Logical Qubits and Quantum Error Correction

Team: The Ducklings
Members: Aryan Malhotra, Richard Davis, Jamie Gill, Pratik Kulkarni

Error Correction: A Primer

The Classical Picture

- Bits can flip due to many reasons during transmission
  - Electromagnetic interference (EMI)
  - Hardware aging
  - Cosmic rays(!)
- Leads to incorrect data being received

The Quantum Picture

- Qubits are also susceptible to errors, but…
  - More kinds of errors (bit *or* phase flip)
  - Extremely sensitive (more so than classical bits)
- Qubits can decohere due to any source of environmental noise, and maybe even spontaneously

Working Around Noise

WORK

$$\underbrace{E_{CZ} \gtrsim E_{mover} \sim E_{unpaired}}_{\text{Z-biased}} > E_{sitter} \sim E_{1-qubit,local} > E_{1-qubit,global}$$

WORK

f

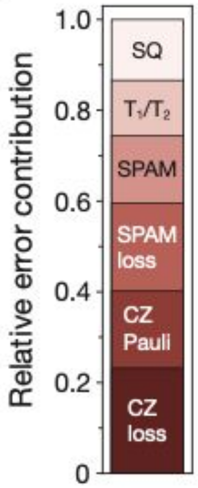


TABLE I

DPQA NOISE MODEL IMPLEMENTED IN QISKIT.

Operation	Noise channel	Strength	Circuit symbol
local 1q U	depol.	$p = 4e - 3$	U lue
global 1q U	depol.	p =4e-4	U gue U gue
atom move	Pauli	[3e-5, 3e-5, 3e-3] †	mve
CZ-spectator	Pauli	[5e-4, 5e-4, 2.5e-3] †	spe
global CZ	Pauli	[1.5e-3, 1.5e-4] ‡	set existing CZ gate as noisy
measurement	Pauli	[6e-3, 0, 0] †	spam

†) $[p_x, p_y, p_z]$, ‡) two-qubit Pauli channel has 15 terms, see Methods. 1st value is for p_{IZ},p_{ZI},p_{ZZ}, 2nd value is for the remaining 12 terms.

Table I summarizes the noise channels for the DPQA con-

github.com/QuEraComputing/bloqade-circuit/blob/14fdf7aa9f0aa1e2476b23

bloqade-circuit / src / bloqade / qasm2 / dialects / noise / model.py

Code Blame 278 lines (236 loc) · 12.6 KB

```
class MoveNoiseModelABC(abc.ABC):

    local_px: float = field(default=4.102e-04, kw_only=True)
    """The error probability for a Pauli-X error during a local single qu
    local_py: float = field(default=4.102e-04, kw_only=True)
    """The error probability for a Pauli-Y error during a local single qu
    local_pz: float = field(default=4.112e-04, kw_only=True)
    """The error probability for a Pauli-Z error during a local single qu
    local_loss_prob: float = field(default=0.0, kw_only=True)
    """The error probability for a loss during a local single qubit gate

    local_unaddressed_px: float = field(default=2.000e-07, kw_only=True)
    """The error probability for a Pauli-X error during a local single qu
    local_unaddressed_py: float = field(default=2.000e-07, kw_only=True)
    """The error probability for a Pauli-Y error during a local single qu
    local_unaddressed_pz: float = field(default=1.200e-06, kw_only=True)
    """The error probability for a Pauli-Z error during a local single qu
    local_unaddressed_loss_prob: float = field(default=0.0, kw_only=True)
    """The error probability for a loss during a local single qubit gate

    global_px: float = field(default=6.500e-05, kw_only=True)
    """The error probability for a Pauli-X error during a global single q
    global_py: float = field(default=6.500e-05, kw_only=True)
    """The error probability for a Pauli-Y error during a global single q
    global_pz: float = field(default=6.500e-05, kw_only=True)
    """The error probability for a Pauli-Z error during a global single q
    global_loss_prob: float = field(default=0.0, kw_only=True)
    """The error probability for a loss during a global single qubit gate

    cz_paired_gate_px: float = field(default=6.549e-04, kw_only=True)
    """The error probability for a Pauli-X error during CZ gate operation
    cz_paired_gate_py: float = field(default=6.549e-04, kw_only=True)
    """The error probability for a Pauli-Y error during CZ gate operation
    cz_paired_gate_pz: float = field(default=3.184e-03, kw_only=True)
    """The error probability for a Pauli-Z error during CZ gate operation
    cz_gate_loss_prob: float = field(default=0.0, kw_only=True)
    """The error probability for a loss during CZ gate operation when two

    cz_unpaired_gate_px: float = field(default=5.149e-04, kw_only=True)
    """The error probability for Pauli-X error during CZ gate operation w
    cz_unpaired_gate_py: float = field(default=5.149e-04, kw_only=True)
    """The error probability for Pauli-Y error during CZ gate operation w
    cz_unpaired_gate_pz: float = field(default=2.185e-03, kw_only=True)
    """The error probability for Pauli-Z error during CZ gate operation w
    cz_unpaired_loss_prob: float = field(default=0.0, kw_only=True)
    """The error probability for a loss during CZ gate operation when ano
```

WORK

WORK

Solutions to Quantum Decoherence

Increase Coherence Time

- Pros:
  - Reduces need for physical qubits
  - Can work with physical qubits more directly
- Cons:
  - Extremely difficult
    - Difficulty scales quickly with number of qubits
    - Extreme conditions need to be maintained

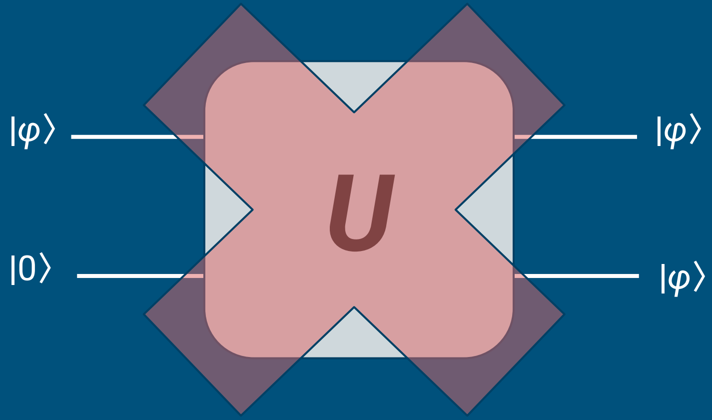
Quantum Error Correction

- Pros:
  - Much easier than increasing coherence time
  - Creates dependable logical qubits
- Cons:
  - Requires more physical qubits as overhead
  - Still pretty difficult

Quantum Complications

You can just copy classical bits...

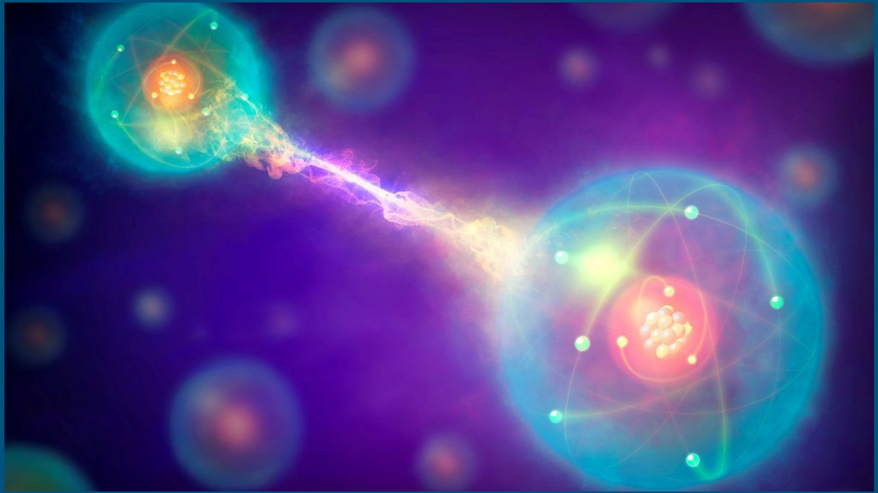
...but you can't copy qubits!



No-Cloning Theorem

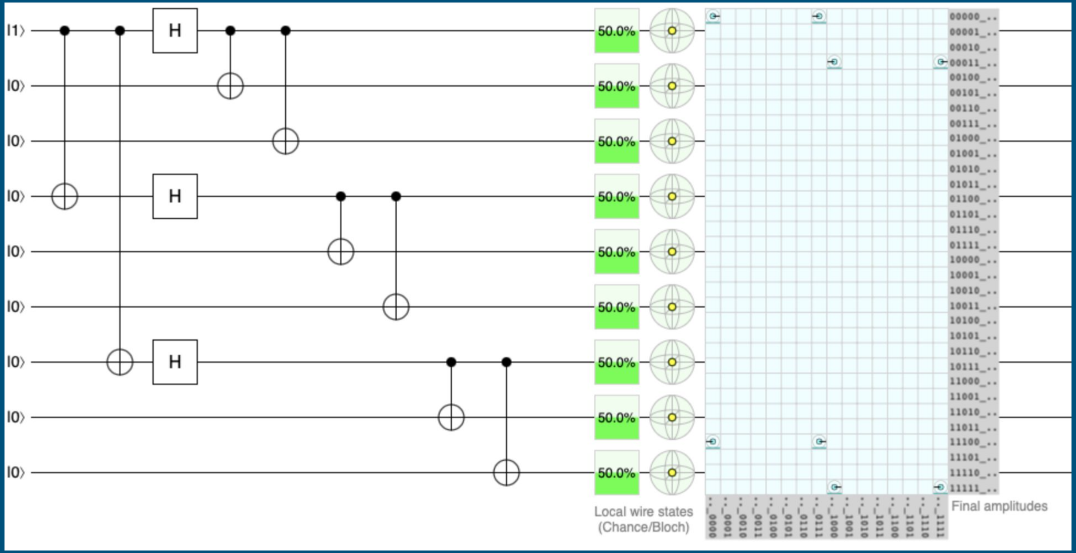
Now What?

Idea:
Use entanglement.



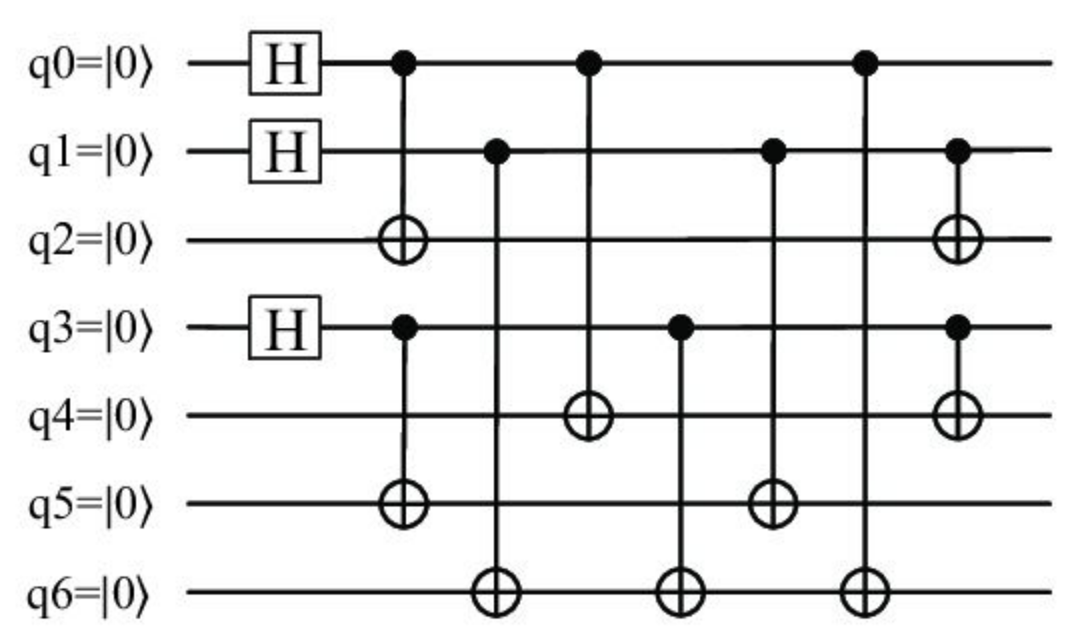
A Simple Example

- The Shor code was the first quantum error-correcting code.
- Uses nine physical qubits to encode one logical qubit
- Can correct both bit-flip errors and phase-flip errors
  - Sufficient to correct arbitrary bit errors



Can We Do Better?

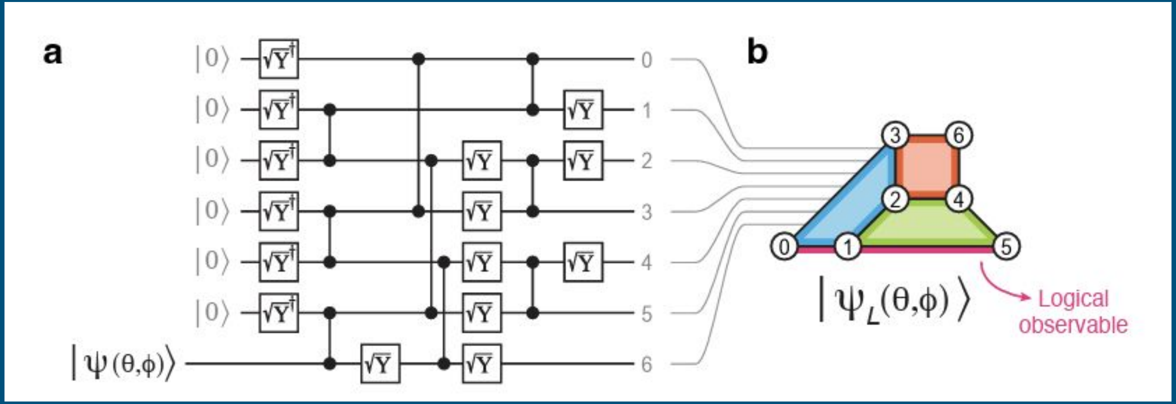
- The Steane code is also able to encode one logical qubit and fix arbitrary errors, but with only *seven* physical qubits instead of nine!
- Constructed from [7, 4, 3] classical Hamming code
- Considered a “color code”



Can We Do Better...Better?

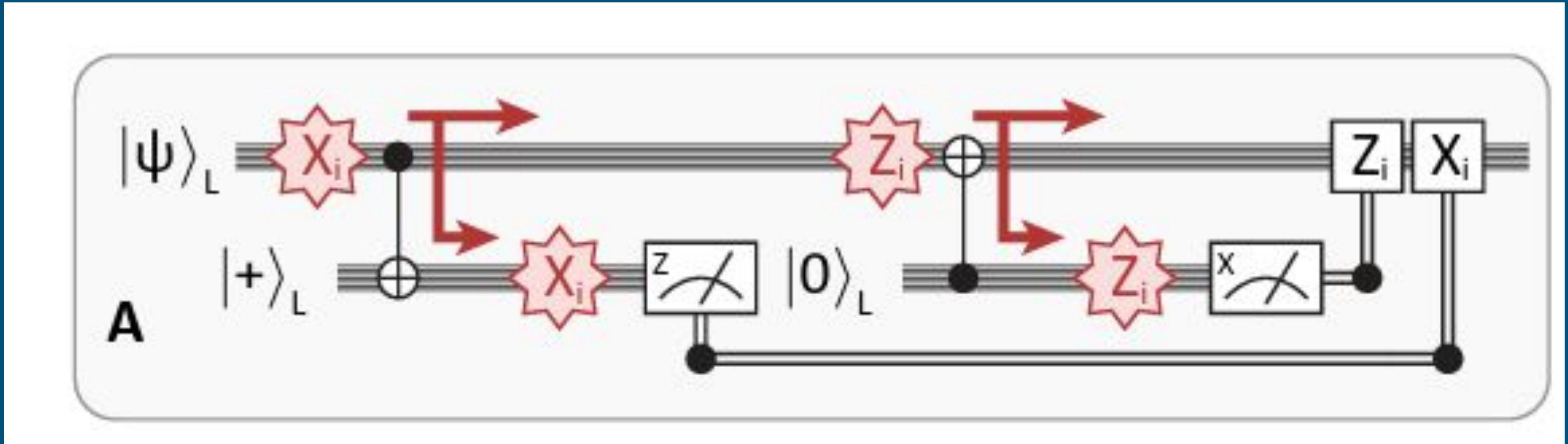
- The textbook Steane code circuit is kind of lengthy
- Bigger circuits are more expensive and provide more opportunity for decoherence
- Can we make a smaller circuit that implements the Steane code?

Here's one idea...

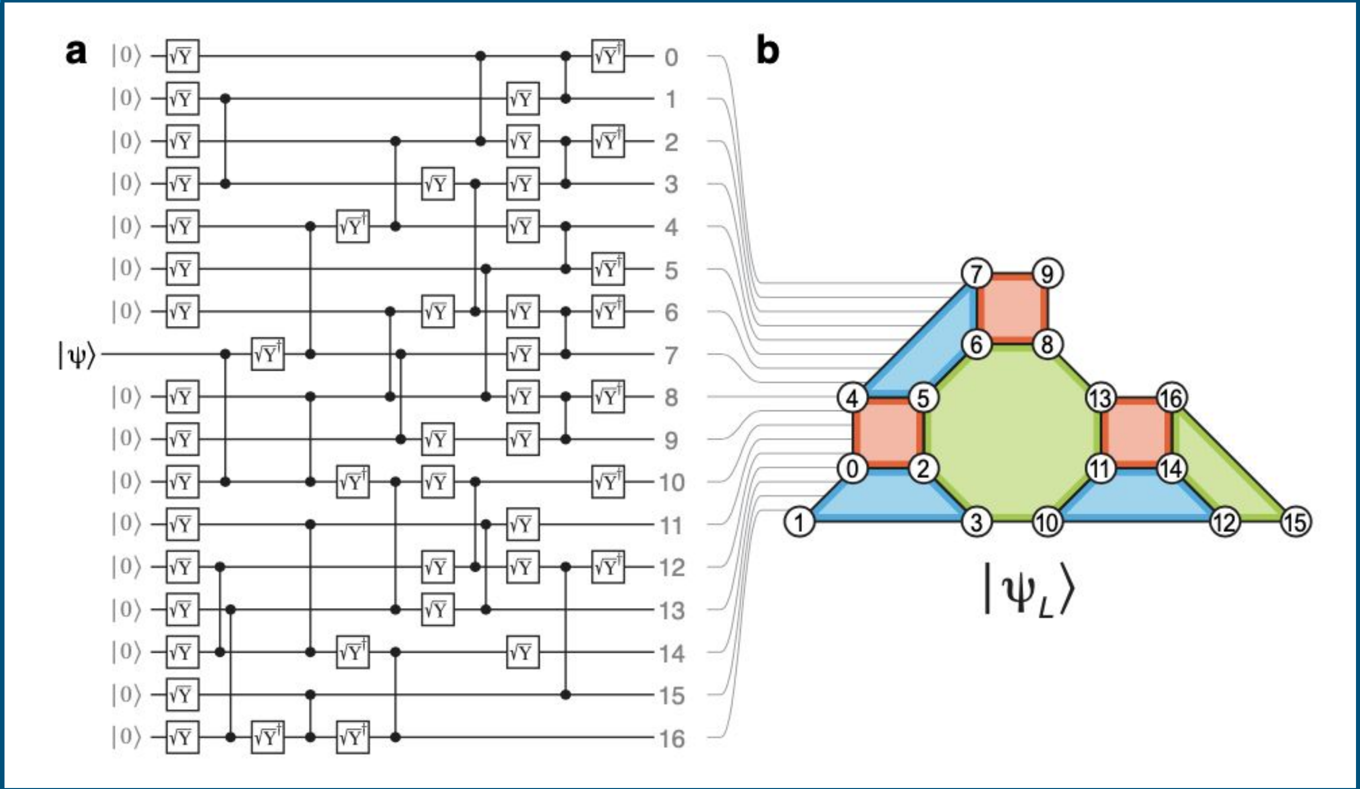


...How Do We Know?

- The circuits previously discussed allow us to encode seven physical qubits into one logical qubit
- We need another circuit leveraging properties of our encoding to actually detect and correct errors
- Luckily, Steane helps us here too



Extending to distance 5 -> [17, 1, 5]



Magic State Distillation -> Tsim

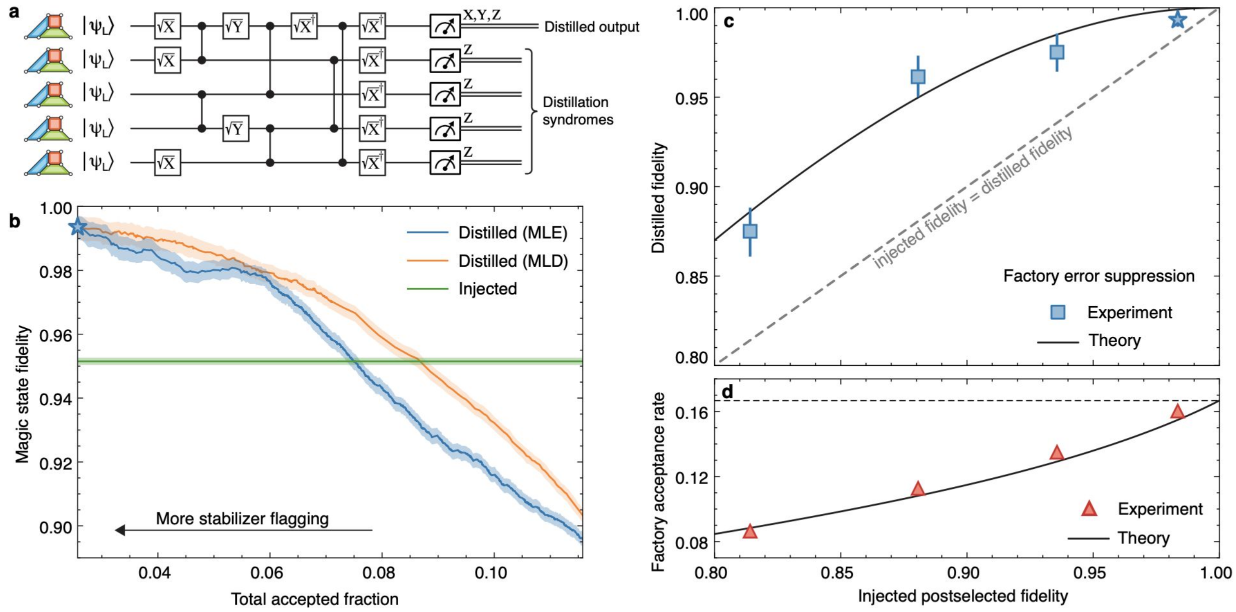


Figure 3. **5-to-1 magic state distillation. a,** Magic state distillation circuit based on the [[5,1,3]] code (distillation code)

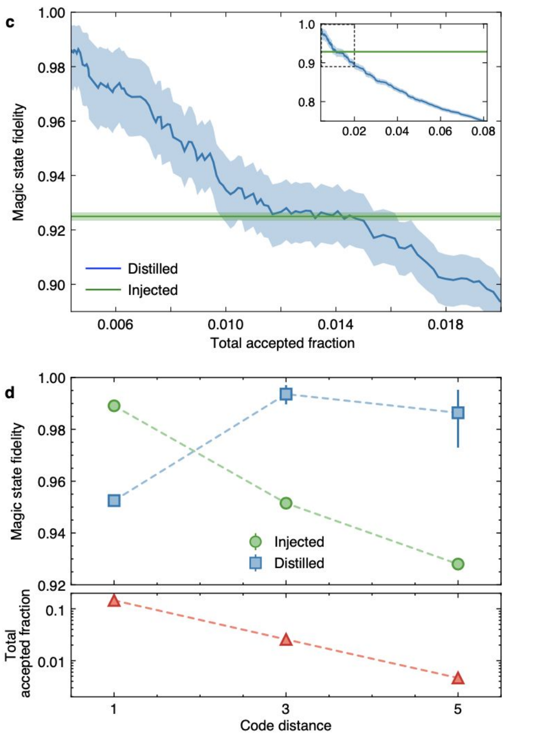


Figure 4. **Scaling of distillation as a function of data code distance. a,** $d = 5$ encoding circuit. **b,** $d = 5$ color code stabilizers. **c,** Output magic state fidelity for $d = 5$ distillation (blue) as a function of the total accepted fraction, again showing improvement over the input magic state fidelity (green). Extended range shown in inset. **d,** Injected (green circles) and distilled (blue squares) magic state fidelity with total acceptance fraction (red), when performing full stabilizer and factory postselection, all as a function of code distance.

Did We Do Better?

- To test the fault-tolerance of our setup, we injected random gate flips (X, Y, Z) into our system and tried to correct them
- We can simulate introducing noise to our system using PyQrack and Stim
- Tried three different error rates: 5%, 25%, 60%; 200 shots per error rate
- Our error rate dropped from 20% to 2% with Steane QEC, just by guessing the error to be the most likely flip
- Without QEC, significant portion of runs would be wasted (scaling with error probability), but QEC allows to salvage most of them