# Classical Simulation of Quantum Circuits with Pauli Propagation

## Low-Weight Heisenberg Evolution and Monte Carlo Error Certification

Steven Ma (ym432)
Supervised by Prof. Hamza Fawzi

MPhil in Data Intensive Science

July 2025

# Why Quantum Simulation?

**Physics and Computation: Simulating Nature**

- Predicting molecular properties (e.g., new catalysts, drugs) relies on simulating quantum systems.
- Many-body quantum systems are exponentially hard to simulate classically.
- The Hilbert space dimension for $n$ qubits: $2^n$.
- Classical brute-force methods (matrix diagonalization, exact statevector) fail for $n \gtrsim 35$.
- Feynman's vision: use quantum systems to simulate other quantum systems!

*Nature isn't classical, dammit, and if you want to make a simulation of Nature, you'd better make it quantum mechanical.*

*— R. P. Feynman (1981)*

# Quantum Advantage: A Moving Target

**Pushing the frontier: why strong classical simulation matters**

- **Quantum advantage** means solving a problem faster with a quantum device than any known classical method.
- The bar for quantum advantage rises whenever classical algorithms improve.
- Every advance in classical simulation—Pauli propagation, tensor networks, Monte Carlo—pushes quantum hardware to higher standards.
- Only by beating the best classical methods do quantum devices have practical value.

# Statevector Simulation: Brute-Force Approach

**Directly simulating the quantum state**

- Quantum state of $n$ qubits: $|\psi\rangle \in \mathbb{C}^{2^n}$.
- Every gate is a $2^n \times 2^n$ matrix acting on the statevector.
- Memory cost: must store $2^n$ complex numbers in RAM.
- **Limits:** Feasible for $n \lesssim 35$ qubits on today's hardware.

## Takeaway

Statevector simulation is exact and universal, but exponentially hard for large $n$.

# Tensor Network Methods: Exploiting Structure

**Efficient for low-entanglement quantum systems**

- Main idea: represent quantum states as networks of low-rank tensors (e.g., Matrix Product States, MPS).
- **Area law:** If entanglement is limited, only a small subset of the Hilbert space is relevant.
- Both space and time scales polynomially in system size, when entanglement is low.
- Enables classical simulation of weakly entangled systems with $n \sim 100$.

## Bottleneck

If entanglement grows rapidly (e.g., deep circuits, high dimensions), tensor network methods also become intractable.

# Summary: Where Classical Methods Fail

**Why new classical algorithms are needed**

- Statevector: exact, but exponential cost—fails for $n \gtrsim 35$.
- Tensor network: efficient for low-entanglement, fails for generic deep/highly entangled circuits.
- **Open challenge:** How can we simulate generic quantum circuits that generate strong entanglement?

# Schrödinger vs. Heisenberg: Two Pictures of Quantum Dynamics

**Who evolves? State or observable?**

- **Schrödinger picture:** Quantum state $|\psi(t)\rangle$ evolves in time, observables $O$ are fixed.
- **Heisenberg picture:** State is fixed, observable $O_H(t)$ evolves in time.

$$\begin{aligned} \text{Schrödinger:} \quad & i\hbar \tfrac{d}{dt} |\psi(t)\rangle = H(t) |\psi(t)\rangle \\ \text{Heisenberg:} \quad & i\hbar \tfrac{d}{dt} O_H(t) = [O_H(t), H(t)] \end{aligned}$$

Here, $H(t)$ is the **Hamiltonian** (total energy operator).
**Key Point:** Both pictures predict the same measurement results, but offer different perspectives and computational advantages.

# Why the Heisenberg Picture?

**Focus on what you want to measure**

- In many experiments and quantum simulations, we only care about the expectation value of an observable at the end.
- Heisenberg picture allows us to **evolve only the observable**, not the full quantum state.
- This is often much more efficient, because calculating the expectation value requires only the evolved observable and the initial state.

# Observables as Pauli Strings

**Pauli operator basis: the language of quantum circuits**

- Any $n$-qubit observable $O$ can be decomposed as a linear combination of Pauli strings:

$$O = \sum_{P \in \{I,X,Y,Z\}^{\otimes n}} a_P P$$

- Pauli strings: tensor products like $X_0 \otimes I_1 \otimes Y_2 \ldots$; they form an orthonormal basis for all $2^n \times 2^n$ operators.
- Quantum circuits map Pauli strings to other Pauli strings (possibly expanding the number of terms).

# Heisenberg Back-Propagation: The Key Idea

**How do Pauli strings evolve through a circuit?**

- In the Heisenberg picture, observables are conjugated by the quantum circuit $U$:

$$O_H = U^\dagger O U$$

- Each quantum gate acts locally, transforming Pauli strings to new combinations.
- By "back-propagating" $O$ through the circuit, we can express the evolved observable as a sum of Pauli strings.
- This sum can grow rapidly—but **most terms' contribution become negligible!**

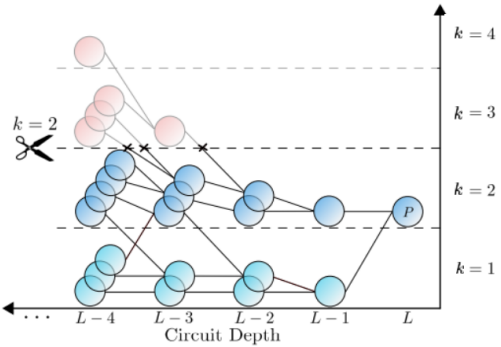



Figure 1: Schematic depiction of Pauli propagation with weight truncation ($k = 2$).
Adapted from [1].

# The Challenge: Exponential Growth of Pauli Terms

**Why naive back-propagation fails**

- Every non-Clifford gate can "mix" a Pauli string into a sum of several new strings.
- After many layers, the number of terms can become exponential in circuit depth.
- Storing all Pauli terms is no better than simulating the full statevector!
- We can keep only the important Pauli terms.

# Pauli Propagation Algorithm: Truncation in Action

**Pauli Propagation Algorithm (sketch)[1]**

1. **Low-weight truncation:**
   Given $O = \sum_P a_P P$, keep only terms with Pauli weight $\leq k$:
   $$O_L^{(k)} = \sum_{|P| \leq k} a_P P$$
2. **Iterative Heisenberg propagation:**
   For each layer $j = L, \ldots, 2$:
   - Back-propagate: $\tilde{O}_{j-1} = U_j^\dagger O_j^{(k)} U_j$
   - Truncate: $O_{j-1}^{(k)} = \sum_{|P| \leq k} \frac{1}{2^n} \mathrm{Tr}[\tilde{O}_{j-1} P] P$
3. **Final expectation:**
   After all layers,
   $$O_U^{(k)} = U_1^\dagger O_1^{(k)} U_1$$
   Compute expectation for $\rho_{\text{init}} = |\psi_{\text{init}}\rangle \langle\psi_{\text{init}}|$:
   $$\tilde{f}_U^{(k)}(O) = \mathrm{Tr}[O_U^{(k)} \rho_{\text{init}}]$$

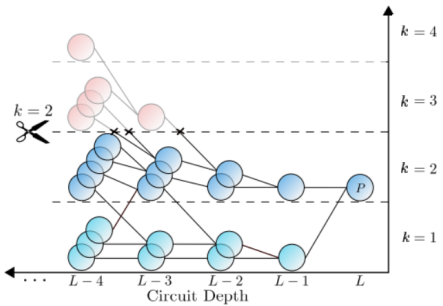



Figure 2: Truncated Pauli propagation ($k = 2$).
Adapted from [1].

# Why Does Pauli Propagation Work?

**Locally scrambling circuits rapidly spread out local information.**

- In a **locally scrambling circuit**, each layer mixes up local details so efficiently that any initially local observable quickly becomes a sum of many high-weight Pauli strings.
- Theorem means **high-weight terms can be safely truncated**, allowing efficient and accurate estimation of observables.

**Theorem (Mean Squared Error Bound)[1]:**

For any locally scrambling circuit $U$, Pauli weight $k \geq 0$, and observable $O$, the mean squared error (MSE) of truncated Pauli propagation is

$$\mathbb{E}_U \left[ \left( \Delta f_U^{(k)}(O) \right)^2 \right] = \mathbb{E}_U \left[ \left( f_U(O) - \tilde{f}_U^{(k)}(O) \right)^2 \right] \leq \left( \frac{2}{3} \right)^{k+1} \|O\|_{\mathrm{Pauli},2}^2,$$

where $\|O\|_{\mathrm{Pauli},2} = \left(2^{-n}\mathrm{Tr}[O^\dagger O]\right)^{1/2}$.

# Theorem: Efficient Simulation with Pauli Propagation

**Pauli propagation achieves any target accuracy with polynomial resources.**

- For any error $\epsilon > 0$ and failure probability $\delta > 0$, set truncation weight $k = O\left(\log \frac{1}{\epsilon\delta}\right)$.
- **Theorem (Complexity Guarantee)[1]:** For a locally scrambling circuit $U$ and observable $O$, the algorithm outputs $\alpha$ such that, with probability at least $1 - \delta$,

$$|\alpha - f_U(O)| \leq \epsilon \|O\|_{\text{Pauli},2}$$

- The runtime is at most

$$L\, n^{O(\log(1/(\epsilon\delta)))}$$

  where $L$ is the number of layers and $n$ is the number of qubits.
- If each layer acts on at most $D$ qubits, runtime further improves to $L\, O(D \log(1/(\epsilon\delta)))$.

# Motivation: Concrete Error for a Given Circuit

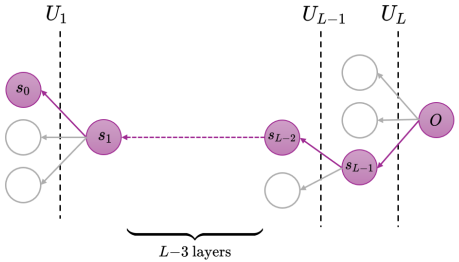
**Theory gives average-case bounds, but experiments need *actual* errors.**

- The mean squared error (MSE) theorem bounds expected truncation error over random circuits.
- For a given quantum circuit $U$, the actual MSE may be much smaller than this average-case bound.
- Monte Carlo sampling provides an unbiased estimate of the truncation error for a specific instance.

# Monte Carlo Estimation: Sampling Pauli Paths

**Estimate the true error via random Pauli path sampling.**

1. Expand observable: $O = \sum a_P P$
2. Back-propagate $O$ through the circuit layer by layer, expanding to a sum over Pauli strings at each step.
3. At each layer, **randomly sample** one Pauli string according to its squared amplitude.
4. Each complete history defines a **Pauli path**: $\gamma = (s_0, s_1, ..., s_L)$
5. The Monte Carlo estimator aggregates the contributions of $N$ such randomly generated paths.




Schematic: At $j$-th layer $U_j$, one Pauli string is sampled. Path terminates at $s_0$.

# Certified Error Estimate: Monte Carlo Guarantees

**How accurate is the Monte Carlo MSE estimate?**

- For any observable $O$ and locally scrambling $L$-layer circuit $U$, the Monte Carlo MSE estimator is **certified**[1]:
  - Draw $N = L\,\varepsilon^{-2}\log(1/\delta)$ independent Pauli path samples.
  - With probability at least $1-\delta$, the estimator $\alpha$ satisfies

$$\left|\alpha - \mathbb{E}_U\left[\left(\Delta f_U^{(k)}(O)\right)^2\right]\right| \leq \varepsilon\,\|O\|_{\mathrm{Pauli},2}^2$$

- This ensures the Monte Carlo error estimate is reliable, with fully controlled accuracy and confidence.

# Implementation Highlights: Bit-Mask Encoding

**Key engineering: efficient bit-level representation**

- Pauli strings are encoded as a single $2n$-bit integer: lower $n$ bits for $X$, higher $n$ bits for $Z$.
- Example: For $n = 3$, $Y_2 \otimes I_1 \otimes X_0$ (Qiskit little-endian):
  $key = 0b100101 = 37$, where $x = 0b101$ (bits 0,1,2), $z = 0b100$ (bits 3,4,5).
- Quantum gates correspond to fast bitwise operations on this key.
- Algorithm implemented in Python, fully compatible with `Qiskit`.

| $x$ | $z$ | Pauli operator |
|---|---|---|
| 0 | 0 | $I$ |
| 1 | 0 | $X$ |
| 0 | 1 | $Z$ |
| 1 | 1 | $Y$ |

Mapping between $(x, z)$ bit mask values and single-qubit Pauli operators.

# Benchmark: Staircase Random $SU(4)$ Circuit

**Why this circuit?**

- Each two-qubit gate: generic, maximally entangling—random $SU(4)$ means a "typical" two-qubit operation.
- Layers alternate in a staircase pattern, creating a locally scrambling circuit that rapidly spreads local information and entangles the system.
- Tensor network methods break down; a demanding test for any simulator.
- **Figure:** Shows $6 \times 6$ topology for one circuit repetition (60 layers); here, "repetition" is not "layer"—each gate is a layer.
- Benchmarks use circuit repetitions 1–5;
- Observable is $Z_{36}$, initial state is $|0\rangle^{\otimes 36}$.

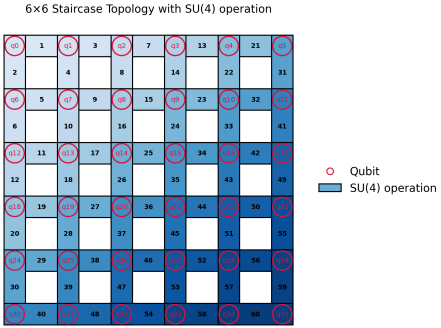



Figure 3: Staircase $6 \times 6$: each rectangle is a random $SU(4)$.

# Why KAK Decomposition?

**Universal hardware requires gate decomposition.**

- Hardware natively supports a limited set of gates.
- Any $SU(4)$ gate must be decomposed for execution.
- KAK (Cartan) decomposition gives an exact, minimal sequence:
  - 3 entangling gates ($R_{XX}$, $R_{YY}$, $R_{ZZ}$)
  - 12 single-qubit Euler rotations
- This enables efficient mapping and apples-to-apples benchmarking.

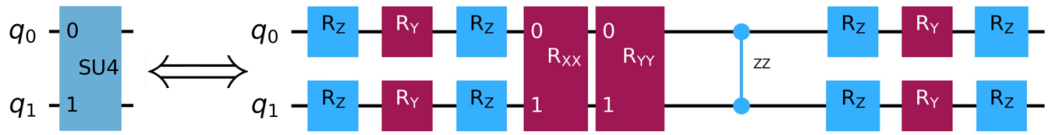



Figure 4: KAK decomposition: $SU(4)$ as 3 entangling and 12 single-qubit rotations.

# Truncation: Theoretical Bound vs. Practice

**Where to truncate in simulation?**

- **Theory:** Error bounds hold when truncating after each $SU(4)$ block.
- **Practice:** We truncate after every decomposed gate.
- **Why?** Truncating after each decomposed gate suppresses Pauli string growth, greatly reducing runtime.

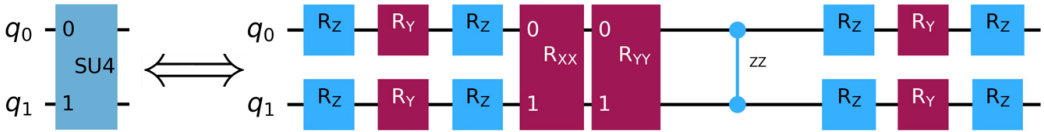



Figure 5: KAK decomposition: $SU(4)$ as 3 entangling and 12 single-qubit rotations.

# Truncation Weight $k$: Runtime Scaling

**Complexity scaling:**

- Runtime increases exponentially in truncation weight $k$, but only polynomially in circuit repetition $L$.
- $O(Ln^k)$ scaling.

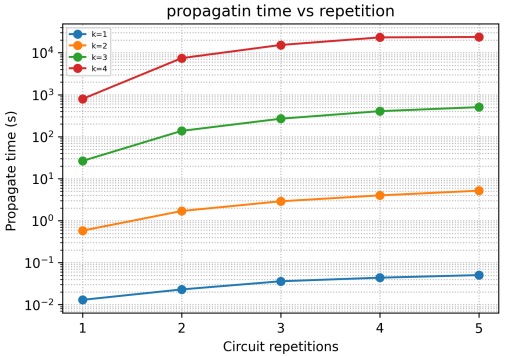



Figure 6: Runtime vs. repetitions for $k = 1, 2, 3, 4$.

# Monte Carlo: Pauli Weight Distribution

**Pauli weight reveals entanglement:**

- Each Monte Carlo sample produces a Pauli path ending in a final string $s_0$.
- Histogram of $|s_0|$ across samples quantifies the effective spread (entanglement) of the evolved observable.
- As circuit depth increases, high-weight Pauli strings dominate.

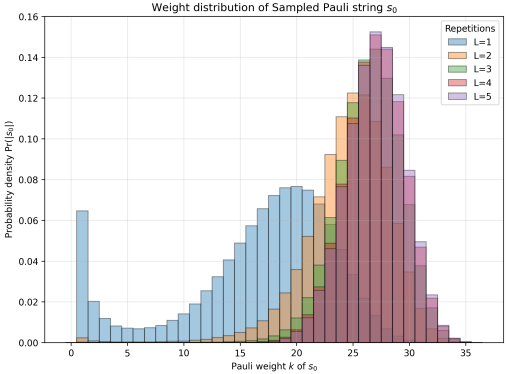



Figure 7: Weight distribution for $s_0$ across repetitions.

# Truncation MSE: Empirical vs. Theoretical Bound

**Benchmarking truncation error:**

- Estimated MSE from Monte Carlo (points), compared to analytic upper bound (dashed).
- For practical circuits, actual error is much smaller than the worst-case bound.
- Data demonstrates exponential decay of MSE with $k$.

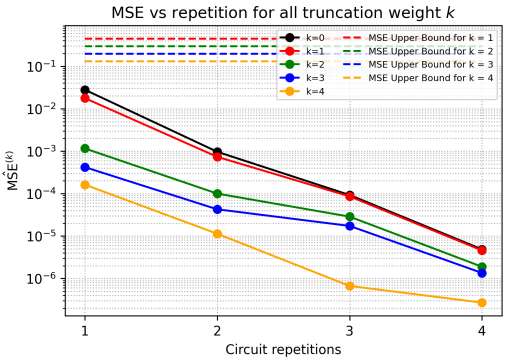



Figure 8: Truncation MSE vs. repetitions and $k$.

# Kicked Transverse-Field Ising Model (TFI)

**A paradigmatic 2D many-body quantum system**

- The TFI model describes $n$ interacting spin-$\frac{1}{2}$ particles arranged on a two-dimensional lattice, with each site hosting a single spin.
- The Hamiltonian is

$$H_{\mathrm{TFI}} = J \sum_{\langle i,j \rangle} Z_i Z_j + h_x \sum_i X_i,$$

  where $J$ is the Ising coupling, $h_x$ is the transverse field strength, and the sum over $\langle i,j \rangle$ runs over all nearest-neighbor pairs.
- **Ising interaction:** Nearest-neighbor spins tend to align along the $z$-axis.
- **Transverse field:** A uniform field along the $x$-direction induces quantum fluctuations.
- Importantly, the two terms do not commute: $[H_{ZZ}, H_X] \neq 0$, making the time evolution highly nontrivial and leading to rapid growth of entanglement, which challenges classical simulators.

# TFI Dynamics and Trotterization

**Simulating quantum many-body dynamics via quantum circuits**

- Schrödinger evolution:

$$i\frac{d}{dt}\left|\psi(t)\right\rangle = H_{\mathrm{TFI}}\left|\psi(t)\right\rangle, \quad \left|\psi(t)\right\rangle = e^{-iH_{\mathrm{TFI}}t}\left|\psi(0)\right\rangle$$

- **Exact evolution:** $e^{-i(H_{ZZ}+H_X)T}$ is not directly implementable since $H_{ZZ}$ and $H_X$ do not commute.
- **Trotterization:** Discretize time $T = L_t\tau$,

$$U(T) = e^{-i(H_{ZZ}+H_X)T} \approx \left[e^{-iH_X\tau}\, e^{-iH_{ZZ}\tau}\right]^{L_t}$$

- Each $e^{-iH_X\tau}$: parallel $R_X$ gates; each $e^{-iH_{ZZ}\tau}$: parallel $R_{ZZ}$ gates:

$$R_X(\theta) = \exp\left(-i\frac{\theta}{2}X\right), \quad R_{ZZ}(\theta) = \exp\left(-i\frac{\theta}{2}Z\otimes Z\right)$$

- Thus, simulating many-body quantum dynamics reduces to sequentially applying local quantum gates over the 2D lattice.

# IBM Eagle Heavy-Hex Topology

**127-qubit processor: native 2D lattice connectivity**

- IBM Eagle arranges 127 qubits in a heavy-hex graph.
- Each node is a qubit; edges indicate native two-qubit connectivity.
- The lattice is not fully regular: each qubit connects to 2 or 3 nearest neighbours.

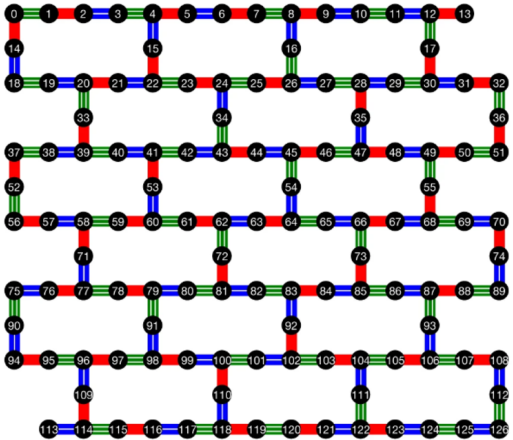



Figure 9: Heavy-hex topology of IBM's 127-qubit Eagle processor. Adapted from [2].

# Experimental Protocol: Kicked Ising Dynamics

**Implementation on IBM Eagle hardware and benchmarking setup**

- Initial state: $|0\rangle^{\otimes 127}$
- Observable: $Z_{62}$ (central qubit in the lattice)
- 20 Trotter steps: Each step
  1. Apply $R_X(\theta_h)$ to every qubit ($\theta_h$ scans field strength)
  2. Apply $R_{ZZ}(\theta_{ZZ})$ to all nearest-neighbour pairs ($\theta_{ZZ} = -\pi/2$)
  3. Hardware constraint: two-qubit gates are scheduled in three parallel colour layers (no overlap)

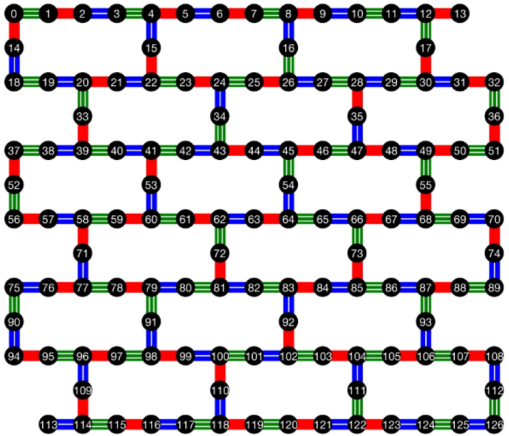



Figure 10: Heavy-hex connectivity and edge-colouring for parallel two-qubit gate scheduling. Adapted from [2]

# Results: Classical vs. Quantum Experiment

- Observable: $\langle Z_{62} \rangle$ after 20 Trotter steps, plotted as a function of $\theta_h$ (the transverse field strength).
- Blue points: IBM Eagle experimental data.
- Magenta dashed: MPS ($\chi = 1024$).
- Solid lines: Pauli propagation for $k = 3$ to 7.
- Pauli propagation matches the experiment for moderate $k$ values; MPS breaks down at large $\theta_h$.
- For $k = 7$, Pauli propagation is roughly $90\times$ faster than MPS [2], while maintaining higher accuracy, especially at strong entanglement.
- Notably, this circuit is not locally scrambling, yet Pauli propagation remains highly accurate.
- Belief-propagation tensor network [3] reach similar accuracy, but require custom adaptations; Pauli propagation is fully general.

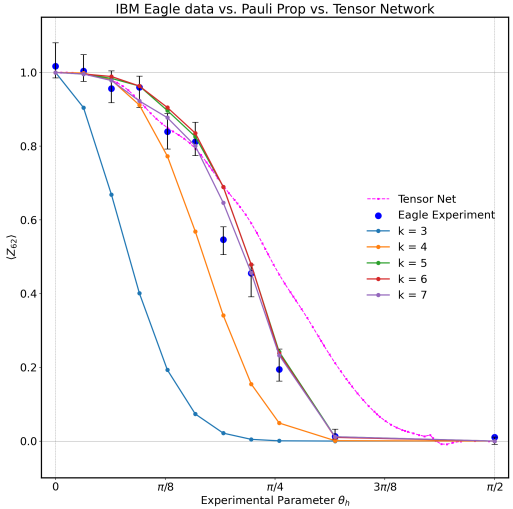



Figure 11: IBM Eagle experimental data and classical simulations. Data from [4].

# Further Research Directions

**Expanding the reach of Pauli propagation**

- **Quantum optimisation:**
  Fast estimation of cost functions in variational algorithms; accelerates parameter sweeps and hybrid quantum–classical routines, with direct applications to finance and logistics.
- **Noisy-device benchmarking:**
  By inserting realistic noise channels (e.g., depolarising, amplitude damping) between gates, Pauli propagation can predict hardware performance, evaluate error mitigation, and benchmark quantum devices against classical simulation.

# Conclusion

**Summary of findings**

- On $6 \times 6$ random $SU(4)$ circuits, low-weight truncation enables observable estimation far beyond statevector and tensor network methods.
- Monte Carlo error is much lower than the theoretical bound, even when truncating after each decomposed gate—not just full $SU(4)$ blocks.
- In IBM Eagle's kicked Ising experiment, Pauli propagation matches quantum hardware for $k = 7$, outperforming MPS in both speed and accuracy.
- Belief propagation tensor network approaches can achieve results comparable to Pauli propagation for specific connectivities such as the heavy-hex lattice, but Pauli propagation stands out for its generality and minimal structural assumptions.
- Crucially, strong performance persists even in non–locally scrambling circuits, highlighting broad applicability.

# References



A. Angrisani, A. Schmidhuber, M. S. Rudolph, M. Cerezo, Z. Holmes, and H.-Y. Huang, "Classically estimating observables of noiseless quantum circuits," 2024.


Y. Kim, A. Eddins, S. Anand, K. X. Wei, E. van den Berg, S. Rosenblatt, H. Nayfeh, Y. Wu, M. Zaletel, K. Temme, and A. Kandala, "Evidence for the utility of quantum computing before fault tolerance," *Nature*, vol. 618, pp. 500–505, 2023.
Open access.


J. Tindall, M. Fishman, E. M. Stoudenmire, and D. Sels, "Efficient tensor network simulation of ibm's eagle kicked ising experiment," *PRX Quantum*, vol. 5, p. 010308, Jan 2024.


Y. Kim, A. Eddins, S. Anand, K. X. Wei, E. van den Berg, S. Rosenblatt, H. Nayfeh, Y. Wu, M. Zaletel, K. Temme, and A. Kandala, "Evidence-for-the-utility-of-quantum-computing-before-fault-tolerance: data and code to reproduce figures." GitHub repository, 2023.
Accessed: 2025-07-18, CC-BY-SA-4.0 license.