# Convolutional Polar Codes on Channels with Memory

Benjamin Bourassa[1*], Maxime Tremblay[1] and David Poulin[1,2]
[1]Département de physique & Institut quantique, Université de Sherbrooke, Sherbrooke, Québec, Canada J1K 2R1
[2]Canadian Institute for Advanced Research, Toronto, Ontario, Canada M5G 1Z8
[*]benjamin.bourassa@usherbrooke.ca

***Abstract*—Arikan's recursive code construction was designed for memoryless channels, but was recently shown to also polarize channels with finite-state memory. The resulting successive cancellation decoder has a complexity that scales like the third power of the channel's memory size. Furthermore, the polar code construction was extended by replacing the block polarization kernel by a convoluted kernel. Here, we extend the polar code efficient decoding algorithm for channels with memory to the family of convolutional polar code. We use numerical simulations to study the performance of these algorithms for practically relevant code sizes and find that the convolutional structure outperforms the standard polar codes on a variety of channels with memory.**

## I. Introduction and Background

In some communication channels, errors tend to occur in burst which motivates the search for good error correcting protocols tailored to correlated noise models. Polar codes [1] are well known to achieve the capacity on symmetric memoryless channels and it was shown in [2] that they also achieve polarization on channels with memory. The successive cancellation decoder can be adapted to a noise channel with a $d$-state memory at the expense of an $d^3$ increase in complexity [3].

In [4], a generalization of polar codes was proposed which replaces the block-structured polarization kernel by a convolutional structure. The main motivation of this generalization is that it preserves the efficiency of successive cancellation decoding while significantly extending the code family. The design and analysis of convolutional polar codes are most easily formulated in terms of tensor networks. The efficiency of successive cancellation decoding of convolutional polar codes is a simple corollary of a well established fact in quantum many-body physics [5]. In [6], analytical and empirical evidence showed that convolutional polar codes outperform regular polar codes on memoryless channels both asymptotically and for finite code size, with only a small constant factor decoding and encoding overhead.

Here, we use tensor networks to describe how the convolutional polar code's successive cancellation decoder can be adapted to channels with a $d$-state memory with a $d^3$ complexity increase. For regular polar codes, this decoding algorithm reduces to the one of [3]. Furthermore, we use numerical simulations to investigate the performance of polar codes and convolutional polar codes on channels with memory, and find that in all cases, convolutional codes outperform the original construction.

### A. Channels with finite-state memory

A channel with a $d$-state memory consists of a collection of $d$ channels $W_C(Y|X)$ labeled by an internal state $C \in \{1, 2, \dots d\}$. When the channel is in state $C$, the transmitted symbol $X$ evolves according to the corresponding channel. Between each transmitted symbol, the channel's internal state evolves according to a Markov chain $P_{C\to C'}$. When this Markov chain is ergodic, it has a unique stationary distribution $P(C)$, which can be used to define the average channel

$$\overline{W}(Y|X) = \sum_C P(C) W_C(Y|X). \tag{1}$$

For sake of illustration, we will consider the Gilbert-Elliott channel [7] which has two internal states, labeled $G$ for *Good* and $B$ for *Bad*. When the channel is in state $G$, the transmitted bits are affected by a Binary Symmetric Channel $\mathrm{BSC}(h_G)$, and when the channel is in state $B$, the transmitted bits are affected by $\mathrm{BSC}(h_B)$. The good channel is less noisy than the bad channel, meaning $h_G < h_B$.

### B. Interleaving

The simplest method to mitigate burst errors is to interleave the bits – i.e., permute their order – before transmission and then apply the reversed interleaver upon reception. Denoting an $N$-bit permutation by $\Pi$ and the original channel with memory acting on $N$ bits by $W_N(Y_1^N|X_1^N)$, the effective channel produced by interleaving is

$$W_I(Y_1^N|X_1^N) = \frac{1}{N!}\sum_\Pi W_N(\Pi(Y_1^N)|\Pi(X_1^N)). \tag{2}$$

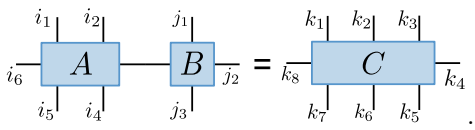
This is an exchangeable sequence of channels and de Finetti's theorem [8] shows that as $N \to \infty$, the channel experienced by any finite bit sequence approaches an i.i.d. channel.

The main drawback of interleaving is the incurred latency. Moreover, interleaving degrades the channel by throwing away potentially useful information. In other words, this technique does not take advantage of the existing structure in the channel, so the resulting performances are typically suboptimal.

### C. Tensor Networks

Tensor networks are a computational and conceptual tool developed in the context of quantum many-body physics [5], [9]. Similar to graphical models used in coding theory, e.g. factor graph, they are used as compact representations of correlated distributions involving a large number of random variables. Recently, computational problems in coding theory have been recast in the tensor network formalism [10], a connection which has found several applications [4], [11], [12], [13].

For the current purposes, rank-$r$ tensor is a real-valued array $A_{i_1 i_2 \ldots i_r}$ with $r$ indices, where each discrete index $i_k$ has a finite range $|i_k| = \chi_k$ called bond dimension. Graphically, we can represent this object by a node with $r$ open edges attach to it, where edge $k$ represents index $i_k$. Given two such tensors (not necessarily of equal rank) $A_{i_1 i_2 \ldots i_r}$ and $B_{j_1 j_2 \ldots j_q}$, a tensor contraction is represented graphically by joining one edge from $A$ with one edge from $B$, and is only allowed if these edges have the same bond dimension. For instance, if $|i_3| = |j_4| = \chi$, then



. (3)

Algebraically, this tensor contraction corresponds to identifying the joined indices and summing over its value, e.g.,

$$\sum_{s=1}^{\chi} A_{i_1 i_2 s \ldots i_r} B_{j_1 j_2 j_3 s \ldots j_q} = C_{k_1 \ldots k_{r+q-2}}, \tag{4}$$

and results in a tensor $C$ of rank $r+q-2$. A tensor network is a diagram containing multiple tensors joined by many edges, and it generalizes the concept of matrix multiplication in a very natural way.

Storing the entries of a tensor network requires an array of size equal to the product of the bond dimension of its non-contracted bonds, so it scales exponentially with the number of non-contracted bonds. For instance, if all bonds of tensors $A$ and $B$ above have dimension $\chi$, then $A \in \mathbb{R}^{\chi^6}$, $B \in \mathbb{R}^{\chi^4}$, and $C \in \mathbb{R}^{\chi^8}$. Thus, only tensor networks with a few non-contracted bonds are of computational interest. Given such a tensor network with a small number of non-contracted bonds, the computational task of evaluating the corresponding array – i.e., summing over all the contracted bonds – is in general computationally hard (#P-hard [14]). This is because the tensor networks obtained at intermediate steps of the contraction will generally have many open bonds.

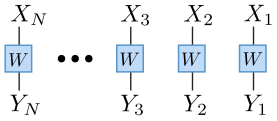
Despite this general hardness, some tensor networks can be efficiently evaluated. It is the case of networks with the geometry of chains or more generally trees [9], which can be evaluated using standard dynamical programming methods. This is a simple extension of a well known fact in coding theory, that belief propagation is exact and efficient on loop-free graphs (trees). More generally, the complexity of evaluating a tensor network scales exponentially with the network's tree-width [14] [15].

As we will explain in the next sections (see ref. [10]), some coding problems are naturally formulated in terms of tensor networks and this connection facilitates the conception of new coding schemes as well as the implementation of certain decoding algorithms. In particular, previous work connecting tensor networks to polar codes with i.i.d. channels [4] naturally extends to channels with memory.

## II. Decoding Algorithm

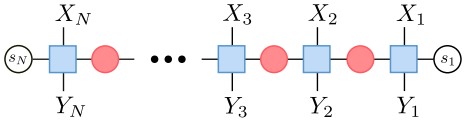
### A. Channel model as a tensor network

A binary discrete memoryless channel $W$ is completely described by a stochastic matrix $W(Y|X)$ where $X$ denotes the input and $Y$ the output. This is a rank-2 tensor with one bond representing the input $X$ and one bond representing the output $Y$, with bond dimensions equal to the corresponding alphabet sizes. The i.i.d. channel acting on an $N$-symbol sequence $X_1^N$ can be represented graphically as a rank-$2N$ tensor as follows

$$W_N(Y_1^N|X_1^N) =$$


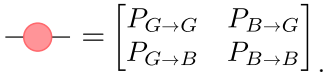
. (5)

The tensor network representing a channel with a $d$-state memory acting on $N$ symbols is a chain which combines rank-2 and rank-4 tensors (and rank-1 tensors on its boundary) given by

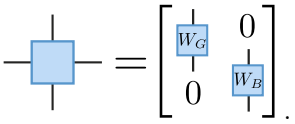
$$W_N(Y_1^N|X_1^N) =$$


. (6)

This structure is quite familiar in condensed matter physics where it goes under the name *matrix product operator* (MPO) [16]. The horizontal bonds all have dimension $d$, the size of the channel's memory, while the top vertical bonds have the dimension of the input alphabet $X$ and lower vertical bonds have the dimension of the output alphabet $Y$. The rank-2 tensors are the stochastic matrices $P_{C \to C'}$ defining the channel with memory. For the case of the Gilbert-Elliott channel, this would be


$= \begin{bmatrix} P_{G\to G} & P_{B\to G} \\ P_{G\to B} & P_{B\to B} \end{bmatrix}$. (7)

A rank-4 tensor can be thought of as a matrix of matrices, i.e. for fixed values of $i$ and $j$, the object $A_{i,j,k,l}$ is a rank-2 tensor, a.k.a. a matrix. From this perspective, for fixed values $C$ and $C'$ of its horizontal bonds, the rank-4 tensors appearing in Eq. 6 is the matrix $\delta_{C,C'} W_C$ where $\delta$ denotes the Kronecker delta. In the specific case of the Gilbert-Elliott channel, this would be



. (8)

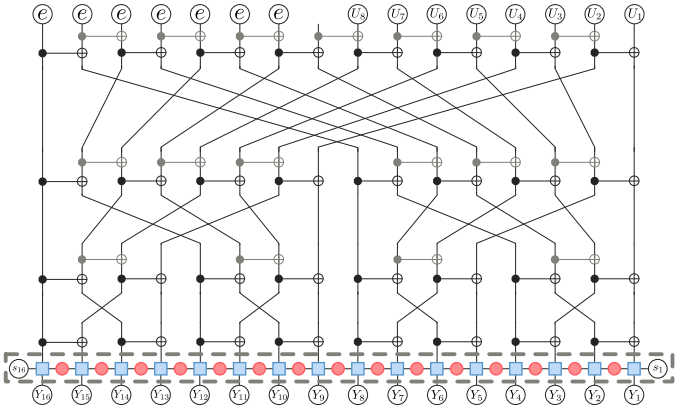


Fig. 1. The likelihood $L(U_1^i|Y_1^N)$ expressed as a tensor network. The encoding circuit corresponds to a convolutional polar code, or to a regular polar code if the shaded gates are removed. The channel is one with memory described at Eq. 6, but could be replaced by an i.i.d. channel by replacing the tensor network in the dotted region by Eq. 5.

Finally, the rank-1 tensor $s_1$ is the stationary distribution $P(C)$ while the rank-1 tensor $s_N$ is the vector containing all 1's. These respectively encode the fact that when the first bit is transmitted, the channel's memory is sampled from its uniform distribution and that at the end of the transmission, the channel's memory can take any value.

### B. Decoding as tensor network contraction

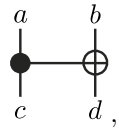
A code can be defined by a reversible encoding matrix $G$ which maps some input sequence $U_1^N$ to some output sequence $X_1^N = GU_1^N$, with some set $\mathcal{F}$ of input bits $U^{\mathcal{F}}$ frozen to the value 0. The non-frozen bits $U^{\mathcal{F}^c}$ ($c$ denotes complement) carry the raw data. The channel maps $X_1^N$ to $Y_1^N$, and maximum likelihood decoding given received message $Y_1^N$ consists in optimizing $W_N(Y_1^N|GU_1^N)$ over $U^{\mathcal{F}^c}$, and where the bits $U^{\mathcal{F}}$ are fixed to 0.

Successive cancellation decoding is a simplification of this procedure where the global optimization over $U^{\mathcal{F}^c}$ is broken into a sequence of optimizations over individual bits of $U^{\mathcal{F}^c}$. Moreover, each optimization is conditioned on the value of previously decoded bits, and freezes to 0 only the bits of $U^{\mathcal{F}^c}$ with a lower index value. In other words, for some $i \in \mathcal{F}^c$, the decoded value is

$$\underset{U_i}{\operatorname{argmax}} \sum_{U_{i+1}^N} W_N(Y_1^N|GU_1^N), \tag{9}$$

where the bits $U_j$, $j < i$ are fixed either to 0 if $j \in \mathcal{F}$ or to their previously decoded value if $j \in \mathcal{F}^c$.

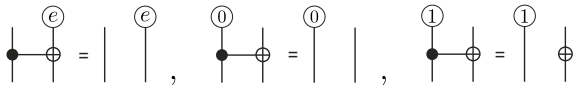
The likelihood $L(U_1^i|Y_1^N) = \sum_{U_{i+1}^N} W_N(Y_1^N|GU_1^N)$ being optimized at Eq. 9 can be represented as the tensor network of Fig. 1. In this representation, a single bit $U_i = 0$ is represented by the rank-one tensor $\binom{1}{0}$ while $U_i = 1$ is represented by $\binom{0}{1}$. The rank-one tensor $e$ is $\binom{1}{1}$ and represents the action of summing over all possible value of the corresponding bit. Finally, while the CNOT is usually viewed as an $\mathbb{F}_2^2$-linear transformation, it is here represented by a rank-4 tensor. By grouping the 4 indices of the tensor into two input indices $a$ and $b$ and two output indices $c$ and $d$ as follows




, (10)

the action of the CNOT on a probability distribution $P(a,b) \in \mathbb{R}^4$ is CNOT: $P(a,b) \to Q(c,d) = P(a, a \oplus b)$.

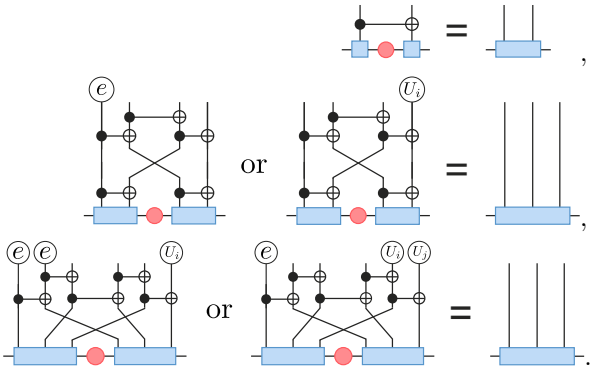
Sequential cancellation decoding thus consists in evaluating this tensor network. A priori, this tensor network has a tree-width that scales with $N$, so it cannot be contracted efficiently. However, there exists algebraic relations between the elementary tensors which leads to important simplifications. These identities are



, (11)

and should have a clear intuitive meaning. Applying these algebraic identities to the tensor network of Fig. 1 produces a tensor network of constant tree-width, so it can be efficiently evaluated.

The evaluation of the tensor network uses graphical identities, which simply perform some summations over connected bonds:




, (12)

, (13)

. (14)

This notation encodes only the type of tensors being involved in a calculation, without specifying the entries of each tensors. This suffices to prove that all stages of the computation involve constant-bounded degree tensors. Therefore it has constant complexity. A recursive use of these identities leads to the evaluation of the entire tensor network. Specifically, successive cancellation decoding of the polar code under correlated noise makes use of the identity Eq. 12, while the convolutional polar code makes use of Eq. 13 and Eq. 14.

Note that the only difference incurred by the correlations in the channel on these contraction identities is the presence of horizontal bonds. The presence of correlation thus increases the rank of the tensors involved in these identities by two. In other words, the scalar entries of the tensor network representing an uncorrelated channel become matrix entries (rank-2 tensors) in the presence of correlations. So each scalar multiplication involved in the successive cancellation decoder

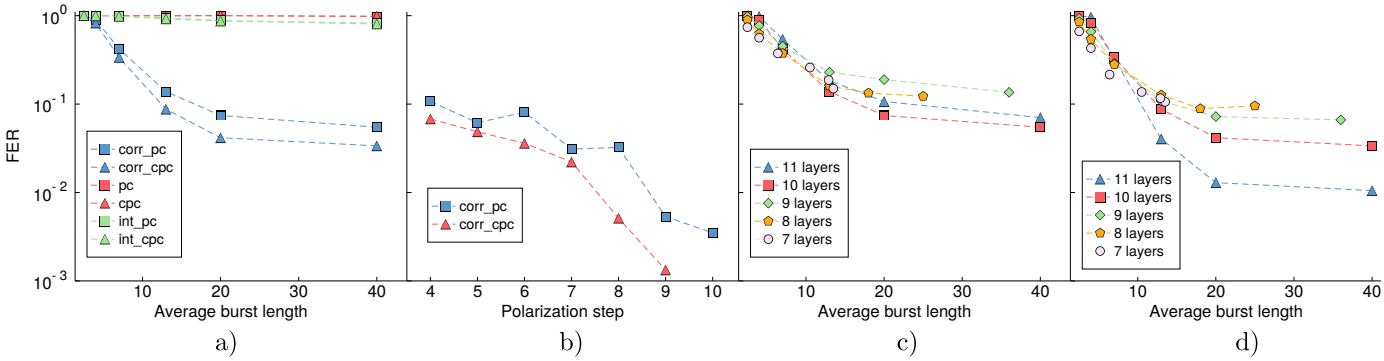



Fig. 2. Frame error rate (FER) as a function of various codes, decoders and Gilbert channel parameters. a) Varying the channel's average burst length for fixed $h = 0.9$, $\rho = 5$, and average error $P(B)h = 0.15$, the exact parameters can be found in Table I. Squares show polar code (pc) while triangles show convolutional polar code (cpc), both with 10 polarization steps and rate $\frac{1}{2}$. The regular decoder (red) results in high FER, using an interleaver (green) to mitigate correlations offers little improvement, while using the correlated decoder (blue) leads to a substantial reduction of the FER. b) Correlated decoder used on polar code (pc, blue squares) and convolutional polar codes (cpc, red triangles) on a channel with parameters $h = 0.9$, $P_{B\to G} = 0.05$ and $P_{G\to B} = 0.01$ with a code rate of $\frac{1}{3}$ as a function of the number of polarization steps. c) & d) As in a) for various polarization steps and using the correlated decoder for c) polar code and d) convolutional polar code.

in absence of correlations is replaced by a $d \times d$ matrix multiplication in the presence of correlations, hence a $d^3$ complexity overhead.

## III. Numerical Results

The encoding circuit $G$ of Fig. 1 does not entirely specify the code: one must in addition specify the frozen-bit locations $\mathcal{F}$. In our simulations, we used the following heuristic code construction. For each position $i$, we evaluate the tensor network of Fig. 1 with all bits $U_1^{i-1}$ fixed to 0, bit $U_i$ fixed to 1, and all positions to the left of $i$ fixed to the rank-1 tensor $e$. This tensor evaluates to a value $E(i)$, and we freeze the bits with the largest value of $E(i)$. The quantity $E(i)$ relates to the probability of an undetected error at position $i$, and serves as a good proxy for channel selection.

We simulated three types of decoding algorithms and corresponding frozen bit sets. In the first set of simulations, we use the standard decoding algorithm and code construction. In that case we use the average channel Eq. 1 in our code construction and decoding algorithm, but use the actual correlated channel to generate errors. The second set of simulations uses the same average channel Eq. 1 for code construction and decoding, but generates errors according to the interleaved procedures. We will denote these simulations with the prefix *int*. The third set of simulations exploit the full correlated structure of the channel both in the code design and decoding algorithm, as described in the previous section. We use the prefix *corr* to denote these simulations. Finally, all three simulation sets include polar codes (*pc*) and convolutional polar codes (*cpc*).

Our numerical simulations used the Gilbert-Elliott channel with $h_G = 0$ so that state $G$ is completely noiseless, and we denote $h_B = h$. This special case is usually referred to as the Gilbert channel. The error burst length $\ell_B$ corresponds to the length of a consecutive stay in state $B$. It has a geometric distribution with average $\langle \ell_B \rangle = 1/P_{B\to G}$. Another channel characteristic is the *good-to-bad ratio*

$$\rho = \frac{P(G)}{P(B)} = \frac{P_{B\to G}}{P_{G\to B}}.$$

The limits $\rho \to \infty$ and $\rho \to 0$ correspond respectively to a noiseless channel and BSC($h$).

| $\langle \ell_B \rangle$ | $P_{B\to G}$ | $P_{G\to B}$ |
|---|---|---|
| 2.5 | 0.4 | 0.08 |
| 4 | 0.25 | 0.05 |
| 7 | 0.145 | 0.029 |
| 13 | 0.075 | 0.015 |
| 20 | 0.05 | 0.01 |
| 40 | 0.025 | 0.005 |

TABLE I
Channel parameters for the simulations of Fig. 2a), c) & d).

Our main results are presented at Fig. 2, showing the frame error rate (FER) for various codes, decoding and channel parameters. We observe on Fig. 2a) that, for both polar and convolutional polar codes, the use of an interleaver to mitigate burst errors offers only a mild improvement of performance, but that using a decoder that exploit the correlations offers a significant improvement. This is true for a variety of channel parameters, and as expected the improvement is reduced when the burst length becomes close to 1, in which case we recover an i.i.d. channel.

Figure 2b) presents a direct comparison of polar and convolutional polar codes under correlated decoders. In both cases, we observe a suppression of the FER as a function of the number of polarization steps, but the convolutional polar code achieves a larger suppression rate. This is consistent with the findings of [4] in the context of i.i.d. channels and is suggestive that convolutional polar codes may achieve a larger error exponent on a wide variety of channels.

Finally, Fig. 2c) and d) present the performance of the c) polar code and d) convolutional polar code with varying

number of polarization steps as a function of the channel's average burst length. Again, the performance of the convolutional polar code are systematically better than those of the polar code. In particular, for the convolutional polar code, we observe a threshold behavior: there is a critical average burst length $\langle \ell_B \rangle^* \approx 10$ beyond which the FER decreases with the number of polarization steps. This threshold behavior is not observed for the polar code. This observation is suggestive of two possible hypotheses: 1) In this parameter regime, the error probability will asymptotically vanish for the convolutional polar code but not for the polar code, i.e. polar codes cannot polarize these channels while convolutional polar codes can. 2) The finite size effects are much more prominent in the polar code than in the convolutional polar code. Both of these hypotheses are interesting and warrant further investigation of convolutional polar codes.

## IV. Conclusion

Tensor network is a natural language in which to formulate the decoding problem. In this language, the efficiency of successive cancellation decoding of polar codes follows immediately from the tree structure of the corresponding tensor network. The generalization to convolutional polar codes is also efficient because the corresponding tensor network has constant tree-width. What we have shown here is that decoding a channel with memory increases the tree-width of the resulting tensor network only by a constant amount, so the decoding complexity is only affected by a constant factor. The simplicity of these results illustrates the use of tensor network in coding theory.

Our numerical simulations confirm that a substantial improvement (up to 12dB increase noise suppression in our simulations) is obtained when taking channel correlations into account during the decoding process and code construction. Moreover, we found that convolutional polar codes outperform regular polar codes (up to 5dB increase noise suppression in our simulations) on a wide variety of channels with memory, extending the findings of [4] beyond the i.i.d. case. In particular, our simulation suggest the existence of a range of channel parameters where polar codes fail at polarizing while convolutional polar codes succeed, an observation which warrants further investigation into convolutional polar codes.

## Acknowledgment

This work was supported by Canada's NSERC. Computations were provided by Compute Canada and Calcul Québec.

## References

[1] **E. Arikan** (2009). *Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels*. IEEE Transactions on Information Theory, 55(7), 3051–3073.

[2] **E. Sasoglu and I. Tal** (2016). *Polar coding for processes with memory* arXiv:1602.01870

[3] **R. Wang, J. Honda, H. Yamamoto, R. Liu, and Y. Hou** (2015). *Construction of polar codes for channels with memory*. in Proc. of the IEEE Inform. Theory Workshop, Jeju, South Korea, pp. 187–191.

[4] **A. J. Ferris, C. Hirche, and D. Poulin** (2017). *Convolutional Polar Codes*, 1–25. arXiv:1704.00715.

[5] **G. Evenbly and G. Vidal** (2014). *A class of highly entangled many-body states that can be efficiently simulated*. Phys. Rev. Lett. vol. 112, p. 240502.

[6] **A. J. Ferris and D. Poulin** (2014). *Branching MERA codes: A natural extension of classical and quantum polar codes*. IEEE International Symposium on Information Theory, pp. 1081–1085.

[7] **E. N. Gilbert** (1960). *Capacity of a Burst-Noise Channel*. Bell System Technical Journal, 39(5), 1253–1265.

[8] **P. Diaconis and D. Freedman** (1980). *Finite exchangeable sequences*. Annals of Probability. 8(4), 745–764.

[9] **Y.-Y. Shi, L.-M. Duan and G. Vidal** (2016). *Classical simulation of quantum many-body systems with a tree tensor network*. Physical Review A 74, 022320.

[10] **A. J. Ferris and D. Poulin** (2014). *Tensor networks and quantum error correction*. Physical Review Letters, vol. 113, no. 3, p. 030501, arXiv:1312.4578.

[11] **A. S. Darmawan and D. Poulin** (2017). *Tensor network simulations of the surface code under realistic noise*. Phys. Rev. Lett. 119, 040502.

[12] **A. S. Darmawan and D. Poulin** (2018). *An efficient general decoding algorithm for the surface code*. arXiv:1801.01879.

[13] **F. Pastawski, B. Yoshida, D. Harlow, and J. Preskill** (2015) *Holographic quantum error-correcting codes: Toy models for the bulk/boundary correspondence*. JHEP 06 149, arXiv:1503.06237.

[14] **I. Arad and Z. Landau** (2010). *Quantum Computation and the Evaluation of Tensor Networks*. SIAM Journal on Computing, vol. 39, no. 7, pp. 3089–3121.

[15] **I. L. Markov and Y. Shi** (2008) *Simulating quantum computation by contracting tensor networks*. SIAM Journal on Computing, 38(3):963–981.

[16] **B. Pirvu, V. Murg, J. I. Cirac, and F. Verstraete** (2010). *Matrix product operator representations*. New Journal of Physics 12, 025012, arXiv:0804.3976.