Project Proposal — Programming Track

Title. A surface-code simulator with a Qiskit front end and MWPM decoding

What I'm doing. I'm constructing a compact quantum error correction simulator, authoring circuits in Qiskit to facilitate grading and primarily using Stim for large sweeps.

I'll create stabilizer circuits, sample some syndromes, decode using PyMatching's MWPM and plot logical error vs physical error with bootstrap confidence intervals.

I chose this because the scope is realistically semester-suitable, but sufficiently thorough to teach me tangible fault-tolerance mechanics.

Why me. I'm currently an Art Unit 2613 USPTO Patent Examiner, so I will maintain assumptions and steps documented so a secondary reader can replicate results without guessing.

My most recent work at JHU trained me to keep pipelines reproducible through fixed seeds and all commands used for any figure saved in a log.

Scope. Rotated surface-code memory experiments at distances d \in {3,5,7,9}.

The noise to be induced first is circuit-level depolarizing and readout error; secondary simple biased-Pauli case to evaluate threshold shifts.

The decoders available to start with are a small local baseline decoder that I have implemented plus MWPM via PyMatching.

The command line interface will involve sweeps over (p, d, \text{rounds}, \text{shots}), deterministic RNG, and CSV logs with metadata and git short SHA.

Qiskit first, Stim for speed. All plots will be replicable using Qiskit and Aer noise models.

For speed of throughput I will export detection events to Stim and parity-check smaller cases with Aer before scaling up the number of shots.

Plan.

Week 1: Qiskit repetition code import, unit tests on front end, pinned seeds.

Week 2: Qiskit noise model settings, Aer reference runs, Stim exporting capabilities, Aer parity checks.

Week 3: MWPM wired up via documentation, completed baseline decoder, first curves at d=3.

Week 4: curves at d=5,7 added, bootstrap CIs applied, CLI UX tightened.

Week 5: curves at d=9 included, biased noise study added, runtime profiling done.

Week 6: All sweeps and figures finalized and paper drafted with five pages total.

Week 7: Polished README documentation, Colab demo and packaged.

Outputs. Public facing repository with /src (qiskit_frontend, stim_backend, decoders), /experiments, /figs, /tests and notebooks (qiskit_demo.ipynb = one-click Colab).

Every figure will have a saved command and fixed seed logged in the CSV header.

The paper will contain methods, experiments, results, limitations and future directions.

Risks and responses.

If large-d runs are slow I will trim rounds/shots and use Stim while keeping a Qiskit-only route open for every figure to limit issues.

If my threshold estimate fluctuates I will consolidate my shots to crossing points and document the CI in conjunction with the point estimate.

If MWPM struggles to come together I'll benchmark it on repetition codes with property testing before implementing it on surface codes.

References (starter). Dennis–Kitaev–Landahl–Preskill 2002; Fowler et al. 2012 surface codes; classical processing Fowler–Whiteside–Hollenberg 2012; Stim Gidney 2021; PyMatching Higgott 2021.

Evidence of process to help humans determine authorship: keep a Google Doc or git history that documents drafts over time with commit messages corresponding to figures + a short lab-notebook for design decisions made along the way. These pieces of evidence matter because detectors are not necessarily definitive.