Magic State Distillation From Quadratic Residue based CSS codes

Placeholder Subtitle

Mihir Talati, Leonardo S.P Velloso

December 9, 2025

Presentation Outline

Presentation Outline

- Background
  - Magic State Distillation Overview
  - Transversal Gates
  - Self-Dual Codes
  - Doubled QR Codes
  - Distillation Protocol
- Motivation and Approach
  - Methodology
  - Hypothesis
  - Yield
  - Scaling
- Simulation
  - Algorithm
  - Results
  - Analysis
- Conclusion
- Potential Future Work

Background

Magic State Distillation Overview

- Fault-Tolerant QC: In many stabilizer codes, Clifford gates are transversal while T is not
- Applying $T \longrightarrow$ spreads error $\longrightarrow$ **NOT FAULT-TOLERANT**
- Motivation:
  i) Universal Gate Set $\longrightarrow$ we want "net effect" of a T gate without actually using it
  ii) Magic state distillation is the process of "refining" imperfect magic states
  iii) Sequence of Pauli measurements and Clifford group operations generate less, but better magic states
  iv) Higher fidelity

Transversal Gatesets Background

Transversal Gate

A quantum operation $\mathcal{F}$ acting on m blocks (n qubits per block) is called transversal if it can be written as a tensor product of gates G_i, i.e

$$\mathcal{F} = \bigotimes_{i=1}^{n} G_i$$

where

- G_i acts on the i^{th} qubit of each block
- G_i acts on an m-qubit space (one qubit from each of the m blocks)
- No G_i ever touches more than one qubit in the same block

Transversal Gatesets Background

- Properties for Fault-Tolerant Quantum Computing:
  i) GPP – Applying gate avoids error blow-up inside the block
  ii) GCP – makes the codes' correctable errors behave nicely through the gate

Constraints Imposed by Transversal Gates

- **Eastin–Knill Theorem:** No QECC can realize a universal gate set using only transversal gates.
- Transversal gates preserve the structure of stabilizer codes: they map Pauli errors to Pauli errors (GCP).
- Logical gates implementable transversally are restricted to a **finite subgroup** of the Clifford hierarchy.
- Non-Clifford logical gates (e.g. T) cannot be implemented transversally in standard codes $\Rightarrow$ need magic states.

CSS Code Formalization

- Built from two classical linear codes

$$C_Z \subset C_X \subset \{0,1\}^n.$$

- Encodes $k = \dim(C_X) - \dim(C_Z)$ logical qubits into n physical qubits.
- Logical basis states are uniform superpositions over cosets:

$$|x + C_Z\rangle.$$

- Error structure splits cleanly through stabilizers:
  - C_Z corrects bit-flip (X) errors
  - $C_X$corrects phase-flip (Z) errors
- Useful in distillation: Pauli-type noise handled independently, supports some Clifford Gates

Self-Dual Codes

- A CSS code is **self-dual** if its classical code satisfies

$$C = C^{\perp}$$

- Then the X and Z-stabilizers come from the same linear code
- This creates strong X/Z symmetry in the error structure
- Such symmetry allows more Clifford gates to be transversal (e.g., H, sometimes CNOT), which is useful for distillation

Doubled QR Codes: Construction Summary

- Begin with two ingredients:
  - a **self-dual, doubly-even** (Hamming weight is multiple of 4) CSS code (gives X/Z symmetry and transversal Cliffords)
  - a **QR-derived doubly-even** CSS code (provides high distance)
- Apply a **doubling map** to combine these two codes
- The resulting code is **weakly triply-even** (mulitple of 8), enabling a transversal logical T gate.
- Final product: a family of $[n, 1, d]$ codes with high distance and lower qubit overhead for transversal Clifford and T

Codes Diagram

Bravyi–Haah Magic State Distillation

- Uses **triorthogonal** (or weakly triply-even) CSS codes to distill high-fidelity $|T\rangle$ magic states
- Input: multiple noisy copies of $|T\rangle$ with physical error rate p
- Protocol applies only **Clifford operations** and **Pauli measurements** on the encoded blocks
- Output: a smaller number of magic states with error rate suppressed to $O(p^k)$ where k depends on the code (Bravyi–Haah has $k \geq 3$)
- Assumptions: transversal Clifford gates available; code satisfies triorthogonality (or weak triply-even structure) for logical T
- Target: preparation of high-fidelity logical $|T\rangle$ states for universal FTQC.

Motivation and Approach

Bravyi-Haah Protocol for TE* and triorthogonal codes

- Same BH protocol applied to each specific QR-based TE* code.
- Inputs:
  - H_X matrix (rows = X stabilizers),
  - Logical-Z vector $z_{\log}$,
  - Physical error rate p on each T-state,
  - Noise model: i.i.d. Z noise for magic state injection.
- For each code:

$$s(p) = \Pr[H_X e^T = 0], \qquad p_{\text{out}}(p) = \Pr[z_{\log} \cdot e = 1 \mid \text{accepted}]$$

- We compute these numerically per-block.

Hypothesis

- TE* / QR-based codes have **better finite-size overhead** than:
  - Standard BH triorthogonal codes,
  - Generic doubled self-dual codes.
- Due to:
  - High distances at small n,
  - Structure inherited from QR code weight distributions,
  - Particularly low-weight X-checks satisfying mod-8 conditions.
- Anticipated result:
  - Better yield/overhead for $n \lesssim 30 \to 100$ (depending on how many we can simulate),
  - But asymptotic exponent still $\gamma \to 2$.

Distillation Yield

- Yield quantifies "magic states out per magic state in":

$$Y(p) = \frac{k \cdot s(p)}{n}$$

- For Jain–Albert codes:
  - Typically $k = 1$ so $Y = s(p)/n$.
  - Small and medium n have surprisingly high yields due to small block sizes.
- Comparison baseline:
  - Bravyi–Haah triorthogonal families ($n = 3k + 8$),
  - Self-dual doubled families used in prior constructions.

Scaling With Code Length

- Key theoretical fact from Jain–Albert:

$$d(n) \approx \Theta(\sqrt{n})$$

  for both TE* and triorthogonal families constructed.
- For a BH-style distillation:

$$p_{\text{out}}(p) \sim Cp^{\alpha}, \quad \alpha \approx d_Z$$

  where d_Z = minimum weight undetected Z logical error.
- Thus,

$$\alpha(n) \approx \Theta(\sqrt{n})$$

- But:

$$\gamma_n = \log_{\alpha}(n/k) \to 2$$

  meaning asymptotically the codes do not beat BH's 1.585 exponent.
- However: **finite-size performance may be significantly better**.

Simulation

Algorithm Overview [WIP]

- For each code:
  1. Import H_X and logical-Z.
  2. Sample error patterns $e \sim \text{Bernoulli}(p)^n$.
  3. Check acceptance: $H_X e^T = 0$.
  4. For accepted blocks, compute logical parity $z_{\log} \cdot e$.
- Metrics:

$$s(p) = \frac{\text{accepted}}{N}, \quad p_{\text{out}}(p) = \frac{\text{harmful and accepted}}{\text{accepted}}, \quad Y = \frac{s(p)}{n}$$

- Y is a proxy for approximate per code distillation yield
- Complexity:
  - Monte Carlo: $O(Nrn)$ for r stabilizers.
  - Exhaustive enumeration for small n to compute α exactly.

Pretty Graphs

- Placeholder description of the simulation setup and noise models.
- Mention computational tools or libraries anticipated for the study.

Contextualizing Results

- Placeholder interpretation of simulated performance metrics.
- Discussion points comparing outcomes to expectations or baselines.

Conclusion

Hypothesis vs Results

- Placeholder statement of the working hypothesis before analysis.
- Criteria used to judge success or failure.

Potential Future Work

What's next?

- Placeholder list of follow-up experiments and protocol refinements.
- Suggestions for code design or distillation strategy improvements.