Hypergraph Product of Convolutional Codes

Nicholas Lyu
Physics Department, Massachusetts Institute of Technology
(Dated: May 12, 2023)

Abstract

This study explores quantum convolutional codes, focusing on the application of hypergraph products for their construction, a novel approach in quantum coding theory. Our research demonstrates that the hypergraph product of convolutional codes yields valid CSS-type codes which retain their convolutional structure. Despite the challenges associated with the lower rates of CSS-type quantum convolutional codes, our findings underscore the potential of generalizing product constructions. Additionally, we present quantum adaptations of the message and syndrome Viterbi algorithms, enabling efficient decoding of quantum convolutional codes even in the presence of syndrome noise.

I. INTRODUCTION

Classical convolutional codes constitute a powerful class of error-correcting codes, playing an indispensable role in digital communication systems due to their intuitive nature and efficient decoding ability. Unlike block codes, convolutional codes employ a memory-based structure, generating encoded output based on the convolution of the input sequence. This characteristic often allows them to outperform linear block codes in terms of code rate given bit error rate, while also enabling continuous data encoding [1].

In comparison to linear block codes, the quantum counterparts of convolutional codes remain less studied. Although earlier studies have introduced mathematical tools to analyze classical convolutional codes, only a few systematic constructions of quantum convolutional codes exist [2–6].. Existing works suggest that quantum convolutional codes, like their classical counterparts, may offer higher code rates than quantum block codes with comparable error-correcting properties. Previous constructions either demonstrate single cases (not yielding a family of codes with various rates and distances) or impose orthogonality constraints on the parent codes.

The hypergraph product construction by Tillich and Zémor introduces the possibility of a less constrained construction. The hypergraph product of two arbitrary linear block codes results in a CSS block code with a linear rate and minimum distance proportional to the square root of the block length. This project aims to investigate whether the hypergraph product of classical convolutional codes with linear block codes and convolutional codes are valid quantum codes, and how we can efficiently decode these codes.

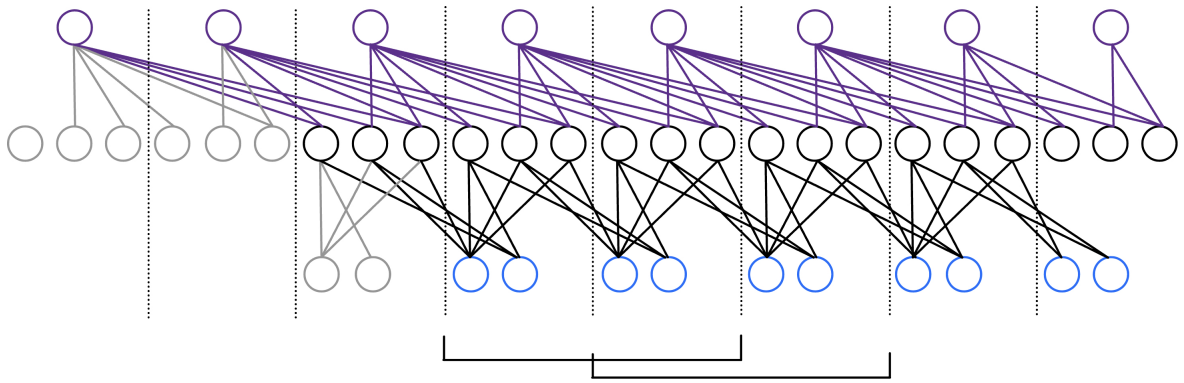

FIG. 1: Relation between input bits (blue), message bits (black), syndrome bits (purple), and padding bits (gray) for the example $(3, 2)$-CC

II. CONVOLUTIONAL CODES

A. Encoding and Syndrome Equations

The encoding process for convolutional codes (CC) employs a sliding window to calculate the message from input bits. Message bits are computed by combining different subsets of input bits within the window using addition in $\mathbb{F}_2$. The window size, referred to as the code's constraint length m, overlaps with itself and slides by k bits. When the process outputs n bits per window, the asymptotic rate of the code is k/n.

For the remainder of this project, we consider the $(n = 3, k = 2)$-CC, given by the generating relations as presented in [7]:

$$\begin{cases} y_1^t = x_1^t + x_2^t + x_2^{t+1} \; y_2^t = x_1^t + x_1^{t+1} + x_2^{t+1} \; y_3^t = x_1^t + x_1^{t+1} \end{cases} \tag{1}$$

Given an input sequence, we can split it into two sequences x_1^t, x_2^t and use the relations above to obtain three output sequences y_1^t, y_2^t, y_3^t, which can be merged into one sequence as required. From the window perspective, the window of this code outputs $n = 3$ bits per window of constraint length $m = 4$, and the window slides in strides of $k = 2$.

As illustrated in 1, the sliding window partitions input and encoded bits into frames of size 2 and 3, respectively. We can introduce a delay operator D to formally capture the dependence between frames: D is a map on discrete time functions $f : \mathbb{N} \to F$ such that $(D, f)(n) \equiv f(n-1)$. We can then represent an input sequence $x_{i0 \leq t}^t$ via its D-transform:

This leads us to the polynomial formalism of classical convolutional codes:

Definition II.1 (Classical Convolutional Code) *A (n, k)-convolutional code over a*

field $\mathbb{F}$ *linearly maps the set of* k *sequences* $\mathbf{x} = \begin{bmatrix} x_1(D) & x_2(D) & \cdots & x_k(D) \end{bmatrix}^T$ *to the set of* $n \geq k$ *sequences* $\mathbf{y} = \begin{bmatrix} y_1(D) & \cdots & y_n(D) \end{bmatrix}^T$ *via the relation* $\mathbf{y} = G\mathbf{x}$*. Its generator*

$$G = \begin{bmatrix} g_{11}(D) & g_{12}(D) & \cdots & g_{1k}(D) \\ g_{21}(D) & g_{22}(D) & \cdots & g_{2k}(D) \\ \vdots & & & \\ g_{n1}(D) & g_{n2}(D) & \cdots & g_{nk}(D) \end{bmatrix}$$

is a $n \times k$ *matrix of polynomials of* D *over* $\mathbb{F}$ *with maximum degree* M *with invariant factor decomposition* $G = A \begin{bmatrix} I_k & \mathbf{0} \end{bmatrix} B$*.*

The constraint length of the code is seen to be $(M+1)k$. Since we need to introduce Mk zero padding bits at the beginning of the sequence, the CC encodes kt input bits into $(n+1)t$ bits. The code has asymptotic rate k/n in the limit $k \to \infty$. It's easy to see from the graph above that the minimum-weight message sequence we can generate from a nontrivial input sequence has weight 3. We can also read the encoding and parity-check matrices G, H for our example code off the figure above:

$$G = \begin{bmatrix} 1 & 1+D \\ 1+D & D \\ 1+D & 0 \end{bmatrix} \qquad H = \begin{bmatrix} D+D^2 & 1+D^2 & 1+D+D^2 \end{bmatrix}$$

We denote by $\mathbb{F}[D]$ the ring of all polynomials of D over $\mathbb{F}$. Our generating matrix G generates a module over $\mathbb{F}[D]$ just as a matrix with field elements generates vector spaces (column space) over the underlying field. In particular, $\mathbb{F}[D]$ is a principal ideal domain: its units are zero-degree polynomials, primes are monic irreducible polynomials, and each polynomial has a unique prime factorization up to units. The main theorem concerning modules over a principle ideal domain is the following structure theorem [8]:

Theorem II.2 (Invariant-Factor Theorem) *A* $n \times k$ *matrix over* $\mathbb{F}[D]$ *has the invariant-factor decomposition* $G = A\Gamma B$ *where* A, Γ, B *are* $n \times n, n \times k, k \times k$ *respectively. Moreover,* A, B *are products of elementary operations (given below) and both have inverses with elements in* $\mathbb{F}[D]$*. The invariant factors of* G *are encoded in* Γ *of form*

$$\Gamma = \begin{bmatrix} \Gamma_1 \\ \mathbf{0}_{(n-k)\times k} \end{bmatrix} \quad \Gamma_1 = \mathrm{diag}(\gamma_1, \gamma_2, ..., \gamma_k)$$

Forney has shown that in order for the encoder G to be non-catastrophic i.e. a finite number of message bit errors not causing infinite number of decoding errors, all invariant factors of G must be 1. Leaving the computational procedure to the appendix, the invariant factor decomposition of the example encoder G is given by

$$G = \begin{bmatrix} 1 & 0 & 0 \\ 1+D & 1+D+D^2 & 1+D \\ 1+D & 1+D^2 & D \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1+D \\ 0 & 1 \end{bmatrix} = A\Gamma B$$

Apart from giving the condition for a CC to be non-catastrophic, the invariant factor decomposition also allows us to compute the parity-check relations H from the encoding relation G. Consider

$$A = [A_1 \mid A_2] \quad A^{-1} = \begin{bmatrix} A_1' \\ A_2' \end{bmatrix}$$

where A_1, A_2, A_1', A_2' have dimensions $n \times k,\ n \times (n-k),\ k \times n,\ (n-k) \times n$. Then

$$A^{-1}A = \begin{bmatrix} A_1'A_1 & A_1'A_2 \\ A_2'A_1 & A_2'A_2 \end{bmatrix} = \begin{bmatrix} I_k & \\ & I_{n-k} \end{bmatrix}$$

constitutes a valid choice of the parity-check matrix is seen to be $H = A_2'$ since

$$HG = A_2' \begin{bmatrix} A_1 \mid A_2 \end{bmatrix} \begin{bmatrix} \Gamma_1 \\ \mathbf{0}_{n-k} \end{bmatrix} B = \begin{bmatrix} A_2'A_1 & A_2'A_2 \end{bmatrix} \begin{bmatrix} \Gamma_1 \\ \mathbf{0}_{n-k} \end{bmatrix} B = \begin{bmatrix} \mathbf{0}_{(n-k)\times k} & \mathbf{I}_{n-k} \end{bmatrix} \begin{bmatrix} \Gamma_1 \\ \mathbf{0}_{n-k} \end{bmatrix} B = \mathbf{0}$$

In particular, for $(n,1)$-CCs of form $G = \begin{bmatrix} g_1 & \cdots & g_n \end{bmatrix}^T$, we may choose

$$H = \begin{bmatrix} g_2 & g_1 & 0 & \cdots & 0 \\ g_3 & 0 & g_1 & \cdots & 0 \\ \vdots & & & \ddots & \\ g_n & 0 & 0 & \cdots & g_1 \end{bmatrix}$$

The parity-check matrix for our example code obtained by the procedure above is $H = A_2' = \begin{bmatrix} D+D^2 & 1+D^2 & 1+D+D^2 \end{bmatrix}$. It corresponding to the syndrome equation $s^t = y_1^{t+1} + y_1^{t+2} + y_2^t + y_2^{t+1} + y_3^t + y_3^{t+1} + y_3^{t+2}$

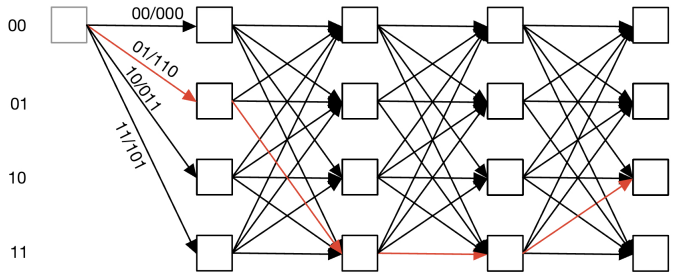

FIG. 2: Trellis of the $(3, 2)$-CC. Transitions are associated with an input (new frame) and output (newly encoded message). The path corresponding to the encoding 01111110 is shown in red

B. Message and Syndrome Viterbi Decoding

The Viterbi algorithm is an efficient maximum-likelihood decoding algorithm for convolutional codes [9]. The encoding process of a convolutional code can be viewed as a finite state machine with memory, consisting of the contents of the encoding window less the earliest frame and transitions according to the contents of the new encountered frame of the current timestep. Each transition is associated with n output bits, and the Viterbi algorithm outputs the sequence of inputs to the state machine that generates the output closest to the received message sequence. The algorithm maintains a trellis which unrolls all possible operations of the state machine along the temporal axis. There are 2^{Mk} trellis states and 2^k transitions at every single step. Both the space and runtime of the Viterbi algorithm are $O(T \cdot 2^{k(M+1)})$ for an input sequence of length Tk.

The Viterbi algorithm, as originally developed, is not immediately suitable for quantum purposes since it requires access to the full message sequence. The analogue of the original Viterbi algorithm (now referred to as "message Viterbi"), which only accesses the syndrome as given in [7], is a straightforward optimization of the following idea: given a syndrome sequence $S = \begin{bmatrix} s_1 & \cdots & s_{n-k} \end{bmatrix}^T$ for a (n, k)-CC whose parity-check matrix has a maximum degree M. The syndrome vector $s^{t_0}_{1\leq j\leq n-k}$ at time t_0 uniquely depends on $e^{t_0-(M+1)\leq t\leq t_0}_{1\leq j\leq n}$. Assuming operation over $\mathbb{F} = 0, 1$, we can build a trellis with 2^{Mn} states corresponding to unique values of $e^{t_0-(M+1)\leq t<t_0-1}_{1\leq j\leq n}$ and 2^n transitions from each state corresponding to new values of $e^{t_0-M\leq t<t_0}_{1\leq j\leq n}$ under the current timestep. The space and time complexity of this syndrome Viterbi algorithm 1 is also $O(T \cdot 2^{n(M+1)})$.

Some thought suggests that we do not need to store all 2^{nM} possible preceding error

Algorithm 1 Simple Syndrome Viterbi Algorithm over $\mathbb{F}_2$

Require: Code parameters (n, k, M); syndrome function $S : \{0,1\}^{n\times(M+1)} \to \{0,1\}^{n-k}$ computing syndrome for single time-step; syndrome sequence vector $s_{1\le j\le n-k}^{1\le t\le T}$. The syndrome should be computed on padded transmitted sequence $y_{1\le j\le n}^{1\le t\le T+M}$ such that $y_j^{T<t\le T+M} = 0$

$D_{i\in\{0,1\}^M} \leftarrow 0; \quad D'_{i\in\{0,1\}^M} \leftarrow \infty$

$t \leftarrow 0$

while $t < T$ **do**

  for $e_{t:t+M} \in \{0,1\}^{n\times M}$ **do**

    for $e_{t+M} \in \{0,1\}^n$ **do**

      Compute branch syndrome $p^t = S\left[e_{t:t+M} \mid e_{t+M}\right] \in \{0,1\}^{n-k}$

      if $p_{1\le j\le n-k}^t = s_{1\le j\le n-k}^t$ **then**

        $D'_{e_{t+1:t+M+1}} \leftarrow \min\left(D'_{e_{t+1:t+M+1}}, D_{e_{t:t+M}} + \|e_{t+M}\|\right)$

        Record $P^t_{e_{t+1:t+M+1}} \leftarrow e_t$ if an update happened

      end if

    end for

  end for

  $D_{i\in\{0,1\}^M} \leftarrow D'_i; \quad D'_{i\in\{0,1\}^M} \leftarrow \infty$

  $t \leftarrow T + 1$

end while

D_{0^M} records the minimum-weight of the most likely error, and backtracing via $P^T_{0^M}$ yields the vector of most likely error sequences $e_{1\le j\le n}^{1\le t\le T}$

terms: $\ker H$ has dimension $n-k$, so we should be able to reduce our states and transitions by a factor of $2^{M(n-k)}$ and 2^{n-k}, respectively. We may solve for this compression by solving for the free part of the syndrome in $s = He$ and only including that in the trellis. The space and time complexity after this optimization via compression is equal to that of the message Viterbi algorithm. Due to its algebraic complexity and comparable performance compared to message Viterbi, syndrome Viterbi decoding is rarely considered for classical CCs. For this reason, classical CCs are frequently specified in terms of G rather than H.

It is of note that syndrome extraction itself (for non-catastrophic CCs) may be viewed as a convolutional encoding process. All relevant processes associated with a CC is given by the diagram below:

$$x \xrightarrow{G} y = Gx \xrightarrow{\text{noise}} \hat{y} = Gx + e \xrightarrow{H} s = He$$

In particular, message and syndrome Viterbi decoding are two different algorithms we may

run on an arbitrary convolutional encoding process $u \xrightarrow{F} v$: message Viterbi decodes x as the output of the general procedure $V_m(v, F) = \text{argmin}_u \|v - Fu\|$ for $v = \hat{y}$, $F = G$; syndrome Viterbi decodes e as the output of $V_s(v, F) = \text{argmin}_{Fu=v} \|u\|$ for $v = s, F = H$.

C. Quantum Convolutional Codes

The polynomial formalism and parity-checks of classical CCs suggest a straightforward generalization to quantum convolutional codes (QCCs): we replace the binary parity-checks with pairs of X, Z-checks. Formally, a (n, k)-QCC is defined by the subspace of the countably infinite Hilbert space stabilized by the pauli subgroup generated by all k-qubit shifts of a set of basic generators on $m \geq n$ qubits. For example, the $(5, 1, 2)$-QCC as given in [10] has basic generators

$$\begin{bmatrix} ZXXZIII \\ IZXXZII \\ IIZXXZI \\ IIIZXXZ \end{bmatrix} \leftrightarrow \left[\begin{array}{ccccccc|ccccccc} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right]$$

With $n = 5$, all $i > 5$ qubit are multiplied by D and translated to the left by 5 qubits, yielding polynomial representation

$$H = \left[\begin{array}{ccccc|ccccc} 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & D & 0 & 1 & 0 & 0 \\ D & 0 & 0 & 0 & 1 & 0 & D & 0 & 1 & 0 \end{array}\right]$$

Compared to quantum block codes, commutativity requirement is slightly more subtle: in order for all generators to commute, all time-shifts of the basic generators must commute. Formally, given basic stabilizer $P(D) = (P_X(D) \mid P_Z(D))$ and let $P^j(D)$ denote the pair of binary vectors corresponding to coefficients of D^j, the generators generated by P, Q pairwise commute if and only if

$$\forall a, b : D^a[P]D^b[Q] = D^b[Q]D^a[P] \iff P_X(D)Q_Z(1/D) + P_Z(D)Q_X(1/D) = 0$$

Forney et. al proposes a construction of CSS-type QCC by by a self-orthogonal CC, similar to the block code CSS construction [4]. We may additionally construct a QCC

via a CC over $\mathbb{F}_4$ that is self-orthogonal under the symplectic inner product [3]. However, both construction require parent codes to obey self-orthogonality constraints in addition to commutativity constraints on the basic generators. The self-orthogonal constraint is additionally required precisely to accomodate QCC's additional requirement that all time-shifts of basic generators, instead of the generators themselves, must commute.

III. HYPERGRAPH PRODUCT

The hypergraph product allows for the construction of a CSS code from two arbitrary linear block codes. Our analysis adopts the notations in [11]: a classical code can be expressed in terms of its bipartite Tanner graph $\mathcal{T}(V, C, E)$, which represents the parity-check relations between check bits C and code bits V. Given block codes with Tanner graphs $\mathcal{T}(V_1, C_1, E_1)$ and $\mathcal{T}(V_2, C_2, E_2)$, their hypergraph product is denoted by the Tanner graph with message qubits $V_1 \times V_2 \cup C_1 \times C_2$ and X, Z check bits $C_1 \times V_2, C_2 \times V_1$, respectively. A check bit $c_1 v_2$ checks a qubit of type $v_1' v_2'$ if $v_2 = v_2'$ and $(c_1, v_1') \in E_1$; it is incident upon a qubit of type $c_1' c_2'$ if $c_1 = c_1'$ and $(c_2', v_2) \in E_2$, with similar relations for the other type of check. To generalize to the hypergraph product of convolutional codes, we consider taking the hypergraph products of potentially infinite Tanner graphs.

Initially, we consider the hypergraph product of a Convolutional Code (CC) $\mathcal{C}_1$ with a linear block code $\mathcal{C}_2$, exemplified by the example $(3, 2)$-CC with the 3-qubit repetition code. The Tanner graphs of the parent codes are displaced on the two axes, and the sequence of 3×3 and 2×1 qubits correspond to the message qubits. The hypergraph product retains the familiar convolutional code structure: each frame now contains $n_p = n_1 n_2 + k_1 k_2 = 11$ qubits, with the parity-check window encompassing $(M_1 + 1)n_p = 33$ qubits.

In particular, the hypergraph construction yields a valid quantum convolutional code. Different from block codes, checks and message bits in the CC $\mathcal{C}_1$ are denoted by pairs in $[n_1 - k_1] \times \mathbb{N}, [n_1] \times \mathbb{N}$ respectively to account for the delay. Any X-check $q = ((c_1, t_0), v_2)$ and Z-check $p = ((v_1, t_1), c_2)$ commute: suppose both checks are incident upon a qubit $v = ((v_1', t_2), v_2')$, then

- q checks $v \iff v_2' = v_2$ and (c_1, t_0) checks (v_1', t_2) in $\mathcal{C}_1$.
- p checks $v \iff (v_1', t_2) = (v_1, t_1)$ and c_2 checks v_2' in $\mathcal{C}_2$.

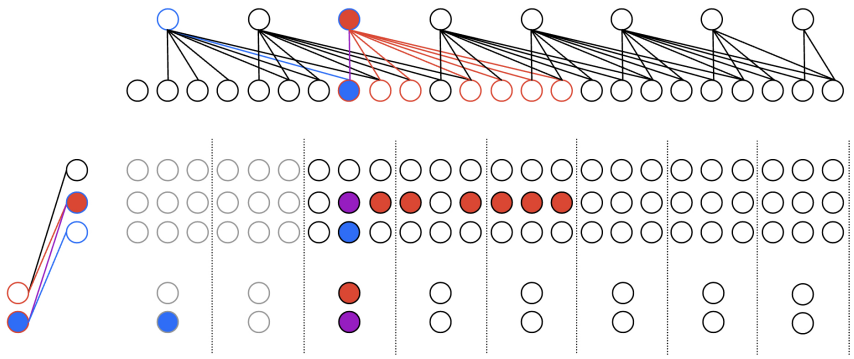

FIG. 3: C_1V_2 (blue): $C_1 = 1, V_2 = 1, t = 2$
C_2V_1 (red): $C_2 = 1, V_1 = 1, t = 2$

This implies that both q, p check v if and only if they check $w = ((c_1, t_3), c_2)$

- q checks $w \iff (c_1', t_3) = (c_1, t_0)$ and c_2' checks v_2 in $\mathcal{C}_2$.
- p checks $w \iff c_2' = c_2$ and (c_1', t_3) checks (v_1, t_1) in $\mathcal{C}_1$.

We see that X, Z-checks p, q overlap if and only if they simultaneously check $((v_1, t_1), v_2)$ and $((c_1, t_0), v_2)$ in the case when (c_1, t_0) checks (v_1, t_1) and c_2 checks v_2. This implies that any two X, Z checks p, q overlap either 0 or 2 times thus commute.

We can similarly show that the hypergraph product of two convolutional codes $\mathcal{C}_1, \mathcal{C}_2$ also have commuting X, Z stabilizers: $((v_1, t_1), (c_2, t_2))$ and $((c_1, t_1'), (v_2, t_2'))$ either simultaneously check $((v_1, t_1), (v_2, t_2'))$ and $((c_1, t_1'), (c_2, t_2))$ or do not overlap at all.

As shown in 4, we may conceptually consider the product Tanner graphs in 3 by collapsing each set of checks incident upon a certain time into a blue dot and each frame corresponding to 11 qubits into a black (gray) dot. The collapsed Tanner graphs (we shifted all parity-checks to the right by M timesteps without loss of generality) demonstrate the time-dependence of parity checks upon qubits in the previous frames. It can also be seen how the 2D hypergraph product of CCs result in a QCC with two independent time parameters and physical qubits being an infinite lattice, instead of a sequence, of frames each with $n_1n_2 + k_1k_2$ qubits.

A. Decoding

As noted in [6], the syndrome Viterbi decoding algorithm 1 readily generalizes to Quantum Convolutional Codes (QCCs): instead of sequences of binary bits, trellis states and

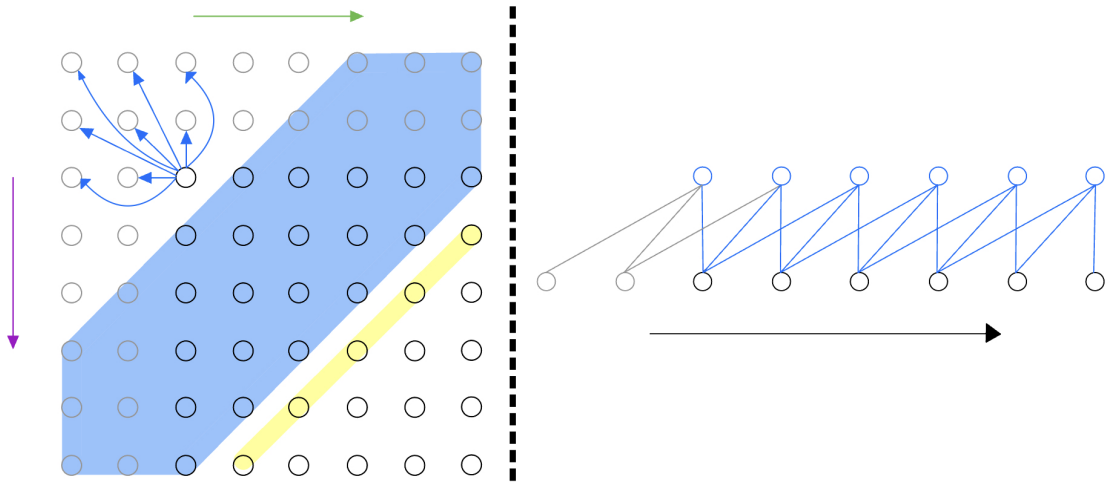

FIG. 4: Collapsed Tanner graphs of 2D and 1D product of $(3, 2)$-CC respectively

transitions are characterized by sequences of pairs of binary bits corresponding to X, Z operators. The quantum version of the syndrome Viterbi algorithm still outputs $V_s(s, H) = \text{argmin}_{He=s}|e|$, the minimum-weight (most likely) error that results in the observed syndrome.

However, we observe that the message Viterbi algorithm can also be applied to QCCs: instead of considering $V_m(y, G)$, for which we are not allowed access for quantum codes, we can consider $V_m(s, H) = \text{argmin}_e\|He - s\|$. When the syndrome is accurate (i.e., in $\text{Im}(H)$), V_m can always output an error estimate $He = s$ that matches the syndrome. Although this may not generally be the maximum-likelihood decoding which minimizes $\|e\|$, the decoded error is equivalent to the maximum-likelihood error up to the parity check and corrects the underlying quantum logical information as desired. Moreover, message Viterbi can accommodate syndrome measurement error: syndrome Viterbi simply fails when $s \notin \text{Im}(H)$ since there's no path in the trellis that results in the observed syndrome, but $V_m(s, H)$ outputs the maximum-likelihood error that results in the closest syndrome.

We also note that the 2D hypergraph product does not permit a practical generalization of Viterbi decoding: consider the 2D product of $(3, 2)$-CC in 4, the parity-checks associated with each frame depend on qubits in the $(M + 1) \times (M + 1)$ frames to its upper-left. Proceeding on both independent time axes simultaneously, the decoding algorithm traverses the diagonals of the lattice (the same conclusion holds if we prioritize one axis). To decode the parity-checks associated with frames on the highlighted diagonal in yellow, the trellis

needs to store all possible backtraces along the frames highlighted in blue. This means that the trellis grows linearly with time, ruling out the practicality of Viterbi decoding since its runtime grows exponentially with respect to the size of the trellis.

IV. CONCLUSION AND FUTURE WORK

This project focused on quantum convolutional codes, a less-explored area in quantum coding theory compared to linear block codes. We started with an overview of classical convolutional codes and the invariant-factor theorem, which allows us to compute the parity-check relations from the encoding relations. We then considered the quantum generalization of the Viterbi algorithm and its variant, the syndrome Viterbi algorithm, and shown that we may decode quantum convolutional codes efficiently and in the presence of syndrome errors. We explored the hypergraph product of convolutional codes and demonstrated that it results in valid CSS codes and retains the familiar convolutional code structure.

While the hypergraph product construction provided a less constrained path for constructing quantum convolutional codes, it has been noted in [6] that CSS-type QCCs have less comparable rates compared to QCCs constructed from self-orthogonal CCs over $\mathbb{F}_4$. However, this research demonstrated the possibility of extending block code product constructions to quantum convolutional codes, and it will be fruitful to explore the efficacy and properties of other product constructions such as those given in [12–14].

Additionally, the error-correcting properties of quantum convolutional codes warrant further consideration. A CC with distance d is guaranteed to correct up to $\lfloor (d-1)/2 \rfloor$ errors every other M frames. This provides a stronger assumption for fault-tolerance analysis compared to block codes which warrants further investigation.

Finally, future work can focus on exploring efficient decoding algorithms for product convolutional codes. While the generalization of the Viterbi algorithm to quantum codes presented here allows for efficient decoding of 1D products, we have seen that 2D products do not admit efficient decoding. Overall, this research has shed light on a novel approach to quantum convolutional codes as generalizations of block code product constructions, opening up new avenues of exploration and posing exciting questions for future research. It is our hope that this study will not only advance our understanding of these codes but also pave the way for the development of more efficient and reliable quantum communication systems.

[1] M. 602, 6.02 lecture 8: Convolutional coding ().

[2] G. D. Forney, M. Grassl, and S. Guha, Convolutional and tail-biting quantum error-correcting codes, IEEE Transactions on Information Theory **53**, 865 (2007).

[3] M. Grassl and M. Rotteler, Quantum block and convolutional codes from self-orthogonal product codes, in *Proceedings. International Symposium on Information Theory, 2005. ISIT 2005.* (IEEE, 2005) pp. 1018–1022.

[4] M. Grassl and M. Rotteler, Constructions of quantum convolutional codes, in *2007 IEEE International Symposium on Information Theory* (IEEE, 2007) pp. 816–820.

[5] A. A. De Almeida and R. Palazzo, A concatenated [(4, 1, 3)] quantum convolutional code, in *Information Theory Workshop* (IEEE, 2004) pp. 28–33.

[6] G. D. Forney and S. Guha, Simple rate-1/3 convolutional and tail-biting quantum error-correcting codes, in *Proceedings. International Symposium on Information Theory, 2005. ISIT 2005.* (IEEE, 2005) pp. 1028–1032.

[7] I. Reed, *New Syndrome Decoding Techniques for Convolutional Codes over GF (q)*, Tech. Rep. (ADAPTIVE SENSORS INC SANTA MONICA CA, 1983).

[8] G. Forney, Convolutional codes i: Algebraic structure, IEEE Transactions on Information Theory **16**, 720 (1970).

[9] M. 602, 6.02 lecture 9: Viterbi decoding of convolutional codes ().

[10] H. Ollivier and J.-P. Tillich, Quantum convolutional codes: fundamentals, arXiv preprint quant-ph/0401134 (2004).

[11] J.-P. Tillich and G. Zémor, Quantum ldpc codes with positive rate and minimum distance proportional to the square root of the blocklength, IEEE Transactions on Information Theory **60**, 1193 (2013).

[12] N. P. Breuckmann and J. N. Eberhardt, Balanced product quantum codes, IEEE Transactions on Information Theory **67**, 6653 (2021).

[13] A. Leverrier and G. Zémor, Quantum tanner codes, in *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)* (IEEE, 2022) pp. 872–883.

[14] P. Panteleev and G. Kalachev, Quantum ldpc codes with almost linear minimum distance, IEEE Transactions on Information Theory **68**, 213 (2021).

[15] A. O. Quintavalle and E. T. Campbell, Reshape: a decoder for hypergraph product codes, IEEE Transactions on Information Theory **68**, 6569 (2022).

Appendix A: Computing Invariant Factors

We compute the invariant-factor decomposition of G by reducing G into $\Gamma = A^{-1}GB^{-1}$ via a sequence of the following operations. A^{-1}, B^{-1} essentially record the row and column operations performed to reduce G, respectively.

1. Swapping rows (columns).

2. Multiplying row (column) i by $\alpha \in \mathbb{F}[D]$ and add to *another* row (column) j.

3. Multiply a row(column) by a nonzero scalar of $\mathbb{F}[D]$

4. *Remark:* multiplying a row / column itself by an non-unit element of $\mathbb{F}[D]$ is not an admissible elementary operation since the element may not have multiplicative inverse. Moreover, over $\mathbb{F}_2$ each elementary operation is its own inverse.

The computational algorithm for computing the invariant factor decomposition is outlined in [8]: given generating matrix

$$G = \begin{bmatrix} g_{11} & \cdots & g_{1k} \\ \vdots & \ddots & \vdots \\ g_{n1} & \cdots & g_{nk} \end{bmatrix}$$

let δ be the greatest common divisor of $g_{11}, ..., g_{1k}, g_{21}, ..., g_{n1}$, then there exists a linear combination of these elements yielding δ. Use this linear combination to eliminate all off-diagonal entries in the first row and column of G.

Recall our $(3, 2)$-CC with generating relations:

$$\begin{cases} y_1^t = x_1^t + x_2^t + x_2^{t+1} \\ y_2^t = x_1^t + x_1^{t+1} + x_2^{t+1} \\ y_3^t = x_1^t + x_1^{t+1} \end{cases} \qquad G = \begin{bmatrix} 1 & 1+D \\ 1+D & D \\ 1+D & 0 \end{bmatrix}$$

- Use g_{11} to eliminate g_{12}, g_{21}, g_{22}. This yields $\begin{bmatrix} 1 & 0 \\ 0 & 1+D+D^2 \\ 0 & 1+D^2 \end{bmatrix}$

- The greatest common divisor of g_{22}, g_{23} is 1: starting from $\begin{bmatrix}1+D+D^2 & 1+D^2\end{bmatrix}^T$, add second row to first row $\begin{bmatrix}D & 1+D^2\end{bmatrix}^T$, add D times first row to second row $\begin{bmatrix}D & 1\end{bmatrix}^T$. Add D times second row to first row and swap rows to yield $\begin{bmatrix}1 & 0\end{bmatrix}^T$, as desired.

Unrolling the elementary operations above yields

$$\Gamma = \begin{bmatrix}1 & 0\\ 0 & 1\\ 0 & 0\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\\ 1+D & D & 1+D\\ D+D^2 & 1+D^2 & 1+D+D^2\end{bmatrix} G \begin{bmatrix}1 & 1+D\\ 0 & 1\end{bmatrix} = A^{-1}GB^{-1}$$

from which we may solve for

$$G = \begin{bmatrix}1 & 0 & 0\\ 1+D & 1+D+D^2 & 1+D\\ 1+D & 1+D^2 & D\end{bmatrix} \begin{bmatrix}1 & 0\\ 0 & 1\\ 0 & 0\end{bmatrix} \begin{bmatrix}1 & 1+D\\ 0 & 1\end{bmatrix} = A\Gamma B$$

Appendix B: Compressing Syndromes

Let $L\begin{bmatrix}I_{n-k} & \mathbf{0}\end{bmatrix}$ be the decomposition of H, we may rewrite the syndrome equation

$$s = He = L\begin{bmatrix}I_{n-k} & \mathbf{0}\end{bmatrix} Re \iff L^{-1}s = \begin{bmatrix}I_{n-k} & \mathbf{0}\end{bmatrix}(Re) \iff \sigma = \begin{bmatrix}I_{n-k} & \mathbf{0}\end{bmatrix}\epsilon$$

by identifying $\sigma \equiv L^{-1}s; \quad e = R^{-1}\epsilon$. The general solution for ϵ is found to be

$$\epsilon_j = \begin{cases}\sigma_j & 1 \le j \le n-k\\ f_{j-(n-k)} & n-k < j \le n\end{cases}$$

Here f_j is the free part of the error vector. We can now consider a more efficient algorithm for syndrome decoding as follows:

- Given syndrome s, compute $\sigma = L^{-1}s$
- Consider trellis over states of possible free error components $f_{1\le j\le k}^{1\le t\le M} \in \{0,1\}^{M\times k}$ and transitions $f^{M+1} \in \{0,1\}^k$. Note that this trellis is the same for all codes with the same parameters (n,k,M): different codes only yield different branch metrics.
- The assumed noise $e_{1\le j\le n}^t \in \{0,1\}^n$ at time t associated with transitioning from $f_{t:t+M} \in \{0,1\}^{M\times k}$ via $f_{t+M} \in \{0,1\}^k$ is $R^{-1}\begin{bmatrix}\sigma_{1\le j\le n-k}^{t\le t'\le t+M}\\ f_{t:t+M} \mid f_{t+M}\end{bmatrix}$. More precisely, the

last term is applying R^{-1} to the D-transform of the given vector of length-$(M+1)$ sequences with $(n-k)+k$ components. Concatenation of $f_{t:t+M} \mid f_{t+M}$ is along the temporal axis while concatenation with σ is on the component axis.