A Universal Systematic Method to Generate Error Patterns on Memoryless Channels

Marwan Jalaleddine, Jiajie Li, Syed Mohsin Abbas, Warren J. Gross

Abstract

The high computational cost of approaching the performance of Maximum-likelihood (ML) decoding has limited its practical use for decades. Because the complexity grows exponentially with the message length, researchers have spent years developing algorithms like Ordered Statistics Decoding (OSD), Partial Ordered Statistics Decoding (POSD) and Guessing Random Additive Noise decoding (GRAND) which try to approach ML performance. OSD, POSD and GRAND work by trying to guess the error patterns affecting the received signals. However, there does not exist a systematic method to extend the error pattern guesses to novel channels. This work introduces a systematic method that uses the Probability Density Function (PDF) of a memoryless channel to generate a set of error patterns that can be applied on any future received signal on this channel. Simulation results show that our proposed method applied on GRAND, OSD and POSD generally matches or outperforms current pre-generated error patterns on additive white Gaussian noise (AWGN) channel, mixture of Gaussian distribution channels, Rayleigh fading channel with perfect knowledge of Channel State Information (CSI) and Rayleigh fading channel with no perfect knowledge of Channel State Information (NCSI).

I. Introduction

In 1922, the statistical principle of Maximum-likelihood (ML) estimation established a benchmark to the optimality of the estimate [1]. This notion became an important concept in coding theory, which emerged in 1948 [2], as the ML estimate provides the optimal performance target for any decoder. ML decoding identifies the most likely codeword given a received channel signal; however, ML decoding is an NP complete problem [3] as it require s evaluating 2^k candidates for binary codes with k being the message length. The infeasible computational complexity of ML decoding has driven the search for practical approximate ML decoding algorithms.

In the 1950s, one of the first attempts at approximate ML decoding emerged through sequential decoding [4]. The sequential decoder was the first practical algorithm capable of approaching ML decoding performance on low-rate convolutional codes. In the 1960s, a simplified sequential decoding algorithm with low memory requirements [5] refined the original approach. Due to the sequential decoder's effectiveness in the low signal-to-noise conditions, NASA deployed the sequential decoder on the Pioneer 9 mission and used it as the standard coding system for deep space [6], [7]. Sequential decoders face a fundamental limitation known as the cutoff rate. Exceeding this rate causes the number of states to grow exponentially, leading to high computational and time complexities. In a separate line of work, in the 1960s, an efficient exact ML Viterbi decoder for convolutional codes [8] emerged that had Ohigher memory and computational requirements than the previously proposed sequential decoders. Unlike the previously proposed sequential decoders, the Viterbi decoder was amenable for hardware implementations using shift registers as it did not require complex control logic to lead the search for the codeword. Additionally, the Viterbi decoder provided deterministic latency in contrast to the previously proposed sequential decoders' unpredictable latency.

For block codes, the exhaustive ML search remained intractable since using the ML Viterbi decoder on the equivalent trellis results in high computational complexity [9]. Among the earliest near-ML decoding approaches on linear block codes, a scheme proposed in 1987 [10]–[13] generates candidate codewords by applying error patterns to the message bits in order of increasing likelihood. This method effectively approximates the ML search by examining error patterns starting from the most probable ones. However, enumerating error patterns in this order is not well suited to hardware implementation and is difficult to parallelize, limiting the practicality of the approach in modern communication systems.

OSD, introduced in 1995 [14], leverages the most reliable k received bits to generate near-ML codewords. OSD first reorders the received bits by decreasing reliability (so that the first k positions correspond to the most reliable bits) and then applies Gaussian elimination (GE) to the permuted generator matrix to obtain a systematic form with these k bits as the information basis. OSD generates candidate codewords by flipping combinations of these most reliable bits according to a predetermined set of error patterns. The analysis shows that OSD can achieve a decoding performance equivalent to ML decoding by querying all error patterns with Hamming weight up to $\lceil d_H/4 - 1 \rceil$, where d_H denotes the minimum distance of the code [14]. Many works have proposed complexity-reduction techniques for OSD, such as the box-and-match method [15], the linear-equation technique [16], and the segmentation discarding technique [17], all of which aim to discard unpromising error patterns. Alternatively, the approach in [18] exploits the structure of Reed Solomon codes to avert the use of GE for Bose-Chaudhuri-Hocquenghem

Marwan Jalaleddine, Jiajie Li and Warren J. Gross are with the Department of Electrical and Computer Engineering at McGill university, Montreal, Quebec, Canada. Their emails are: marwan.jalaleddine@mail.mcgill.ca, jiajie.li@mail.mcgill.ca, warren.gross@mcgill.ca. Syed Mohsin Abbas is with Tampere University, Finland. His email is: mohsin.abbas@tuni.fi

(BCH) codes [18] at the cost of limiting OSD's use for BCH codes. Although fixed error patterns and error-pattern elimination methods reduce OSD's complexity, the required GE for all codes except BCH codes impedes efficient hardware implementation [19].

In 2001, a ML method [20] that directly enumerates likely noise (error) sequences affecting the codeword emerged. Instead of guessing codewords directly, this method generates error patterns in order of decreasing likelihood and subtracts each pattern from the received signal. The first valid codeword produced by this method corresponds to the ML codeword, albeit at the cost of a hardware-inefficient error-pattern scheduling. Since 2019, the underlying technique for guessing error patterns has been referred to as GRAND [21]. The ML error pattern generation technique, which produces the error patterns in the same order to those produced by [20], is referred to as Soft Guessing Random Additive Noise decoding (SGRAND) [22]. Numerous hardware-efficient error-pattern ordering strategies have been developed for GRAND, most notably the logistic-weight error patterns used in ORBGRAND in 2021 [23], [24]. These error patterns were the first to approximate the ML decoding behavior of GRAND with only minor performance degradation. Logistic weight error patterns and other hardware-friendly error patterns [25], [26], have driven the development of efficient hardware implementations for GRAND [25]–[29]. While such patterns mitigate implementation complexity, they do not address a core algorithmic limitation of GRAND: it typically must guess the entire error vector affecting the codeword, which becomes impractical for codes with large error-correcting capability.

To decode codes of lower rates without the need of GE, Partial Ordered Statistics Decoding (POSD) [30]–[32], initially proposed in 2009 can be used as a simplification of the ML approach proposed in [10]–[13]. The original POSD [30], [31] employed fixed error patterns inspired from the OSD framework, but the choice of error patterns led to relatively poor decoding performance as the patterns used did not map correctly the error patterns seen on the channel. Subsequently in 2023, improved error-patterns inspired by [24], [33] achieved enhanced decoding performance with POSD while retaining the hardware-friendly, predetermined error patterns approach. In a separate line of work, Guessing Codeword Decoding (GCD) [34], introduced in 2024, is effectively a rediscovery of the original 1986 method [10]–[13] upon which POSD is based on. In fact, a summary of this algorithm proposed in 1986 can be found in the earlier literature (see, e.g., p. 361-362 of [12]) in english and the equivalency of the distance metrics is established in Appendix A in this work.

While GRAND, POSD, and OSD algorithms can be extended to novel channel models, a systematic framework for such adaptations, without the use of Monte Carlo simulations or piecewise-linear approximation techniques, remains elusive. Although Monte Carlo methods effectively characterize error patterns in low Signal-to-Noise Ratio (SNR) regimes, their utility diminishes at higher SNRs. In these regions, the scarcity of observed test error patterns prevents a statistically significant ranking of test error patterns. Moreover, the piecewise-linear approximation technique proposed in ORBGRAND [24] can cause a large degradation in performance compared to the ML error patterns when the channel's log-likelihood ratio distribution is non-smooth.

In this work, we introduce a novel method to generate channel adaptive error patterns for OSD, POSD and GRAND on memoryless channels. We first determine the Probability Density Function (PDF) of the sorted reliabilities of the received channel signal, the bits GRAND, OSD and POSD operate on. Then, we generate error patterns in increasing expected value of Weighted Hamming Distance (WHD) based on the fact that the codeword that minimizes the WHD is the ML codeword. These generated error patterns can be loaded into the decoder and the same error patterns can be used on any received channel signal. We validate our proposed method on AWGN, Rayleigh Fading with no knowledge of CSI, Rayleigh Fading with perfect knowledge of CSI and two mixture of Gaussian channels with novel additive noise profiles. We consistently show that our proposed predetermined error patterns result in a decoding performance equivalent or better than the previous state-of-the-art predetermined error patterns. We also note that our proposed error patterns can be used to improve the error pattern approximations [24] through generating a set of error patterns to be saved in memory for the regions that cannot be easily linearized.

The remainder of this paper is organized as follows. Section II establishes the preliminaries and the notations used in this work. Section III details existing code-agnostic decoding algorithms, specifically GRAND, OSD, and POSD, and explores various existing error pattern ordering techniques. In Section IV, we propose a theoretical framework that utilizes both Log-Likelihood Ratio (LLR) and received signal distributions to generate optimized error patterns. The subsequent sections evaluate this framework across diverse channel environments: Section V addresses the Additive White Gaussian Noise (AWGN) channel, providing a performance analysis of Test Error Patterns (TEPs); Section VI extends this to Mixture of White Gaussian channels; and Section VII examines uncorrelated fast-fading scenarios. Finally, Section VIII concludes the paper with a summary of our findings.

II. Preliminaries and Notations

A. Notations

Matrices and vectors are denoted by a bold letter symbol ($\mathbf{M}$ or $\mathbf{v}$). The transpose operator is represented by $^\top$. The number of k-combinations from a given set of n elements is noted by $\binom{n}{k}$. $\mathbf{1}_z$ is the indicator vector where all locations except the z^{th} are 0 and the z^{th} is 1. All the indices start at 1. The i'th element of vector $\mathbf{v}$ is represented as $\mathbf{v}[i]$. The subvector composed of elements i to j from vector $\mathbf{v}$ is represented as $\mathbf{v}[i:j]$. The symbols $\implies$, $\not\Longrightarrow$ denote *implies* and *does not imply*

respectively. $\oplus$ denotes the XOR binary operation. For this work, the Galois field with 2 elements is noted as $\mathbb{F}_2$. Furthermore, we restrict ourselves to (n, k) linear block codes, where n is the code length and k is the message length. Throughout this analysis and simulations, we assume Binary Phase Shift Keying (BPSK) modulation and that bit 0 and bit 1 are equiprobable.

B. Definitions

Definition 1 (Linear Block Code): A linear block code is a linear mapping $\mathbb{F}_2^k \rightarrow \mathbb{F}_2^n$. To characterise any linear block code, there exists a $k \times n$ matrix $\mathbf{G}$ called the generator matrix. The generator matrix is in standard form if it is represented as $\mathbf{G} = [I^k|P]$ where $\mathbf{I}^k$ is the identity matrix of size $k \times k$ and $\mathbf{P}$ is an $k \times n - k$ matrix. We use a notation [n,k] to define the codeword length and length of the used code. A codeword $\mathbf{c}$ of length n belonging to the linear block code (represented by codebook $\mathbb{C}$) can be generated by multiplying the message ($\mathbf{u}$) of length k with the generator matrix:

$$\mathbf{c} = \mathbf{u} \times \mathbf{G} \tag{1}$$

The corresponding systematic form of the parity-check matrix is $\mathbf{H} = [\mathbf{P}^T|I_{n-k}]$. Any codeword belonging to the codebook of generator matrix $\mathbf{G}$ produces a zero syndrome:

$$\mathbf{s} = \mathbf{c}\mathbf{H}^T = \mathbf{0} \quad | \quad \mathbf{c} \in \mathbb{C} \tag{2}$$

Definition 2 (Binary Phase-Shift Keying (BPSK)): BPSK is a linear modulation scheme that maps each binary symbol $b \in \{0, 1\}$ to a real-valued signal point $x \in \{+1, -1\}$ via the mapping $x = 1 - 2b$.

Definition 3 (Log Likelihood Ratio (LLR)): The LLR (L) of an instance of the received signal of the received channel signal is the measure on our confidence that the received coded bit c is 0 as opposed to 1 :

$$L = \mathcal{L}(y) = \ln\left(\frac{P(y \mid c = 0)}{P(y \mid c = 1)}\right). \tag{3}$$

A positive LLR, is demodulated to a 0 bit and a negative LLR is demodulated to a 1 bit. The absolute value of the LLR $|L|$ represents how confident we are in our demodulated result. The higher the value of $|L|$, the higher the reliability of this received signal. The corresponding vector of LLRs is represented as $\mathbf{L}$.

Definition 4: The hard demodulation function (θ) thresholds the received LLRs :

$$\theta(L) = 0 \quad ; \quad L \geq 0. \tag{4}$$

$$\theta(L) = 1 \quad ; \quad L < 0. \tag{5}$$

Definition 5 (Weighted Hamming Distance): The weighted Hamming distance (also known as the ellipsoidal weight, the correlation discrepancy, soft analog weight) represents the soft distance of the current codeword from the received signal:

$$w_H(\mathbf{c}, \mathbf{L}) = \sum_{i=1}^{N} (\mathbf{c}[i] \oplus \theta(\mathbf{L}[i])) \times |\mathbf{L}[i]|. \tag{6}$$

Noting that the $(\mathbf{c}[i] \oplus \theta(\mathbf{L}[i]))$ is equivalent to the hard error pattern ($\mathbf{e}[i]$) that was applied by the decoder onto the hard demodulated signals, we can represent the weighted Hamming distance in terms of the error pattern:

$$w'_H(\mathbf{e}, \mathbf{L}) = \sum_{i=1}^{N} \mathbf{e}[i] \times |\mathbf{L}[i]|. \tag{7}$$

If all codewords have equal probability of being transmitted, the codeword that minimizes the weighted Hamming distance in a memoryless channel is the maximum likelihood codeword [35]. In case the soft channel information are not known, the weighted Hamming distance becomes the Hamming distance which can be calculated as:

$$hw'(\mathbf{e}) = \sum_{i=1}^{N} \mathbf{e}[i]. \tag{8}$$

Hence, the maximum likelihood criteria for a hard valued channel is the codeword that minimizes the number of bit-flips from the received signal.

TABLE I
AN EXAMPLE OF THE EXECUTION OF WEIGHT MINIMIZATION PROCEDURE ASSUMING WEIGHTS OF INDEX $1, 2$ AND 3 ARE $0.3, 0.4$ AND 0.5 RESPECTIVELY, AND THE WEIGHT FUNCTION IS ADDITIVE.

t	Z_c	$w(Z_c)$	j^*	S	V	A
1	{}	0.0	0	{(0.3,{1},1),(0.4,{2},1,),(0.5,{3},1,)}	{1,2,3}	{({},0)}
2	{1}	0.3	1	$\{(0.3, \{1\}, 1), (0.4, \{2\}, 1), (0.5, \{3\}, 1)\}$	{1,2,3}	{({},0), ({1},0.3)}
3	{2}	0.4	2	$\{(0.3, \{1\}, 1,), (0.7, \{1, 2\}, 2), (0.5, \{3\}, 0)\}$	{2,3}	{({},0), ({1},0.3), ({2},0.4)}
4	{3}	0.5	3	$\{(0.3, \{1\}, 1), (0.7, \{1, 2\}, 2), (0.8, \{1, 3\}, 2)\}$	{2,3}	{({},0), ({1},0.3), ({2},0.4), ({3},0.5)}
5	{1,2}	0.7	2	$\{(0.3, \{1\}, 1), (0.7, \{1, 2\}, 2), (0.8, \{1, 3\}, 2)\}$	{3}	{({},0), ({1},0.3), ({2},0.4), ({3},0.5),({1,2},0.7)}
6	{1,3}	0.8	3	$\{(0.3, \{1\}, 1), (0.7, \{1, 2\}, 2), (0.9, \{2, 3\}, 3)\}$	{3}	{({},0), ({1},0.3), ({2},0.4), ({3},0.5),({1,2},0.7), $(\{1, 3\}, 0.8)$}
7	{2,3}	0.9	3	$\{(0.3, \{1\}, 1), (0.7, \{1, 2\}, 2), (1.2, \{1, 2, 3\}, 5)\}$	{3}	{({},0), ({1},0.3), ({2},0.4), ({3},0.5), ({1,2},0.7), $(\{1, 3\}, 0.8), (\{2, 3\}, 0.9)$}
8	{1,2,3}	1.2	3	$\{(0.3, \{1\}, 1), (0.7, \{1, 2\}, 2), (1.2, \{1, 2, 3\}, 5)\}$	{}	{({},0), ({1},0.3), ({2},0.4), ({3},0.5),({1,2},0.7), $(\{1, 3\}, 0.8), (\{2, 3\}, 0.9), (\{1, 2, 3\}, 1.2)$}

C. Computing Lists of Binary Vectors That Optimize a Given Weight Function

The authors in [20] presented an efficient algorithm for generating binary vectors (patterns) in strictly increasing order of a given weight function w, an improvement of an original algorithm proposed by Battail [11]. Let z be the number of bit positions under consideration. Each pattern $\mathbf{e}$ is a binary vector of length N, and the set Z holds the set of the positions of 1's in $\mathbf{e}$. The weight function $w'(\mathbf{e}) = w(Z)$ satisfies the monotonicity property or all positions $j \notin Z, j \notin Z'$:

$$w(Z) \leq w(Z \cup \{j\}) \quad , \tag{9}$$

$$w(Z) \leq w(Z') \implies w(Z \cup \{j\}) \leq w(Z' \cup \{j\}), \tag{10}$$

where $\{j\}$ corresponds to a set with element j (1 on position j of the $\mathbf{e}$ vector) and $Z \cup \{j\}$ describes a set Z with an additional element j.

The algorithm begins by allocating a list of arrays ($S[1 : z]$). Each $S[j]$ contains exactly one candidate pattern whose *last* error occurs at position j. Each list element is a tuple (w, Z, p) where w is the weight $w(Z)$ of the pattern, Z is the set of positions where of 1's and p is pointer to the parent pattern in history list A. The history list (A) is a sequential record of all patterns that have been generated and processed, stored as (w, Z) pairs. Additionally, the validity set determines whether all the binary vectors originating from that vector have been generated.

The algorithm allocates only one new pattern Z_s each time a new Z_c pattern is chosen. If a successor cannot be allocated, the pattern is considered not valid and no further patterns are derived from it.

The algorithm guarantees that the patterns are generated in strictly increasing order of weight ($w(Z_1) \leq w(Z_2) \leq \cdots \leq w(Z_M)$). Additionally, this algorithm guarantees completeness as all patterns can be generated exactly once. TABLE I shows an example of the operation of the algorithm on a set of 3 elements. We note that for sequential error generation, all the elements of A with an index less than the minimum p_{next} in set S can be removed. Additionally, the algorithm can be optimized to reduce memory use.

III. CODE AGNOSTIC DECODING ALGORITHMS AND ERROR PATTERNS

In this section, we will discuss different code agnostic decoders. Most of the discussed decoders have variants that can achieve close to ML performance. Hence, the decoders aim to :

1) Generate a result that is a codeword belonging to the codebook.
2) Minimize the weighted Hamming distance compared to all the other codewords in the codebook.

The maximum-likelihood version of GRAND [20], [22] traverses its error pattern in a manner that minimizes the weighted Hamming distance, but each of its results is not necessary a codeword. Alternatively, OSD [14] and POSD [30]–[32] generate elements that are codewords in the codebook, but in an order that does not necessarily minimize the weighted Hamming distance. This section will discuss the details of these decoding methods.

A. Guessing Random Additive Noise Decoding

Guessing Random Additive Noise Decoding (GRAND) [20], [24], [36] has emerged as a universal decoding framework, with certain variants capable of achieving maximum-likelihood performance [20], [22]. The algorithm operates by estimating the additive noise vector $\mathbf{e}$ that has corrupted the transmitted codeword. Some variations [20], [22] utilize the soft information, in the form of bit reliability metrics $\mathbf{L}[i]$, to order potential noise sequences in descending order of their probability using an

Algorithm 1: Pattern Generation by Increasing Weight

Input: Weight values $w[1], w[2], \ldots, w[N]$ for each position
Input: Maximum number of patterns M
Output: Sequence of sets of patterns $Z[1], Z[2], \ldots, Z[M]$ with $w(Z[1]) \leq w(Z[2]) \leq \cdots \leq w(Z[M])$

Initialization:
$A[1] = (\{\}, 1)$;
for $j = 1$ **to** k **do**
  $S[j] = (\{j\}, w(\{j\}), 1)$;
  $V = V \cup \{j\}$;

Main Loop:
for $t = 1$ **to** M **do**
  Select minimal weight pattern from the valid set:
  $j^* = \arg\min_{j \in V} S[j].w$;
  $Z_{\mathrm{c}} \leftarrow S[j^*].Z$;
  Update History:
  Append $(Z_{\mathrm{c}}, w(Z_{\mathrm{c}}))$ to A;
  Find Successor:
  $p_{\mathrm{p}} \leftarrow S[j^*].p$;
  $p_{\mathrm{n}} \leftarrow p_{\mathrm{p}} + 1$;
  while $p_n < |A|$ ***and*** *(*$A[p_n].w \geq S[j^*].w$ ***or*** $\max(A[p_n].Z) \geq j^*$*)* **do**
    $p_{\mathrm{n}} \leftarrow p_{\mathrm{n}} + 1$;
  Update List:
  if $p_n < |A|$ **then**
    $Z_{\mathrm{s}} \leftarrow A[p_{\mathrm{n}}].Z \cup \{j^*\}$;
    $S[j^*] \leftarrow (Z_{\mathrm{s}}, w(Z_{\mathrm{s}}), p_{\mathrm{n}})$;
  else
    $V = V - \{j^*\}$;

Algorithm 2: GRAND Algorithm

Input : Received word LLRs $\mathbf{L}$, parity-check matrix $\mathbf{H}$, maximum query count M
Output: Decoded codeword $\widehat{\mathbf{x}}$ or $ABANDON$
while $n_b \leq M$ **do**
  $\mathbf{e} \leftarrow$ NextNoiseSequence $(\phi(\mathbf{L}), n)$;
  $\widehat{\mathbf{x}} = \theta(\mathbf{L}) \oplus \phi_1^{-1}(\mathbf{e})$;
  if $\widehat{\mathbf{x}}\mathbf{H}^T = \mathbf{0}$ **then**
    return $\widehat{\mathbf{x}}$;
  $n_b + +$;

algorithm similar to Algorithm 1 [20]. These ordered noise patterns are sequentially applied to the received word, commencing with the most likely (zero bit-flips) and proceeding to a predetermined maximum number of queries.

To determine whether a candidate vector $\widehat{\mathbf{x}}$ constitutes a valid codeword, GRAND employs the parity-check matrix $\mathbf{H}$. The syndrome $\mathbf{s}$ is computed as:

$$\mathbf{s} = \widehat{\mathbf{x}}\mathbf{H}^T. \tag{11}$$

A syndrome vector $\mathbf{s} = \mathbf{0}$ thus indicates that $\widehat{\mathbf{x}} \in \mathbb{C}$. To avoid the computational complexity associated with computing the optimized weight sequences, using Algorithm 1 on each of the received channel signals [20], some variations of GRAND aim to create channel specific error patterns that only depend on the ordered bit reliabilities based on the LLRs [24] [25] [33].

B. Ordered Statistics Decoding

OSD [14] is a decoder capable of achieving ML decoding performance at the cost of substantial computational complexity, particularly through its use of GE for generator matrix reordering [14]. Multiple architectures have been proposed to mitigate OSD latency through early termination criteria [37] [38] or by constraining the search space of test patterns [17]. Given that the GE complexity scales as $\mathcal{O}(n^3)$, implementing an efficient low-latency universal OSD decoder remains challenging.

The OSD algorithm proceeds as follows. First an initial processing step involves reordering the received codeword according to decreasing reliability magnitudes. This permutation, denoted $\phi_1()$, sorts the absolute values of the log-likelihood ratios

Algorithm 3: OSD-o Algorithm

Input : Received word LLRs $\mathbf{L}$, permuted generator matrix $\mathbf{G}''$, maximum number of queries M
Output: $\widehat{\mathbf{x}}$

$n_b = 0$ // Current bit-flip count
$w_H = \infty$ // Minimum WHD
$\mathbf{u_c} = \theta(\phi_2(\phi_1(\mathbf{L})))[n-k:n]$
while $n_b \leq M$ **do**
  $\mathbf{e_s} \leftarrow$NextNoiseSequence$(\phi_2(\phi_1(\mathbf{L}))[n-k:n], k)$
  $\mathbf{e} = \mathbf{e_s} \times \mathbf{G}''$
  $\mathbf{x_c} = (\mathbf{u_c} \times \mathbf{G}'') \oplus \mathbf{e}$
  if *WHD*$(\mathbf{x_c}, \phi_2(\phi_1(\mathbf{L})) < w_H$ **then**
    $\widehat{\mathbf{x}} = \phi_1^{-1}(\phi_2^{-1}(\mathbf{x_c}))$
    $w_H = \text{WHD}(\mathbf{x_c}, \phi_2(\phi_1(\mathbf{L}))$
  n_b++
return $\widehat{\mathbf{x}}$

Algorithm 4: POSD Algorithm

Input : Received word LLRs $\mathbf{L}$, generator matrix $\mathbf{G}$, maximum number of queries M
Output: $\widehat{\mathbf{x}}$

$n_b = 0$ // Current bit-flip count
$w_H = \infty$ // Minimum WHD
$\mathbf{u_c} = \theta(\phi_1(\mathbf{L}[1:k]))$
while $n_b \leq M$ **do**
  $\mathbf{e_s} \leftarrow$ NextNoiseSequence $(\phi_1(\mathbf{L}[1:k]), k)$
  $\mathbf{e} = \mathbf{e_s} \times \mathbf{G}''$
  $\mathbf{u_c} = \theta(\mathbf{L})[1:k]$
  $\mathbf{x_c} = \mathbf{u_c} \times \mathbf{G} \oplus \phi_1^{-1}(\mathbf{e})$
  if *WHD*$(\mathbf{x_c}, \mathbf{L}) < w_H$ **then**
    $\widehat{\mathbf{x}} = \mathbf{x_c}$
    $w_H = \text{WHD}(\mathbf{x_c}, \mathbf{L})$
return $\widehat{\mathbf{x}}$

(LLRs) $|\mathbf{L}[i]|$ in descending magnitude order. Subsequently, the generator matrix $\mathbf{G}$ undergoes column permutation via ϕ_1 to yield $\mathbf{G}'$. To ensure linear independence among the first k columns of $\mathbf{G}'$, GE is applied to obtain a systematic form $\mathbf{G}''$, generating an additional permutation $\phi_2()$:

$$\mathbf{G}'' = \phi_2(\phi_1(\mathbf{G})) \tag{12}$$

The received vector LLRs are also mapped to a binary vector via hard decision function $\theta()$. The order-o OSD algorithm processes the k most reliable bits according to Algorithm 3.

The algorithm initializes by applying the most probable noise sequence to the k most reliable bits, generating candidate vector $\hat{\mathbf{u}}_c$. This candidate is multiplied by $\mathbf{G}''$ to produce codeword $\mathbf{x_c}$. The codeword minimizing this weighted Hamming distance is selected as the output.

C. Partial Ordered Statistics Decoding

Partial ordered statistics decoding (POSD), detailed in Algorithm 4, is a simplified ordered statistics decoder that foregoes the requirement of GE [30]. Unlike OSD which applies error patterns on the most reliable bits of the received signal (which are mostly error free), POSD applies its error patterns on the first k bits of the received signal and tries to generate a codeword which minimizes the ML distance criterion. Hence, OSD usually needs to only sort the first k bits of the received signal, reducing the size of the sorting circuitry to be employed with it which is usually the bottleneck in GRAND methods [39]. POSD offers a middle ground between GRAND, which under performs with medium rate codes, and OSD, which generally requires the computationally expensive GE for every decoding attempt. The original version of POSD variant [30], [31] did not evaluate the channel noise sequences to determine the error patterns to be applied on the first k bits which resulted in a degraded error correction ability. Later work [32] solved this issue by suggesting enhanced error patterns, inspired from GRAND [24].

We note that the base algorithm of the POSD proposed in 2023 [32] is identical to themethod proposed in 1986 [11]–[13] and the base GCD algorithm [34] proposed in 2024. The POSD variation relies on hardware-friendly error patterns that are

generated based on the ordered statistics of the LLRs; however the 1986 [11]–[13] method and GCD proposed using a error geneneration strategy similar to the one discussed in Section II-C. The need for using this optimized error pattern generation method limits its hardware compatibility.

Additionally due to the similarities between OSD and POSD, with POSD being a simplified version of OSD, most of the complexity reduction techniques and modifications for OSD could be applied to POSD. For example, the box and match method [15], could be applied on POSD. For all the modifications that can be applied to OSD and could be later adapted to POSD, we refer the reader to [40].

D. Test Error Pattern (TEP) Generation

We examine various TEP generation methods that can be used with the aforementioned decoders.

1) Hamming Weight (HW) Order: The HW ordered TEP generatation method generates test error patterns according to increasing Hamming weight, defined as:

$$HW(\mathbf{e}) = \sum_{i=1}^{n} \mathbf{e}[i]. \tag{13}$$

These error patterns were initially proposed to be used with OSD [14]. Interestingly, these error patterns can operate independent of the soft channel information.

2) Logistic Weight (LW) Order: The logistic weight order, initially proposed for the GRAND variant Ordered reliability bits GRAND (ORBGRAND) [24], generates test error patterns according to ascending logistic weight, computed as:

$$LW(\mathbf{e}) = \sum_{i=1}^{n} i \times \mathbf{e}[i]. \tag{14}$$

3) Improved Logistic Weight (ILW) Order: The improved logistic weight order, initially proposed with GRAND [33], prioritizes LW TEPs with lower Hamming weight. The objective function that is minimized is:

$$ILW(\mathbf{e}) = \sum_{i=1}^{\sum(\mathbf{e}[j])} \text{index}(\mathbf{e}[i]) \times i, \tag{15}$$

where index denotes the index of the ordered vector of indices corresponding to nonzero entries in $\mathbf{e}$:

$$\mathbf{im} = \mathbf{ind}[\mathbf{e}[i] == 1]. \tag{16}$$

4) Maximum Likelihood (ML) Patterns: Using Algorithm 1 applied on the currently received LLRs, the error patterns can be produced in increasing weighted Hamming distance [20]. This method has been shown to outperform other pre-generated error patterns with Soft GRAND and achieve ML performance with a large amount of error patterns [22]. In [35], the authors applied this method to generate error patterns for OSD and showed that the Hamming weight error patterns have a similar decoding performance to the ML patterns. This is due to the fact that :

$$\underset{\mathbf{e}[1:k]}{\arg\min}\, w_H(\mathbf{e}[1:k], \mathbf{L}[1:k]) \nRightarrow \underset{\mathbf{e}[1:N]}{\arg\min}\, w_H(\mathbf{e}[1:N], \mathbf{L}[1:N]) \tag{17}$$

In [10]–[13], the authors suggested a the optimized version of Algorithm 1 to be used to generate error patterns on POSD. Similarly, [34] suggests using it on GCD. Regardless of the proven performance gains with GRAND, no hardware architecture has been proposed of GRAND with this ML error pattern generation technique due to its hardware incompatibility resulting from its sequential nature and the dynamic memory allocation.

5) Empirical Error Patterns (LUT): Following the methodology established in [26] for GRAND, TEPs can be generated according to their empirical likelihood of occurrence. After sorting the received channel signal by increasing reliability, Monte-Carlo simulations can be conducted to record the most frequently occurring TEPs. This empirical set, stored in memory, can be utilized with both GRAND [26], OSD and POSD [32]. We refer to these error patterns in this work as lookup table (LUT) error patterns.

IV. Proposed Theoretical Framework, The Ellipsoidal Weight (EW) Order

The probability of occurrence of an error pattern with a certain Hamming weight given the probability of error p is given by the binomial term:

$$P(hw'(\mathbf{e}) = x) = \binom{n}{x} p^{\,x} (1-p)^{n-x}, \tag{18}$$

Considering the example of an AWGN channel with BPSK modulation and with $\frac{E_s}{N_0} = 8dB$, the probability of error can be directly calculated as:

$$p = Q\left(\sqrt{2\,\frac{E_s}{N_0}}\right) = 1.91 \times 10^{-4}. \tag{19}$$

Hence, the chance of encountering an error pattern with $hw'(\mathbf{e}) = 3$ in a codeword of length 128 bits can be calculated as:

$$P(hw'(\mathbf{e}) = 3) = \binom{128}{3} (1.91 \times 10^{-4})^3 (1 - 1.91 \times 10^{-4})^{125} = 2.3 \times 10^{-6}. \tag{20}$$

In other words, among 10^6 error patterns, only approximately 2.3 are expected to have a Hamming weight of 3. Although such error patterns rarely occur, their impact on frame error rate (FER) performance is significant. If error patterns of Hamming weight 3 are not queried, the FER performance of the decoder is bounded from below by 2.3×10^{-6} irrespective of the error correcting capability of the underlying code. This FER floor might be unsuitable for some URLCC applications [41], [42] where extremely low target FERs are required. Moreover, characterizing these rare error patterns through Monte Carlo simulation is inherently challenging precisely because of their scarcity. To overcome the time complexity of generating the error patterns using Monte-Carlo simulations and to have a more robust ordering of the error patterns, we develop a systematic approach that aims to reduce the expected value of the weighted Hamming distance. We discuss two variations, one where the distribution of the LLRs exists in closed form and another where the distribution of the LLRs is difficult to define in closed form:

A. Using the LLR distribution

This analysis assumes the use of BPSK modulation on a continuous-value memoryless channel. Let f_L denote the probability density function of the LLR at the receiver.The operations that GRAND, OSD, POSD, rely on is first determining the reliability of the signal through calculating the absolute value of the LLRs followed by sorting the LLRs in terms of their reliabilities and finally generating error patterns based the order of bit reliabilities (or their values in the ML error patterns). In the following algorithm we will follow step-by-step what happens to the PDF of the individual bits throughout the steps.

1) Finding the reliabilities: Calculating the reliability of a bit is equivalent to calculating the absolute value of the LLRs. This will provide us a measure of how far away we are from the 0 threshold which determines whether the bit is 0 or 1. After applying the absolute value on the LLRs, the PDF defined by the $f_L(l)$ function and Cumulative Density Function (CDF) defined by $F_L(l)$ function of the distribution of the LLR is "folded" providing the following distribution:

$$f_{|\mathbf{L}|}(l) = f_{\mathbf{L}}(l) + f_{\mathbf{L}}(-l), \qquad l \geq 0 \tag{21}$$

$$F_{|\mathbf{L}|}(l) = F_{\mathbf{L}}(l) + F_{\mathbf{L}}(-l), \qquad y \geq 0 \tag{22}$$

2) Sorting the reliabilities: After the reliabilities are determined, the reliabilities are sorted. Assuming sorting happens in ascending order of reliabilities, the CDF of the ordered statistics of the resulting distribution follows a binomial distribution [43]:

$$F_{\phi(|\mathbf{L}|)[i]}(y) = \sum_{i=k}^{N} \binom{N}{i} \left[F_{|\mathbf{L}|}(y)\right]^i \left[1 - F_{|\mathbf{L}|}(y)\right]^{N-i}. \tag{23}$$

The corresponding PDF can be obtained through differentiating the CDF to obtain:

$$f_{\phi(|\mathbf{L}|)[i]}(y) = \frac{N!}{(i-1)!(N-i)!} \left[F_{|\mathbf{L}|}(y)\right]^{i-1} \times \left[1 - F_{|\mathbf{L}|}(y)\right]^{N-i} f_{|\mathbf{L}|}(y), \tag{24}$$

where $i \in [1, N]$, and the full derivation can be seen in [43].

3) Calculating the expected values: Given the PDF of the sorted reliabilities at each bit location, we can compute the expected values of the bit positions in the ordered reliability statistics:

$$\mathbb{E}(\phi(|\mathbf{L}|)[i]) = \int_0^\infty f_{\phi(|\mathbf{L}|)[i]}(l) \times l \;\; dl \tag{25}$$

This gives us an expected value of the LLR in each of the position in the received signal.

From this expected value, we can calculate the expected weighted Hamming distance of any error pattern GRAND, POSD and OSD applies. Owing to the linearity of expectation and given that the error patterns are deterministic constants, we can calculate the expected value of w_H of an error pattern as:

$$\mathbb{E}(w_H) = \mathbb{E}\left(\sum_{i=1}^{N} \mathbf{e}[i] \times |\mathbf{L}[i]|\right) = \sum_{i=1}^{N} \mathbf{e}[i] \times \mathbb{E}(|\mathbf{L}[i]|). \tag{26}$$

4) Generating Error Patterns: Rather than trying to produce error patterns that minimize w_H which can only be done with the knowledge of the current set of received LLRs, we choose to generate TEPs offline which minimize the expected value of w_H, $\mathbb{E}[w'_H(\mathbf{e}, \mathbf{L})]$. Computing lists of TEPs that minimize $\mathbb{E}(w_H)$ is done using the Algorithm 1 since the weighted Hamming distance, a non-negative number, satisfies the monotonicity property. The first condition specified in (9) is satisfied by noting that the expected value of the weighted Hamming distance is always a non-negative number :

$$\mathbb{E}(w_H(\mathbf{e} \cup \mathbf{1}_j)) = \left(\sum_{i=1}^{N} e[i] \times \mathbb{E}(|\mathbf{L}[i]|)\right) + \mathbb{E}(|\mathbf{L}[i]|), \tag{27}$$

$$= \mathbb{E}(w_H(\mathbf{e})) + \mathbb{E}(w_H(\mathbf{1}_j)) \geq \mathbb{E}(w_H(\mathbf{e})) \quad . \tag{28}$$

For the second condition specified in (10), :

$$\mathbb{E}(w_H(\mathbf{e})) \leq \mathbb{E}(w_H(\mathbf{e}')) \implies \mathbb{E}(w_H(\mathbf{e})) + \mathbb{E}(w_H(1_j)) \leq \mathbb{E}(w_H(\mathbf{e}')) + \mathbb{E}(w_H(1_j)), \tag{29}$$

$$\implies \mathbb{E}(w_H(\mathbf{e} \cup 1_j)) \leq \mathbb{E}(w_H(\mathbf{e}' \cup 1_j)). \tag{30}$$

Using Algorithm 1 allows us to generate a list of TEPs to be used with any memoryless channel given the knowledge of the LLR distribution of the channel. We refer to the TEPs produced using this method as EW TEPs.

B. Using the distribution of the received signal

Sometimes, the formulation of the LLR is difficult to calculate especially when $\mathcal{L}(y)$ is a non-linear, non-invertible function of y. In these cases, finding the distribution of the LLR analytically requires solving a difficult transformation-of-random-variables problem. In this case, we suggest an alternative method which depends on $f_y(y)$, the PDF of the received signal.

1) Sort the received channel signal: The, algorithm starts by sorting y in ascending order. This step gives us the distribution of each of the received bits in ascending value of received signal:

$$f_{\phi(\mathbf{y})[i]}(y) = \frac{N!}{(i-1)!(N-i)!} \left[F_{\mathbf{y}}(y)\right]^{i-1} \times \left[1 - F_y(\mathbf{y})\right]^{N-i} f_{\mathbf{y}}(y) \tag{31}$$

2) Calculate the expected value of the received signal: Then the expected value of each received signal on each bit can be calculated as:

$$\mathbb{E}(\phi(\mathbf{y})[i]) = \int_0^\infty f_{\phi(\mathbf{y})[i]}(y) \times y \;\; dy. \tag{32}$$

3) Calculate the LLR of the expected values of the received signal: The LLR of each $\mathbb{E}(\phi(|\mathbf{y}|)[i])$ is calculated:

$$\mathcal{L}(\mathbb{E}(\phi(\mathbf{y})[i])) = \ln \frac{P(\mathbb{E}(\phi(\mathbf{y})[i])|c=0)}{P(\mathbb{E}(\phi(\mathbf{y})[i])|c=1)}. \tag{33}$$

Then the reliabilities of the bits can be determined calculating the absolute values of the LLRs $|\mathcal{L}(\mathbb{E}(\phi(\mathbf{y})[i]))|$.

4) Generating Error Patterns: After sorting the LLRs in increasing $|\mathcal{L}(\mathbb{E}(\phi(\mathbf{y})[i]))|$ to produce $\phi(|\mathcal{L}(\mathbb{E}(\phi(\mathbf{y})[i])|)$, Algorithm 1 can be used to generate lists of binary error vectors by increasing:

$$\sum_{i=1}^{N} \mathbf{e}[i] \times |\mathcal{L}(\mathbb{E}(\phi(\mathbf{y}))| \, [i].$$

C. Applicability on GRAND,OSD and POSD

It is important to know that GRAND applies TEPs on all the recieved bits. The LLRs associated with the bits follow the distribution of :

$$f_{\phi(|\mathbf{L}|)[i]}(y) = \frac{n!}{(i-1)!(n-i)!} \left[F_{|\mathbf{L}|}(y)\right]^{i-1} \times \left[1 - F_{|\mathbf{L}|}(y)\right]^{n-i} f_{|\mathbf{L}|}(y). \tag{34}$$

POSD queries against the first k message bits which follow the distribution :

$$f_{\phi(|\mathbf{L}|)[i]}(y) = \frac{k!}{(i-1)!(k-i)!} \left[F_{|\mathbf{L}|}(y)\right]^{i-1} \times \left[1 - F_{|\mathbf{L}|}(y)\right]^{k-i} f_{|\mathbf{L}|}(y) \tag{35}$$

where $i \in [1, k]$. For OSD, the distribution we observe is the one shown in (24); however, since OSD operates on the set of most reliable bits, the least reliable bit in the sequence is the $(n-k)^{th}$ bit from the total distribution:

$$f_{\phi(|\mathbf{L}|)[i]}(y) = \frac{n!}{(n-k+i-1)!(k-i)!} \left[F_{|\mathbf{L}|}(y)\right]^{n-k+i-1} \times \left[1 - F_{|\mathbf{L}|}(y)\right]^{k-i} f_{|\mathbf{L}|}(y) \tag{36}$$

where $i \in [1, k]$.

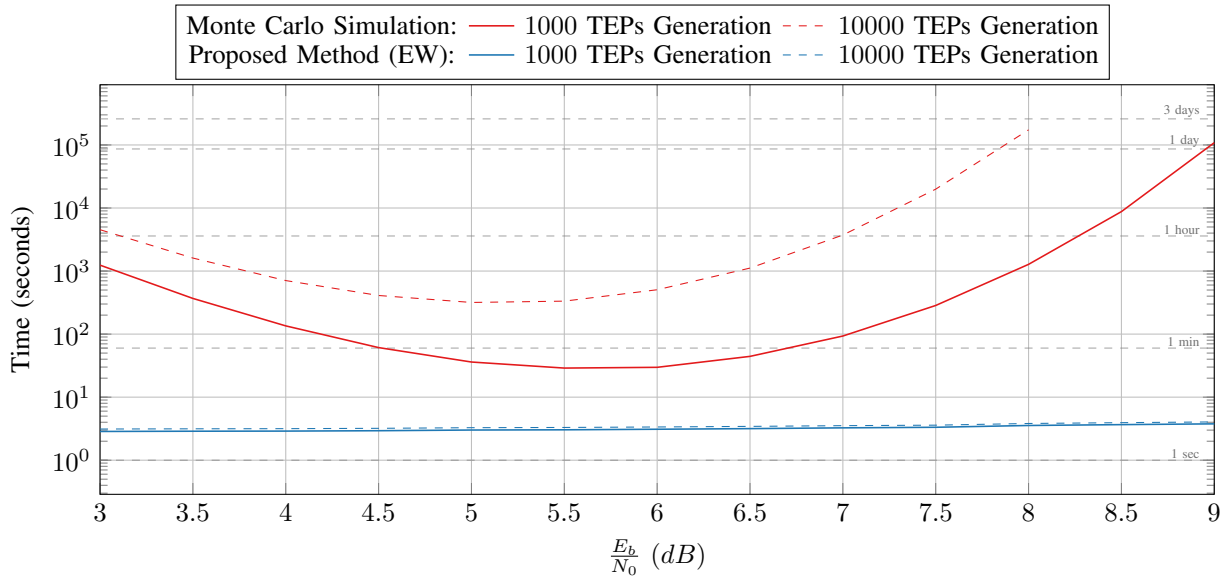

Fig. 1. Time needed to generate TEPs with Monte-Carlo Simulations and our proposed method as a function of $\frac{E_b}{N_0}$.

Hence, the distributions of the LLRs that GRAND and POSD operate on are the same distribution of the entire channel (with different sampling sizes). Therefore, it follows that the error patterns used with GRAND would be similar to the ones used with POSD. However, OSD only uses the most reliable basis that exhibits a different PDF. We will show in later sections that POSD and GRAND usually require similar error patterns, with those error patterns usually differing from the error patterns used with OSD.

Note: The error pattern $\mathbf{e}$ of GRAND is entirely generated without any input of the generator matrix. This is different from that of POSD or OSD which is partially pre-generated (for k bits) and the rest is computed up by using the generator matrix. Hence, the expected value of the weighted Hamming distance of a GRAND error pattern can be known in advance without any relation to the structure of the code; while that of a POSD / OSD error pattern depends on the structure of the code. For OSD, it is difficult to approximate the effect of the structure of the code on the weighted Hamming distance of the generated error patterns as $\mathbf{G}''$ changes with each received signal, while for POSD, it is possible to approximate the effect of the structure of the code as one generator matrix is used irregardless of the received signal.

D. Complexity Analysis

All the steps required to generate these error patterns can be performed prior to deployment in the field. Our proposed method requires complex calculations which are often difficult to do by hand. To simplify the necessary calculations needed by our methods, it is generally sufficient to use numerical integration techniques to approximate the required integrals. As such, the set of most frequent TEPs could be generated rapidly, which is a significant advantage over Monte Carlo simulations which demand long runtimes to gather a representative set of frequent TEPs.

Fig. 1 shows the time needed to generate TEPs with Monte Carlo simulations and our proposed method on an AWGN channel. We run the simulations using MATLAB on an AMD Ryzen 5 7600X 6-Core Processor (4.70 GHz) with 32 GB of RAM. To ensure reliable frequency-based ranking, Monte Carlo simulations require a minimum of 10 occurrences for each TEP. Fig. 1 shows that our proposed method is consistently able to generate 10000 TEPs in less than 4 seconds across all simulated $\frac{E_b}{N_0}$. On the other hand, the time needed to generate 10000 TEPs with Monte Carlo simulations exceeds 2 days at $\frac{E_b}{N_0} = 8$ dB, $45471\times$ that of the EW TEPs generation time.

In terms of implementation, the main overhead lies in memory, as our pre-generated TEPs must be stored. As a result, a hardware architecture similar to the fixed-latency LUT-based GRAND [26] is well suited. For example, storing MQ TEPs would require $n \times Q$ bits with GRAND [26] (or $(n-k) \times MQ$ if only the syndromes are stored [29]) and $k \times MQ$ bits with OSD/POSD, where MQ refers to a maximum number of queries. For $MQ = 1024$ TEPs and $n = 128, k = 64$, the number of stored bits is 131072 bits (16.384 kB) with GRAND using a hardware similar to [26] and 65536 bits (8.192 kB) with GRAND using a hardware similar to [29]. For OSD and POSD we need 65536 bits (8.192 kB) to store the TEPs. Finally, the proposed method can be combined with the piecewise linear approach to refine estimation, particularly in regions that are difficult to linearize.

V. Additive White Gaussian Noise

For the additive white Gaussian channel, the received channel signal can be represented as:

$$\mathbf{y} = \mathbf{x} + \mathbf{n}, \tag{37}$$

where $\mathbf{x}$ is the BPSK modulated signal and $\mathbf{n}$ is a Gaussian distribution with mean 0 and variance σ^2. After receiving the signal, the LLR of $\mathbf{y}$ can be calculated as :

$$L = \mathcal{L}(y) = \ln \frac{P(y|c=0)}{P(y|c=1)} = \ln \frac{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-1)^2}{2\sigma^2}}}{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y+1)^2}{2\sigma^2}}} = \frac{(y+1)^2 - (y-1)^2}{2\sigma^2} = \frac{2y}{\sigma^2} \tag{38}$$

A. Generating the list of TEPs

1) Finding the PDF of the reliabilities: The PDF of the LLR L of each of the received symbols can be represented as [44]:

$$f_{\mathbf{L}}(l|c=0) = \frac{1}{\sqrt{8\pi/\sigma^2}} e^{-\frac{\left(l-\frac{2}{\sigma^2}\right)^2}{\frac{8}{\sigma^2}}} \quad , \tag{39}$$

$$f_{\mathbf{L}}(l|c=1) = \frac{1}{\sqrt{8\pi/\sigma^2}} e^{-\frac{\left(l+\frac{2}{\sigma^2}\right)^2}{\frac{8}{\sigma^2}}} \quad . \tag{40}$$

Fig. 2 (a) shows a sample distribution of L. The highlighted area in blue shows the area where $c = 1$ is interpreted as 0 and the highlighted area in red shows the area where a $x = 0$ is interpreted as 1.

Owing to the law of total probability, we can represent the PDF of the LLR as:

$$f_{\mathbf{L}}(l) = P(x=+1) f_{\mathbf{L}}(l|x=+1) + P(x=-1) f_{\mathbf{L}}(l|x=-1), \tag{41}$$

$$= \frac{1}{2} \times \frac{1}{\sqrt{8\pi/\sigma^2}} e^{-\frac{\left(l-\frac{2}{\sigma^2}\right)^2}{\frac{8}{\sigma^2}}} + \frac{1}{2} \times \frac{1}{\sqrt{8\pi/\sigma^2}} e^{-\frac{\left(l+\frac{2}{\sigma^2}\right)^2}{\frac{8}{\sigma^2}}} . \tag{42}$$

After generating the absolute values of the LLRs, the distribution changes based on (21) and we obtain :

$$f_{|\mathbf{L}|}(l) = \frac{1}{\sqrt{8\pi/\sigma^2}} \left(e^{-\frac{\left(l-\frac{2}{\sigma^2}\right)^2}{\frac{8}{\sigma^2}}} + e^{-\frac{\left(l+\frac{2}{\sigma^2}\right)^2}{\frac{8}{\sigma^2}}} \right) \quad , l \in [0, +\infty), \tag{43}$$

$$F_{|\mathbf{L}|}(l) = \int_0^l f_{|\mathbf{L}|}(x)dx = \frac{1}{2} \left(erfc\left(\frac{\sqrt{2/\sigma^2} + l}{2\sqrt{2/\sigma^2}} \right) - erfc\left(\frac{\sqrt{2/\sigma^2} - l}{2\sqrt{2/\sigma^2}} \right) \right) \quad , l \in [0, +\infty). \tag{44}$$

The resulting distribution of the absolute value of the LLR is plotted in Fig. 2 (b).

2) Sorting the reliabilities: Plugging $F_{|\mathbf{L}|}$ (44) and $f_{|\mathbf{L}|}$ (43) into (24), we obtain the distributions of the individual sorted LLRs. We use MATLAB to calculate and plot the individual distributions of a subset of the bits in Fig. 2 (c).

3) Expected Value Calculation: From these distributions we can use (25) to calculate the expected values of the individual bits which is plotted in Fig. 2 (d).

4) Generating Error Patterns: The algorithm in section II-C is applied on the expected values of the LLRs calculated in the previous step to generate the list of EW TEPs.

B. Decoding Performance Comparison

Fig. 3 shows the performance of GRAND, POSD and OSD with the proposed EW, and the existing LW, HW, ILW and ML TEPs. High-rate codes are decoded using GRAND, whereas medium-rate to low-rate codes are decoded using POSD and OSD. This is because GRAND needs to guess the entire error pattern affecting the codeword, and it struggles to match the decoding performance of POSD and OSD when there are many bits in error. We discuss the performance of each of these decoders with different TEPs and varying codes.

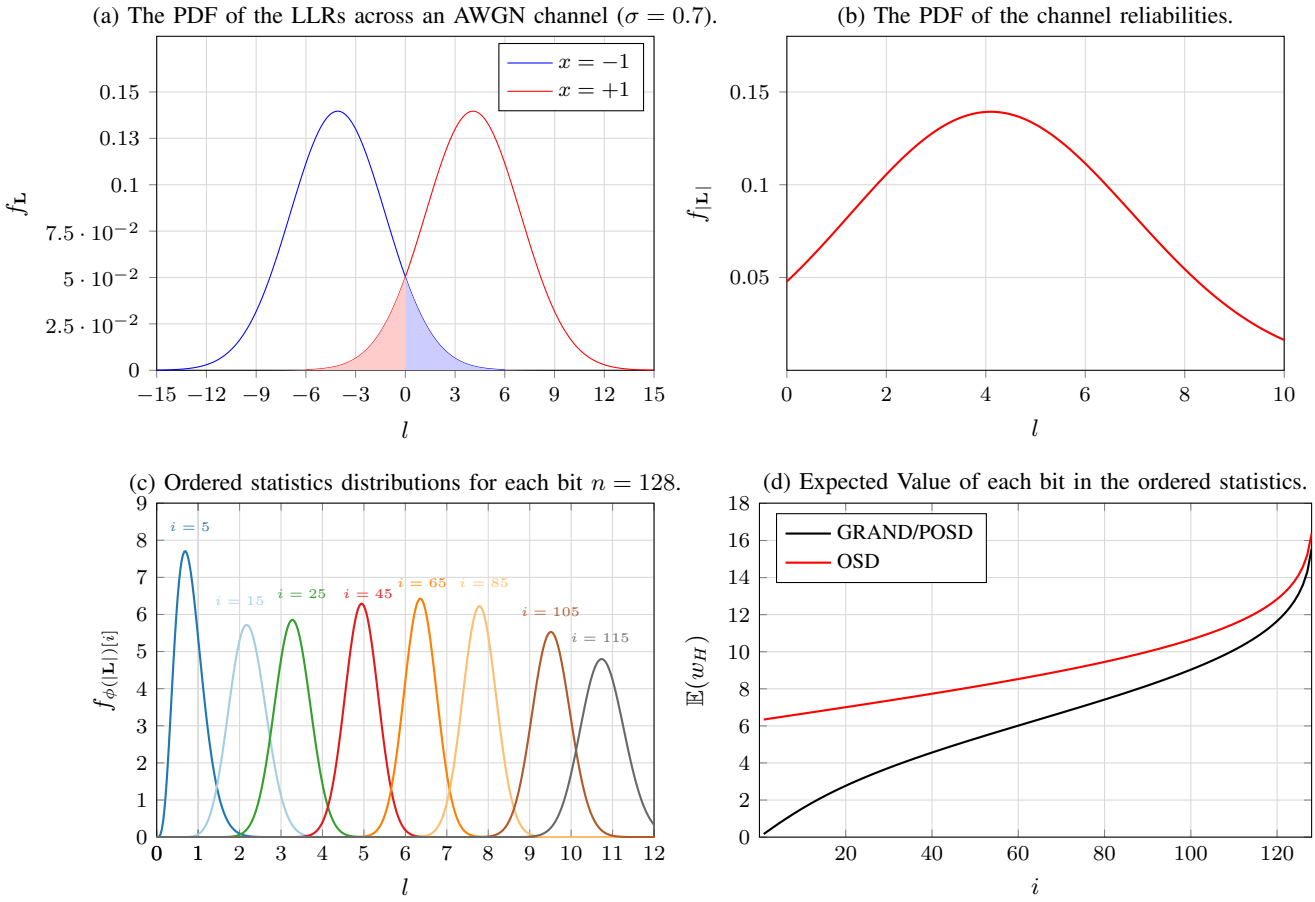

Fig. 2. The steps followed to generate the EW error patterns in an AWGN channel.

1) GRAND: We simulate GRAND on Random Linear Code (RLC) [127,85] and 5G Cyclic Redundancy Check Aided (CA)-Polar code [128,105+11] (with 11 Cyclic Redundancy Check (CRC) bits). For a maximum number of queries of 10^6, we can observe that the EW error patterns result in a $0.2 \rightarrow 0.3$ dB gain compared to LW error patterns and 0.1 dB gain compared to ILW error patterns at target FER 10^{-5}. Using the ML TEPs result in 0.3 dB performance gain compared to the EW error patterns with RLC [127,85] code and a 0.1 dB performance gain compared to the EW error patterns with CA-Polar [128,105+11] code. HW TEPs result in a performance loss of $\approx 2 \rightarrow 2.5$ dB compared to the EW error patterns target FER 10^{-5}.

For a maximum number of queries of 10^3, the ML, EW, ILW, LW, and HW TEPs exhibit the same trend of decoding performance as what we observed with $MQ = 10^6$. However, for RLC [127,85] and a target FER 10^{-5}, we observe a slight 0.1 dB gain of the LUT error patterns compared to the EW error patterns. This performance gain is not observed with CA-Polar [128,105+11] code as both EW and LUT error patterns perform similarly.

2) POSD: We simulate POSD on BCH [127,64] and RLC code [128,32]. For a maximum number of queries of 10^5, we can observe that the EW error patterns and the LW error patterns result in the same performance. Additionally, we can observe a $0.2 \rightarrow 0.5$ dB gain compared to ILW error patterns at target FER 10^{-4}. Using the ML TEPs results in 0.1 dB performance gain compared to the EW error patterns with BCH [127,64] and RLC code [128,32] at target FER 10^{-4}. HW TEPs also achieve poor performance as they result in a performance loss of ≈ 2.3 dB compared to the EW error patterns at target FER 10^{-4}.

For a maximum number of queries of 10^3 we observe that the performance of the error patterns changes between BCH [127,64] and RLC [128,32]. At target FER 10^{-4} with BCH [127,64], LUT TEPs outperform EW and ILW TEPs by 0.1 dB. Additionally, ILW and EW error pattern outperform LW TEPs by 0.1 dB. However, for RLC [128,32] and a target FER 10^{-4}, we observe a slight 0.1 dB gain of the EW error patterns compared to the to the LUT error patterns. For this code, the LW and LUT error patterns produce almost the same decoding performance, but obtain a 0.3 dB gain compared to the LW TEPs at target FER 10^{-4}.

3) OSD: We simulate OSD on BCH [127,64] and RLC code [128,32]. For a maximum number of queries of 10^4, we can observe that the EW, HW, ML TEPs result in the same decoding performance. Additionally, we can observe a $0.1 \rightarrow 0.3$ dB gain compared to ILW TEPs at target FER 10^{-5}. LW TEPs achieve poor performance as they result in a performance loss of $\approx 1.5 \rightarrow 2.5$ dB compared to the EW error patterns at target FER 10^{-4}. For a maximum number of queries of 10^3 we

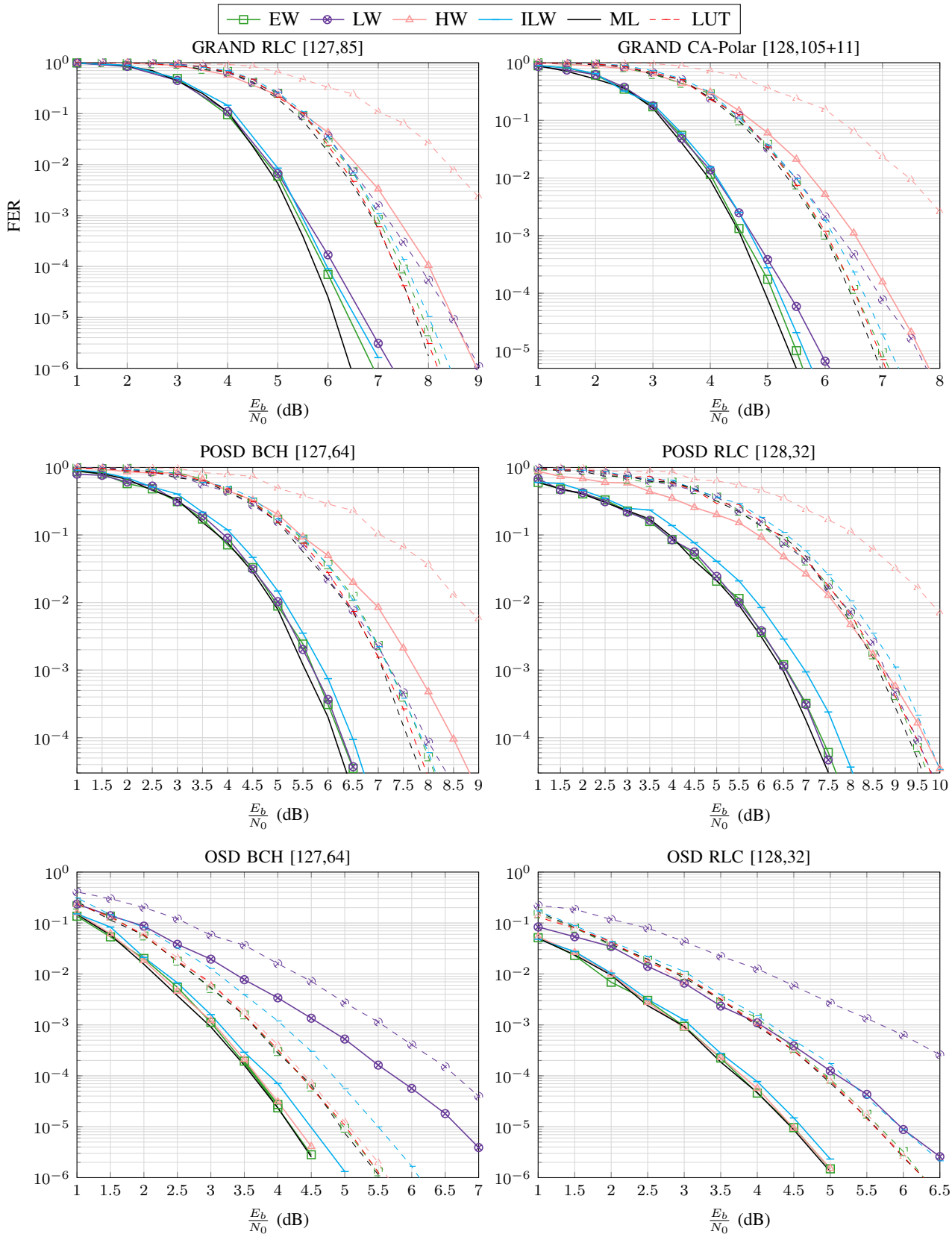

Fig. 3. Comparison of decoding performance of different sets of TEPs with GRAND, POSD and OSD on an AWGN channel. Straight lines represent simulations run with 10^6, 10^5 and 10^4 maximum number of queries with GRAND, POSD and OSD respectively. Dashed lines represent simulations run with a maximum number of queries of 10^3.

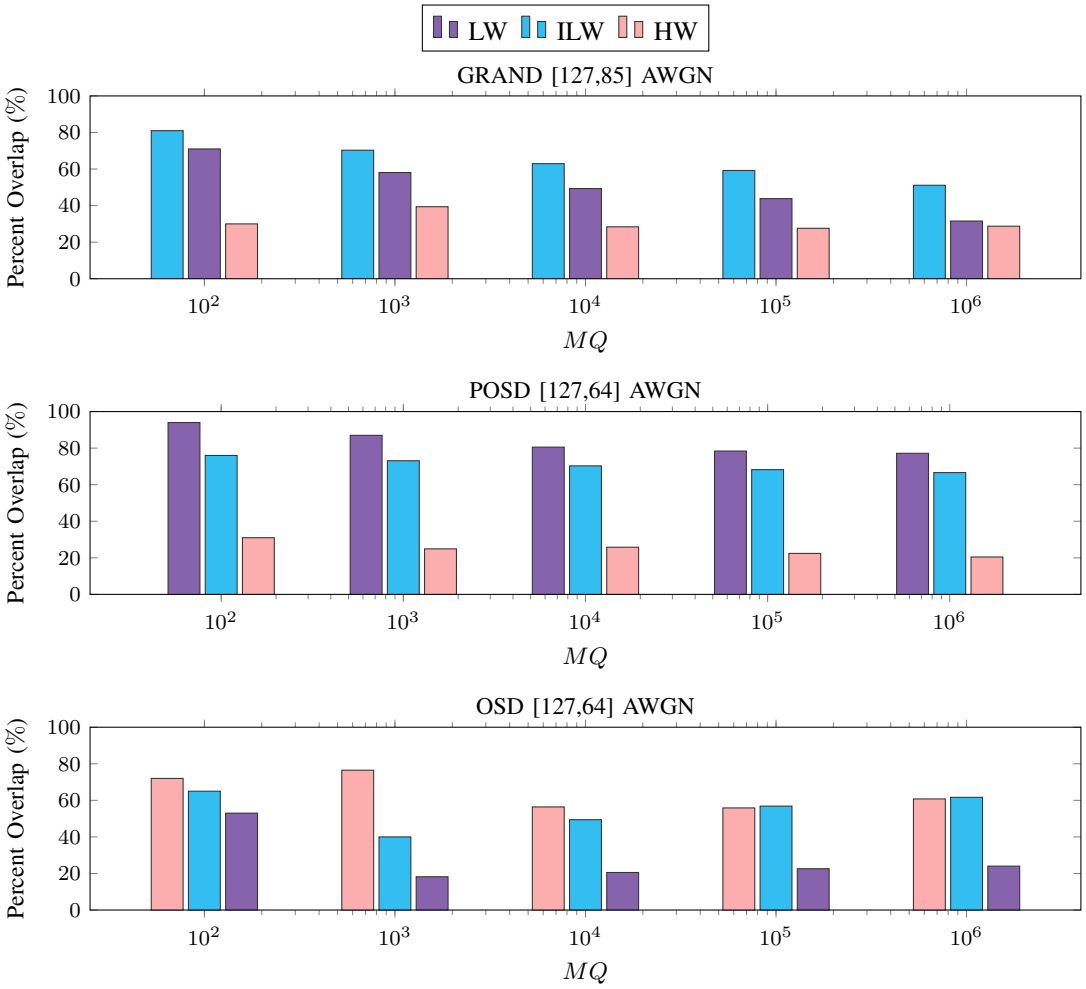

Fig. 4. Percentage of overlap of the pre-generated test error patterns (TEPs) with EW TEPs for GRAND, POSD, and OSD over an AWGN channel.

observe that the LUT, HW, EW and ML TEP performance is the same. At target FER 10^{-5} we observe that the ILW TEPs attain a 0.4 dB to 0.5 dB loss compared to the LUT, HW, EW and ML. This degradation in performance is more pronounced than that observed with $MQ = 10^4$.

C. TEP analysis

Fig. 4 shows the percentage of overlap between the EW TEPs and the LW, ILW and HW TEPs with GRAND, POSD and OSD. We discuss the correlation between the percentage overlap of TEPs and the decoding performance below.

1) GRAND: ILW error patterns have the largest percentage of overlap with the EW TEPs of 81% at $MQ = 10^2$ and 51% at $MQ = 10^6$. This explains why the performance of the ILW error patterns supersedes that of the LW error patterns with 71% common TEPs with EW TEPs at $MQ = 10^2$ and 31% at $MQ = 10^6$. The HW TEPs consistently have low number of overlapping TEPs with EW with 30% common TEPs at $MQ = 10^2$ and 29% at $MQ = 10^6$. This results in $\approx 1.5 \rightarrow 2.5$ dB performance loss compared to EW TEPs.

2) POSD: The LW error patterns have the largest overlap with the EW TEPs of $94\% \rightarrow 78\%$ at $MQ = 10^2 \rightarrow 10^5$ compared to ILW which has $76\% \rightarrow 68\%$ overlapping TEPs with EW error patterns. This lower number of common TEPs explains why ILW error patterns are less performing for POSD than LW error patterns. HW TEPs share $31\% \rightarrow 20\%$ overlapping TEPs with EW TEPs, which results in $\approx 1.5 \rightarrow 2.5$ dB performance loss compared to EW TEPs.

3) OSD: HW TEPs share the highest percentage of common TEPs with EW, with an overlap of 72% at $MQ = 10^2$ and 56% at $MQ = 10^4$. This is followed by ILW TEPs with 65% at $MQ = 10^2$ and 49% at $MQ = 10^4$. In contrast, LW TEPs show a larger drop in common TEPs with EW, falling from 53% at $MQ = 10^2$ to approximately 21% at $MQ = 10^4$.

D. Conclusion

We can consistently see that the performance of the predetermined TEPs correlates with the percentage of common TEPs with the EW TEPs. In general, EW TEPs either obtain a similar performance or outperform other TEPs except for the ML

Table II: Parameters of the Gaussian distributions in the mixed Gaussian channel.

.

i	Channel 1			Channel 2		
	$\mu[i]$	$\omega[i]$	$\sigma[i]$	$\mu[i]$	$\omega[i]$	$\sigma[i]$
1	-3.0	0.29	0.3555	-2.7	0.29	0.3555
2	-0.1	0.01	0.13	-0.1	0.01	0.13
3	0.0	0.40	0.10	0.0	0.40	0.10
4	0.1	0.01	0.13	0.1	0.01	0.13
5	3.0	0.29	0.3555	2.7	0.29	0.3555

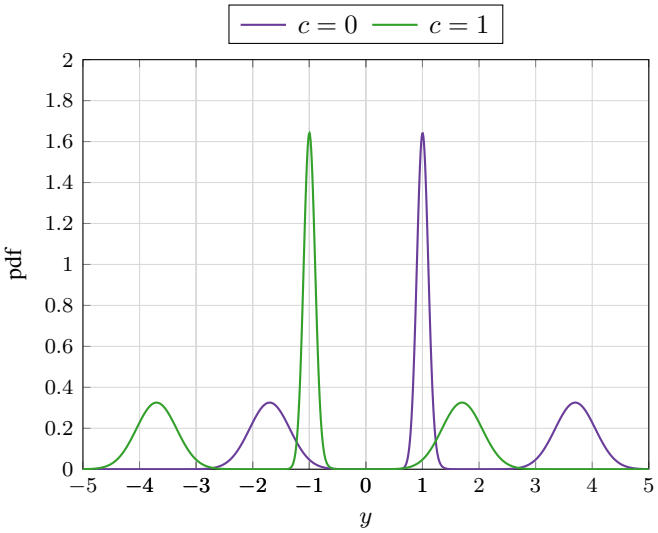

Fig. 5: The distribution of the PDF of the transmitted codeword bits along the mixed Gaussian channel 2.

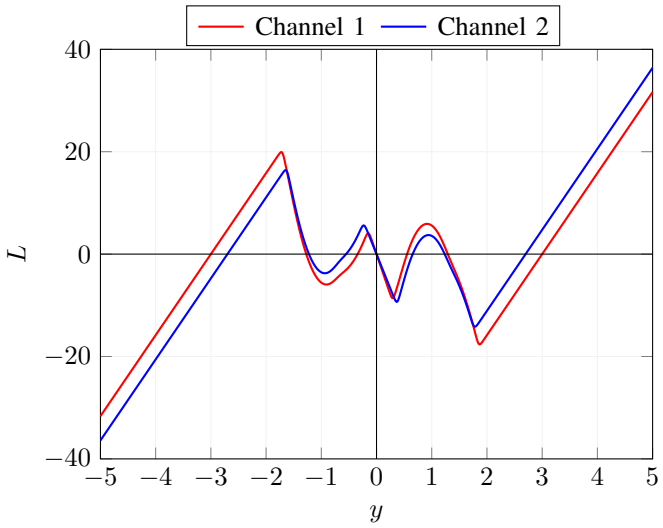

Fig. 6: The LLR value as a function of the received signal.

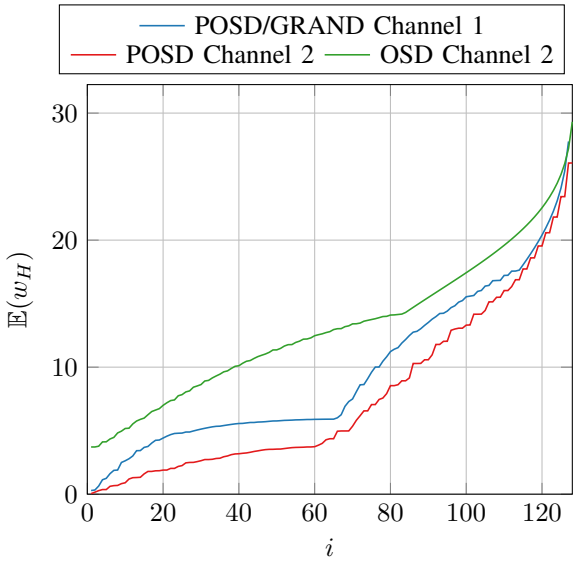

Fig. 7: The expected value of the ordered statistics of the absolute values of the LLRs for 128 bits used in each algorithm for channels 1 and 2 as a function of ordered bit position.

TEPs which are hardware incompatible. EW TEPs generally obtain the same performance as the LUT error patterns, but suffer a small performance degradation of 0.1 dB with GRAND on RLC [127,85] and POSD [127,64] due to slight variations in the error patterns produced by the LUT method and the EW method with $MQ = 10^3$ TEPs. This degradation in performance is generally acceptable in light of our ability to generate more EW error patterns on demand, without requiring long simulation times.

VI. Mixture of White Gaussian Channel

In this section we discuss the case of a mixture of white Gaussian channel. We choose this type of channel since it can approximate any smooth density function with any specific nonzero amount of error with enough components [45]. We consider an example of this channel and produce TEPs catered for this channel using the method discussed in section IV-B. We then compare against the SOTA TEPs suggested in Section III-D.

The density function of a mixture of white Gaussian channel noise with γ Gaussian elements can be represented as [46]:

$$f_{\mathbf{y}}(y) = \sum_{i=1}^{\gamma} \frac{\omega[i]}{\sqrt{2\pi\sigma[i]^2}} e^{-\frac{y^2}{2\sigma[i]^2}} \tag{45}$$

TABLE III
APPROXIMATION OF THE LLR FUNCTION FOR A MIXTURE OF GAUSSIAN DISTRIBUTIONS.

Segment	Channel 1			Channel 2		
	Slope	Intercept	Domain	Slope	Intercept	Domain
1	15.8	47.4	$[-\infty, -1.7)$	15.8	42.7	$[-\infty, -1.65)$
2	-35.3	-42.9	$[-1.7, -0.94)$	-31.1	-36.5	$[-1.65, -0.94)$
3	11.5	3.8	$[-0.94, -0.17)$	13.8	8.1	$[-0.94, -0.24)$
4	-30.5	0	$[-0.17, 0.3)$	-26.5	0	$[-0.24, 0.38)$
5	23.8	-14.12	$[0.3, 0.96)$	24.7	-17.26	$[0.38, 0.94)$
6	-27	34.19	$[0.96, 1.83)$	-22.9	27.53	$[0.94, 1.78)$
7	15.8	-47.43	$[1.83, +\infty)$	15.8	-42.7	$[1.78, +\infty)$

where:

$$\sum_{i=1}^{\gamma} \omega[i] = 1. \tag{46}$$

The approximation of the LLR in this case is not always trivial as it is not always linear with respect to the received signal. Hence, we use the method discussed in Section IV-B. Rather than finding the $f_{\mathbf{L}}(l)$, we focus on finding $f_{\mathbf{y}}(y)$, the PDF of the received signal. Due to our assumption about BPSK modulation, this is the shifted PDF of the noise signal. For a modulated signal of $+1$, the distribution of the received signal at the receiver end is:

$$f_{\mathbf{y}}(y|c=0) = \sum_{i=1}^{\gamma} \frac{\omega[i]}{\sqrt{2\pi\sigma[i]^2}} e^{-\frac{(y-1)^2}{2\sigma[i]^2}}. \tag{47}$$

For a modulated signal of -1 , the distribution of the received signal at the receiver end is:

$$f_{\mathbf{y}}(y|c=1) = \sum_{i=1}^{\gamma} \frac{\omega[i]}{\sqrt{2\pi\sigma[i]^2}} e^{-\frac{(y+1)^2}{2\sigma[i]^2}}. \tag{48}$$

Following the law of total probabilities, the distribution of the received signal is hence:

$$f_{\mathbf{y}}(y) = P(x=+1) \times f_{\mathbf{y}}(y|x=+1) + f_{\mathbf{y}}(y|x=-1) \times P(x=-1) \tag{49}$$

$$= \frac{1}{2} \sum_{i=1}^{\gamma} \frac{\omega[i]}{\sqrt{2\pi\sigma[i]^2}} e^{-\frac{(y-1)^2}{2\sigma[i]^2}} + \frac{1}{2} \sum_{i=1}^{\gamma} \frac{\omega[i]}{\sqrt{2\pi\sigma[i]^2}} e^{-\frac{(y+1)^2}{2\sigma[i]^2}}. \tag{50}$$

Additionally the LLR of each received signal y is:

$$\mathcal{L}(y) = \ln \frac{P(y|x=1)}{P(y|x=-1)} = \ln \frac{\sum_{i=1}^{\gamma} \frac{\omega[i]}{\sqrt{2\pi\sigma[i]^2}} e^{-\frac{y^2}{2\sigma[i]^2}}}{\sum_{i=1}^{\gamma} \frac{\omega[i]}{\sqrt{2\pi\sigma[i]^2}} e^{-\frac{y^2}{2\sigma[i]^2}}} \tag{51}$$

A. Channel Models

Assume two channels defined by the parameters in TABLE II. We have a mixture of 5 Gaussian distributions for each channel. For custom channel 1, the main components are situated at -3, 0 and 3. For custom channel 2, the main components are situated at -2.7, 0 and 2.7. We plot the distribution of channel 2 in Fig. 5. The LLR corresponding to the received signal distributions in channel 1 and 2 can be seen in Fig. 6. We note that this LLR is non-linear with multiple x-axis intersections. Any received signal that maps to a positive LLR is considered the bit 0 and every received signal that maps to a negative LLR is considered a 1 bit.

After completing the method described in Section IV-B, we obtain the expected values of the received signal. Then the LLR of the received signal is calculated as: The LLR of each $\mathbb{E}(\phi(|y|)[i])$ is:

$$\mathcal{L}(\mathbb{E}(\phi(|y|)[i])) = \ln \frac{P(\mathbb{E}(\phi(|y|)[i])|x=1)}{P(\mathbb{E}(\phi(|y|)[i])|x=-1)} = \ln \frac{\sum_{i=1}^{\gamma} \frac{\omega[i]}{\sqrt{2\pi\sigma[i]^2}} e^{-\frac{(\mathbb{E}(\phi(|y|)[i])-1)^2}{2\sigma[i]^2}}}{\sum_{i=1}^{\gamma} \frac{\omega[i]}{\sqrt{2\pi\sigma[i]^2}} e^{-\frac{(\mathbb{E}(\phi(|y|)[i])+1)^2}{2\sigma[i]^2}}}, \tag{52}$$

and it is shown in Fig. 7. We note that channel 1 is more reliable than channel 2 hence, we will analyze channel 1 with POSD and GRAND that have limited error correcting capabilities and channel 2 with OSD.

B. Performance Analysis

Since this channel has a fixed SNR, we discuss the FER performance of the decoders as a function of maximum number of queries shown in Fig 8. Due to the computational complexity of simulating these channels and calculating the LLRs, obtaining the LUT error patterns proved to be infeasible within a reasonable time duration of 7 days with 50 CPU cores. Hence, we will present the performance of the remaining TEPs with this mixture of white Gaussian channel.

1) GRAND: With channel 1, we can observe that for RLC (127,85) and a target FER 10^{-2}, GRAND with ILW and LW requires $3.8\times$ and $13\times$ the number of queries required with EW error patterns respectively. Compared to the ML TEPs, 40% more EW TEPs are needed to achieve target FER 10^{-2}. Similarly with CA-Polar (128,105) code, and a target FER 10^{-2}, GRAND with ILW and LW require $1.8\times$ and $14.7\times$ the number of queries required with EW error patterns respectively. Compared to the ML TEPs, 30% more EW TEPs are needed to achieve target FER 10^{-2}. The HW TEPs do not reach target FER 10^{-2} in both simulations.

2) POSD: For POSD, we can observe with channel 1 for BCH (127,64) and a target FER 10^{-3}, GRAND LW require $3.3\times$ the number of queries required with EW error patterns. Additionally, 50% more EW TEPs are needed to achieve target FER 10^{-3} compared to the ML TEPs. The HW and ILW TEPs do not reach target FER 10^{-3} in this simulation. Alternatively with channel 2, we can observe for RLC (128,32) and a target FER 10^{-3}, GRAND with LW error patterns requires the almost the same number of queries as required with EW error patterns. ILW error patterns require 35% more TEPs than EW. Additionally, 30% more EW TEPs are needed to achieve target FER 10^{-3} compared to the ML TEPs. The HW TEPs do not reach target FER 10^{-3} in this simulation.

3) OSD: For OSD, we can observe with channel 2 for BCH (127,64) and a target FER 10^{-4}, GRAND with HW and LW TEPs require $3\times$ and $3.7\times$ the number of queries required with EW error patterns respectively. In this case, the EW, ML and ILW TEPs provide the same performance. Alternatively with RLC (127,85) and a target FER 10^{-2}, GRAND with LW error patterns requires $11.5\times$ the number of queries required with EW TEPs to obtain the same performance. ILW error patterns require 37% more TEPs than EW, and the HW TEPs perform similarly to the EW error patterns. Additionally, 10% more EW TEPs are needed to achieve target FER 10^{-2} compared to the ML TEPs. In this graph, we also observe that between $MQ = 10^5$ and $MQ = 10^6$ increasing the MQ of ILW, EW and ML TEPs results in little improvement in FER performance.

C. Similarity between TEPs

Fig. 9 shows the percentage of overlap between LW, ILW and HW TEPs with EW TEPs with GRAND, POSD and OSD along mixed Gaussian channels 1 and 2. For each maximum number of queries, we compare the percentage of overlap and relate it to the decoding performance.

1) GRAND: LW error patterns have the largest overlap with the EW TEPs of 94% at $MQ = 10^2$ and goes down to 46.5% at $MQ = 10^6$. ILW TEPs start with 69% common TEPs with EW TEP at $MQ = 10^2$ and reach at 51.6% at $MQ = 10^6$. Notably, the ILW TEPs' overlap surpasses LW at high query counts and correlates with how ILW TEPs outperform LW TEPs in our simulations which are conducted with $MQ = 10^6$. The HW TEPs consistently share a low percentage of overlap with EW of 14% overlap TEPs at $MQ = 10^2$ and 11.8% at $MQ = 10^6$, which results in a large decoding performance loss compared to EW TEPs.

2) POSD: For POSD with channel 1, the LW error patterns have the largest overlap with the EW TEPs of 83% at $MQ = 10^2$ and 62% at $MQ = 10^5$. This is also the case with channel 2 with an overlap of 88% at $MQ = 10^2$ and 62.1% at $MQ = 10^5$. The ILW TEPs have a lower percentage overlap of 68% / 72% common TEPs with EW error patterns at $MQ = 10^2$ decreasing to 59% / 60% at $MQ = 10^5$ on channels 1 / 2 respectively. This lower number of common TEPs correlates with why ILW error patterns are less performing for POSD than LW error patterns. HW TEPs share $30\% \rightarrow 35\%$ common TEPs with EW at $MQ = 10^2$ dropping to $\approx 20\%$ at $MQ = 10^5$ on channel 1 and 2 respectively.

3) OSD: ILW TEPs share the highest percentage of overlapping TEPs with EW, with an overlap of 82% at $MQ = 10^2$ increasing to 86% at $MQ = 10^6$. LW TEPs follow with an overlap of overlap, moving from 56% at $MQ = 10^2$ to approximately 55% at $MQ = 10^6$. Meanwhile, HW TEPs show a significant drop in common TEPs with EW, falling from 49% at $MQ = 10^2$ (close to that obtained with LW) to approximately 32% at $MQ = 10^6$.

D. Conclusion

We can see that the performance of the predetermined TEPs generally correlates with the percentage of common TEPs with the EW TEPs along channels 1 and 2. Additionally, we have shown that the EW error patterns perform most closely to the ML error patterns even without the need to derive the PDF of the LLRs. Consistently, EW TEPs are still able to map to newly encountered additive noise profiles.

VII. Uncorrelated Fast Fading

The Rayleigh fading channel models the case where there is no direct line of sight between transmitter and receiver. The transmitted signal is scattered off surfaces and the same transmitted signal interferes either destructuctively or constructively

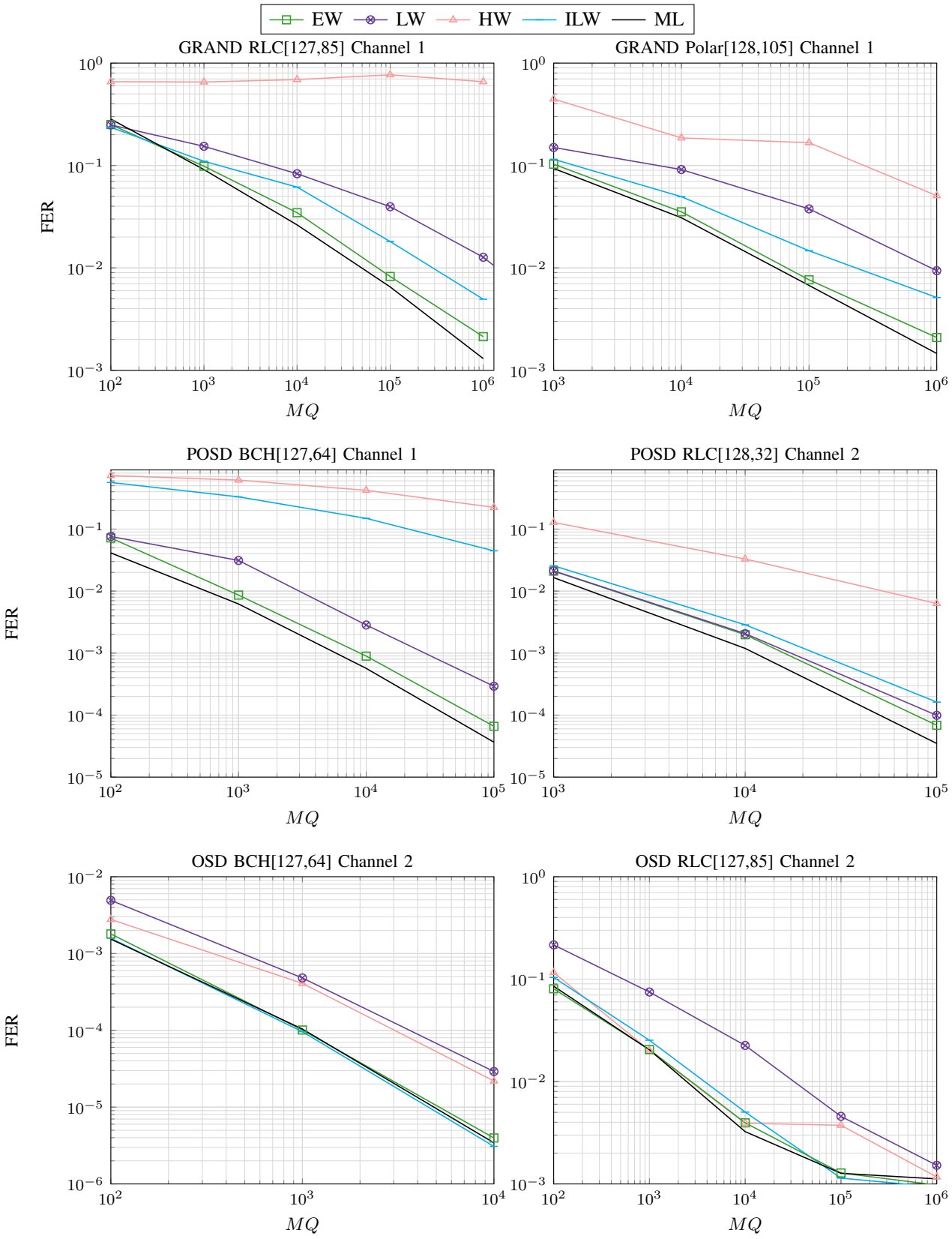

Fig. 8. Decoding performance of the TEPs on custom mixture of Gaussian channels.

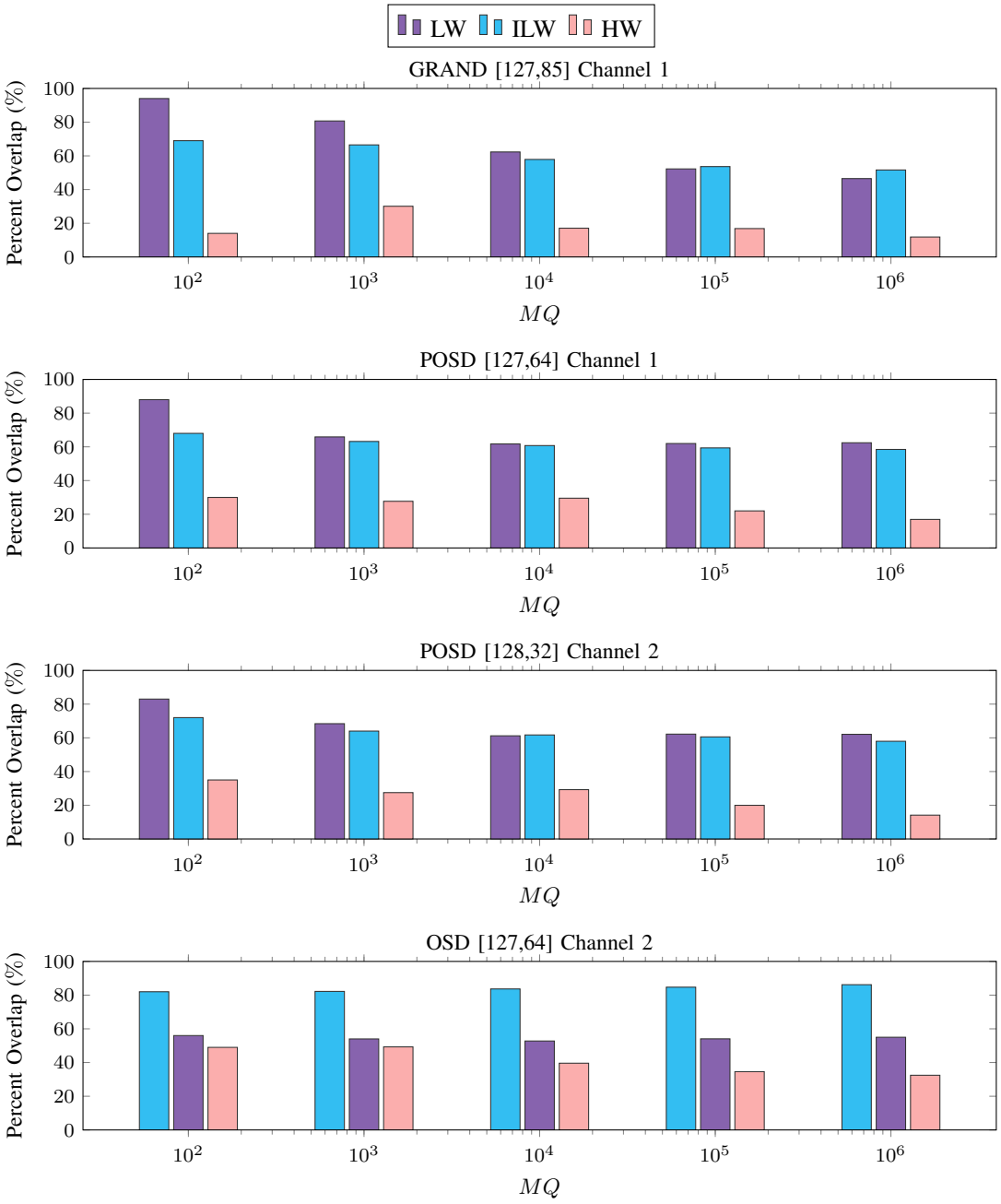

Fig. 9. Percentage of the overlap of the TEPs generated by SOTA methods with the EW error patterns with a mixture of Gaussian distribution channels.

with itself. This process can be modeled by a fading gain h multiplied to the modulated signal. As such, in a Rayleigh fading channel, the receiver receives a channel signal with value:

$$y = h \times x + n \tag{53}$$

where $n \sim \mathcal{N}(0, \sigma^2)$ and h is Rayleigh distributed with a average power of 1 ($E[h^2] = 1$).

The distribution of the channel signal conditioned on fading coefficient and the sent code bit is [44]:

$$f_{\mathbf{y}}(y|h, x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\left(\frac{(y-hx)^2}{2\sigma^2}\right)} \tag{54}$$

We can easily observe that the channel is output-symmetric defined by the notion of $f(-y|h, -x) = f(y|h, x)$ [47]:

$$f_{\mathbf{y}}(-y|h, -x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\left(\frac{(-y+hx)^2}{2\sigma^2}\right)} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\left(\frac{(y-hx)^2}{2\sigma^2}\right)} = f_{\mathbf{y}}(y|h, x). \tag{55}$$

The notion of output-channel symmetry helps in simplifying the analysis. First remark that:

$$f_{\mathbf{L}}(l) = P(x=+1)f_L(l|x=+1) + P(x=-1)f_L(l|x=-1). \tag{56}$$

Hence, to calculate the reliability of the individual bits, we can expand (21) with $l \geq 0$:

$$\begin{aligned} f_{|\mathbf{L}|}(l) =& f_L(l) + f_{\mathbf{L}}(-l), && (57)\\ =& P(x=+1)f_{\mathbf{L}}(l|x=+1) + P(x=-1)f_{\mathbf{L}}(l|x=-1), && (58)\\ &+ P(x=+1)f_{\mathbf{L}}(-l|x=+1) + P(x=-1)f_{\mathbf{L}}(-l|x=-1), && (59)\\ =& f_{\mathbf{L}}(l|x=+1) + f_{\mathbf{L}}(-l|x=+1). && (60) \end{aligned}$$

Where the last equation follows from the fact that the similary colored terms are equal. Specifically, we can see that:

$$\begin{aligned} P(x=+1) =& P(x=-1) = 0.5, && \text{(Equiprobable input symbols)}, && (61)\\ f_{\mathbf{L}}(l|x=+1) =& f_{\mathbf{L}}(-l|x=-1), && \text{(Symmetry)}, && (62)\\ f_{\mathbf{L}}(l|x=-1) =& f_{\mathbf{L}}(-l|x=+1), && \text{(Symmetry)}. && (63) \end{aligned}$$

Hence, to simplify the analysis, we can use the distribution of the LLR conditioned on the $x = 1$ or $c = 0$ without the need to calculate the conditional probability given that $x = -1$ being transmitted. In this section we will first discuss how EW TEPs can be generated for a Rayleigh fading channel with perfect knowledge of CSI and then discuss how they can be generated for a Rayleigh fading channel without perfect knowledge of channel state information NCSI.

A. Ideal Channel State Information (CSI)

In the uncorrelated fast fading channel with ideal channel state information (CSI), the value of h is known for the receiver. Hence, after the signal is received, the fading gain is removed from the received signal producing:

$$\frac{y}{h} = x + \frac{n}{h}. \tag{64}$$

The LLR in this case can be calculated as [44]:

$$L = \frac{2}{\sigma^2} y \times h \tag{65}$$

The distribution of the LLRs conditioned on $c = 0$ being sent is [44]:

$$f_{\mathbf{L}}(l|c=0) = \frac{\sigma}{\sqrt{2\pi}} e^{\frac{-l(\sqrt{2\sigma^2+1}-1)}{2}} \int_0^\infty e^{-\frac{\left(\frac{\sigma^2}{2h} l - h\sqrt{2\sigma^2+1}\right)^2}{2\sigma^2}} dh. \tag{66}$$

Hence, the PDF of the reliabilities of the bits can be calculated for $l \geq 0$ as:

$$f_{|\mathbf{L}|}(l) = \frac{\sigma}{\sqrt{2\pi}} e^{\frac{-l(\sqrt{2\sigma^2+1}-1)}{2}} \int_0^\infty e^{-\frac{\left(\frac{\sigma^2}{2h} l - h\sqrt{2\sigma^2+1}\right)^2}{2\sigma^2}} dh + \frac{\sigma}{\sqrt{2\pi}} e^{\frac{l(\sqrt{2\sigma^2+1}-1)}{2}} \int_0^\infty e^{-\frac{\left(\frac{\sigma^2}{2h} l + h\sqrt{2\sigma^2+1}\right)^2}{2\sigma^2}} dh. \tag{67}$$

We use the same remaining steps as described in Section IV-A to generate the TEPs for this channel.

1) Decoding Performance Comparison: Fig. 10 shows the performance of GRAND, POSD and OSD with the proposed EW, and the existing LW, HW, ILW and ML TEPs. We discuss the performance of each of these decoders with different TEPs and varying codes.

GRAND: We simulate GRAND on RLC [127,85] and 5G CA-Polar code [128,105+11]. For a maximum number of queries of 10^6, we can observe that the EW error patterns result in a $0.1 \rightarrow 0.3$ dB gain compared to LW error patterns and 1.3 dB gain compared to ILW error patterns at target FER 10^{-5}. Using the ML TEPs result in 0.3 dB performance gain compared to the EW error patterns with RLC [127,85] code and a 0.1 dB performance gain compared to the EW error patterns with CA-Polar [128,105+11] code. HW TEPs do not reach the target FER 10^{-5} within the simulated range of Signal to Noise Ratio (SNR)s. For a maximum number of queries of 10^3 we observe the same trend of decoding performance of the EW, LW, ILW and the HW error patterns as what we observed with $MQ = 10^6$. LUT TEPs perform similarly to the EW TEPs with $MQ = 10^3$ and RLC [127,85] and Polar [128,105+11].

POSD: We simulate POSD on BCH [127,64] and RLC code [128,32]. For a maximum number of queries of 10^5, we can observe that the EW error patterns result in $0.4 \rightarrow 0.6$ dB gain in performance compared to LW TEPs at target FER 10^{-4}. Additionally, we can observe a $1.5 \rightarrow 2$ dB gain compared to ILW error patterns at target FER 10^{-4}. Using the ML TEPs results in $0.1 \rightarrow 0.3$ dB performance gain compared to the EW error patterns . HW TEPs also achieve poor performance and they do not reach target FER 10^{-4} in the simulated SNR range. For a maximum number of queries of 10^3 we observe the same pattern in decoding performance of the EW, LW, ILW and HW error patterns compared to the $MQ = 10^5$ case. A wider performance loss is achieved using ILW TEPs with $MQ = 10^3$ of ≈ 2.5 dB at target FER 10^{-4} compared to the $1.5 \rightarrow 2$ dB

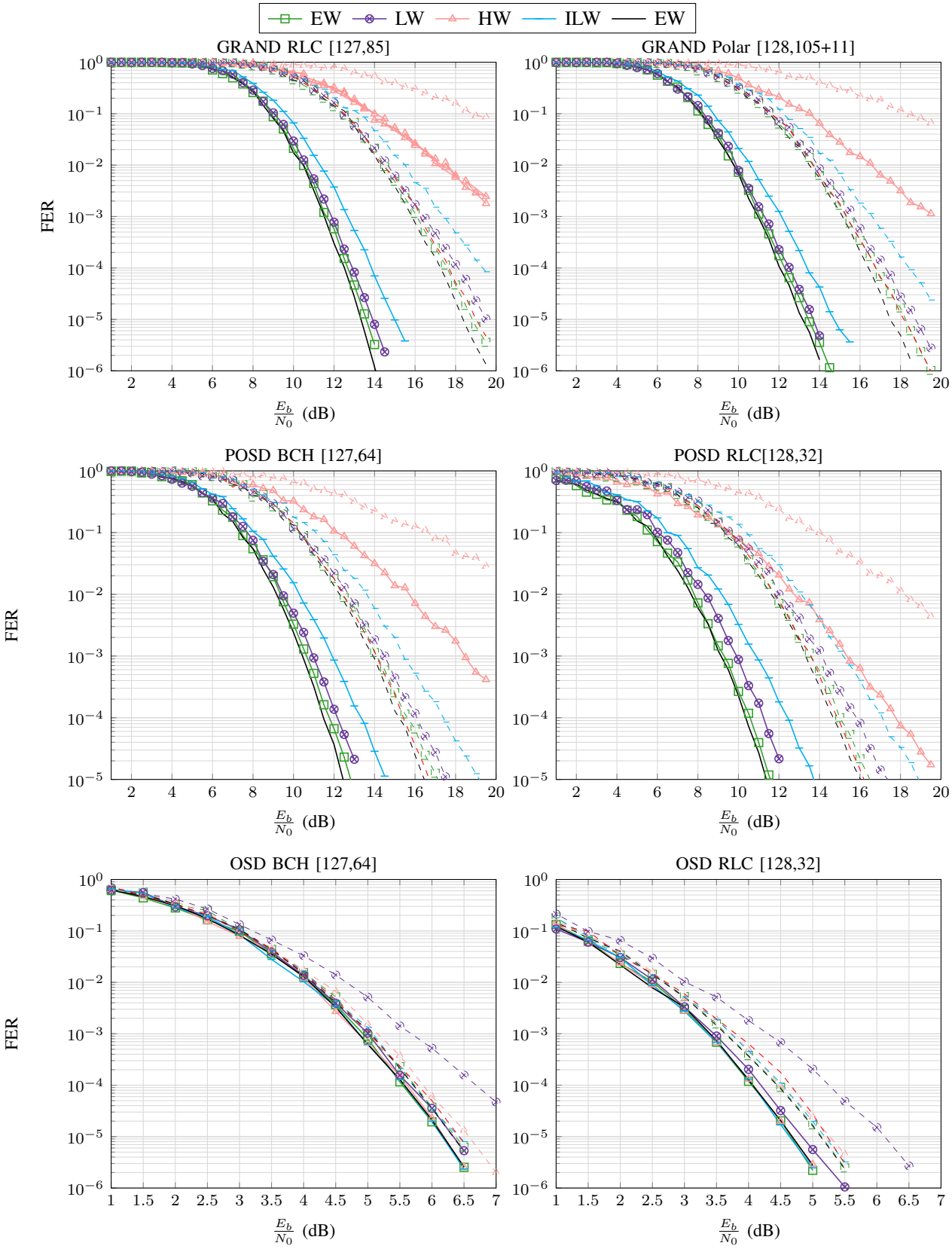

Fig. 10. Comparison of decoding performance of different error patterns with GRAND, POSD and OSD on a Rayleigh fading channel with perfect knowledge of CSI . Straight lines represent simulations run with 10^6, 10^5 and 10^4 maximum number of queries with GRAND, POSD and OSD respectively. Dashed lines represent simulations run with a maximum number of queries of 10^3.

loss in performance at $MQ = 10^5$. LUT TEPs outperform EW TEPs slightly (by less than 0.1 dB) at target FER 10^{-4} with $MQ = 10^3$.

OSD: We simulate OSD on BCH [127,64] and RLC code [128,32]. For a maximum number of queries of 10^4, we can observe that the EW, HW, ML, ILW TEPs result in the same performance. LW TEPs result in a performance loss of ≈ 0.1 dB compared to the EW error patterns at target FER 10^{-5}. For a maximum number of queries of 10^3 we observe that the ILW, ML and EW TEP performance is approximately the same. At target FER 10^{-5} we observe that the HW TEPs attain a 0.1 dB to 0.2 dB loss compared to the ILW, EW and ML. We also observe that the LUT TEPs achieve almost the same performance as the EW and ML TEPs except with RLC [128,32] at target FER 10^{-5} where they ≈ 0.1 dB loss compared to EW and ML TEPs.

B. No Channel Side Information (NCSI)

In the absence of CSI, the LLR of this channel can be approximated as [44]:

$$L = \frac{2}{\sigma^2} y \times E[h] \tag{68}$$

with the PDF of the LLR [44]:

$$f_{\mathbf{L}}(l|c=0) = \frac{\sigma\Delta^2}{2E[h]} e^{\frac{-\Delta^2\sigma^2 l^2}{4E[h]^2}} \times \left[\sqrt{\frac{2}{\pi}} e^{-\frac{\Delta^2 l^2}{8E[h]^2}} + \frac{\Delta l}{E[h]} Q\left(\frac{-\Delta l}{2E[h]}\right) \right], \tag{69}$$

where $\Delta = \sqrt{\frac{\sigma^2}{2\sigma^2+1}},\ E[h] = 0.8862,\ Q(x) = \frac{1}{2} erfc(\frac{x}{\sqrt{2}})$. The same error generation procedure is followed as was done in the case of ideal CSI.

1) Decoding Performance Comparison: Fig. 11 shows the performance of GRAND, POSD and OSD with the proposed EW, and the existing LW, HW, ILW and ML TEPs. We discuss the performance of each of these decoders with different TEPs and varying codes.

GRAND: We simulate GRAND on RLC [127,85] and 5G CA-Polar code [128,105+11]. For a maximum number of queries of 10^6, we can observe that the EW, ML and LW error patterns result in almost the same decoding performance. We can also observe that EW error patterns have a $0.8 \rightarrow 1$ dB gain compared to ILW error patterns at target FER 10^{-5}. HW TEPs do not reach the target FER 10^{-5} within the simulated range of SNRs. For $MQ = 10^3$ queries, the EW, LW, ILW, and HW TEPs exhibit the same ranking (in terms of which performs better than the other) as that observed with $MQ = 10^6$. LUT TEPs perform similarly to the EW TEPs with $MQ = 10^3$, with RLC [127,85] and CA-Polar [128,105+11].

POSD: We simulate POSD on BCH [127,64] and RLC code [128,32]. For a maximum number of queries of 10^5, we can observe that the EW and LW error patterns result in almost the same decoding performance at target FER 10^{-4} with a $0 \rightarrow 0.4$ dB degradation in performance compared to ML TEPs. Additionally, we can observe a $0.6 \rightarrow 0.8$ dB gain using EW TEPs compared to ILW error patterns at target FER 10^{-4}. HW TEPs also achieve poor decoding performance and they achieve at least a 5.5 dB loss in performance at FER 10^{-4} compared to EW TEPs. For $MQ = 10^3$ queries, the EW, LW, and LUT TEPs exhibit the same decoding performance. We can observe that the ML TEPs outperform the EW, LW, and LUT TEPs by at most 0.2 dB at target FER 10^{-4}. Meanwhile ILW TEPs exhibit a performance loss of 1 dB to 1.5 dB compared to the ML TEPs.

OSD: We simulate OSD on BCH [127,64] and RLC code [128,32]. For a maximum number of queries of 10^4, we can observe that the EW, HW, ML, ILW TEPs result in the same performance. LW TEPs result in a performance loss of $1.4 \rightarrow 1.9$ dB compared to the EW error patterns at target FER 10^{-5}. For a maximum number of queries of 10^3 we observe that the LUT, HW, ML and EW TEP performance is approximately the same. At target FER 10^{-5} we observe that the ILW TEPs attain a 0.4 dB to 1 dB loss compared to the LUT, HW, EW and ML.

C. Overlap in Error Patterns with AWGN

Fig. 12 shows the overlap in EW TEPs generated for the AWGN model and the Rayleigh fading models. We can see that the overlap between the EW TEPs produced with Rayleigh fading channel (NCSI) and the AWGN channel is more than $75\% \rightarrow 95\%$ across all MQ. This is due to the fact that the additive noise impacting the Rayleigh fading channel with no CSI is the same as that of the AWGN channel. The Rayleigh fading with ideal CSI has less common TEPs with the AWGN channel with around $40\% \rightarrow 75\%$ overlapping TEPs with AWGN channel. This is due to the fact that the additive noise is modified before decoding, as it is divided by the fading coefficient when the fading coefficient is known. This results in a lower overlap of TEPs than the case of the NCSI fading channel.

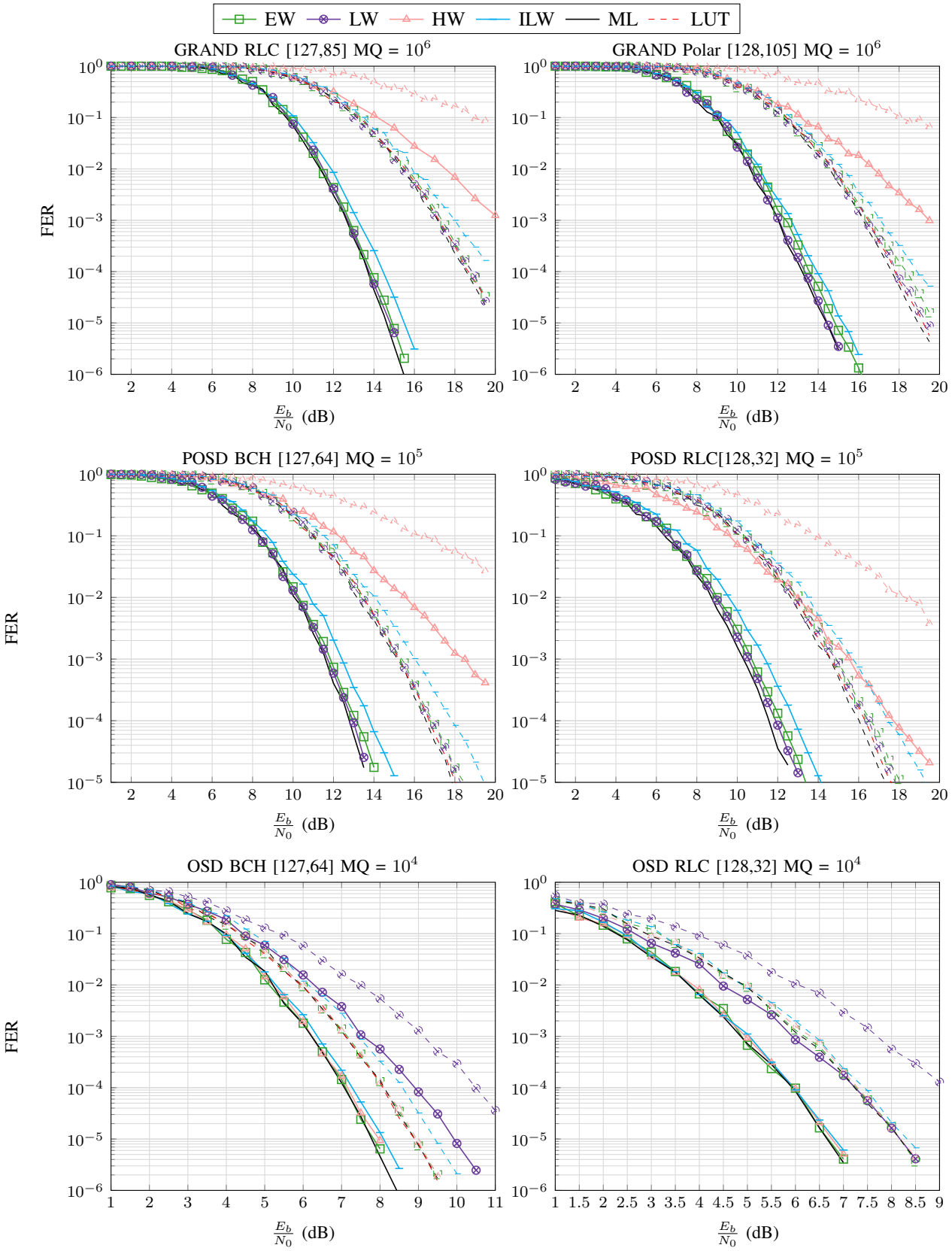

Fig. 11. Comparison of decoding performance of different error patterns with GRAND, OSD and POSD on a Rayleigh fading channel with no perfect knowledge of channel state information. Straight lines represent simulations run with 10^6, 10^5 and 10^4 maximum number of queries with GRAND, POSD and OSD respectively. Dashed lines represent simulations run with a maximum number of queries of 10^3.

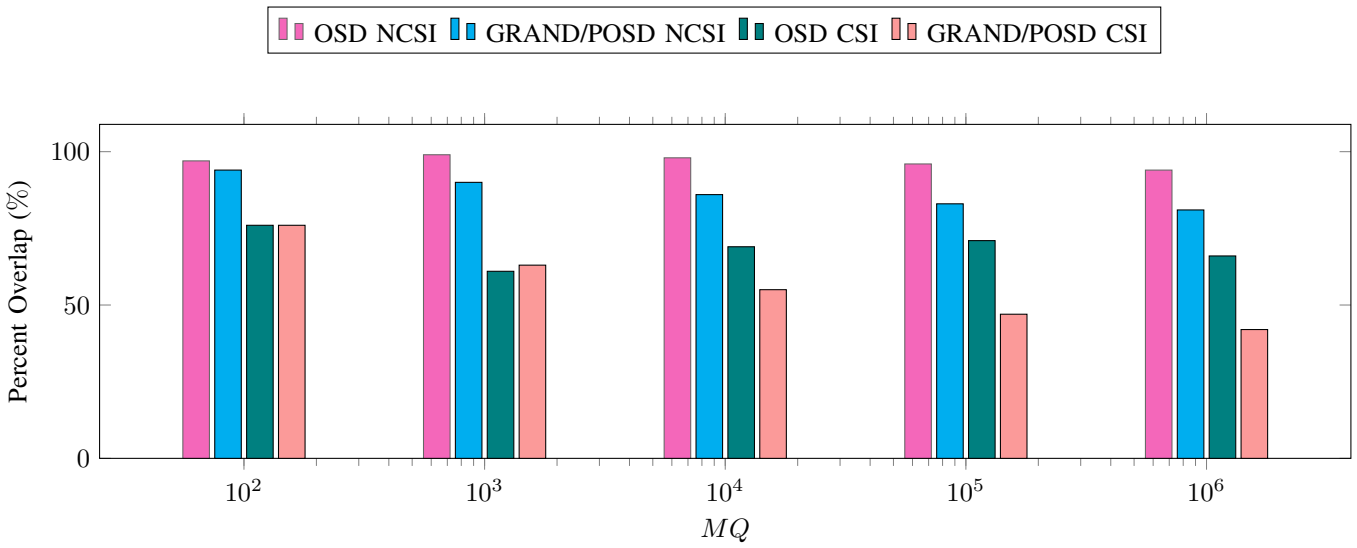

Fig. 12. Percentage of overlap of the EW error patterns generated on an AWGN channel with different decoders and types of fading channels. EW error patterns are generated for code [127,64].

D. Conclusion

Since GRAND, POSD and OSD try to remove the effect of additive noise on the received signal, the error patterns used with each of these decoders with an additive Gaussian noise should not change irrespective whether there is a fading multiplicative gain. We observe that when the noise profile is the same, the EW error patterns are very similar. However, when the additive noise profile is modified, the EW TEPs change to accommodate for a different additive noise profile. We also observe that the EW TEPs can accommodate for Rayleigh fading channels with slight to no degradation in decoding performance compared to the other pre-generated TEPs.

VIII. Conclusion

In this work, we introduced two systematic frameworks that use the channel PDFs to generate channel-adaptive error patterns catered for each of GRAND, POSD and OSD. By ranking test error patterns based on their expected weighted Hamming distance, we provide a systematic alternative to Monte-Carlo simulations which are often unsuitable for high SNR scenarios or hard to simulate channels. Our theoretical framework systematically allows the inclusion of rare, high-weight error patterns that are notoriously difficult to capture via empirical Monte-Carlo methods at high SNRs. This makes the proposed method highly suitable for Ultra-Reliable Low-Latency Communication (URLLC) applications. Simulation results confirm that our proposed error patterns generally match or outperform the current state-of-the-art pre-generated error pattern on memoryless channels, including AWGN, Rayleigh fading and mixture of Gaussian channels. The strong performance of EW patterns on all the different variations of channels and different decoder configurations demonstrate that GRAND, POSD and OSD decoders can be calibrated entirely offline for specific, non-ideal real-world noise environments without requiring Monte-Carlo simulations or complex analytical work. Future work will look to extend our proposed framework to accommodate higher-order modulation schemes and to explore TEP generation for channels with memory.

References

[1] R. A. Fisher, "On the mathematical foundations of theoretical statistics," *Philosophical Transactions of the Royal Society of London, Series A: Containing Papers of a Mathematical or Physical Character*, vol. 222, pp. 309–368, Jan. 1922.
[2] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, July 1948.
[3] E. Berlekamp, R. McEliece, and H. van Tilborg, "On the inherent intractability of certain coding problems (Corresp.)," *IEEE Transactions on Information Theory*, vol. 24, no. 3, pp. 384–386, 1978.
[4] J. M. Wozencraft, "Sequential decoding for reliable communication," *Research Laboratory of Electronics, Massachusetts Institute of Technology*, 1957.
[5] R. Fano, "A heuristic discussion of probabilistic decoding," *IEEE Transactions on Information Theory*, vol. 9, pp. 64–74, Apr. 1963.
[6] J. L. Massey, "Deep-Space Communications and Coding: A Marriage Made in Heaven," *Advanced Methods for Satellite and Deep Space Communications. Heidelberg and New York: Springer*, pp. 1–17, 1992.
[7] J. Heller, "Performance of Pioneer-Type Sequential Decoding Communications Systems with Noisy Oscillators," *Communications Sytems Research: Sequential Decoding*, vol. 3, pp. 37–53, 1968.
[8] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, pp. 260–269, Apr. 1967.
[9] F. Kschischang and V. Sorokine, "On the trellis structure of block codes," *IEEE Transactions on Information Theory*, vol. 41, pp. 1924–1937, Nov. 1995.
[10] G. Battail, "Décodage pondéré optimal des codes linéaires en blocs," *Annales des Télécommunications*, vol. 38, pp. 443–459, Nov. 1983.
[11] G. Battail and J. Fang, "Décodage pondéré optimal des codes linéaires en blocs II. - Analyse et résultats de simulation," *Annales des Télécommunications*, vol. 41, pp. 580–604, Nov. 1986.
[12] G. Battail, "We Can Think of Good Codes, and Even Decode Them," in *Eurocode '92: International Symposium on Coding Theory and Applications* (P. Camion, P. Charpin, and S. Harari, eds.), pp. 353–368, Springer, 1993.

[13] J. Fang, *Décodage pondère des codes en blocs et quelques sujets sur la complexité du décodage*. thesis, Paris, ENST, Jan. 1987.
[14] M. Fossorier and S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics," *IEEE Transactions on Information Theory*, vol. 41, pp. 1379–1396, Sept. 1995.
[15] A. Valembois and M. Fossorier, "Box and match techniques applied to soft-decision decoding," *IEEE Transactions on Information Theory*, vol. 50, pp. 796–810, May 2004.
[16] C. Yue, M. Shirvanimoghaddam, G. Park, O.-S. Park, B. Vucetic, and Y. Li, "Linear-Equation Ordered-Statistics Decoding," *IEEE Transactions on Communications*, vol. 70, pp. 7105–7123, Nov. 2022.
[17] C. Yue, M. Shirvanimoghaddam, Y. Li, and B. Vucetic, "Segmentation-discarding ordered-statistic decoding for linear block codes," in *2019 IEEE Global Communications Conference, GLOBECOM 2019 - Proceedings*, pp. 1–6, 2019.
[18] L. Yang, J. Zhao, X. Li, L. Chen, H. Zhang, and J. Tong, "Efficient Ordered Statistics Decoding of BCH Codes Without Gaussian Elimination," *IEEE Transactions on Information Theory*, vol. 71, pp. 8294–8311, Nov. 2025.
[19] M. Fossorier, M. Shakiba-Herfeh, and H. Zhang, "Modified OSD Algorithm With Reduced Gaussian Elimination," *IEEE Communications Letters*, vol. 28, pp. 1755–1759, Aug. 2024.
[20] A. Valembois and M. Fossorier, "An improved method to compute lists of binary vectors that optimize a given weight function with application to soft-decision decoding," *IEEE Communications Letters*, vol. 5, pp. 456–458, Nov. 2001.
[21] K. R. Duffy and M. Médard, "Guessing random additive noise decoding with soft detection symbol reliability information - SGRAND," in *IEEE International Symposium on Information Theory - Proceedings*, pp. 480–484, 2019.
[22] A. Solomon, K. R. Duffy, and M. Médard, "Soft Maximum Likelihood Decoding using GRAND," in *2020 IEEE International Conference on Communications*, pp. 1–6, 2020.
[23] K. R. Duffy, "Ordered Reliability Bits Guessing Random Additive Noise Decoding," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, (Toronto, ON, Canada), pp. 8268–8272, IEEE, June 2021.
[24] K. R. Duffy, W. An, and M. Médard, "Ordered Reliability Bits Guessing Random Additive Noise Decoding," *IEEE Transactions on Signal Processing*, vol. 70, pp. 4528–4542, 2022.
[25] S. M. Abbas, M. Jalaleddine, C.-Y. Tsui, and W. J. Gross, "Improved Step-GRAND: Low-Latency Soft-Input Guessing Random Additive Noise Decoding," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 33, no. 4, pp. 1028–1041, 2025.
[26] C. Condo, "A fixed latency ORBGRAND decoder architecture with LUT-aided error-pattern scheduling," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 5, pp. 2203–2211, 2022.
[27] S. M. Abbas, M. Jalaleddine, and W. J. Gross, *Guessing Random Additive Noise Decoding: A Hardware Perspective*. Springer International, 2023.
[28] A. Riaz, A. Yasar, F. Ercan, W. An, J. Ngo, K. Galligan, M. Medard, K. R. Duffy, and R. T. Yazicigil, "A Sub-0.8pJ/b 16.3Gbps/mm2 Universal Soft-Detection Decoder Using ORBGRAND in 40nm CMOS," in *IEEE International Solid- State Circuits Conference*, pp. 432–434, 2023.
[29] S. M. Abbas, T. Tonnellier, F. Ercan, M. Jalaleddine, and W. J. Gross, "High-Throughput and Energy-Efficient VLSI Architecture for Ordered Reliability Bits GRAND," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 30, pp. 681–693, June 2022.
[30] S. Alnawayseh and P. Loskot, "Ordered statistics-based list decoding techniques for linear binary block codes," *EURASIP Journal on Wireless Communications and Networking*, no. 1, p. 314, 2012.
[31] S. Alnawayseh and P. Loskot, "Low-complexity soft-decision decoding techniques for linear binary block codes," in *2009 International Conference on Wireless Communications Signal Processing*, pp. 1–5, 2009.
[32] M. Jalaleddine, H. Zhou, J. Li, and W. J. Gross, "Partial Ordered Statistics Decoding with Enhanced Error Patterns," in *IEEE International Symposium on Information Theory*, 2023.
[33] C. Condo, V. Bioglio, and I. Land, "High performance low-complexity error pattern generation for ORBGRAND decoding," in *2021 IEEE Globecom Workshops*, pp. 1–6, 2021.
[34] X. Ma, "Guessing what, Noise or Codeword?," in *2024 IEEE Information Theory Workshop (ITW)*, pp. 460–465, Nov. 2024.
[35] A. Valembois and M. Fossorier, "A Comparison between "Most-Reliable-Basis Reprocessing" Strategies," *IEICE Trans. Fundamentals, A*, vol. 85, pp. 1727–1741, July 2002.
[36] K. R. Duffy, J. Li, and M. Médard, "Capacity-Achieving Guessing Random Additive Noise Decoding," *IEEE Transactions on Information Theory*, vol. 65, pp. 4023–4040, July 2019.
[37] Y. Wu and C. N. Hadjicostis, "Soft-decision decoding using ordered recodings on the most reliable basis," *IEEE Transactions on Information Theory*, vol. 53, no. 2, pp. 829–836, 2007.
[38] Y. Wu and C. N. Hadjicostis, "Soft-decision decoding of linear block codes using preprocessing and diversification," *IEEE Transactions on Information Theory*, vol. 53, no. 1, pp. 378–393, 2007.
[39] S. M. Abbas, M. Jalaleddine, and W. J. Gross, "Hardware Architecture for Guessing Random Additive Noise Decoding Markov Order (GRAND-MO)," *Journal of Signal Processing Systems*, May 2022.
[40] C. Yue, M. Shirvanimoghaddam, B. Vucetic, and Y. Li, "A revisit to ordered statistic decoding: distance distribution and decoding rules," *IEEE Transactions on Information Theory*, 2021.
[41] D. Feng, L. Lai, J. Luo, Y. Zhong, C. Zheng, and K. Ying, "Ultra-reliable and low-latency communications: applications, opportunities and challenges," *Science China Information Sciences*, vol. 64, p. 120301, Jan. 2021.
[42] G. Hampel, C. Li, and J. Li, "5G Ultra-Reliable Low-Latency Communications in Factory Automation Leveraging Licensed and Unlicensed Bands," *IEEE Communications Magazine*, vol. 57, pp. 117–123, May 2019. IEEE Communications Magazine.
[43] George Casella and Roger Berger, *Statistical Inference*, vol. Second edition of *Texts in Statistical Science Series*. Chapman and Hall/CRC, 2024.
[44] J. Hou, P. Siegel, and L. Milstein, "Performance analysis and code optimization of low density parity-check codes on Rayleigh fading channels," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 5, pp. 924–934, 2001.
[45] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. Cambridge, Massachusetts: The MIT Press, 2016.
[46] D. A. Le, H. V. Vu, N. H. Tran, M. C. Gursoy, and T. Le-Ngoc, "Approximation of Achievable Rates in Additive Gaussian Mixture Noise Channels," *IEEE Transactions on Communications*, vol. 64, no. 12, pp. 5011–5024, 2016.
[47] T. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Transactions on Information Theory*, vol. 47, pp. 599–618, Feb. 2001.

APPENDIX A
EQUIVALENCE OF THE GENERALIZED WEIGHTS TO THE WEIGHTED HAMMING DISTANCE

We note that the notations used in this section are defined in Section II. Definition 5 is restated as it is relevant for the proof. The weighted Hamming distance is defined as:

$$w_H(\mathbf{x}, \mathbf{L}) = \sum_{i=1}^{N} (\mathbf{c}[i] \oplus \theta(\mathbf{L}[i])) \times |\mathbf{L}[i]|. \tag{70}$$

While [12] has the same algorithm as [34], the metric used is the generalized distance defined as:

$$\eta(\mathbf{y}, \mathbf{c}) = \sum_{j=1}^{n} v(y[j], c[j]), \tag{71}$$

where:

$$v(y,c) = ln\left(\frac{P(y \mid \theta(L))}{P(y \mid c)}\right). \tag{72}$$

For BPSK with bits 0 and 1 equiprobable, we will prove that the weighted Hamming distance is equivalent to the generalized distance metric through enumerating all possible c and $\theta(L)$:

1) When the assumed codeword bit is 0, and the received LLR is positive ($\theta(L) \oplus c = 0$), the following equality holds:

$$v(y,0) = \ln\left(\frac{P(y|0)}{P(y|0)}\right) = 0. \tag{73}$$

2) When the assumed codeword bit is 0, and the received LLR is negative ($\theta(L) \oplus c = 1$), the following equality holds:

$$v(y,0) = \ln\left(\frac{P(y|1)}{P(y|0)}\right) = -L = |L|. \tag{74}$$

3) When the assumed codeword bit is 1, and the received LLR is positive ($\theta(L) \oplus c = 1$), the following equality holds:

$$v(y,1) = \ln\left(\frac{P(y|0)}{P(y|1)}\right) = L = |L|. \tag{75}$$

4) When the assumed codeword bit is 1, and the received LLR is negative ($\theta(L) \oplus c = 0$), the following equality holds:

$$v(y,1) = \ln\left(\frac{P(y|1)}{P(y|1)}\right) = 0. \tag{76}$$

Hence the value $v(y,c) = (c \oplus \theta(L)) \times |L|$ and $\eta(\mathbf{y},\mathbf{c}) = w_H(\mathbf{c},\mathbf{L})$ for the BPSK case.