Requirements of the interface

Could potentially limit to say that can only do Clifford quantum error correction codes? (I can do some research to see how many codes require more complicated gates)

Need to be able to specify:

Set of data qubits

Set of measurement qubits

Type of measurement qubits

Need to specify if a RESET is required before or after measuring (most likely yes - but doesn’t always seem to be the case)

Unsure how to do, but it seems to be that eg with X stabilisers, sometimes measure

If the code is toric

Need to have a way of representing a toric code as well

Unsure if this is actually the case, as could just add CNOTs between data qubits and measurement qubits in the last and first row/column – but may look a bit messy

The initial start state of any particular qubits

Eg any clifford operations to do after the initial reset

The 1-qubit operations that occurs

The 2-qubit operations that occur

Eg be able to draw in CNOT gates (and more importantly specify which is the target and which is the control)

What are the measurement and data qubits that apply to them

If there are different rounds to the 2-qubit operations

These also need to be specified

Need to clearly define what a 'round' is

It appears to be that I have used 'round' to mean 2 different things:

The order in which apply CNOT gates (as can't all do as one big line)

Have decided to implement this automatically (as part of generating the stim code the CNOTs will be safe)

If there are different measurement and detector rounds

E.g. first do this, measure, detect, then repeat … (like with the honeycomb code)

This would have to be implemented manually by the user

The logical observable

The logical observable is what can be used to represent a single logical qubit

I was thinking that you would only need to specify a single logical observable

But some codes seem to have multiple (honey-comb code had 1 per round)

Still not really sure about how to use the logical observable properly

The noise that will use for the different types of noise that will use (or the list of noises to use)

Do not need to specify the DETECTORs

I think all measurements can just use detectors of this round results and the previous round results

Additional details:

Name

Distance

Number of rounds

Questions need to think about:

I am currently unsure about the DETECTORs that are done using the data\_qubits (once they have been measured near the end)

But they also appear to be optional

Bonus features that could add (potential extensions - but these could be considered crucial to the design)

Could try and make it efficient with the development (ie have some ability to do copy and paste of patterns)

Could try do it so that if you specify what the code distance is (eg 3) - then could try and just say that the distance is now 5 - and hope that it can generate that successfully?

Need to think about ERRORs

Not very useful if the STIM output fails to make a detector\_error\_model (what is then used by decoders such as pymatching)

I could identify the most common errors and try and stop those?

Eg check that all of the DETECTORs are deterministic

According to a stack exchange response, common errors (that could be caused by this hypothetical interface) are

* A missing CNOT
* A missing or additional or wrong qubit rotation
* A reset in the wrong basis
* An extra measurement
* A DETECTOR that should not have been there in the first place (e.g. asserting X basis stabilizers are deterministic in the first round when initializing in the Z basis)

So would want to warn the user if these are occurring?

Later improvement:

Currently just output is a stim.Circuit

Could instead make the graphs etc.

This would require that the output works with pymatching and that can safely make a detector error model