Project Proposal: Enhancing a simulator for quantum stabilizer circuits

October 10, 2024

1 Introduction

Quantum programs are written using logical qubits. However, due to limitations on physical quantum computers, in order to achieve a single logical qubit, several data qubits are required. This is because during execution the individual qubits are subject to lots of different types of noise. Quantum error correction involves having qubits dedicated to measuring, and by measuring the parity of the qubits, we can detect errors. By running a decoder that runs classically, one can calculate which data qubits have errors, and then compensate for this in the final classical measurement of the system.

To ensure that the error correction code works correctly simulation is done. Stim is a simulator for quantum error correction code that runs faster than other simulators.

1.1 Description

The project aims to design and write an application that can receives as input a quantum error correction code that has been represented graphically, and, given the input, outputs the corresponding Stim python script. The produced application should act as an abstraction for being able to use a high speed stabiliser without understanding how Stim works directly. Understanding the dissertation will not require the reader to have an in-depth understanding of quantum error correction.

2 Starting Point

I am in the process of familiarising myself with quantum error correction as well as learning more about quantum computation in general. No design decisions have been made nor code written.

3 Project Implementation

1. Familiarise myself with different error correction codes and how Stim works
2. Manually write out the Stim code for around five different error correction codes. While writing these out, figure out the process of how I would automate going from the graphical representation of the error correcting code to the Stim code for it.
3. Design a very simple user interface for drawing out the error correction circuits, this could be using an existing circuit drawing library, or potentially with representing them as graphs.
4. Implement the front end and the main body of the code. The aim of the program is that given an input of an arbitrary error correction code it outputs the corresponding Stim code. This will be a separate application that can work with Stim.

3.1 Extensions

Currently, the project involves constructing a very simple user interface tool in order to get the input of a arbitrary error correction code. An extension would be to do proper user research when designing the user interface.

4 Project Evaluation

There exist papers where given an error correction code it specifies the threshold and other properties of it. I could construct the error correction code in my designed user interface, and then after running it in Stim, I could check if I get the same results for the threshold. I could set aside an evaluation set of error correction codes (which I did not do manually initially) for this testing.

5 Resources Required

I will use my personal laptop for both code development and dissertation writing. I accept full responsibility for this machine and I have made contingency plans to protect myself against hardware and/or software failure. I will back up my codebase in GitHub and write my dissertation in Overleaf. Any other software or libraries that I use will be open source. Stim software has an Apache license so is open source.

6 Timetable and Milestones

1. Michaelmas weeks 1–2
   Planned work

- Further research into quantum error correction and different types of quantum error correction codes, including Shor code and Steane code.
- Familiarise myself with how Stim inputs work.

2. Michaelmas weeks 3–4
   Planned work
   - Write out the Stim implementations for five different error correction codes.
   - Devise an algorithm for this conversion.

   Milestones
   - Implemented five different error correction codes.
3. Michaelmas weeks 5–6
   Planned work
   - Finalise writing out the algorithm for the conversion from graphical representation to Stim code.
   - Design a simple user interface for error correction code representation.
   - Start developing the front end.

   Milestones
   - Design decisions made.
4. Michaelmas weeks 7–8
   Planned work
   - Continue developing the front end.
5. Christmas Holidays (05/12 - 22/01)
   Planned work
   - Finish developing the front end.
   - Develop the implementation of the algorithm and integrate it with the front end.

   Milestones
   - Front end of the application is complete.
   - First draft of the final implementation complete.
6. Lent weeks 1–2
   Planned work

- Test the implementation against error correction codes and ensure that it works correctly.
- Correct the code, and repeat testing.

7. Lent weeks 3–4
   Planned work
   - Prepare the progress report and presentation
8. Lent weeks 5–6
   Planned work
   - Test the application with at least seven different error correction codes.

   Milestones
   - Data collected for evaluation section
9. Lent weeks 7–8
   Planned work
   - Begin the dissertation: write the first two chapters.
10. Easter Holidays (20/03 - 09/04)
    Planned work
    - Write dissertation

    Milestones
    - Full dissertation draft sent for review
11. Easter Holidays (10/04 - 30/04)
    Planned work
    - Make changes to dissertation based on the feedback

    Milestones
    - Final dissertation ready
12. Easter weeks 1–3
    Planned work
    - Submit dissertation
    - Submit source code

    Milestones
    - Final dissertation and source code submitted.