Computationally Pumped Bath Noise, Version 2:
A Multi-Scale, History-Dependent Noise Model for Quantum Circuit Simulation in PKTron

Zuhair Ahmed
Centre of Excellence for Technology, Quantum and AI (CETQAP), Rawalpindi, Pakistan
Correspondence: research@cetqap.org

Abstract

PKTron is an open-source, PyPI-distributed quantum circuit simulator aimed at education and low-resource research settings. This paper documents Computationally Pumped Bath Noise version 2 (CPBN v2), a noise subsystem added in PKTron 13.0.0 that models decoherence as the output of one or more classical "bath" processes whose instantaneous error-pump strength depends on recent circuit activity rather than being fixed in advance. We describe the governing update rule, the three execution paths through which CPBN v2 is exposed — exact shot-sampled Pauli trajectories, Pauli-frame sampling for Clifford circuits, and a closed-form combined-channel approximation — and the statistical and hardware-comparison tooling that accompanies it. We report internal consistency checks (the combined-channel path reproduces trajectory sampling to within 1×10^{-10}, and a bounded-parameter case reproduces the prior PKTron 12 noise model exactly) and single-machine runtime benchmarks across three backends. We are explicit about what this work does and does not claim: CPBN v2 is a phenomenological, software-level noise model intended for simulation, education, and algorithm-development use; it has not been validated against physical device noise spectra, and the benchmark numbers are sandbox measurements on one CPU, not a claim of general performance. We discuss the model's assumptions, limitations, and appropriate scope of use.

Keywords

quantum circuit simulation; non-Markovian noise; open quantum systems; Pauli channels; Clifford simulation; Pauli-frame sampling; open-source scientific software

1. Introduction

Realistic noise modeling is one of the persistent gaps between classical quantum-circuit simulators used in teaching and early-stage research, and the behavior of actual quantum hardware. Most simulators that include noise at all apply a fixed, time-independent error channel per gate: a depolarizing or dephasing channel with a constant probability, independent of what the circuit has been doing. This is a reasonable default, but it does not capture a feature widely reported in superconducting and trapped-ion devices: that error rates drift and respond to recent activity — heating from repeated driving, crosstalk that builds up with gate density, and leakage that accumulates and slowly relaxes. These effects are history-dependent and non-Markovian in the sense that the noise at a given moment depends on a decaying memory of what came before it, not only on the current instant.

CPBN (Computationally Pumped Bath Noise) is PKTron's attempt to give users a lightweight, classically simulable way to explore this regime without requiring a full open-quantum-systems solver. The first version (introduced in an earlier PKTron release and referred to here as CPBN v1/“v12”, after the PKTron release it shipped in) modeled a single bath whose pump strength depended on circuit activity. CPBN v2, shipped in PKTron 13.0.0, generalizes this in three ways: (i) it supports multiple baths with independent memory and pump parameters running in parallel (CPBNMultiScaleEnvironment), (ii) it exposes the model through three distinct, mutually cross-checked execution paths rather than one, and (iii) it ships with calibration and statistical-comparison utilities so that a user fitting the model to an activity record, or comparing two circuit histories, has tested tools rather than ad hoc scripts. This paper is a methods and implementation paper: it documents the model and its software realization, not a claim about any specific physical device.

2. The CPBN v2 Model

2.1 Single-bath dynamics

In CPBN, each qubit is associated with one or more classical scalar “bath” variables that track a leaky, activity-driven memory of recent circuit events. A bath component is parameterized by a memory coefficient (how slowly it decays between events), a pump strength (how strongly a circuit event drives it upward), and, for multi-qubit registers, a spatial decay coefficient governing how strongly activity on one qubit leaks into the bath of a neighboring qubit. At each circuit step the bath value decays toward zero by the memory factor and is then incremented by the pump term associated with any gate acting on, or spatially near, that qubit. The bath value is mapped to an instantaneous single-qubit Pauli error probability through a saturating function bounded above by a configurable ceiling, CPBNConfig.max_rate. This ceiling exists for two reasons: it keeps the model numerically well-behaved regardless of how much activity accumulates, and it gives a single, auditable parameter that recovers earlier model behavior as a special case — setting max_rate to 1 reproduces the unbounded CPBN v1 (“v12”) model exactly, which we treat as a regression test (Section 4.1) rather than a scientific result in its own right.

2.2 Multiple baths and CPBNMultiScaleEnvironment

A single memory/pump pair captures one timescale of drift. Reported noise behavior in real devices is rarely one timescale — there is typically a fast component (e.g., crosstalk that rises and falls within a few gates) and a slower component (e.g., gradual heating over a longer sequence). CPBN v2's CPBNMultiScaleEnvironment allows an arbitrary list of CPBNComponent objects, each with its own memory and pump_strength, to run in parallel on the same qubit register; the resulting error probability is a combination of all active baths rather than a single one. In our example configuration we use two components — one with high memory (0.99) and low pump (0.3), representing a slow-building background process, and one with low memory (0.5) and high pump (0.8), representing a fast, short-lived response — run side by side on a four-qubit Bell-chain circuit.

2.3 Three execution paths

CPBN v2 is deliberately exposed through three different simulation strategies so that users can trade exactness for speed, and so that the strategies can be cross-checked against one another:

- run_cpbn_trajectories — an exact, shot-by-shot sampler. For each simulated shot, a fresh Pauli error is drawn at each step from the current bath-determined probability, and the bath itself evolves independently of the sampled outcome (the bath is "outcome-independent": it is driven by which gates occur, not by which errors were sampled). This path is the ground truth against which the other two are checked, and it is also the only path that natively supports non-Clifford circuits.
- run_cpbn_clifford — a Pauli-frame sampling method restricted to the Clifford gate set (H, S, Sdg, X, Y, Z, ID, CX, CZ, SWAP). Because Clifford operations map Pauli errors to Pauli errors under conjugation, the error record for an entire shot can be tracked as a single evolving Pauli frame rather than a full density matrix or state vector, which is asymptotically cheaper for large, Clifford-heavy circuits (this is the basis for the 60-qubit benchmark in Section 5).
- run_with_cpbn — a closed-form approximation that collapses the time-dependent bath process into one combined, time-averaged Pauli channel applied per qubit, then runs the circuit once through PKTron's standard noisy-channel machinery (the same code path used for execute(..., noise=cfg, ...)). This is far cheaper than shot-sampling when only the marginal, not the trajectory-level, statistics are needed.

Because all three paths implement the same underlying bath model, PKTron 13.0.0's test suite checks them against each other: run_with_cpbn agrees with shot-sampled trajectories to within 1×10^{-10} in the bounded-rate regime (Section 4.1), which is the primary internal-consistency evidence for the implementation presented in this paper.

3. Calibration and Statistical Tooling

Because CPBN parameters (memory, baseline rate, pump gain) are not something a user is expected to guess correctly, PKTron 13.0.0 ships a companion module, pktron.cpbn_calibration, with two functions used in this study. simulate_series generates a synthetic bath-driven time series given an activity record and a set of candidate parameters, and fit_cpbn_series fits those parameters back out of an observed series using bootstrap resampling to report parameter uncertainty, rather than a single point estimate.

A second module, pktron.cpbn_stats, provides paired-comparison tools for the common experimental question "does condition A differ from condition B on the same qubits," given that CPBN is history-dependent and so paired designs are more informative than unpaired ones. paired_report computes signed before/after differences together with bootstrap confidence intervals and a sign-flip permutation test; we note explicitly, as a methodological point, that the alternative label-swap permutation test on absolute differences is degenerate for this design and is therefore not used. matched_control_experiment runs two specified activity histories (for example, two different circuit-execution orders) against a shared CPBNConfig and reports whether the bath-driven noise differs between them for matched qubits. We flag an important honesty caveat here: a zero result from

matched_control_experiment is a legitimate and expected outcome for configurations or histories where the chosen activity pattern does not meaningfully drive the bath apart — it is not a software defect, and should not be ignored or treated as noise.

4. Validation

4.1 Internal consistency checks

We treat two internal agreements as the core correctness evidence for this release, both run as part of PKTron's automated test suite (218 tests at the time of this release) rather than as one-off demonstrations:

- Backward compatibility: with CPBNConfig.max_rate = 1, CPBN v2's combined-channel output matches the unbounded CPBN v1 ("v12") model's output exactly, confirming that the v2 generalization is a strict superset of the earlier model rather than a replacement with different behavior at the boundary.
- Cross-path agreement: run_with_cpbn's closed-form combined channel matches the distribution produced by exact shot-sampled trajectories (run_cpbn_trajectories) to within 1×10^{-10}, for the bounded-rate case used in our example configuration.

We did not, in this study, validate CPBN v2's bath dynamics against measured noise-spectrum data from physical quantum hardware. The hardware_bridge module (Section 4.2) supports exporting PKTron circuits to Qiskit and running them on IBM backends via to_qiskit and ibm_runner, which makes such a comparison possible in future work, but no such comparison is reported here.

4.2 Hardware bridge

pktron.hardware_bridge provides build_matched_pairs (constructing circuit pairs that differ only in a controlled activity history), simulated_runner (a CPBN-noise simulated backend usable as a drop-in runner), analyze_matched_pairs, compare_counts, and format_comparison_report, together with save_results_json, a numpy-safe JSON serializer for result dictionaries. In the configuration we tested (a four-qubit register, four matched pairs, 1000 shots per circuit), these utilities executed end-to-end and produced a well-formed comparison report and a numpy-safe JSON artifact; we report this as a software-functionality check, not as an empirical finding about any noise effect size, since the specific pairs used here were chosen to exercise the code path rather than to isolate a physically meaningful effect.

5. Performance Benchmarks

We measured wall-clock execution time for three CPBN v2 backends using pktron.noise_benchmarks.run_noise_benchmarks, on a single CPU core in a development sandbox (a shared virtual machine with no guaranteed dedicated hardware or thermal isolation). We report these numbers as an indicative, single-run characterization of relative backend cost, not as a general

performance claim; absolute timings will vary across machines, Python versions, and system load, and we did not perform repeated trials or report variance.

Backend	Qubits	Gates	Shots	No-noise (s)	C
density_matrix	8	48	–	0.075	
trajectories	12	54	256	–	
clifford_pauli_frame	60	60	1000	–	

Two qualitative points follow from this table. First, the Pauli-frame Clifford path (clifford_pauli_frame) is, as expected from its tracking a single Pauli frame rather than sampling full trajectories or materializing a density matrix, markedly cheaper at larger qubit counts (60 qubits, 1000 shots, 0.020 s) than the density-matrix path is at a much smaller size (8 qubits, 0.159 s with CPBN noise). Second, CPBN noise roughly doubled the density-matrix backend's runtime relative to the noiseless case in this configuration (0.075 s → 0.159 s); we did not test whether this ratio holds at other circuit sizes or activity levels, and we do not generalize this single data point into a scaling law.

6. Worked Example

The configuration below illustrates the full v2 feature set on a small Bell-chain circuit (H on qubit 0, then a chain of CX gates), executed under a four-qubit CPBNConfig with baseline_rate = 0.002, max_rate = 0.1, pump_strength = 0.8, memory = 0.95, and spatial_decay = 0.25, with two additional CPBNComponent baths layered on top via CPBNMultiScaleEnvironment:

```
cfg = CPBNConfig(n_qubits=4, baseline_rate=0.002, max_rate=0.1,
                 pump_strength=0.8, memory=0.95, spatial_decay=0.25, seed=3)
qc = QuantumCircuit(4)
qc.h(0); qc.cx(0, 1); qc.cx(1, 2); qc.cx(2, 3)
comps = [CPBNComponent(memory=0.99, pump_strength=0.3),
         CPBNComponent(memory=0.5, pump_strength=0.8)]

run_cpbn_trajectories(qc, cfg, shots=500, components=comps)
run_cpbn_clifford(qc, cfg, shots=500)
execute(qc, noise=cfg, shots=500, seed=1)
```

All three calls return a shot-count distribution over the four-qubit outcome space; because the circuit here is entirely within the Clifford set, all three execution paths are applicable to it, which is why this example was chosen to demonstrate the cross-path comparison described in Section 4.1. The calibration and matched-control examples in Sections 3 and 4.2 follow the same pattern: a declared configuration, a fixed seed, and a result dictionary that can be serialized and compared, which is the basic experimental unit CPBN v2 is designed around.

7. Discussion, Scope, and Limitations

CPBN v2 is a phenomenological model: its bath variables are a convenient classical bookkeeping device for “recent activity,” not a derivation from a microscopic Hamiltonian coupling a qubit to an environment. This is a deliberate design choice — it keeps the model classically simulable at modest cost

— but it means CPBN v2's parameters (memory, pump strength, spatial decay, max_rate) should be understood as simulation knobs to be fit to a target behavior (via pktron.cpbn_calibration) or explored for algorithm-robustness testing, not as first-principles physical constants. We have not claimed, and do not believe the present evidence supports claiming, that CPBN v2 reproduces the noise spectrum of any specific physical quantum processor.

A second limitation concerns the qubit-ordering convention used when exporting to Qiskit via to_qiskit and executing on IBM hardware via ibm_runner. PKTron's internal qubit ordering is reversed relative to Qiskit's convention for these operations; the bridge documentation reports that this reversal is handled correctly in the export and run path, but any user comparing per-qubit CPBN behavior against per-qubit hardware behavior should independently verify the ordering for their specific circuit before drawing conclusions about which physical qubit corresponds to which simulated one.

Third, the benchmark figures in Section 5 are single-run, single-machine measurements and should not be read as asymptotic complexity results, even though the backends' underlying algorithms (density matrix, trajectory sampling, Pauli-frame tracking) do have known, different theoretical scaling behavior; we have not independently verified empirical scaling trends across a range of qubit counts in this study.

Fourth, the matched-control and paired-comparison tools (Sections 3 and 4.2) are designed to detect whether a history-dependent effect exists for a given configuration; a null result from these tools is informative but should not be mistaken for a general proof that history-dependence is absent from the model, since it may instead reflect a choice of activity pattern or configuration that does not strongly separate the baths.

8. Conclusion

CPBN v2, introduced in PKTron 13.0.0, extends PKTron's single-bath, history-dependent noise model to multiple parallel baths with independent timescales, exposes the model through three cross-checked execution paths (exact trajectory sampling, Pauli-frame Clifford sampling, and a closed-form combined channel), and ships with calibration and paired-statistics tooling intended to make history-dependent noise experiments reproducible rather than ad hoc. We verified backward compatibility with the prior model and agreement between the exact and approximate execution paths to within 1×10^{-10}, and we report indicative, single-machine timing figures for three backends. We have deliberately scoped this paper to the software and model-consistency claims we can support with the evidence available, and we identify validation against physical hardware noise spectra as the natural next step, made feasible by the existing Qiskit/IBM hardware bridge.

Availability

PKTron is distributed on PyPI. The CPBN v2 modules described here (pktron.cpbn, pktron.cpbn_calibration, pktron.cpbn_stats, pktron.hardware_bridge, pktron.noise_benchmarks) are included in PKTron 13.0.0 and later.

Acknowledgments

This work was carried out at the Centre of Excellence for Technology, Quantum and AI (CETQAP).

References

[1] Ahmed, Z. PKTron: A Quantum Circuit Simulator for Education and Low-Resource Research. Python Package Index. https://pypi.org/project/pktron/

[2] Breuer, H.-P., & Petruccione, F. (2002). The Theory of Open Quantum Systems. Oxford University Press.

[3] Aharonov, D., & Gottesman, D. (2008). Improved simulation of stabilizer circuits. Physical Review A, 70, 052328.

[4] IBM Quantum. Qiskit: An Open-Source Framework for Quantum Computing. https://qiskit.org

[5] Nielsen, M. A., & Chuang, I. L. (2010). Quantum Computation and Quantum Information. Cambridge University Press.