BCH码的低时延分阶统计译码

Low-Latency Ordered Statistics Decoding of BCH Codes

- 陈立 教授
- 中山大学 电子与信息工程学院

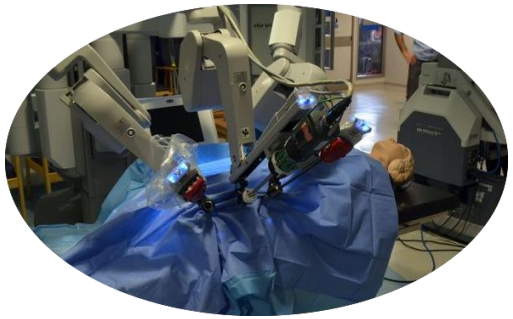
Outline

- Background
- BCH Codes
- Ordered Statistics Decoding
- Low-Latency OSD
- Hybrid Soft Decoding

Background

- Future scenarios featured by URLLC

Unmanned Driving



Wise Information Technology of Med (WITMED)

Factory Automation

Extended Reality (XR)

→ Short-to-medium length codes

Background

- Capacity-approaching codes

Turbo codes **LDPC codes** **Polar codes**

n is large → randomness & iterative decoding, channel polarization

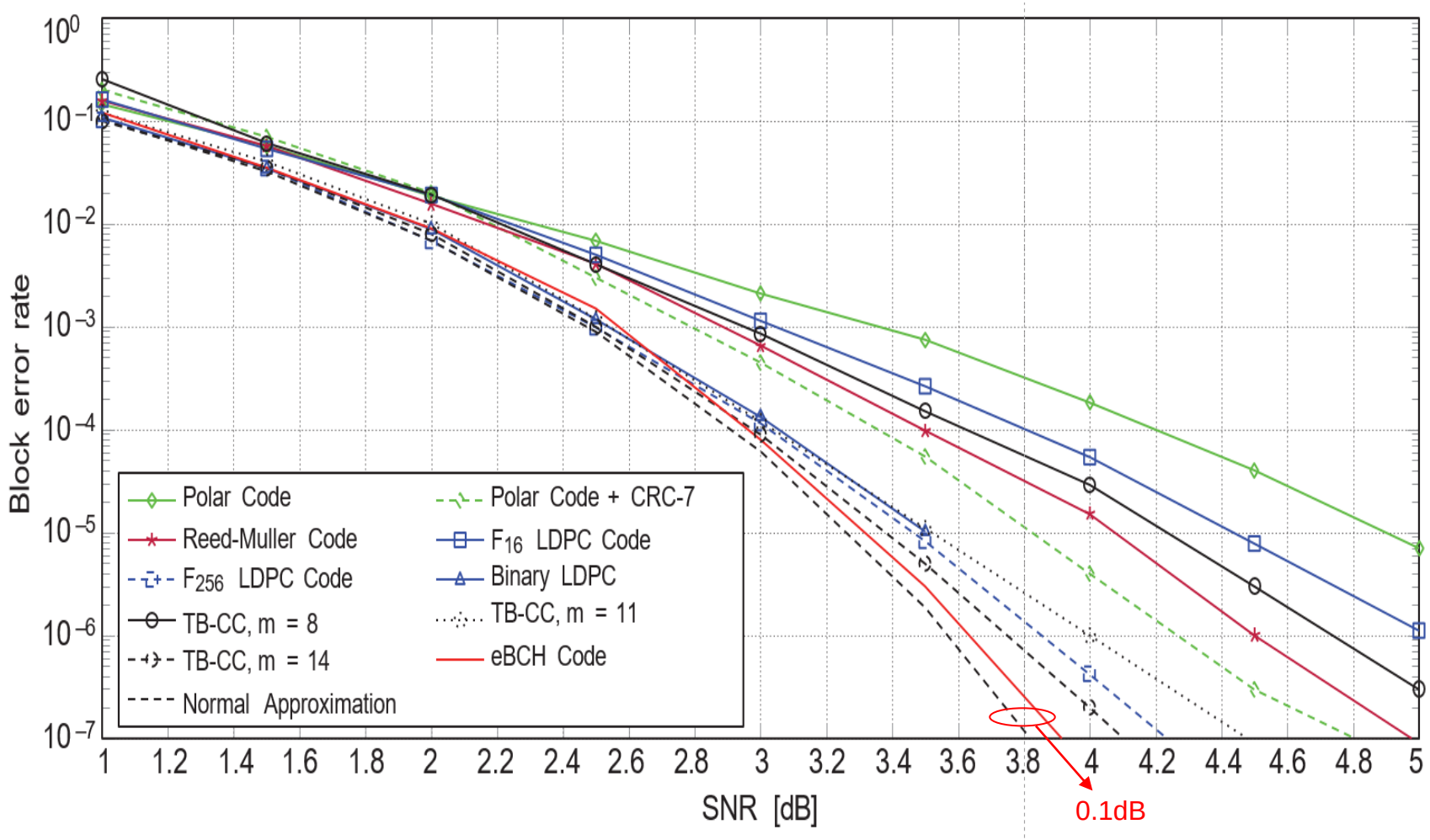
- Good short-to-medium length codes

BCH codes (OSD) **TBCC (Circular Viterbi)** **Polar codes (SCL)** **PAC codes (Fano)**

n is short-to-medium → near ML decoding, list decoding

Background

- Near ML decoding performance: rate R = 1/2 & length n = 128 [1]



[1] M. Shirvanimoghaddam *et al*., "Short block-length codes for ultrareliable low latency communications," *IEEE Commun. Mag*., vol. 57, no. 2, pp. 130–137, Feb. 2019.

BCH Codes

- Encoding of BCH codes

(n, k) binary primitive BCH code $\mathcal{C}_{\mathrm{BCH}}[n, k]$

- Primitive element of $\mathbb{F}_{2^m}$: σ
- Codeword length: $n = 2^m - 1$
- Designed distance: $d = 2t + 1$

  - Generator polynomial: $g(x) = g_0 + g_1 x + \cdots + g_{n-k} x^{n-k} \in \mathbb{F}_2[x]$ is the minimal $(\deg g(x))$ nonzero polynomial such that

$$g(\sigma) = g(\sigma^2) = \cdots = g(\sigma^{2t}) = 0$$

  - Message $\underline{\boldsymbol{f}} = (f_0, f_1, \ldots, f_{k-1}) \in \mathbb{F}_2^k$, in poly. $f(x) = f_0 + f_1 x + \cdots + f_{k-1} x^{k-1} \in \mathbb{F}_2[x]$

    Codeword $\underline{\boldsymbol{c}} = (c_0, c_1, \ldots, c_{n-1}) \in \mathbb{F}_2^n$, in poly. $c(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1} = f(x)g(x) \in \mathbb{F}_2[x]$

    Parity-check condition: $c(\sigma) = c(\sigma^2) = \cdots = c(\sigma^{2t}) = 0$

BCH Codes

- Encoding of BCH codes
  - Generator matrix:

$$\mathbf{G} = \begin{bmatrix} g_0 & g_1 & \cdots & g_{n-k} & 0 & \cdots & 0 \\ 0 & g_0 & \cdots & \cdots & g_{n-k} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & g_0 & g_1 & \cdots & g_{n-k} \end{bmatrix} \in \mathbb{F}_2^{k \times n}$$

Encoding process: $\underline{\boldsymbol{c}} = \underline{\boldsymbol{f}} \cdot \mathbf{G}$

  - Parity-check matrix:

$$\mathbf{H} = \begin{bmatrix} (\sigma)^0 & (\sigma)^1 & \cdots & (\sigma)^{n-1} \\ (\sigma^2)^0 & (\sigma^2)^1 & \cdots & (\sigma^2)^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ (\sigma^{2t})^0 & (\sigma^{2t})^1 & \cdots & (\sigma^{2t})^{n-1} \end{bmatrix} \in \mathbb{F}_{2^m}^{2t \times n}$$

Parity-check matrix of the $(n, n-2t)$ RS code

Parity-check equation:

$$c(\sigma) = c(\sigma^2) = \cdots = c(\sigma^{2t}) = 0 \Leftrightarrow \underline{\boldsymbol{c}} \cdot \mathbf{H}^{\mathrm{T}} = \underline{\mathbf{0}}$$

BCH Codes

- Decoding of BCH codes
  - Hard-decision decoding (utilizing algebraic structure)

Berlekamp-Massey (BM)

Euclidean

Guruswami-Sudan (GS)

→

- Efficient – algebra fluency & hardware friendly
- Limited performance

  - Soft-decision decoding (utilizing soft information)

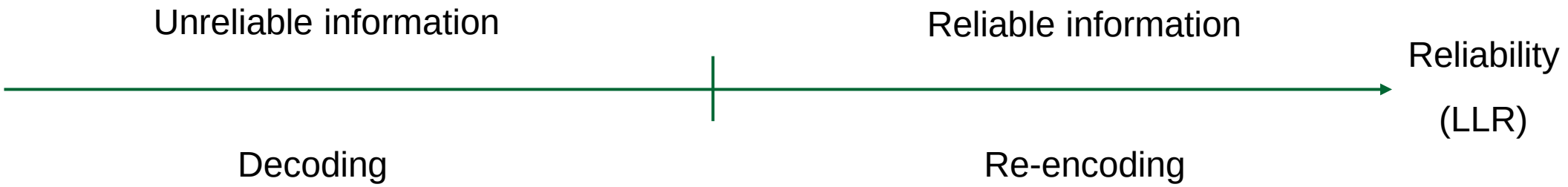
Ordered Statistics Decoding (OSD)

Algebraic Chase decoding (Chase-BM, Chase-GS)

→

- High performance (~ ML)
- High complexity & latency

Soft-Decision Decoding



Decoding

- Algebraic structure
- Equation system

(Syndrome-based or interpolation-based)

Re-encoding

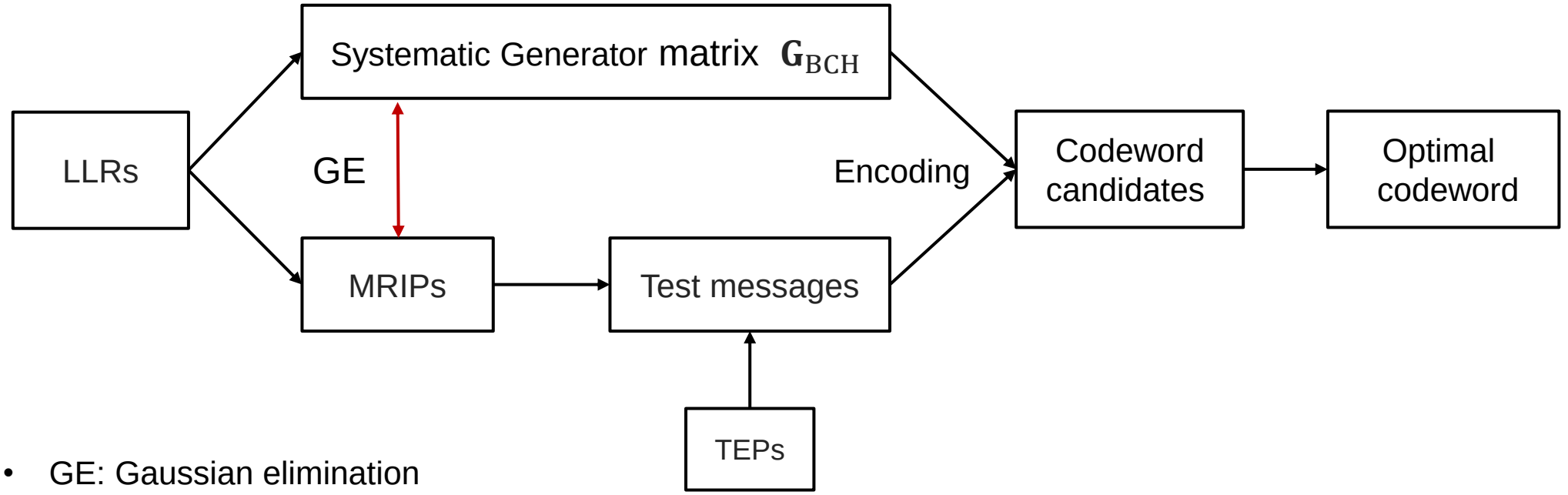
- Information set
- Systematic generator matrix

Low-latency OSD: use $\mathbf{G}_{\mathrm{RS}}$ to replace $\mathbf{G}_{\mathrm{BCH}}$

Hybrid soft decoding: use algebraic Chase decoding to identify test error patterns (TEPs)

Ordered Statistics Decoding

- Overview of OSD



- GE: Gaussian elimination
- LLRs: log likelihood ratios
- MRIPs: most reliable independent positions
- TEPs: test error patterns

Ordered Statistics Decoding

- Construct the systematic generator matrix
  - Channel
    - Received LLR sequence: $\underline{\boldsymbol{L}} = (L_0, L_1, \ldots, L_{n-1}) \in \mathbb{R}^n$
    - Hard-decision received word: $\underline{\boldsymbol{r}} = (r_0, r_1, \ldots, r_{n-1}) \in \mathbb{F}_2^n$
  - Sort based on reliability and determine the most reliable independent positions (MRIPs)

$$\underline{\boldsymbol{r}}' = \Lambda(\underline{\boldsymbol{r}}) = (r_{j_0}, r_{j_1}, \ldots, r_{j_{n-1}}) \qquad |L_{j_0}| \geq |L_{j_1}| \geq \cdots \geq |L_{j_{n-1}}|$$

  - Construct the systematic generator matrix $\mathbf{G}_{\mathrm{BCH}}$

$$\mathbf{G} \longrightarrow \mathbf{G}' = \Lambda(\mathbf{G}) \longrightarrow \mathbf{G}_{\mathrm{BCH}}$$

Gaussian elimination

Ordered Statistics Decoding

- Generate BCH candidate codewords
  - Re-encoding process

Initial message: $\underline{\boldsymbol{f}}^{(0)} = (r_{j_0},\ r_{j_1}, \dots,\ r_{j_{k-1}}) \in \mathbb{F}_2^k$

$\boldsymbol{+}$

Test error pattern: $\underline{\boldsymbol{e}}^{(\omega)} = (e_{j_0}^{(\omega)}, e_{j_1}^{(\omega)}, \dots, e_{j_{k-1}}^{(\omega)}) \in \mathbb{F}_2^k \qquad \omega = 0,1, \dots, N_{\text{TEPs}} - 1$

$\boldsymbol{=}$

Test message: $\underline{\boldsymbol{f}}^{(\omega)} = (f_{j_0}^{(\omega)}, f_{j_1}^{(\omega)}, \dots, f_{j_{k-1}}^{(\omega)}) \in \mathbb{F}_2^k$

BCH codeword candidate: $\hat{\underline{\boldsymbol{c}}}^{(\omega)} = (\hat{c}_0^{(\omega)}, \hat{c}_1^{(\omega)}, \dots, \hat{c}_{n-1}^{(\omega)}) = \Lambda^{-1}(\underline{\boldsymbol{f}}^{(\omega)} \cdot \mathbf{G}_{\text{BCH}}) \in \mathbb{F}_2^n$

  - Number of TEPs (= number of BCH codeword candidates)

OSD with order τ ➡ $d_{\text{H}}(\underline{\boldsymbol{e}}^{(\omega)}, \underline{\mathbf{0}}) \le \tau$

$$N_{\text{TEPs}} = \binom{k}{0} + \binom{k}{1} + \cdots + \binom{k}{\tau}$$

Number of TEPs $\underline{\boldsymbol{e}}^{(\omega)}$ s.t. $d_{\text{H}}(\underline{\boldsymbol{e}}^{(\omega)}, \underline{\mathbf{0}}) = 1$

Repetitive processing

(parallel)

Ordered Statistics Decoding

- Challenges of OSD
  - Exponential complexity
  - τ–OSD complexity: $O(k^{\tau})$

  → Reduction of TEPs: skipping rule; stopping rule
  → Trade off with performance: multiple bases; validation rule
  → Trade off with storage: box-and-match algorithm

  - Decoding latency bottleneck
  - GE

  → Offline computed systematic generator matrices → Only an auxiliary method
  → Alternative solution? Utilize algebra of the code

Low-Latency OSD

- BCH codes and RS codes
  - **Subfield subcode**

    Given two linear codes $\mathcal{C} \subset \mathbb{F}_2^n$ and $\mathcal{C}' \subset \mathbb{F}_{2^m}^n$, if $\mathcal{C} = \mathcal{C}' \cap \mathbb{F}_2^n$, $\mathcal{C}$ is called the subfield subcode of $\mathcal{C}'$ over $\mathbb{F}_2$.

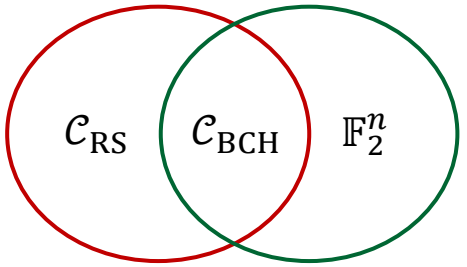
  - **Lemma 1**

    An (n, k) t-error-correcting BCH code defined over $\mathbb{F}_2$ is a subfield subcode of an (n, k') t-error-correcting RS code defined over $\mathbb{F}_{2^m}$. i.e., $\mathcal{C}_{\mathrm{BCH}}[n, k, 2t+1] = \mathcal{C}_{\mathrm{RS}}[n, k', 2t+1] \cap \mathbb{F}_2^n$

    BCH code $d = 2t + 1 < n - k + 1$

    RS code $d = 2t + 1 = n - k' + 1$ (MDS)

    ➡ Dimension: $k < k'$

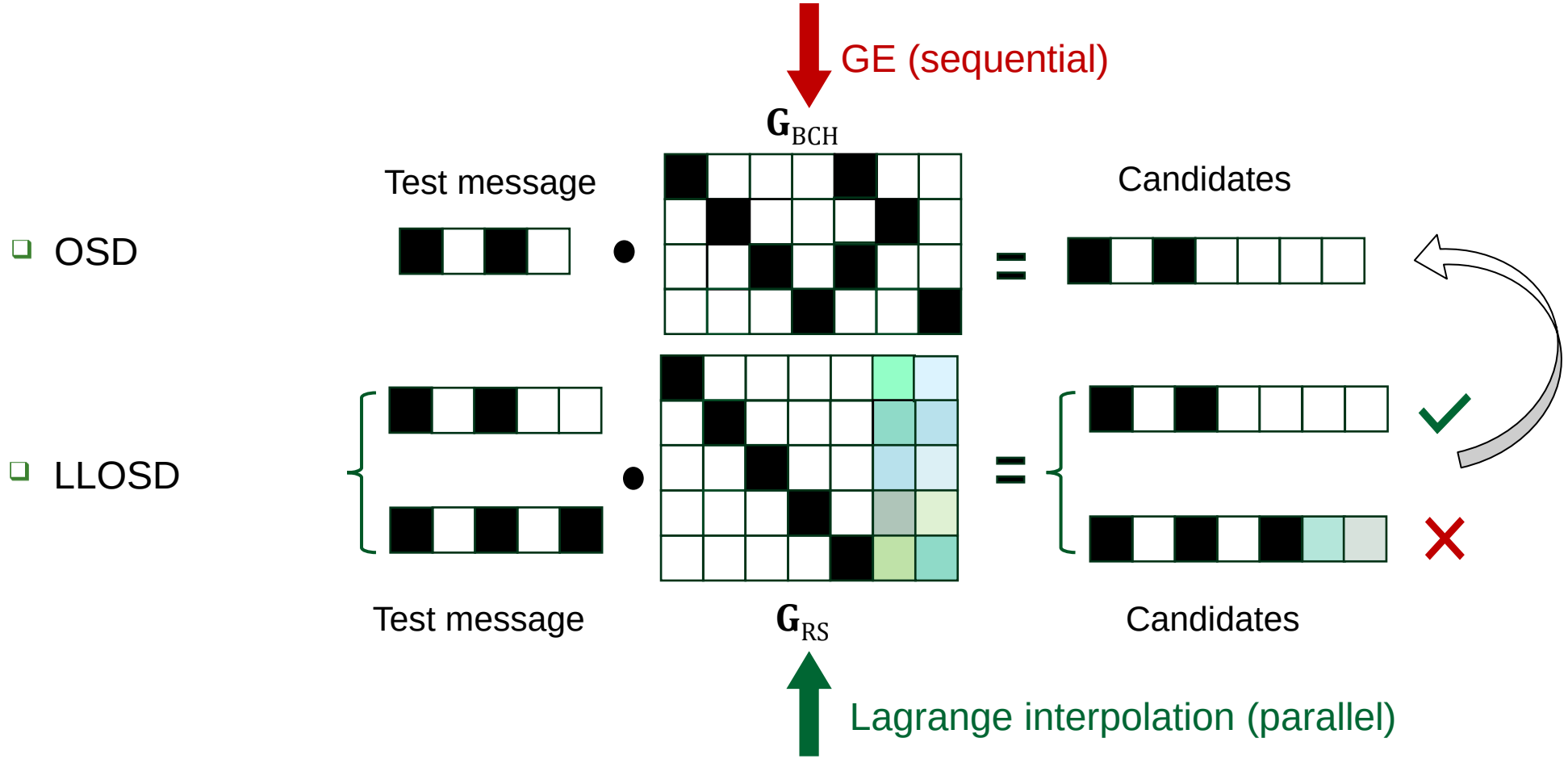
    

  - ***Example 1***

    The $(7, 4)$ BCH code is the binary subcode of the $(7, 5)$ RS code

Low-Latency OSD

- Basic idea of low-latency OSD (LLOSD)



Low-Latency OSD

- Generation of $\mathbf{G}_{RS}$
  - Encoding of an (n, k') RS code

    Message $\underline{\boldsymbol{u}} = (u_0, u_1, \ldots, u_{k'-1}) \in \mathbb{F}_{2^m}^{k'}$, in poly. $u(x) = u_0 + u_1 x + \cdots + u_{k'-1} x^{k'-1} \in \mathbb{F}_{2^m}[x]$

    Codeword $\underline{\boldsymbol{v}} = (u(\alpha_0), u(\alpha_1), \ldots, u(\alpha_{n-1})) \in \mathbb{F}_{2^m}^{n}$

    $v_0 \quad v_1 \quad \ldots \quad v_{n-1}$

    $\alpha_0, \alpha_1, \ldots, \alpha_{n-1} \in \mathbb{F}_{2^m} \backslash \{0\}$ are the code locators

- ***Example 2*** For a (7, 5) RS code over $\mathbb{F}_8$

  Message $\underline{\boldsymbol{u}} = (2, 3, 5, 0, 1)$, in poly. $u(x) = 2 + 3x + 5x^2 + x^4$

  Codeword $\underline{\boldsymbol{v}} = (u(1), u(2), u(4), u(3), u(6), u(7), u(5)) = (5, 0, 4, 7, 6, 1, 3)$

  - $\mathbb{F}_8$ is an extension field of $\mathbb{F}_2$, defined by $p(x) = x^3 + x + 1$
  - In $\mathbb{F}_8$, $\{0, \sigma^0, \sigma^1, \sigma^2, \sigma^3, \sigma^4, \sigma^5, \sigma^6\} = \{0, 1, 2, 4, 3, 6, 7, 5\}$

Low-Latency OSD

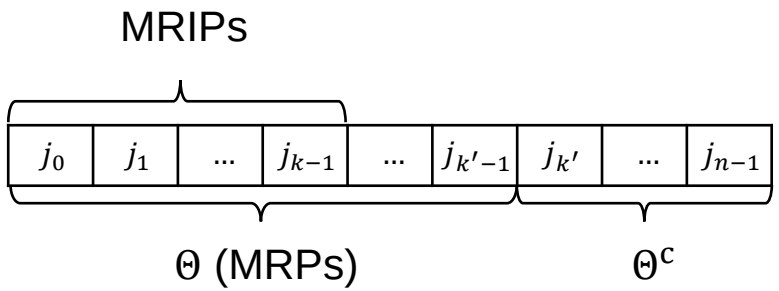
- Generation of $\mathbf{G}_{RS}$
  - Determine the MRIPs

Permuted received word: $\underline{\boldsymbol{r}}' = \Lambda(\underline{\boldsymbol{r}}) = (r_{j_0}, r_{j_1}, \ldots, r_{j_{n-1}})$ ⟵ $|L_{j_0}| \geq |L_{j_1}| \geq \cdots \geq |L_{j_{n-1}}|$

Most reliable positions (MRPs):

$$\Theta = \{j_0, j_1, \ldots, j_{k'-1}\}$$

$$\Theta^c = \{j_{k'}, j_{k'+1}, \ldots, j_{n-1}\}$$



MDS property → MRPs are linearly independent

For RS codes: MRPs ⟺ MRIPs

Low-Latency OSD

- Generation of $\mathbf{G}_{RS}$

Message $\underline{\boldsymbol{u}} = (r_{j_0}, r_{j_1}, \ldots, r_{j_{k'-1}}) \in \mathbb{F}_{2^m}^{k'}$, defined by $\Theta = \{j_0, j_1, \ldots, j_{k'-1}\}$

- Construct the Lagrange interpolation polynomials

$$T_j(x) = \prod_{j' \in \Theta, j' \neq j} \frac{x - \alpha_{j'}}{\alpha_j - \alpha_{j'}}$$

where $T_j(\alpha_j) = 1$, and $T_j(\alpha_{j'}) = 0$ if $j' \in \Theta$ and $j' \neq j$.

- Form the systematic message polynomial of $\underline{\boldsymbol{u}} = (r_{j_0}, r_{j_1}, \ldots, r_{j_{k'-1}})$

$$\mathcal{H}_{\underline{\boldsymbol{u}}}(x) = \sum_{j \in \Theta} r_j T_j(x)$$

$\mathcal{H}_{\underline{\boldsymbol{u}}}(\alpha_j)$ generates
- systematic message symbols r_j, if $j \in \Theta$;
- parity-check symbols, if $j \in \Theta^c$.

Low-Latency OSD

- ***Example 3***

Given a $(7,4)$ BCH code and an LLR sequence $\underline{\boldsymbol{L}} = (-2.447, 5.115, -4.771, -1.349, -7.096, 0.443, -3.485)$

- Mother code: $(7,5)$ RS code
- Hard-decision received word: $\underline{\boldsymbol{r}} = (1,0,1,1,1,0,1)$
- MRPs: $\Theta = \{4,1,2,6,0\}$
- Lagrange interpolation polynomials:

$$T_4(x) = \frac{x-\alpha_1}{\alpha_4-\alpha_1}\cdot\frac{x-\alpha_2}{\alpha_4-\alpha_2}\cdot\frac{x-\alpha_6}{\alpha_4-\alpha_6}\cdot\frac{x-\alpha_0}{\alpha_4-\alpha_0}$$

$$T_2(x) = \frac{x-\alpha_4}{\alpha_2-\alpha_4}\cdot\frac{x-\alpha_1}{\alpha_1-\alpha_2}\cdot\frac{x-\alpha_6}{\alpha_2-\alpha_6}\cdot\frac{x-\alpha_0}{\alpha_2-\alpha_0}$$

$$T_0(x) = \frac{x-\alpha_4}{\alpha_0-\alpha_4}\cdot\frac{x-\alpha_1}{\alpha_0-\alpha_1}\cdot\frac{x-\alpha_2}{\alpha_0-\alpha_2}\cdot\frac{x-\alpha_6}{\alpha_0-\alpha_6}$$

$$T_1(x) = \frac{x-\alpha_4}{\alpha_1-\alpha_4}\cdot\frac{x-\alpha_2}{\alpha_1-\alpha_2}\cdot\frac{x-\alpha_6}{\alpha_1-\alpha_6}\cdot\frac{x-\alpha_0}{\alpha_1-\alpha_0}$$

$$T_6(x) = \frac{x-\alpha_4}{\alpha_6-\alpha_4}\cdot\frac{x-\alpha_1}{\alpha_6-\alpha_1}\cdot\frac{x-\alpha_2}{\alpha_6-\alpha_2}\cdot\frac{x-\alpha_0}{\alpha_6-\alpha_0}$$

Low-Latency OSD

- Generation of $\mathbf{G}_{\mathrm{RS}}$

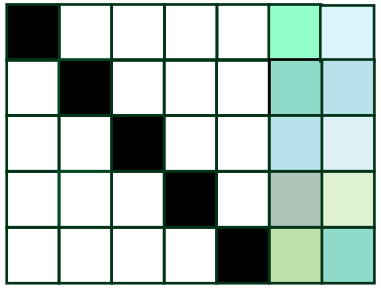
$$\left.\begin{aligned} \underline{\boldsymbol{u}}_{j_0} &= (1,0,0,\ldots,0) \\ \underline{\boldsymbol{u}}_{j_1} &= (0,1,0,\ldots,0) \\ &\vdots \\ \underline{\boldsymbol{u}}_{j_{k'-1}} &= (0,0,0,\ldots,1) \end{aligned}\right\} \longrightarrow \mathbf{G}_{\mathrm{RS}} = \begin{bmatrix} \text{codeword of } \underline{\boldsymbol{u}}_{j_0} \\ \text{codeword of } \underline{\boldsymbol{u}}_{j_1} \\ \vdots \\ \text{codeword of } \underline{\boldsymbol{u}}_{j_{k'-1}} \end{bmatrix}$$

$$\longrightarrow \Lambda(\mathbf{G}_{\mathrm{RS}}) = \begin{bmatrix} 1 & 0 & \cdots & 0 & \mathcal{H}_{\underline{\boldsymbol{u}}_{j_0}}(\alpha_{j_{k'}}) & \cdots & \mathcal{H}_{\underline{\boldsymbol{u}}_{j_0}}(\alpha_{j_{n-1}}) \\ 0 & 1 & \cdots & 0 & \mathcal{H}_{\underline{\boldsymbol{u}}_{j_1}}(\alpha_{j_{k'}}) & \cdots & \mathcal{H}_{\underline{\boldsymbol{u}}_{j_1}}(\alpha_{j_{n-1}}) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & \mathcal{H}_{\underline{\boldsymbol{u}}_{j_{k'-1}}}(\alpha_{j_{k'}}) & \cdots & \mathcal{H}_{\underline{\boldsymbol{u}}_{j_{k'-1}}}(\alpha_{j_{n-1}}) \end{bmatrix}$$

$k' \times k'$ identity submatrix

α_{j_0} α_{j_1} … $\alpha_{j_{k'-1}}$ $\alpha_{j_{k'}}$ … $\alpha_{j_{n-1}}$

Θ (MRPs) Θ^c



Low-Latency OSD

- Generation of $\mathbf{G}_{\mathrm{RS}}$
  - The row-i column-j entry of $\mathbf{G}_{\mathrm{RS}}$:

$$\mathcal{H}_{\underline{\boldsymbol{u}}_i}(\alpha_j) = \frac{\prod_{j' \in \Theta} \left(\alpha_j - \alpha_{j'}\right)}{\left(\alpha_j - \alpha_i\right) \prod_{j' \in \Theta, j' \neq i} \left(\alpha_i - \alpha_{j'}\right)} \qquad \prod_{j=0}^{n-1} \alpha_j = 1 \qquad \mathcal{H}_{\underline{\boldsymbol{u}}_i}(\alpha_j) = \frac{\alpha_i \prod_{j' \in \Theta^{\mathrm{c}}, j' \neq j} \left(\alpha_i - \alpha_{j'}\right)}{\alpha_j \prod_{j' \in \Theta^{\mathrm{c}}, j' \neq j} \left(\alpha_j - \alpha_{j'}\right)}$$

  where $j \in \Theta^{\mathrm{c}}$

$$|\Theta| = k' \qquad |\Theta^{\mathrm{c}}| = n - k'$$

  - Relationship: systematic generator matrix & systematic parity-check matrix
  - Complexity of the generation of $\mathbf{G}_{\mathrm{RS}}$: $2n \cdot \min\{n - k', k'\}$

Low-Latency OSD

- Generation of $\mathbf{G}_{RS}$
  - ***Example 4*** continues from ***Example 3*** with MRPs being $\Theta = \{4, 1, 2, 6, 0\}$

$$\underline{\boldsymbol{u}}_4 = (1,0,0,0,0) \longrightarrow \mathcal{H}_{\underline{\boldsymbol{u}}_4}(x) = \frac{x-\alpha_1}{\alpha_4-\alpha_1}\cdot\frac{x-\alpha_2}{\alpha_4-\alpha_2}\cdot\frac{x-\alpha_6}{\alpha_4-\alpha_6}\cdot\frac{x-\alpha_0}{\alpha_4-\alpha_0}$$

$$\underline{\boldsymbol{u}}_1 = (0,1,0,0,0) \longrightarrow \mathcal{H}_{\underline{\boldsymbol{u}}_1}(x) = \frac{x-\alpha_4}{\alpha_1-\alpha_4}\cdot\frac{x-\alpha_2}{\alpha_1-\alpha_2}\cdot\frac{x-\alpha_6}{\alpha_1-\alpha_6}\cdot\frac{x-\alpha_0}{\alpha_1-\alpha_0}$$

$$\underline{\boldsymbol{u}}_2 = (0,0,1,0,0) \longrightarrow \mathcal{H}_{\underline{\boldsymbol{u}}_2}(x) = \frac{x-\alpha_4}{\alpha_2-\alpha_4}\cdot\frac{x-\alpha_1}{\alpha_1-\alpha_2}\cdot\frac{x-\alpha_6}{\alpha_2-\alpha_6}\cdot\frac{x-\alpha_0}{\alpha_2-\alpha_0}$$

$$\underline{\boldsymbol{u}}_6 = (0,0,0,1,0) \longrightarrow \mathcal{H}_{\underline{\boldsymbol{u}}_6}(x) = \frac{x-\alpha_4}{\alpha_6-\alpha_4}\cdot\frac{x-\alpha_1}{\alpha_6-\alpha_1}\cdot\frac{x-\alpha_2}{\alpha_6-\alpha_2}\cdot\frac{x-\alpha_0}{\alpha_6-\alpha_0}$$

$$\underline{\boldsymbol{u}}_0 = (0,0,0,0,1) \longrightarrow \mathcal{H}_{\underline{\boldsymbol{u}}_0}(x) = \frac{x-\alpha_4}{\alpha_0-\alpha_4}\cdot\frac{x-\alpha_1}{\alpha_0-\alpha_1}\cdot\frac{x-\alpha_2}{\alpha_0-\alpha_2}\cdot\frac{x-\alpha_6}{\alpha_0-\alpha_6}$$

$$\mathcal{H}_{\underline{\boldsymbol{u}}_4}(\alpha_3) = \frac{\alpha_3-\alpha_1}{\alpha_4-\alpha_1}\cdot\frac{\alpha_3-\alpha_2}{\alpha_4-\alpha_2}\cdot\frac{\alpha_3-\alpha_6}{\alpha_4-\alpha_6}\cdot\frac{\alpha_3-\alpha_0}{\alpha_4-\alpha_0} = 5$$

&

$$\mathcal{H}_{\underline{\boldsymbol{u}}_4}(\alpha_3) = \frac{\alpha_4(\alpha_4-\alpha_5)}{\alpha_3(\alpha_3-\alpha_5)} = 5$$

$$\mathbf{G}_{RS} = \begin{bmatrix} 0 & 0 & 0 & 5 & 1 & 3 & 0 \\ 0 & 1 & 0 & 4 & 0 & 2 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 4 & 0 & 3 & 1 \\ 1 & 0 & 0 & 5 & 0 & 2 & 0 \end{bmatrix}$$

Locators : α_0 α_1 α_2 α_3 α_4 α_5 α_6

Low-Latency OSD

- Generation of BCH codeword candidates
  - Generate the initial codeword:

    Initial message: $\underline{\boldsymbol{u}} = (r_{j_0}, r_{j_1}, \dots, r_{j_{k'-1}}) \in \mathbb{F}_2^{k'}$

    Initial RS codeword: $\underline{\hat{\boldsymbol{v}}}^{(0)} = (\hat{v}_0^{(0)}, \hat{v}_1^{(0)}, \dots, \hat{v}_{n-1}^{(0)}) = \underline{\boldsymbol{u}} \cdot \mathbf{G}_{\mathrm{RS}} \qquad \hat{v}_j^{(0)} = r_j, \forall j \in \Theta$

  - Generate any (n, k') systematic RS codeword:

    The ω-th test error pattern: $\underline{\boldsymbol{e}}'^{(\omega)} = (e'^{(\omega)}_{j_0}, e'^{(\omega)}_{j_1}, \dots, e'^{(\omega)}_{j_{k'-1}}) \in \mathbb{F}_2^{k'}$

    The ω-th test message: $\underline{\boldsymbol{u}}^{(\omega)} = \underline{\boldsymbol{u}} + \underline{\boldsymbol{e}}'^{(\omega)}$

    The ω-th systematic RS codeword: $\underline{\hat{\boldsymbol{v}}}^{(\omega)} = (\hat{v}_0^{(\omega)}, \hat{v}_1^{(\omega)}, \dots, \hat{v}_{n-1}^{(\omega)}) \in \mathbb{F}_{2^m}^n$

$$\underline{\hat{\boldsymbol{v}}}^{(\omega)} = \underline{\boldsymbol{u}}^{(\omega)} \cdot \mathbf{G}_{\mathrm{RS}} = \underline{\hat{\boldsymbol{v}}}^{(0)} + \underline{\boldsymbol{e}}'^{(\omega)} \cdot \mathbf{G}_{\mathrm{RS}}$$

Low-Latency OSD

- Generation of BCH codeword candidates
  - **Theorem 2** (Identify invalid TEPs)

    If $\hat{v}_j^{(0)} + \sum_{i \in \Theta, e_i'^{(\omega)} \neq 0} \mathcal{H}_{\underline{u}_i}(\alpha_j) \in \{0,1\}, \forall j \in \Theta^c$, $\underline{\hat{v}}^{(\omega)}$ is a BCH codeword.

  - ***Example 5*** continues from ***Example 4***

    Initial message : $\underline{u} = (1,0,1,1,1)$

    Initial RS codeword: $\underline{\hat{v}}^{(0)} = \underline{u} \cdot \mathbf{G}_{\mathrm{RS}} = (1,0,1,5,1,3,1)$

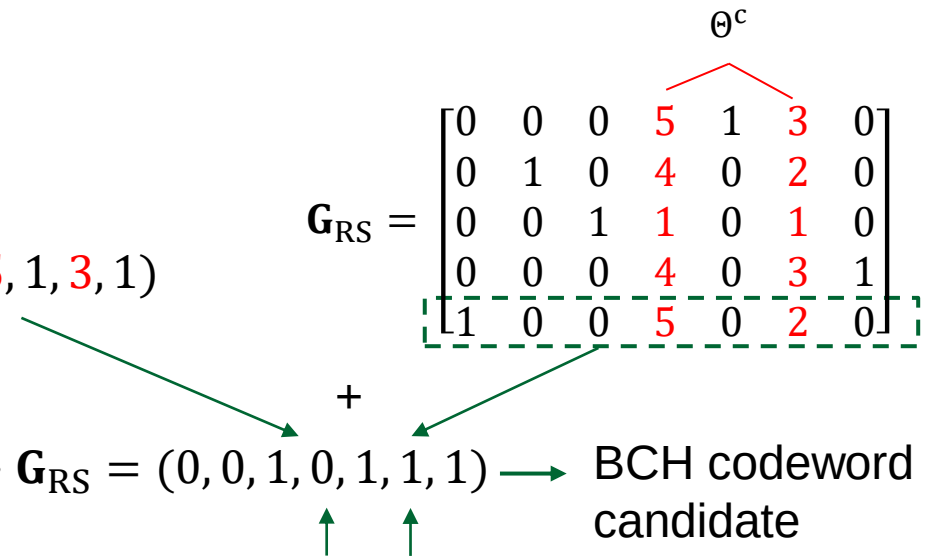
    Test error pattern 1: $\underline{e}'^{(1)} = (0,0,0,0,1)$

    Systematic RS codeword 1: $\underline{\hat{v}}^{(1)} = \underline{\hat{v}}^{(0)} + \underline{e}'^{(1)} \cdot \mathbf{G}_{\mathrm{RS}} = (0,0,1,0,1,1,1)$ → BCH codeword candidate

    ⋮

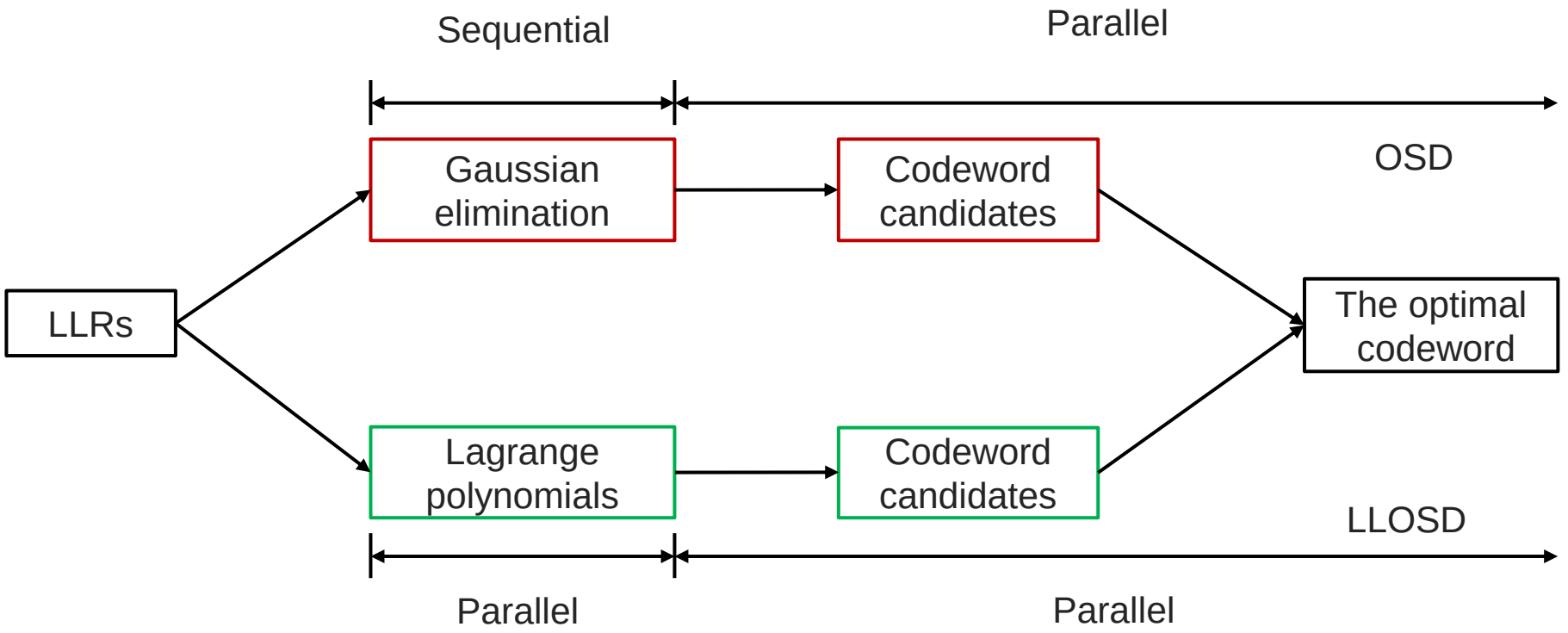
    Test error pattern 3: $\underline{e}'^{(3)} = (0,0,1,0,0)$

    Systematic RS codeword 3: $\underline{\hat{v}}^{(3)} = \underline{\hat{v}}^{(0)} + \underline{e}'^{(3)} \cdot \mathbf{G}_{\mathrm{RS}} = (1,0,0,\times,1,\times,1)$ → Invalid



Low-Latency OSD

- Overview of LLOSD



Low-Latency OSD

- Complexity comparison of LLOSD and OSD

Algorithms	Operations	Complexity
OSD (τ)	GE	$n \cdot (\min\{n-k, k\})^2$
	Compute $\underline{\hat{\boldsymbol{c}}}^{(0)}$	$k \cdot (n-k)$
	Compute $\underline{\hat{\boldsymbol{c}}}^{(\omega)}$	$(n-k) \cdot \sum_{\lambda=1}^{\tau} \lambda \binom{k}{\lambda}$
	Find $\underline{\hat{\boldsymbol{c}}}_{\text{opt}}$	$n \cdot \sum_{\lambda=0}^{\tau} \binom{k}{\lambda}$
LLOSD (τ)	Compute $\mathbf{G}_{\text{RS}}$	$2n \cdot \min\{n-k', k'\}$
	Compute $\underline{\hat{\boldsymbol{v}}}^{(0)}$	$k' \cdot (n-k')$
	Compute $\underline{\hat{\boldsymbol{c}}}^{(\omega)}$	$\sum_{\lambda=1}^{\tau} \binom{k}{\lambda} + \tau \sum_{j'=1}^{n-k'} N_{j'}$
	Find $\underline{\hat{\boldsymbol{v}}}_{\text{opt}}$	$nN_{n-k'}$

- Key differences

Operation type : binary → finite field

Compute $\mathbf{G}_{\text{BCH}}$ / $\mathbf{G}_{\text{RS}}$: $O(n^3) \rightarrow O(n^2)$

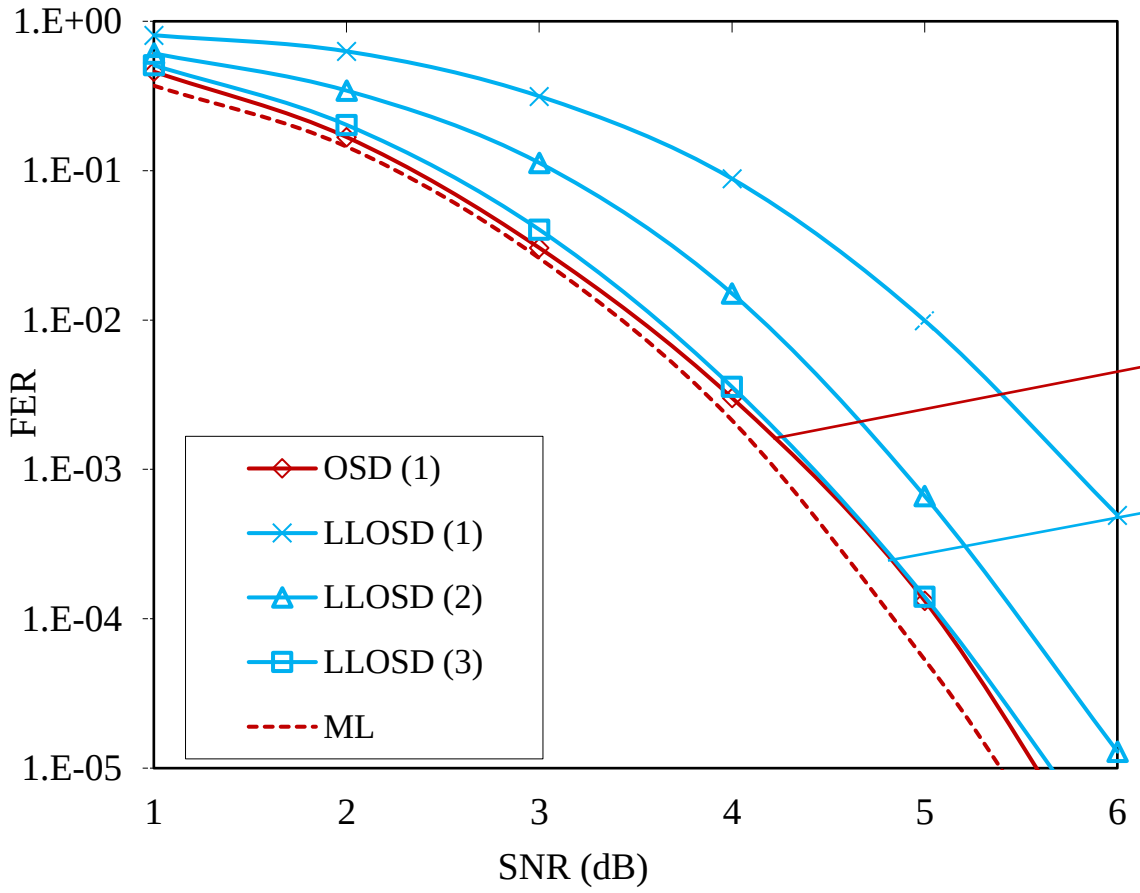
sequential → parallel

Candidate list : $\sum_{\lambda=0}^{\tau} \binom{k}{\lambda} \rightarrow N_{n-k'} = N_{\text{BCH}}$

$N_{j'}$ is the number of TEPs that yield binary symbols after the j' th judgement as in **Theorem 2**.

Low-Latency OSD

- Performance of the (63, 45) BCH code, AWGN, BPSK



- Complexity and latency comparison

Algorithms	SNR (dB)	Complexity		Latency (μs)
		$\mathbb{F}_2/\mathbb{F}_{64}$ oper.	Floating oper.	
OSD (1)	4	2.78×10^4	81	6.58×10^2
	5	2.60×10^4	19	5.34×10^2
	6	2.56×10^4	8	5.06×10^2
LLOSD (3)	4	1.81×10^4	15	1.99×10^3
	5	5.21×10^3	8	4.36×10^2
	6	2.58×10^3	7	1.32×10^2

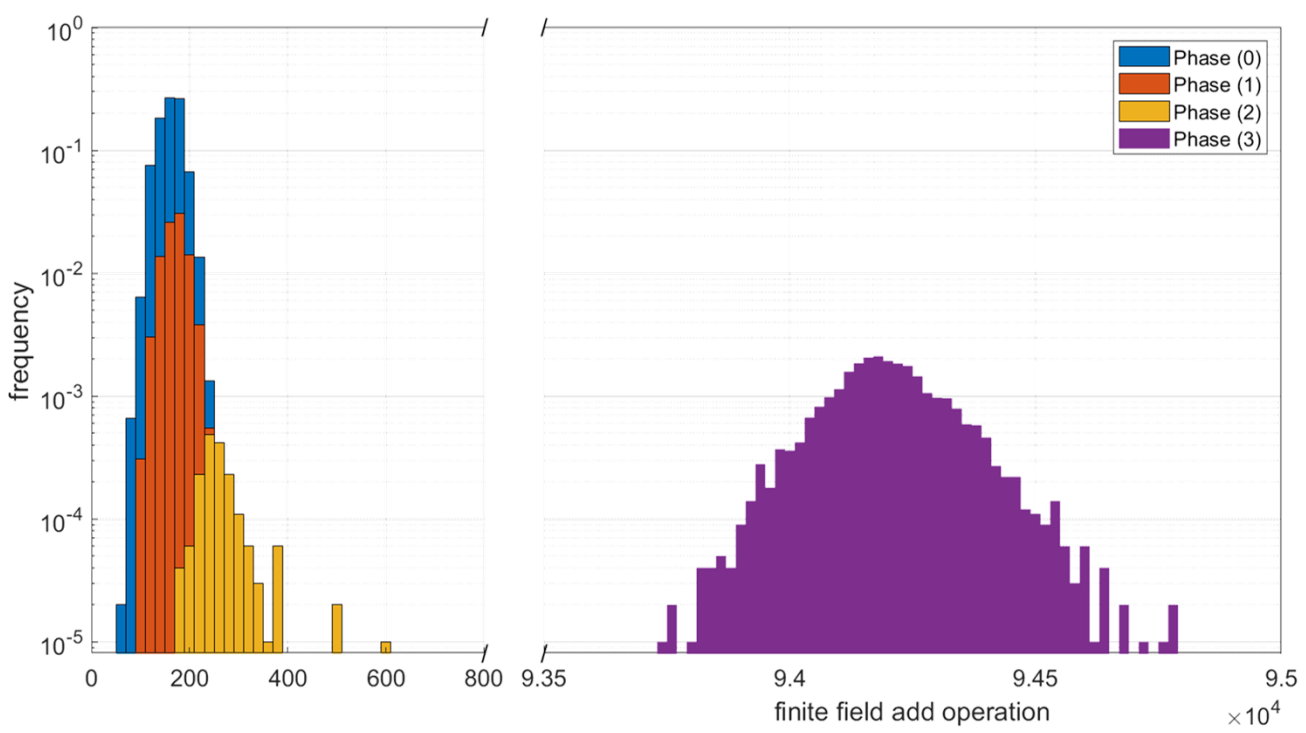
Simulation environment: Intel core i7-10710U CPU

Stopping rule: ML criterion [2]

[2] T. Kaneko *et al.*, "An efficient maximum-likelihood-decoding algorithm for linear block codes with algebraic decoder," *IEEE Trans. Inf. Theory*, vol. 40, no. 2, pp. 320–327, 1994.

Low-Latency OSD

- Re-encoding complexity distribution: LLOSD (3) of the (63,45) BCH code with ML criterion, SNR = 5.0 dB



- Colors are used to distinguish the phases that the re-encoding terminates at
- Phase (i): the re-encoding phase for weight-i TEPs

Terminate at	Ratio	Ave. #Additions
Phase (0)	87.89%	171
Phase (1)	9.22%	181
Phase (2)	0.18%	273
Phase (3)	2.71%	94,216

- Reasons for Multiple Peaks

  For the re-encoding phase of TEPs with a larger weight
  - The complexity is **greater**
  - The ML criterion is **more difficult** to satisfy
  - The probability of finding a BCH codeword is **lower**

Low-Latency OSD

- Segmented variation of LLOSD
  - Analysis
    - The error probability of the LLOSD is upper bounded by

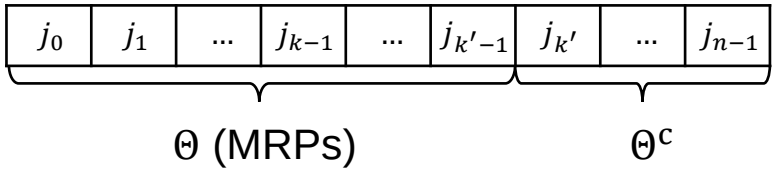
$$P_{\mathrm{e,LLOSD}}(\tau) \le P_{\mathrm{e,ML}} + P_{\mathrm{list}}(\tau)$$

- $P_{\mathrm{e,ML}}$: ML decoding error probability,

  depends on the weight distribution of the code

- $P_{\mathrm{list}}(\tau)$: List decoding error probability,

  = Prob{the channel introduces more than τ errors in MRPs}



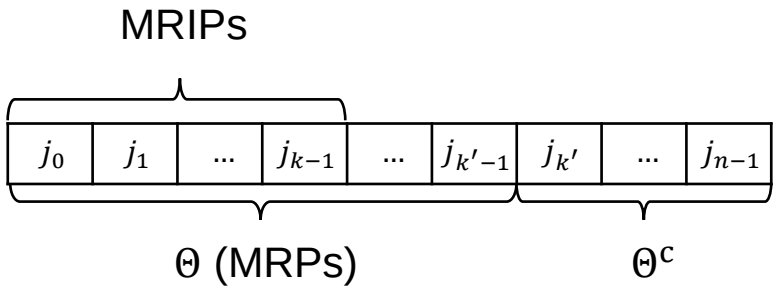
Low-Latency OSD

- Segmented variation of LLOSD
  - For OSD

If $\tau \geq \min\left\{\left\lceil \frac{d}{4} - 1 \right\rceil, k\right\}$ → The list error probability of OSD: $P_{\text{list}}(\tau) \ll P_{\text{e,ML}}$

→ The OSD can produce ~ ML decoding performance

→ We only need the decoding order of τ for the MRIPs



  - For LLOSD

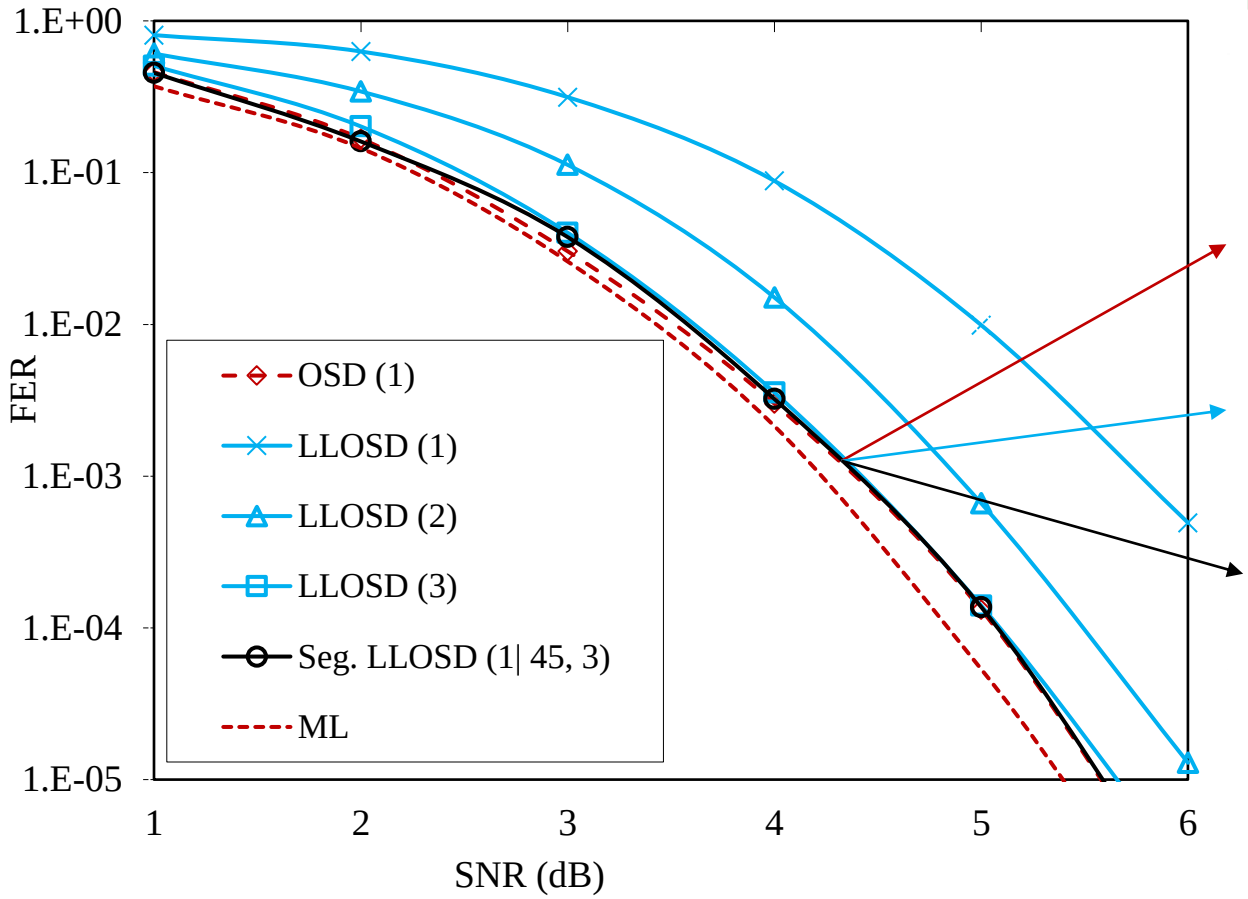
TEPs: $\underline{\boldsymbol{e}}'^{(\omega)} = (e'^{(\omega)}_{j_0}, e'^{(\omega)}_{j_1}, \ldots, e'^{(\omega)}_{j_{k-1}}, e'^{(\omega)}_{j_k}, e'^{(\omega)}_{j_{k+1}}, \ldots, e'^{(\omega)}_{j_{k'-1}})$

$\tau_1 = \min\left\{\left\lceil \frac{d}{4} - 1 \right\rceil, k\right\}$ is sufficient extra order τ_2

Number of TEPs N_{TEPs} : $\sum_{i=0}^{\tau} \binom{k'}{i} \rightarrow \sum_{i_1=0}^{\tau_1} \binom{k}{i_1} \cdot \sum_{i_2=0}^{\tau_2} \binom{k'-k}{i_2}$

Low-Latency OSD

- Performance of the (63, 45) BCH code, AWGN, BPSK



- Complexity and latency comparison

Algorithms	SNR (dB)	Complexity		Latency (μs)
		$\mathbb{F}_2/\mathbb{F}_{64}$ oper.	Floating oper.	
OSD (1)	4	2.78×10^4	81	6.58×10^2
	5	2.60×10^4	19	5.34×10^2
	6	2.56×10^4	8	5.06×10^2
LLOSD (3)	4	1.81×10^4	15	1.99×10^3
	5	5.21×10^3	8	4.36×10^2
	6	2.58×10^3	7	1.32×10^2
Seg. LLOSD (1 | 45, 3)	4	3.69×10^3	8	2.71×10^2
	5	2.64×10^3	7	1.44×10^2
	6	2.45×10^3	7	1.17×10^2

Simulation environment: Intel core i7-10710U CPU

Stopping rule: ML criterion

Low-Latency OSD

- Concatenated perspective
  - Systematic parity-check matrix $\Lambda(\mathbf{H}_{\mathrm{RS}}) = [\mathbf{P} \quad \mathbf{I}_{n-k'}]$ $\longleftrightarrow$ $\mathcal{C}_{\mathrm{RS}}[n, k']$ $\longleftrightarrow$ Systematic generator matrix $\Lambda(\mathbf{G}_{\mathrm{RS}}) = [\mathbf{I}_{k'} \quad \mathbf{P}^{\mathrm{T}}]$

1. $\mathbb{F}_{2^m} \cong \mathbb{F}_2^m$
2. Row permutation

$$\Lambda(\mathbf{H}_{\mathrm{BCH}}) = \begin{bmatrix} \mathbf{P}^{(0)} & \mathbf{I}_{n-k'} \\ \mathbf{P}^{(1)} & \mathbf{0} \\ \vdots & \vdots \\ \mathbf{P}^{(m-1)} & \mathbf{0} \end{bmatrix} \text{ (over } \mathbb{F}_2\text{)}$$

$[\mathbf{P}^{(0)} \quad \mathbf{I}_{n-k'}]$: systematic parity-check matrix of an (n, k') binary code $\mathcal{C}^*[n, k']$

$\mathbf{P}^* = \begin{bmatrix} \mathbf{P}^{(1)} \\ \vdots \\ \mathbf{P}^{(m-1)} \end{bmatrix}$: parity-check matrix of $\mathcal{C}_{\mathrm{BCH}}^{\Theta^{\mathrm{c}}}[k', k]$ ($\mathcal{C}_{\mathrm{BCH}}[n, k]$ punctured on the positions in Θ^{c})

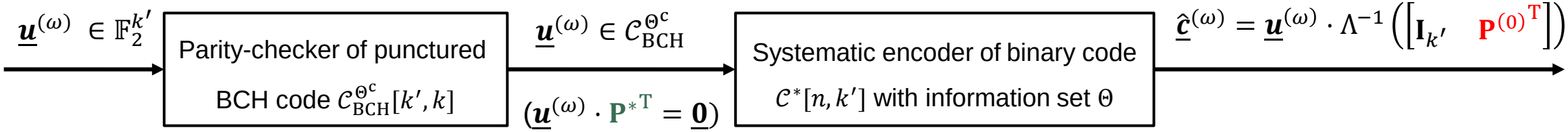
- If $\underline{\boldsymbol{u}}^{(\omega)} \cdot \mathbf{P}^{*\mathrm{T}} = \underline{\mathbf{0}}$, let $\hat{\underline{\boldsymbol{c}}}^{(\omega)} = \Lambda^{-1}(\underline{\boldsymbol{u}}^{(\omega)} \cdot [\mathbf{I}_{k'} \quad \mathbf{P}^{(0)\mathrm{T}}])$

$$\Longrightarrow \hat{\underline{\boldsymbol{c}}}^{(\omega)} \cdot \mathbf{H}_{\mathrm{BCH}}^{\mathrm{T}} = \underline{\mathbf{0}} \iff \hat{\underline{\boldsymbol{c}}}^{(\omega)} \in \mathcal{C}_{\mathrm{BCH}}$$

$\underline{\boldsymbol{u}}^{(\omega)} \in \mathbb{F}_2^{k'}$ → Parity-checker of punctured BCH code $\mathcal{C}_{\mathrm{BCH}}^{\Theta^{\mathrm{c}}}[k', k]$ → $\underline{\boldsymbol{u}}^{(\omega)} \in \mathcal{C}_{\mathrm{BCH}}^{\Theta^{\mathrm{c}}}$ $(\underline{\boldsymbol{u}}^{(\omega)} \cdot \mathbf{P}^{*\mathrm{T}} = \underline{\mathbf{0}})$ → Systematic encoder of RS code $\mathcal{C}_{\mathrm{RS}}[n, k']$ with information set Θ → $\hat{\underline{\boldsymbol{c}}}^{(\omega)} = \underline{\boldsymbol{u}}^{(\omega)} \cdot \mathbf{G}_{\mathrm{RS}} \in \mathcal{C}_{\mathrm{BCH}}$

Low-Latency OSD

- Concatenated perspective



- ***Example 6***

$$\mathbf{H}_{\mathrm{RS}} = \begin{bmatrix} \sigma^3 & 1 & \sigma^3 & \sigma & \sigma & 1 & 0 \\ \sigma^4 & 1 & \sigma^5 & \sigma^5 & \sigma^4 & 0 & 1 \end{bmatrix} \xrightarrow{\mathbb{F}_2 m \cong \mathbb{F}_2^m} \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \xrightarrow{\text{row perm.}} \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} = \mathbf{H}_{\mathrm{BCH}}$$

(1) $\underline{\boldsymbol{u}}^{(\omega)} = (1,0,1,1,0)$: $\underline{\boldsymbol{u}}^{(\omega)} \cdot \mathbf{P}^{*\mathrm{T}} = (1,1,0,1)$

(2) $\underline{\boldsymbol{u}}^{(\omega)} = (1,0,1,0,0)$: $\underline{\boldsymbol{u}}^{(\omega)} \cdot \mathbf{P}^{*\mathrm{T}} = \underline{\mathbf{0}} \longrightarrow \begin{cases} \hat{c}_5^{(\omega)} = \underline{\boldsymbol{u}}^{(\omega)} \cdot (1,1,1,0,0)^{\mathrm{T}} = 0 \\ \hat{c}_6^{(\omega)} = \underline{\boldsymbol{u}}^{(\omega)} \cdot (0,1,1,1,0)^{\mathrm{T}} = 1 \end{cases} \Rightarrow \hat{\underline{\boldsymbol{c}}}^{(\omega)} = (1,0,1,1,0,0,1)$

Low-Latency OSD

- Concatenated perspective
  - For LLOSD (τ)

    N_{TEPs} : number of TEPs

    $$N_{\mathrm{TEPs}} = \sum_{i=0}^{\tau} \binom{k'}{i}$$

    N_{BCH} : number of BCH codeword candidates:
  - **Theorem 3**

    If the channel condition is sufficiently good (SNR $\longrightarrow \infty$)

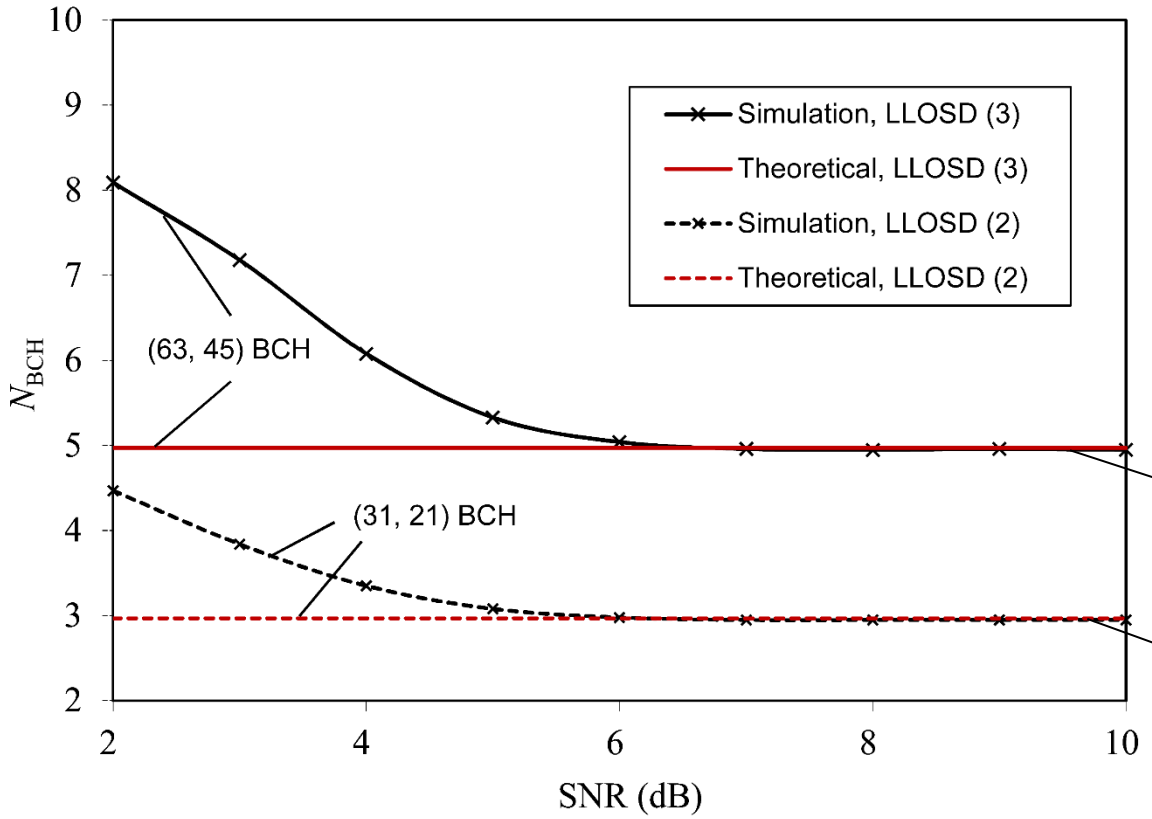
    $$\Longrightarrow N_{\mathrm{BCH}} = \sum_{i=0}^{\tau} A_i$$

    A_i: number of weight-i codewords in $\mathcal{C}_{\mathrm{BCH}}^{\Theta^{\mathrm{c}}}[k', k]$
    - The set of punctured positions Θ^{c} varies for each decoding event, and A_i varies accordingly
  - In general

    $$\sum_{i=0}^{\tau} A_i \ll \sum_{i=0}^{\tau} \binom{k'}{i}$$

Low-Latency OSD

- Concatenated perspective
  - The number of BCH codeword candidates N_{BCH} in decoding the $\mathcal{C}_{\mathrm{BCH}}[63, 45]$ and $\mathcal{C}_{\mathrm{BCH}}[31, 21]$



  - Theoretical line

    Randomly puncturing $n - k'$ positions of $\mathcal{C}_{\mathrm{BCH}}[n, k]$, obtaining $\mathcal{C}_{\mathrm{BCH}}^{\Theta^c}[k', k]$ and its weight distribution $\{A_i\}$

$N_{\mathrm{TEPs}} = 30914$

$N_{\mathrm{TEPs}} = 379$

Low-Latency OSD

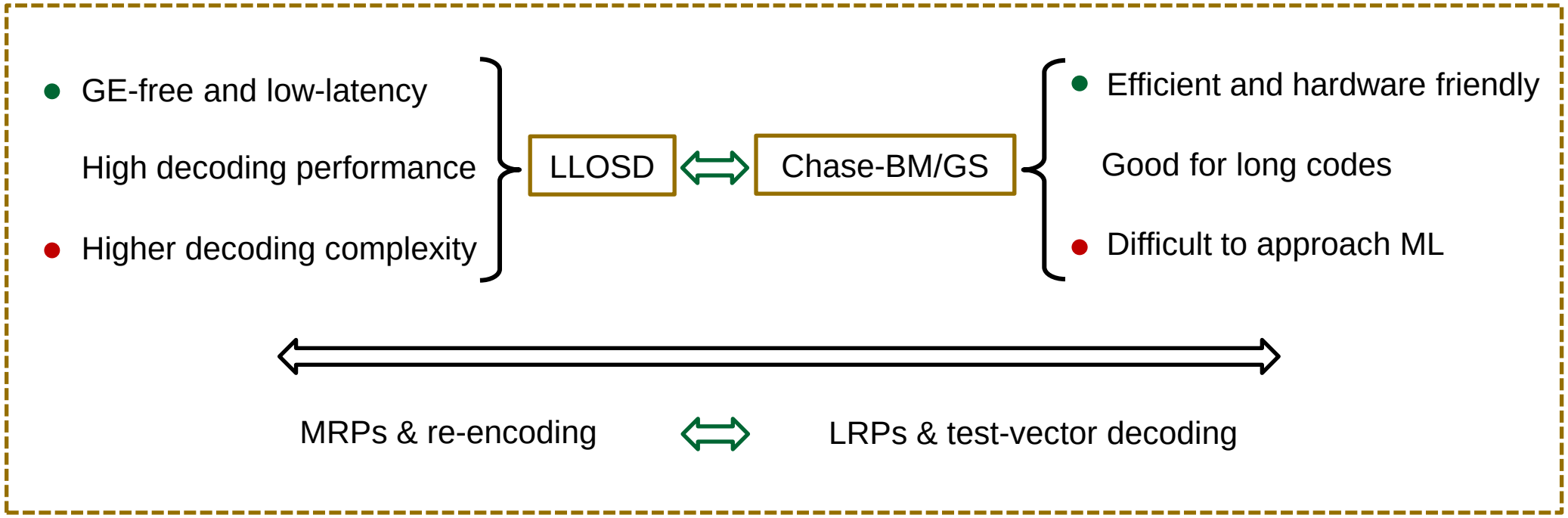
- The average number of BCH codeword candidates in LLOSD (3) of the (63,45) BCH code

TEP weight	#Weight-i TEPs	N_{BCH}		
		SNR $=$ 2dB	4dB	6dB
0	1	0.14	0.65	0.97
1	57	0.27	0.27	0.03
2	1,569	0.56	0.20	0.07
3	29,260	7.10	4.97	3.97

- Remarks
  - The probability of finding a BCH codeword decreases as the TEP weight increases
  - As the SNR increases, the likelihood of generating a BCH codeword using a **weight-0 TEP** increases, while the likelihood decreases when using a TEP of a larger weight

Hybrid Soft Decoding

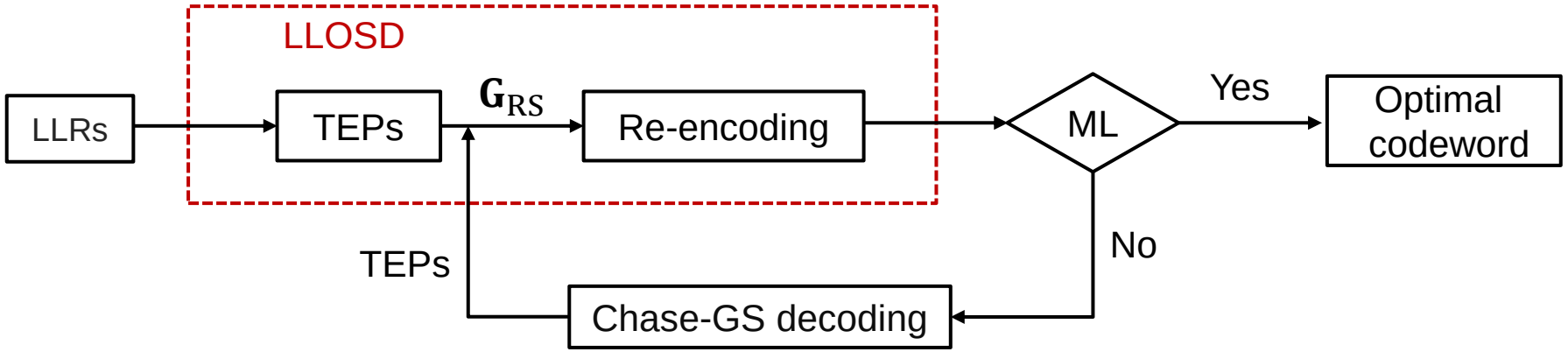
- LLOSD vs. Chase - BM / GS
  - It remains challenging to decode longer BCH codes with LLOSD



LRPs: Least reliable positions

Hybrid Soft Decoding

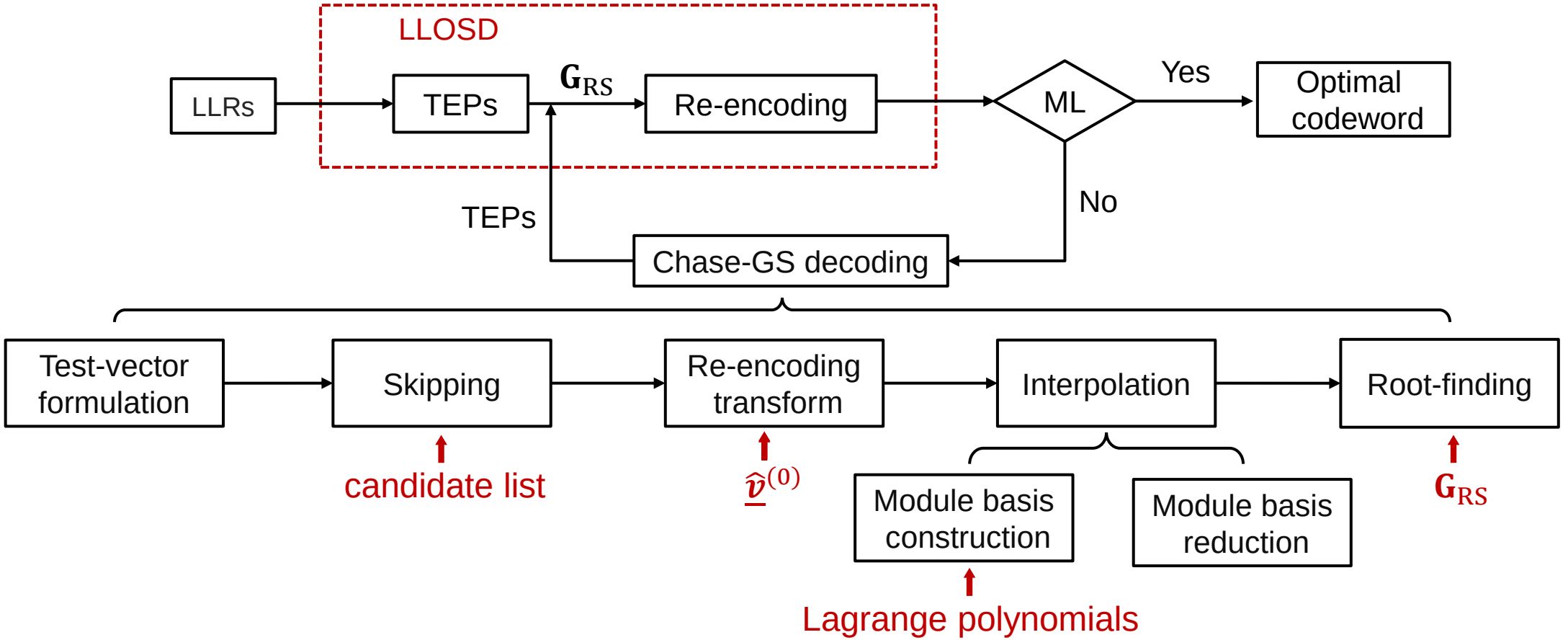
- Integrating LLOSD and Chase-GS decoding



  - Advantages
    - LLOSD output list can be utilized
    - Computation sharing
    - Low complexity root-finding

Hybrid Soft Decoding

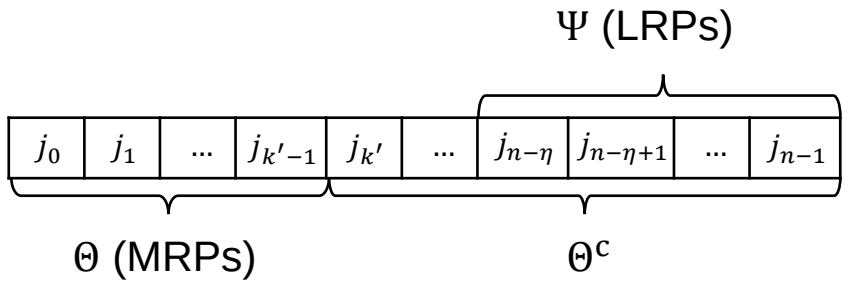
- Integrating LLOSD and Chase-GS decoding



Hybrid Soft Decoding

- Key steps of HSD
  - **Test-vector formulation**

$\Psi = \{j_{n-\eta}, j_{n-\eta+1}, \cdots, j_{n-1}\}$: η least reliable positions (LRPs)



The 2^η test-vectors can be formulated as

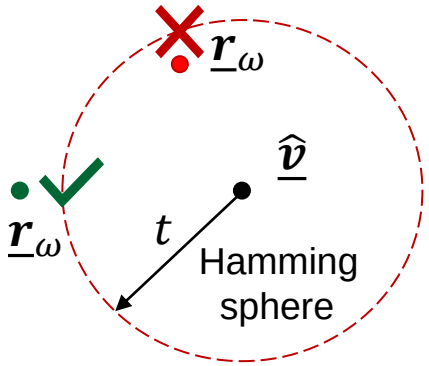
$$\Lambda(\underline{\boldsymbol{r}}_\omega) = (r_{j_0}, r_{j_1}, \cdots, r_{j_{n-\eta-1}}, \underline{r_{j_{n-\eta}}^{(\omega)}, \cdots, r_{j_{n-1}}^{(\omega)}})$$

00…000
00…001
00…010
00…011
…
11…110
11…111

$$0 \le \omega \le 2^\eta - 1$$

  - **Skipping** (based on candidate list)

If $d_{\mathrm{H}}(\underline{\hat{\boldsymbol{v}}}, \underline{\boldsymbol{r}}_\omega) \le t$, $\underline{\boldsymbol{r}}_\omega$ will be decoded as $\underline{\hat{\boldsymbol{v}}}$ → $\underline{\boldsymbol{r}}_\omega$ can be skipped



$d_{\mathrm{H}}(\underline{\hat{\boldsymbol{v}}}, \underline{\boldsymbol{r}}_\omega)$: Hamming distance between $\underline{\hat{\boldsymbol{v}}}$ and test-vector $\underline{\boldsymbol{r}}_\omega$

Hybrid Soft Decoding

- Key steps of HSD

  - **Re-encoding transform** (based on the initial RS codeword)

    Initial RS codeword (from LLOSD): $\underline{\hat{\boldsymbol{v}}}^{(0)} = \underline{\boldsymbol{u}} \cdot \mathbf{G}_{\mathrm{RS}} \in \mathbb{F}_{2^m}^n$

    Transformed test vector: $\underline{\boldsymbol{z}}_\omega = \underline{\boldsymbol{r}}_\omega - \underline{\hat{\boldsymbol{v}}}^{(0)} \rightarrow \Lambda(\underline{\boldsymbol{z}}_\omega) = (0, 0, \ldots, 0, z_{j_{k'}}^{(\omega)}, \ldots, z_{j_{n-1}}^{(\omega)})$

    Finding a Gröbner basis of the interpolation module

    $$\mathcal{M}_\omega = \{Q \in \mathbb{F}_{2^m}[x, y] \mid \deg_y Q \le 1; Q\left(\alpha_j, z_j^{(\omega)}/V(\alpha_j)\right) = 0, \forall j \in \Theta^c\}$$

  - **Interpolation — module basis construction** (based on the Lagrange polynomials)

    Interpolation points:

    $$\underbrace{(\alpha_{j_0}, 0), (\alpha_{j_1}, 0), \ldots, (\alpha_{j_{k'-1}}, 0)}_{\text{Interpolated by } V(x) = \prod_{j \in \Theta}(x - \alpha_j)}, \underbrace{(\alpha_{j_{k'}}, \frac{z_{j_{k'}}^{(\omega)}}{V(\alpha_{j_{k'}})}) \ldots, (\alpha_{j_{n-1}}, \frac{z_{j_{n-1}}^{(\omega)}}{V(\alpha_{j_{n-1}})})}_{\text{Lagrange interpolation}}$$

Hybrid Soft Decoding

- Key steps of HSD
  - **Interpolation — module basis construction** (based on the Lagrange polynomials)

Seed polynomials:

$$\mathcal{G}(x) = \prod_{j \in \Theta^c} (x - \alpha_j) \quad \& \quad R_\omega(x) = \sum_{j \in \Theta^c} z_j^{(\omega)} T_j'(x)$$

$$T_j'(x) = \frac{\prod_{j' \in \Theta^c, j' \neq j} (x - \alpha_{j'})}{\prod_{j'=0, j' \neq j}^{n-1} (\alpha_j - \alpha_{j'})}$$

Computed in generating $\mathbf{G}_{\mathrm{RS}}$

For $\underline{\boldsymbol{r}}_\omega$ ($\underline{\boldsymbol{z}}_\omega$), module $\mathcal{M}_\omega$ can be generated by

$$\begin{cases} P_{\omega,0}(x,y) = \mathcal{G}(x) \\ P_{\omega,1}(x,y) = y - R_\omega(x) \end{cases}$$

Basis reduction → The Gröbner basis of $\mathcal{M}_\omega$

→ The interpolation polynomial

$$Q_\omega(x,y) = V(x) Q_\omega^{*(0)}(x) + Q_\omega^{*(1)}(x) y$$

Hybrid Soft Decoding

- Key steps of HSD
  - **Root-finding**

Estimated message polynomial: $\hat{u}_\omega(x) = \dfrac{V(x)Q_\omega^{*(0)}(x)}{Q_\omega^{*(1)}(x)}$

  - ***Lemma* 4**

The evaluation values of polynomial $\hat{u}_\omega(x)$ over MRPs form a TEP of the LLOSD, i.e.,

$$\underline{\hat{\boldsymbol{e}}}_\omega = (\hat{u}_\omega(\alpha_{j_0}), \hat{u}_\omega(\alpha_{j_1}), \dots, \hat{u}_\omega(\alpha_{j_{k'-1}}))$$

Codeword candidate of LLOSD: $\underline{\hat{\boldsymbol{v}}}^{(\omega)} = \underline{\boldsymbol{e}}'^{(\omega)} \cdot \mathbf{G}_{\mathrm{RS}} + \underline{\hat{\boldsymbol{v}}}^{(0)}$

Codeword candidate of Chase-GS: $\underline{\hat{\boldsymbol{v}}}_\omega = \left(\hat{u}_\omega(\alpha_0), \hat{u}_\omega(\alpha_1), \dots, \hat{u}_\omega(\alpha_{n-1})\right) + \underline{\hat{\boldsymbol{v}}}^{(0)}$

$$= \left(\hat{u}_\omega(\alpha_{j_0}), \hat{u}_\omega(\alpha_{j_1}), \dots, \hat{u}_\omega\left(\alpha_{j_{k'-1}}\right)\right) \cdot \mathbf{G}_{\mathrm{RS}} + \underline{\hat{\boldsymbol{v}}}^{(0)}$$

$$= \underline{\hat{\boldsymbol{e}}}_\omega \cdot \mathbf{G}_{\mathrm{RS}} + \underline{\hat{\boldsymbol{v}}}^{(0)}$$

Hybrid Soft Decoding

- Key steps of HSD
  - **Partial root-finding** (based on $\mathbf{G}_{\mathrm{RS}}$)

The estimated TEP $\underline{\hat{\boldsymbol{e}}}_{\omega} = (\hat{u}_{\omega}(\alpha_{j_0}), \hat{u}_{\omega}(\alpha_{j_1}), \ldots, \hat{u}_{\omega}(\alpha_{j_{k'-1}}))$ can be determined as

$$\hat{u}_{\omega}(\alpha_j) = \begin{cases} 1, & \text{if } Q_{\omega}^{*(1)}(\alpha_j) = 0 \\ 0, & \text{otherwise} \end{cases} \qquad \forall j \in \Theta$$

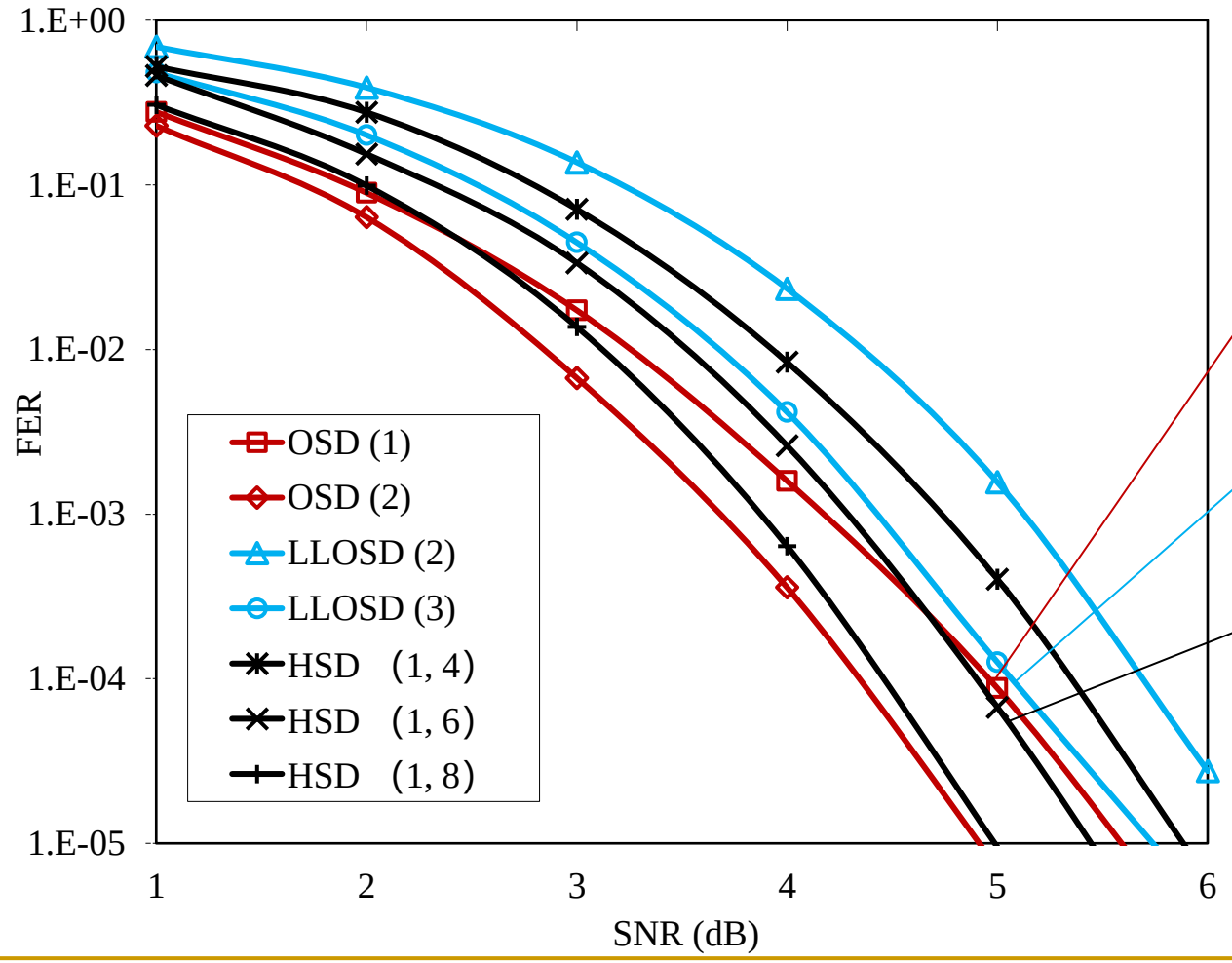
The estimated codeword $\underline{\hat{\boldsymbol{v}}}_{\omega}$ can be generated as

Provided by Chase-GS

$$\underline{\hat{\boldsymbol{v}}}_{\omega} = (\hat{u}_{\omega}(\alpha_{j_0}), \hat{u}_{\omega}(\alpha_{j_1}), \ldots, \hat{u}_{\omega}(\alpha_{j_{k'-1}})) \cdot \mathbf{G}_{\mathrm{RS}} + \underline{\hat{\boldsymbol{v}}}^{(0)} = \underbrace{\underline{\hat{\boldsymbol{e}}}_{\omega} \cdot \mathbf{G}_{\mathrm{RS}} + \underline{\hat{\boldsymbol{v}}}^{(0)}}_{\text{LLOSD}}$$

Hybrid Soft Decoding

- Performance of the (63, 39) BCH code



OSD (1)
Complexity: 4.08×10^4
(BOPs)

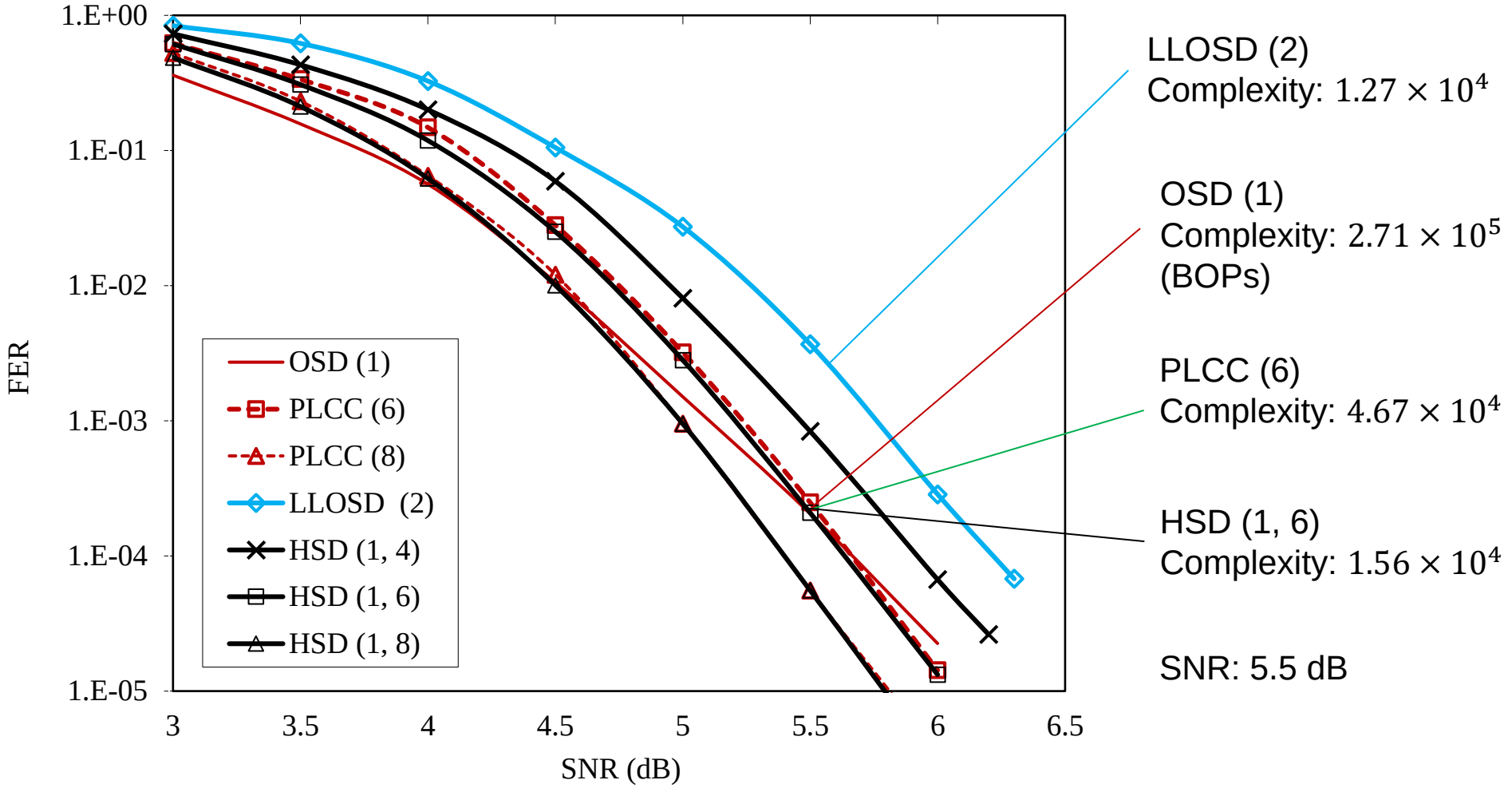
LLOSD (3)
Complexity: 4.45×10^3

HSD (1, 6)
Complexity: 2.71×10^3

SNR: 5 dB

Hybrid Soft Decoding

- Performance of the (255, 223) BCH code



PLCC decoding is the progressive variant of the Chase-GS