "current" and "previous" generators. Such a convolutional code is said to have a "memory" or *constraint length* of one 3-block ($\nu = 1$).

C. Block codes from convolutional codes

In data communications, where information symbols are often transmitted in a continuous stream, the non-block structure of convolutional codes is often a virtue rather than a problem. However, for the main applications currently envisioned for quantum error-correcting codes, such as protection of the state of a quantum computer, a block structure is desirable. In this subsection, we will discuss two methods of making a convolutional code into a block code: termination and tail-biting.

To construct a terminated block code $\mathcal{B}$ from a convolutional code $\mathcal{C}$, we simply take the generators of $\mathcal{B}$ to be the subset of all generators of $\mathcal{C}$ whose support lies in some given interval. Since $\mathcal{B}$ is a subcode of $\mathcal{C}$, it will be self-orthogonal if $\mathcal{C}$ is self-orthogonal, and its minimum distance must be at least as great as that of $\mathcal{C}$. The rate of $\mathcal{B}$ will in general be less than that of $\mathcal{C}$, but it will approach the rate of $\mathcal{C}$ as the length of the interval increases.

The orthogonal block code $\mathcal{B}^\perp$ will then be the code generated by the truncations of the generators of the orthogonal convolutional code $\mathcal{C}^\perp$ to the same interval. $\mathcal{B}^\perp$ may therefore have a smaller minimum distance than $\mathcal{C}^\perp$, and may not be self-orthogonal even if $\mathcal{C}^\perp$ is self-orthogonal.

There are therefore two ways that we might think of terminating a convolutional stabilizer code defined by a classical $\mathbb{F}_4$-linear convolutional code $\mathcal{C} = L(S)$. We might try terminating $\mathcal{C}^\perp$ to a block code $\mathcal{B}^\perp$, thus keeping the minimum distance of $\mathcal{B}^\perp$ equal to that of $\mathcal{C}^\perp$. However, the dual block code $\mathcal{B}$ will then be a truncation of the convolutional code $\mathcal{C}$, which is not guaranteed to be self-orthogonal, so in general $\mathcal{B}$ will not define a valid block stabilizer code. The alternative is to terminate $\mathcal{C}$ to a block code $\mathcal{B}$, thus ensuring self-orthogonality, but in this case the truncated code $\mathcal{B}^\perp$ is not guaranteed to have the same minimum distance as $\mathcal{C}^\perp$.

Example 1 (cont.) For example, taking the convolutional code as the rate-1/3 convolutional code $\mathcal{C}$ of Example 1 and an interval of three consecutive 3-blocks, the following two generators of $\mathcal{C}$ have support contained in the given interval:

$$
\begin{array}{ccc|ccc|ccc}
1 & 1 & 1 & 1 & \omega & \overline{\omega} & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & \omega & \overline{\omega}
\end{array}
$$

These generate a $(9,2)$ $\mathbb{F}_4$-linear terminated block code $\mathcal{B}$, which is evidently self-orthogonal.

The orthogonal block code $\mathcal{B}^\perp$ is the $(9,7)$ block code generated by the following seven truncated generators of $\mathcal{C}^\perp$:

$$
\begin{array}{ccc|ccc|ccc}
1 & \omega & \overline{\omega} & 0 & 0 & 0 & 0 & 0 & 0 \\
\overline{\omega} & \omega & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & \omega & \overline{\omega} & 0 & 0 & 0 \\
0 & 0 & 0 & \overline{\omega} & \omega & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & \omega & \overline{\omega} \\
0 & 0 & 0 & 0 & 0 & 0 & \overline{\omega} & \omega & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1
\end{array}
$$

The minimum distance of $\mathcal{B}^\perp$ is now only 2 (*e.g.*, the sum of the first two generators has weight 2).

Alternatively, taking the convolutional code as the rate-2/3 convolutional code $\mathcal{C}^\perp$ of Example 1 and the same three-block interval, the following five generators of $\mathcal{C}^\perp$ have support contained in the given interval:

$$
\begin{array}{ccc|ccc|ccc}
\overline{\omega} & \omega & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & \omega & \overline{\omega} & 0 & 0 & 0 \\
0 & 0 & 0 & \overline{\omega} & \omega & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & \omega & \overline{\omega} \\
0 & 0 & 0 & 0 & 0 & 0 & \overline{\omega} & \omega & 1
\end{array}
$$

These generate a $(9,5,3)$ $\mathbb{F}_4$-linear terminated block code $\mathcal{B}^\perp$.

The orthogonal block code is the $(9,4)$ $\mathbb{F}_4$-linear block code $\mathcal{B}$ generated by the following four truncated generators of $\mathcal{C}$:

$$
\begin{array}{ccc|ccc|ccc}
1 & \omega & \overline{\omega} & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & \omega & \overline{\omega} & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & \omega & \overline{\omega} \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1
\end{array}
$$

Now $\mathcal{B}$ is not even self-orthogonal, because the truncated generators are not self-orthogonal. $\square$

For our purposes, *tail-biting* (see, *e.g.,* [8]) is a better method of making a convolutional code into a block code. To construct a tail-biting block code $\mathcal{B}$ from a convolutional code $\mathcal{C}$, we take the generators of $\mathcal{B}$ to be the subset of all generators of $\mathcal{C}$ whose "starting time" lies in some given interval. If the "ending time" lies outside the given interval, then we wrap the generator around to the beginning of the interval in "tail-biting" fashion; see the example below. We assume that the length of the tail-biting interval is greater than that of the support of any generator.

It can be easily shown that the rate of $\mathcal{B}$ will be the same as that of $\mathcal{C}$ if the generators are noncatastrophic (see Section IV). There is now no guarantee that the minimum distance of $\mathcal{B}$ will be as great as that of $\mathcal{C}$; however, in general the minimum distance will be preserved if the tail-biting interval is long enough [19]. The orthogonal block code $\mathcal{B}^\perp$ will be the corresponding tail-biting block code derived from $\mathcal{C}^\perp$. Finally, if $\mathcal{C}$ is self-orthogonal, then $\mathcal{B}$ will be self-orthogonal.

A tail-biting code $\mathcal{B}$ derived from a self-orthogonal convolutional code $\mathcal{C}$ may therefore be used to specify a block stabilizer code with the same rate as the convolutional stabilizer code derived from $\mathcal{C}$, and, provided that the block length is large enough, the same minimum distance.

Example 2 (rate-1/3, single-error-correcting, $\mathbb{F}_4$-linear tail-biting stabilizer code). If we again take the convolutional code as the rate-1/3 convolutional code $\mathcal{C}$ of Example 1 and a tail-biting interval of length three 3-blocks, then we obtain the following three tail-biting generators:

$$
\begin{array}{ccc|ccc|ccc}
1 & 1 & 1 & 1 & \omega & \overline{\omega} & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & \omega & \overline{\omega} \\
1 & \omega & \overline{\omega} & 0 & 0 & 0 & 1 & 1 & 1
\end{array}
$$

Notice how the last generator has been "wrapped around." Thus these generators generate a $(9,3)$ $\mathbb{F}_4$-linear tail-biting code $\mathcal{B}$. Moreover, since $\mathcal{C}$ is self-orthogonal, $\mathcal{B}$ is self-orthogonal.

The orthogonal block code is the $(9, 6)$ $\mathbb{F}_4$-linear tail-biting code $\mathcal{B}^\perp$ that is generated by the following six tail-biting generators:

$$
\begin{array}{ccc|ccc|ccc}
\overline{\omega} & \omega & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & \omega & \overline{\omega} & 0 & 0 & 0 \\
0 & 0 & 0 & \overline{\omega} & \omega & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & \omega & \overline{\omega} \\
0 & 0 & 0 & 0 & 0 & 0 & \overline{\omega} & \omega & 1 \\
1 & \omega & \overline{\omega} & 0 & 0 & 0 & 1 & 1 & 1
\end{array}
$$

The minimum distance of $\mathcal{B}^\perp$ turns out to be $d^\perp = 3$, so $\mathcal{B}$ defines a $[9, 3, 3]$ block stabilizer code.

$\square$

D.   Decoding algorithms

We now discuss how to decode a convolutional stabilizer code that has been constructed from a classical self-orthogonal $\mathbb{F}_4$-linear rate-k/n convolutional code $\mathcal{C}$.

As shown in Section II, we may first measure each generator $\mathbf{g}_j$ of the convolutional code $\mathcal{C}$ to obtain a sequence $\mathbf{S}$ of $\mathbb{F}_4$-syndromes $S_j = \langle L(\mathbf{E}), \mathbf{g}_j \rangle \in \mathbb{F}_4$, where $L(\mathbf{E})$ denotes the quaternary error label sequence $L(\mathbf{E})$, at a rate of k $\mathbb{F}_4$-syndromes for each n-block. The syndrome sequence $\mathbf{S}$ determines a coset $\mathcal{C}^\perp + \mathbf{t}(\mathbf{S})$ of the orthogonal convolutional code $\mathcal{C}^\perp$, where $\mathbf{t}(\mathbf{S})$ is any error sequence whose syndrome sequence is $\mathbf{S}$.[5] We then need to find the minimum-weight coset leader in that coset, which is an entirely classical computation.

A standard way of finding the leader of a coset $\mathcal{C}^\perp + \mathbf{t}(\mathbf{S})$ of a convolutional code $\mathcal{C}^\perp$ is to represent the coset by a trellis diagram in which there is one-to-one correspondence between coset sequences and trellis paths, and then search for the lowest-weight trellis path by the Viterbi algorithm (VA) [12]. The trellis diagram may be taken as any trellis diagram for $\mathcal{C}^\perp$, with all code sequences translated by the representative error sequence $\mathbf{t}(\mathbf{S})$.

For example, the rate-2/3 convolutional code $\mathcal{C}^\perp$ of Example 1 has a minimal trellis diagram with 4 states at each 3-block boundary, and 64 transitions between trellis states during each 3-block. A VA search through this trellis requires of the order of 64 computations per 3-block.

If our objective is merely correction of single errors, however, then we can use the following much simpler algorithm. As long as all syndromes are zero, we assume that no errors have occurred. Then, if a nonzero syndrome S_j occurs, we assume that a single error has occurred in one of the three qubits in the jth block; the error is characterized by a label 3-tuple $\mathbf{e}_j = L(\mathbf{E}_j)$. The nine possible weight-1 error 3-tuples $\mathbf{e}_j$ lead to the following syndromes (S_j, S_{j+1}) during blocks j and $j+1$:

---

[5]For example, if $\mathbf{S} = \mathbf{e}H^T$ and $(H^{-1})^T$ is any left inverse of H^T, then we may take $\mathbf{t}(\mathbf{S}) = \mathbf{S}(H^{-1})^T$.

$\mathbf{e}_j$	(S_j, S_{j+1})
100	$(1, 1)$
$\omega 00$	(ω, ω)
$\overline{\omega} 00$	$(\overline{\omega}, \overline{\omega})$
010	$(\overline{\omega}, 1)$
$0\omega 0$	$(1, \omega)$
$0\overline{\omega} 0$	$(\omega, \overline{\omega})$
001	$(\omega, 1)$
00ω	$(\overline{\omega}, \omega)$
$00\overline{\omega}$	$(1, \overline{\omega})$

Since these 9 syndrome pairs are distinct, we may map (S_j, S_{j+1}) to the corresponding single-error label 3-tuple $\mathbf{e}_j$ using a simple 9-entry table lookup, and then correct the error as indicated. (If (S_j, S_{j+1}) is not in the table— *i.e.,* if $S_{j+1} = 0$— then we have detected a weight-2 error.)

We see that this simple algorithm can correct any single-error pattern $\mathbf{E}_j$, provided that there is no second error during blocks j and $j + 1$. The decoder synchronizes itself properly whenever a zero syndrome occurs, and subsequently can correct one error in every second block, provided that every errored block is followed by an error-free block.

To decode the $(9, 6, 3)$ tail-biting code $\mathcal{B}^\perp$ of Example 2, we may use the same algorithm, but now on a "circular" time axis. Specifically, if only a single error occurs, then one of the three resulting $\mathbb{F}_4$-syndromes will be zero, and the other two nonzero. The zero syndrome tells which block the error is in; the remaining two nonzero syndromes determine the error pattern according to the 9-entry table given above. Thus again we need only a 9-entry table lookup. (Notice that the existence of a single-error-correcting decoder for $\mathcal{B}^\perp$ proves that its minimum distance is $d^\perp = 3$.)

More generally, there are several methods of adapting the Viterbi algorithm to decode tail-biting codes [8]. The tail-biting code trellis diagram may be taken as a finite-length segment of the corresponding convolutional code trellis, with the further constraint that valid paths must start and end in the same state. An optimal decoding method is VA decoding of each subtrellis consisting of all paths starting and ending in a given state, followed by selection of the best of these decoded paths. A simpler suboptimal method is to regard the tail-biting trellis as being defined on a circular time axis, and to use a single VA search from an arbitrary initial state around and around the circular trellis, until convergence is obtained.

E. Comparison of single-error-correcting codes

We now briefly compare the rate, performance and decoding complexity of our single-error-correcting convolutional and tail-biting codes with those of comparable previous unrestricted QECCs.

First, we will compare the decoding error probability per encoded qubit of our convolutional and tail-biting codes to that of the single-error-correcting 5-qubit block code (Example A). We assume that the probability of an error in any qubit is p, independent of errors in other qubits. Our estimates do not depend on the relative probabilities of X, Y or Z errors.

For the 5-qubit block code, a decoding error occurs if there are 2 errors in any block, so the error probability is of the order of $\binom{5}{2} p^2 = 10 p^2$ per block, or per encoded qubit.

For our rate-1/3 convolutional code, for each 3-qubit block, a decoding error may occur if there are 2 errors in that block, or 1 in that block and 1 in the subsequent block. The error probability

is therefore of the order of $(3 + 3^2)p^2 = 12p^2$ per 3-qubit block, or per encoded qubit.

Finally, for our $[9, 3, 3]$ tail-biting code, a decoding error may occur if there are 2 errors in a block of 9 qubits, so the error probability is of the order of $\binom{9}{2}p^2 = 36p^2$ per block, or $12p^2$ per encoded qubit.

We conclude that the decoding error probability per encoded qubit is very nearly the same for any of these three codes.

With regard to rate and decoding complexity, our quantum convolutional and tail-biting codes have rate $1/3$, which is greater than that of any previous simple single-error-correcting quantum code, block or convolutional. Our decoding algorithm involves only a 9-entry table lookup, which is at least as simple as that of any previous quantum code.

Our convolutional code rate and error-correction capability (one error every two 3-blocks) are comparable to those of a $[6, 2, 3]$ block stabilizer code. However, no $[6, 2, 3]$ block stabilizer code exists (by the "quantum Hamming bound," since it would be a nondegenerate quantum MDS code).

Our tail-biting code is a $[9, 3, 3]$ block stabilizer code. We could obtain a code with the same parameters by shortening a $[21, 15, 3]$ quantum Hamming code. However, such a shortened code would not necessarily have such a simple structure as our tail-biting code, nor such a simple decoding algorithm.

F.   CSS-type convolutional codes

Similarly, as with CSS-type block codes, we may construct a CSS-type convolutional stabilizer code with minimum distance d from a classical self-orthogonal binary convolutional code $\mathcal{C}$ whose orthogonal convolutional code $\mathcal{C}^\perp$ has minimum Hamming distance $d^\perp = d$. Again, if the rate of $\mathcal{C}$ is $(n - k)/2n$, then the rate of $\mathcal{C}^\perp$ will be $(n + k)/2n$, and the rate of the convolutional stabilizer code will be k/n. We continue to focus on rate-$1/n$ codes.

As with CSS-type block codes, we will find that the parameters of CSS-type convolutional codes are in general poorer than those of codes based on general $\mathbb{F}_4$-linear codes, but that they may offer complexity advantages, since bit flip errors and phase flip errors may be decoded by two independent decodings of the binary convolutional code $\mathcal{C}^\perp$.

Example 3 (rate-1/3, single-error-correcting, CSS-type convolutional code). Consider the binary rate-1/3 convolutional code $\mathcal{C}$ whose generators are the shifts by an integral number of 3-blocks of the single basic generator $\mathbf{g} = (\ldots|000|111|100|110|000|\ldots)$, whose D-transform is $\mathbf{g}(D) = (1 + D + D^2, 1 + D^2, 1)$:

$$
\begin{array}{c}
\cdots \\
\begin{array}{c|ccc|ccc|ccc|ccc|ccc|c}
\cdots & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\
\cdots & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & \cdots
\end{array} \\
\cdots
\end{array}
$$

The "memory" of $\mathcal{C}$ is thus two 3-blocks (*i.e.*, its constraint length is $\nu = 2$).

The stabilizer group S is then generated by sequences of Pauli matrices that correspond to multiples of the above generators by ω and $\overline{\omega}$. Thus the generators of S are the shifts by an integral number of 3-blocks of two basic generators, $(\ldots|III|XXX|XII|XXI|III|\ldots)$ and $(\ldots|III|ZZZ|ZII|ZZI|III|\ldots)$. Since these stabilizers affect only bit flip and phase flip bits, respectively, the code $\mathcal{C}$ is the direct product of two independent binary codes that protect the bit flip and phase flip bits, respectively.

16

It is easy to verify that $\mathbf{g}$ is orthogonal to itself and to any shift of itself under the usual binary inner product. This suffices to show that $\mathcal{C}$ is self-orthogonal.

The generators of the orthogonal rate-2/3 binary convolutional code $\mathcal{C}^\perp$ are the shifts of two basic generators, $\mathbf{h}_1 = (\ldots|000|110|011|000|\ldots)$ and $\mathbf{h}_2 = (\ldots|000|001|110|000|\ldots)$, whose D-transforms are $\mathbf{h}_1(D) = (1, 1 + D, D)$ and $\mathbf{h}_2(D) = (D, D, 1)$, respectively:

$$
\ldots
$$

$$
\begin{array}{c|ccc|ccc|ccc|ccc|ccc|c}
\ldots & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots \\
\ldots & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots \\
\ldots & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & \ldots \\
\ldots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & \ldots
\end{array}
$$

$$
\ldots
$$

It is easy to verify that the minimum Hamming distance of $\mathcal{C}^\perp$ is $d^\perp = 3$, with the only weight-3 codewords being shifts of $\mathbf{h}_2$. Thus the rate-1/3 convolutional stabilizer code defined by $\mathcal{C}$ has minimum Hamming distance 3, and is single-error-correcting. $\square$

We now consider how to decode the Example 3 code. We will discuss only how to decode bit flip errors; phase flip errors may be corrected independently and identically.

For bit flip errors, we first measure each generator $\mathbf{g}_j$ of $\mathcal{C}$ to obtain a sequence $\mathbf{s}$ of binary syndromes $s_j = \langle \ell_1(\mathbf{E}), \mathbf{g}_j \rangle \in \mathbb{F}_2$, where $\langle \ell_1(\mathbf{E}), \mathbf{g}_j \rangle$ denotes the binary inner product of the generator $\mathbf{g}_j$ with the bit flip error label sequence $\ell_1(\mathbf{E})$, at a rate of one binary syndrome for each 3-block.

Again, instead of VA decoding the 4-state trellis of the rate-2/3 code $\mathcal{C}^\perp$, we may use a simple single-error-correction algorithm, as follows. As long as all syndromes are zero, we assume that no errors have occurred. When a nonzero syndrome s_j occurs, we assume that a single error has occurred in one of the three bit flip bits in block j, corresponding to a binary label 3-tuple $\mathbf{e}_j = \ell_1(\mathbf{E}_j)$. The three possible weight-1 error 3-tuples $\mathbf{e}_j$ lead to the following syndrome sequences:

$\mathbf{e}_j$	(s_j, s_{j+1}, s_{j+2})
100	$(1, 1, 1)$
010	$(1, 0, 1)$
001	$(1, 0, 0)$

Since the three syndrome sequences are distinct, we can map (s_{j+1}, s_{j+2}) to the corresponding single-error pattern $\mathbf{e}_j$ using a simple 3-entry table lookup, and then correct the error as indicated. (If (s_{j+1}, s_{j+2}) is not in the table— *i.e.,* if $(s_j, s_{j+1}, s_{j+2}) = (1, 1, 0)$— then we have detected a weight-2 error.)

We see that this simple algorithm can correct any single-error pattern $\mathbf{e}_j$, provided that there is no second error during blocks j through $j + 2$. The decoder synchronizes itself properly whenever a zero syndrome occurs, and subsequently can correct one error in every third block.

Finally, we consider tail-biting codes derived from the rate-1/3 convolutional code of Example 3. In this case, it turns out that a tail-biting interval of N 3-blocks results in no loss of minimum distance whenever $N \geq 5$.

Example 4 (rate-1/3, single-error-correcting, CSS-type tail-biting code). Taking the rate-2/3 binary convolutional code $\mathcal{C}^\perp$ of Example 3 and a tail-biting interval of five 3-blocks, we obtain the following 10 tail-biting generators:

$$
\begin{array}{ccc|ccc|ccc|ccc|ccc}
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{array}
$$

These generate a $(15, 10, 3)$ binary tail-biting code $\mathcal{B}^\perp$ with minimum distance $d^\perp = 3$. The orthogonal block code $\mathcal{B}$ is the corresponding $(15, 5)$ binary tail-biting code $\mathcal{B}$ derived from $\mathcal{C}$, which is necessarily self-orthogonal. Thus $\mathcal{B}$ specifies a $[15, 5, 3]$ CSS-type single-error-correcting block stabilizer code. $\square$

To decode this code, we can use the same simple decoding algorithm as for the corresponding convolutional code, but now on a circular time axis. If only a single error occurs, then the first syndrome 1 after two zeroes (on a circular time axis) identifies the 3-tuple block of the error, and the next two bits determine its position within the block, according to the 3-entry table above. Indeed, the fact that these 15 syndrome 5-tuples are distinct proves that single-error-correction is possible, and thus that $\mathcal{B}^\perp$ has minimum distance 3.

G.   Comparison of single-error-correcting CSS-type codes

We now compare our single-error-correcting CSS-type convolutional and tail-biting codes with the single-error-correcting CSS-type 7-qubit Steane code (Example B).

For decoding error probability, we again assume that the probability of an error in any qubit is p, independent of errors in other qubits. We do not take into account that, because of the independence of the two decoders, there are some weight-2 error patterns that are correctible (*e.g.*, X and Z); this would yield a minor improvement (a factor of 7/9) in our estimates.

For the 7-qubit Steane code, a decoding error may occur if there are 2 errors in any block, so the error probability is of the order of $\binom{7}{2}p^2 = 21p^2$ per block, or per encoded qubit.

For our rate-1/3 convolutional code, for each 3-block, a decoding error may occur if there are two errors in that 3-block, or one in that 3-block and one in the two subsequent 3-blocks. The error probability is therefore of the order of $(3 + 3 \cdot 6)p^2 = 21p^2$ per 3-block, or per encoded qubit.

Lastly, for our $[15, 5, 3]$ tail-biting code, a decoding error requires 2 errors in a block of 15 qubits, so the error probability is of the order of $\binom{15}{2}p^2 = 105p^2$ per block, or $21p^2$ per encoded qubit.

Again, we conclude that the decoding error probability is very nearly the same for any of these codes, and is about twice that of the $\mathbb{F}_4$-linear codes considered earlier.

With regard to rate and decoding complexity, our CSS-type convolutional and tail-biting codes again have rate 1/3, greater than that of any previous simple CSS-type single-error-correcting code. The decoder for Examples 3 and 4 only uses a 3-entry table lookup twice, and is arguably even simpler than the simple decoder for Examples 1 and 2.

18

Our convolutional code rate and error-correction capability (one error every three 3-blocks) are comparable to those of a $[9, 3, 3]$ CSS-type block code. However, it can be shown (by linear programming) that no $[9, 3, 3]$ CSS-type block code exists [15, 16].

Our tail-biting code is a $[15, 5, 3]$ CSS-type block code. We could obtain a code with the same parameters by shortening a $[31, 21, 3]$ CSS-type block code. However, such a shortened code would not necessarily have such a simple structure as our tail-biting code, nor such a simple decoding algorithm.

IV. Algebraic theory of $\mathbb{F}_4$-linear convolutional codes

In this section we give a brief presentation of the algebraic theory of $\mathbb{F}_4$-linear convolutional codes. We focus on those results which are most helpful in searching for good codes and for analyzing their performance. The theory of $\mathbb{F}_2$-linear convolutional codes is analogous. For further background, see [11], or any text that covers convolutional codes, such as [20].

A. Rate-$1/n$ convolutional codes

We have defined a rate-$1/n$ $\mathbb{F}_4$-linear shift-invariant convolutional code $\mathcal{C}$ with constraint length ν as the set of all $\mathbb{F}_4$-linear combinations of the shifts of a single basic finite generator sequence $\mathbf{g} = \{g_{jk}, 1 \leq j \leq n, 0 \leq k \leq \nu\}$ by an integral number of n-blocks.

It is helpful to use "D-transform" (generating function) notation, as is standard for convolutional codes. A sequence of n-blocks such as $\mathbf{g}$ is written as an n-tuple $\mathbf{g}(D) = \{g_j(D), 1 \leq j \leq n\}$ of D-transforms $g_j(D) = \sum_k g_{jk}D^k$, where D is an indeterminate, called the shift operator. For example, the generator $\mathbf{g} = (\dots, 000, 111, 1\omega\overline{\omega}, 000, \dots)$ is written as the polynomial 3-tuple

$$\mathbf{g}(D) = (g_1(D), g_2(D), g_3(D)) = (1 + D, 1 + \omega D, 1 + \overline{\omega} D),$$

where we have aligned the first nonzero block with the index $k = 0$.

A shift of $\mathbf{g}(D)$ by an integral number ℓ of n-blocks is then represented by the D-transform $D^\ell \mathbf{g}(D) = \{D^\ell g_j(D), 1 \leq j \leq n\}$, where $D^\ell g_j(D) = \sum_k g_{jk}D^{k+\ell}$. For example, a one-block shift of the generator $\mathbf{g}(D)$ is represented by $D\mathbf{g}(D) = (D + D^2, D + \omega D^2, D + \overline{\omega} D^2)$.

The rate-$1/n$ convolutional code $\mathcal{C}$ is then the set of "all" $\mathbb{F}_4$-linear combinations of the shifted generators $\{D^\ell \mathbf{g}(D), \ell \in \mathbb{Z}\}$; i.e.,

$$\mathcal{C} = \{\sum_{\ell \in \mathbb{Z}} u_\ell D^\ell \mathbf{g}(D) \mid u_\ell \in \mathbb{F}_4\} = \{u(D)\mathbf{g}(D)\},$$

where we have defined $u(D)$ as the D-transform $\sum_\ell u_\ell D^\ell$ of the coefficient sequence $\{u_\ell\}$.

We have put "all" in quotation marks because usually, for technical reasons, the coefficient sequence $\{u_\ell\}$ is required to have only finitely many nonzero u_ℓ with negative indices $\ell < 0$. Such a sequence is called a Laurent power series, and the set of all Laurent power series in D over $\mathbb{F}_4$ is denoted by $\mathbb{F}_4((D))$. In short,

$$\mathcal{C} = \{u(D)\mathbf{g}(D) \mid u(D) \in \mathbb{F}_4((D))\}.$$

The set $\mathbb{F}_4((D))$ is shift-invariant; i.e., $D^\ell \mathbb{F}_4((D)) = \mathbb{F}_4((D))$ for any $\ell \in \mathbb{Z}$. Consequently, $\mathcal{C}$ is shift-invariant: $D^\ell \mathcal{C} = \mathcal{C}$.

19

The set $\mathbb{F}_4((D))$ is actually a field under sequence (componentwise) addition and sequence (polynomial) multiplication (*i.e.*, $a(D)b(D) = \sum_k D^k \sum_{k'} a_{k'} b_{k-k'}$). In particular, every nonzero sequence $u(D) \in \mathbb{F}_4((D))$ has a multiplicative inverse $1/u(D)$, which may be found by polynomial long division. For example, the inverse of $1 + D$ is $1 + D + D^2 + \cdots$.

The set $\mathbb{F}_4((D))^n$ of all n-tuples of Laurent power series is an n-dimensional vector space over the field $\mathbb{F}_4((D))$. A rate-$1/n$ code $\mathcal{C}$ is therefore simply a one-dimensional shift-invariant subspace of $\mathbb{F}_4((D))^n$. Any nonzero code sequence $u(D)\mathbf{g}(D) \in \mathcal{C}$ may thus be taken as a generator for $\mathcal{C}$.

We wish to choose a canonical generator $\mathbf{g}(D) \in \mathcal{C}$ that has the most desirable properties. For most purposes, the best choice is a polynomial code sequence $\mathbf{g}(D) \in \mathcal{C}$ of least degree, where we define $\deg \mathbf{g}(D) = \max_j \{\deg g_j(D)\}$. Given any nonzero polynomial code sequence $\mathbf{c}(D) = (c_1(D), \ldots, c_n(D)) \in \mathcal{C}$, a minimum-degree polynomial generator is $\mathbf{g}(D) = \mathbf{c}(D)/d(D)$, where $d(D)$ is the greatest common divisor of the polynomials $c_j(D)$. Conversely, a polynomial generator $\mathbf{g}(D)$ is minimum-degree if and only if its components $g_j(D)$ are relatively prime. The minimum-degree polynomial generator $\mathbf{g}(D)$ is thus unique up to multiplication by nonzero scalars in $\mathbb{F}_4$.

A minimum-degree polynomial generator $\mathbf{g}(D)$ has the following properties [11, 20]:

- A code sequence $\mathbf{c}(D) = u(D)\mathbf{g}(D) \in \mathcal{C}$ is polynomial if and only if $u(D)$ is polynomial; *i.e.*, the set of all polynomial code sequences is $\{u(D)\mathbf{g}(D), u(D) \in \mathbb{F}_4[D]\}$, where $\mathbb{F}_4[D]$ denotes the set of all polynomials in D over $\mathbb{F}_4$. This is called the *noncatastrophic property*.

- The constraint length $\nu = \deg \mathbf{g}(D)$ is minimized.

B. Orthogonality

The *Hermitian inner product* of two Laurent power series $a(D), b(D) \in \mathbb{F}_4((D))$ is defined as

$$\langle a(D), b(D) \rangle = \sum_{k \in \mathbb{Z}} a_k^\dagger b_k.$$

The sum is well defined if and only if there are only finitely many nonzero summands $a_k^\dagger b_k$.

The Hermitian inner product of two Laurent n-tuples $\mathbf{a}(D), \mathbf{b}(D) \in \mathbb{F}_4((D))^n$ is defined as

$$\langle \mathbf{a}(D), \mathbf{b}(D) \rangle = \sum_{j=1}^{n} \langle a_j(D), b_j(D) \rangle.$$

The *cross-correlation sequence* of $\mathbf{a}(D), \mathbf{b}(D)$ is defined as

$$R_{\mathbf{ab}}(D) = \sum_{j=1}^{n} a_j^\dagger(D^{-1}) b_j(D),$$

again assuming well-defined products $a_j^\dagger(D^{-1}) b_j(D)$. Thus

$$R_{\mathbf{ab}, \ell} = \sum_j \sum_k a_{jk}^\dagger b_{j,k-\ell} = \langle \mathbf{a}(D), D^\ell \mathbf{b}(D) \rangle.$$

Therefore a sequence $\mathbf{a}(D)$ is orthogonal to all shifts $D^\ell \mathbf{b}(D)$ of a sequence $\mathbf{b}(D)$ if and only if

$$R_{\mathbf{ab}}(D) = 0;$$

i.e., if and only if all cross-correlation terms $R_{\mathbf{ab}, \ell}$ are equal to zero.

A rate-$1/n$ linear shift-invariant convolutional code $\mathcal{C}$ with generator $\mathbf{g}(D)$ is thus self-orthogonal if and only if

$$R_{\mathbf{gg}}(D) = \sum_{j=1}^{n} g_j^\dagger(D^{-1}) g_j(D) = 0.$$

Example 1. The sequence $\mathbf{g}(D) = (1 + D, 1 + \omega D, 1 + \overline{\omega} D)$ is orthogonal to all of its shifts since

$$
\begin{aligned}
R_{\mathbf{gg}}(D) &= (1 + D^{-1})(1 + D) + (1 + \overline{\omega}D^{-1})(1 + \omega D) + (1 + \omega D^{-1})(1 + \overline{\omega} D) \\
&= (D^{-1} + D) + (\overline{\omega}D^{-1} + \omega D) + (\omega D^{-1} + \overline{\omega} D) = 0.
\end{aligned}
$$

Thus the convolutional code $\mathcal{C}$ generated by all shifts of $\mathbf{g}(D)$ is self-orthogonal. $\square$

Example 3. The sequence $\mathbf{g}'(D) = (1 + D + D^2, 1 + D^2, 1)$ is orthogonal to all of its shifts since

$$
\begin{aligned}
R_{\mathbf{g}'\mathbf{g}'}(D) &= (1 + D^{-1} + D^{-2})(1 + D + D^2) + (1 + D^{-2})(1 + D^2) + 1 \\
&= (D^{-2} + 1 + D^2) + (D^{-2} + D^2) + 1 = 0.
\end{aligned}
$$

Thus the convolutional code $\mathcal{C}'$ generated by all shifts of $\mathbf{g}'(D)$ is self-orthogonal. $\square$

The orthogonal code $\mathcal{C}^\perp$ to a rate-$1/n$ convolutional code $\mathcal{C}$ with generator $\mathbf{g}(D)$ is the set of all sequences $\mathbf{a}(D)$ that are orthogonal to all shifts $D^k \mathbf{g}(D)$, and thus are uncorrelated with $\mathbf{g}(D)$— *i.e.,* such that $R_{\mathbf{ag}}(D) = 0$. It follows that $\mathcal{C}^\perp$ is a rate-$(n-1)/n$ $\mathbb{F}_4$-linear shift-invariant convolutional code— *i.e.,* an $(n-1)$-dimensional shift-invariant subspace of $\mathbb{F}_4((D))^n$.

It is again desirable to choose as generators for $\mathcal{C}^\perp$ a set of $n-1$ linearly independent minimum-degree polynomial generators $\mathbf{h}_i(D), 1 \leq i \leq n-1$, such that $R_{\mathbf{h}_i \mathbf{g}}(D) = 0$. This can be done by an exhaustive search for low-degree orthogonal polynomial sequences, or by various algebraic methods. Again, such a minimal-degree generator set has the following properties [11, 20]:

- A code sequence $\mathbf{c}(D) = \sum_i u_i(D)\mathbf{h}_i(D) \in \mathcal{C}^\perp$ is polynomial if and only if $\mathbf{u}(D)$ is polynomial; *i.e.,* the generator set is *noncatastrophic*.

- The total constraint length $\nu^\perp = \sum_i \nu_i^\perp = \sum_i \deg \mathbf{h}_i(D)$ of the generator set is minimized, and is equal to the constraint length ν of $\mathcal{C}$ [13]. This latter property may be used to check whether a set of independent orthogonal generators is a minimal-degree set.

Example 1. For the code $\mathcal{C}$ generated by the degree-1 generator $\mathbf{g}(D) = (1+D, 1+\omega D, 1+\overline{\omega} D)$, the two sequences $\mathbf{h}_1(D) = \mathbf{g}(D)$ and $\mathbf{h}_2(D) = (\overline{\omega}, \omega, 1)$ are independent, are orthogonal to $\mathbf{g}(D)$, and have degrees $\nu_1^\perp = 1$ and $\nu_2^\perp = 0$ that sum to $\nu = 1$. Therefore $\{\mathbf{h}_1(D), \mathbf{h}_2(D)\}$ is a minimal-degree set of generators for the orthogonal rate-2/3 code $\mathcal{C}^\perp$. $\square$

Example 3. For the code $\mathcal{C}'$ generated by the degree-2 generator $\mathbf{g}'(D) = (1 + D + D^2, 1 + D^2, 1)$, the two sequences $\mathbf{h}_1'(D) = (1, 1 + D, D)$ and $\mathbf{h}_2'(D) = (D, D, 1)$ are independent, are orthogonal to $\mathbf{g}'(D)$, and have degrees $\nu_1^\perp = 1$ and $\nu_2^\perp = 1$ that sum to $\nu = 2$. Therefore $\{\mathbf{h}_1'(D), \mathbf{h}_2'(D)\}$ is a minimal-degree set of generators for the orthogonal rate-2/3 code $\mathcal{C}'^\perp$. $\square$

In view of the noncatastrophic property, the minimum Hamming distance $d^\perp$ of the orthogonal code $\mathcal{C}^\perp$ is the minimum weight of any nonzero polynomial code sequence $\sum_i u_i(D)\mathbf{h}_i(D)$, where $\{u_i(D), 1 \leq i \leq n-1\}$ is any set of polynomial sequences. For simple codes, the minimum-weight nonzero sequence will be a polynomial sequence of low degree, and will often be obvious by inspection. For instance, for both of our example codes, one generator $\mathbf{h}_i(D)$ has weight 3, and it is easy to see that no nonzero sequence in $\mathcal{C}^\perp$ can have weight less than 3, so $d^\perp = 3$.

C.   Convolutional code symmetries

In searching for generators of good codes, it is helpful to observe that there are certain symmetries that preserve the most important properties of convolutional codes. A symmetry that converts $\mathbf{g}(D), \mathbf{h}(D) \in \mathbb{F}_4((D))^n$ to $\mathbf{g}'(D), \mathbf{h}'(D) \in \mathbb{F}_4((D))^n$ will be called *weight-preserving* if Hamming weights are preserved, and *orthogonality-preserving* if $R_{\mathbf{gh}}(D) = 0$ implies $R_{\mathbf{g}'\mathbf{h}'}(D) = 0$.

Theorem 1 (Convolutional code symmetries) *The following symmetries of $\mathbb{F}_4((D))^n$ are both weight-preserving and orthogonality-preserving:*

1. *Multiplication of any component $g_j(D)$ by any monomial αD^ℓ, $\alpha \neq 0, \ell \in \mathbb{Z}$.*

2. *Conjugation: $\mathbf{g}(D) \to \mathbf{g}^\dagger(D)$.*

3. *Time-reversal: $\mathbf{g}(D) \to \mathbf{g}(D^{-1})$.*

4. *Modulation: $\mathbf{g}(D) \to \mathbf{g}(\alpha D)$ for any nonzero scalar $\alpha \in \mathbb{F}_4$.*

5. *Permutation of the components $g_j(D)$.*

Proof. It is obvious that each symmetry is weight-preserving.

To show that each symmetry is orthogonality-preserving, we argue in each case that if $R_{\mathbf{gh}}(D) = 0$ and $\mathbf{g}(D)$ and $\mathbf{h}(D)$ are changed to $\mathbf{g}'(D)$ and $\mathbf{h}'(D)$, then $R_{\mathbf{g}'\mathbf{h}'}(D) = 0$, using

$$R_{\mathbf{gh}}(D) = \sum_{j=1}^{n} g_j^\dagger(D^{-1}) h_j(D).$$

1. If $g_j(D), h_j(D) \to \alpha D^\ell g_j(D), \alpha D^\ell h_j(D)$, then $R_{\mathbf{g}'\mathbf{h}'}(D) = R_{\mathbf{gh}}(D)$, since $\alpha^\dagger D^{-\ell} g_j^\dagger(D^{-1}) \alpha D^\ell h_j(D) = g_j^\dagger(D^{-1}) h_j(D)$.

2. If $\mathbf{g}(D), \mathbf{h}(D) \to \mathbf{g}^\dagger(D), \mathbf{h}^\dagger(D)$, then $R_{\mathbf{gh}}(D) \to R_{\mathbf{gh}}^\dagger(D)$.

3. If $\mathbf{g}(D), \mathbf{h}(D) \to \mathbf{g}(D^{-1}), \mathbf{h}(D^{-1})$, then $R_{\mathbf{gh}}(D) \to R_{\mathbf{gh}}(D^{-1})$.

4. If $\mathbf{g}(D), \mathbf{h}(D) \to \mathbf{g}(\alpha D), \mathbf{h}(\alpha D)$, then $R_{\mathbf{gh}}(D) \to R_{\mathbf{gh}}(\alpha D)$.

5. Permutation of the $g_j(D)$ and $h_j(D)$ in the same way does not affect $R_{\mathbf{gh}}(D)$. $\square$

In particular, if $\mathbf{g}(D)$ is a self-orthogonal generator, then the modified generator $\mathbf{g}'(D)$ under any of these symmetries is self-orthogonal.

The first symmetry shows that, without loss of generality, we may assume that all component generators $g_j(D)$ are *monic* polynomials; *i.e.,* the zero-degree coefficient $g_{j,0}$ of $g_j(D)$ is 1.

Example 1. It is easy to see that a degree-1 generator is self-orthogonal if and only if it is equivalent to $\mathbf{g}(D) = (1 + D, 1 + \omega D, 1 + \overline{\omega} D)$ under one of these symmetries (see Section V). $\square$

Example 3. The degree-2 binary generator $\mathbf{g}'(D) = (1 + D + D^2, 1 + D^2, 1)$ is invariant under conjugation, and effectively invariant under time-reversal. There are 6 equivalent binary generators under component permutations. No further equivalent binary generators are produced by the symmetry $\mathbf{g}(D) \to \mathbf{g}(\alpha D)$. As we will see in Section V, $\mathbf{g}'(D)$ is the unique monic degree-2 binary self-orthogonal generator, up to component permutations. $\square$

Example 5. The degree-2 generator $\mathbf{g}''(D) = (1+D+D^2, 1+\omega D+D^2, 1+D)$ satisfies $R_{\mathbf{g}''\mathbf{g}''}(D) = 0$, so the convolutional code $\mathcal{C}''$ generated by all shifts of $\mathbf{g}''(D)$ is self-orthogonal (see Section VI). A minimal-degree generator set for the orthogonal code $(\mathcal{C}'')^{\perp}$ is $\{\mathbf{h}_1(D) = (\omega D, \overline{\omega}D, 1+D), \mathbf{h}_2(D) = (1, 1+\overline{\omega}D, 1+\overline{\omega}D)\}$. The minimum distance of $(\mathcal{C}'')^{\perp}$ is $d^{\perp} = 4$. There are 6 equivalent generators to $\mathbf{g}''(D) = (1+D+D^2, 1+\omega D+D^2, 1+D)$ under conjugation and the symmetry $\mathbf{g}(D) \to \mathbf{g}(\alpha D)$, or 36 if component permutations are also considered. $\square$

V. Rate-$1/n$ single-error-correcting codes

Using the theoretical development of Section IV, it is straightforward to find all possible short-constraint-length, single-error-correcting, convolutional stabilizer codes based on both binary and $\mathbb{F}_4$-linear rate-$1/n$ convolutional codes.

In order that a rate-$1/n$ linear shift-invariant convolutional code $\mathcal{C}$ generated by $\mathbf{g}(D) = (g_1(D), \ldots, g_n(D))$ has an orthogonal code $\mathcal{C}^{\perp}$ with minimum distance $d^{\perp} \geq 3$, it is necessary and sufficient that all component generator polynomials $g_j(D)$ be linearly independent, so that no weight-2 error pattern can cause a zero syndrome. If all generator polynomials $g_j(D)$ are restricted to be monic, then this reduces to the requirement that all $g_j(D)$ be different.

To find single-error-correcting stabilizer codes, it therefore suffices to list all monic polynomials $g(D)$ of low degree, with their autocorrelation functions $R_{gg}(D) = g^{\dagger}(D^{-1})g(D)$, and to identify all subsets $\mathbf{g}(D)$ of size n such that $R_{\mathbf{g}\mathbf{g}}(D) = \sum_{j=1}^{n} R_{g_j g_j}(D) = 0$.

In Table I, we therefore list all binary polynomials $g(D)$ of degree 3 or less, with the non-negative-degree components $[R_{gg}(D)]_{0+}$ of their autocorrelation functions (the negative-degree components are symmetric). For $3 \leq n \leq 8$, subsets of size n are identified such that the corresponding autocorrelation functions sum to zero. There exists a unique binary self-orthogonal rate-$1/n$ convolutional code with constraint length $\nu = 2$: namely, the rate-1/3 Example 3 code. Seven further codes are listed with constraint length $\nu = 3$ and rates from 1/4 down to 1/8. (It is easy to verify that none of these generator sets is catastrophic.) In turn, these binary codes yield single-error-correcting CSS-type convolutional stabilizer codes with quantum code rates ranging from 1/3 up to 6/8.

$g(D)$	$[R_{gg}(D)]_{0+}$	$\frac{1}{3}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{6}$	$\frac{1}{8}$
1	1	$*$	$*$		$*$	$*$		$*$	$*$
$1+D$	D			$*$		$*$	$*$	$*$	$*$
$1+D^2$	D^2	$*$	$*$		$*$	$*$		$*$	$*$
$1+D+D^2$	$1+D^2$	$*$		$*$	$*$			$*$	$*$
$1+D^3$	D^3			$*$		$*$	$*$	$*$	$*$
$1+D+D^3$	$1+D+D^2+D^3$		$*$		$*$	$*$	$*$		$*$
$1+D^2+D^3$	$1+D+D^2+D^3$			$*$	$*$		$*$		$*$
$1+D+D^2+D^3$	$D+D^3$		$*$				$*$	$*$	$*$

Table I. Self-orthogonal binary rate-$1/n$ convolutional codes.

To decode these rate-$1/n$ CSS-type codes, $n \geq 4$, as with our rate-1/3 CSS-type code, bit flip and phase flip bits may be decoded independently in two binary decoders. Since there are only n possible single-error patterns $\mathbf{e}_j$ in the jth n-block, decoding requires only an n-entry table lookup.

Decoding will succeed if there is no second error during blocks j through $j + 3$; *i.e.,* each decoder can correct 1 error in every 4 n-blocks.

The minimum-length single-error-correcting tail-biting code that can be derived from any of these codes is easily determined by finding the minimum tail-biting length for which all cyclic shifts of all n single-error syndromes are distinct. For the eight codes listed in Table I, the minimum-length corresponding tail-biting codes are listed in Table II. Additionally, we give the number $N_{d^\perp}$ of words of weight $d^\perp$ in $\mathcal{B}^\perp$, and an upper bound d_{CSS} on the minimum distance of a CSS-type code. Again, these codes may be decoded by the same simple n-entry table lookup algorithm, operating on a circular time axis.

rate	ν	$N_{d^\perp}$	$\mathcal{B}$	$\mathcal{B}^\perp$	stabilizer code	d_{CSS}
1/3	2	2	$(15, 5, 6)$	$(15, 10, 3)$	$[15, 5, 3]$	3
1/4	3	4	$(20, 5, 8)$	$(20, 15, 3)$	$[20, 10, 3]$	3
1/4	3	2	$(20, 5, 8)$	$(20, 15, 3)$	$[20, 10, 3]$	3
1/5	3	6	$(30, 6, 12)$	$(30, 24, 3)$	$[30, 18, 3]$	3
1/5	3	9	$(35, 7, 10)$	$(35, 28, 3)$	$[35, 21, 3]$	3–4
1/5	3	3	$(35, 7, 14)$	$(35, 28, 3)$	$[35, 21, 3]$	3–4
1/6	3	15	$(42, 7, 14)$	$(42, 35, 3)$	$[42, 28, 3]$	3–4
1/8	3	28	$(56, 7, 20)$	$(56, 49, 3)$	$[56, 42, 3]$	3–4

Table II. CSS-type rate-$1/n$ tail-biting codes.

Similarly, in Table III we list all 16 monic quaternary polynomials $g(D)$ of degree 2 or less, along with the non-negative-degree components $[R_{gg}(D)]_{0^+}$ of their autocorrelation functions. For $3 \leq n \leq 16$, certain subsets of size n are identified such that the sums of the corresponding autocorrelation functions are zero. There exists a unique $\mathbb{F}_4$-linear self-orthogonal rate-$1/n$ convolutional code with constraint length $\nu = 1$: namely, the rate-1/3 Example 1 code. Five further codes are listed with constraint length $\nu = 2$ and rates 1/4, 1/5, 1/6, 1/10 and 1/16. (Again, none of these generator sets is catastrophic.) These codes yield single-error-correcting convolutional stabilizer codes with quantum code rates ranging from 1/3 up to 14/16.

In this case, since there are only $3n$ possible quaternary single-error patterns $\mathbf{E}_j$ in the jth n-block, decoding requires only a $3n$-entry table lookup. Decoding will succeed if there is no second error during blocks j through $j + 2$; *i.e.,* each decoder can correct 1 error in every 3 n-blocks.

Again, the minimum-length corresponding single-error-correcting tail-biting code may be determined by finding the minimum tail-biting length for which all cyclic shifts of all $3n$ single-error syndromes are distinct. For the six codes listed in Table III, these tail-biting codes are listed in Table IV. Additionally, we give the number $N_{d^\perp}$ of words of weight $d^\perp$ in $\mathcal{B}^\perp$, and an upper bound d_{opt} on the minimum distance of a general quantum code. Most of the codes meet this bound on minimum distance. Again, these codes may be decoded by the same simple $3n$-entry table lookup algorithm, operating on a circular time axis.

$g(D)$	$[R_{gg}(D)]_{0+}$	$\frac{1}{3}$	$\frac{1}{4}$	$\frac{1}{5}$	$\frac{1}{6}$	$\frac{1}{10}$	$\frac{1}{16}$
1	1		$*$			$*$	$*$
$1+D$	D	$*$	$*$	$*$	$*$		$*$
$1+\omega D$	ωD	$*$		$*$	$*$		$*$
$1+\overline{\omega}D$	$\overline{\omega}D$	$*$		$*$	$*$		$*$
$1+D^2$	D^2		$*$		$*$		$*$
$1+\omega D^2$	ωD^2				$*$		$*$
$1+\overline{\omega}D^2$	$\overline{\omega}D^2$				$*$		$*$
$1+D+D^2$	$1+D^2$					$*$	$*$
$1+D+\omega D^2$	$1+\overline{\omega}D+\omega D^2$					$*$	$*$
$1+D+\overline{\omega}D^2$	$1+\omega D+\overline{\omega}D^2$					$*$	$*$
$1+\omega D+D^2$	$1+D+D^2$	$*$	$*$			$*$	$*$
$1+\omega D+\omega D^2$	$1+\overline{\omega}D+\omega D^2$					$*$	$*$
$1+\omega D+\overline{\omega}D^2$	$1+\overline{\omega}D^2$					$*$	$*$
$1+\overline{\omega}D+D^2$	$1+D+D^2$				$*$	$*$	$*$
$1+\overline{\omega}D+\omega D^2$	$1+\omega D^2$					$*$	$*$
$1+\overline{\omega}D+\overline{\omega}D^2$	$1+\omega D+\overline{\omega}D^2$					$*$	$*$

Table III. Self-orthogonal $\mathbb{F}_4$-linear rate-$1/n$ convolutional codes.

rate	ν	$N_{d^\perp}$	$\mathcal{B}$	$\mathcal{B}^\perp$	stabilizer code	d_{opt}
$1/3$	1	3	$(9,3)$	$(9,6,3)$	$[9,3,3]$	3
$1/4$	2	12	$(20,5)$	$(20,15,3)$	$[20,10,3]$	4
$1/5$	2	15	$(15,3)$	$(15,12,3)$	$[15,9,3]$	3
$1/6$	2	33	$(30,5)$	$(30,25,3)$	$[30,20,3]$	4
$1/10$	2	108	$(40,4)$	$(40,36,3)$	$[40,32,3]$	3
$1/16$	2	600	$(80,5)$	$(80,75,3)$	$[80,70,3]$	3–4

Table IV. $\mathbb{F}_4$-linear rate-$1/n$ tail-biting codes.

VI. Rate-1/3 codes with $d > 3$

We have performed a computer search for both binary and $\mathbb{F}_4$-linear self-orthogonal convolutional codes with constraint lengths up to $\nu = 12$ and $\nu = 6$, respectively. The best codes found have orthogonal codes with Hamming distances $d^\perp = 10$ and $d^\perp = 9$, respectively.

   We examined one code from each equivalence class under the symmetries of Theorem 1. In particular, we considered only monic generators $\mathbf{g}(D)$. We eliminated catastrophic generators. For each code found, we found a minimal-degree pair of orthogonal generators, $\mathbf{h}_1(D)$ and $\mathbf{h}_2(D)$, such that $\deg \mathbf{h}_1(D) + \deg \mathbf{h}_2(D) = \deg \mathbf{g}(D)$. We then found the minimum distance of the orthogonal code by a trellis search. For the $\mathbb{F}_4$-linear codes, we found the notation of Jönsson [21] to be helpful.

   Table V shows the best binary codes found for constraint lengths $2 \le \nu \le 12$. The best code is the one whose orthogonal code has the greatest minimum distance $d^\perp$; to resolve ties, minimization of the number $N_{d^\perp}$ of code sequences of weight $d^\perp$ is used as a secondary criterion. A unique best

code (up to the symmetries of Theorem 1) was found for $\nu = 2, 3, 5, 6, 7, 10, 11$ and 12. For brevity, we represent polynomials by their coefficient sequences; *e.g.,* $1101 = 1 + D + D^3$.

ν	$\mathbf{g}(D)$			$\mathbf{h}_1(D), \mathbf{h}_2(D)$			$d^\perp$	$N_{d^\perp}$
2	1	101	111	11	10	11	3	2
				01	11	10		
3	111	1101	1111	101	101	100	4	3
				01	10	11		
4	1111	11001	10101	101	010	001	4	1
				111	111	100		
4	1101	10011	11011	0001	1000	1001	4	1
				11	11	10		
4	1101	11001	11011	1001	1001	1000	4	1
				01	10	11		
5	11111	101101	101111	1011	0001	0100	5	1
				111	110	111		
6	111001	1100111	1001111	1111	1000	1101	6	2
				1101	0011	0110		
7	1010001	11110101	11100011	10111	00001	01100	7	7
				1101	1110	1101		
8	11010101	110100101	111111011	10101	11010	11011	7	1
				11101	01001	00100		
8	11001001	111000101	100110101	01011	11000	11101	7	1
				11011	00111	01000		
8	10100001	111011101	110111111	00111	11010	10011	7	1
				10001	11011	11100		
8	10110001	111110011	101101111	100101	000011	011000	7	1
				1001	1110	1011		
9	101000001	1100111101	1110011111	110111	101001	100000	8	3
				11101	00110	01101		
9	111011011	1011000001	1000111111	111111	101100	100001	8	3
				10011	01111	00010		
10	10111110101	11110101001	10101110110	1011111	1011000	0101111	9	8
				11001	10111	11000		
11	100001010111	110010101011	101110000010	1110011	1011101	1010100	9	1
				010011	000110	110101		
12	1110010000010	1101110010011	1011111000111	0111111	1001010	1000111	10	5
				1010001	1111001	1010100		

Table V. Self-orthogonal binary linear rate-1/3 convolutional codes.

Similarly, Table VI shows the best $\mathbb{F}_4$-linear codes found for constraint lengths $1 \le \nu \le 6$. For $1 \le \nu \le 5$, the codes are unique up to equivalence.

ν	$\mathbf{g}(D)$			$\mathbf{h}_1(D)$			$\mathbf{h}_2(D)$			$d^\perp$	$N_{d^\perp}$
1	11	1ω	$1\overline{\omega}$	$\overline{\omega}$	ω	1	11	1ω	$1\overline{\omega}$	3	3
2	111	$1\omega 1$	110	0ω	$0\overline{\omega}$	11	10	$1\overline{\omega}$	$1\overline{\omega}$	4	12
3	1001	$111\overline{\omega}$	$1\omega\overline{\omega}\omega$	10	$1\overline{\omega}$	$1\overline{\omega}$	$\omega 0\omega$	$1\overline{\omega}1$	$00\overline{\omega}$	5	3
4	$1\omega\overline{\omega}\omega 1$	$1\overline{\omega}01\overline{\omega}$	$111\omega\omega$	$\overline{\omega}\omega 1$	$\overline{\omega}\omega$	$\overline{\omega}\omega 1$	$\overline{\omega}\overline{\omega}1$	$\omega 11$	$0\overline{\omega}$	6	3
5	$11\omega 0\overline{\omega}1$	$11\overline{\omega}10\overline{\omega}$	$1\overline{\omega}\omega\omega\omega\omega$	$\overline{\omega}\omega 1$	$10\overline{\omega}$	$\omega\overline{\omega}\omega$	$\overline{\omega}10\overline{\omega}$	$\omega 0\overline{\omega}1$	$00\omega\omega$	8	75
6	$1\overline{\omega}\omega 1\omega 0\omega$	$11\overline{\omega}00\overline{\omega}\overline{\omega}$	$100\omega 1\overline{\omega}1$	$\overline{\omega}\omega 11$	$11\overline{\omega}$	0101	$\omega\omega\overline{\omega}1$	$10\omega 1$	$\overline{\omega}\overline{\omega}\overline{\omega}$	9	78
6	$1\overline{\omega}\omega 1\omega 0\overline{\omega}$	$1\omega 0\omega\overline{\omega}\omega\omega$	$11\omega 0\omega\overline{\omega}1$	$\omega\overline{\omega}\omega 1$	$0\omega 1$	$\overline{\omega}101$	$1\omega 01$	$\overline{\omega}1\omega 1$	$0\omega\overline{\omega}$	9	78
6	$1\omega 1\overline{\omega}\overline{\omega}0\overline{\omega}$	$1\overline{\omega}\omega\overline{\omega}\overline{\omega}11$	$1001\omega 1\omega$	$\omega\overline{\omega}11$	$\overline{\omega}\overline{\omega}$	$0\omega\omega 1$	$10\overline{\omega}1$	0111	$\overline{\omega}\overline{\omega}\overline{\omega}$	9	78
6	$11\omega 110\overline{\omega}$	$10\omega\omega 0\omega\omega$	$1\overline{\omega}1\overline{\omega}\omega\overline{\omega}1$	$1\omega\omega 1$	$\overline{\omega}0\overline{\omega}$	$01\omega 1$	$\omega 111$	$01\overline{\omega}1$	$\overline{\omega}0\overline{\omega}$	9	78

Table VI. Self-orthogonal $\mathbb{F}_4$-linear rate-1/3 convolutional codes.

For each of these QCCs, we also found the minimum-length corresponding tail-biting code that preserves minimum distance. We first found the ==minimum weight per cycle ("slope") α== of the orthogonal code, and then evaluated the upper bound $L \leq \lceil d^\perp/\alpha \rceil$ of Handlery *et al.* [19] on the minimum number L of 3-blocks in the corresponding rate-2/3 tail-biting code. Using MAGMA [4], we then found the minimum distances of tail-biting codes of up to this length to determine L.

Table VII shows the minimum-length tail-biting codes corresponding to the rate-1/3 binary convolutional codes of Table V.

ν	$d^\perp$	α	$\lceil d^\perp/\alpha \rceil$	$N_{d^\perp}$	$\mathcal{B}$	$\mathcal{B}^\perp$	stabilizer code	d_{CSS}
2	3	1/2	6	2	$(15,5)$	$(15,10,3)$	$[15,5,3]$	3
3	4	1/2	8	3	$(21,7)$	$(21,14,4)$	$[21,7,4]$	4
4	4	2/6	12	1	$(24,8)$	$(24,16,4)$	$[24,8,4]$	4
4	4	1/3	12	1	$(21,7)$	$(21,14,4)$	$[21,7,4]$	4
4	4	1/3	12	1	$(21,7)$	$(21,14,4)$	$[21,7,4]$	4
5	5	1/3	15	1	$(39,13)$	$(39,26,5)$	$[39,13,5]$	5–6
6	6	4/15	23	2	$(54,18)$	$(54,36,6)$	$[54,18,6]$	6–8
7	7	5/18	26	7	$(63,21)$	$(63,42,7)$	$[63,21,7]$	7–10
8	7	3/11	26	1	$(69,23)$	$(69,46,7)$	$[69,23,7]$	8–10
8	7	4/16	28	1	$(69,23)$	$(69,46,7)$	$[69,23,7]$	8–10
8	7	3/14	33	1	$(60,20)$	$(60,40,7)$	$[60,20,7]$	7–9
8	7	5/20	28	1	$(63,21)$	$(63,42,7)$	$[63,21,7]$	7–10
9	8	7/31	36	3	$(84,28)$	$(84,56,8)$	$[84,28,8]$	8–12
9	8	10/45	36	3	$(69,23)$	$(69,46,8)$	$[69,23,8]$	8–10
10	9	9/41	41	8	$(99,33)$	$(99,66,9)$	$[99,33,9]$	9–14
11	9	11/52	43	1	$(105,35)$	$(105,70,9)$	$[105,35,9]$	10–15
12	10	4/22	55	5	$(114,38)$	$(114,76,10)$	$[114,38,10]$	10–16

Table VII. CSS-type rate-1/3 ==tail-biting== codes.

Similarly, Table VIII shows the minimum-length tail-biting codes corresponding to the rate-1/3 $\mathbb{F}_4$-linear convolutional codes of Table VI.

ν	$d^\perp$	α	$\lceil d^\perp/\alpha \rceil$	$N_{d^\perp}$	$\mathcal{B}$	$\mathcal{B}^\perp$	stabilizer code	d_{opt}
1	3	1/1	3	3	$(9,3)$	$(9,6,3)$	$[9,3,3]$	3
2	4	2/3	6	12	$(15,5)$	$(15,10,4)$	$[15,5,4]$	4
3	5	1/3	15	3	$(24,8)$	$(24,16,5)$	$[24,8,5]$	5–6
4	6	1/3	18	3	$(39,13)$	$(39,26,6)$	$[39,13,6]$	7–10
5	8	6/14	19	75	$(45,15)$	$(45,30,8)$	$[45,15,8]$	8–11
6	9	6/42	63	78	$(57,19)$	$(57,38,9)$	$[57,19,9]$	9–14
6	9	51/132	24	78	$(54,18)$	$(54,36,9)$	$[54,18,9]$	9–13
6	9	18/45	23	78	$(57,19)$	$(57,38,9)$	$[57,19,9]$	9–14
6	9	8/20	23	78	$(57,19)$	$(57,38,9)$	$[57,19,9]$	9–14

Table VIII. $\mathbb{F}_4$-linear rate-1/3 tail-biting codes.

VII.   Conclusion

In this paper, we have introduced two types of quantum convolutional codes based on classical self-orthogonal rate-$1/n$ $\mathbb{F}_4$-linear and $\mathbb{F}_2$-linear convolutional codes, respectively, with corresponding decoders. We have also introduced quantum tail-biting block codes based on these codes, which have the same rate, performance and decoding complexity. We have shown that these codes have a potentially attractive tradeoff between performance and complexity for moderate-complexity applications.

In classical coding, convolutional coding was the next step beyond block coding. The next step was to concatenate convolutional codes with algebraic (Reed-Solomon) outer codes for higher performance. Finally, in the past decade, capacity-approaching codes such as low-density parity-check (LDPC) codes and turbo codes with iterative decoding have become the preferred techniques for highest performance. One may anticipate an analogous sequence of advances in quantum coding. Indeed, MacKay *et al.* have already taken a step toward quantum LDPC codes [23], although not without some difficulties.

Acknowledgments

References

[1] A. C. A. de Almeida and R. Palazzo, Jr., "A concatenated [(4, 1, 3)] quantum convolutional code," *Proc. 2004 IEEE Inform. Theory Workshop* (San Antonio, TX), Oct. 2004.

[2] C. H. Bennett, D. P. DiVicenzo, J. Smolin and W. K. Wootters, "Mixed state entanglement and quantum error correction," *Phys. Rev. A*, vol. 54, pp. 3824-3851, 1996. ArXiv: `quant-ph 9604024`.

[3] Th. Beth and M. Grassl, "The quantum Hamming and hexacodes," *Fortschritte der Physik*, vol. 46, pp. 459–491, 1998.

[4] W. Bosma, J. J. Cannon and C. Playoust, "The Magma Algebra System I: The user language," *Journal of Symbolic Computation*, vol. 24, pp. 235–266, 1997.

[5] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Phys. Rev. A*, vol. 54, pp. 1098–1105, Aug. 1996. ArXiv: `quant-ph 9512032`.

[6] A. R. Calderbank, E. M. Rains, P. W. Shor and N. J. A. Sloane, "Quantum error correction and orthogonal geometry," *Phys. Rev. Lett.*, vol. 78, pp. 405–408, 1997. ArXiv: `quant-ph 9605005`.

[7] A. R. Calderbank, E. M. Rains, P. W. Shor and N. J. A. Sloane, "Quantum error correction via codes over $GF(4)$," *IEEE Trans. Inform. Theory*, vol. 44, pp. 1369–1387, July 1998. ArXiv: `quant-ph 9608006`.

[8] A. R. Calderbank, G. D. Forney, Jr. and A. Vardy, "Minimal tail-biting trellises: The Golay code and more," *IEEE Trans. Inform. Theory*, vol. 45, pp. 1435–1455, July 1999.

[9] H. F. Chau, "Quantum convolutional error-correcting codes," *Phys. Rev. A*, vol. 58(2), pp. 905–909, 1998.

[10] H. F. Chau, "Good quantum convolutional error-correction codes and their decoding algorithm exist," *Phys. Rev. A*, vol. 60(3), pp. 1966–1974, 1999.

[11] G. D. Forney, Jr., "Convolutional codes I: Algebraic structure," *IEEE Trans. Inform. Theory*, vol. IT-16, pp. 720–738, Nov. 1970.

[12] G. D. Forney, Jr., "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268–278, March 1973.

[13] G. D. Forney, Jr., "Structural analysis of convolutional codes via dual codes," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 512–518, July 1973.

[14] D. Gottesman, "A theory of fault-tolerant quantum computation," *Phys. Rev. A*, vol. 57, pp. 127–137, 1998. ArXiv: `quant-ph 9702029`.

[15] M. Grassl, *Fehlerkorrigierende Codes für Quantensysteme: Konstruktionen und Algorithmen*, Aachen: Shaker, 2002.

[16] M. Grassl, "Tables of quantum error-correcting codes." WWW: `http://www.codetables.de`.

[17] M. Grassl and Th. Beth, "Cyclic quantum error-correcting codes and quantum shift registers," *Proc. Royal Society London A*, vol. 456, pp. 2689–2706, Nov. 2000. ArXiv: `quant-ph 9910061`.

[18] M. Grassl and M. Rötteler, "Noncatastrophic encoders and encoder inverses for quantum convolutional codes," *Proc. 2006 IEEE Intl. Symp. Inform. Theory* (Seattle, WA), pp. 1109–1113, July 2006. ArXiv: `quant-ph 0602129`.

[19] M. Handlery, S. Höst, R. Johannesson and V. V. Zyablov, "A distance measure tailored to tail-biting codes," *Probl. Inform. Transm.* (*Probl. Pered. Inform.*), vol. 38, pp. 280–295, Oct.-Dec. 2002.

[20] R. Johannesson and K. Sh. Zigangirov, *Fundamentals of Convolutional Coding.* Wiley–IEEE Press, 1999.

[21] D. Jönsson, "Investigation of convolutional codes over $GF(4)$," masters thesis, Dept. I. T., Lund U. (Sweden), Aug. 2003.

[22] R. Laflamme, C. Miquel, J.-P. Paz and W. H. Zurek, "Perfect quantum error-correction code," *Phys. Rev. Lett.*, vol. 77, pp. 198–201, 1996. ArXiv: `quant-ph 9602019`.

[23] D. J. C. MacKay, G. Mitchison and P. L. McFadden, "Sparse-graph codes for quantum error correction," *IEEE Trans. Inform. Theory*, vol. 50, pp. 2315–2330, Oct. 2004.

[24] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information.* Cambridge, UK: Cambridge University Press, 2000.

[25] H. Ollivier and J.-P. Tillich, "Description of a quantum convolutional code," *Phys. Rev. Lett.*, vol. 91(17), pp. 1779021–4, 2003. ArXiv: `quant-ph 0304189`.

[26] H. Ollivier and J.-P. Tillich, "Quantum convolutional codes: Fundamentals," 2004. ArXiv: `quant-ph 0401134`.

[27] J. Preskill, *Lecture notes for Physics 229: Quantum information and computation.* Calif. Inst. Tech., Pasadena, 1998. WWW: `www.theory.caltech.edu/people/preskill/ph229`.

[28] P. Shor, "Scheme for reducing decoherence in quantum computer memory," *Phys. Rev. A*, vol. 52, pp. 2493–2496, 1995.

[29] A. M. Steane, "Error-correcting codes in quantum theory," *Phys. Rev. Lett.*, vol. 77, pp. 793–797, July 1996.

[30] H. N. Ward, "A bound for divisible codes," *IEEE Trans. Inform. Theory*, vol. 38, pp. 191-194, Jan. 1992.