第一章 表面码数学原理

1.1 稳定子码

1.1.1 泡利群、稳定子群、编码空间

泡利群(Pauli group) $\mathcal{P}_n$ 为泡利算子 $\langle X, Y, Z\rangle$ 的 n 次张量积：

$$\mathcal{P}_n = \{\pm 1, \pm \mathrm{i}\} \cdot \{\pm \mathrm{I}, \pm \mathrm{i}I, \pm X, \pm Y, \pm Z\}^{\otimes n}.$$

其满足：1. $\mathcal{P}_n$ 中的任意两个元素是对易(commute)或反对易的(anticommute)；2. $\mathcal{P}_n$ 中的任意元素是厄米(Hermitian)或反厄米的(anti-Hermitian)；3. $\mathcal{P}_n$ 中的任意元素是幺正的(unitary)。

稳定子群(stabilizer group) $\mathcal{S}$ 是泡利群的一个阿贝尔子群(abelian subgroup)，即 $\mathcal{S}$ 中的任意两个元素是对易的。注意我们定义 $-I \notin \mathcal{S}$。

定义 1 （编码空间） 编码空间(codespace) $\mathcal{T}(\mathcal{S})$ 是 $\mathcal{S}$ 中特征值为 $+1$ 的特征向量构成的空间：

$$\mathcal{T}(\mathcal{S}) = \{|\psi\rangle \mid S|\psi\rangle = |\psi\rangle, \forall S \in \mathcal{S}\}.$$

1.1.2 中心化子、正规化子

中心化子(centralizer) $\mathcal{C}(\mathcal{S})$ 是泡利群中与 $\mathcal{S}$ 中的所有元素对易的元素构成的子群：

$$\mathcal{C}(\mathcal{S}) = \{E \in \mathcal{P}_n \mid [E, S] = 0, \forall S \in \mathcal{S}\}.$$

正规化子(normalizer) $\mathcal{N}(\mathcal{S})$ 是泡利群中与 $\mathcal{S}$ 中的所有元素通过共轭保持不变的元素构成的子群：

$$\mathcal{N}(\mathcal{S}) = \left\{E \in \mathcal{P}_n \mid E^\dagger S E \in \mathcal{S}, \forall S \in \mathcal{S}\right\}.$$

定理 1 在泡利群中，$\mathcal{C}(\mathcal{S}) = \mathcal{N}(\mathcal{S})$。

证明 泡利群中任意两个元素是对易或反对易的，则有

$$E^\dagger S E = \pm E^\dagger E S = \pm S,$$

又 $-I \notin \mathcal{S}$，即 $-S \notin \mathcal{S}$，有

$$\begin{aligned} & E^\dagger S E = S \\ \Rightarrow\ & S E = E S. \end{aligned}$$

因此 $\mathcal{C}(\mathcal{S}) = \mathcal{N}(\mathcal{S})$ 得证。

■

对于任意 $E \in \mathcal{C}(\mathcal{S}) - S$，其为一个无法检测的逻辑错误(logical error)，即 E 作用在编码空间 $\mathcal{T}(\mathcal{S})$ 上的结果仍然在编码空间中，但是又会改变当前的态。因此可称之为一个逻辑算子(logical operator)，能进行逻辑 $\langle X, Y, Z\rangle$ 操作。

定义 2 （码距） 泡利算子 $P \in \mathcal{P}_n$ 的权 $|P|$ 是它产生非平凡作用（即并非 I 算子）的量子比特的个数。该稳定子码的码距(code distance)定义为 $\mathcal{C}(\mathcal{S}) - \mathcal{S}$ 中元素的最小权的大小。

1.1.3 生成元与错误症候

稳定子群 $\mathcal{S}$ 的生成元 $\{S_1, S_2, ..., S_r\}$ 被称为校验算子(check operators)。

定义 3 （错误症候） 给定一个泡利错误 $E \in \mathcal{P}_n$，其症候(syndrome) $\sigma(E)$ 定义如下：

$$\sigma_S(E) = \begin{cases} 0 & \text{if } [S,E] = 0 \quad \text{(commute)} \\ 1 & \text{if } \{S,E\} = 0 \text{ (anticommute)} \end{cases}$$

$$\sigma(E) = \left(\sigma_{S_1}(E), \sigma_{S_2}(E), ..., \sigma_{S_r}(E)\right)$$

1.1.4 纠错理论

给定一个错误症候 $\sigma(E)$，解码器根据它来选择一个纠错算子 $R \in \mathcal{P}_n$，若 $RE \in \mathcal{S}$，就有 $RE|\psi\rangle = |\psi\rangle$，纠错成功；若 $RE \in \mathcal{P}_n - \mathcal{C}(\mathcal{S})$，就有 $RE|\psi\rangle$ 不处于 $\mathcal{T}(\mathcal{S})$ 编码空间中，不是一个正确编码，纠错失败。

定理 2 当 R 和 E 的症候相同时，$RE \in \mathcal{C}(\mathcal{S})$。

证明 对于任意一个稳定子 $S \in \mathcal{S}$，有

$$RES = R(ES) = (-1)^{\sigma_S(E)} RSE$$
$$SRE = (SR)E = (-1)^{\sigma_S(R)} RSE.$$

由于症候相同，即 $\sigma_S(E) = \sigma_S(R)$，因此

$$(RE)S = S(RE), \forall S \in \mathcal{S}$$
$$\Rightarrow RE \in \mathcal{C}(\mathcal{S})$$

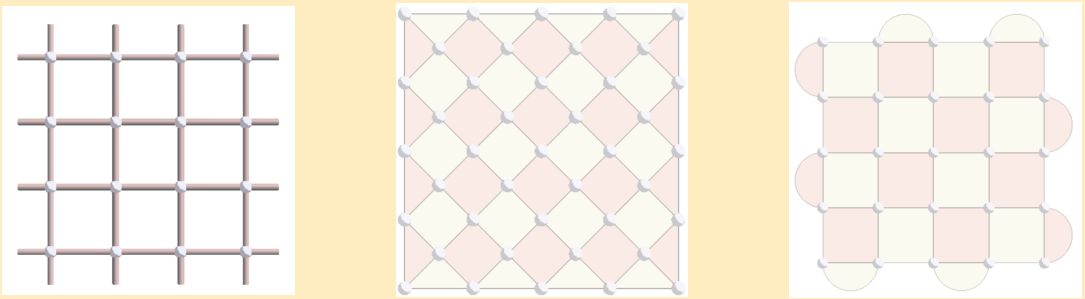
得证。

此时若 $RE \in \mathcal{C}(\mathcal{S}) - \mathcal{S}$，则发生一个逻辑错误(logical error)，逻辑比特的逻辑态被改变。

■

1.2 表面码

注记 1 你可能见过以下几种表面码的图示：



(a) (b) (c)

Figure 1: Different surface code representations

实际上三者是一致的，只是进行了一些旋转以及边和顶点的交换，我们会在之后展示。这里使用 Figure 1 中的图(a)来介绍表面码的基本概念。

1.2.1 表面码的构成

在图(a)当中，我们将 **数据比特放置在每条边** 上；在每个顶点处放置 Z 型稳定子，称之为 **顶点稳定子(vertex stabilizers)**；在每个面心处放置 X 型稳定子，称之为 **面稳定子(plaquette stabilizers)**。在后面的检错与纠错阶段，我们将看到这样定义的方便之处。

下图是两种稳定子的示意，红色意味着 X 型稳定子，其作用在该面的四条边上；蓝色则为 Z 型稳定子，作用在顶点相连的四条边上。

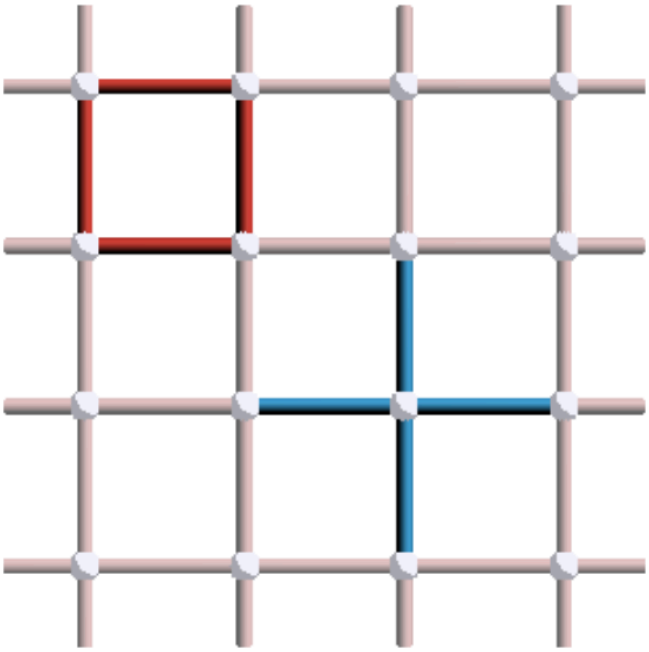


Figure 2: Examples of vertex and plaquette stabilizers

接下来分析两种稳定子的具体电路构造。在 Figure 3a 中可以看到 Z 型稳定子的实现，在中心会有一个辅助比特(ancilla qubit)用于测量，该投影测量会强迫临近四个数据比特进入该稳定子 $Z_aZ_bZ_cZ_d$ 的本征态中，即 $Z_aZ_bZ_cZ_d|\psi\rangle = Z_{abcd}|\psi\rangle, Z_{abcd} = \pm 1$，显然当辅助比特的测量结果为 $|0\rangle$ 时，表征本征值为$+1$，反之为 -1。在 Figure 3b 中可以看到 X 型稳定子的类似实现，以将数据比特投影到 $X_aX_bX_cX_d$ 的本征态中。

注意到 Figure 3a 中额外施加了一个 I 门，这是为了与 Figure 3b 中施加的 Hadamard 门保持同步，减少测量时间不匹配导致的误差。

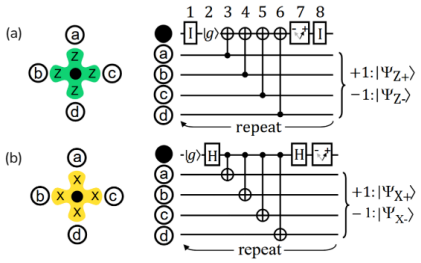


Figure 3: $X\&Z$ stabilizers quantum circuit construction

根据定义，要构成一个稳定子群，这些稳定子必须相互对易。显然 X 型稳定子之间是对易的，Z 型稳定子之间也是对易的。而 X 型稳定子与 Z 型稳定子之间总是有零或两个重合的边。

定理 3 在上述表面码结构中，X 型稳定子与 Z 型稳定子总是对易的。

证明 若二者没有重合边，则它们的作用在数据比特上是完全独立的，显然对易；若二者有两个重合边，即测量同样两个数据比特，假设 a, c 为重合边，则有

$$\begin{aligned}(X_aX_bX_cX_d)(Z_aZ_eZ_cZ_f) &= (X_aZ_a)\otimes X_b\otimes Z_e\otimes (X_cZ_c)\otimes X_d\otimes Z_f\\ &= -(Z_aX_a)\otimes Z_e\otimes X_b\otimes -(Z_cX_c)\otimes Z_f\otimes X_d\\ &= (Z_aZ_eZ_cZ_f)(X_aX_bX_cX_d).\end{aligned}$$

简单来看，X_a, Z_a 与 X_c, Z_c 两对反对易算子相互抵消，结果是 X 型稳定子与 Z 型稳定子相互对易。

■

当然，X 型稳定子与 Z 型稳定子是完全对称的。在 Figure 4 左可以看到，X 型稳定子的组合会构成格(lattice)中的环(loops)；对于 Z 型稳定子，我们以每条边的中点为心将每条边旋转 90°（灰色虚线表征旋转后的结果），这样就将面和顶点进行了交换，紫色实线即为三个 Z 型稳定子的组合结果，显然与左图结构是一致的。形式上，这被称作在对偶格(dual lattice)中表示算子。

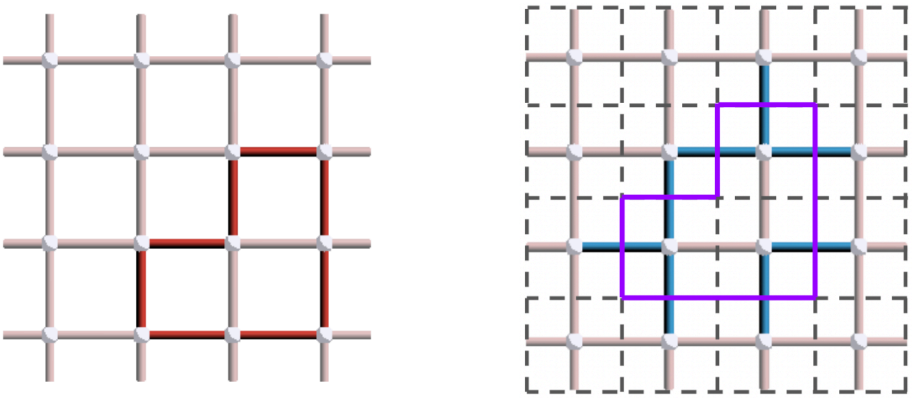


Figure 4: Dual Lattice

1.2.2 错误检测

往表面码中插入一些泡利 X 错误，症候图和相应的错误如下：

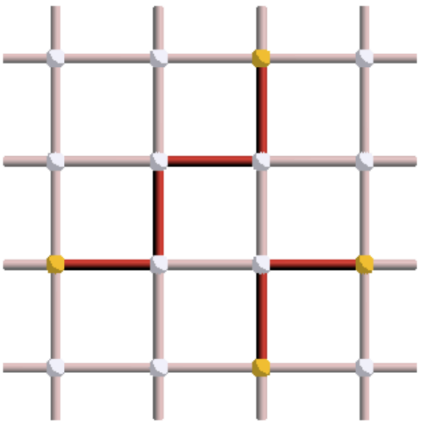


Figure 5: Syndrome and X Error

黄色顶点是与错误反对易的一些 Z 型稳定子，他们是错误症候的一部分，依照定义 3，其满足 $\sigma_{S_i}(E) = 1$，它们通常被称为 **激励(excitations)** 或者 **缺陷(defects)**。

我们把这些相连的错误称为一条 **错误链(error chain)**。可以发现缺陷通常存在于错误链的端点处，因此当错误链成环时，缺陷消失，此时该错误与所有稳定子对易，即 $E \in \mathcal{C}(\mathcal{S})$。

1.3 逻辑比特与逻辑算子

我们如果从流形和拓扑的角度来考量表面码，就能发现它其实可以转换成一个环面，这样就能更轻易地判断一个环是平凡的（即 $M \in \mathcal{S}$）还是非平凡的（即 $M \in \mathcal{C}(\mathcal{S}) - \mathcal{S}$），不过在此不表。

将所有数据比特用一个逻辑态 $|u\rangle_L$ 表征，辅助比特的状态 $|v\rangle$ 表征症候，整个阵列的量子态即为

$$|\psi\rangle = |u\rangle_L \otimes |v\rangle.$$

比如对于 Figure 4 的左图，如果我们把红色边视作一整条错误链，其成环之后，就相当于三个 X 型稳定子的组合，作用在当前的量子态上不改变任何东西。

而对于 Figure 6 左图，可以发现它不能由任何稳定子生成，并且也与所有稳定子对易，作用之后，会翻转逻辑态 $|u\rangle_L$ 而不改变稳定子测量结果，其被称为一个 **逻辑 X 算子(X_L)**。右图与左图相比只是增加一些 X 型稳定子，依然是一个 X_L。可以发现，联通左右边界或者上下边界的一条链即为一个非平凡的逻辑算子。

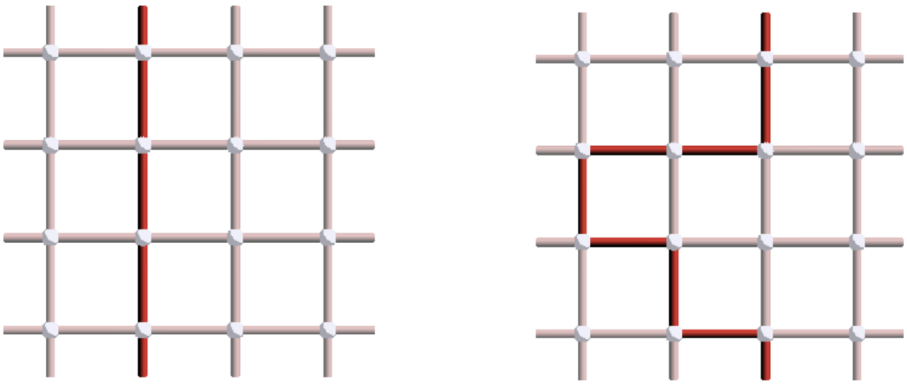


Figure 6: Logical X Operator

可以发现对于图中的表面码，水平和垂直分别有两个逻辑算子 X_{1L}, Z_{1L} 和 X_{2L}, Z_{2L}，因此可以编码 2 个逻辑比特；又根据定义 2 可知该表面码的码距为 4（因为上下联通，左右联通）。

1.4 具有开放边界的表面码

考虑以下版本的表面码 Figure 7，其中顶部和底部边界上的 Z 型稳定子以及左侧和右侧边界上的 X 稳定子现在仅作用在三个数据比特上。

我们将上下边界称为平滑边界(smooth boundaries)，将左右边界称为粗糙边界(rough boundaries)。可以看出，该表面码的码距为 5；水平方向仅有一个逻辑 X 算子 X_L，垂直方向仅有一个逻辑 Z 算子 Z_L，因此可以编码 1 个逻辑比特。此时 X_L 必须连接粗糙边界，而 Z_L 必须连接平滑边界。

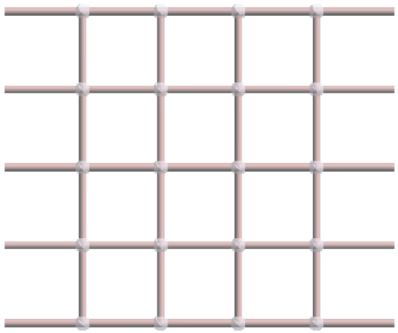


Figure 7: Surface Code with Open Boundaries

第二章 表面码解码

根据前文纠错理论，稳定子测量后给出一个错误症候 $\sigma(E)$，解码器需要根据该症候来选择一个纠错算子 R，使得 $RE \in \mathcal{S}$。

假设观察到症候如下 Figure 8，并想找到一个合适的纠错算子。

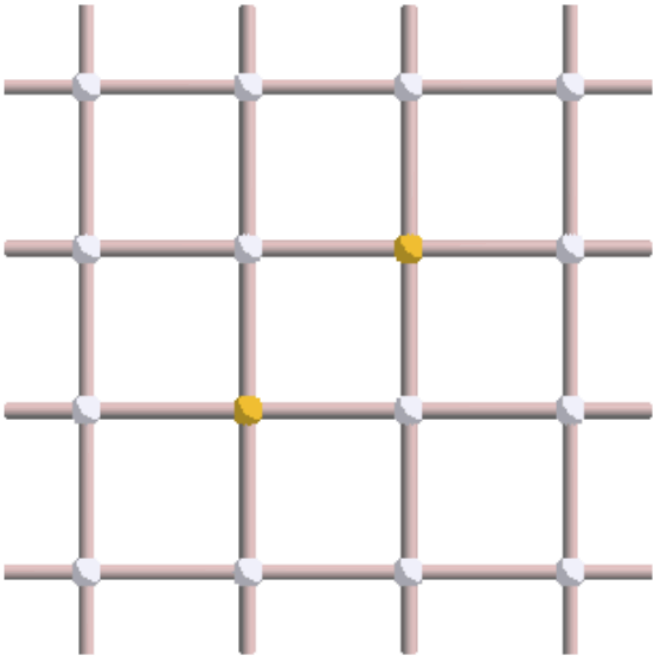


Figure 8: A Syndrome Example

产生这种症候的错误链可能有很多种，比如以下三张图

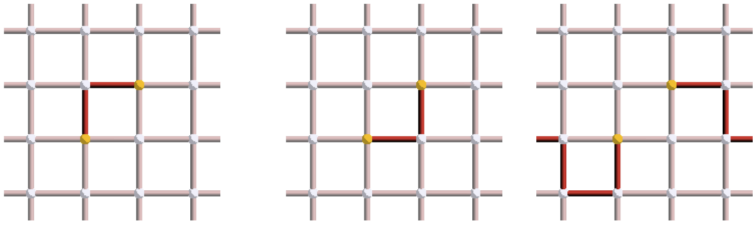


Figure 9: Possible Errors

假设真实错误是第一张图，而我们选择纠错算子为第二张图，作用结果相当于一个稳定子，纠错成功。但如果选择第三张图做纠错，作用结果如 Figure 10，黄色边代表纠错算子。显然这是一个逻辑 X 算子，纠错失败，将产生一个逻辑错误。

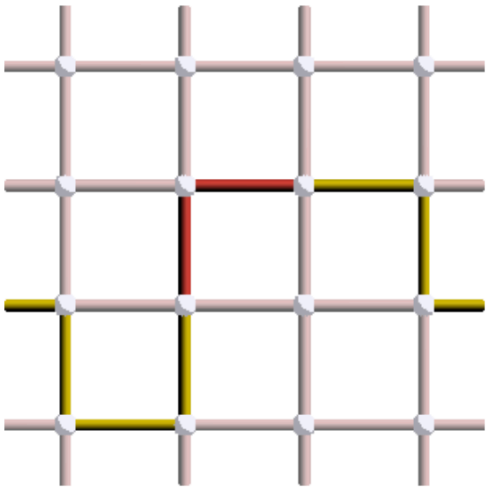


Figure 10: Failed Correction

因此解码就是找到概率最高的错误，其等价于找到符合症候的最小错误（错误链越短，发生概率越高）。对于表面码，这相当于将缺陷与最小权重链进行匹配，这完全等同于解决 **最小权重完美匹配***(minimum-weight perfect matching, MWPM)* 问题。

2.1 MWPM 解码器

可以把解码过程分解为两个部分：

1. 使用 *Local Dijkstra* 算法找到连接各个缺陷的最短路径，构建完全图。
2. 使用 *Blossom* 算法或者其他更优化版本得到一组完美匹配，根据第一部分获得相应错误链构建纠错算子。

例如考虑以下解码问题：

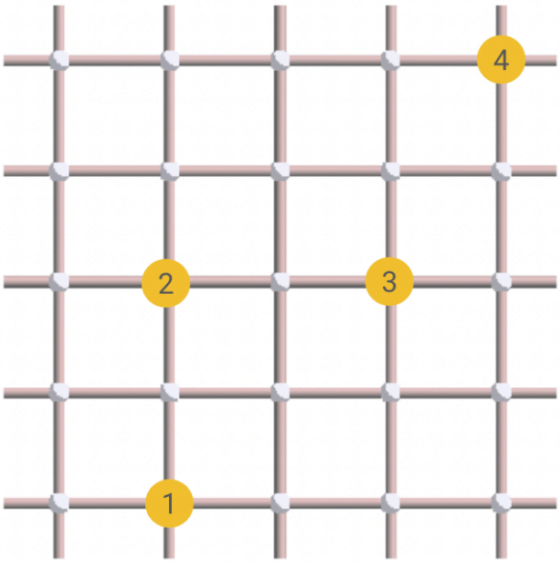


Figure 11: Decode Example

假设噪声独立同分布，即每条边发生错误的概率相同，则两个缺陷之间的边的权重可以由二者的曼哈顿距离表征，如 Figure 12 左图所示，构建完全图并枚举所有可能的匹配以找到最小匹配。右图为对应的错误链或者纠错算子。

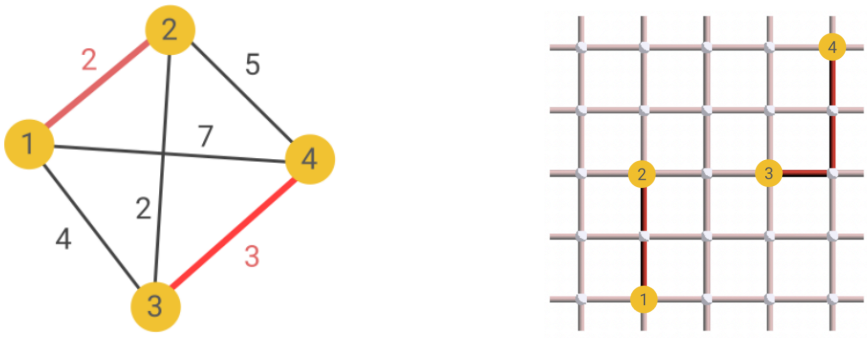


Figure 12: Minimum-Weight Matching

另外，如果数据比特发生错误的概率并不相同，例如发生 X error 的概率为 p_i，则在图中相应边的权重应写为 $w_i = \log\left(\frac{1-p_i}{p_i}\right)$[1]。

2.2 旋转表面码

从解码过程可以发现，将数据比特放置在边上，而将稳定子放置在顶点和面上，非常方便将缺陷连接为完全图来做解码。现在我们可以来看一开始提及的，关于表面码的其他两种图示。

对于 Figure 13 左图，数据比特位于顶点，稳定子位于面上，这种表示被称为校正晶格(rectified lattice)。从右图可以看出二者等价。这种表示方法的优点之一是，表面码排列更为紧凑。

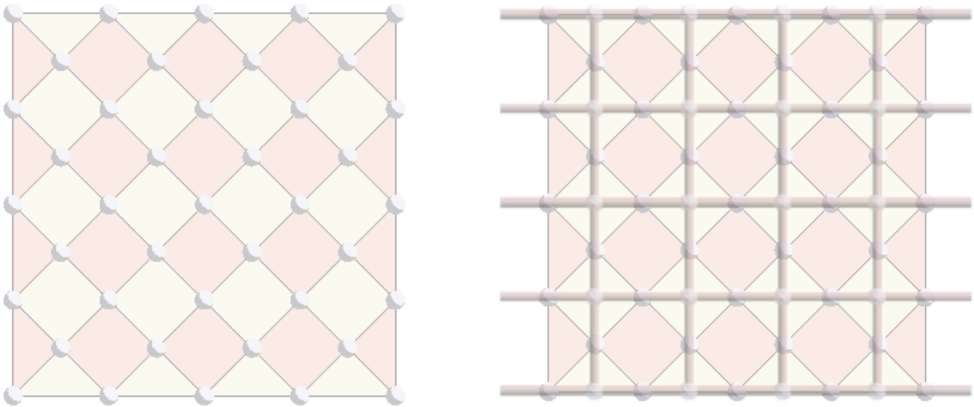


Figure 13: Rectified Lattice

取其中心部分并添加一些边界稳定子，就能得到第三种表示如 Figure 14，称为旋转表面码(rotated surface code)。

[1]Dennis, Eric, Alexei Kitaev, Andrew Landahl, and John Preskill. ‘Topological Quantum Memory’, 24 October 2001. https://doi.org/10.1063/1.1499754.

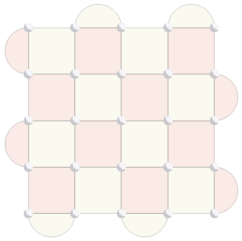


Figure 14: Rotated Surface Code

2.3 逻辑错误率的统计模型

定义 4（阈值） 表面码的 **阈值(threshold)** 是其物理错误率的一个临界点。当超过该阈值时，逻辑错误率随着码距增加而增大；否则，码距增加能够降低逻辑错误率，以实现容错量子计算。

首先需要注意的是，表面码没有单一的阈值：它很大程度上取决于我们使用的噪声通道和解码器。

通常分为三类噪声模型：

1. **代码容量模型(code-capacity model)**，其中代码的所有物理量子位都可能出现错误，但测量被认为是准确的。
2. **现象学噪声模型(phenomenological noise model)**，每个稳定子测量存在固定的失败概率。
3. **电路级噪声模型(circuit-level noise model)**，其中考虑了表面码的初始化和症候提取过程的电路，并假设每个量子门之后存在一定的错误概率。

代码容量模型的对应阈值是最容易估计的，并且可以用来粗略地了解给定表面码或解码器的性能。

当得到阈值后，也可以估计相应的逻辑错误率。

定理 4 对于使用 MWPM 解码器，码距为 d 的表面码，其逻辑错误率近似为

$$P_L \propto \left(\frac{p}{p_{\text{thr}}}\right)^{d_e},$$

其中 p_{thr} 为阈值，$d_e = \lceil \frac{d}{2} \rceil$。

证明 考虑 Figure 15 中的症候，其最可能与次可能错误链如图左与图右所示，若实际发生的错误是右图，使用 MWPM 解码器总会给出如图左的纠错算子，此时发生逻辑错误。该错误链的链长就为 d_e。

当然可能对更长的错误链发生错误识别，由于发生概率与链长成指数关系，因此是小量可以暂不考虑。

则将求出该类型错误链的发生概率即可得到逻辑错误率

$$P_L \propto d \cdot \mathrm{C}_d^{d_e} \cdot p^{d_e} = d \cdot \frac{d!}{(d-d_e)! \cdot d_e!} \cdot p^{d_e}.$$

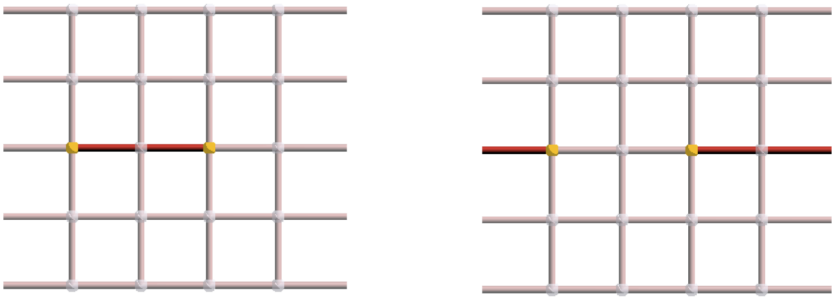


Figure 15: Logical Error Model

■