Dynamic Automorphism Codes

Sunil Vittal

May 2024

1 Introduction

The authors of [3] recently publsihed work on a new kind of quantum error correction codes called Dynamic Automorphism codes. These build off of the work on *Floquet Codes*, presented by Haastings and Haah in [2]. The huge distinction between these codes and traditional stabilizer codes is the codespaces. In the former, the codespace is changing throughout time, not only presenting a flow of information through the codes, but also providing a better interpretation of fault tolerance in terms of space time [3]. Moreover, the measurements are of anticommuting operators which essentially changes the code from one time step to the next. Meanwhile, in stabilizer codes, the codespace is unchanged and measurements are those involving commuting stabilizers. The codespaces in such dynamic codes result from measurements over time that form time-dependent stabilizer groups. The very fact that such codes exist and function well with respect to error and fault tolerance increases interest in the dynamic properties of Floquet and Automorphism codes. In particular, what kind of operations evolve our codespace in a way that preserves information?

Before defining dynamic automorphism codes in depth, we first provide background on other dynamic quantum error corrections codes before talking about dynamic automorphism codes. In particular, Floquet Codes are a prime example of such dynamic codes, where logical qubits are also 'dyanamically generated' via the above process. Moreover, we cover basics of the Color Code. Since the authors describe the codes using anyons, we also cover anyon theories and condensations before formulating these ideas in stabilizer measurements. We then cover major results from [3] on the *Dynamic Automorphism Color Code* in regards to implementing the code and the error correction protocol.

1.1 Floquet Codes

1.1.1 Subsystem Codes

Definition 1.1. Let $\mathcal{H}$ be a Hilbert space with orthogonal decomposition into $\mathcal{C} \oplus \mathcal{C}^\top$ where $\mathcal{C}$ is the codespace of some error correction code. For *subsystem codes*, $\mathcal{C}$ has two subsystems, so $\mathcal{C} = A \otimes B$. We focus on errors in the subsystem A while errors in B are neglected and provide degrees of freedom. [1]

Like Stabilizer codes, we have the same definition of Stabilizer groups for all $E \in S$, $E|\psi\rangle = |\psi\rangle$ for all $|\psi\rangle \in \mathcal{H}$. Note that since we have these two subsystems we define two groups $\mathcal{L}, \mathcal{G} \subset \mathcal{P}_n$, the logical and gauge groups, respectively where $\mathcal{P}_n$ is the group of Pauli Operators on n qubits. To respect the tensor product structure on $\mathcal{C}$, we require that elements in $\mathcal{L}, \mathcal{G}$ commute with each other. That is for $L \in \mathcal{L}, G \in \mathcal{G}$, $[L, G] = 0$. The group $\mathcal{G}$ is often called the Gauge group, and one can observe that the Gauge group induces an equivalence relation on the Code space. Indeed for distinct states $|\psi\rangle, |\phi\rangle \in \mathcal{C}$, we say that these states are equivalent if $|\psi\rangle = G|\phi\rangle$ for $G \in \mathcal{G}$. We denote this relation by $|\psi\rangle \sim_{\mathcal{G}} |\phi\rangle$

Proposition 1.2. *$\sim_{\mathcal{G}}$ is an equivalence relation on $\mathcal{C}$.*

Proof. For reflexivity, note that $I \in \mathcal{G}$, so $|\psi\rangle \sim_{\mathcal{G}} |\psi\rangle$ trivially. For the symmetric property, we observe that if $|\psi\rangle \sim_{\mathcal{G}} |\phi\rangle$, we get $|\psi\rangle = G|\phi\rangle$ if and only if $G^-|\psi\rangle = |\phi\rangle$, meaning $|\phi\rangle \sim_{\mathcal{G}} |\psi\rangle$. For the symmetric property, suppose $|\psi\rangle \sim_{\mathcal{G}} |\phi\rangle$ and $|\phi\rangle \sim_{\mathcal{G}} |\varphi\rangle$. Then we have $|\psi\rangle = G_1|\phi\rangle$ and $|\phi\rangle = G_2|\varphi\rangle$. It follows that $|\psi\rangle = G_1G_2|\varphi\rangle$, so $|\psi\rangle \sim_{\mathcal{G}} |\varphi\rangle$, as desired. □

In view of Proposition 1.2, we observe that for $L \in \mathcal{L}, G \in \mathcal{G}$, L and $L \otimes G$ induce equivalent actions on states $|\psi\rangle \in \mathcal{C}$. To construct the Stabilizer group of $\mathcal{C}$, first choose $2n$ elements $X_i, Z_i, 1 \leq i \leq n$ of $\mathcal{P}_n$ that satisfy the commutator of the normal Pauli matrices ($[X_i, Z_j] = 2\delta_{ij} X_i Z_j$). Choose $r < n$ elements $S_1, \ldots, S_r$ such that $\mathcal{S} = \langle S_1, \ldots, S_r \rangle$ is our stabilizer group. One notes that the centralizer of this group is simply $C(\mathcal{S}) = \langle iI, S_1, \ldots, S_r, X'_{r+1}, \ldots, X'_n, Z'_{r+1}, \ldots, Z'_r \rangle$. [1]

It's intuitive that $\mathcal{G}$ should be formed by the Stabilizer group since the action of the Gauge Group on the code space is the same as the Identity. We can see this concretely via a construction from choosing the number of 'gauge' qubits the group acts on. Suppose we have m gauge qubits such that $r + m < n$, so that we can use operators $X'_{r+1}, \ldots X'_{r+m}, Z'_r, \ldots, Z'_{r+m}$ as operators that generate $\mathcal{G}$. As a result, we find

$$\mathcal{G} = \langle iI, S_1, \ldots, S_r, X'_{r+1}, \ldots, X'_{r+m}, Z'_{r+1}, \ldots, Z'_{r+m} \rangle$$

Since the Gauge Group acts on $r + q$ qubits, we now have that $\mathcal{L}$ acts on $k = n - r - q$ qubits. As a result, A is a 2^k dimension subspace of $\mathcal{H}$ and B is a 2^q dimensional subspace of $\mathcal{H}$. Moreover, since we don't care about errors on B, we want errors to be on this subsystem since our logical qubits are safe. We can also go from a subsystem code to a stabilizer code and vice versa. To go from subsystem to stabilizer, we let the stabilizer group act on the gauge qubits as well so the extra q gauge qubits are stabilized. Going from a stabilizer code to a subsystem code is trickier since we do not know with certainty what q can be.

1.1.2 Honeycomb Floquet Code

The Honeycomb Code is constructed by considering a tiling of a lattice via Hexagonal Plaquettes where each vertex of the Hexagon is a physical qubit. There are three kinds of edges on each

Plaquette, so we simply assign the edges for each direction of the edges. For instance, we can start from some edge and assign the edges cyclically with x, y, z. For two qubits connected via an x edge, the check is XX. Similarly, for qubits connected by a y or z edge, their checks are YY and ZZ respectively. Note then, that each qubit is on the honeycomb is assigned a unique Pauli Check. We can add further classification since the tiling of hexagons in the plane induces a $3-$colorable graph, meaning we can assign each hexagon a type $t \in \{0, 1, 2\}$. Therefore, each edge on a hexagon can be idenitifed by one of x, y, z and one of $0, 1, 2$, giving way to 9 types of edges.

Unlike other codes, the dynamic aspect of the Honeycomb code is that we measure checks on hexgaons of a certain type at each round. In particular, in the rth round, we have measured checks on hexagons of type r mod 3. This added time step component to measuring the checks allows us to define Stabilizers dependning on the round, rather than global stabilizers throughout each round. As a result, there is no fixed subspace of logical qubits. In fact, when viewed as a subsystem code on the torus, the honeycomb has no logical qubits. [2]

Indeed, consider n_p honeycombs tiling the torus, so that there are $2n_p$ physical qubits but $3n_p$ edges. Note that the product of all $3n_p$ checks on the torus is the identity, so we have a redundancy and the Gauge group has dimension $3n_p - 1$ since the check operators generate the Gauge group. To find the dimension of the stabilizer group, we observe that it has dimension $n_p + 1$, thus, the number of gauge qubits is $n_p - 1$, so there are no logical qubits when this code is viewed as a subsystem code.

How does one get around this and define stabilizer groups and logical checks for this code? We can define an *Instantaneous Stabilizer Group*. Since in general stabilizer formalism, if a state $|\psi\rangle$ is stabilized by $\mathcal{S}$, measuring the state with a Pauli operator P projects $|\psi\rangle$ to another stabilized state. We can track the stabilizer group of this stabilized state after measurements of our state in each round.

- If $P \in \mathcal{S}$, then we know what the measurement outcome should be and the Stabilizer Group is unchanged.
- If $\pm P \notin \mathcal{S}$ but $[P, Q] = 0$ for all $Q \in \mathcal{S}$, then we expand our Stabilizer group to become $\mathcal{S}' = \langle \mathcal{S}, \pm P \rangle$ where the sign on P depends on the measurement outcome of P.
- If $\pm P \notin \mathcal{S}$ but anticommutes with some $Q \in \mathcal{S}$, then we get $\mathcal{S}' = \langle \mathcal{S}_0, \pm P \rangle$ as our new stabilizer subgroup where $\mathcal{S}_0 \subset \mathcal{S}$ is a subset of elements that commute with P

Due to the periodicity of measurements of checks in this code, the ISG for the honeycomb code is consistently evolving, yet it preserves quantum information. Moreover, the transformations from codespace to codespace in each round also create a codespace that is equivalent to an effective toric code. These transformations also effectively implement an nontrivial automorphism on underlying topological order of the code, causing the authors of [3] to write their paper. In particular, *Dynamic Automorphism Codes*, are generalized Floquet codes in the sense that they dynamically generate

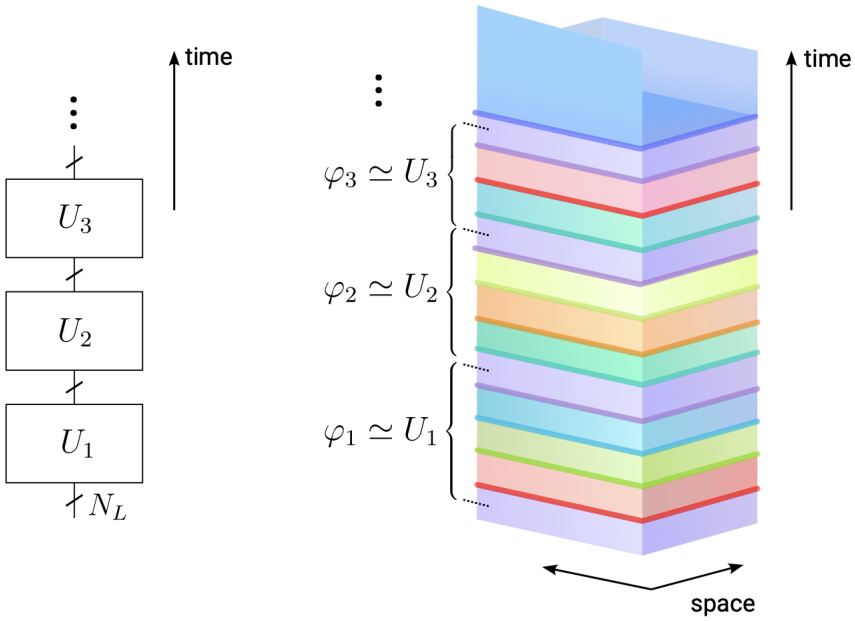

Figure 1: Fig 1 from [3]. On the left we have the sequence of logical operators acting on N_L logical qubits corresponding to automorphisms on the right that transform the codespace periodically. At every time-step, the darker lines indicate few-qubit measurmenets that drive the information through time while measuring syndromes for error correction purposes.

logically qubits via a sequence of automorphisms on the topology of the error correction code. We can now give a proper definition of this class of codes via the following:

Definition 1.3. A Dynamic Automorphism Code is defined by pairs of sequences $\{U_1, U_2, \dots\}$ and $\{\varphi_1, \varphi_2, \dots\}$ where the U_i are multi-qubit logical gates and the φ_i are automorphisms on the codespace that implements the measurements on the U_i. As a result, each φ_i corresponds to U_i.

2 The Dynamic Automorphism Color Code

For the purposes of this literature review, we are considering two-dimensional color codes. Stabilizer color codes are essentially tilings of the 2D plane with triangles in a way such that vertices of the triangles form a 3 regular graph and the triangles themselves are 3 colorable, just as the honeycomb code was. We can then define checks on these triangles via tensor product of X and Z operators on the vertices of each triangle.[4]

The basic building blocks of the Dynamic Color Code involve sequences of two qubit measurements which induce a nontrivial automorphism of the color code. These sequences fully describe the code since each measurement block can be put together to form the desired sequence of nontrivial automorphisms. A key claim for this code is that *all* nontrivial automorphisms of the color code can be implemented by these sequences of two qubit measurements.

2.1 Properties of the Color Code

Here we summarize the exposition of the Color Code properties from [3] by using their formulation of the code via anyon theory.

2.1.1 Anyon Theory of the Color Code

Definition 2.1. An *anyon* is a two dimensional quasi-particle that can be described by their phase and their interactions with other anyons. We define the set of these interactions to be an *anyon model/theory* $\mathcal{M}$.

One observes that these anyon theories have properties akin to an abelian group. They have a *vacuum* element 1 which serves as an identity elements. Nontheless, we can describe multiple operations/interactions between anyons via fusion, exchange, and braiding. *Fusion* is similar to multiplication in that it is the process of bringing two anyons $a, b \in \mathcal{M}$ in such a way that they behave like a third anyon $c \in \mathcal{M}$. We denote this operation by $a \times b = c$. Like groups, fusion with the vacuum anyon returns the initial anyon so $a \times 1 = a$. As one one expect, we can use these properties to deduce that every anyon has an anti-particle a' such that $a \times a' = 1$. In the color code and toric code, every anyon is a self-inverse so $a \times a = 1$ for these codes. *Exchanging* two identical abelian anyons results in a complex phase θ which is effectively a spin. Particles that self-exchange and get a complex phase of $+1$ are called bosons while particles that self-exchange and get a phase of -1 are called fermions. Braiding anyons is the process of rotation one anyon a around the other b before a returns to its initial position, thus creating a phase called monodromy, labeled $M_{a,b}$. Fortunately for the color code, braiding anyons only results in trivial $(+1)$ and nontrivial (-1) monodromy phases. [5]

The Color Code has 9 bosons, each associated a color label r, g, b and a pauli label x, y, z. The product of three bosons of the same color is trivial and the product of three bosons of the same pauli label is also trivial since the product of two bosons in any row/column yields the third boson in that row/column. Another property is that the braiding of two bosons of the same color or pauli label is trivial. If these are both distinct, the braid yields a monodromy of -1. This gives rise to an intuition that the color code can be characterized by two toric codes together. One toric code representing the colors, and another for the pauli labels. There are also six fermions of the color code, $f_c, \overline{f}_c$ for each color $c \in \{r, g, b\}$. These can be expressed via products of pairs of bosons that braid nontrivially.

Definition 2.2. Given an anyon theory $\mathcal{M}$. An automorphism of $\mathcal{M}$ is a map $\varphi : \mathcal{M} \to \mathcal{M}$ that preserves the fusion and braiding rules of $\mathcal{M}$

For instance, for the toric code consisting of two bosons e, m, the vacuum anyon 1, and a fermion f, the only nontrivial automorphism of the anyon theory is the automorphism that swaps the e and m bosons. This indicates a topological order $\mathbb{Z}_2$ on the toric code. For the color code, we

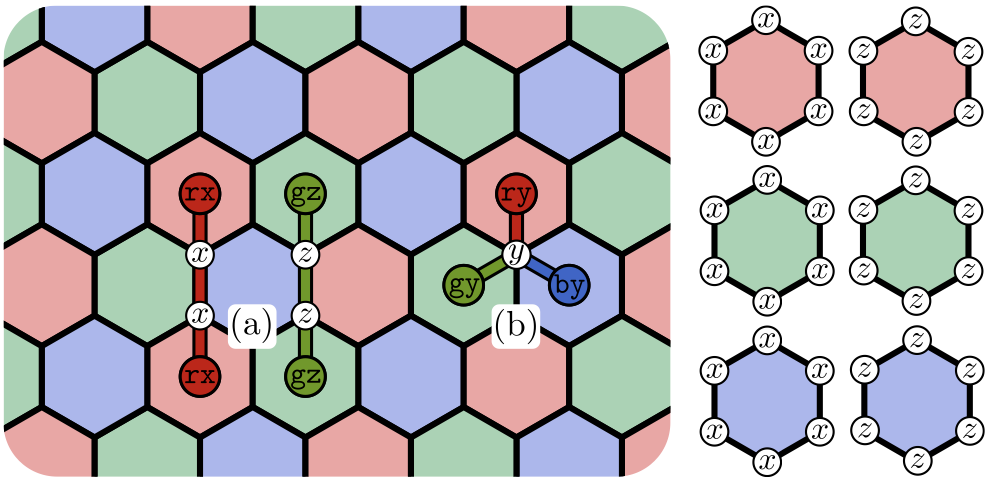

Figure 2: On the left we have a color code lattice picture and on the right we show the stabilizers on each hexagon in the tiling. A Tiling of the 2D plane with hexagons is three colorable just like in the honeycomb code. Each hexagon is assigned a color and pauli. In (a), we see edges between hexagons containing two physical qubits defines anyons on the hexagons based on the pauli rotation violating a stabilizer. In this case, the red edge contains qubit with extra X rotations and the green edge contains qubits with extra Z rotations. In (b), we see that a single qubit can define anyons for each hexagon it is part of. This diagram and explanation are credited to [5]

can permute both colors and pauli labels, leading to a larger automorphism group. Indeed, the automorphism group becomes $(S_3 \times S_3) \rtimes \mathbb{Z}_2$. where the two copies of S_3 result from permuting the color and pauli labels, as such permutations do not affect the fusion and braiding properties of the anyons. In particular, if the colors or pauli labels are permuted, the fusion rules between bosons of the same color remain true and the braiding rules between bosons of the same color/pauli label still hold. The $\mathbb{Z}_2$ subgroup comes from the fact that the bosons of the color code can conveniently be placed in a 'boson table' that is can be transposed. Note that these effectively describe the symmetries of this table and displays that the color code is two decoupled toric codes. In total, we've found 72 automorphisms of the color code. [3][5]

rx	ry	rz
gx	gy	gz
bx	by	bz

$\triangleq$

1m	em	e1
mm	ff	ee
m1	me	1e

Figure 3: Boson square with color code bosons on the left and two copies of the Toric code bosons on the right. Taken from [3]

Next, we go over condensations of the color code, which [3] describes as moving an anyon from a topologically ordered phase to a one with fewer anyons. To briefly motivate this in the context of the Dynamic Automorphism Color Code, If we start with a condensed copy of the color code, we

can perform a sequence of reversible condensations over multiple time steps, effectively evolving the underlying decoupled toric codes of the color code into a different decoupling of the toric codes and thus a difference condensed color code. Since this transformation of the anyons in the color code is cyclic, we know there's an automorphism that describes this action. The goal is then to show that we can implement all 72 automorphisms of the color code via these cycles of condensations. This is perhaps quite vague, so we make this precise with more definitions. While these terms are yet to be defined, this transformation from condensation to condensation essentially corresponds to a formulation of Instantaneous Stabilizer Groups that were mentioned for Floquet Codes. In fact, the sequence of condensations represents a new stabilizer group at each time step. First, we make the condesation language precise and then explain the result with stabilizers.

Given an abelian anyon theory $\mathcal{M}$ consider a subgroup (under fusion) of bosons $\mathcal{A}$ such that any two $a_1, a_2 \in \mathcal{A}$ braid trivially, so condensing the anyons in this subgroup cause anyons that don't braid trivially with it to become *confined* since there's some nontrivial phase associated to the interaction between these anyons. Observe that now the term condensation becomes evident since we can now define a *child anyon theory* $\mathcal{M}_{\mathcal{A}} = \{m\mathcal{A} : M_{a,m} = 1 \forall a \in \mathcal{A}\}$ since these m are considered to be *deconfined*. Confined anyons don't have well defined topological phases, giving another meaning to the word since they're confined within the parent anyon theory.

2.2 Reversible Condensations and Measurements

Definition 2.3. A *reversible condensation* is a map between two child anyon theories $\mathcal{C}_1, \mathcal{C}_2$ by lifting from $\mathcal{C}_1$ to the parent theory $\mathcal{M}$ and then condensing a different set of bosons $\mathcal{A}_2$ to get $\mathcal{C}_2$. This defines an isomorphism between the child theories $\mathcal{C}_1, \mathcal{C}_2$ defines 'compatibility' of condensations.

The authors of [3] start with parent anyon theory formed by two copies of the color code and then define two kinds of condensations, one within each color code and the other independent of each code which yields toric code variants. In particular, an *interlayer condensation* consists of condensing each color code via a fusion product of trivially braided bosons and an *intralayer condensation* involves condensing a boson of the color code separately within each code. Interlayer condensations get us a color code structure with 9 equivalence classes of bosons that obey the color code fusion and braiding properties. Intralayer condensations get us two copies of an adapted toric code.

One observes that we can effectively swap between these condensations by condensing the bosons required to get the other kind of condensation. As [3] points out, there are natural analogous formulations in terms of stabilizer codes, where condensations are effectively measurements and we can shift between stabilizer groups by performing the necessary measurements. Note that these are particularly useful formulations since by staring with an interlayer condensation and then a sequence of intralayer condensations, we realize the sequence of reversible condensations mentioned the at the beginning of this exposition. If we maintain the principle that $\widetilde{CC}$ is formed by condensing rz_1rz_2 and bz_1bz_2, note that the condensation from $\widetilde{CC}$ and $TC(a_1) \boxtimes TC(a_2)$ is reversible if and only if a_1

and a_2 are distinct colors and are either $(x,x),(y,y),(x,y),$ or (y,x) pauli labelled. Since the more important part of this code will be reversible condensations between intralayer condensations, we observe that all is needed to go from $TC(a_1) \boxtimes TC(a_2) \to TC(a_1') \boxtimes TC(a_2')$ is to have nontrivial braiding between a_1 and a_1' and a_2 and a_2' respectively, so we can go from an intralayer condensation of $a_1, a_2 \to a_1', a_2'$ provided each corresponding pair is of different color and pauli label. [5]

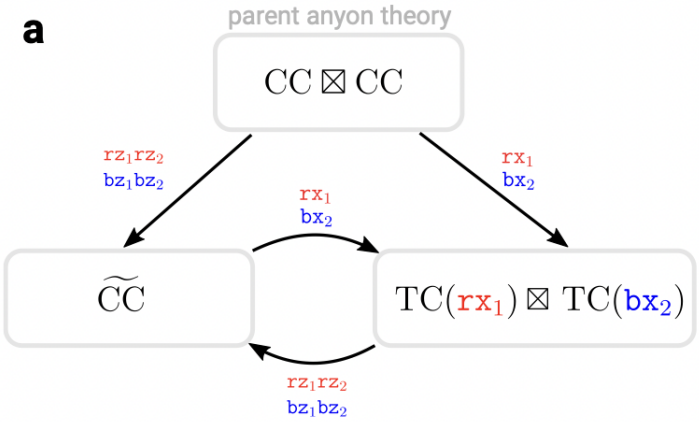

Figure 4: Part (a) of figure 3 in [3]. A diagram showing the kinds of condensations on two copies of the color code. Note that condensing rz_1rz_2 and bz_1bz_2 also condenses gz_1gz_2 due to closure under fusion.

We now define stabilizer groups. Since we're starting with a bilayer tensor product of color codes, our initial stabilizer code of the parent code is

$$\mathcal{S}(CC \boxtimes CC) = \langle P_c(X_l), P_c(Z_l) \rangle_{c \in \{r,g,b\}, l \in \{1,2\}}$$

where P represents measurmenets on plaquettes. Note that this is just the tensor product of stabilizer groups of the color code. To establish an interlayer code, where we require interactions between the first and second layer, we need to measure the vertical edges between the plaquettes we use. As a result, we define these measurements to be the operator $V(Z_1Z_2)$. With this new operator, we get

$$\mathcal{S}(\widetilde{CC}) = \langle V(Z_1Z_2), P_c(X_1X_2), P_c(Z_1) \rangle$$

where we get this definition from the action of $V(Z_1Z_2)$ on the stabilizer group $\mathcal{S}(CC \boxtimes CC)$.

For the Stabilizer Group $\mathcal{S}(TC(a_1) \boxtimes TC(a_2))$, we perform measurements of the edges on each code corresponding to the color and pauli of the bosons condensed. We provide a figure from [3] to explain. For instance, to go from $\mathcal{S}(CC \boxtimes CC) \to \mathcal{S}(TC(rx_1) \boxtimes TC(bx_2))$, we perform measurments $E_r(X_1)$ and $E_b(X_2)$, yielding a stabilizer group

$$\mathcal{S}(TC(rx_1) \boxtimes TC(bx_2)) = \langle E_r(X_1), P_r(X_1), P_b(Z_1), P_g(Z_1) \rangle \times \langle E_b(X_2), P_b(X_2), P_r(Z_2), P_g(Z_2) \rangle$$

To further explain the derivations above, we provide an image from [3] Now that we have more

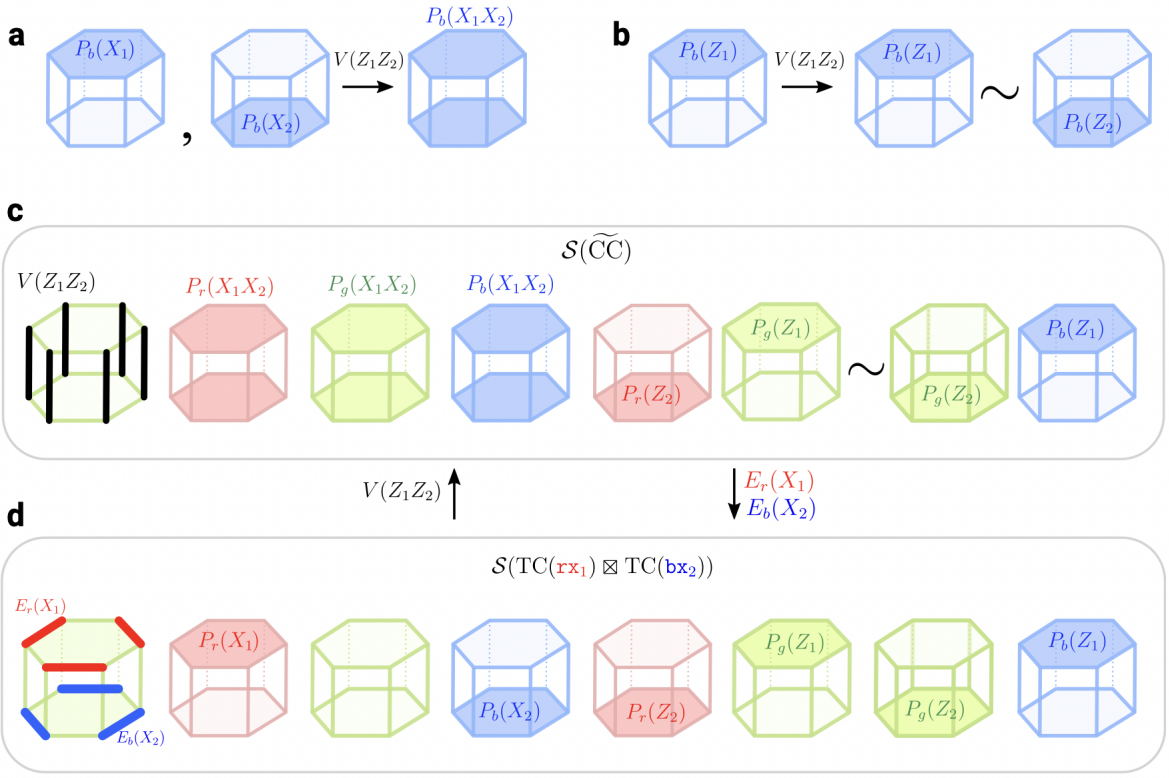

Figure 5: In (a), we see the action of $V(Z_1Z_2)$ on $P_b(X_1) \otimes P_b(X_2)$, causing the two X stabilizers to become one after the introduction of $V(Z_1Z_2)$. In (b), we see that $P_b(Z_1)$ is equivalent to $P_b(Z_2)$ after using $V(Z_1Z_2)$. In (c) and (d), we see the reversible condensations between the two child codes.

stabilizer formalisms, we can interpret the condensation languages we had before in terms of stabilizer groups. In particular, to go from $\mathcal{S}(\widetilde{CC}) \to \mathcal{S}(TC(c_1\sigma_1) \boxtimes TC(c_2\sigma_2))$, we require $V(Z_1Z_2)$ anticommutes with $E_{c_1}(\sigma_1), E_{c_2}(\sigma_2)$ which occurs when $c_1 \neq c_2$ and $\sigma_1, \sigma_2 \in \{X, Y\}$. Intuitively, since this transformation of stabilizer groups would represent an unfolding of the color code into the two copies of the toric code, this reversible condensation can be represented by an unfolding unitary $U_{c_1\sigma_1,c_2\sigma_2}$. For example, consider U_{rx_1,bx_2} that maps $\widetilde{CC} \to TC(rx_1) \boxtimes TC(bx_2)$ where we need some anticommuting operators which we form by taking $A_{i,j} = \pm \prod_{k=1}^{j} Z_1^{(i,k)} Z_2^{(i,k)}$ and $B_{i,j} = \pm X^{(i,j)}_{(3+(-1)^j)/2} X^{(i,j+1)}_{(3+(-1)^j)/2}$ where the superscripts indicate acting on the ith green plaquette and the kth qubit around the plaquette and $j = 1, \ldots, 5$. Note that these operators anticommute since $A_{i,j}B_{i',j'} = (-1)^{\delta_{ii'}\delta_{jj'}} B_{i',j'}A_{i,j}$. As discussed in [6], we can define an unfolding unitary via

$$U_{rx_1,bx_2} = \bigotimes_i \bigotimes_{j=1}^{5} \frac{A_{i,j} + B_{i,j}}{2}$$

2.3 Measurement Sequences

If we start with the effective color code $\widetilde{CC}$ at $t = 0$, we undergo reversible condensations over time for $t = 1, \ldots, t-1$ moving from intralayer condensations to other intralayer condensations. From timesteps $t-1$ to t, we go from an intralayer condensation to an interlayer condensation, so we've ended where we began at $\widetilde{CC}$. We can describe this sequence of reversible condensations for $t = 1, \ldots, t-1$ by a unitary U_{seq}. Then for the first and last condensation, we get that the unitary describing this cyclic evolution through time is $U = U^{\dagger}_{rx_1,bx_2} U_{\text{seq}} U_{rx_1,bx_2}$, so this U is one of the things we measurements that corresponds to an automorphism φ on the codespace as we discussed at the beginning of the paper.

To describe how we can perform all 72 automorphisms from unitary transformations like U_{seq}, we note that the automorphism group of the color code is generated by 3 automorphisms. φ_1 which transposes the 3×3 square of bosons of the code from left to right (along the left diagonal), φ_2 which transposes the square of bosons from right to left (along the right diagonal), and φ_3 which swaps the bottom two rows of the square. Indeed, these account for all automorphisms since transpositions with the row swaps allow us to permute colors and pauli labels. From the image, we

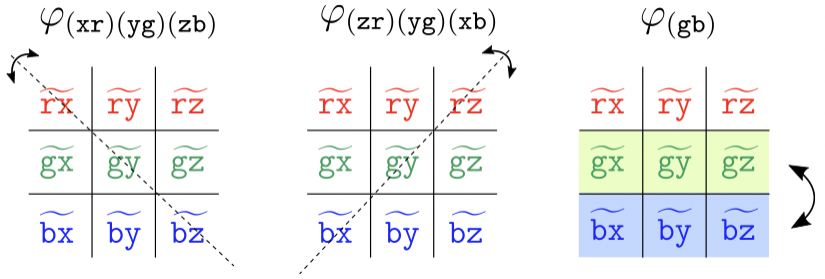

Figure 6: Actions of the Three Automorphism Generators. From right to left, we have $\varphi_1, \varphi_2, \varphi_3$ respectively. Taken from [3]

can effectively think of these automorphisms as $\varphi_1(\varphi_2)$ swapping e, m on the first (second) copy of the toric code while leaving the second (first) one alone provided the other boson in the pair is trivial. Meanwhile, φ_3 copies e/m from one code. These interpretations are better described by the actions of the automorphisms which are

$$\varphi_1 : 1e \leftrightarrow 1e, 1m \leftrightarrow 1m, e1 \leftrightarrow m1 \tag{2.1}$$

$$\varphi_2 : e1 \leftrightarrow e1, m1 \leftrightarrow m1, 1e \leftrightarrow 1m \tag{2.2}$$

$$\varphi_3 : m1 \leftrightarrow mm, 1e \leftrightarrow ee \tag{2.3}$$

which fully describes the automorphisms since other swaps can be deduced from fusion closure. Note that these automorphisms should be familiar since they describe some well-known quantum logic gates. Indeed, we notice that φ_3 does some sort of CNOT action on the bosons while φ_1, φ_2

have some sort of swap gate element to it. We see a more precise formulation of these observations in the following table from [3]

Aut/Gate	Cond./meas.	Sequence					
		$t=0$	1	2	3	4	5
$\varphi_{(\mathtt{xr})(\mathtt{yg})(\mathtt{zb})}$	cond.	$\mathtt{rz}_1\mathtt{rz}_2$ $\mathtt{bz}_1\mathtt{bz}_2$	$\mathtt{rx}_1$ $\mathtt{bx}_2$	$\mathtt{gy}_1$ –	$\mathtt{bz}_1$ $\mathtt{gz}_2$	$\mathtt{rx}_1$ $\mathtt{bx}_2$	$\mathtt{rz}_1\mathtt{rz}_2$ $\mathtt{bz}_1\mathtt{bz}_2$
$(H_1 \otimes H_3)\mathrm{SWAP}_{13}$	meas.	$V(Z_1Z_2)$	$E_r(X_1)$ $E_b(X_2)$	$E_g(Y_1)$ –	$E_b(Z_1)$ $E_g(Z_2)$	$E_r(X_1)$ $E_b(X_2)$	$V(Z_1Z_2)$
$\varphi_{(\mathtt{zr})(\mathtt{yg})(\mathtt{xb})}$	cond.	$\mathtt{rz}_1\mathtt{rz}_2$ $\mathtt{bz}_1\mathtt{bz}_2$	$\mathtt{rx}_1$ $\mathtt{bx}_2$	– $\mathtt{ry}_2$	$\mathtt{bz}_1$ $\mathtt{gz}_2$	$\mathtt{rx}_1$ $\mathtt{bx}_2$	$\mathtt{rz}_1\mathtt{rz}_2$ $\mathtt{bz}_1\mathtt{bz}_2$
$(H_2 \otimes H_4)\mathrm{SWAP}_{24}$	meas.	$V(Z_1Z_2)$	$E_r(X_1)$ $E_b(X_2)$	– $E_r(Y_2)$	$E_b(Z_1)$ $E_g(Z_2)$	$E_r(X_1)$ $E_b(X_2)$	$V(Z_1Z_2)$
$\varphi_{(\mathtt{gb})}$	cond.	$\mathtt{rz}_1\mathtt{rz}_2$ $\mathtt{bz}_1\mathtt{bz}_2$	$\mathtt{rx}_1$ $\mathtt{bx}_2$	$\mathtt{bz}_1$ $\mathtt{ry}_2$	$\mathtt{rx}_1$ $\mathtt{gx}_2$	$\mathtt{rz}_1\mathtt{rz}_2$ $\mathtt{bz}_1\mathtt{bz}_2$	– –
$\mathrm{CNOT}_{12}\mathrm{CNOT}_{34}$	meas.	$V(Z_1Z_2)$	$E_r(X_1)$ $E_b(X_2)$	$E_b(Z_1)$ $E_r(Y_2)$	$E_r(X_1)$ $E_g(X_2)$	$V(Z_1Z_2)$	– –

Figure 7: Measurement/Condensation Sequences that form the Automorphism generators for the 72 automorphisms

2.4 Error Correction

The error correction described in this section will consist of the DA Color Code on the torus.

2.4.1 Padding Sequences for Automorphisms

While previously we saw that we can generate all automorphisms by using $\varphi_1, \varphi_2, \varphi_3$, on a torus, we cannot realize all automorphisms immediately via condensation sequences. In particular, we can only realize a subgroup of automorphisms, so we use *padding* to get all the automorphisms by adding in an additional measurement sequence between automorphisms that implements the trivial automorphism. While this doesn't change the structure on the code, it allows us to put detectors/checks in for error correction. This provides many choice for such a padding sequence. For instance, consider the sequence that is just the identity, but choosing a sequence that implements the same automorphism multiple times allows for better placings of detectors. Let π represent an automorphism used for a padding sequence with $\pi^2 = I$.

Given an automorphism φ, we know that we can implement φ with the automorphism group generators $\varphi_1, \varphi_2, \varphi_3$, so $\varphi = \prod_{k\in\{1,2,3\}} \varphi_k$. When we put in padding sequences in between every automorphism, the new decomposition becomes

$$\varphi = \pi \circ \left(\prod_{k\in\{1,2,3\}} \pi \circ \varphi_k \circ \pi \right) \circ \pi$$

so when performing measurements, we get measurement blocks corresponding to automorphisms $B_k = \pi \circ \varphi_k \circ \pi$ and such blocks are what we refer to as padding sequences, as indicated by

the conjugation of π. Note that in the circuit world, we use $M(\varphi)$ to refer to the measurement corresponding to φ, so $B_k = M(\pi) \circ M(\varphi_k) \circ M(\pi)$. The use of padding sequences is crucial to the error detection model since detectors will come between each error padding block or after two padding measurements.

A particular π that [3] chooses is $\varphi_{rb,zx}$ which swaps the blue and red colors while also swapping the z and x pauli labels which causes behavior similar to the honeycomb code, so by padding with these sequences you're basically manifesting some of the structure of the honeycomb code into this DA color code. This is an example of how dynamic automorphism codes can implements other codes in its protocols. There's another reason for this choice. The generating automorphisms don't generate the full ISG for the DA Color Code. They almost generate the full thing but φ_1 doesn't measure $P_b(X_1, X_2)$, φ_2 doesn't measure $P_r(X_1, X_2)$, and φ_3 doesn't measure $P_r(Z_1)$, yet $\varphi_{rb,zx}$ can generate the full ISG, so using this sequence to pad the automorphism sequences allows for proper generation of logical qubits in the DA color code.

2.4.2 Identifying Errors

For a pauli in layer $l = 1, 2$ and round n, let σ_n^l where σ is a Pauli. If we encounter an error that is σ_n^l that occurs before $n+1$, we denote such an error by ε_n^l. Meanwhile, if this kind of error occurs in $n+1$, we denote this error by ε_{n+1}^l. If the error isn't that of n or $n+1$, we can denote this error via $i\varepsilon_{n+1}^l\varepsilon_n^l$, so this kind of error is basically an error that occurs after n and $n+1$. This describes Pauli errors for this code. For measurement errors on a qubit, we just have to locate the spacetime location of the qubit that incurs this error and the recovery operator at this spacetime location is simply the check applied at this timestep. As a result, we've defined an error basis for the DA color code.

2.4.3 Detectors/Logical Checks for the DA Color Code

As mentioend before we place detectors in between every padding sequence B_k or it simply terminates within the padded sequence. This guarantees that the detectors support won't include other measurements we need to perform. We can form such a set of detectors constructively by looking at the combinations of measurements in different rounds. In the DA color code, these combinations of measurements occur on a plaquette of a specific color, so we can identify detectors with a color and plaquette. Next we need to determine which Pauli these checks are looking for. Due to the definition of our error basis, the pauli label of the detector will be exactly the pauli that anticommutes with errors.

We also need bilayer detectors, so combined with single layer detectors we have a basis of detectors formed by finding times when the plaquettes: $P_c(\sigma_1), P_c(\sigma_2), P_c(X_1X_2), P_c(X_1Y_2), P_c(Y_1X_2)$, $P_c(Z_1Z_2), P_c(Y_1, Y_2)$ are measured. We then take products of these plaquettes to find combinations of plaquette measurements that yield constant values in the absence of errors. To determine

the support of these plaquettes, we mark time steps where these plaquettes can be inferred from measurements.

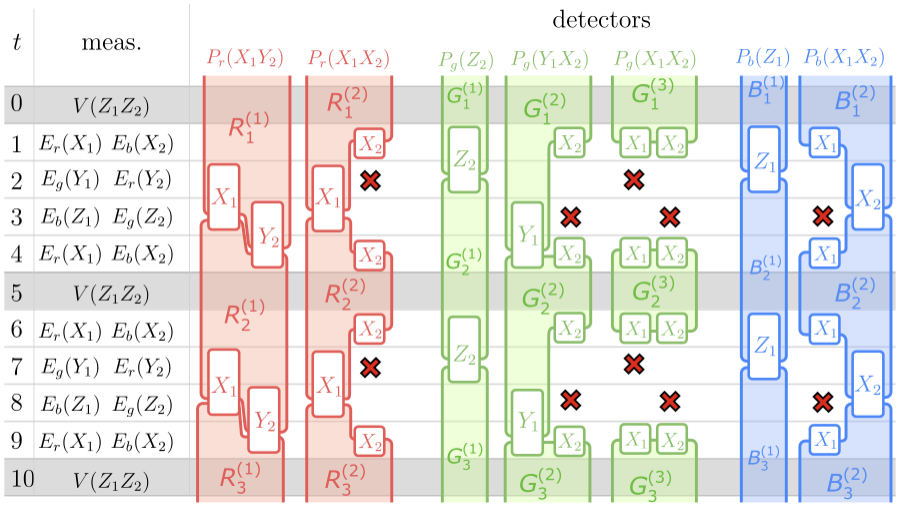

Figure 8: Implementation of the padding automorphism π with detectors. The colored regions represent the support of the detectors and the Paulis represent the check that is being performed to determine the value of the detector. The x's mark the times and points when the plaquette values are randomized by measurements

3 Conclusion

Dynamic Automorphism Codes are an interesting class of codes that further develop the idea of dynamically generating logical qubits from the Floquet Code. Interestingly, these dynamic codes model the flow of quantum information throughout time, creating a more natural picture of error correction. Nonetheless, the preservation of information throughout time within these codes points to the idea that while Stabilizer Groups change over time, these changes are modeled by automorphisms on the code space.

In this review paper, I discussed some results in [3] and built up context needed to understand the formulations in this paper such as the Floquet Code, Anyon Theory, and the 2D Color code. [3] discusses much more than the results mentioned in this paper. For instance, they utilize Topological Quantum Field Theory to gain a more mathematical perspective of the DA Color Code. Moreover, they discuss the 3D DA Color Code and the 2D Color Code on different lattices. Overall, while their paper seems like multiple papers in one, these different kinds of automorphism codes display new features compared to the 2D Color Code. For instance, the $2D$ DA Color Code on a triangular lattice implements the entire clifford group while the $3D$ DA Color Code implements universal computation.

These dynamic codes are much different in nature than stabilizer or subsystem codes since their logical operators also change throughout time. Nonetheless, while this is an entirely different view

on quantum error correction, as [3] points out, the distance of such dynamic codes is unclear and the practicality of these codes leads to exhaustive computation since one would have to search over the space of measurement sequences to find the sequence that implements an automorphism best.

Overall, while I didn't understand much of what was going on (mainly just detectors and the error basis constructions) beyond the larger picture of what the goal of dynamic automorphism codes are, I thought it was particularly interesting that CSS codes could be implemented via automorphism sequences, bringing the question of which non-dynamic codes can be implemented dynamically and what is the advantage of using a dynamic approach?

References

[1] Nikolas P. Breuckmann. Quantum Subsystem Codes: Their Theory and their Use.

[2] Matthew B. Hastings and Jeongwan Haah. Dynamially Generated Logical Qubits.

[3] Margarita Davydova, Nathanan Tantivasadakarn, Shankar Balasubramanian, and David Aasen. Quantum Computing on Dynamic Autormophism Codes

[4] Aleksander Marek Kubica. The ABCs of the color code: A study of topological quantum codes as toy models for fault-tolerant quantum computation and quantum phases of matter.

[5] M. S. Kesselring, J. C. M. de la Fuente, F. Thomsen, J. Eisert, S. D. Bartlett, and B. J. Brown, Anyon condensation and the color code (2022).

[6] D. Aasen, J. Haah, Z. Li, and R. S. K. Mong, Measurement quantum cellular automata and anomalies in floquet codes (2023), arXiv:2304.01277 [quant-ph].