

The Baseline Trap: One Hidden Setting Can Flip Quantum Decoder Comparisons

Tushar Pandey
Texas A&M University

QTMML 2026 extended abstract (submission 49)

Abstract

Benchmarks of quantum error-correction decoders, including the now-common “our learned decoder beats BP-OSD by a factor X ” claims, hinge on how the classical baseline is configured. We show the BP-OSD baseline’s accuracy is governed by a single hyperparameter, the belief-propagation min-sum scaling factor, whose optimum flips between noise models. At code-capacity on bivariate-bicycle (BB) QLDPC codes the optimum is near 0.5 and the value shipped as the `ldpc` library default (1.0) is catastrophic, raising the logical error rate by more than two orders of magnitude (robust across three held-out seeds, and large enough to invert code-distance scaling). At circuit level the order reverses: the optimum is the library default 1.0, and the code-capacity-optimal value is several times worse (1.7x, up to about 8x at the lowest shot counts), a reversal we confirm on two bivariate-bicycle codes at two distances ($[[72,12,6]]$ at $d = 6$ and $[[144,12,12]]$ at $d = 12$). We contribute a frozen, held-out, in-regime tuning protocol; a clock-independent operation-count cost metric (belief-propagation message-updates); and matched-accuracy cost comparisons. With fair tuning, surface-code BP-OSD matches the matching decoder MWPM in accuracy but costs orders of magnitude more to run (88x more BP operations from $d = 3$ to $d = 7$, about 7400x in wall-clock at $d = 7$); on BB codes, where MWPM cannot run, fairly-tuned BP-OSD shows below-threshold distance scaling that is statistically resolved across the full $d = 6/12/18$ family at circuit level. For the learned-decoder community, a neural network we train on $[[144,12,12]]$ has its apparent standing against BP-OSD swing by up to 65 times purely from the baseline’s tuning. The message is simple and actionable: tune your classical baseline in the regime you report, and state the tuning, before you compare.

1 Why this matters for QTMML

Learned decoders are a growing thread in quantum-plus-machine-learning, and their headline number is the margin over a classical belief-propagation with ordered-statistics (BP-OSD) baseline. Those comparisons are only fair if the classical baseline is tuned, and tuned in the noise regime being reported. We show that requirement is routinely unmet and that the optimal tuning does not transfer between regimes. QTMML has accepted decoder-benchmark contributions of exactly this shape (for instance a MaxSAT-vs-BP-OSD study at QTMML 2024), so the venue’s “benchmarking and complexity” and QEC scope is the right home for it. This is a methodological contribution rather than a new model.

2 Results

2.1 The optimal tuning is regime-dependent and flips between noise models

Changing only the min-sum scaling factor (ms), on BB codes at code-capacity $p = 0.03$ (three reported seeds), moves the logical error rate (LER) as shown in Table 1. The gross code $[[144,12,12]]$ alone shifts by a factor of 52. Under the default the LER increases with distance (it looks above threshold); fairly tuned it decreases with distance (below threshold). A held-out sweep over 0.3 to 1.0, repeated on three disjoint seeds (99, 100, 101), puts the optimum near 0.5 (LER about 0.0006)

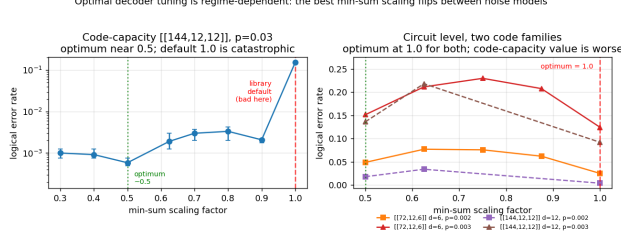


Figure 1: The best min-sum scaling flips between noise models. Left: code-capacity $[[144,12,12]]$ at $p = 0.03$, optimum near 0.5; library default 1.0 is catastrophic. Right: circuit level on two code families, optimum at 1.0.

with the whole 0.3 to 0.9 range good, while only the default 1.0 is catastrophic (LER 0.153 at all three seeds, std about 0.0007): a 250x, seed-independent cliff.

code	default ms=1.0	in-regime ms=0.625
$[[72,12,6]]$	0.121	0.037
$[[144,12,12]]$	0.152	0.0029
$[[288,12,18]]$	0.207	$< 8 \times 10^{-5}$ (0/12000 shots)

Table 1: Code-capacity BB LER at $p = 0.03$: the library default versus the in-regime optimum.

At circuit level (BB $[[72,12,6]]$, 6 rounds, $n = 2400$ per point) the order reverses (Table 2). The circuit-level optimum is the library default 1.0 (raising ms past 1.0 does not help: 1.0 and 1.2 tie, 1.5 is much worse), and the code-capacity-optimal range is the worst choice; at circuit level ms=1.0 also makes BP converge more often, the opposite of its code-capacity behavior. The same reversal holds on a second bivariate-bicycle code at a larger distance: at circuit level $[[144,12,12]]$ ($d = 12$, 12 rounds) the optimum is again 1.0 (LER 0.092 vs 0.218 at ms=0.625 for $p = 0.003$, two-proportion $z = 5.6$; and 0.004 vs 0.034 for $p = 0.002$, $z = 3.5$ but on small counts, 2 vs 17 failures in $n = 500$), even though there BP never converges (non-convergence about 1.0), so the scaling acts purely through the OSD stage (Figure 1).

min-sum scaling	0.5	0.625	0.75	0.875	1.0 (default)
circuit LER, $p = 0.002$	0.049	0.077	0.076	0.062	0.025
circuit LER, $p = 0.003$	0.151	0.211	0.230	0.208	0.124

Table 2: Circuit-level tuning curve on BB $[[72,12,6]]$ (6 rounds, $n = 2400$ per point). The optimum is the library default 1.0, and the code-capacity-optimal range is the worst choice.

2.2 Fair, cost-aware, matched-accuracy comparison

With the in-regime protocol, on the surface code BP-OSD is matched-or-better than MWPM in accuracy yet costs far more to decode: the BP operation count grows 88-fold from $d = 3$ to $d = 7$, and the wall-clock gap reaches about 7400x at $d = 7$, the price of a universal decoder over a special-purpose matching one. Under the library default, the same surface-code data would have said MWPM is significantly more accurate than BP-OSD at all three distances (two-proportion $z = 2.5, 3.6, 3.7$, the $d = 7$ gap surviving a Bonferroni correction), a conclusion that fair tuning overturns to matched. On BB codes, where MWPM cannot run at all (column weight 3, non-graphlike), fairly-tuned BP-OSD shows below-threshold distance scaling statistically resolved across the full $d = 6/12/18$ family at circuit level (every distance gap significant, z from 2.6 to 10.4).

2.3 Illustration for learned-decoder benchmarks

For the ML audience, we train a multilayer-perceptron decoder (syndrome to logical class, identical success metric to BP-OSD) on code-capacity $[[144,12,12]]$ and compare it on identical test data

p	MLP	BP-OSD def.	BP-OSD in-reg.
0.03	0.175	0.158	0.0027
0.04	0.339	0.369	0.010

Table 3: Learned MLP decoder versus BP-OSD on code-capacity $[[144,12,12]]$, identical test data.

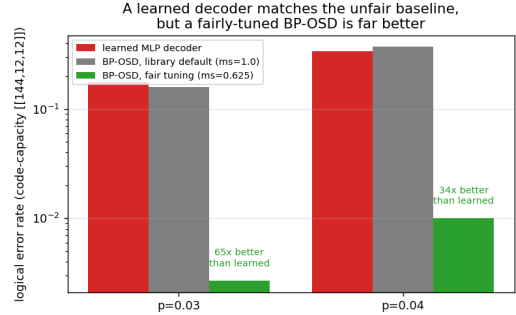


Figure 2: A learned MLP decoder matches the unfairly-tuned baseline; a fairly-tuned BP-OSD is far better.

to BP-OSD at the default ($ms=1.0$) and the in-regime tuning ($ms=0.625$), shown in Table 3. At $p = 0.04$ the network beats the default baseline (0.339 vs 0.369), which a paper could report as beating BP-OSD, but the in-regime baseline is 34 times better than the network; at $p = 0.03$ it ties the default while the in-regime baseline is 65 times better (Figure 2). This is decoder-agnostic: any decoder with logical error rate L is reported as the ratio $L_{\text{BP-OSD}}/L$ “better than BP-OSD,” and $L_{\text{BP-OSD}}$ moves with the baseline’s tuning, so the headline factor inherits that movement.

3 Methods

Decoders: MWPM (pymatching) and the universal belief-propagation family from `ldpc` (plain BP, BP+OSD at order 0 and combination-sweep, BP-LSD). Codes: rotated surface and the Bravyi et al. bivariate-bicycle family $[[72,12,6]]$, $[[144,12,12]]$, $[[288,12,18]]$ (arXiv:2308.07915). Noise: code-capacity bit-flip and circuit-level depolarizing; circuit-level BB uses the published edge-colored CSS syndrome construction (arXiv:2109.14609). Tuning is selected by grid search on a held-out seed (99, disjoint from the reported accuracy seeds 0, 1, 2) and frozen before reporting; we sweep the min-sum scaling in each regime separately (code-capacity on three held-out seeds; circuit level on independent multi-core chunks, $n = 2400$ per point). The cost metric is $op = iter \times nnz(H)$, the BP message-updates per shot, independent of clock speed and validated against wall-clock; it represents the BP stage and is most faithful where BP converges (code-capacity), with wall-clock reported where the cost moves to OSD (circuit level). The learned decoder in Result 3 is an MLP (torch). Statistics are pooled two-proportion z -tests; every record is git-hash, library-version, and seed stamped. Data and code are in the project repository; 32 unit tests and several independent adversarial review passes are in the git history.

4 Limitations

The dramatic tuning effect (two orders of magnitude) is a code-capacity result; at circuit level the effect is real but modest (1.7x to about 8x), and the circuit-level reversal, confirmed on two BB codes at two distances, rests on moderate shot counts ($n = 500$ to 2400 per point; the largest, about 8x, ratio rests on a low-count point), with both codes from the same BB construction rather than structurally distinct families; larger distances, shot counts, and a non-BB family would tighten it further. The learned-decoder illustration is at code-capacity; a feedforward MLP does not decode circuit-level QLDPC competitively (we measured LER 0.83 to 0.98 versus BP-OSD’s 0.03 to 0.21), which is itself why circuit-level learned decoders use recurrent or transformer architectures. A single CPU is used (the cost metric is clock-independent but not validated across architectures); energy is motivation only (no measured Joules); the circuit-level absolute LER is unvalidated against a published number, so we report scaling, not thresholds.

References

- [1] S. Bravyi et al. High-threshold and low-overhead fault-tolerant quantum memory. *Nature* **627**, 778 (2024). arXiv:2308.07915.
- [2] General-CSS syndrome-extraction circuit. arXiv:2109.14609.
- [3] Machine learning decoding of circuit-level noise for bivariate bicycle codes. arXiv:2504.13043 (an example of a learned-vs-BP-OSD comparison).
- [4] M. Noormandipour and T. Haug. MaxSAT Quantum Error Correction Decoders: a new formulation and benchmark study. QTML 2024 (a classical decoder-vs-BP-OSD benchmark at this venue).
- [5] Ambiguity Clustering, arXiv:2406.14527; MWPF, arXiv:2508.04969; ECCentric, arXiv:2511.01062.
- [6] Tools: stim, pymatching, ldpc, torch.