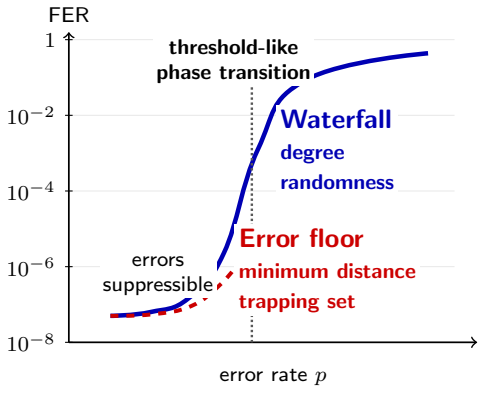
Bringing Classical LDPC Design Principles to Quantum LDPC Codes

Kenta Kasai

Institute of Science Tokyo

Based on arXiv:2601.08824

Waterfall vs. Error Floor



Approximation in p

$$\mathrm{FER}(p,n) \approx \mathrm{FER}_{\mathrm{WF}}(p,n) + \mathrm{FER}_{\mathrm{EF}}(p)$$

Waterfall

$$\mathrm{FER}_{\mathrm{WF}}(p,n) \approx Q\left(\frac{\sqrt{n}\,(p_{\mathrm{th}} - p)}{\alpha}\right)$$

$$Q(x) = \Pr[Z > x], \quad Z \sim N(0,1)$$

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-t^2/2}\, dt$$

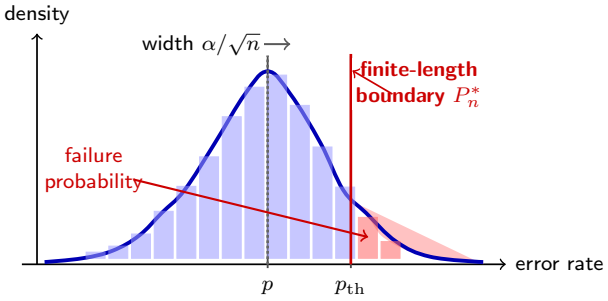
Error floor

$$\mathrm{FER}_{\mathrm{EF}}(p) \approx \sum_{s \in \mathcal{S}} A_s p^{w_s}$$

finite-length scaling
residual size $O(n)$

rare structures
residual size $O(1)$

Why the Waterfall Has a Q-Shape

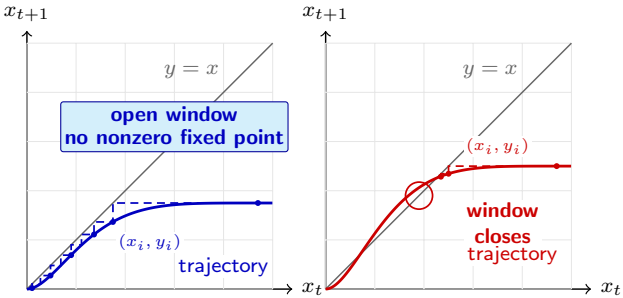


LDPC finite-length scaling

$$\mathrm{FER}_{\mathrm{WF}}(p,n) \approx Q\left(\frac{\sqrt{n}\,(p_{\mathrm{th}}-p)}{\alpha}\right)$$

- BP trajectory follows density evolution.
- Finite-length fluctuation is asymptotically Gaussian.
- Failure occurs when it crosses the threshold boundary.
- α: ensemble/channel-dependent scaling parameter.

Density-Evolution Trajectory



$p < p_{\mathrm{BP}}$ $\qquad$ $p > p_{\mathrm{BP}}$

The BP threshold is where a positive fixed point first appears.

DE recursion

$$x_{t+1} = f_p(x_t)$$

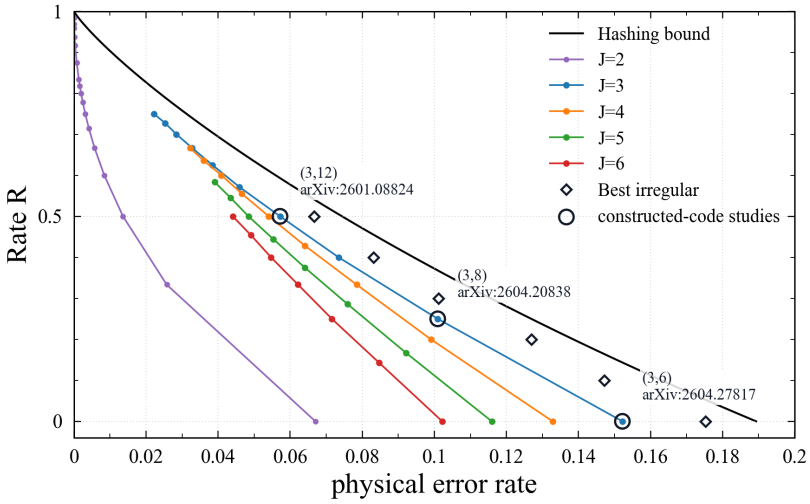
$$x_0 = p$$

Threshold

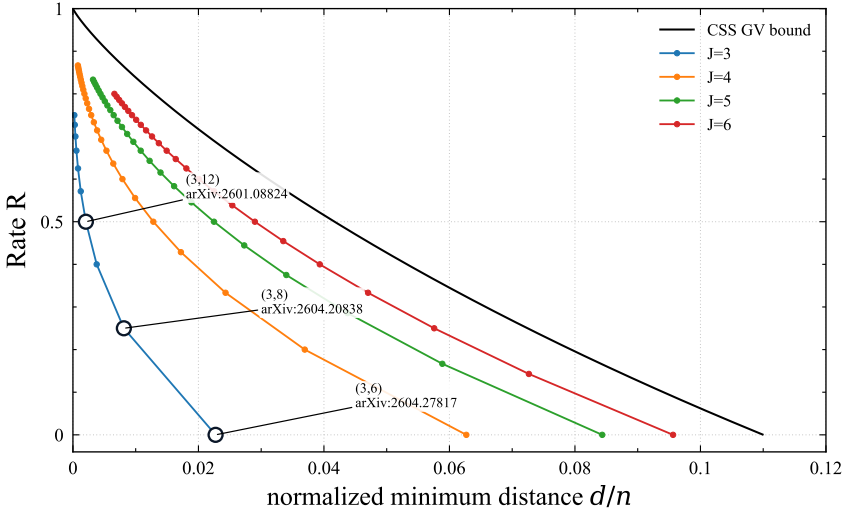
$$p_{\mathrm{BP}} = \sup\{p : \text{no nonzero fixed point}\}$$

At threshold, a nonzero fixed point appears by tangency.

Classical LDPC Benchmark: BP Threshold



Classical LDPC Benchmark: Minimum Distance



Design Freedom vs. CSS Commutativity

Classical LDPC

Any sparse H

Free choice of one sparse graph

- degree distribution
- randomness
- large girth
- minimum distance

Quantum LDPC code

$$H_X H_Z' = 0$$

X and Z sides are coupled by commutativity

- two sparse graphs
- orthogonality constraint
- coupled design variables
- room for degeneracy

Enable classical-LDPC-style design freedom under CSS commutativity.

Minimum Distance

Classical code

$$d = \min \operatorname{Ker} H \setminus \{\mathbf{0}\}$$

Minimum nonzero vector in the kernel

Quantum CSS code

$$d_Z := \min \operatorname{Ker} H_X \setminus \operatorname{Row} H_Z,$$
$$d_X := \min \operatorname{Ker} H_Z \setminus \operatorname{Row} H_X,$$
$$d = \min(d_X, d_Z)$$

Smallest nontrivial logical operator

Problem: Latent Rows Become Logicals

Parent construction

$$\hat{H}_X = \begin{bmatrix} H_X \\ \tilde{H}_X \end{bmatrix}, \qquad \hat{H}_Z = \begin{bmatrix} H_Z \\ \tilde{H}_Z \end{bmatrix}, \qquad \hat{H}_X(\hat{H}_Z)' = 0$$

- Latent rows = parent rows not used as active stabilizer checks.
- Full orthogonality forces $H_X(\tilde{H}_Z)' = 0$, $H_Z(\tilde{H}_X)' = 0$.
- Then $\tilde{H}_X, \tilde{H}_Z$ can become logicals.
- $d_{\min} \leq$ row weight.

Design Principle

- Active orthogonality only: $H_X H_Z' = 0$.
- Latent blocks non-orthogonal: $H_X(\tilde{H}_Z)' \neq 0$, $H_Z(\tilde{H}_X)' \neq 0$.
- Latent distances: $d_X^{(\mathrm{lat})}, d_Z^{(\mathrm{lat})}$.

Latent-based distances

$$d_X^{(\mathrm{lat})} := \min\{(\operatorname{Ker} H_Z \cap \operatorname{Row} \tilde{H}_X) \setminus \operatorname{Row} H_X\},$$
$$d_Z^{(\mathrm{lat})} := \min\{(\operatorname{Ker} H_X \cap \operatorname{Row} \tilde{H}_Z) \setminus \operatorname{Row} H_Z\}.$$
$$d_{\min} \leq \min\{d_X^{(\mathrm{lat})}, d_Z^{(\mathrm{lat})}\}$$

Code Construction

Example $J = 3, L = 12$.

$$\hat{H}_X = \left(\begin{array}{cccccc|cccccc} F_0 & F_1 & F_2 & F_3 & F_4 & F_5 & G_0 & G_1 & G_2 & G_3 & G_4 & G_5 \\ F_5 & F_0 & F_1 & F_2 & F_3 & F_4 & G_5 & G_0 & G_1 & G_2 & G_3 & G_4 \\ F_4 & F_5 & F_0 & F_1 & F_2 & F_3 & G_4 & G_5 & G_0 & G_1 & G_2 & G_3 \\ \hline F_3 & F_4 & F_5 & F_0 & F_1 & F_2 & G_3 & G_4 & G_5 & G_0 & G_1 & G_2 \\ F_2 & F_3 & F_4 & F_5 & F_0 & F_1 & G_2 & G_3 & G_4 & G_5 & G_0 & G_1 \\ F_1 & F_2 & F_3 & F_4 & F_5 & F_0 & G_1 & G_2 & G_3 & G_4 & G_5 & G_0 \end{array}\right), \qquad \hat{H}_X(\hat{H}_Z)' = \left(\begin{array}{cccccc|cccccc} \Psi_0 & \Psi_1 & \Psi_2 & \Psi_3 & \Psi_4 & \Psi_5 \\ \Psi_5 & \Psi_0 & \Psi_1 & \Psi_2 & \Psi_3 & \Psi_4 \\ \Psi_4 & \Psi_5 & \Psi_0 & \Psi_1 & \Psi_2 & \Psi_3 \\ \hline \Psi_3 & \Psi_4 & \Psi_5 & \Psi_0 & \Psi_1 & \Psi_2 \\ \Psi_2 & \Psi_3 & \Psi_4 & \Psi_5 & \Psi_0 & \Psi_1 \\ \Psi_1 & \Psi_2 & \Psi_3 & \Psi_4 & \Psi_5 & \Psi_0 \end{array}\right),$$

$$\hat{H}_Z = \left(\begin{array}{cccccc|cccccc} G'_0 & G'_5 & G'_4 & G'_3 & G'_2 & G'_1 & F'_0 & F'_5 & F'_4 & F'_3 & F'_2 & F'_1 \\ G'_1 & G'_0 & G'_5 & G'_4 & G'_3 & G'_2 & F'_1 & F'_0 & F'_5 & F'_4 & F'_3 & F'_2 \\ G'_2 & G'_1 & G'_0 & G'_5 & G'_4 & G'_3 & F'_2 & F'_1 & F'_0 & F'_5 & F'_4 & F'_3 \\ \hline G'_3 & G'_2 & G'_1 & G'_0 & G'_5 & G'_4 & F'_3 & F'_2 & F'_1 & F'_0 & F'_5 & F'_4 \\ G'_4 & G'_3 & G'_2 & G'_1 & G'_0 & G'_5 & F'_4 & F'_3 & F'_2 & F'_1 & F'_0 & F'_5 \\ G'_5 & G'_4 & G'_3 & G'_2 & G'_1 & G'_0 & F'_5 & F'_4 & F'_3 & F'_2 & F'_1 & F'_0 \end{array}\right), \quad \Psi_r := \sum_{u=0}^{L/2-1} \left(F_u G_{r-u} + G_{r-u} F_u\right), \quad r \in [L/2].$$

Code Construction (Cont.)

Example $J = 3, L = 12$.

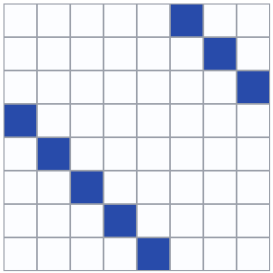
$$\hat{H}_X(\hat{H}_Z)' = \left(\begin{array}{cccc|cc} \Psi_0 & \Psi_1 & \Psi_2 & \Psi_3 & \Psi_4 & \Psi_5 \\ \Psi_5 & \Psi_0 & \Psi_1 & \Psi_2 & \Psi_3 & \Psi_4 \\ \Psi_4 & \Psi_5 & \Psi_0 & \Psi_1 & \Psi_2 & \Psi_3 \\ \hline \Psi_3 & \Psi_4 & \Psi_5 & \Psi_0 & \Psi_1 & \Psi_2 \\ \Psi_2 & \Psi_3 & \Psi_4 & \Psi_5 & \Psi_0 & \Psi_1 \\ \Psi_1 & \Psi_2 & \Psi_3 & \Psi_4 & \Psi_5 & \Psi_0 \end{array}\right)$$

Γ	G_0	G_1	G_2	G_3	G_4	G_5
F_0	C	C	C		C	C
F_1	C	C		C	C	C
F_2	C		C	C	C	C
F_3		C	C	C	C	C
F_4	C	C	C	C	C	
F_5	C	C	C	C		C

$$\Delta := \{(k-i) \bmod (L/2) : 0 \leq i, k \leq J-1\}, \quad \Delta = \{0, 1, 2, 4, 5\}.$$

CPM, APM, and GPM

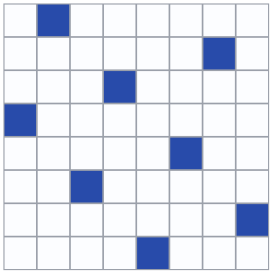
CPM



$P = 8,\ f(x) = x + 3$

strong algebra, limited randomness

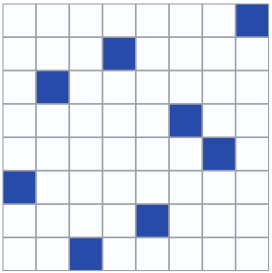
APM



$P = 8,\ f(x) = 5x + 3$

more randomness, still modular arithmetic

GPM



$\pi = (5, 2, 7, 1, 6, 3, 4, 0)$

maximum randomness, hard algebraic control

CPM → APM → GPM: more randomness

GPM → APM → CPM: easier algebraic control

Affine Permutation Matrices

Required constraints on F_i, G_j:

- Commute on Γ
- Avoid short cycles
- Full search is combinatorial.

Search strategy:

- Restrict to affine permutations on $\mathbb{Z}_P$:

$$f_i(x) = a_i x + b_i, \qquad g_j(x) = c_j x + d_j.$$

- Checks are P-independent.
- Sequential construction is fast[1].

[1] `github.com/kasaikenta/construct_apm_css_code`

Constructed Code Example

- Girth-8 $(3, 12)$-regular $[[9216, 4612, \leq 48]]$ with $P = 768$.
- Explicit weight-48 logicals $\Rightarrow d_{\min} \leq 48$.
- $d_X^{(\mathrm{lat})} = d_Z^{(\mathrm{lat})} = 48$ (proof omitted).
- No logical failures observed $\Rightarrow d_{\min}$ likely near 48.

Decoding Algorithm (BP + Post-Processing)

1. **Joint BP:** decode on H_X, H_Z using X/Z correlations.
2. **Trigger:** if unsatisfied checks are small, start post-processing.
3. **ETS library:** match odd-check pairs to precomputed $b = 2$ ETSs.
4. **PP:** solve the restricted residual on suspect set K.

$$\mathbf{s}_X = (H_Z)_K(\mathbf{x})_K \oplus (H_Z)_{\overline{K}}(\hat{\mathbf{x}})_{\overline{K}}.$$

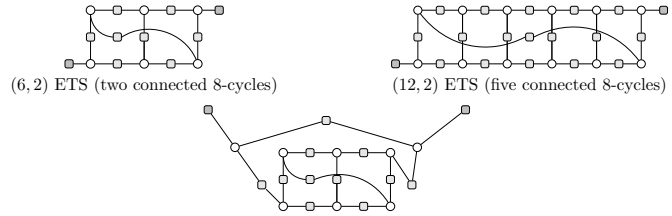
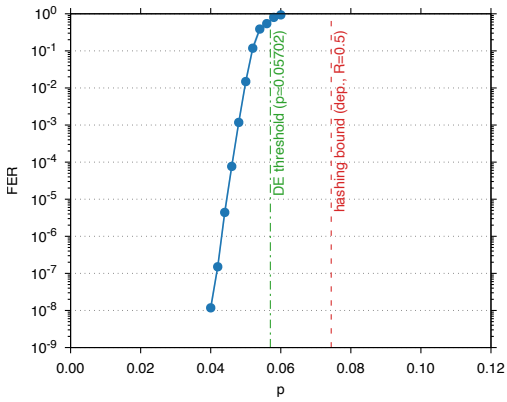


Figure 1: Representative ETSs used in the library.

Performance

- **Code:** girth-8, (3,12)-regular $[[9216, 4612, \leq 48]]$.
- **Decoding:** joint BP + PP reaches FER 10^{-8} at $p = 4\%$.
- **Degeneracy:** many weight-12 degenerate errors are corrected.
- **Benchmark:** BP aligns with DE (cycle-free, random non-orthogonal (3,12) code).



Conclusion

Main message

We built a quantum LDPC setting where classical LDPC design principles and know-how naturally remain useful.

- Degree distribution determines the BP threshold.
- APMs create room for randomness, large girth, and distance under CSS commutativity.
- BP approaches the classical sparse-graph benchmark.