# Efficient Quantum Stabilizer Codes: LDPC and LDPC-Convolutional Constructions

Peiyu Tan, *Student Member, IEEE*, and Jing Li, *Senior Member, IEEE*

***Abstract*—Existing quantum stabilizer codes constructed from the classic binary codes exclusively belong to the special subclass of Calderbank–Shor–Steane (CSS) codes. This paper fills in the gap by proposing five systematic constructions for non-CSS stabilizer codes, the first four of which are based on classic binary quasi-cyclic low-density parity-check (LDPC) codes and last on classic binary LDPC-convolutional codes. These new constructions exploit structured sparse graphs, make essential use of simple and powerful coding techniques including concatenation, rotation and scrambling, and generate rich classes of codes with a wide range of lengths and rates. We derive the sufficient, and in some cases also the necessary, conditions for each construction to satisfy the general symplectic inner product (SIP) condition, and develop practical decoder algorithms for these codes. The resulting codes are the first classes of non-CSS quantum LDPC codes and non-CSS quantum convolutional codes rooted from classic binary codes (rather than codes in GF$(4)$), and some of them perform as well as or better than the existing quantum codes.**

***Index Terms*—Quantum error correction coding, quantum convolutional codes, quantum low-density parity-check (LDPC) convolutional codes, quantum LDPC codes, stabilizer codes.**

## I. Introduction

QUANTUM error-correction codes (QECCs) are intended to protect the fragile quantum states from unwanted evolutions and to allow robust implementations of quantum processing devices. Compared to their classic counter parts, error correction in quantum communication channels is challenging, since a quantum bit, or, a *qubit*—the information-bearing physical system in quantum computation and communication systems—has continuous states rather than two discrete states. Implementation of practical encoders and decoders in quantum mechanics is further complicated by the facts that a quantum state is affected by measurement, and that a qubit with an unknown state cannot be duplicated (i.e., the no-cloning theorem).

While quantum states and quantum errors both take *continuous* values, the ground-breaking work [2] in 1995 proved that an arbitrary unitary error on a qubit can be corrected, provided that the QECC is simultaneously equipped with the ability to correct two basic error types, the bit-flip error and the phase-flip error. Exemplifying this idea, [2] also demonstrated the first constructive quantum code, a 9-qubit single-error correcting code. A year after, another major breakthrough was made by Calderbank and Shor [5], and Steane [6], whose work laid down the celebrated CSS framework and lead to a 7-qubit CSS single-error correcting code [4]. An optimal construction of single-error-correcting codes, due to Laflamme, Miguel, Paz, and Zurek, was reported in 1997 [3], which achieved the quantum Hamming bound of five qubits, the minimum possible integer number of qubits needed to correct arbitrary errors on a single qubit using linear operations.

The establishment of the theory of *stabilizer codes* in 1997 [7] set another major milestone. Perhaps the only near-mature theory in quantum coding, stabilizers codes subsume CSS codes as a subclass, and open up a large possibility for quantum error correcting codes. A variety of stabilizer constructions have since been proposed and analyzed, including, for example, quantum BCH codes [18], quantum Reed–Solomon codes [19], quantum convolutional codes [16], [17], and more recently, quantum low-density parity-check (LDPC) codes [11]–[13], [15]. Unlike the early codes that correct a single error in a block of a few qubits, these newer stabilizer codes correct multiple errors in a block of many qubits, and most of them find roots in classic—either binary or nonbinary—error correction codes. Perhaps the only exception that is unrelated to classic codes is [27], which exploits Boolean functions to construct quantum stabilizer codes.

Leveraging results from classic codes simplifies QECC construction, and the elegant CSS formalism—to which most of the existing stabilizer codes belong—further streamlines the construction. Making essential use of classic binary dual codes, a CSS code is generated systematically from two classic linear codes: an $[n,k_1]$ code with codeword space $C_1$ and an $[n,k_2]$ code with codeword space $C_2$, where $k_1 < k_2$ and $C_1 \subset C_2$ [5], [6]. The popularity of the CSS construction (over the unrestricted stabilizer construction) can be attributed, in part, to its simplicity and close relation with classic dual codes, and, in part, to the lack of formal methods to satisfy the *general*, rather than a special case of, the *symplectic inner product condition* (SIP) required for stabilizer codes (see Section II). However, the constrained formalism of CSS codes also narrows the choice of codes, as well as limits their error-correction capability. For instance, it takes seven qubits for a CSS code to achieve a minimum distance of 3 [4], whereas an unrestricted stabilizer code requires only five qubits to achieve the same minimum distance [3].

This paper investigates efficient and systematic ways to construct *unrestricted* stabilizer codes from classic *binary* codes.

Manuscript received September 26, 2007; revised August 17, 2009. Current version published December 23, 2009. This work is supported in part by National Science Foundation under Grant CCF-0430634 and CCF-0635199, and by the Commonwealth of Pennsylvania through the Pennsylvania Infrastructure Technology Alliance (PITA).

The authors are with Electrical and Computer Engineering Department, Lehigh University, Bethlehem, PA 18015 USA (e-mail: pet3@ece.lehigh.edu; jingli@ece.lehigh.edu).

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

Color versions of Figures 1–9 in this paper are available at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2009.2034794

Borrowing rich results from binary LDPC codes, we devise five systematic constructions for non-CSS stabilizer codes, the first four of which are based on classic binary quasi-cyclic (QC) low-density parity-check (LDPC) codes and last on classic binary LDPC-convolutional codes.

The first construction of quantum LDPC codes, due to Postol [11], uses finite geometry in the CSS method. Mackay developed four more approaches based on cyclic matrices, difference sets and (exhaustive) random search to construct CSS quantum LDPC codes [12]. He also provided simulation curves for these codes. Exploiting group theory, [15] constructed quantum LDPC codes using classic GF4 codes. More recently, [13] proposed the method of matrix splitting and merging, and [26] proposed a nonsearch-based quasi-cyclic formalism, to construct CSS LDPC codes.

In the realm of quantum convolutional codes (QCC), Ollivier and Tillich first generalized the concept of stabilizer codes from block codes to convolutional codes, and presented the first quantum convolutional code, a rate-$1/5$ code [14]. Using similar ideas as Shor's 9-qubit code, Almeida and Palazzo constructed a rate-$1/4$ quantum convolutional code by serially concatenating two rate-$1/2$ codes to separately handle bit-flip and phase-flip errors [16]. The first *class* of quantum convolutional codes—rather than a *single* code—are a family of rate-$1/n$ quantum convolutional codes constructed by Forney, Grassl, and Guha using the CSS method [17]. Lately, Grassl proposed a new CSS construction for rate $2/4$ QCC [28].

It is worth noting that, all the the quantum LDPC codes and quantum convolutional code reported thus far either are constructed from classic *nonbinary* (e.g., GF4) codes, or, if they are based on *classic* binary codes, exclusively rely on the CSS formalism. In comparison, the quantum LDPC codes and the quantum convolutional codes developed here are the first classes of non-CSS stabilizer codes based on classic binary codes. Rooting to powerful binary codes, our codes are capable of correcting a large number of quantum errors, including bursty errors, in a single block. All the code families presented here enjoy a wide range of lengths and rates, including very high rates that approach 1. Additionally, we also present a few simple but useful "general-purpose" operations, that allow us to derive variations from existing codes, as well as combine codes from different families to form hybrid ones.

Another notable contribution of this work is the derivation and elucidation of decoding algorithms for stabilizer LDPC codes and stabilizer LDPC-convolutional codes. Quantum coding research is still at its early age and the technologies are quite immature. Not only does decoder implementation (in quantum mechanics) pose an exceptional challenge, but to derive the decoding algorithm (at the system level) is also nontrivial—except for the special class of CSS codes. This also explains why the majority of the non-CSS stabilizer codes discussed in literature have presented well-defined code books but not a practical decoding algorithm.

The organization of the paper is as follows. Section II presents the basic concepts and notations about quantum stabilizer codes. Section III introduces several important coding formalisms and techniques upon which our quantum codes will be based. Section IV discusses in detail five new constructions for the quantum stabilizer codes. Section V derives decoding algorithms for the proposed stabilizer LDPC codes and stabilizer LDPC-convolutional codes. Section VI demonstrates the simulation results of our codes on realistic quantum channels. Finally, Section VII concludes this paper.

## II. Preliminary on Stabilizer Codes

We first introduce stabilizer codes using the language of finite group theory, which makes easy connection to the binary language of quantum mechanics. In quantum error protection, the structure to store quantum information is a subspace of the total Hilbert space of the physical qubits, thereafter referred to as the code space $\mathcal{C}$.

The state of a qubit takes an arbitrary linear combination (known as *superposition* in Quantum literature) of two basic states: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $\alpha$ and $\beta$ are complex numbers satisfying $|\alpha|^2 + |\beta|^2 = 1$. The quantum state of $n$ qubits can be expressed as $(\alpha_0|0\cdots 00\rangle + \alpha_1|0\cdots 01\rangle + \cdots + \alpha_{2^n-1}|1\cdots 11\rangle)$, where $|\alpha_0|^2 + |\alpha_1|^2 + \cdots + |\alpha_{2^n-1}|^2 = 1$, and the $2^n$ states $|0\cdots 00\rangle, |0\cdots 01\rangle, \ldots, |1\ldots 11\rangle$ are shorthand for the tensor products $|0\rangle \otimes \ldots \otimes |0\rangle, |0\rangle \otimes \ldots \otimes |1\rangle, \ldots, |1\rangle \otimes \ldots \otimes |1\rangle$.

The mathematical representation of an erroneous state $|\psi'\rangle$ is the product of the original state $|\psi\rangle$ and the noise $E$. It is common practice to neglect the amplitude of $E$ (which can be handled by normalization) and consider $E$ as an arbitrary unitary linear operator.

*Definition 2.1:* [Error Operator] Suppose a vector $|\psi\rangle$ of size $n$ is sent through the depolarizing channel. The outcome of the transmission can be written as $E|\psi\rangle$, where the error operator $E$ takes the form of $E = \tau_1 \otimes \tau_2 \otimes \ldots \otimes \tau_n$, and $\tau_i \in \mathcal{G} = \{I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}\}, \forall\, i = 1, 2, \cdots n$. Here, Pauli matrices $X, Z$ and $Y$ correspond, respectively, to a bit flip, a phase flip, and a combination of bit and phase flip.

*Lemma 2.2:* [Error Correcting Capability] A quantum code can correct any unitary type error if it can simultaneously correct $X$ error(s) and $Z$ error(s).

### A. Stabilizer Codes as A Finite Group

*Definition 2.3:* [Stabilizer Codes] A *stabilizer code* has a code space $\mathcal{C}$ that is the largest subspace stabilized by an Abelian group $\mathcal{S}$ acting on the $N$ physical qubits of the code. A *stabilizer* $\mathcal{S}$ is some Abelian subgroup of $\mathcal{G}_N$ (the tensor products of $N$ matrices in $\mathcal{G} = \{I, X, Y, Z\}$) satisfying (i) neither $i$ nor $-1$ is in $\mathcal{S}$; and (ii) $\mathcal{S}$ fixes all the codewords, i.e.,

$$\mathcal{C} = \{|\psi\rangle, \quad \text{s.t. M}|\psi\rangle = |\psi\rangle, \quad \forall \text{M} \in \mathcal{S}\}. \qquad (1)$$

The set of valid codewords are eigenvectors of all the operators in $\mathcal{S}$ with eigenvalue $+1$. Since the stabilizer $\mathcal{S}$ can be described using independent stabilizer generators $\{S_i\}$ such that the set $\mathcal{S}$ comprises $S_i$ and all the products of $S_i$, it is sufficient to define a quantum codeword as a state $|\psi\rangle$ such that $S_i|\psi\rangle = |\psi\rangle$ for all $i$.

When an error operator changes the original state of a qubit, the corrupted state may fall either inside the code space or outside. The former case results in what is known in the (classic) coding literature as *undetectable errors*, a type of errors which exceed the error-correction capability of any code and which should occur very rarely in a well-designed code. The latter case is the task of quantum error correction codes. Since it occurs when the error operator anticommutes with some element of $\mathcal{S}$, stabilizer generators $S_i$ can act as diagnostic operators to evaluate the commutation property of a given state. This can be achieved by, for example, using $S_i$ to measure the eigenvalue array, known as the *syndrome*, of a given state. Further, a syndrome identifies the error operator that can counteract the specific error, which can therefore be applied to the corrupted state to recover the original state.

## B. Binary Formalism of Stabilizer Codes

To connect and compare quantum error-correction codes with classic (digital) error-correction codes, below we introduce stabilizer codes using the language of linear algebra in GF(2) field. To differentiate quantum codes and classic codes, we thereafter denote a classic code that encodes $K$ binary bits to $N$ binary bits as an $[N, K]$ code, and denote a quantum code that encodes $K$ qubits to $N$ qubits as an $[[N, K]]$ code.

The $N-K$ stabilizer generators of an $[[N, K]]$ code can be collectively described as the concatenation of a *pair* of $(N-K)\times N$ binary matrices, $\mathbf{A} = (\mathbf{A}_1|\mathbf{A}_2)$, such that each row in $\mathbf{A}$ corresponds to a unique stabilizer generator, and each pair of columns—the corresponding columns in $\mathbf{A_1}$ and $\mathbf{A_2}$—correspond to a qubit. A "0" entry in $\mathbf{A}$ represents an $I$ operator, and a "1" entry represents either an $X$ or $Y$ operator if it is in the $\mathbf{A}_1$-matrix, or a $Z$ or $Y$ operator if it is in the $\mathbf{A}_2$-matrix.

*Example 1:* A [[5, 1]] cyclic quantum code is defined by the following stabilizer:

$$\begin{array}{ccccc} X & Z & Z & X & I \\ I & X & Z & Z & X \\ X & I & X & Z & Z \\ Z & X & I & X & Z \end{array} \tag{2}$$

which takes a binary form of

$$\mathbf{A} = (\mathbf{A}_1|\mathbf{A}_2) = \left(\begin{array}{ccccc|ccccc} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right). \tag{3}$$

In the binary formalism, the condition that two stabilizers commute with each other translates to that their *symplectic inner product* (SIP) is 0. Let $r_m = (x_m, z_m)$ be the $m$th row in $\mathbf{A}$, where $x_m$ and $z_m$ denote the $m$th rows in $\mathbf{A}_1$- and $\mathbf{A}_2$-matrices, respectively. The symplectic inner product $\odot$ of the $m$th and the $m'$th rows is defined as

$$\begin{aligned} r_m \odot r_{m'} &= (x_m, z_m) \odot (x_{m'}, z_{m'}) \\ &= x_m \cdot z_{m'} + x_{m'} \cdot z_m \mod 2 \end{aligned} \tag{4}$$

where $x_m \cdot z_{m'} = \sum_i x_{mi} z_{m'i}$ represents the regular inner product for vectors. If we write them in compact matrix forms, we get the following.

*Theorem 2.4:* [Symplectic Inner Product Condition] [10] A stabilizer with binary representation $\mathbf{A} = (\mathbf{A}_1|\mathbf{A}_2)$ satisfies the symplectic inner product condition if and only if:

$$\mathbf{A}_1\mathbf{A}_2^T + \mathbf{A}_2\mathbf{A}_1^T = 0 \tag{5}$$

where $T$ stands for matrix transpose.

The symplectic inner product is also known in the literature as the *twisted product* [12].

*Definition 2.5:* [Binary Formalism of Stabilizer Codes] A full-rank binary matrix $\mathbf{A}$, having size $M_Q \times 2N$ and satisfying the symplectic inner product condition in (5), defines a quantum *stabilizer* code with rate $R_Q = (N - M_Q)/N$ that encodes $N - M_Q$ qubits into $N$ qubits.

*Definition 2.6:* [CSS Codes] [5], [6] An $[[N, N - M_1 - M_2]]$ *CSS code* is specified by a stabilizer with the following binary formalism:

$$\mathbf{A} = \left(\begin{array}{c|c} \mathbf{H} & \mathbf{0} \\ \mathbf{0} & \mathbf{G} \end{array}\right) \tag{6}$$

where $\mathbf{H}$ and $\mathbf{G}$ are $M_1 \times N$ and $M_2 \times N$ matrices ($M_1$ not necessarily equals $M_2$) satisfying $\mathbf{H}\mathbf{G}^T = 0$.

One may further restrict $\mathbf{H} = \mathbf{G}$, such that

$$\mathbf{A} = \left(\begin{array}{c|c} \mathbf{H} & \mathbf{0} \\ \mathbf{0} & \mathbf{H} \end{array}\right) \tag{7}$$

and $\mathbf{H}\mathbf{H}^T = 0$. This results in the so-called *CSS codes based on dual-containing codes*, since $\mathbf{H}$ comes from the parity check matrix of a dual-containing (or, weakly self-dual) classic code.

The CSS method provides a very convenient means to satisfy the symplectic inner product condition, but also significantly narrows the choice of quantum codes. For example, it is not possible to construct the [[5, 1]] stabilizer code in Example 1 using the CSS method. Sad enough, to date, there are only a few reported quantum stabilizer codes that fall outside the subclass of CSS codes [5], [6]. Among them, most belong to *convolutional* codes, rather than *block* codes. This paper well fills the research gap, since four out of the five constructions we propose result in block non-CSS stabilizer codes.

## III. Preliminary on Coding Techniques

## A. Quasi-Cyclic Matrices and Basic Operations

We now introduce quasi-cyclic matrices and a few basic coding operations, using which the proposed quantum codes are constructed.

*Definition 3.1:* [Cyclic Matrices] Let $\mathbf{I}$ be an $m \times m$ identity matrix. A *cyclic matrix* $\mathbf{I}_x$ is a shifted identity matrix with the rows of $\mathbf{I}$ circularly shifted to the right by $x$ positions, where $x \in [0, m-1]$ is called the *offset*. For convenience, we assume $\mathbf{I}_0 = \mathbf{I}$ and $\mathbf{I}_{x \pm km} = \mathbf{I}_x$ for any integer $k$.

*Definition 3.2 [Weight-t Cyclic Matrices]:* A *weight-t binary cyclic matrix*, with $t \geq 1$, is the (binary) sum of $t$ shifted identity matrices of the same dimensionality but different offsets:

$$\begin{aligned}\mathbf{B} = \mathbf{I}_{x_1} + \mathbf{I}_{x_2} + \cdots + \mathbf{I}_{x_t},\\ 0 \leq x_1 < x_2 < \cdots < x_t < m.\end{aligned} \tag{8}$$

We call a cyclic matrix single-weight if $t = 1$, and otherwise multiweight. Notice that Definitions 3.1 and 3.2 define a cyclic matrix to be a *square* matrix and hence the same uniform weight column-wise and row-wise. Although the code constructions proposed in this paper will almost exclusively use cyclic square matrices, or, simply, *cyclic blocks*, the definition of cyclic matrices need not constrain on a square dimensionality. In general, a single-weight cyclic square matrix has a full rank, whereas a multiweight cyclic square matrix does not. Hence, we may delete some or all of the redundant rows in a weight-$t$ cyclic (square) matrix, and still call the remaining part a weight-$t$ (row weight) cyclic matrix. For instance, in Example 1, the cyclic matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ shown in (3) are both row-deleted versions of weight-2 square matrices, and both have full rank of 4.

*Definition 3.3:* [Concatenation of Matrices] Consider a set of matrices $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_t$. The *horizontal concatenation* of these matrices, assuming they all have the same number of rows, results in matrix $[\mathbf{A}_1\ \mathbf{A}_2\ \ldots\ \mathbf{A}_t]$. The *vertical concatenation* of these matrices, assuming they all have the same number of columns, results in matrix $[\mathbf{A}_1^T,\ \mathbf{A}_2^T, \ldots, \mathbf{A}_t^T]^T$. In the case when one matrix has fewer rows than the other(s), one may append to this matrix either zero rows or any linear combinations of the existing rows, to bring it up to the needed size. In the sequel, unless otherwise stated, a concatenation by default refers to a horizontal concatenation.

The rotational operation of a matrix, in the context of constructing (classic) ECC and in the notion of "$\pi$-rotation", was first discussed in [25]. A simple operation, $\pi$-rotation enables the transformation of one matrix to a different one using a single parameter, the degree (i.e., $\pi/2, \pi$, or $3\pi/2$) of the rotation. Previous work demonstrated the possibility of using $\pi$-rotation to construct simple and good classic LDPC codes [25], but did not provide a mathematical expression of nor much insight into the rotation operation. Exploring this operation for quantum stabilizer codes, below we mathematically formulate the $\pi$-rotation operation, discuss its properties, and further discuss a generalized concept of "rotation."

*Definition 3.4 [$\pi$-Rotation Operation]:* The *$\pi$-rotation*, or, simply, *rotation*, of a matrix $\mathbf{X}$, denoted as $\pi(\mathbf{X})$, is to rotate matrix $\mathbf{X}$ counterclockwise by 90°.[1] This operation can be mathematically described as

$$\pi\{\mathbf{X}\} = \mathbf{F}\mathbf{X}^T \tag{9}$$

where

$$\mathbf{F} = \begin{pmatrix} 0 & \cdots & 0 & 1 \\ 0 & \cdots & 1 & 0 \\ \vdots & & \vdots & \vdots \\ 1 & \cdots & 0 & 0 \end{pmatrix} \tag{10}$$

[1]To be strict, this operation should probably be termed "$\frac{\pi}{2}$-rotation," but the authors in [25] called it "$\pi$-rotation" and we follow their notation.

is called the *$\pi$-rotational matrix*. It should be noted that $\mathbf{F}^T = \mathbf{F}^{-1}$. In fact, they also both equal $\mathbf{F}$.

Through the repeated application of (9), rotating a matrix counter-clockwise by 180°, 270°, 360°, or any integer multiple of 90°, can be described, respectively, as

$$\pi^2\{\mathbf{X}\} = \pi\{\pi\{\mathbf{X}\}\} = \mathbf{F}\mathbf{X}\mathbf{F}^T, \tag{11}$$
$$\pi^3\{\mathbf{X}\} = \pi\{\pi^2\{\mathbf{X}\}\} = \mathbf{X}^T\mathbf{F}^T \tag{12}$$
$$\pi^4\{\mathbf{X}\} = \pi\{\pi^3\{\mathbf{X}\}\} = \pi^0\{\mathbf{X}\} = \mathbf{X} \tag{13}$$
$$\pi^{4k+i}\{\mathbf{X}\} = \pi^i\{\mathbf{X}\}, \quad i = 0, 1, 2, 3. \tag{14}$$

*Lemma 3.5:* [Properties of Cyclic Matrices] Let $\mathbf{X}_1$ and $\mathbf{X}_2$ be two cyclic square matrices:

1) Transposition, multiplication and addition preserve cyclicity: $\mathbf{X}_i^T$ and $\mathbf{X}_1\mathbf{X}_2^T + \mathbf{X}_2\mathbf{X}_1^T$ are also cyclic squares.
2) Commutativity: $\mathbf{X}_1\mathbf{X}_2 = \mathbf{X}_2\mathbf{X}_1$, and $\mathbf{X}_1\mathbf{X}_2^T = \mathbf{X}_2^T\mathbf{X}_1$.
3) $\pi$-rotational symmetry: $\pi^2\{\mathbf{X}_i\} = \mathbf{F}\mathbf{X}_i^T\mathbf{F}^T = \mathbf{X}_i^T$, and $\mathbf{F}\mathbf{X}_i = \mathbf{X}_i^T\mathbf{F}^T = (\mathbf{F}\mathbf{X}_i)^T$.

*Definition 3.6:* [General Rotation Operation] We call binary matrix $\mathbf{P}$ a *general rotational matrix* if $\mathbf{P}^T = \mathbf{P}^{-1}$. We further define the following matrix operation as the *general rotation operation*:

$$\begin{aligned}\Pi\{\mathbf{X}\} &= \mathbf{P}\mathbf{X}^T,\\ \Pi^k\{\mathbf{X}\} &= \Pi\{\Pi^{k-1}\{\mathbf{X}\}\}, \quad k = 2, 3, \ldots\end{aligned} \tag{15}$$

Notice that

$$\begin{aligned}\Pi^2\{\mathbf{X}\} &= \mathbf{P}\mathbf{X}\mathbf{P}^T\\ \Pi^3\{\mathbf{X}\} &= \mathbf{P}^2\mathbf{X}^T\mathbf{P}^T \neq \mathbf{X}^T\mathbf{P}^T\\ \Pi^4\{\mathbf{X}\} &= \mathbf{P}^2\mathbf{X}(\mathbf{P}^T)^2 \neq \mathbf{X}\end{aligned}$$

and in general,

$$\Pi^k\{\mathbf{X}\} = \begin{cases} \mathbf{P}^{\lceil k/2 \rceil}\mathbf{X}^T(\mathbf{P}^T)^{\lfloor k/2 \rfloor}, & \text{for odd } k \\ \mathbf{P}^{k/2}\mathbf{X}(\mathbf{P}^T)^{k/2}, & \text{for even } k \end{cases} \tag{16}$$

$$\Pi^{4k+i}\{\mathbf{X}\} \neq \Pi^i\{\mathbf{X}\}. \tag{17}$$

Apparently, $\pi$-rotation is a subcase of general rotation, with $\mathbf{P}$ taking the special form of $\mathbf{F}$. A binary matrix $\mathbf{Q}$ having exactly one "1" per row and per column is referred to, in literature, as a *scrambling matrix*, a *permutation matrix*, or, an *interleaving matrix*. The name is such because it is a random permutation of the identity matrix $\mathbf{I}$, and when it operates on a (column) vector $x$, the result, $\mathbf{Q}x$, is a scrambled version of $x$. It is easy to verify that any scrambling matrix $\mathbf{Q}$ satisfies $\mathbf{Q}\mathbf{Q}^T = \mathbf{I}$, or, $\mathbf{Q}^{-1} = \mathbf{Q}^T$, and can therefore be used to perform general rotation.

From (16) and $\mathbf{P}^T = \mathbf{P}^{-1}$, we get $\Pi^2\{\mathbf{X}\}\mathbf{P} = \mathbf{P}\mathbf{X}$. Thus, we have the following lemma.

*Lemma 3.7:* [Property of General Rotation] If matrix $\mathbf{X}$ is invariant under the 2nd order general rotation, i.e., $\Pi^2\{\mathbf{X}\} = \mathbf{X}$, then $\mathbf{X}$ commutes with the general rotational matrix $\mathbf{P}$: $\mathbf{X}\mathbf{P} = \mathbf{P}\mathbf{X}$.

## B. General Techniques for Constructing Quantum Codes

To design (good) stabilizer codes is challenging, because of the lack of formal methods to device a pair of matrices $\mathbf{A}_1$ and $\mathbf{A}_2$, such that i) they satisfy the symplectic inner product condition, and ii) combined together, they specify a code which has practical encoder and decoder, and hopefully good error-correcting performance. The advancement in (classic) LDPC coding in the last decade is truly cause for excitement—among other significant impacts, it also opens a potentially fruitful direction for stabilizer construction. Specifically, if we select $\mathbf{A}_1$ and $\mathbf{A}_2$ to be both sparse-graph-based, then their combination still represents a sparse graph, which after some manipulation, may well satisfy the zero-SIP condition. As will be discussed in Section V, the combined sparse graph also lends itself a practical decoding algorithm, which is similar in spirit to message-passing in the classic case.

Before formulating specific rules for (sparse matrices) $\mathbf{A}_1$ and $\mathbf{A}_2$ to satisfy the general SIP condition, we note that a few trivial cases exist for $\mathbf{A}_1$ and $\mathbf{A}_2$ to produce zero-SIP. These include i) $\mathbf{A}_1 = \mathbf{0}$, ii) $\mathbf{A}_2 = \mathbf{0}$, and iii) $\mathbf{A}_1 = \mathbf{A}_2$. Since $\mathbf{A}_1$ handles $Z$-errors and $\mathbf{A}_2$ handles $X$-errors, the above three cases result in quantum codes that correct, respectively, phase-flips only, bit-flips only, and bit-and-phase flips only. Hence, they are useful only on special quantum channels, but not on a general quantum channel where all the three types of errors coexist.

We start by discussing a few general-purpose operations and techniques that preserve zero-SIP. Although simple, these techniques turn out to be quite useful in varying and extending new stabilizer codes from existing ones.

*Lemma 3.8:* [Concatenation Preserves Symplectic Inner Product Condition] Consider two families of matrices: $\{\mathbf{A}_{1i}\}$ and $\{\mathbf{A}_{2i}\}$, $\quad i = 1, 2, \cdots, n$. Let $\mathbf{A}_1 \triangleq [\mathbf{A}_{11},\ \mathbf{A}_{12}, \ldots, \mathbf{A}_{1n}]$ and $\mathbf{A}_2 \triangleq [\mathbf{A}_{21}, \mathbf{A}_{22}, \ldots, \mathbf{A}_{2n}]$ be their respective (horizontal) concatenations. If $\mathbf{A}_{1i}\mathbf{A}_{2i}^T + \mathbf{A}_{2i}\mathbf{A}_{1i}^T = 0$ for any integer $i \in [1, n]$, then $\mathbf{A}_1\mathbf{A}_2^T + \mathbf{A}_2\mathbf{A}_1^T = 0$.

*Lemma 3.9:* [Row Deletion Preserves Symplectic Inner Product Condition] Suppose that $\mathbf{A}_1$ and $\mathbf{A}_2$ produce zero-SIP. Then deleting a few corresponding row(s) from both matrices results in two smaller matrices which continue to produce zero-SIP.

*Lemma 3.10:* [Row Insertion Preserves Symplectic Inner Product Condition] Suppose that $\mathbf{A}_1$ and $\mathbf{A}_2$ produce zero-SIP. Then appending one or multiple pairs of rows to $\mathbf{A}_1$ and $\mathbf{A}_2$ preserve zero-SIP, provided that each appended pair of rows correspond to the same linear combination of the existing rows in $\mathbf{A}_1$ and $\mathbf{A}_2$, respectively.

*Remark:* It is worth mentioning that neither column deletion nor column addition preserves the zero-SIP condition.

*Lemma 3.11:* [Addition Preserves Symplectic Inner Product condition] If $\mathbf{A}_1$ and $\mathbf{A}_2$, and $\mathbf{B}_1$ and $\mathbf{A}_2$, both produce zero-SIP, then $(\mathbf{A}_1 + \mathbf{B}_1)$ and $\mathbf{A}_2$ produces zero-SIP.

These Lemmas follow straightforward from matrix computations, and their proofs are therefore omitted. Despite their simplicity, these rules are rather helpful in transforming existing codes to different ones. For example, insertion and deletion help adjust the size of matrices so that they can be concatenated. Deletion also helps eliminate short cycles in the code graph and thereby improves the efficacy of message-passing decoding. A heavily-exploited tool in this paper is the horizontal concatenation, which constructs longer and higher rate codes from the existing shorter and lower rate ones, thus considerably enriching the choice for code lengths and rates. This is useful addition to the literature, since the quantum codes reported thus far are mostly low-rate codes. In the context of sparse-graph codes, concatenation and addition also provide a convenient means for adding irregularity to the code, which may in turn improve code performance. It should be noted that these tools are auxiliary to and rely on (good) "base" constructions. For example, concatenating two trivial codes with stabilizers $[\mathbf{0}|\mathbf{B}]$ and $[\mathbf{C}|\mathbf{0}]$ does not produce a nontrivial code, since the stabilizer, $[\mathbf{0}, \mathbf{C}|\mathbf{B}, \mathbf{0}]$, corrects only phase-flips for the first section of qubits and only bit-flips for the remainder section. Additionally, although concatenation increases the apparent code minimum distance, the normalized minimum distance (normalized with respect to the codeword length) does not increase. Hence concatenating codes with poor distance spectrum does not effectively improve the situation.

In what follows, we will present five systematic constructions, the first four of which lead to non-CSS block stabilizer codes and the last leads to non-CSS convolutional stabilizer codes. These new codes possess simple construction, a wide range of sizes and rates (including very high rates), and the capability of correcting hundreds of errors.

## IV. New Constructions

### A. Construction I: Multiweight QC-LDPC Construction

Classic QC-LDPC codes can be constructed from single-weight cyclic matrices and multiweight ones. For quantum codes, since vertical concatenation does not easily preserve zero-SIP, and since (conventional) QC-LDPC codes constructed from multiweight cyclic blocks have minimum distance on par with those from single-weight cyclic blocks [31], we thereby consider multiweight cyclic blocks, so that reasonable column weights are achieved without the need for vertical concatenation. Below we provide the sufficient condition for even-weight cyclic blocks, and the sufficient and necessary condition for weight-2 cyclic blocks, to satisfy zero-SIP.

*Theorem 4.1:* [Sufficient and Necessary Condition for Weight-2 Cyclic Matrices to Produce Zero-SIP] Let $\mathbf{A}_1 = \mathbf{I}_{a_1} + \mathbf{I}_{a_2}$ and $\mathbf{A}_2 = \mathbf{I}_{b_1} + \mathbf{I}_{b_2}$, where $b_1 < a_1 < a_2$ and $b_1 < b_2$, and $a_1, a_2, b_1, b_2 \in [0, m-1]$, where $m$ is the dimension of $\mathbf{I}$. The necessary and sufficient condition to satisfy zero symplectic inner product is as follows.

If $m$ is even, $a_1 \neq a_2 \neq b_1 \neq b_2$,

$$\begin{aligned}&(1)\quad a_1 + a_2 = b_1 + b_2 \pmod{m};\ \text{or}\\ &(2)\quad a_1 + a_2 = 2b_2 = 2b_1 + m;\ \text{or}\\ &(3)\quad b_1 + b_2 = 2a_1 = 2a_2 - m;\ \text{or}\\ &(4)\quad a_2 = a_1 + m/2, \quad b_1 = b_2 + m/2.\end{aligned} \tag{18}$$

If $m$ is odd, $a_1 \neq a_2 \neq b_1 \neq b_2$,

$$a_1 + a_2 = b_1 + b_2 \pmod{m}. \tag{19}$$

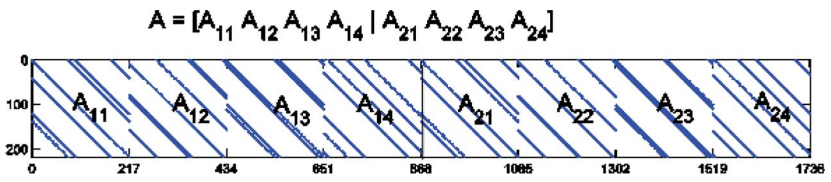

Fig. 1. Example 2: A rate $3/4$ regular QC-LDPC quantum code.

*Theorem 4.2:* [Sufficient Condition for Even-Weight Cyclic Matrices to Produce Zero-SIP] Let $\mathbf{A}_1 = \mathbf{I}_{a_1} + \mathbf{I}_{a_2} + \cdots + \mathbf{I}_{a_t}$ and $\mathbf{A}_2 = \mathbf{I}_{b_1} + \mathbf{I}_{b_2} + \cdots + \mathbf{I}_{b_t}$, where $t$ is even. A sufficient condition to satisfy the symplectic inner product condition is:

$$\begin{aligned} a_1 + a_2 = a_3 + a_4 = \cdots = a_{t-1} + a_t \\ = b_1 + b_2 = b_3 + b_4 = \cdots = b_{t-1} + b_t, \quad (\bmod m) \end{aligned} \tag{20}$$

and

$$a_1 \neq \cdots \neq a_t \neq b_1 \neq \cdots \neq b_t. \tag{21}$$

*Proof:* To prove Theorem 4.1, we consider three cases each of which comprising a couple of subcases. The proof straight-forward but somewhat long. A sketch is provided in the Appendix to avoid distracting readers. Theorem 4.2 can be proved by mathematical induction. □.

Theorems 4.1 and 4.2 provide an efficient way to construct cyclic stabilizer codes through a matching pair of cyclic (square) matrices $\mathbf{A}_1$ and $\mathbf{A}_2$. The resultant code, such as the one in Example 1, will likely have code rate close to 0. Codes of higher rates can be constructed by horizontally concatenating $\tau$ pairs of size-$m$ cyclic square matrices. The result is a quasi-cyclic non-CSS LDPC stabilizer code with code length $m\tau$ (qubits) and code rate close to $(\tau - 1)/\tau$. With $\tau = 2, 3, \ldots$, the code rate covers $\approx 1/2, 2/3, \cdots$ and can get arbitrarily close to 1.

For good error correction capability, we recommend building quantum QC-LDPC codes from classic QC-LDPC codes that have good performances. Here is an illustrating example.

*Example 2:* [Regular QC-LDPC Stabilizer Codes from Construction I] Suppose $\tau = 4$. [24] listed a good $[828, 651]$ (classic) regular QC-LDPC code with column weight $\mathrm{j} = 3$ and row weight $\mathrm{k} = 12$, where the base identity matrices have uniform sizes of $217 \times 217$, and the parity check matrix is in the form of

$$[\mathbf{I}_0 + \mathbf{I}_{121} + \mathbf{I}_{137}, \quad \mathbf{I}_8 + \mathbf{I}_{79} + \mathbf{I}_{85}, \quad \mathbf{I}_1 + \mathbf{I}_{11} + \mathbf{I}_{100}, \\ \mathbf{I}_{29} + \mathbf{I}_{165} + \mathbf{I}_{207}].$$

Following Theorem 4.2, we add an additional weight in each submatrix to make the weight even and to satisfy (21). One possible choice is to use offsets 41, 156, 110, and 126, respectively, since $0 + 41 = 121 + 137 \ (\bmod 217), 8 + 156 = 79 + 85, 1 + 110 = 11 + 100$, and $29 + 126 = 165 + 207 \ (\bmod 217)$

$$\begin{aligned} \mathbf{A}_1 &= [\mathbf{A}_{11},\ \mathbf{A}_{12},\ \mathbf{A}_{13},\ \mathbf{A}_{14}] \\ &= [\mathbf{I}_0 + \mathbf{I}_{121} + \mathbf{I}_{137} + \mathbf{I}_{41},\ \mathbf{I}_8 + \mathbf{I}_{79} + \mathbf{I}_{85} + \mathbf{I}_{156}, \\ &\quad \mathbf{I}_1 + \mathbf{I}_{11} + \mathbf{I}_{100} + \mathbf{I}_{110},\ \mathbf{I}_{29} + \mathbf{I}_{165} + \mathbf{I}_{207} + \mathbf{I}_{126}]. \end{aligned}$$

Again, apply Theorem 2.4 to get a QC matrix $\mathbf{A}_2$ that matches $\mathbf{A}_1$; for example,

$$\begin{aligned} \mathbf{A}_2 = [&\mathbf{I}_1 + \mathbf{I}_{120} + \mathbf{I}_{138} + \mathbf{I}_{40},\ \mathbf{I}_6 + \mathbf{I}_{81} + \mathbf{I}_{83} + \mathbf{I}_{158}, \mathbf{I}_2 \\ &+ \mathbf{I}_{10} + \mathbf{I}_{101} + \mathbf{I}_{109},\ \mathbf{I}_{34} + \mathbf{I}_{160} + \mathbf{I}_{212} + \mathbf{I}_{121}]. \end{aligned}$$

$\mathbf{A}_1$ and $\mathbf{A}_2$ are capable of correcting phase-flip errors and bit-flip errors respectively. Both have column weight 4 and row weight 16. Concatenating them leads to the stabilizer (in its binary form) of a regular QC-LDPC code with rate close to 0.75 and block length 868 (qubits), as shown in Fig. 1.

This simple method allows us to borrow results from classic QC-LDPC codes and to assemble quantum QC-LDPC codes with a large variety of code lengths and code rates. The downside of this construction is the existence of length-4 cycles in the code graph when a submatrix has column weight $t \geq 4$, which may impair the performance of message-passing decoding. Two improvements are investigated to make the code perform better. The first is to add irregularity and randomness into the stabilizer matrices to reduce the number of short cycles. This may be achieved by, for example, the rotational operations discussed in the next two sections. The second is to extend beyond the kingdom of LDPC codes and to construct convolutional codes from QC-LDPC codes. This second approach will be discussion in Section IV-E.

## *B. Construction II: $\pi$-Rotation Based Construction*

We introduce two constructions based on the $\pi$-rotation, a convenient operation for transforming matrices. The resultant LDPC stabilizer codes are easy to describe and render simple decoder implementation.

*Theorem 4.3 [$\pi$-Rotation Construction]:* Consider two cyclic (square) matrices of arbitrary weights and offsets, $\mathbf{X}_1$ and $\mathbf{X}_2$. Let

$$\mathbf{A}_i = \begin{pmatrix} \mathbf{X}_i, & \pi\{\mathbf{X}_i\}, & \pi^2\{\mathbf{X}_i\}, & \pi^3\{\mathbf{X}_i\} \\ \pi^3\{\mathbf{X}_i\}, & \mathbf{X}_i, & \pi\{\mathbf{X}_i\}, & \pi^2\{\mathbf{X}_i\} \\ \pi^2\{\mathbf{X}_i\}, & \pi^3\{\mathbf{X}_i\}, & \mathbf{X}_i, & \pi\{\mathbf{X}_i\} \\ \pi\{\mathbf{X}_i\}, & \pi^2\{\mathbf{X}_i\}, & \pi^3\{\mathbf{X}_i\}, & \mathbf{X}_i \end{pmatrix} \quad i = 1, 2. \tag{22}$$

We have $\mathbf{A}_1\mathbf{A}_2^T + \mathbf{A}_2\mathbf{A}_1^T = 0$; that is, $[\mathbf{A}_1|\mathbf{A}_2]$ forms a quantum stabilizer code.

*Proof:* For simplicity, let $\mathbf{A}_{1i}, \mathbf{A}_{2i}, \mathbf{A}_{3i}$, and $\mathbf{A}_{4i}$ denote the four *row-blocks* in $\mathbf{A}_i$ in (22), respectively

$$\begin{aligned} \mathbf{A}_{1i} &\triangleq (\mathbf{X}_i, \pi\{\mathbf{X}_i\}, \pi^2\{\mathbf{X}_i\}, \pi^3\{\mathbf{X}_i\}) \\ \mathbf{A}_{2i} &\triangleq (\pi^3\{\mathbf{X}_i\}, \mathbf{X}_i, \pi\{\mathbf{X}_i\}, \pi^2\{\mathbf{X}_i\}) \\ \mathbf{A}_{3i} &\triangleq (\pi^2\{\mathbf{X}_i\}, \pi^3\{\mathbf{X}_i\}, \mathbf{X}_i, \pi\{\mathbf{X}_i\}) \\ \mathbf{A}_{4i} &\triangleq (\pi\{\mathbf{X}_i\}, \pi^2\{\mathbf{X}_i\}, \pi^3\{\mathbf{X}_i\}, \mathbf{X}_i), \quad i = 1, 2. \end{aligned}$$

To prove Theorem 4.3, it is enough to show

$$\mathbf{A}_{j1}^T\mathbf{A}_{k2}^T + \mathbf{A}_{j2}\mathbf{A}_{k1}^T = 0, \quad \text{for } 1 \le j \le k \le 4. \tag{23}$$

This follows directly from the properties of cyclic square matrices (Lemma 3.5) and the fact that $\mathbf{F} = \mathbf{F}^T = \mathbf{F}^{-1}$. For instance, when $j = 1$ and $k = 4$, we get

$$\begin{aligned}
&\mathbf{A}_{11}\mathbf{A}_{42}^T + \mathbf{A}_{12}\mathbf{A}_{41}^T \\
&\quad = \left[\mathbf{X}_1,\ \mathbf{F}\mathbf{X}_1^T,\ \mathbf{F}\mathbf{X}_1\mathbf{F}^T,\ \mathbf{X}_1^T\mathbf{F}^T\right] \\
&\qquad \times \left[\mathbf{F}\mathbf{X}_2^T,\ \mathbf{F}\mathbf{X}_2\mathbf{F}^T,\ \mathbf{X}_2^T\mathbf{F}^T,\ \mathbf{X}_2\right]^T \\
&\qquad + \left[\mathbf{X}_2,\ \mathbf{F}\mathbf{X}_2^T,\ \mathbf{F}\mathbf{X}_2\mathbf{F}^T,\ \mathbf{X}_2^T\mathbf{F}^T\right] \\
&\qquad \times \left[\mathbf{F}\mathbf{X}_1^T,\ \mathbf{F}\mathbf{X}_1\mathbf{F}^T,\ \mathbf{X}_1^T\mathbf{F}^T,\ \mathbf{X}_1\right]^T \\
&\quad = \mathbf{X}_1\mathbf{X}_2\mathbf{F}^T + \mathbf{F}\mathbf{X}_1^T\mathbf{F}\mathbf{X}_2^T\mathbf{F}^T \\
&\qquad + \mathbf{F}\mathbf{X}_1\mathbf{F}^T\mathbf{F}\mathbf{X}_2 + \mathbf{X}_1^T\mathbf{F}^T\mathbf{X}_2^T \\
&\qquad + \mathbf{X}_2\mathbf{X}_1\mathbf{F}^T + \mathbf{F}\mathbf{X}_2^T\mathbf{F}\mathbf{X}_1^T\mathbf{F}^T \\
&\qquad + \mathbf{F}\mathbf{X}_2\mathbf{F}^T\mathbf{F}\mathbf{X}_1 + \mathbf{X}_2^T\mathbf{F}^T\mathbf{X}_1^T \\
&\quad = \mathbf{X}_1\mathbf{X}_2\mathbf{F} + \mathbf{X}_1\mathbf{X}_2^T\mathbf{F} \\
&\qquad + \mathbf{X}_1^T\mathbf{X}_2^T\mathbf{F} + \mathbf{X}_1^T\mathbf{X}_2\mathbf{F} \\
&\qquad + \mathbf{X}_2\mathbf{X}_1\mathbf{F} \\
&\qquad + \mathbf{X}_2\mathbf{X}_1^T\mathbf{F} + \mathbf{X}_2^T\mathbf{X}_1^T\mathbf{F} \\
&\qquad + \mathbf{X}_2^T\mathbf{X}_1\mathbf{F} \\
&\quad = (\mathbf{X}_1\mathbf{X}_2\mathbf{F} + \mathbf{X}_2\mathbf{X}_1\mathbf{F}) \\
&\qquad + \left(\mathbf{X}_1\mathbf{X}_2^T\mathbf{F} + \mathbf{X}_2^T\mathbf{X}_1\mathbf{F}\right) \\
&\qquad + \left(\mathbf{X}_1^T\mathbf{X}_2^T\mathbf{F} + \mathbf{X}_2^T\mathbf{X}_1^T\mathbf{F}\right) \\
&\qquad + \left(\mathbf{X}_1^T\mathbf{X}_2\mathbf{F} + \mathbf{X}_2\mathbf{X}_1^T\mathbf{F}\right) = 0.
\end{aligned}$$

□

The advantage of this construction is that it imposes no other constraint on $\mathbf{X}_i$ except for their being cyclic, which provides a rich choice for new quantum codes. The disadvantage, however, is that with $\mathbf{X}_i$ being cyclic and each of its rotated versions being somewhat similar to it, the resultant matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ take a quasi-cyclic form and generally possess many short cycles (in the code graph), such as shown in Fig. 2(A). It is possible to reduce short cycles by deleting some rows. Recall that row deletion preserves zero-SIP, as well as introduces variations to existing codes.

Column deletion will in general break the zero-SIP condition, but for the rotational construction presented in (22), a new construction can be derived from a coordinated deletion of both row blocks and column blocks:

$$\mathbf{A}_i = (\pi^t\{\mathbf{X}_i\} \quad \pi^k\{\mathbf{X}_i\}), \quad i = 1, 2 \tag{24}$$

where $\mathbf{X}_1$ and $\mathbf{X}_2$ are two cyclic matrices of arbitrary weights and offsets, and $t$ and $k$ are arbitrary integers $t \ne k \pmod 4$. The resultant LDPC stabilizer code, $[\mathbf{A}_1|\mathbf{A}_2]$, has rate $1/2$ and

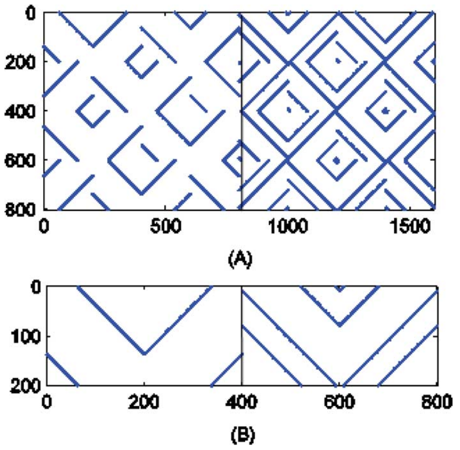

Fig. 2. Examples of $\pi$-rotation construction II.

has a much lower chance of getting short cycles than that in (22). By properly choosing the weights and offsets for cyclic matrices $\mathbf{X}_1$ and $\mathbf{X}_2$, and the rotational degrees $t$ and $k$, it is possible to completely eliminate length-4 cycles. For example, the stabilizer code in Fig. 2(A) from (22) contains a large number of length-4 cycles due to excessive regularity, whereas the construction in Fig. 2(B) from (24) has considerably fewer.

## C. Construction III: General-Rotation-Based Construction

Constructions based on general rotations can expect to provide better randomness than the $\pi$-rotation.

*Theorem 4.4:* [General Rotation Construction] Let

$$\mathbf{A}_i = \begin{pmatrix} \mathbf{X}_i, & \left(\Pi(\mathbf{X}_i^T)\right)^T \\ \Pi(\mathbf{X}_i), & \Pi^2(\mathbf{X}_i) \end{pmatrix}, \quad i = 1, 2. \tag{25}$$

$\mathbf{A}_1\mathbf{A}_2^T + \mathbf{A}_2\mathbf{A}_1^T = 0$, when both $\mathbf{X}_1$ and $\mathbf{X}_2$ are symmetric (i.e., $\mathbf{X}_i = \mathbf{X}_i^T$), or when both $\mathbf{X}_1$ and $\mathbf{X}_2$ are cyclic and square.

*Proof:* We first write the symplectic inner product as (26).

i) It is easy to see that when $\mathbf{X}_1$ and $\mathbf{X}_2$ are both symmetric, then all the terms in (25) become zero.

ii) Recall that the transpose, addition and multiplication of cyclic matrices remain cyclic, and any two cyclic matrices commute with each other. Thus, if $\mathbf{X}_1$ and $\mathbf{X}_2$ are both cyclic squares, (25) becomes zero as shown in the unnumbered equation at the bottom of the page. □

The general rotation operation in (25) can make use of any matrix $\mathbf{P}$ satisfying $\mathbf{P}^T = \mathbf{P}^{-1}$, and in particular, scrambling matrices. This provides ample freedom in constructing codes, especially when used in combination with other constructions and the general operation rules discussed in Lemmas 3.8–3.11. Here is an example.

$$\mathbf{A}_1\mathbf{A}_2^T + \mathbf{A}_2\mathbf{A}_1^T = \begin{pmatrix} \mathbf{X}_1\mathbf{X}_2^T + \mathbf{X}_2\mathbf{X}_1^T + \mathbf{X}_1^T\mathbf{X}_2 + \mathbf{X}_2^T\mathbf{X}_1, & (\mathbf{X}_1\mathbf{X}_2 + \mathbf{X}_2\mathbf{X}_1 + \mathbf{X}_1^T\mathbf{X}_2^T + \mathbf{X}_2^T\mathbf{X}_1^T)P^T \\ P(\mathbf{X}_1\mathbf{X}_2 + \mathbf{X}_2\mathbf{X}_1 + \mathbf{X}_1^T\mathbf{X}_2^T + \mathbf{X}_2^T\mathbf{X}_1^T), & P(\mathbf{X}_1^T\mathbf{X}_2 + \mathbf{X}_2^T\mathbf{X}_1 + \mathbf{X}_1\mathbf{X}_2^T + \mathbf{X}_2\mathbf{X}_1^T)P^T \end{pmatrix}$$

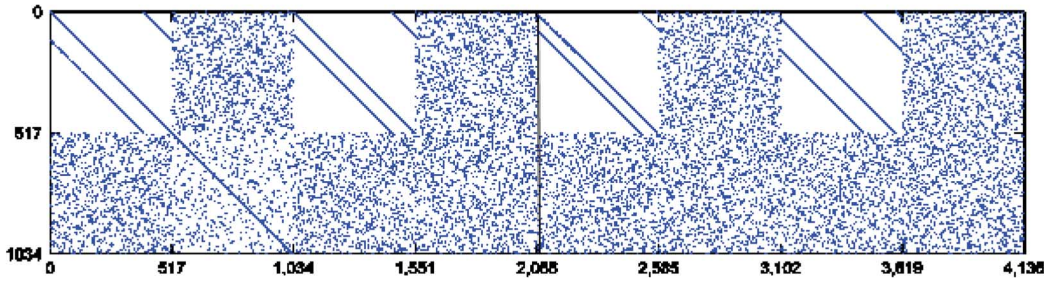

Fig. 3. Example of the LDPC stabilizer code from construction IV.

*Example 3:* **[LDPC Stabilizer Codes From Construction II]** Randomly select two weight-2 cyclic base matrices of dimension 517 $\times$ 517, such as

$$\mathbf{X}_1 = [\mathbf{I}_0 + \mathbf{I}_{121}], \mathbf{X}_2 = [\mathbf{I}_1 + \mathbf{I}_{100}]. \tag{27}$$

Randomly generate a scrambling matrix $\mathbf{P}$ of size 517, apply $\mathbf{P}$ on $\mathbf{A}_1$ and $\mathbf{A}_2$ according to (25), and generate a stabilizer code of length 1034 (qubits) and rate close to 0

$$\left(\begin{array}{cc|cc} \mathbf{X}_1, & \mathbf{X}_1^T\mathbf{P}^T & \mathbf{X}_2, & \mathbf{X}_2^T\mathbf{P}^T \\ \mathbf{P}\mathbf{X}_1^T, & \mathbf{P}\mathbf{X}_1\mathbf{P}^T & \mathbf{P}\mathbf{X}_2^T, & \mathbf{P}\mathbf{X}_2\mathbf{P}^T \end{array}\right) \tag{28}$$

Repeat this procedure to generate a second stabilizer code based on cyclic matrices,

$$\mathbf{Y}_1 = [\mathbf{I}_8 + \mathbf{I}_{79}], \quad \mathbf{Y}_2 = [\mathbf{I}_{29} + \mathbf{I}_{165}] \tag{29}$$

and a random scrambling matrix $\mathbf{Q}$. Applying horizontal concatenation leads to a stabilizer code of length $N = 2068$ (qubits) and rate approximately $1/2$, as in (30) shown at the bottom of the page. The binary representation for this quantum code consists of a pair of matrices each having size 1034 $\times$ 2068, row weight 8 and column weight 4, are depicted[2] in Fig. 3.

Here are two comments: i) Compared to the previous constructions, the general-rotation-based construction embraces both regularity and randomness—the former being implicit in the code description, and the latter being explicit in the matrix (graph) realization. It thus promises easy description and good performance, as will be confirmed by our simulation results. ii) The code in (30) can, in fact, be simplified to an equivalent form of (31) shown at the bottom of the page, which provides the same level of randomness.

[2]The graph is very sparse, although certain parts appear dense. The denseness is the artifact of using thick points to denote "1"'s, since otherwise these weights become hard to see.

## *D. Construction IV: Extension From General-Rotation Based Constructions*

*Theorem 4.5:* [General Rotation construction 2] Let $P$ be an arbitrary scrambling matrix. Let

$$\mathbf{A}_i = \begin{pmatrix} \mathbf{X}_i & \mathbf{X}_i^T\mathbf{P}^T \end{pmatrix}, \quad i = 1, 2. \tag{32}$$

A valid stabilizer code $[\mathbf{A}_1|\mathbf{A}_2]$ results,

i) if both $\mathbf{X}_1$ and $\mathbf{X}_2$ are symmetric, i.e., $\mathbf{X}_i = \mathbf{X}_i^T$; or
ii) if $\mathbf{X}_1$ and $\mathbf{X}_2$ are cyclic square matrices (of either single- or multiple-weights); or
iii) if $(\mathbf{X}_1\mathbf{X}_2^T + \mathbf{X}_1^T\mathbf{X}_2)$ is symmetric.

The proof is straightforward, and is therefore omitted for succinctness. This construction may be viewed as a row-deleted variation of the construction in Theorem 4.4, but condition (3) is new and unique, and provides additional coding choices. Further, the resultant stabilizer code naturally has a code rate of $1/2$ (without the need for horizontal concatenation). We next present two code examples, constructed from conditions (1) and (3) in Theorem 4.5, to demonstrate the rich possibilities.

*Example 4:* **[LDPC Stabilizer Codes From Construction III(1)]** It is easy to build a self-symmetric matrix $\mathbf{X}_i = \mathbf{M}_i \times \mathbf{M}_i^T$ from an arbitrary (full-rank) square matrix $\mathbf{M}_i$, and to accordingly use condition (1) to form stabilizer codes. To achieve a good coding performance, however, it is desirable to consider sparse matrices with appropriate column weights. In this example, we consider $\mathbf{M}_i$ of dimension 77-by-77 and of (near-)uniform row weight and column weight of 3 ($i = 1, 2$). Matrices $\mathbf{M}_i$ may each be obtained, for example, by binary addition of three randomly-selected scrambling matrices. Let $\mathbf{X}_i = \mathbf{M}_i \times \mathbf{M}_i^T + \mathbf{I} \pmod 2$, where adding $\mathbf{I}$ is optional but helps introduce irregularity to the column weight. Next, randomly generate a scrambling matrix $\mathbf{P}$ of size 77, and apply the rule in (32) to construct a stabilizer

$$\left(\begin{array}{cccc|cccc} \mathbf{X}_1, & \mathbf{X}_1^T, & \mathbf{Y}_1, & \mathbf{Y}_1^T & \mathbf{X}_2, & \mathbf{X}_2^T, & \mathbf{Y}_2, & \mathbf{Y}_2^T \\ \mathbf{P}\mathbf{X}_1^T, & \mathbf{P}\mathbf{X}_1, & \mathbf{Q}\mathbf{Y}_1^T, & \mathbf{Q}\mathbf{Y}_1 & \mathbf{P}\mathbf{X}_2^T, & \mathbf{P}\mathbf{X}_2, & \mathbf{Q}\mathbf{Y}_2^T, & \mathbf{Q}\mathbf{Y}_2 \end{array}\right) \tag{30}$$

$$\left(\begin{array}{cccc|cccc} \mathbf{X}_1, & \mathbf{X}_1^T, & \mathbf{Y}_1, & \mathbf{Y}_1^T & \mathbf{X}_2, & \mathbf{X}_2^T, & \mathbf{Y}_2, & \mathbf{Y}_2^T \\ \mathbf{P}\mathbf{X}_1^T, & \mathbf{P}\mathbf{X}_1, & \mathbf{Q}\mathbf{Y}_1^T, & \mathbf{Q}\mathbf{Y}_1 & \mathbf{P}\mathbf{X}_2^T, & \mathbf{P}\mathbf{X}_2, & \mathbf{Q}\mathbf{Y}_2^T, & \mathbf{Q}\mathbf{Y}_2 \end{array}\right) \tag{31}$$

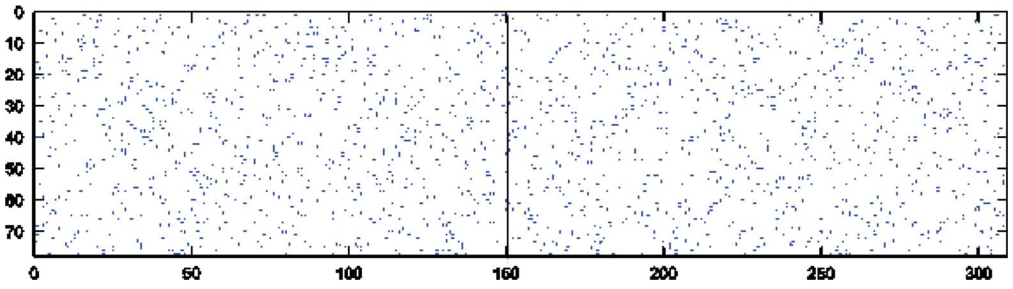
Fig. 4. Parity check matrix for the quantum LDPC code in Example 4.

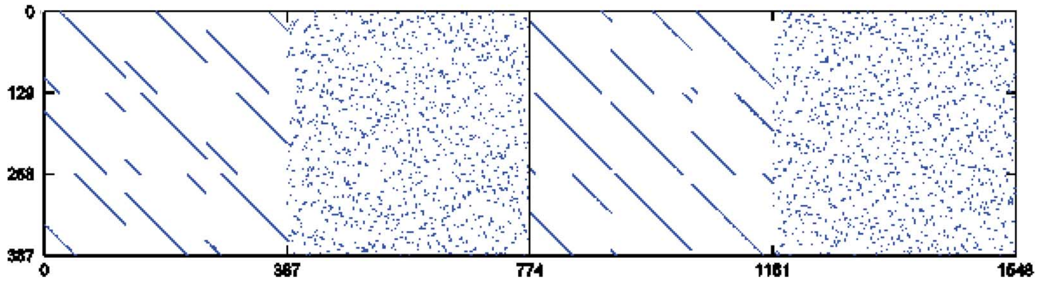
Fig. 5. Parity check matrix for the quantum LDPC code in Example 5.

code of length 154 qubits: $[\mathbf{X}_1, \mathbf{X}_1^T \mathbf{P}^T | \mathbf{X}_2, \mathbf{X}_2^T \mathbf{P}^T]$. The binary representation of this stabilizer generator, as depicted in Fig. 4, is a sparse and random matrix with nonuniform column weights of $2, 4$, and $6$.

*Example 5:* **[LDPC Stabilizer Codes from Construction III(3)]** There are multiple ways to obtain matrices that satisfy condition (3) in Theorem 4.5. The example here makes use of quasi-cyclic $\mathbf{X}_i$ $(i = 1, 2)$, whose symplectic inner product is made self-symmetric by properly selecting the cyclic offsets.

Consider forming two quasi-cyclic matrices $\mathbf{X}_1$ and $\mathbf{X}_2$ using a set of single-weight cyclic blocks placed in a $3 \times 3$ array

$$\mathbf{X}_1 = \begin{pmatrix} \mathbf{I}_{24}, & \mathbf{I}_{49}, & \mathbf{I}_{99} \\ \mathbf{I}_{99}, & \mathbf{I}_{24}, & \mathbf{I}_{49} \\ \mathbf{I}_{49}, & \mathbf{I}_{99}, & \mathbf{I}_{24} \end{pmatrix}$$
$$\mathbf{X}_2 = \begin{pmatrix} \mathbf{I}_{114}, & \mathbf{I}_{68}, & \mathbf{I}_{8} \\ \mathbf{I}_{8}, & \mathbf{I}_{114}, & \mathbf{I}_{68} \\ \mathbf{I}_{68}, & \mathbf{I}_{8}, & \mathbf{I}_{114} \end{pmatrix}. \quad (33)$$

With each cyclic block having size $129 \times 129$and their offsets as such, it is easy to verify that $\mathbf{X}_1 \mathbf{X}_2^T + \mathbf{X}_1^T \mathbf{X}_2$ is symmetric. Randomly select a scrambling matrix $\mathbf{P}$ and apply it to $\mathbf{X}_1$ and $\mathbf{X}_2$ according to Theorem 4.5. The resultant stabilizer generator, whose binary representation is shown in Fig. 5, defines a rate $1/2$ quantum LDPC code with code length $774$ qubits and a uniform column weight of $3$.

## E. Construction V: LDPC-Convolutional Construction

The existing quantum convolutional codes [14], [16], [17], being low-rate and single-error correcting codes, are rather inefficient. In contrast, the class of quantum convolutional codes developed here have high rates of $(n-1)/n$ and are capable of correcting many errors in a single block. They are based on the afore-discussed multiweight QC-LDPC construction (Construction I), and belong to non-CSS stabilizer codes.

Several attempts were made, in classic coding, to connect block codes and especially QC block codes with convolutional codes back in the seventies. Major advances surfaced after QC-LDPC codes, whose *quasi-cyclic, sparse* parity check matrices enabled the construction of a class of low-density convolutional codes, known in some literature as *LDPC-convolutional codes* [22], [23]. These convolutional codes result from replicating the constraint structure of the QC-LDPC block code to infinity, and can be encoded and decoded efficiently using streaming pipelines.

*Definition 4.6:* [Convolutional Codes] A rate $R = b/c$ (classic) convolutional code with codewords $v = (v_0, v_1, \ldots, v_t, \ldots)^T, v_t \in \mathbb{F}_2^c$ can be defined by an infinite parity check matrix $\tilde{\mathbf{H}}$, where $\tilde{\mathbf{H}} v = 0$. The transpose of $\tilde{\mathbf{H}}$, called the *syndrome former*, takes the form of[3]

$$\tilde{\mathbf{H}}^T = \begin{pmatrix} \mathbf{H}_0^T(0), & \cdots & \mathbf{H}_{m_s}^T(m_s), & & \\ & \ddots & & \ddots & \\ & & \mathbf{H}_0^T(t), & \cdots & \mathbf{H}_{m_s}^T(t+m_s) \\ & & \ddots & & \ddots \end{pmatrix}. \quad (34)$$

Each sub-matrix $\mathbf{H}_i^T(t+i)$ for $i = 1, \ldots, m_s$ is a $c \times (c-b)$ binary matrix. The largest $i$ with which $\mathbf{H}_i^T(t+i)$ is a nonzero matrix for some $t$ is the memory size and denoted as $m_s$.

Constructing $\tilde{\mathbf{H}}$ in (34) from a classic QC-LDPC code requires reorganizing the parity check matrix of the QC-LDPC code, $\mathbf{H}$, to a cyclic (*not* circularly cyclic) form, such that every $c$ columns become a vertically shifted version of the previous $c$ columns. This can be achieved by first unwrapping the circulant matrix $\mathbf{H}$ in a diagonal staircase manner, and then repeatedly stacking the unwrapped matrices in the same diagonal staircase

[3]For notational clarity, we use ˜ to represent the infinite matrices associated with convolutional codes.

TABLE I
SUMMARY OF FIVE QUANTUM CODES CONSTRUCTIONS

| Memo | Construction | Requirements |
|---|---|---|
| I. Multi-weight QC-LDPC construction | $\mathbf{A}_1 = \mathbf{I}_{a_1} + \mathbf{I}_{a_2} + \cdots + \mathbf{I}_{a_t}$<br>$\mathbf{A}_2 = \mathbf{I}_{b_1} + \mathbf{I}_{b_2} + \cdots + \mathbf{I}_{b_t}$ | $a_1 + a_2 = a_3 + a_4 = \cdots = a_{t-1} + a_t =$<br>$b_1 + b_2 = b_3 + b_4 = \cdots = b_{t-1} + b_t$, (mod m),<br>$a_1 \neq \cdots \neq a_t \neq b_1 \neq \cdots \neq b_t$,<br>$t$ is even. |
| II. $\pi$-rotation based construction and its extension | $\mathbf{A}_i = \begin{pmatrix} \mathbf{X}_i & \pi\{\mathbf{X}_i\} & \pi^2\{\mathbf{X}_i\} & \pi^3\{\mathbf{X}_i\} \\ \pi^3\{\mathbf{X}_i\} & X_i & \pi\{\mathbf{X}_i\} & \pi^2\{\mathbf{X}_i\} \\ \pi^2\{\mathbf{X}_i\} & \pi^3\{\mathbf{X}_i\} & \mathbf{X}_i & \pi\{\mathbf{X}_i\} \\ \pi\{\mathbf{X}_i\} & \pi^2\{\mathbf{X}_i\} & \pi^3\{\mathbf{X}_i\} & \mathbf{X}_i \end{pmatrix}$<br>for $i$=1, 2.<br>$\mathbf{A}_i = \left(\pi^t\{\mathbf{X}_i\}, \pi^k\{\mathbf{X}_i\}\right), i = 1, 2$ | $\mathbf{X}_1$ and $\mathbf{X}_2$ are cyclic (square) matrices of arbitrary weights and offsets. $t$ and $k$ are arbitrary integers. |
| III. General-rotation based construction | $\mathbf{A}_i = \begin{pmatrix} \mathbf{X}_i & (\Pi(\mathbf{X}_i^T))^T \\ \Pi(\mathbf{X}_i) & \Pi^2(\mathbf{X}_i) \end{pmatrix}$<br>for i=1, 2. | If both $\mathbf{X}_1$ and $\mathbf{X}_2$ are symmetric ($\mathbf{X}_i = \mathbf{X}_i^T$),<br>or if $\mathbf{X}_1$ and $\mathbf{X}_2$ are cyclic and square. |
| IV. Extension from general-rotation based construction | $\mathbf{A}_i = \begin{pmatrix} \{\mathbf{X}_i\} & \mathbf{X}_i^T\mathbf{P}^T\} \end{pmatrix}$<br>for i=1, 2. | If (1) $\mathbf{X}_1$ and $\mathbf{X}_2$ are both symmetric; or (2) $\mathbf{X}_1$ and $\mathbf{X}_2$ are cyclic square matrices; or (3) if $\mathbf{X}_1\mathbf{X}_2^T + \mathbf{X}_1^T\mathbf{X}_2$ is self-symmetric. |
| V. LDPC-convolutional construction | $\mathbf{A}_1(D) = D^{a_1} + D^{a_2} + \cdots + D^{a_t}$<br>$\mathbf{A}_2(D) = D^{b_1} + D^{b_2} + \cdots + D^{b_t}$ | $a_1 + a_2 = a_3 + a_4 = \cdots = a_{t-1} + a_t =$<br>$b_1 + b_2 = b_3 + b_4 = \cdots = b_{t-1} + b_t$,<br>$a_1 \neq \cdots \neq a_t \neq b_1 \neq \cdots \neq b_t$,<br>$t$ is even. |

manner. We recommend using $\mathcal{D}$-domain polynomials instead of binary formats for compact representation. Before exemplifying this procedure, recall that a stabilizer quantum code is constructed from a pair of classic codes whose parity check matrices satisfy the zero-SIP. This symplectic inner product condition is translated, in the context of convolutional codes, to the following lemma.

*Lemma 4.7:* [Symplectic Inner Product Condition for Quantum Convolutional Codes] [14] Consider two $\mathcal{D}$-domain polynomials $\mathbf{A}_1(D)$ and $\mathbf{A}_2(D)$. In order for $(\mathbf{A}_1(D)|\mathbf{A}_2(D))$ to represent the stabilizer for a quantum convolutional code, it requires

$$\mathbf{A}_1(D)\mathbf{A}_2(1/D)^T + \mathbf{A}_2(D)\mathbf{A}_1(1/D)^T = 0. \tag{35}$$

*Theorem 4.8:* [Sufficient and Necessary Condition for Convolutional Codes Converted From Weight-2 Cyclic Matrices to Satisfy Zero-SIP] Let $\mathbf{A}_1(D) = D^{a_1} + D^{a_2}$ and $\mathbf{A}_2 = D^{b_1} + D^{b_2}$, where $a_1 \neq a_2$ and $b_1 \neq b_2$. The necessary and sufficient condition to satisfy $\mathbf{A}_1(D)\mathbf{A}_2(1/D)^T + \mathbf{A}_2(D)\mathbf{A}_1(1/D)^T = 0$ is

$$a_1 + a_2 = b_1 + b_2. \tag{36}$$

*Proof:* This theorem can be proved using a similar approach as in Theorem 4.1. A sketch is provided in the Appendix.

*Example 6:* **[LDPC-Convolutional Stabilizer Codes]** Consider $\mathbf{A}_1$ in the $[5,1]$ cyclic stabilizer code in Example 1.

It translates to a $\mathcal{D}$-domain representation of $\mathbf{A}_1(D) = D^0 + D^3 = 1 + D^3$. An equivalent but less compact representation in the form of Definition (4.6) may be obtained by unwrapping $\mathbf{A}'_1$ in a diagonal staircase manner and keeping stacking the

$$\mathcal{A}'_1 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix} = I_c - I_3$$

$$\tilde{\mathbf{A}}_1 = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ & & & & & & & \ddots \end{pmatrix}$$

where $(\mathbf{H}_0^T(0), \mathbf{H}_1^T(1), \mathbf{H}_2^T(2), \mathbf{H}_3^T(3)) = (1,0,0,1)$, and $m_s = 3$.

Similarly, the $\mathbf{A}_2$ matrix in Example 1 translates to $\mathbf{A}_2(D) = D^1 + D^2$. The resultant LDPC-convolutional code therefore has a stabilizer $(\mathbf{A}_1(D)|\mathbf{A}_2(D))$.

In the original block code in Example 1, the last row is omitted due to its linear dependency on other rows. When converting it to a convolutional code, the omitted row must be patched in order for the cyclic pattern to repeat. In general, the convolutional code has a (slightly) lower code rate than the QC block counterpart. In this example, the convolutional stabilizer code has rate 0, while the original block stabilizer code has rate $1/5$. Horizontal concatenation can be used to increase the code rate.

*Example 7:* **[LDPC-Convolutional Stabilizer Codes]** Consider a QC-LDPC stabilizer code (from Construction I) with stabilizer matrix $(\mathbf{A}_1|\mathbf{A}_2)$, where

$$\mathbf{A}_1 = [\mathbf{I}_0 + \mathbf{I}_{121}, \mathbf{I}_8 + \mathbf{I}_{79}]$$
$$\mathbf{A}_2 = [\mathbf{I}_1 + \mathbf{I}_{120}, \mathbf{I}_6 + \mathbf{I}_{81}]$$

and all the cyclic square matrices $\mathbf{I}_t$ have a dimensionality of $217 \times 217$. Using polynomial expressions, we get the $\mathcal{D}$-domain equivalences:

$$\mathbf{A}_1(D) = (1 + D^{121},\ D^8 + D^{79})$$
$$\mathbf{A}_2(D) = (D + D^{120}, D^6 + D^{81}). \tag{37}$$

$(\mathbf{A}_1(D)|\mathbf{A}_2(D))$ forms a convolutional stabilizer code of rate exactly $1/2$, and the code length can be arbitrary.

For readers' convenience, the five systematic constructions are summarized in Table I.

## V. Decoding Strategies

For stabilizer codes derived from classic codes, the additional constraints imposed and the difference of the resultant quantum code with their classic counterparts have been profoundly emphasized, but the implementation of decoding procedures has not received adequate attention. For CSS (LDPC) quantum codes, the (message-passing) decoder is largely identical to that of the classic codes and does not require much illustration. However, for general stabilizer codes, the architecture needs to be modified, because the parity check matrix (i.e., the stabilizer) takes the GF(4) formalism, yet the syndrome takes binary forms. This section explicitly elucidates the decoding architecture for stabilizer LDPC codes and stabilizer LDPC-convolutional codes.

### *A. Decoding of LDPC Stabilizer Codes*

Decoding of quantum stabilizer codes generally takes two steps. The first step estimates errors that are compatible with the observed syndromes, and the second step recovers the erroneous qubits based on the estimated errors.

From the discussion in Section II, a stabilizer can either be described in the binary form, $\mathbf{A} = (\mathbf{A}_1|\mathbf{A}_2)$, where $\mathbf{A}_1$ and $\mathbf{A}_2$ satisfy zero-SIP, or in a quaternary form, where $\mathbf{A}_1$ and $\mathbf{A}_2$ are packed into a single quaternary matrix with elements $I, X, Y$ and $Z$. Since $\{I, X, Z, Y\}$ correspond to $\{0, 1, \omega, \omega^2 = \bar{\omega}\}$ in GF(4), where $\omega$ is the primary element in GF(4)[4], a stabilizer code takes a similar flavor as a GF(4) classic code. However, a distinctive feature of stabilizer decoding is that the *syndrome*, rather than a noisy reception of the codeword (as in the classic case), is available for use in error estimation and recovery. Furthermore, when the stabilizer is constructed from a pair of matching classic binary codes, the syndrome takes *binary*, rather than quaternary, values. Let $M_j = (m_{1j}, m_{2j})$ be a binary row in $(\mathbf{A}_1|\mathbf{A}_2)$, and let $E = (e_1, e_2)$ be a binary error vector. Without loss of generality, rewrite $E$ and $M_j$ in GF(4) as: $E = e_1 + e_2\omega, M_j = m_{1j} + m_{2j}\omega$. The syndrome corresponding to $E$ is computed as

$$\begin{aligned} s_j(E) &= \text{trace}\left(E \cdot \bar{M}_j^T\right) \\ &= \text{trace}((e_1 + e_2\omega)\cdot(m_1 + m_2\bar{\omega})^T) \\ &= \text{trace}\left(e_1 m_1^T + e_1 m_2^T\bar{\omega} + e_2 m_1^T\omega + e_2 m_2^T\omega\bar{\omega}\right) \\ &= e_1 \cdot m_2^T + e_2 \cdot m_1^T \end{aligned} \tag{38}$$

which becomes a binary vector (since $\text{trace}(0) = \text{trace}(1) = 0$ and $\text{trace}(\omega) = \text{trace}(\bar{\omega}) = 1$).

Now consider an LDPC stabilizer $\mathbf{A} = (\mathbf{A}_1|\mathbf{A}_2)$, where the code rates of the classic codes $\mathbf{A}_1$ and $\mathbf{A}_2$ are both $R$, and the code rate of the quantum code is $R_Q = R$. Suppose the binary sparse matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ each have a factor graph as shown in Fig. 6(A), where small boxes represent parity check nodes and small circles represent variable nodes. To put binary syndromes in perspective, consider merging all the "parity checks" (boxes) from $\mathbf{A}_1$ with those from $\mathbf{A}_2$ one-by-one in their original order; see Fig. 6(B). The result is a factor graph pertaining to a classic binary LDPC code with parity check matrix $[\mathbf{A}_1, \mathbf{A}_2]$ and rate $(1+R)/2$, which righteously captures the syndrome formalism in (38). Hence decoding this LDPC stabilizer is equivalent to

[4]Element $I$ must map to $0$; the mapping between $X, Y, Z$ and $1, \omega, \omega^2$ is arbitrary, as long as the mapping is one-to-one.

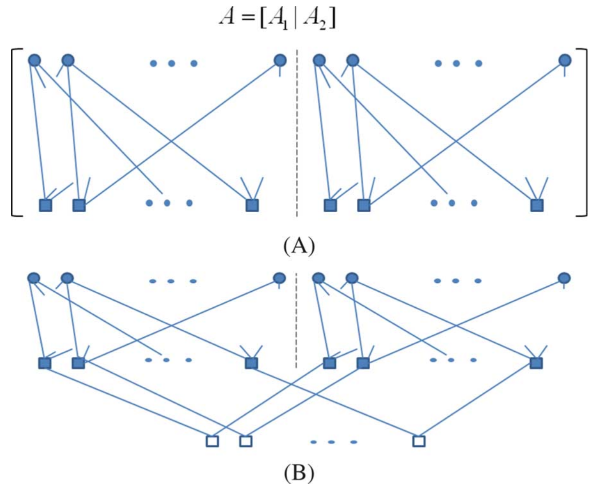

Fig. 6. Factor graph to illustrate the decoding of quantum LDPC code.

a syndrome version of message passing on the code graph of $[\mathbf{A}_1, \mathbf{A}_2]$.

### *B. Decoding of LDPC-Convolutional Stabilizer Codes*

Before discussing decoding algorithms for LDPC-convolutional stabilizer codes, let us first review the decoding technique for classic LDPC convolutional codes that result from classic QC-LDPC codes. Since these convolutional codes generally have very large memories $m_s$, traditional trellis-based decoding techniques take a prohibitive amount of storage on the order of $O(2^{m_s})$. Hence, LDPC-convolutional codes take rather nonconventional ways that directly use parity-check matrices to decode [23].

The most popular method to decode a classic LDPC-convolutional code is to apply a windowed message-passing algorithm on its sparse Tanner graph [23]. The decoding window, having a size of $(m_s + 1)I$, where $m_s$ is the memory size and $I$ is the desired number of decoding iterations, slides continuously across the received sequence from one bit position to another. At each window position, a total of $I$ processors simultaneously perform message-passing decoding, each on *a different* parity-check, for a total of $I$ iterations in a pipelined fashion. In each iteration, the $i$th processor takes in messages from the $(i-1)$th processor, exchanges message once with the $m_s + 1$ variable-nodes (code-bits) incident to its designated parity-check, and then passes the updated messages onto the $(i+1)$th processor. After an initial delay that equals the window size $(m_s+1)I$, the decoder starts to output decoding results, one bit at a time. More detailed discussion of this decoding strategy can be found in [23]. Compared to the trellis-based algorithm, this decoder requires a significantly smaller amount of memory $(O((m_s + 1)I))$, and runs in a highly parallelized fashion.

Now consider quantum LDPC-convolutional codes. Like stabilizer block codes, decoding here must rely on *binary-valued syndrome vectors* rather than the noisy reception of codewords. The result is a syndrome-version of the windowed message-passing decoder on an "equivalent" classic LDPC-convolutional code whose transfer polynomial is $\mathbf{A}_1(D^2) + D\mathbf{A}_2(D^2)$. The following example illustrates this point.

*Example 6:* **[Decoding LDPC-convolutional stabilizer codes]** Consider the LDPC-convolutional code in Example 4, where $\mathbf{A}(D) = [\mathbf{A}_1(D)|\mathbf{A}_2(D)] = (1 + D^3|D + D^2)$. We can represent $\mathbf{A}_1(D)$ and $\mathbf{A}_2(D)$ using (infinite) matrices and treat the code as a stabilizer block code. From the previous discussion, the equivalent matrix on which the syndrome version message passing is performed as (39) shown at the bottom of the page. To conform this matrix to Definition 4.6, interleave the right half columns with the left half, such that the original left half columns take all the odd column positions and the original right half columns take all the even positions as shown in (40) at the bottom of the page.

This new (infinite) matrix corresponds to a convolutional code with transfer polynomial, $(1 + D^3 + D^5 + D^6)$, which is equivalent to $\mathbf{A}_1(D^2) + D\mathbf{A}_2(D^2)$.

While LDPC-convolutional quantum codes are good in their own merits (such as flexible lengths), one of our initiatives was to reduce length-4 cycles in the QC-LDPC quantum codes we constructed (Construction I). This goal is fulfilled, since it is proven in [21] that the cycles in the LDPC-convolutional code can only result from the originating QC-LDPC code, yet not all the cycles in the QC-LDPC code will preserve. Further, the free distance $d_{\text{free}}$ of the LDPC-convolutional code is lower-bounded by that minimum distance $d_{\min}$ of the QC-LDPC code [21].

It is also worth noting that the proposed LDPC-convolutional stabilizer codes with rate $(\tau - 1)/\tau$ are, to the best of the authors' knowledge, the first class of high-rate quantum convolutional codes. There are only a few quantum convolutional codes reported in literature, and all of them have rather low rates such as $1/4$ (e.g., [17] and [16]).

## VI. Code Performance

### A. Simulation Results

There are several ways to characterize a quantum channel, some of which relate to classic channel models [12]. The most popular one is the depolarizing channel, where bit flips ($X$ errors), phase flips ($Z$ errors), and bit-and-phase flips ($Y$ errors) occur independently and with equal probability $f/3$. Another one models the quantum channel as two independent binary symmetric channels (BSC), in which $X$ errors and $Z$ errors occur independently (and may co-occur to a qubit) with equal probability $p$ [12]. This makes bit flips, phase flips, and bit-and-phase flips to happen with probabilities of $p(1-p), p(1-p)$ and $p^2$, respectively, and therefore a total flip probability of $f = 2p - p^2$. Our simulations use both the depolarizing channel and the independent BSC channel, and we measure the frame error rate (FER) and the qubit error rate (BER) as a function of the total flip probability $f$.

We evaluated the performance of the proposed constructions.

1) Code A is a QC-LDPC stabilizer code resulted from Construction I. It has a codeword length of $N = 1034$ qubits, code rate of $R_Q \approx 1/2$, and a stabilizer generator $[\mathbf{A}_1|\mathbf{A}_2]$, where

$$\mathbf{A}_1 = [\mathbf{I}_0 + \mathbf{I}_{121} + \mathbf{I}_{137} + \mathbf{I}_{258}, \mathbf{I}_8 + \mathbf{I}_{79} + \mathbf{I}_{85} + \mathbf{I}_{156}] \tag{41}$$

$$\mathbf{A}_2 = [\mathbf{I}_1 + \mathbf{I}_{120} + \mathbf{I}_{138} + \mathbf{I}_{257}, \mathbf{I}_6 + \mathbf{I}_{81} + \mathbf{I}_{83} + \mathbf{I}_{158}] \tag{42}$$

where the identity matrix has a dimensionality of $517 \times 517$. The submatrices $\mathbf{A_1}$ and $\mathbf{A_2}$ both have row weight 4 and column weight 4, and both have some length-4 cycles.

2) Code B is the block stabilizer code described in Example 3, which is based on Construction III (general-rotation construction). It has code rate $R_Q \approx 1/2$ and length $N = 2068$ qubits.

3) Code C is the LDPC-convolutional stabilizer code described in Example 7. It has code rate $R_Q = 1/2$, and length $N = 534$ qubits.

The FER performance of Code A and Code B on depolarizing channels is presented in Fig. 7, and compared to the two quantum LDPC codes reported in [15]. These two comparison codes both have rate $1/2$, and one has row weight 3, column weight 6, and length $N = 3600$ qubits, and the other has row

$$\begin{bmatrix}
1&0&0&1&0&0&0&0&\cdots&0&1&1&0&0&0&0&0&\cdots\\
0&1&0&0&1&0&0&0&\cdots&0&0&1&1&0&0&0&0&\cdots\\
0&0&1&0&0&1&0&0&\cdots&0&0&0&1&1&0&0&0&\cdots\\
0&0&0&1&0&0&1&0&\cdots&0&0&0&0&1&1&0&0&\cdots\\
0&0&0&0&1&0&0&1&\cdots&0&0&0&0&0&1&1&0&\cdots\\
0&0&0&0&0&1&0&0&\cdots&0&0&0&0&0&0&1&1&\cdots\\
\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdots&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdots
\end{bmatrix}. \tag{39}$$

$$\begin{bmatrix}
1&0&0&1&0&1&1&0&0&0&0&0&0&0&0&0&\cdots\\
0&0&1&0&0&1&0&1&1&0&0&0&0&0&0&0&\cdots\\
0&0&0&0&1&0&0&1&0&1&1&0&0&0&0&0&\cdots\\
0&0&0&0&0&0&1&0&0&1&0&1&1&0&0&0&\cdots\\
0&0&0&0&0&0&0&0&1&0&0&1&0&1&1&0&\cdots\\
0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&1&\cdots\\
\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdot&\cdots
\end{bmatrix}. \tag{40}$$

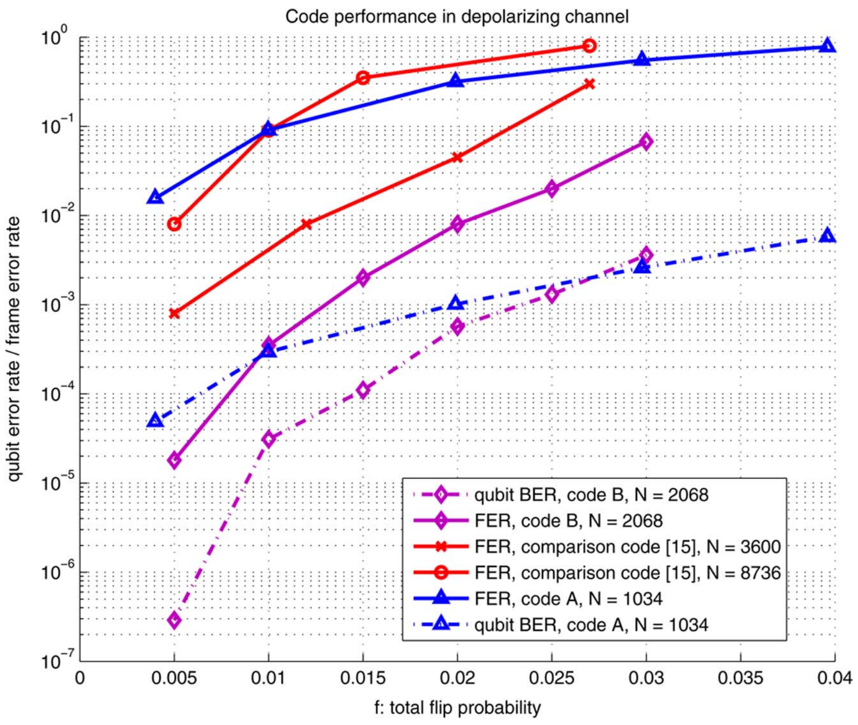

Fig. 7. Frame error rate and qubit error rate of the quantum QC-LDPC code with length $N = 1034$ qubits (Code A) and quantum LDPC code with length $N = 2068$ qubits (Code B) compared with two quantum LDPC codes reported in [15].

weight 4, column weight 8, and length $N = 8736$ qubits. Despite the shorter block sizes, Code A of length $N = 1034$ qubits performs on par with the comparison code of length $N = 8736$ qubits[5], and Code B of length $N = 2068$ qubits notably outperforms both comparison codes. We also plot the BER curves of our codes to give an idea of how BER may relate to FER.

The BER and FER performance of Code C, the LDPC-convolutional stabilizer code, on depolarizing channels is provided in Fig. 8. The windowed message-passing decoding is performed with $I = 100$ iterations. We do not have a comparison curve, since ours is the first successful simulation of quantum convolutional codes known to date.

## *B. Rate Bounds*

[12] discussed benchmark communication rates for symmetric channels. Three rate bounds on classic symmetric channels are discussed: capacity (the Shannon limit) of the BSC, the classic Gilbert–Varshamov (GV) rate of the BSC, and the capacity of the classic 4-ary symmetric channel

$$C_{\mathrm{BSC}}(f_m) = 1 - H_2(f_m) \tag{43}$$
$$R_{\mathrm{GV}}(f_m) = 1 - H_2(2f_m) \tag{44}$$
$$C_{4B}(f_m) = 1 - (2H_2(3f_m/2) + 3f_m \log_2 3)/4 \tag{45}$$

where $C_{4B}(f_m)$ denotes the maximum rate at which reliable communication can be achieved over each half of the 4-ary symmetric channel with marginal flip probability $f_m$. For dual-containing CSS stabilizer codes, since the rate of the quantum code $R_Q$ relates to the rate of the classic code $R$ by $R_Q = 2R - 1$, [12] derived three corresponding rate bounds for this particular class of codes.

Here we consider the general case of unrestricted quantum codes with stabilizer $\mathbf{A} = (\mathbf{A}_1|\mathbf{A}_2)$ and binary-valued syndromes. Three kinds of code rates are of relevance as follows.

1) $R_Q$: the code rate of the quantum code.
2) $R_c$: the code rate of the pair of classic binary codes from which the quantum code is constructed, namely, the code rate corresponds to parity check matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ (assuming $\mathbf{A}_1$ and $\mathbf{A}_2$ bear the same code rate). Clearly, $R_Q = R_c$.
3) $R_s$: the code rate of the "equivalent" classic (binary) code with which decoding is performed, namely, the code rate of the classic code with parity check matrix $[\mathbf{A}_1\mathbf{A}_2]$. We have $R_s = (R_c + 1)/2$, or, $R_c = 2R_s - 1$.

When deriving the rate bounds for non-CSS codes, it is appropriate to consider $R_Q = R_c = 2R_s - 1$, and to upper-limit $R_s$ by the channel capacity.

Likewise, a dual-containing CSS code with stabilizer, $\mathbf{A} = \begin{pmatrix}\mathbf{H} & 0\\ 0 & \mathbf{H}\end{pmatrix}$can be related to the same three kinds of code rates: $R_Q, R_c$ and $R_s$. Since the off-diagonal blocks are zero, the performance of the CSS quantum code can be approximated by that of its diagonal block $\mathbf{H}$ (with rate $R_c$). Hence, $R_Q = 2R_c - 1, R_c = R_s$, and $R_s$ is upper bounded by the channel capacity.

It is interesting to note that the relations between $R_s$ and $R_Q$ are identical for both CSS and non-CSS codes, and $R_s$ is restricted by the upper bounds in (43)–(45). The immediate implication is that the same rate bounds [12] derived for CSS codes also hold for non-CSS codes

$$C_{\mathrm{BSC}}^{\mathrm{quantum}}(f_m) = 1 - 2H_2(f_m) \tag{46}$$

[5]The simulation results of Code A in our ISIT paper [30] was incorrect. The authors apologize for their mistake and for any confusion it may have caused.

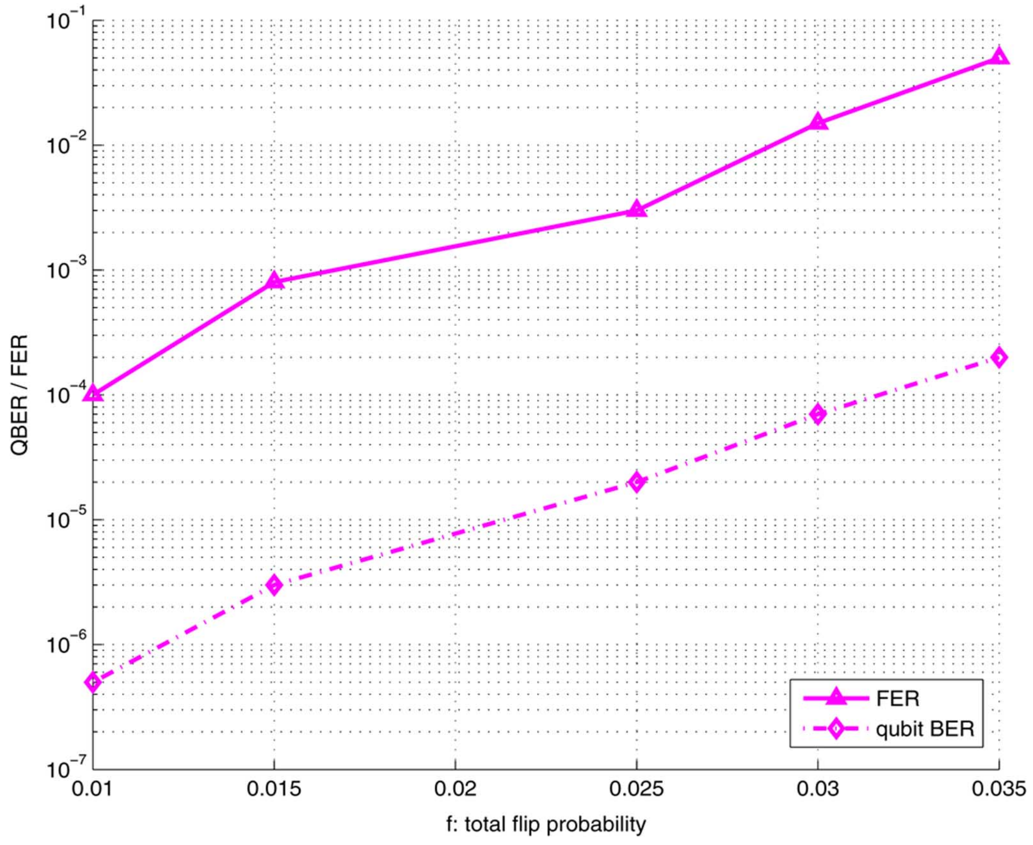

Fig. 8. Performance of the proposed LDPC-convolutional stabilizer code (Code C) on depolarizing channel. $m_s = 258, I = 100$.

$$R^{\text{quantum}}_{\text{GV4}}(f_m) = 1 - 2H_2(2f_m) \quad (47)$$

$$C^{\text{quantum}}_{4}(f_m) = 1 - (H_2(3f_m/2) + (3f_m/2)\log_2 3). \quad (48)$$

However, having the same asymptotic bounds does not necessarily imply identical error correction capability at *finite lengths*. As discussed before, to correct a single error, the non-CSS construction takes 2 qubits fewer than the CSS construction. The performance difference can also be illuminated from their code structures and decoding features. Consider a dual-containing CSS code and a non-CSS code having the same code rate $R_Q$ and codeword length $N$ qubits. The classic binary code equivalent to the non-CSS code has a parity check matrix with $2N$ columns and $N(1-R_Q)$ rows, whereas the code equivalent to the (dual-containing) CSS code (i.e., a diagonal block) has a parity check matrix that is just half the size, $N$ columns and $N(1-R_Q)/2$ rows. In other words, while (dual-containing) CSS codes and non-CSS codes both converge to the same asymptotic rate bounds, the latter converges faster.

The rate bounds in (46)–(48) are plotted in Fig. 9, together with our Code B (at three block sizes and code rates) and a CSS LDPC code proposed in [12]. Both codes are simulated using the symmetric BSC channel model with the marginal noise level $f_m$ [12], and both are evaluated at the block error probability of $10^{-4}$. Our construction is simpler, but falls slightly short of that in [12].

## VII. Conclusion

We have developed simple and systematic ways to construct five rich classes of non-CSS stabilizer codes, including four classes of LDPC stabilizer codes and one class of convolutional stabilizer codes. These codes, designed from the (general) symplectic inner product condition rather than the special CSS case, provide a wide range of rates and lengths. The exploitation of general-purpose operations, such as row-concatenation, row-deletion, row-insertion and one-sided addition, allow us to derive new codes from the existing ones to further enrich the family of stabilizer code.

## Appendix

• *Proof of Theorem 4.1:* We start from calculating their symplectic inner product

$$\begin{aligned}
&\mathbf{A}_1\mathbf{A}_2^T + \mathbf{A}_2\mathbf{A}_1^T \\
&\quad = (\mathbf{I}_{a_1} + \mathbf{I}_{a_2})(\mathbf{I}_{b_1} + \mathbf{I}_{b_2})^T \\
&\qquad + (\mathbf{I}_{b_1} + \mathbf{I}_{b_2})(\mathbf{I}_{a_1} + \mathbf{I}_{a_2})^T \\
&\quad = (\mathbf{I}_{a_1} + \mathbf{I}_{a_2})(\mathbf{I}_{-b_1} + \mathbf{I}_{-b_2}) \\
&\qquad + (\mathbf{I}_{b_1} + \mathbf{I}_{b_2})(\mathbf{I}_{-a_1} + \mathbf{I}_{-a_2}) \\
&\quad = \mathbf{I}_{(a_1-b_1)\bmod m} + \mathbf{I}_{(a_1-b_2)\bmod m} \\
&\qquad + \mathbf{I}_{(a_2-b_1)\bmod m} + \mathbf{I}_{(a_2-b_2)\bmod m} \\
&\qquad + \mathbf{I}_{(b_1-a_1)\bmod m} + \mathbf{I}_{(b_1-a_2)\bmod m} \\
&\qquad + \mathbf{I}_{(b_2-a_1)\bmod m} + \mathbf{I}_{(b_2-a_2)\bmod m}. \quad (49)
\end{aligned}$$

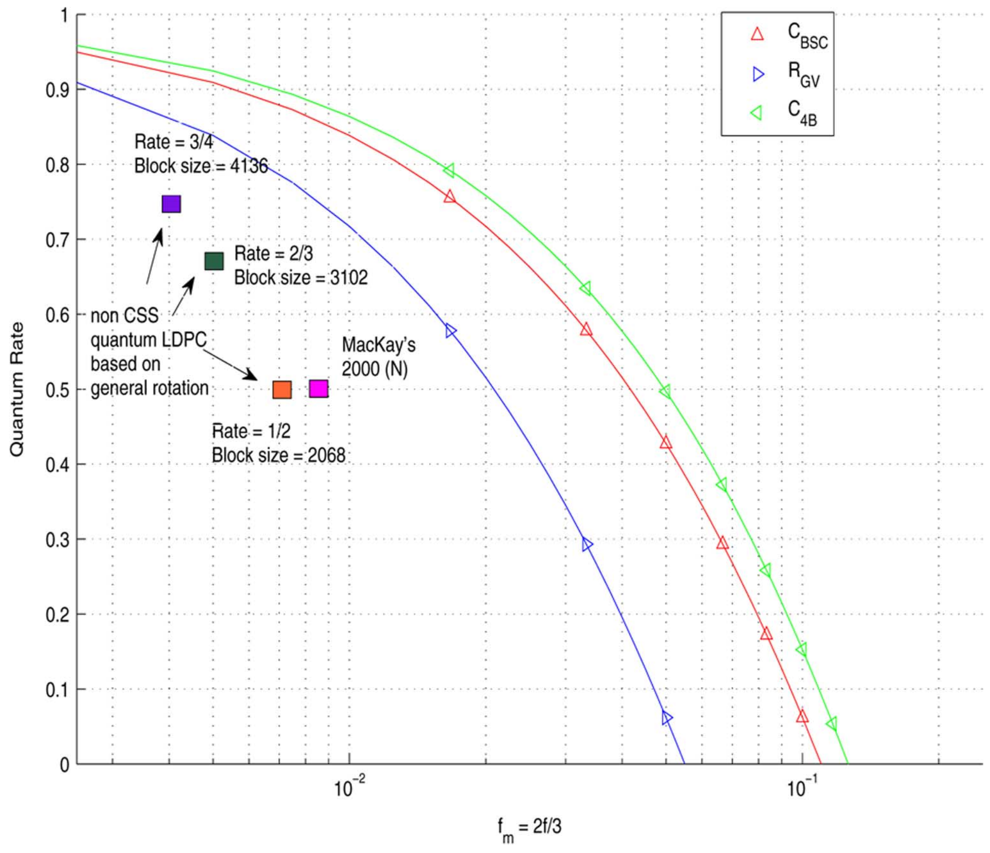

Fig. 9. Bounds for CSS and non-CSS codes, performance comparison with MacKay's quantum code at length 2000 qubits [12].

We consider three cases:

CASE 1: When $b_1 < b_2 < a_1 < a_2$.

Let $\lambda_1 = a_1 - b_1, \lambda_2 = a_1 - b_2, \lambda_3 = a_2 - b_1, \lambda_4 = a_2 - b_2$. We have $\lambda_i \geq 0$ for $i = 1, 2, 3, 4$.

$$\begin{aligned}\mathbf{A}_1\mathbf{A}_2^T + \mathbf{A}_2\mathbf{A}_1^T &= \mathbf{I}_{(a_1-b_1)} + \mathbf{I}_{(a_1-b_2)} + \mathbf{I}_{(a_2-b_1)} + \mathbf{I}_{(a_2-b_2)} \\ &\quad + \mathbf{I}_{(b_1-a_1)+m} + \mathbf{I}_{(b_2-a_1)+m} \\ &\quad + \mathbf{I}_{(b_1-a_2)+m} + \mathbf{I}_{(b_2-a_2)+m} \\ &= \mathbf{I}_{\lambda_1} + \mathbf{I}_{\lambda_2} + \mathbf{I}_{\lambda_3} + \mathbf{I}_{\lambda_4} + \mathbf{I}_{m-\lambda_1} + \mathbf{I}_{m-\lambda_2} \\ &\quad + \mathbf{I}_{m-\lambda_3} + \mathbf{I}_{m-\lambda_4}. \end{aligned} \tag{50}$$

These matrices cancel each other if their offsets, $\lambda_1, \lambda_2, \lambda_3, \lambda_4, m - \lambda_1, m - \lambda_2, m - \lambda_3, m - \lambda_4$, cancel. Since $\lambda_1 - \lambda_2 = \lambda_3 - \lambda_4 = b_2 - b_1 > 0$, we convert this problem to an equivalent problem of canceling out four segments:

$\Lambda_1$: starts at $\lambda_1$, ends at $\lambda_3$;

$\Lambda_2$: starts at $\lambda_2$, ends at $\lambda_4$;

$\Lambda_3$: starts at $m - \lambda_3$, ends at $m - \lambda_1$;

$\Lambda_4$: starts at $m - \lambda_4$, ends at $m - \lambda_2$.

If $\Lambda_1$ cancels out $\Lambda_2$ and $\Lambda_3$ cancels out $\Lambda_4$, then $\lambda_1 = \lambda_2$ and $\lambda_3 = \lambda_4$. This results in $b_1 = b_2$, (i.e., $\mathbf{A_2} = \mathbf{0}$), which violates the assumption.

If $\Lambda_1$ cancels out $\Lambda_3$ and $\Lambda_2$ cancels out $\Lambda_4$, then $\lambda_1 = m - \lambda_3$ and $\lambda_2 = m - \lambda_4$. This again results in $b_1 = b_2$, or, $\mathbf{A_2} = \mathbf{0}$).

If $\Lambda_1$ cancels out $\Lambda_4$ and $\Lambda_2$ cancels out $\Lambda_3$, then $\lambda_1 = m - \lambda_4$ and $\lambda_2 = m - \lambda_3$. This results in $a_1 + a_2 = b_1 + b_2 + m$ and $a_1 \neq a_2 \neq b_1 \neq b_2$.

Similar treatment can be applied to CASE 2 when $b_1 < a_1 < b_2 < a_2$, and CASE 3 when $b_1 < a_1 < a_2 < b_2$. Analysis of CASE 2 leads to i) $a_1 + a_2 = 2b_2 = 2b_1 + m$, or ii) $b_1 + b_2 = 2a_2 - m = 2a_1$, or iii) $a_1 + m/2 = a_2$ and $b_1 + m/2 = b_2$. Analysis of CASE 3 leads to i) $b_1 = b_2$, or ii) $a_1 + a_2 = b_1 + b_2$.

In general, CASE 3 does not apply to odd $m$, but all the cases apply to even m. Gathering all these cases leads to Theorem 4.1. □

• *Proof of Theorem 4.8:*

$$\begin{aligned}\mathbf{A}_1(D)\mathbf{A}_2(1/D)^T + \mathbf{A}_2(D)\mathbf{A}_1(1/D)^T &= (D^{a_1} + D^{a_2})(D^{-b_1} + D^{-b_2})^T \\ &\quad + (D^{b_1} + D^{b_2})(D^{-a_1} + D^{-a_2})^T \\ &= D^{(a_1-b_1)} + D^{(a_1-b_2)} + D^{(a_2-b_1)} \\ &\quad + D^{(a_2-b_2)} + D^{(b_1-a_1)} \\ &\quad + D^{(b_1-a_2)} + D^{(b_2-a_1)} + D^{(b_2-a_2)}. \end{aligned} \tag{51}$$

Let $\lambda_1 = a_1 - b_1, \lambda_2 = a_1 - b_2, \lambda_3 = a_2 - b_1, \lambda_4 = a_2 - b_2$. Without loss of generality, assume $a_2 > a_1, b_1 > b_2$; hence $\lambda_4 > \lambda_3, \lambda_2 > \lambda_1$, but the signs of $\lambda_i$ $(i = 1, 2, 3, 4)$ are unknown.

To warrant zero-SIP, we group $\lambda_1, \lambda_2, \lambda_3, \lambda_4, -\lambda_1, -\lambda_2, -\lambda_3, -\lambda_4$ into appropriate pairs, such that elements in each pair cancel out. Noticing $\lambda_1 - \lambda_2 = \lambda_3 - \lambda_4$, we convert this problem to the cancelation of four equal-length segments:

$\Lambda_1$: starts at $\lambda_1$, ends at $\lambda_2$;

$\Lambda_2$: starts at $\lambda_3$, ends at $\lambda_4$;

$\Lambda_3$: starts at $-\lambda_2$, ends at $-\lambda_1$;

$\Lambda_4$: starts at $-\lambda_4$, ends at $-\lambda_3$.

Possible cases to cancel out these segments include the following.

If $\Lambda_1$ cancels out $\Lambda_3$ and $\Lambda_2$ cancels out $\Lambda_4$, then $\lambda_1 = -\lambda_2$ and $\lambda_3 = -\lambda_4$. This results in $a_1 = a_2$, or, $\mathbf{A}_1(D) = 0$.

If $\Lambda_1$ cancels out $\Lambda_2$, and $\Lambda_3$ cancels out $\Lambda_4$, then $\lambda_1 = \lambda_3$ and $\lambda_2 = \lambda_4$. This results in $a_1 = a_2$ and $b_1 = b_2$, which implies $\mathbf{A}_1(D) = 0$ and $\mathbf{A}_2(D) = 0$.

If $\Lambda_1$ cancels out $\Lambda_4$, and $\Lambda_2$ cancels out $\Lambda_3$, then $\lambda_1 = -\lambda_4$ and $\lambda_2 = -\lambda_3$. This results in $a_1 + a_2 = b_1 + b_2$ and $a_1 \neq a_2 \neq b_1 \neq b_2$; otherwise, $\mathbf{A}_1(D) = 0$ or $\mathbf{A}_2(D) = 0$.

Gathering all these cases, we get that $\mathbf{A}_1(D)$ and $\mathbf{A}_2(D)$ satisfy the zero-SIP condition in (36) if and only if $a_1+a_2 = b_1+b_2$ and $a_1 \neq a_2 \neq b_1 \neq b_2$. □

## REFERENCES

[1] E. Knill and R. Laflamme, "A theory of quantum error correcting codes," *Phys. Rev. A*, vol. 55, pp. 900–911, 1997.

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

[3] R. Laflamme, C. Miguel, P. Paz, and W. Zurek, "Perfect quantum error correcting code," *Phys. Rev. Lett.*, vol. 77, pp. 198–201, 1996.

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

[5] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Phys. Rev. A*, vol. 54, pp. 1098–1106, 1996.

[6] A. Steane, "Multiple particle interference and quantum error correction," *Proc. Roy. Soc. Lond. A*, vol. 452, pp. 2551–, 1996.

[7] 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.

[8] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[9] D. Gottesman, "Stabilizer Codes and Quantum Error Correction," Ph.D. dissertation, California Institute of Technology, Pasadena, CA, 1997, quant-ph/9705052.

[10] A. R. Calderbank, E. M. Rains, P. M. Shor, and N. J. A. Sloane, "Quantum error correction via codes over GF(4)," *IEEE Trans. Inf. Theory*, vol. 44, no. 4, pp. 1369–1387, July 1998.

[11] M. S. Postol, "A proposed quantum low density parity check code," quant-ph/0108131.

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

[13] H. Lou and J. Garcia-Frias, "On the application of error-correcting codes with low-density generator matrix over different quantum channels," in *Proc. Int. Symp. Turbo Codes*, Apr. 2006.

[14] H. Ollivier and J.-P. Tillich, "Description of a quantum convolutional code," *Phys. Rev. Lett.*, vol. 91, no. 17, p. 1779021-4, 2003.

[15] T. Camara, H. Ollivier, and J.-P. Tillich, "Constructions and performance of classes of quantum LDPC codes," arXiv quant-ph/0502086.

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

[17] G. D. Forney, Jr., M. Grassl, and S. Guha, "Convolutional and tail-biting quantum error-correcting codes," Nov. 2005, ArXiv quant-ph 0511016v1.

[18] M. Grassl and T. Beth, "Quantum BCH codes," arXiv quant-ph/9910060.

[19] M. Grassl and T. Beth, Quantum Reed-Solomon Codes arXiv quant-ph/9910059.

[20] A. Ashikhmin and E. Knill, "Non-binary quantum stabilizer codes," *IEEE Trans. Inf. Theory*, 2000.

[21] R. M. Tanner, D. Sridhara, A. Sridharan, T. E. Fuja, and D. J. Costello, "LDPC block and convolutional codes based on circulant matrices," *IEEE Trans. Inf. Theory*, vol. 50, pp. 2966–2984, Dec. 2004.

[22] R. M. Tanner, Convolutional Codes Form Quasi-Cyclic Codes: A Link Between the Theories of Block and Convolutional Codes Univ. Calif. Santa Cruz, Tech. Rep., 1987.

[23] A. J. Feltstrom and K. S. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," *IEEE Trans. Inf. Theory*, vol. 45, pp. 2181–2191, Sept. 1999.

[24] M. Shin, J. Kim, and H. Song, "Minimum distance bounds of irregular QC-LDPC codes and their applications," in *Proc. IEEE Int. Symp. Inf. Theory*, Chicago, IL, Jun. 2004.

[25] R. Echard and S. C. Chang, "The wF-rotation low-density parity check codes," in *Proc. GLOBECOM 2001*, Nov. 2001, pp. 980–984.

[26] M. Hagiwara and H. Imai, "Quantum quasi-cyclic LDPC codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Nice, France, Jun. 2007.

[27] V. Aggarwal and A. R. Calderbank, "Boolean functions, projection operators and quantum error correcting codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Nice, France, June 2007.

[28] M. Grassl and M. Roetteler, "Constructions of quantum convolutional codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Nice, France, Jun. 2007.

[29] T. Brun, I. Devetak, and M. Hsieh, "General entanglement-assisted quantum error-correcting codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Nice, France, June 2007.

[30] P. Tan and J. Li, "On construction of two classes of efficient quantum error-correction codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Nice, France, Jun. 2007.

[31] R. Smarandache and P. O. Vontobel, "On regular quasi-cyclic LDPC codes from binomials," in *Proc. 2004 IEEE Int. Symp. Inf. Theory*, 2004, p. 274.

[32] I. B. Djordjevic, "Quantum LDPC codes from balanced incomplete block designs," *IEEE Commun. Lett.*, vol. 12, no. 5, May 2008.

[33] S. A. Aly, "A class of quantum LDPC codes constructed from finite geometries," in *Proc. IEEE GLOBECOM*, Dec. 2008.

**Peiyu Tan** (S'05) received the B.S. and M.S. degrees in electrical engineering from Tianjin University, Tianjin, China, in 1999 and 2002, respectively.

She is currently working toward the Ph.D. degree in the Department of Electrical and Computer Engineering, Lehigh University, Bethlehem, PA. She spent summers 2007, 2008, and 2009 with LSI Corporation, Infineon Technologies, and DoCoMo Communications Laboratories USA, conducting coding and MIMO research in cellular and satellite communication systems. Her research interests include communication and signal processing, coding and information theory. Her recent research focuses on error correction coding, turbo equalization, distributed source coding, and quantum coding.

**Jing Li** (S'98–M'02–SM'08) received the Bachelor's degree in computer science from Peking University, Beijing, China, in 1997 and the Master's and the Ph.D. degrees in electrical engineering from Texas A&M University, College Station, in 1999 and 2002, respectively.

After receiving the Ph.D. degree, she joined the Electrical and Computer Engineering Department at Lehigh University, Bethlehem, PA, where she now holds an Associate Professorship. Her research falls in the general area of communication, signal processing and networking, with special focus on error correction coding, packet erasure coding, network coding, and quantum coding.

Dr. Tiffany was the recipient of the 2001 TAMU Ethel Ashworth-Tsutsui Memorial Award for Research. She also served as a symposium Co-Chair for IEEE GLOBECOM'06, WirelessCom'06, ChinaCom'07, and IEEE ICC'08.