Technical details in simulating magic preparation

Wang Yifei
Institute for Advanced Study, Tsinghua University

Date created: 2025-07-28
Date modified: 2025-07-28

Abstract

In this document, we discuss the technical details in simulating magic preparation circuit using generalized lattice surgery.

Contents

1. Simulating QRM code preparation

1.1. Handling or measurement errors

By a measurement error, we mean that we project the state correctly, but read the wrong result. A computational basis measurement with error rate p is formulated by measurement channel

$$\begin{aligned}\mathcal{M}_p\rho &= \sum_{m=0,1}((1-p)(\Pi_m\rho\Pi_m, m) + p(\Pi_m\rho\Pi_m, m+1)) \\ &= \sum_{m=0,1}((1-p)\Pi_m\rho\Pi_m + p\Pi_{m+1}\rho\Pi_{m+1}, m).\end{aligned} \tag{1}$$

Here

$$\Pi_m = \frac{1+(-1)^m Z}{2}, \quad \Pi_{m+1} = X\Pi_m X. \tag{2}$$

Therefore a measurement error is simulated by two correlated X errors before and after the measurement.

Conclusion: In real simulation, if the measurement is followed by a reset, we just put an X error before the measurement. If we will do more things after the measurement, we must put correlated errors.

1.2. Rounds of final measurement

We would like to think about how many rounds of measurements is needed in the final destructive measurement. Consider a general setting: we want to measure a logical X destructively on a code with Z-distance d. Before measurement, there are idling depolarization error. And there are measurement errors. Let each error occur independently with probability p. Suppose we post-selection on the syndrome of X-checks.

As we have discussed, we can decompose the measurement error into correlated Z errors before and after the measurement (note that we are measuring Z). Also, only the Z channel of depolarizion affect the measurement result. We can therefore merge these errors before the measurement, leading to Z errors with probability of order p. So the probability of a logical Z error after post-selection is trivially $O(p^d)$. In this case measuring more than one round can only enlarge the coefficient of the logical error rate. So one round of measurement is enough.

Note that the decomposition of measurement errors simplify our analysis: we do not need to analyse the complex situation of mixing two kinds of errors. One round of measurement even free us from considering the correlation of decomposed errors.

Conclusion: One round of destructive logical measurement is optimal.

1.3. Optimization of syndrome measurement without flags

In simulating the $[\![15, 1, 3]\!]$ QRM code, we need to first prepare a logical $|\overline{+}\rangle$ state. This is done by preparing a product state $|+\rangle^{\otimes 15}$ and measuring a complete set of Z-checks. The code has redundancy in the Z-checks. Trivially we choose the 4-qubit checks. However, we still have some freedom in choosing the checks. Our goal is two-fold:

1. The measurement can be parallelised into 4 rounds of CNOTs.
2. Each qubit is checked in a balanced number of times.

The first requirement is natural, although designing it is like a Sudoku game. The second requirement, on the one hand, makes the first easier to satisfy (you cannot achieve requirement 1 if there is a qubit being checked more than 4 times), and on the other hand, ensures good error suppression, as we will analyse later.

I number the qubits in QRM code as the normal Reed-Muller way. Qubits are labelled by 4-bit binaries with 0000 excluded.

0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Put on tetrahedron:

- vertices: qubits with one '1' (1, 2, 4, 8)
- edges: qubits with 2 '1's, or adding 2 vertices (3, 5, 6, 9, 10, 12)
- faces: qubits with 3 '1's, or adding 3 vertices (7, 11, 13, 14)
- body: qubits with 4 '1's, or adding 4 vertices (15)

We choose the Z-checks by the following rules:

- For each of the 4 triangles 124, 248, 481, 812, check two quadrilaterals adjcent to the edge with first two labels. This gives 8 checks, each vertex is checked twice, middle points of edges 12, 24, 48, 81 each is checked 3 times, middle points of edges 14, 28 each is checked twice, each face center is checked twice. One can check that $4 \times 2 + 4 \times 3 + 2 \times 2 + 4 \times 2 = 32 = 8 * 4$.
- add two checks that links edges 14, 28 and the body center. Now middle points of these two edges each is checked 1 more time, each face center is checked 1 more time, and the body center is checked twice.

Finally, each vertex and the body center is checked 2 times (totally 10), each edge center and face center is checked 3 times (totally 30). The list of checks are as follows.

```
QRM_Z_CHECKS_REDUCED = [
[ 1, 3, 5, 7],
[ 3, 2, 7, 6],
[ 2, 6,14,10],
[ 6,14,12, 4],
[13,12, 4, 5],
[12, 8, 9,13],
[ 8, 9,10,11],
[ 9, 1,11, 3],
[ 5, 7,13,15],
[10,11,15,14]
]
```

We have designed this list carefully such that a number only occurs once in each column. This meets our 2 requirements.

1.4. Optimization of syndrome measurement using meta-checks

We may also measure all 18 stabilizer generators and utilizing meta-checks. This is because if you just measure 10 independent checks, you must measure 3 rounds to ensure distance-3. But when considering mata-checks, these 18 checks form a $[18, 10, 3]$ classical code. One round of measurement is then enough for distance-3.

The checks are

```
QRM_Z_CHECKS = [
[ 1, 3, 5, 7],
[ 3, 2, 7, 6],
[ 2, 6,14,10],
[ 6,14,12, 4],
[13,12, 4, 5],
[12, 8,   13, 9],
[ 8, 9,10,11],
[ 9, 1,11, 3],
[ 5, 7,13,15],
[10,11,15,14],
# redundancies
[ 4, 5,       7, 6],
[14,    8,   10,12],
[       9, 1, 5,13],
[   10, 2,   11, 3],
```

```
[   15, 6,   14, 7],
[15,13,      12,14],
[ 7,    3,   15,11],
[11,       9,13,15]
]
```

The meta-checks are

```
QRM_META_CHECKS = [
    [ 1,18, 8, 9],
    [ 1,18,13,17],
    [ 2,10,14,15],
    [ 2,10, 3,17],
    [ 5,15, 4, 9],
    [ 5,15,11,16],
    [ 7,16, 6,10],
    [ 7,16,12,18],
]
```

When implementing, we fill the empty positions of checks by 0, and add if-conditions when appyling the CNOT.

1.5. Handling feedback on applying transversal gates

After a correct syndrome measurement, we get the results of 10 independent checks. They determine the code we get, which can be converted to the standard QRM code by an X string. The X string $X(s)$ ($s \in \mathbb{F}_2^{15}$) to convert our code to the standard one is solved from the measurement result $m \in \mathbb{F}_2^{10}$, by solving

$$H_Z s = m, \tag{3}$$

where $H_Z \in \mathbb{F}_2^{10\times 15}$ and $s \in \mathbb{F}_2^{15}$. Now the logical state of the code abtained and the standard QRM code has the relation

$$|\overline{+}\rangle = X(s)|\overline{+}\rangle_{\text{std}}, \quad |\overline{=}\rangle = \overline{Z}|\overline{+}\rangle = (-1)^{w(s)}X(s)|\overline{=}\rangle_{\text{std}}, \tag{4}$$

since $\overline{Z}$ can be taken as $Z^{\otimes 15}$. Consequently, in computational basis,

$$|\overline{0}\rangle = X(s)\overline{X}^{w(s)}|\overline{0}\rangle_{\text{std}}, \quad |\overline{1}\rangle = X(s)\overline{X}^{w(s)}|\overline{1}\rangle_{\text{std}}. \tag{5}$$

Therefore, the logical T on the new code is

$$\overline{T} = \overline{X}^{w(s)}X(s)T^{\dagger\otimes 15}X(s)\overline{X}^{w(s)}, \tag{6}$$

which is a transversal T conjugated by some X-string. Note that

$$XT^\dagger X = T = T^\dagger S \tag{7}$$

up to some overall phase, therefore we should apply

$$\overline{T} = \overline{X}^{w(s)}T^{\dagger\otimes 15}S(s)\overline{X}^{w(s)} \tag{8}$$

The feedback contains two parts:

- The $S(s)$ part can be implemented as follows. We first find a matrix $C \in \mathbb{F}_2^{15\times 10}$ such that $H_Z C = I_{10}$. Therefore we have $s = Cm$. In numerical simulation, such feedback can be implemented by adding a Z gate to qubit i conditioned on measurement j iff $C_{i,j} = 1$.

- The $\overline{X}^{w(s)}$ part can be eliminated by requiring that every column of C has even weight. This can be done by adding any logical X vector to any row of C with odd weight.

For our numerics, we simulate an S gate. The analysis above still holds with T replaced by S since $XS^\dagger X = S = S^\dagger Z$.

1.6. Error propagation in syndrome measurement

Let's analyse the error propagation in measuring the listed checks. There are three types of errors: errors in data qubits, errors in ancilla qubits and measurement errors. If there is only errors in data qubits, code distance and post-selection ensures a p^3 logical error suppression. If there is only measurement errors, it can always be detected. For errors in ancilla, X errors can always be detected and Z error can be spread to data qubits, which when affecting one data qubits, can be viewed as a single data qubit error, but when affecting two data qubits, introduces correlated two-qubit Z errors in data qubits. The latter case can destroy the p^3 suppression in the final destructive measurement. *This is the first dangeous point.* Let's consider combined errors.

1.7. Optimization of syndrome measurement with flags

Standard.

2. Growing surface code

Initialize the data qubits to be growed to $|0\rangle$. Add detectors to the stabilziers that should be 1.

It is important not to change the indices of used qubits.