Fault-tolerant quantum computation

Theory and practice

Vuillot, Christophe

DOI
10.4233/uuid:7cb715f4-eaf0-4526-8552-9f97cc864383

Publication date
2020

Document Version
Final published version

Citation (APA)
Vuillot, C. (2020). *Fault-tolerant quantum computation: Theory and practice*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:7cb715f4-eaf0-4526-8552-9f97cc864383

FAULT-TOLERANT QUANTUM COMPUTATION: THEORY AND PRACTICE

FAULT-TOLERANT QUANTUM COMPUTATION: THEORY AND PRACTICE

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology,
by the authority of the Rector Magnificus prof. dr. ir. T.H.J.J. van der Hagen,
Chair of the Board of Doctorates,
to be defended publicly on Wednesday 15th, January 2020 at 12:30 o'clock

by

Christophe VUILLOT

Master of Science in Computer Science,
Université Paris Diderot, Paris, France,
born in Clamart, France.

This dissertation has been approved by the promotor.

promotor: prof. dr. B. M. Terhal

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof. dr. B. M. Terhal	Technische Universiteit Delft, *promotor*
Independent members:	
Prof. dr. C.W.J. Beenakker	Universiteit Leiden
Prof. dr. L. DiCarlo	Technische Universiteit Delft
Prof. dr. R.T. König	Technische Universität München
Dr. A. Leverrier	Inria Paris
Prof. dr. ir. L.M.K. Vandersypen	Technische Universiteit Delft
Prof. dr. R.M. de Wolf	Centrum Wiskunde & Informatica

Keywords: quantum computing, quantum error correction, fault-tolerance

Printed by: Gildeprint - www.gildeprint.nl

Front: Kandinsky Vassily (1866-1944), *Auf Weiss II*, 1923
Photo © Centre Pompidou, MNAM-CCI, Dist. RMN-Grand Palais / image Centre Pompidou, MNAM-CCI

ISBN 978-94-6384-097-2

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

This thesis is dedicated to my daughter Andréa and her mother Lisa.

CONTENTS

SUMMARY

Quantum computation is the modern version of Schrödinger's cat experiment. It is backed up in principle by the theory and thinking about it can make people equally uncomfortable and excited. Besides, its practical realization seems so extremely challenging that some people even doubt it is possible. On the other hand, we are nowadays much closer to realizing quantum computation and in addition, it has much more implications than Schrödinger's original cat experiment.

One of the major difficulties in realizing quantum computation is the inevitable presence of noise in realistic quantum devices which makes the direct realization of quantum computers impossible. In order to protect quantum information and quantum processes against noise, quantum error correction and fault-tolerance have been devised. Although the gap between experiments and the requirements of fault-tolerance is still daunting, the field of quantum error correction and fault-tolerance extends and influences architectural decisions from the hardware to the ideal quantum programs that we want to run. That is why it has the potential to make or break the practicality of quantum computation and a lot of research effort goes into this field.

In this thesis we investigate and improve several aspects of fault-tolerant schemes and quantum error correction. We implement an experiment which validates on a small device the usefulness of fault-tolerance for quantum computation. We investigate the advantages of harnessing quantum continuous degrees of freedom present in the lab to protect discrete quantum information in a scalable way. We establish a framework to analyze the fault-tolerant properties of code deformation techniques which are versatile techniques to process quantum information protected by an error correcting code. We also present some novel code deformation techniques with the potential to increase reliability. Finally we define a new class of quantum error correcting codes, quantum pin codes, with built in capabilities for fault-tolerant quantum gates. We give some practical constructions and show some protocols with interesting parameters.

The roads towards universal and fault-tolerant quantum computation are still steep but research efforts are pushing in the right directions.

PREFACE

WHY SHOULD we build a quantum computer? With this foreword I would like to try to present a case for investing in building a quantum computer. This case mainly consists in what makes me excited about working in the field of quantum computing. Besides, this is the humble opinion of a young and inexperienced researcher probably lacking a more broad view of science, technology and academia.

One of the fascinating aspects of science is that advancing our understanding actually allows us to modify the world around us. Not necessarily in a useful way: making fireworks, drawing beautiful fractals, going to the moon or trapping and monitoring the presence of a single photon for milliseconds are not really directly useful. When we uncover such ways of conjuring something new, and when someone takes on themselves to figure out how to use it, it often has dramatic consequences. However, the marvelous part to me is more in the former than the latter.

The field of computing is a prime example which obviously dramatically changed the world. It is also constructed from many such marvelous understandings that were necessary to produce the final object, a computer. The original question that was raised was: is it possible to compute using physical mechanized processes, or really any kind of physical process? The first surprising results about this question was that there are problems that cannot be solved by any machine we could come up with. Problems that would require such a machine computing forever before giving an answer. The second surprising answer is that there is a simple machine that can perform all the feasible computations. Sure enough this machine, the computer, got built at some point. On the physics side, an ever growing understanding of condensed matter and its electrical properties brought us to a place where we can build these computers so efficiently and so small that they can do billions of operations per seconds and we can store trillions of bits in our pockets. Computers are truly fascinating objects even before considering all the work they do for us.

The field of quantum physics is also a major example filled with surprising understandings. From thought experiments about light and particles, it brought us to a place where real experiments can measure time more precisely than we will ever be able to experience. All this progress highlights a powerful aspect of science which is that well crafted theories can sometimes predict more than what they were designed for. In other words, they sometimes still hold even when they are pushed to their farthest-reaching conclusions. That is why scientists so much like pushing theories to their limits as it allows them to discover new aspects of the world or limitations of the theory most of the time. In the case of quantum mechanics, after many such successful extensions, we are still in this pushing phase. Maybe the first notable success was to apply the particle-wave duality broadly to any particle. It has been followed by the successful violation of Bell's inequalities which continues even today to make people unsure about the status of our

understanding.

But remarkably we have not finished to push quantum mechanics to its most dramatic conclusion, namely the possibility of quantum computation. Schrödinger's cat thought experiment was designed to test our faith in quantum physics by making ourselves uncomfortable about its consequences. Yet so far they are still holding and larger and larger systems have been put in superposition. The idea of a quantum computer is even more dramatic and can test quantum physics even further and should make ourselves even more uncomfortable.

Indeed, the field of computing and quantum physics entered a collision phase a few decades ago when it was noted that computers are pretty inefficient at simulating quantum processes. This presented a direct challenge to the universality of computers in practice and sparked many research questions in both fields.

This exploration is now well underway both from the theoretical side and the experimental side but its outcome, sitting at the interface, is still very much clouded. If possible to build, what is exactly the advantage quantum computers can provide compared to classical computers, is it only polynomial, exponential for some problems, or non-existent? Is it actually possible to build this machine and harness complex quantum processes on a macroscopical level? If not possible to build, what parts of quantum physics are unreliable and should be modified?

This perspective is enough for me to want to build at least one quantum computer at least once and use it as a scientific tool to explore quantum physics and computation.

Christophe Vuillot
Delft, September 2019

1

INTRODUCTION

The context and fundamental notions of quantum mechanics, quantum computation, quantum error correction and fault-tolerance as well as a reading guide for the rest of this thesis are presented.

1

Since appearing first as a mathematical trick to solve the unexplained radiation of a black body in the ultraviolet regime [1], quantum mechanics has been gradually expanding and demonstrated that it actually stands as a fundamental theory governing the behavior of elementary physical systems. Some of its most far reaching implications for the physical world have now been experimentally demonstrated, notably the violation of Bell inequalities [2]. As of today, the main consequence of quantum mechanics lacking an experimental demonstration is the possibility of quantum computation and it is probably the most dramatic one. The main reason for this, is the tremendous challenge that controlling elementary quantum degrees of freedom of a system represents, together with the impossibility of perfect controls and perfect isolation from outside perturbations. This thesis is focused on the realization of quantum computation in its noiseless definition. For this to be realizable it is therefore imperative to introduce a layer between the physical systems realized in laboratories and the quantum algorithm acting ideally on perfect systems. This layer implements protocols designed to deal with the inevitable errors and disturbances in the system in a way that allows to drive down arbitrarily the error rates seen by the layer of quantum algorithms above. This is the field of quantum error correction and fault-tolerance. It is the interface between hardware and software whose performance could make or break the possibility of implementing large scale quantum algorithm on actual experimental devices.

In this context, this thesis presents a series of results contributing to bridging the gap between experiments and systems implementing large scale algorithms by providing a proof of concept, studying and improving quantum error correcting protocols and devising new quantum error correcting codes.

In the present chapter we give an introduction and present the terminology needed for the rest of the thesis. For some more comprehensive and well crafted introductory works, we refer the reader to [3–7].

1.1. INTRODUCTION TO QUANTUM COMPUTING

1.1.1. QUANTUM MECHANICS

The theory of quantum mechanics can be nicely axiomatized in a few principles. We present them and illustrate their main features here.

STATES

The first principle concerns the states a physical system can be in. It posits that they can always be described using the structure of a Hilbert space over the complex numbers. We usually denote the complex Hilbert space as $\mathscr{H}$ and the vectors in it using the Dirac ket notation, like $|\psi\rangle \in \mathscr{H}$. Being a complex Hilbert space, $\mathscr{H}$ possesses an inner product between vectors, denoted $\langle\phi|\psi\rangle \in \mathbb{C}$, which is conjugate symmetric, $\langle\phi|\psi\rangle = \langle\psi|\phi\rangle^*$, right-linear, $\langle\phi|\alpha\psi_1 + \beta\psi_2\rangle = \alpha\langle\phi|\psi_1\rangle + \beta\langle\phi|\psi_2\rangle$, and positive definite, $\langle\psi|\psi\rangle = 0 \Leftrightarrow |\psi\rangle = 0$. The physical states, more precisely, are the rays in $\mathscr{H}$, represented by normalized vectors $|\psi\rangle$ ($\langle\psi|\psi\rangle = 1$), equivalent under multiplication by a global phase: $|\psi\rangle \sim e^{i\theta}|\psi\rangle$. As hinted in the above notation the Dirac bra notation $\langle\psi|$ denotes the linear form dual to the state $|\psi\rangle$ and is convenient to form inner products.

The interpretation given to two states, $|\psi\rangle$ and $|\phi\rangle$, that are orthogonal, i.e. $\langle\psi|\phi\rangle = 0$, is that they are perfectly distinguishable by some measurement, whereas they are more

and more similar and less and less distinguishable when their overlap grows. That is to say that the dimension of $\mathcal{H}$ gives the number of perfectly distinguishable states the system can be in, given for example by vectors forming an orthonormal basis of $\mathcal{H}$. Linear combinations with complex coefficients of basis vectors by definition also belong to $\mathcal{H}$ and therefore also constitute valid states for the system. They are referred to as superpositions of basis states. Since there is in general no preferred basis, being in superposition is mostly a matter of point of view.

We can now readily explain that the state space of the union of two physical systems is given by the tensor product of their respective Hilbert spaces, denoted as $\mathcal{H}_1 \otimes \mathcal{H}_2$. Indeed if the first system has d_1 perfectly distinguishable states and the second one d_2 then surely the union has $d_1 \times d_2$ perfectly distinguishable states. Since the joint system should still fall in the formalism, the tensor product of the Hilbert spaces is the correct choice.

The states described this way are also called pure states, as they exactly and fully describe the state of a system. In certain situations the information about the state of a system is not complete and one has to use a probabilistic mixture of pure states to describe it. That is to say, a set $\{(p_i, |\psi_i\rangle)\}$ where p_i represent the probability for the system to be in state $|\psi_i\rangle$, hence with $\sum p_i = 1$. This representation is not unique as different choices of mixtures can produce states which are virtually equivalent operationally. That is why it is actually more convenient to represent mixed states as density operators. Namely, $\rho = \sum p_i |\psi_i\rangle \langle\psi_i|$, a weighted sum of the outer products of the states in the mixture. Density operators can be simply characterized as Hermitian operators with non-negative eigenvalues and trace one. For pure states, $\rho = |\psi\rangle \langle\psi|$ is a rank-one projection onto the state $|\psi\rangle$.

Interestingly mixed states can arise as soon as one discards part of a system originally in a pure state: this is the phenomenon referred to as entanglement. The smallest example of entanglement can be constructed using two systems with Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$ and two orthogonal states from each, $|\psi_1\rangle, |\psi_1^\perp\rangle \in \mathcal{H}_1$ and $|\psi_2\rangle, |\psi_2^\perp\rangle \in \mathcal{H}_2$. Using these states one can, for example, form the following state

$$|\Psi\rangle = \frac{1}{\sqrt{2}} \left(|\psi_1\rangle \otimes |\psi_2\rangle + |\psi_1^\perp\rangle \otimes |\psi_2^\perp\rangle \right). \tag{1.1}$$

The state $|\Psi\rangle$ is not separable, meaning that there does not exist any way to write it in product form: $|\Psi\rangle \neq |\tilde{\psi}_1\rangle \otimes |\tilde{\psi}_2\rangle$. States that are not separable are called entangled states. If one were to discard the second part of the system, $\mathcal{H}_2$, there is no way to write the remaining state on the first system as a pure state. It becomes a mixed state given by

$$\rho_1 = \frac{1}{2} \left(|\psi_1\rangle \langle\psi_1| + |\psi_1^\perp\rangle \langle\psi_1^\perp| \right). \tag{1.2}$$

The formal rule to discard a part of a system is to take the partial trace on density matrices.

OBSERVABLES AND MEASUREMENTS

The second principle states that any observable physical quantity is represented by a Hermitian operator, say $\hat{O}$, acting on $\mathcal{H}$. So we have that $\hat{O}^\dagger = \hat{O}$, where $\hat{O}^\dagger$ is the transpose and conjugate operator to $\hat{O}$. Such an $\hat{O}$ is therefore called an observable. Being

Hermitian, $\hat{O}$ can be diagonalized to an orthogonal set of eigenvectors and the spectrum of $\hat{O}$ gives the value of the observable can take. The rule to describe a measurement of $\hat{O}$ is as follows: the outcome of the measurement is one of the eigenvalues of $\hat{O}$ with probability given by the overlap which the measured state has with the corresponding eigenspace. The resulting state after measurement is given by its orthogonal projection onto the outcome eigenspace.

This type of measurement is called a projective measurement, it is fully characterized by the set of projectors onto the different eigenspaces of $\hat{O}$. Denoting the different projectors $\{P_j\}$, the measured state $|\psi\rangle$ and the output state after oberving outcome j, $|\psi_j\rangle$, we have

$$\mathbb{P}(\text{outcome } j) = \left|\left|P_j|\psi\rangle\right|\right|^2, \qquad |\psi_j\rangle = \frac{P_j|\psi\rangle}{\left|\left|P_j|\psi\rangle\right|\right|}. \tag{1.3}$$

Note that for a set of projectors, $\{P_j\}$, to form a valid measurement they have to obey $\sum_j P_j = \mathbb{1}$. In a mixed state situation the description becomes

$$\mathbb{P}(\text{outcome } j) = \mathrm{Tr}(P_j\rho), \qquad \rho_j = \frac{P_j\rho P_j}{\mathrm{Tr}(P_j\rho)}. \tag{1.4}$$

Measurements can also be presented as a more general procedure than projective measurement, which is equivalent once we add unitary evolution and auxiliary systems.

EVOLUTION

The third principle states that the evolution of an isolated quantum system is governed by the Schrödinger equation, given here for pure and mixed states

$$i\hbar\frac{\mathrm{d}}{\mathrm{d}t}|\psi(t)\rangle = \hat{H}(t)|\psi(t)\rangle, \qquad i\hbar\frac{\mathrm{d}}{\mathrm{d}t}\rho(t) = \left[\hat{H}(t),\rho(t)\right], \tag{1.5}$$

where $\hat{H}$ is the observable corresponding to the total energy of the system and $[A,B] = AB - BA$ denotes the commutator. One can deduce from this equation that the transformation between any two instants t_1 and t_2 is always a unitary transformation, it is linear and preserves the inner product. So any evolution of a quantum system can be written using some unitary $U \in \mathrm{U}(\mathscr{H})$

$$|\psi(t_2)\rangle = U|\psi(t_1)\rangle, \qquad \rho(t_2) = U\rho(t_1)U^\dagger. \tag{1.6}$$

The evolution of a system which is not isolated can be deduced by adding any necessary auxiliary system to it until one obtains an isolated system, computing the unitary evolution of the full system and then discarding the auxiliary system. This kind of general evolution is also called a quantum channel. One can verify that these transformations on density operators are the completely positive and trace preserving maps. Completely positive means that it maps positive operators to positive operators, even when the map is applied only on part of a larger system. Such maps can be represented using what is called a Kraus representation, see for example [6], which says

$$\mathscr{E}(\rho) = \sum_{j=1}^{m} A_j\rho A_j^\dagger, \tag{1.7}$$

where the A_j are called the Kraus operators which can be any linear operator. They need to obey $\sum_j A_j^\dagger A_j = \mathbb{1}$ for the channel to be trace preserving. For a given channel, different sets of Kraus operators can be found to represent it, the different representation are related by a unitary transformation. One can operationally interpret such a Kraus representation as a classical mixture of different outcome states given by each term in the sum with the probabilities given by the norm of the terms.

1.1.2. ELEMENTARY QUANTUM SYSTEMS

In the rest of this thesis we encounter primarily two types of elementary quantum systems, qubits and oscillators.

QUBITS

A qubit is arguably the simplest quantum system, it has exactly two distinguishable states, we denote them $|0\rangle$ and $|1\rangle$, and we have $\mathscr{H} = \mathbb{C}^2$. These basis states are usually referred to as the computational basis states. The name comes from the fact that a classical bit has two states, 0 and 1. Any qubit pure state $|\psi\rangle$ can be written as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \qquad (\alpha,\beta) \in \mathbb{C}^2, |\alpha|^2 + |\beta|^2 = 1. \tag{1.8}$$

Some important operators acting on qubits are the Pauli operators, represented as matrices they are:

$$X = \begin{array}{c} \\ |0\rangle \\ |1\rangle \end{array} \begin{array}{c} \langle 0| \quad \langle 1| \\ \end{array} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{array}{c} \\ |0\rangle \\ |1\rangle \end{array} \begin{array}{c} \langle 0| \quad \langle 1| \\ \end{array} \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{array}{c} \\ |0\rangle \\ |1\rangle \end{array} \begin{array}{c} \langle 0| \quad \langle 1| \\ \end{array} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{1.9}$$

The Pauli matrices are at the same time Hermitian and unitary and so can be interpreted both as observables and operations on qubits. Together with the identity and the phase i, they form an algebra

$$X^2 = Y^2 = Z^2 = \mathbb{1}, \qquad XY = iZ, \qquad YZ = iX, \qquad ZX = iY, \tag{1.10}$$

and follow anti-commutation relations

$$\{X,Y\} = XY + YX = \{Y,Z\} = \{Z,X\} = 0. \tag{1.11}$$

Moreover with the identity, they form an orthogonal basis for the linear operators acting on $\mathscr{H}$, that is to say any 2×2 complex matrix, M, can be decomposed as

$$M = \alpha\mathbb{1} + \beta X + \gamma Y + \delta Z, \tag{1.12}$$

where $(\alpha,\beta,\gamma,\delta) \in \mathbb{C}^4$ and M is Hermitian if and only if $(\alpha,\beta,\gamma,\delta) \in \mathbb{R}^4$. In particular any qubit density matrix ρ can be parameterized in the following way

$$\rho = \frac{1}{2}(\mathbb{1} + r_X X + r_Y Y + r_Z Z), \tag{1.13}$$

where $\boldsymbol{r} = (r_X, r_Y, r_Z) \in \mathbb{R}^3$ is a real vector with norm at most unity, $|\boldsymbol{r}| \leq 1$.

The Pauli operators have eigenvalues +1 and −1 and their eigenstates form three important bases called the X, Y and Z bases and given by

$$\left\{|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}\right\}, \qquad \left\{|+i\rangle = \frac{|0\rangle + i\,|1\rangle}{\sqrt{2}}, |-i\rangle = \frac{|0\rangle - i\,|1\rangle}{\sqrt{2}}\right\}, \qquad \{|0\rangle, |1\rangle\}. \tag{1.14}$$

Most of the time we will deal not with one but many qubits. The Hilbert space for n qubits is $\mathscr{H}^{\otimes n} = \mathbb{C}^{2^n}$. The Pauli operators extend to the multi-qubit case simply by forming tensor product of Pauli operators. They form a group called the Pauli group on n qubits and denoted $\mathscr{P}_n$. Using the fact that $ZX = iY$, it is possible to represent every element of the group using a binary symplectic vector and a phase using the notation

$$P(\boldsymbol{e}) = X^{u_1} Z^{v_1} \otimes \cdots \otimes X^{u_n} Z^{v_n}, \qquad \boldsymbol{e} = (\boldsymbol{u}, \boldsymbol{v}) \in \mathbb{Z}_2^{2n}, \tag{1.15}$$

so that

$$\mathscr{P}_n = \left\{i^s P(\boldsymbol{e}) \,|\, \forall s \in \{0,1,2,3\}, \forall \boldsymbol{e} \in \mathbb{Z}_2^{2n}\right\}. \tag{1.16}$$

The commutation relation between two elements can then be computed in term of the symplectic inner product

$$P(\boldsymbol{e})P(\boldsymbol{e}') = (-1)^{\boldsymbol{e}'\Omega\boldsymbol{e}^{\mathrm{T}}} P(\boldsymbol{e}')P(\boldsymbol{e}), \qquad \Omega = \begin{pmatrix} 0 & \mathbb{1}_{n\times n} \\ -\mathbb{1}_{n\times n} & 0 \end{pmatrix}. \tag{1.17}$$

The n-qubits Pauli operators with trivial phases also form a basis for the n-qubits operators.

1D PARTICLES AND OSCILLATORS

Oscillators are ubiquitous in many experimental setups and a very natural elementary quantum system. In striking contrast to qubits, the Hilbert state space of one oscillator is infinite-dimensional. Given by $\mathscr{H} = L^2(\mathbb{R})$, it is the space of square-integrable complex functions of the real line. The direct interpretation of this Hilbert space comes more naturally from a 1D particle interpretation, where a point-like particle can evolve in 1D space and is characterized by its wave function, $\psi \in L^2(\mathbb{R})$. This means that the probability density to find the particle at the position $x \in \mathbb{R}$ is given by $|\psi(x)|^2$, and is therefore normalized $\int_{-\infty}^{\infty} |\psi(x)|^2 \mathrm{d}x = 1$. The inner product between ϕ and ψ is given by $\int_{-\infty}^{\infty} \phi^*(x)\psi(x)\mathrm{d}x$.

With this Hilbert space being infinite-dimensional, some aspects are more subtle than for finite-dimensional spaces. For example, $\mathscr{H}$ possesses some bras (linear forms) which do not have valid dual kets in $\mathscr{H}$. The most notable and useful examples are

$$\langle x_0| : |\psi\rangle \mapsto \psi(x_0) = \int_{-\infty}^{\infty} \delta(x_0 - x)\psi(x)\mathrm{d}x, \quad \langle p_0| : |\psi\rangle \mapsto \mathrm{FT}[\psi](p_0) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \mathrm{e}^{-ip_0 x}\psi(x)\mathrm{d}x, \tag{1.18}$$

where $\mathrm{FT}[\psi]$ denotes the Fourier transform of ψ. The natural objects which would be dual kets to these bras are Dirac delta distributions and plane waves:

$$|x_0\rangle = \delta(x_0 - x), \qquad |p_0\rangle = \frac{\mathrm{e}^{ip_0 x}}{\sqrt{2\pi}}, \tag{1.19}$$

which are not square-integrable functions. However, they each have all the properties of an orthogonal basis and can be seen as some limit of states which are valid. Hence, we will use them conveniently as if they were valid kets. These two bases are called respectively the dimensionless position and momentum bases. Since we work with dimensionless variables we set the reduced Planck constant to unity: $\hbar = 1$. A wave function ψ can be seen as decomposed over the position basis, i.e. $\{|x_0\rangle\}_{x_0\in\mathbb{R}}$, and its Fourier transform, $\mathrm{FT}[\psi]$ as the same state but decomposed over the momentum basis, i.e. $\{|p_0\rangle\}_{p_0\in\mathbb{R}}$. These two bases are also the eigenbases of the position and momentum observables, which we denote as $\hat{x}$ and $\hat{p}$, and therefore can be written

$$\hat{x} = \hat{x}^\dagger = \int_{-\infty}^{\infty} x\,|x\rangle\langle x|\,\mathrm{d}x, \qquad \hat{p} = \hat{p}^\dagger = \int_{-\infty}^{\infty} p\,|p\rangle\langle p|\,\mathrm{d}p. \tag{1.20}$$

The position and momentum operators act in the position basis as follows

$$\hat{x}\,|\psi\rangle = |x\psi\rangle\,, \qquad \hat{p}\,|\psi\rangle = |\frac{1}{i}\frac{\partial\psi}{\partial x}\rangle\,. \tag{1.21}$$

Using this representation, one can readily check that the position and momentum operators satisfy the canonical commutation relation:

$$\left[\hat{x},\hat{p}\right] = i\mathbb{1}. \tag{1.22}$$

This commutation relation and the spectrum of the two operators completely characterizes the system. For the system to describe an harmonic oscillator one has the following Hamiltonian, specifying the energy of the system

$$\hat{H} = \frac{\omega}{2}\left(\hat{x}^2 + \hat{p}^2\right), \tag{1.23}$$

for some real value ω. This Hamiltonian is more easily analyzed by introducing new operators called annihilation and creation operators which are conjugate of one another and are denoted respectively a and $a^\dagger$

$$a = \frac{\hat{x} + i\hat{p}}{\sqrt{2}}, \qquad a^\dagger = \frac{\hat{x} - i\hat{p}}{\sqrt{2}}. \tag{1.24}$$

Using these operators the Hamiltonian rewrites

$$\hat{H} = \omega\left(a^\dagger a + \frac{1}{2}\right) = \omega\left(\hat{n} + \frac{1}{2}\right), \tag{1.25}$$

where we also defined the number operator $\hat{n} = a^\dagger a$. Using the fact that $\hat{n}$ is a positive operator by construction and the commutation relation $\left[a, a^\dagger\right] = \mathbb{1}$, it is fairly straightforward to find that the spectrum of $\hat{n}$ is $\mathbb{N}$ (in our notation we consider $0 \in \mathbb{N}$) and that the eigenstates, also called Fock states and denoted as $|n\rangle$, are such that

$$\forall n \in \mathbb{N}, \qquad a\,|n\rangle = \sqrt{n}\,|n-1\rangle\,, \quad a^\dagger\,|n\rangle = \sqrt{n+1}\,|n+1\rangle\,. \tag{1.26}$$

These Fock states are the energy eigenstates of the harmonic oscillator, they are evenly spaced in energy and the indexing number n is often referred to as the number of elementary energy excitation of the state also called the photon number. The Fock states

form an orthonormal basis of the Hilbert space, $\{|n\rangle\}_{n\in\mathbb{N}}$, and correspond in the position basis to the Hermite functions, which can be derived from (1.21), (1.24) and (1.26).

We now introduce the equivalent to the Pauli matrices on qubits for oscillators which are formed by exponentiation of the position and momentum operators and are called displacement operators:

$$U(u,v) = \mathrm{e}^{iu\hat{x}+iv\hat{p}}, \tag{1.27}$$

where u and v are two real numbers. Displacement operators are unitary operators and their name comes from their action on the position and momentum bases,

$$U(0,v)\,|x\rangle = |x-v\rangle\,, \qquad U(u,0)\,|p\rangle = |p+u\rangle\,. \tag{1.28}$$

Their composition can be derived from the Baker-Campbell-Haussdorff formula and reads

$$U(u_1,v_1)U(u_2,v_2) = \mathrm{e}^{\frac{i}{2}(v_1u_2-u_1v_2)}U(u_1+u_2,v_1+v_2), \tag{1.29}$$

and hence they follow the commutation relation:

$$U(u_1,v_1)U(u_2,v_2) = \mathrm{e}^{i(v_1u_2-u_1v_2)}U(u_2,v_2)U(u_1,v_1). \tag{1.30}$$

Remarkably the displacement operators form a basis for the operators acting on $\mathscr{H}$.

If we consider n different oscillators at once, so $\mathscr{H}^{\otimes n} = L^2(\mathbb{R})^{\otimes n}$, we can identify each by its canonically commuting position and momentum operators, for $1 \le j \le n$, $\left(\hat{x}_j, \hat{p}_j\right)$, where $\hat{x}_j$ is a shorthand notation for $\mathbb{1}^{\otimes(j-1)} \otimes \hat{x} \otimes \mathbb{1}^{\otimes(n-j)}$ and similarly for $\hat{p}_j$. They are characterized by the commutation relations

$$\left[\hat{x}_j, \hat{p}_k\right] = i\delta_{jk}, \qquad \left[\hat{x}_j, \hat{x}_k\right] = 0, \qquad \left[\hat{p}_j, \hat{p}_k\right] = 0, \tag{1.31}$$

where δ_{jk} denotes the Kronecker delta.

On such a system we can also consider the group formed by tensor products of displacement operators. Similarly to multi-qubit Pauli operators, multi-oscillator displacement operators can be characterized by a phase and a symplectic vector of real numbers. We use the following notation

$$U(\boldsymbol{e}) = \bigotimes_{j=1}^{n} U(u_j,v_j) = \prod_{j=1}^{n} \mathrm{e}^{iu_j\hat{x}_j+iv_j\hat{p}_j}, \quad \boldsymbol{e} \equiv (\boldsymbol{u},\boldsymbol{v}) \in \mathbb{R}^{2n}, \tag{1.32}$$

where we aggregate in a single vector $\boldsymbol{e}$ all the displacement parameters for every oscillator, with first the $\hat{x}$ part, $\boldsymbol{u}$, and then the $\hat{p}$ part, $\boldsymbol{v}$. The composition rule extends to these operators as follows

$$U(\boldsymbol{e})U(\boldsymbol{e}') = \mathrm{e}^{\frac{i}{2}\boldsymbol{e}'\Omega\boldsymbol{e}^{\mathrm{T}}}U(\boldsymbol{e}+\boldsymbol{e}'), \tag{1.33}$$

where Ω is the symplectic form, see Eq. (1.17), and the commutation relation becomes

$$U(\boldsymbol{e})U(\boldsymbol{e}') = \mathrm{e}^{i\boldsymbol{e}'\Omega\boldsymbol{e}^{\mathrm{T}}}U(\boldsymbol{e}')U(\boldsymbol{e}). \tag{1.34}$$

These operators together with arbitrary phases form an irreducible representation of the Heisenberg group.

Similarly to changing variables, it is also possible to redefine position and momentum operators by taking linear combinations of the original ones. It is imperative that the new operators still fulfill the correct commutation relation given in (1.31). If we denote as $\hat{\boldsymbol{r}} = (\hat{\boldsymbol{x}}, \hat{\boldsymbol{p}})$ the vector of all position operators followed by all momentum operators, we can write down the transformed vector $\hat{\boldsymbol{r}}'$ using a $2n \times 2n$ real matrix A,

$$\hat{\boldsymbol{r}}' = A\hat{\boldsymbol{r}}. \tag{1.35}$$

Enforcing the canonical commutation relation for $\hat{\boldsymbol{r}}'$ can be shown to be equivalent to demanding that the matrix A is symplectic, i.e.

$$A\Omega A^{\mathrm{T}} = \Omega. \tag{1.36}$$

with Ω the symplectic form given in Eq. (1.17). Some unitary transformations, called Gaussian operations, physically realize this kind of transformations.

1.1.3. QUANTUM COMPUTATION

Questions about the nature and limits of computation have emerged in the 20th century. Since then they have had a large influence on many topics in science. The first questions asked were about what problems could be solved by a mechanized process (or really any physical process), what functions could be computed. This question of course depends on what processes are allowed. Several models of computation have been devised, trying to capture the full extent of what physical processes can do. The most famous is called the Turing machine [8] (several others turned out to be equivalent such as Church's λ-calculus), and this model is still the most complete definition we have for computation. This is often referred to as the Church-Turing thesis. It states that every physically realizable computation can be simulated on a Turing machine. Note that this notion, called computability, does not include any consideration for the size of the system doing the computation nor the time required for the computation to finish.

That is why some time later, the notion of complexity was introduced and is still widely studied today. For an in depth introduction to complexity theory we refer the reader for example to [9, 10]. Roughly this notion refines the question to what problems can be solved on a machine given a certain amount of resources in space (size of the machine) and time (for the execution). It turns out that the most fruitful way to talk about the complexity of some problem is in terms of the scaling of the procedure which computes the solution. More precisely, given a problem with instances of growing sizes, how does the space or the time required by the computation to solve the most difficult instance of a given size grow with this size. This is called the worst-case complexity. Notably, it has been observed that the class of problems which have a worst-case polynomial scaling between the size of the input and the time of the computation constitute a good definition for feasible computation. With this refined notion of feasibility the model of computation becomes more important: in order to be equivalent two different models need not only to be able to simulate one another but the simulation needs to be efficient as well. An extended version of the Church-Turing thesis was recently formulated asserting that every physically realizable computation can be simulated efficiently on a Turing machine.

It was also observed that no efficient simulation of quantum phenomena on a Turing machine seems to exist [11]: all the known simulation techniques need exponential time. This observation motivated the exploration of the computational capabilities of quantum processes since they seem to be at odds with the extended Church-Turing thesis. The first definitions of quantum Turing machines appeared in [12, 13] followed by a circuit model version in [14] which is the most commonly used now. One of the most striking pieces of evidence that quantum computing could be more powerful than classical computing was discovered by Shor [15]: a polynomial time quantum algorithm to find the prime factorization of a number. This problem is not believed to have an efficient classical algorithm although no proof of this has been established.

The circuit model of quantum computation works as follows. Given a problem to solve, for each input size there is a classical procedure constructing a quantum circuit acting on a number of qubits depending on the size of the input. A quantum circuit is a sequence of gates, i.e. unitary operations acting on a subset of the qubits. The circuit is run on the input encoded in the corresponding computational basis state and any auxiliary qubit initialized in zero. Then the output quantum state is measured in the computational basis and the outcome is interpreted as the solution. If the size of the quantum circuit is polynomial in the size of the input and the procedure constructing the circuit also has a polynomial complexity then it is considered to be an efficient quantum computation. It is also possible to include measurements in the body of the circuit, not only at the end, and even adaptively using the measurement outcomes without changing the complexity class defined. There even is an equivalent model of computation using quantum states independent of any problem but performing adaptive measurements depending on the problem to solve. This is called measurement based quantum computation [16]. In this thesis we take the point of view of the circuit model.

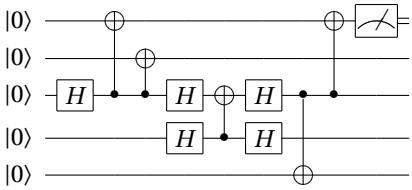


Figure 1.1: An example of quantum circuit: There are five qubits initialized in the $|0\rangle$ state. Then a sequence of Hadamard gates (H) on individual qubits and controlled-not gates (CNOT) on pairs of qubits where the $\bullet$ symbol represent the control qubit and the $\oplus$ symbol the target. Finally the top qubit is measured in the computational basis outputting a single classical bit. See Chapter 2 for its use.

Quantum circuits have a convenient graphical representation using wires and boxes, see Figure 1.1. Each wire represent a qubit and boxes represent gates, i.e. unitary operations, to apply to the qubits they intersect. Measurements have a specific representation, and usually represent measurements in the computational basis unless specified otherwise.

So far we have not specified what gates are allowed in a circuit. In order for the measure of complexity in terms of the size of the circuit to make sense, one requires that each

gate take a constant time to execute. In particular it would not be wise to allow to bundle up large chunks of the circuit and label them as gates since that could hide the complexity of the circuit. To solve this problem we can define a gate set over which to build circuits. The first constraint for the gates is to act on a constant number of qubits, typically one, two or three. Then we would like to still be able to construct any unitary from the gate set which is captured by the notion of universality. A gate set is called universal if one can construct any unitary by using only gates from the set, and approximately universal if one can approximate any unitary using gates from the set arbitrarily well. One important result concerning gate sets, named the Solovay-Kitaev theorem after the name of its two independent discoverers, states that any approximately universal gate set can efficiently approximate any other one. Here efficiently means with an overhead growing poly-logarithmicaly in the inverse desired precision [17]. This means that we should only be concerned that our gate set is approximately universal, since they are all efficiently convertible to one another.

Several universal gate sets have been devised. One of the most used is called Cliffords+T; it is composed of the Hadamard gate, H, the controlled-not gate, CNOT, and the T gate: $\{H, \text{CNOT}, T\}$ [18]. The Hadamard gate is a single-qubit gate which converts between the Z and X bases (see Eq. (1.14)). The CNOT gate is a two-qubit gate which acts in the computational basis as a bit flip on the second qubit (the target) if the first one (the control) is in the $|1\rangle$ state and identity otherwise. The T gate is a single-qubit gate which applies a phase, $e^{i\pi/4}$ to the $|1\rangle$ state. Expressed as matrices they are

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \qquad T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}, \qquad \text{CNOT} = \begin{array}{c} \\ |00\rangle \\ |01\rangle \\ |10\rangle \\ |11\rangle \end{array} \begin{array}{c} \begin{array}{cccc} \langle 00| & \langle 01| & \langle 10| & \langle 11| \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{array}. \tag{1.37}$$

An important subset of the unitary operations generated by these gates are called Clifford operations; they are generated by H, CNOT and $S = T^2$. Clifford operations form a group, which is finite for any fixed number of qubits. A lot of the iconic quantum processes such as teleportation, or quantum key distribution as well as the majority of quantum error correction processes can be realized using only Clifford operations. However, Clifford operations alone are not universal and, even more strikingly, they can be efficiently classically simulated (Gottesman-Knill theorem) [19]. This fact is due to the relation of Clifford operations to Pauli operations which provides another definition for the Clifford operations:

$$\mathscr{C}_n = \left\{ U \in \mathrm{U}\left(2^n\right) \mid \forall P \in \mathscr{P}_n, U^\dagger P U \in \mathscr{P}_n \right\}, \tag{1.38}$$

where $\mathscr{P}_n$ are the Pauli operators as defined in Eq. (1.16). In words, Clifford operations map Pauli operators to Pauli operators. Note that any operator is completely characterized up to a global phase by how it maps the Pauli operators since they form a basis. The Clifford group is also referred to as the second level of the so-called Clifford Hierarchy, $\mathscr{C}_n^{(2)} = \mathscr{C}_n$ and Pauli operators as the first level, $\mathscr{C}_n^{(1)} = \mathscr{P}_n$. The k^{th} level is defined as

$$\mathscr{C}_n^{(k)} = \left\{ U \in \mathrm{U}\left(2^n\right) \mid \forall P \in \mathscr{P}_n, U^\dagger P U \in \mathscr{C}_n^{(k-1)} \right\}. \tag{1.39}$$

Since $\mathscr{P}_n \subset \mathscr{C}_n^{(2)}$, for any $j \leq k$ we have $\mathscr{C}^{(j)} \subset \mathscr{C}_n^{(k)}$. One can check that the T gate is in the third level, $T \in \mathscr{C}_n^{(3)}$, and grants universality to the gate set Cliffords+T. As an alternative to the T gate, the CCZ gate, also from the third level, can be added to the Clifford group to yield a universal gate set. The CCZ gate is a three qubit gate diagonal in the computational basis which flips the phase of the $|111\rangle$ state,

$$\mathrm{CC}Z|b_1 b_2 b_3\rangle = \begin{cases} |b_1 b_2 b_3\rangle & b_1 = 0 \vee b_2 = 0 \vee b_3 = 0 \\ -|b_1 b_2 b_3\rangle & b_1 = b_2 = b_3 = 1 \end{cases}. \tag{1.40}$$

The third level and higher levels do not form groups and their exact structure is not fully understood.

1.1.4. FRAGILITY OF QUANTUM INFORMATION

Several physical systems have been proposed to serve as qubits and are explored in laboratories around the world. Some of the notable candidates, in no particular order, include:

- Trapped ions, where some ions are trapped in a (dynamic) electromagnetic potential and different energy levels of their outer electrons are used as qubit subspace. Manipulation of the qubits is done by shining lasers off or on resonance on the ions.
- Superconducting circuits, where a non-linear element such as a superconducting Josephson junction is added to an LC circuit creating an an-harmonic oscillator permitting to isolate the two lowest energy levels to serve as a qubit. Many different circuits and realizations exists. Manipulations are done by coupling the system to microwave resonators controlled with waveform generators.
- Electron spins, where a 2D electron gas is created using semi-conductor techniques and individual electrons are singled out using electric fields, permitting to use the spin degree of freedom of single electrons or pairs of electrons as qubits. Manipulations are done using the electric and magnetic field.
- Optical photons, where Fock states of optical photonic modes are used to encode a qubit. Manipulations are done using single-photon sources, linear optical transformations and photon counting detectors.
- Bosonic codes, where a large portion of the Hilbert space of a bosonic mode is used to encode a qubit using a bosonic error correcting code. 3D cavities in the microwave regime and the motion degrees of freedom of trapped ions are examples of platforms where this can be realized, e.g. [20–22].
- Majorana particles, where qubits are encoded in the edge modes of a symmetry-protected topologically ordered system.

For all these candidates and in any realistic situation it is impossible to have perfect control over a system as well as complete elimination of undesired interactions. Many different sources of noise exist and modeling them precisely is generally not an easy task

but some approximations can be made. It is often the case for instance that the two states representing the $|0\rangle$ and $|1\rangle$ of a physical qubit have different energies in which case the natural evolution of the system is such that the $|1\rangle$ state has an oscillating phase relative to the $|0\rangle$ state whose period is given by the energy difference. This can be accounted for when knowing the energy difference, but any imprecision or fluctuation in the energy difference will accumulate and randomize the relative phase between the two computational basis states of the qubit. This is called dephasing noise, and a simplified version of this noise can be modeled as follows: with a given probability, $p \in [0,1]$, the phase between $|0\rangle$ and $|1\rangle$ is flipped, i.e. Z is applied to the qubit, and with probability $(1-p)$ nothing happens, acting on the state ρ using the Kraus representation it reads

$$\text{Dephasing}_p(\rho) = (1-p)\rho + pZ\rho Z. \tag{1.41}$$

This is called an incoherent dephasing process, meaning that it is a classical stochastic process deciding to apply the phase error or not. Another type of error could be a symmetric exchange of energy with the environment so that the $|0\rangle$ and $|1\rangle$ states are interchanged. This is called a bit flip and if there is a probability p that it happens then the error channel can be written as

$$\text{Bitflip}_p(\rho) = (1-p)\rho + pX\rho X. \tag{1.42}$$

Another process involving exchange of energy with the environment is called amplitude damping. It models a more common situation where only loss of energy from the qubit to the environment occurs. A simple Kraus representation is

$$\text{Damping}_\gamma(\rho) = E_0\rho E_0^\dagger + E_1\rho E_1^\dagger, \qquad E_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{pmatrix}, E_1 = \begin{pmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{pmatrix}. \tag{1.43}$$

The operator E_1 sends $|1\rangle$ to $|0\rangle$ which corresponds to losing energy to the environment. One can think of a small real number γ as the damping rate. Observe that E_0 is close but not proportional to the identity as it is impossible to have a Kraus representation with an operator proportional to the identity for this channel. This can be understood as the fact that in the event that no energy leaks from the qubit, then it is more likely that the qubit was in the state $|0\rangle$ in the first place. This is also sometimes referred to as a back-action error process. Another interesting channel is called the depolarizing channel, when there is an equal probability for a X, Y or Z error to happen

$$\text{Depolarizing}_p(\rho) = \left(1 - \frac{3p}{4}\right)\rho + \frac{p}{4}\left(X\rho X + Y\rho Y + Z\rho Z\right). \tag{1.44}$$

Interestingly, another characterization of this channel is that with probability p the states becomes completely mixed, i.e. $\mathbb{1}/2$, and all information is lost or it stays intact with probability $(1-p)$. In general any channel which involves randomly applying Pauli operators is called a Pauli channel. It takes the form

$$\text{Pauli}_{p_X,p_Y,p_Z}(\rho) = \left(1 - p_X - p_Y - p_Z\right)\rho + p_X X\rho X + p_Y Y\rho Y + p_Z Z\rho Z. \tag{1.45}$$

Pauli channels are not a particularly realistic models of naturally occurring errors although they already give a good idea of what can happen and can be handled analytically quite well as we will see later. Moreover, any qubit channel can be converted to a

Pauli channel by a process called twirling which consists in randomly choosing a Pauli (or the identity) and applying it before and after the noise [23].

The presence of noise in realistic systems implies that we cannot expect to be able to realize large quantum computations flawlessly with them which is what is expected from the definition of quantum computation that we have presented. In the presence of weak noise it is still unclear if an efficient classical simulation of a quantum system exists. It is also unclear what useful problems can be solved using noisy quantum systems. Indeed, ongoing research tries to understand what is possible to realize using noisy quantum systems [24].

1.2. QUANTUM ERROR CORRECTION

1.2.1. PRINCIPLE OF QUANTUM ERROR CORRECTION

The principle of quantum error correction was first proposed in [25, 26]. The idea is to use a large number of physical quantum degrees of freedom (e.g. qubits) and restrict their possible states to a specific subspace of the original Hilbert space. Therefore encoding a smaller number of logical quantum degrees of freedom. The choice of subspace is such that a large set of possible errors (ideally the most probable ones) occurring on the physical degrees of freedom can be detected and undone so that they do not affect the logical degrees of freedom. More precisely, given the error channel we would like to protect against, there should exist a recovery operation such that if one applies the error channel to a state in the code space followed by the recovery then the output state is the input state. One can derive the necessary and sufficient conditions on the subspace and the set of errors to be corrected, they are called the quantum error correction conditions and were first formulated in [27, 28]. We denote the Hilbert space of the physical system as $\mathscr{H}$ and the code subspace as $C \subset \mathscr{H}$. We denote an orthogonal basis for the code space as $\left\{|\overline{\psi}_j\rangle\right\} \in C$. We also denote the set of error operators that we wish to be able to correct as $\{E_j\}$. The quantum error correction conditions can be stated as

$$\forall i,j,k,l, \quad \langle\overline{\psi}_i| E_j^\dagger E_k |\overline{\psi}_\ell\rangle = \alpha_{jk}\delta_{i\ell}, \tag{1.46}$$

where α_{jk} are complex numbers forming a Hermitian matrix. The proof can be found for example in [4]. The basic idea is this: First, different errors should send the code space to different orthogonal error spaces (or exactly the same error space in which case errors are called degenerate). This is so that errors can be detected by measuring in which error space the system is in. The error spaces can be identified by diagonalizing the Hermitian matrix α_{jk}. Second, errors should not act differently on different code states in the code space, since detecting errors would then leak information or modify the logical state of the system.

One very important feature of these conditions is that if they hold for a set of error operators $\{E_j\}$ then they also hold for any set $\{F_j\}$ where the F_j are linear combinations of the E_j, see also [4]. This remarkably means that if a qubit code is capable of correcting against any t-qubit Pauli error, then it is in fact capable of correcting any t-qubit channel. So one has just to design codes to work against Pauli errors and is guaranteed that they will also work against arbitrary noise. This is referred to as the discretization of errors and it is a crucial feature of quantum error correction.

Most of the time we will consider the situation where the physical space consists of n qubits, so $\mathscr{H} = \mathbb{C}^{2^n}$, and a code space, C. Suppose the code C is designed to correct any t-qubit Pauli errors. Consider as a simple example that there is a qubit noise channel acting independently on each qubit. As a first case we can consider that one of the Kraus operators of this channel is proportional to the identity and happens with probability $(1-p)$ for small p, and the others are errors. Applying this channel independently to the n qubits starting in the code space, and then the recovery operation of C, one can check that all terms where at most t qubits have a non-identity Kraus operator will be corrected. Therefore, the first term which will not be back on the original code state will have a prefactor of p^{t+1} and a combinatorial factor c in front. For this channel the unencoded logical error probability is p, while the encoded one is cp^{t+1}, which is smaller than p for sufficiently small p. There is a subtlety which requires a second consideration, if the channel does not have any representation where one of the Kraus operators is the identity. This is the case for example for the amplitude damping channel presented above. In this case one of the Kraus operators is close to the identity but with some perturbation, let us denote it as $\tilde{\mathbb{1}} = \mathbb{1} + A$, where A is some operator with small norm, say $\sqrt{p}$. When applying the channel independently on every qubit, we can separate the terms in the sum according to how many qubits get acted on by the $\tilde{\mathbb{1}}$ operator. Terms where less than $t+1$ qubits have the $\tilde{\mathbb{1}}$ will have a prefactor of p^{t+1}, if p is the square norm of the other operators than $\tilde{\mathbb{1}}$ in the qubit channel. We are now examining the other terms where $j \leq t$ qubits are acted on by an error, say B, they look like

$$\tilde{\mathbb{1}}^{\otimes(n-j)} \otimes B^{\otimes j} \cdot \rho \cdot \left(\tilde{\mathbb{1}}^{\otimes(n-j)} \otimes B^{\otimes j}\right)^{\dagger}. \tag{1.47}$$

By expanding $\tilde{\mathbb{1}}$ into $\mathbb{1} + A$ we get some terms looking like

$$\mathbb{1}^{\otimes(n-j-k)} \otimes A^{\otimes k} \otimes B^{\otimes j} \cdot \rho \cdot \left(\mathbb{1}^{\otimes(n-j-k)} \otimes A^{\otimes k} \otimes B^{\otimes j}\right)^{\dagger}, \tag{1.48}$$

which get corrected by the recovery map if $j + k \leq t$ or have a prefactor of p^{t+1}. We also get some terms looking like

$$\mathbb{1}^{\otimes(n-j-k_1)} \otimes A^{\otimes k_1} \otimes B^{\otimes j} \cdot \rho \cdot \left(\mathbb{1}^{\otimes(n-j-k_2)} \otimes A^{\otimes k_2} \otimes B^{\otimes j}\right)^{\dagger}, \tag{1.49}$$

with $k_1 \neq k_2$, which are not Kraus-like terms and cannot be directly interpreted like the others. Fortunately by choosing a basis for the errors A and B which send the code space to orthogonal error spaces, one can see that the measurements done during the recovery can make these terms vanish. So in the end we are in the same situation as earlier with the non-corrected terms all having a prefactor cp^{t+1}.

1.2.2. STABILIZER FORMALISM

Now that we have presented the principle of quantum error correction we present how one can construct quantum error correcting codes. We first start by a short presentation of classical linear codes, see [29] for an in depth presentation.

Classical linear codes are designed to protect information which can be represented as classical bits. The basic principle is the same as for quantum error correction. One

has access to n bits, which can be seen as the vector space $\mathbb{F}_2^n$ and one chooses a subspace $C \subset \mathbb{F}_2^n$ of dimension k as a code space encoding k bits. Elements in C are referred to as codewords. Classical bits can only suffer from one type of error, bit flips, so the performance of the code can be assessed by looking at what is the minimal number of bits to flip to go from one code word to another. This is called the distance of the code, denoted as d, and such a code is described as an $[n,k,d]$ code, encoding k bits using n bits with distance d. There are two ways of specifying a classical linear code, either by its generating matrix G or its parity check matrix H. The generating matrix G is a $k \times n$ matrix over $\mathbb{F}_2$ whose rows generate the code space C. The parity check matrix H is an $(n-k) \times n$ matrix over $\mathbb{F}_2$ specifying all the parity checks that the codewords have to obey. A row vector $\boldsymbol{c}$ is a codeword if and only if it passes all the checks:

$$\boldsymbol{c} \in C \Leftrightarrow H\boldsymbol{c}^{\mathrm{T}} = \mathbf{0}^{\mathrm{T}}, \tag{1.50}$$

where the inner product between a row of H and column c^{T} is with respect to $\mathbb{F}_2$ arithmetic. The relation between G and H is given by

$$GH^{\mathrm{T}} = 0. \tag{1.51}$$

Another way of looking at the parity check matrix H is that it is the generating matrix of the code dual to C, denoted as $C^{\perp}$, i.e. the subspace of all vectors orthogonal to C. Thinking of a code in terms of its parity check matrix is useful as, besides defining the code, it also gives a way of performing error correction. Consider a binary vector $\boldsymbol{v}$ which is a codeword $\boldsymbol{c}$ with some added noise $\boldsymbol{\epsilon}$, so that $\boldsymbol{v} = \boldsymbol{c} \oplus \boldsymbol{\epsilon}$ ($\oplus$ is the addition modulo two). Given $\boldsymbol{v}$, one can compute the value of the parity checks, also called the syndrome, $\boldsymbol{s}$, using H,

$$\boldsymbol{s}^{\mathrm{T}} = H\boldsymbol{v}^{\mathrm{T}} = H\boldsymbol{\epsilon}^{\mathrm{T}}, \tag{1.52}$$

where we used the property that the codewords pass every parity check. So we can see that the syndrome directly gives information about the error. One can use the syndrome to find the smallest possible error with the same syndrome as the most probable error. Here the smallest possible error is the vector $\boldsymbol{\epsilon}$ with the smallest number of 1s.

This approach to error correction can be extended to quantum codes. It has been developed in a slightly restricted way in [30, 31] and generalized to the full stabilizer formalism by Gottesman [32]. The idea is to use Pauli operators as parity checks. They constrain the code states as follows: if the Pauli operator is measured the outcome should be the +1 eigenvalue. Equivalently, if $|\psi\rangle$ is in the code space and P is a Pauli operator chosen as a parity check then

$$P|\psi\rangle = |\psi\rangle. \tag{1.53}$$

Said yet differently, P is a stabilizer of the code space. The way to define a stabilizer code is therefore to choose an Abelian subgroup of the Pauli operators $\mathcal{S} \subset \mathcal{P}_n$, and define the code as the common +1 eigenspace of the elements of the group $\mathcal{S}$:

$$C = \left\{|\psi\rangle \in \mathcal{H} \mid \forall S \in \mathcal{S},\, S|\psi\rangle = |\psi\rangle\right\}. \tag{1.54}$$

Note that the set of stabilizers is necessarily a group and Abelian since every Pauli operator in it has to commute with every other, otherwise they could not have a common

eigenspace. Moreover, the operator $-\mathbb{1}$ cannot be included in $\mathcal{S}$ otherwise there would not be any common $+1$ eigenspace. The dimension of the code space C can be simply calculated by counting the number of independent stabilizers in $\mathcal{S}$, where independent here means not generated by taking products of the others. If we consider n physical qubits and $n-k$ independent stabilizers then the code space is that of k logical qubits $\left(\dim(C) = 2^k\right)$.

Stabilizer code are particularly suited to Pauli error channels. Consider a code state $|\psi\rangle \in C$, and a Pauli operator $E \in \mathcal{P}_n$ acting as an error on $|\psi\rangle$, then consider measuring a stabilizer element $S \in \mathcal{S}$. Pauli operators can either commute or anti-commute, see Eq. (1.17), so if they anti-commute then

$$S\left(E|\psi\rangle\right) = -ES|\psi\rangle = -E|\psi\rangle\,, \tag{1.55}$$

and the measurement outcome of S will be -1 but if S and E commute measuring the stabilizer S will not detect the error and the outcome will be $+1$.

If the error E commutes with every stabilizer in $\mathcal{S}$, it can mean two things. Either E is an element of $\mathcal{S}$, in which case no harm is done to the code state. Or it is not in $\mathcal{S}$ and then necessarily E maps the code state to another code state. This is therefore called a logical operation, since it acts within the code space, or in this case is an undetected logical error. The group of all Pauli operators commuting with $\mathcal{S}$ is called the normalizer of $\mathcal{S}$, it includes all logical Pauli operations and is denoted as $\mathcal{N}(\mathcal{S})$:

$$\mathcal{N}(\mathcal{S}) = \{P \in \mathcal{P}_n \mid P\mathcal{S} = \mathcal{S}P\}\,. \tag{1.56}$$

We have $\mathcal{S} \subset \mathcal{N}(\mathcal{S})$ and elements in $\mathcal{S}$ act trivially on the code space. Moreover two elements in $\mathcal{N}(\mathcal{S})$ have identical effects if they are related by multiplication with a stabilizer. This means that to get the group of independent logical Pauli operators, $\mathcal{L}$, one should build the quotient of $\mathcal{N}(\mathcal{S})$ by $\mathcal{S}$,

$$\mathcal{L} = \mathcal{N}(\mathcal{S})/\mathcal{S}. \tag{1.57}$$

This quotient group is not to be confused with $\mathcal{N}(\mathcal{S}) \setminus \mathcal{S}$, which is just the set of Pauli operators preserving the code space and acting non-trivially on it. Finding the element in this latter set, $\mathcal{N}(\mathcal{S}) \setminus \mathcal{S}$, which acts on the smallest number of qubits, gives an equivalent notion of code distance in the case of stabilizer codes

$$d = \min_{L \in \mathcal{N}(\mathcal{S}) \setminus \mathcal{S}} |L|, \tag{1.58}$$

where $|L|$ means the number of qubits that L acts on non-trivially. If d is the distance, then we say we have an $[[n,k,d]]$ stabilizer code. The double bracket is there to distinguish this from the classical code notation with single brackets. Given an $[[n,k,d]]$ stabilizer code, one straightforwardly verifies the quantum error correction conditions, Eq. (1.46), on the set of all Pauli errors acting on at most $\left\lfloor \frac{d-1}{2} \right\rfloor$ qubits. As we have seen in the previous section, the distance then gives an idea of the error correction performance of the code since it roughly allows to reduce the noise from p to $p^{\left\lfloor \frac{d}{2} \right\rfloor}$. Besides being able to correct every error affecting $\left\lfloor \frac{d-1}{2} \right\rfloor$ qubits, a code with distance d is also able to

detect every error affecting $d-1$ qubits. This means the syndrome will always indicate when such errors happen even if trying to correct is not guaranteed to succeed. So by using post-selection one can boost the noise reduction to roughly p^d. The ratio of the parameters k and n is called the encoding rate, $r = k/n$, and is the first measure of the overhead of the code.

Using the binary symplectic vector representation of Pauli operators, see Eq. (1.15), we can rewrite in a matrix form the elements presented so far. We define H to be a $(n-k)\times 2n$ binary matrix whose rows represent the generators of the stabilizer group $\mathcal{S}$. We can consider the rows of H as the first elements of a symplectic basis for the vector space $\mathbb{F}_2^{2n}$, and complete the basis for the full space. The rest of this basis will contain first $(n-k)$ symplectically conjugated vectors to H, we denote them as a $(n-k)\times 2n$ binary matrix D. Secondly it will contain $2k$ pairwise conjugated vectors which we will split in two $k\times 2n$ binary matrices P and Q. The elements in D are sometimes referred to as pure errors or destabilizers as they give Pauli errors which anti-commute with exactly one stabilizer. The elements in P and Q are the logical Pauli operators forming the correct algebra required of the Pauli operators representing k qubits.

If a Pauli error represented by the binary vector $\boldsymbol{e}$ affects the system, the syndrome $\boldsymbol{s}$ that is going to be observed by measuring the stabilizer generators is given by its symplectic inner product with the generators:

$$\boldsymbol{s}^T = H\Omega\boldsymbol{e}^{\mathrm{T}}. \tag{1.59}$$

Given the observed syndrome $\boldsymbol{s}$, one can deduce that the error which affected the system must be such that

$$\exists(\boldsymbol{p},\boldsymbol{q})\in\mathbb{F}_2^{2k}, \exists\boldsymbol{r}\in\mathbb{F}_2^{n-k}, \quad \boldsymbol{e} = \boldsymbol{s}D + \boldsymbol{r}H + \boldsymbol{p}P + \boldsymbol{q}Q. \tag{1.60}$$

A way to think about these quantities is the following: The vector $\boldsymbol{s}D$ represents an error with the correct syndrome, it is completely determined by the choice of D we made for our basis. It is often called the candidate error. The other parts of the error do not change the syndrome as they represent either stabilizers or logical operators. The row vector $\boldsymbol{p}P + \boldsymbol{q}Q$ represents the logical operation done on top of the candidate error. This is the part that needs to be found in order to undo any logical error done to the system. The row vector $\boldsymbol{r}H$ represents the freedom of adding a stabilizer operator to the error. It does not change the logical action but it does change the probability of the error. The task of syndrome decoding then, is to find the most probable $\boldsymbol{p}$ and $\boldsymbol{q}$. The existence of an efficient and accurate decoding algorithm is also crucial for an error correcting code since the running time and the accuracy of the decoding will both affect the overall performance of error correction.

1.2.3. NOTABLE EXAMPLES

In this section we introduce some notable examples and families of stabilizer codes.

CSS CODES

CSS (Calderbank-Steane Shor) codes were the first type of quantum error correcting codes discovered and form a subset of the stabilizer code family. They are stabilizer

codes where the stabilizer generators can be split in two sets, one with only X-type stabilizers (Pauli operators with only X elements) and the other with only Z-type stabilizers (Pauli operators with only Z elements). These codes can be described by two classical codes, $C_X \subset \mathbb{F}_2^n$ and $C_Z \subset \mathbb{F}_2^n$, which generates the vectors corresponding to the X and Z stabilizers. In order for the stabilizers to commute with one another, each code needs to be in the dual of the other which can be expressed using their generating matrices H_X and H_Z

$$H_X H_Z^{\mathrm{T}} = 0. \tag{1.61}$$

Note that we use the letter H for the generating matrices of the classical codes here since they actually correspond to checks of the quantum code.

The number of encoded logical qubits is given by

$$k = n - \dim(C_X) - \dim(C_Z). \tag{1.62}$$

The logical operators are then simply given by the dual codes to C_X and C_Z. More precisely $C_X^{\perp}$ gives the Z-type logical operators and $C_Z^{\perp}$ the X-type logical operators. Similarily as before to obtain the independent logical operators we quotient out the X-stabilizers and Z-stabilizers respectively,

$$\mathscr{L}_X = C_Z^{\perp}/C_X, \qquad \mathscr{L}_Z = C_X^{\perp}/C_Z. \tag{1.63}$$

The notion of distance can be split into the distance of the X-type logical operators, d_X, and for the Z-type logical operators, d_Z,

$$d_X = \min_{\boldsymbol{x} \in C_Z^{\perp} \setminus C_X} |\boldsymbol{x}|, \qquad d_Z = \min_{\boldsymbol{z} \in C_X^{\perp} \setminus C_Z} |\boldsymbol{z}|. \tag{1.64}$$

Then the overall distance of the code is given by

$$d = \min(d_X, d_Z). \tag{1.65}$$

Even if CSS codes are a slightly restricted family of stabilizer codes, this restriction does not seem to be so fundamental, for example it is possible to map any stabilizer code to a CSS code with the same parameters up to constant factors [33]. A lot of work has already been done on classical error correcting codes, and a lot of constructions to find codes with good parameters exist. The main difficulty in transferring these results to the quantum setting resides in the constraint in Eq. (1.61).

THE TORIC CODE AND HOMOLOGICAL CODES

The toric code is probably the most famous quantum error correcting code. It was proposed by Kitaev in [34, 35]. It is a CSS code which is constructed by using the properties of tilings of surfaces and their homology. The idea is to start from a surface which is tiled, see for example the square tiling on a torus represented in Fig. 1.2. Then one places a qubit on each edge of the tiling and defines an X-stabilizer for each vertex acting on the four adjacent edges and Z-stabilizer for each face acting on the four adjacent edges, see Fig. 1.2.

The fact that this defines a valid CSS code can be thought of as coming from a well known property of the boundary map of a tiling, namely “the boundary of the boundary

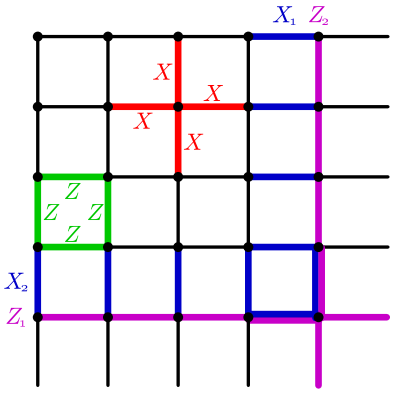


Figure 1.2: Representation of a toric code. The tiling has periodic boundary condition and is therefore a torus. The qubits are placed on the edges, the X-stabilizers are defined by the stars (red) and the Z-stabilizers by the plaquettes (green). There are two logical qubits, whose operators are represented by non-contractible loops around the torus represented in blue for X and purple for Z.

is empty". More precisely, given the set of faces F, edges E, and vertices V, one can define the corresponding $\mathbb{F}_2$ vector spaces $\mathscr{F} = \mathbb{F}_2^F$, $\mathscr{E} = \mathbb{F}_2^E$ and $\mathscr{V} = \mathbb{F}_2^V$. Then the adjacency relation between them can be thought of as a linear map, called the boundary map, denoted as $\partial_{\mathscr{F}\to\mathscr{E}}$ and $\partial_{\mathscr{E}\to\mathscr{V}}$,

$$\mathscr{F} \xrightarrow{\partial_{\mathscr{F}\to\mathscr{E}}} \mathscr{E} \xrightarrow{\partial_{\mathscr{E}\to\mathscr{V}}} \mathscr{V}. \tag{1.66}$$

Since $\partial_{\mathscr{E}\to\mathscr{V}} \circ \partial_{\mathscr{F}\to\mathscr{E}} = 0$, if we write these linear maps as matrices (denoted as $\mathrm{Mat}(\cdot)$), we obtain exactly the CSS condition in Eq. (1.61) by choosing

$$H_Z = \mathrm{Mat}(\partial_{\mathscr{F}\to\mathscr{E}})^{\mathrm{T}}, \qquad H_X = \mathrm{Mat}(\partial_{\mathscr{E}\to\mathscr{V}}). \tag{1.67}$$

Concerning the logical operators we can also make a connection with what is called the homology of the surface. Recall that the Z-logicals are given by $C_X^{\perp}/C_Z$ and an other way of writing this is $\ker(H_X)/\mathrm{im}\left(H_Z^{\mathrm{T}}\right) = \ker(\partial_{\mathscr{E}\to\mathscr{V}})/\mathrm{im}(\partial_{\mathscr{F}\to\mathscr{E}})$. This exactly corresponds to what is called the homology group of the surface. When worked out, the elements in the homology group are sets of edges forming closed loops which cannot be contracted. On a torus there are two loops around the surface in both directions, see Fig. 1.2. Transposing everything one gets the notion of co-homology, which coincides with the X-logicals so $C_Z^{\perp}/C_X = \ker(H_Z)/\mathrm{im}\left(H_X^{\mathrm{T}}\right) = \ker\left(\partial_{\mathscr{F}\to\mathscr{E}}^{\mathrm{T}}\right)/\mathrm{im}\left(\partial_{\mathscr{E}\to\mathscr{V}}^{\mathrm{T}}\right)$.

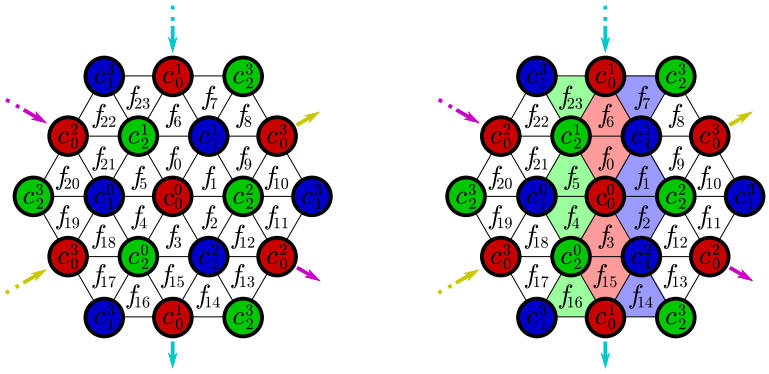
This connection makes it easier to find the parameters of such codes. The number of encoded logical qubits, k, is directly related to the number of non-contractible loops, which is in turn related to the genus (number of handles) and orientability of the surface. For the torus, there are two non-contractible loops, so $k = 2$. The distance is given by the length of the smallest non-contractible loop. For the toric code on a square lattice, this gives the parameters $[[2L^2, 2, L]]$, if we denote as L the side length of our torus.

These features are the reason to call such codes topological codes. This means codes which have geometrically local stabilizers on some manifold and distance growing with the overall size of the manifold.

This procedure works for any tiling of any surface, and it is also possible to cut the surface and introduce boundaries [36, 37]. It also works in higher dimensions, since the property of the boundary map to square to zero still holds. The procedure is then to place qubits on the j-cells, X-stabilizers correspond to $(j-1)$-cells and Z-stabilizers to $(j+1)$-cells.

COLOR CODES

Color codes are another geometrical construction of CSS codes, also part of the category of topological codes, introduced by Bombín in [38, 39], see also [40].



(a) (b)

Figure 1.3: (a) Triangulation of a torus where the vertices are tri-colorable. One can define a color code using this tiling. The qubits are placed on the triangles and every vertex defines both a X-stabilizer and Z-stabilizer on the adjacent triangles. (b) Representation of a red, a green and a blue string wrapping around the torus. They are formed of pairs of triangles sharing an edge linking vertices of a given color.

The prescription to construct a color code in 2D is to take a triangulation of a 2D surface which has the property of being tri-colorable for its vertices. Under such conditions, one can show that putting qubits on the triangles and X-stabilizers and Z-stabilizers on each vertex yields a valid CSS code. Putting a stabilizer on a vertex means that it acts on all the adjacent triangles, for example, in Figure 1.3, the stabilizer associated with the red vertex labeled c_0^0 acts on the six qubits labeled f_0, f_1, f_2, f_3, f_4 and f_5. See Fig. 1.3a for an example of a tri-colorable triangulation of a torus. Pictures of color codes are also often represented in a picture dual to that of Figure 1.3 where each triangle is replaced by a vertex and each colored vertex by a colored face, see Figure 1.4.

The logical operators of such codes can be constructed by forming what is called colored strings. The idea is that putting Pauli operators, say X, on a pair of adjacent triangles anticommutes only with the two Z-stabilizers given by the two opposed vertices which are of the same color. So by forming closed strings of these pairs of triangles joining vertices of a given color, one can form operators commuting with all the stabilizers, see Fig. 1.3b. If the strings are homologically trivial then they turn out to be stabilizers, but logical operators when they are non-trivial. If three strings of each color meet at the three vertices of the same triangle they can be joined together to form what is called a string net. This is represented in Figure 1.4.

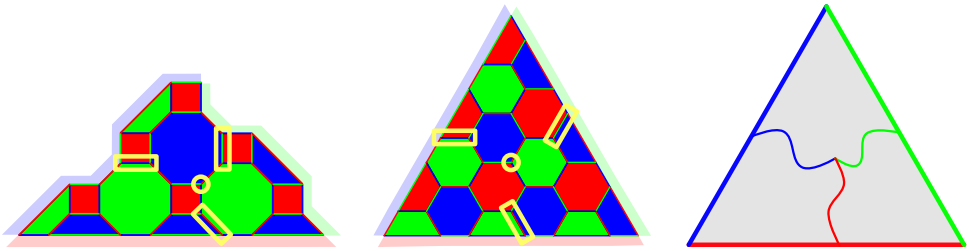


Figure 1.4: Two examples of color codes with boundaries (left and middle) represented in the picture dual to the one in Figure 1.3 and a schematic representation of their logical operators (right). The qubits are placed on the vertices of the lattice and X and Z stabilizers on each colored faces. The color of the boundaries are represented by stripes of color along them. A red boundary is a boundary adjacent to only blue and green faces, and the same by permuting the colors for the other boundaries. The support for a logical Pauli operator is represented for each triangular patch, highlighted by yellow boxes. It consists in three strings of each color attached to the boundary of their color and meeting in the middle of the patch. This is schematically represented on the right. Such logical operators are called string nets.

One can also introduce boundaries to color codes, we represent the concept in Figure 1.4. Different colors for the boundary are possible, if a boundary borders only of green and blue faces, it is called a red boundary. This can be understood by the fact that one could place a red face extending along the red boundary. Blue and green boundaries are defined similarly. Since boundaries are similar to missing stabilizers, one can terminate red strings at red boundaries, blue strings at blue boundaries and green strings at green boundaries. This is illustrated in Figure 1.4. The string nets of triangular patches of color codes, as in Figure 1.4, can also be deformed to be fully supported on any of the three colored boundaries. For example the blue boundary, left in the pictures of Figure 1.4, can be seen as a red string from the bottom red boundary meeting green and blue strings only at the very top vertex.

Color codes can also be constructed in higher dimensions. In dimension D, one should tile a D-manifold with D-simplices (D-dimensional convex hull of $D+1$ vertices) such that the vertices are $(D+1)$-colorable. Then qubits are placed on the D-simplices and X-stabilizers and Z-stabilizers are defined using x-simplices and z-simplices for some integers x and z with $x+z=D-2$. In Chapter 5, we present a generalization of color codes and go over this construction in more detail.

GENERAL MULTI-MODE GKP CODES

As a last example we show here how the stabilizer formalism can be used similarly for codes based on oscillators instead of qubits. We provide a mathematical summary of the formalism of continuous-variable stabilizer codes, as introduced by Gottesman, Kitaev and Preskill (GKP) [41]. We show that these codes contain three main classes, linear oscillator codes, encoding continuous-variable information, proper GKP codes, encoding discrete information and not fully constrained GKP codes, encoding continuous-variable information.

Recall that we write a displacement operator on a n-oscillator Hilbert space as fol-

lows

$$U(\boldsymbol{e}) \equiv \prod_{k=1}^{n} \mathrm{e}^{iu_k\hat{p}_k + iv_k\hat{q}_k}, \quad \boldsymbol{e} \equiv (\boldsymbol{u},\boldsymbol{v}) \in \mathbb{R}^{2n}. \tag{1.68}$$

We also denote by $\hat{O}(\boldsymbol{e})$ the Hermitian operator in the exponent of $U(\boldsymbol{e})$,

$$\hat{O}(\boldsymbol{e}) \equiv \boldsymbol{u}\cdot\hat{\boldsymbol{p}} + \boldsymbol{v}\cdot\hat{\boldsymbol{q}} = \sum_{k=1}^{n} u_k\hat{p}_k + \sum_{k=1}^{n} v_k\hat{q}_k, \qquad U(\boldsymbol{e}) = \exp\left(i\hat{O}(\boldsymbol{e})\right). \tag{1.69}$$

These operators, $U(\boldsymbol{e})$, follow the product rule

$$U(\boldsymbol{e})U(\boldsymbol{e}') = U(\boldsymbol{e}+\boldsymbol{e}')\mathrm{e}^{i\omega(\boldsymbol{e},\boldsymbol{e}')}, \tag{1.70}$$

with the standard symplectic form $\omega(\boldsymbol{e},\boldsymbol{e}') = \boldsymbol{u}\cdot\boldsymbol{v}' - \boldsymbol{v}\cdot\boldsymbol{u}'$.

Take $\mathscr{S} \subset \mathscr{H}_n$ an Abelian subgroup of the displacement operators with no element proportional to the identity with a non-trivial phase, stabilizing some code space, $\mathscr{Q}$. One can simply characterise the structure of such a subgroup. Each element $S \in \mathscr{S}$ can be uniquely written as

$$S = \mathrm{e}^{i\theta}U(\boldsymbol{e}), \tag{1.71}$$

for some real vector $\boldsymbol{e} \in \mathbb{R}^{2n}$, and some phase $\theta \in [0, 2\pi)$. Each vector $\boldsymbol{e}$ can only appear with a unique phase θ in $\mathscr{S}$, otherwise an element proportional to the identity with a non-trivial phase would also be in $\mathscr{S}$. Hence we have an isomorphism between $\mathscr{S}$ and the additive subgroup $G \subset \mathbb{R}^{2n}$, given by all the vectors, $\boldsymbol{e}$, obtained from the decomposition in Eq. (1.71). We can then use the following theorem to characterize the structure of $\mathscr{S}$.

Theorem ([42]). *Let G be a closed additive subgroup of $\mathbb{R}^{2n}$, G can be decomposed in the direct sum of a linear subspace, $G_0 \subset \mathbb{R}^{2n}$, and a discrete lattice, L, generated by some vectors orthogonal to G_0, $\boldsymbol{u}_1,\ldots,\boldsymbol{u}_m \in G_0^{\perp}$:*

$$G = G_0 \oplus L, \quad L = \left\{ \sum_{j=1}^{m} k_j\boldsymbol{u}_j \;\middle|\; k_1,\ldots,k_m \in \mathbb{Z} \right\}.$$

The linear subspace G_0 is the largest linear subspace contained in G.

The condition for G to be closed is without consequences in our case as an open set and its closure would stabilize the same space. If G is open then we can replace it by its closure and appropriately complete $\mathscr{S}$. The limiting case when $G = G_0$ corresponds to the continuous case whereas $G = L$ corresponds to the discrete case. Denote ℓ as the dimension of G_0, one can choose some basis vectors, $G_0 = \langle \boldsymbol{g}_1,\ldots,\boldsymbol{g}_\ell\rangle$. Consider one of the generators of G_0, w.l.o.g. $\boldsymbol{g}_1$, for this generator, given a scalar factor $\lambda \in \mathbb{R}$, there is some angle function $\theta_1(\lambda)$ such that

$$\mathrm{e}^{i\theta_1(\lambda)}U(\lambda\boldsymbol{g}_1) \in \mathscr{S}.$$

It is easy to check that θ_1 obeys Cauchy's functional equation and as such θ_1 is automatically $\mathbb{Q}$-linear: In equations this gives

$$\forall(\lambda,\mu) \in \mathbb{R}^2,\ \theta_1(\lambda+\mu) = \theta_1(\lambda) + \theta_1(\mu) \Rightarrow \forall r \in \mathbb{Q},\ \theta_1(r) = r\theta_1(1) \equiv r\theta_1.$$

Hence, for any code state, $|\overline{\Psi}\rangle \in \mathscr{Q}$, and rational $r \in \mathbb{Q}$, we can write down

$$e^{ir\theta_1} U\left(r\boldsymbol{g}_1\right)|\overline{\Psi}\rangle = \exp\left(ir\left(\theta_1 + \hat{O}(\boldsymbol{g}_1)\right)\right)|\overline{\Psi}\rangle = |\overline{\Psi}\rangle\,. \tag{1.72}$$

The previous equation means that code states are eigenstates of the operator $\hat{O}(\boldsymbol{g}_1)$ with eigenvalue $O(\boldsymbol{g}_1)$ which satisfies

$$\begin{aligned} \forall r \in \mathbb{Q}, \quad & O(\boldsymbol{g}_1) + \theta_1 = 0 \mod 2\pi/r \\ \Leftrightarrow\ & O(\boldsymbol{g}_1) = -\theta_1. \end{aligned}$$

Usually, $\mathscr{S}$ will be chosen such that $\theta_1 = 0$, and $\hat{O}(\boldsymbol{g}_1)$ will be called a *nullifier* as it only takes eigenvalue 0 on the code space. Choosing some non-trivial θ_1 just corresponds to shifting the whole code space by $U(\theta_1 \boldsymbol{d}_1)$, with $\boldsymbol{d}_1$ describing the conjugated pair to $\boldsymbol{g}_1$, i.e. such that $[\hat{O}(\boldsymbol{g}_1), \hat{O}(\boldsymbol{d}_1)] = i$. Similarly, each generator $\boldsymbol{g}_j$ is a nullifier on the code space (up to some possible shift $U(\theta_j \boldsymbol{d}_j)$).

At this point, if $G = G_0$ ($L = \emptyset$), then we have described a linear oscillator code. It is defined by the ℓ nullifiers, $\hat{O}(\boldsymbol{g}_j)$, which each remove a single continuous degree of freedom from the system, leaving $k \equiv n-\ell$ logical oscillator modes. The logical operators can be found by completing the $\boldsymbol{g}_j$ into a full symplectic basis. For the stabilized code space to be non-trivial we therefore require that $\ell < n$.

Consider now that L is non-trivial, then it will constrain the code space as described in Ref. [41, Sec.VI] on the remaining $k = n-\ell$ modes available. Take one of the generators of L, w.l.o.g. $\boldsymbol{u}_1$. The difference with elements of G_0 is that it occurs in $\mathscr{S}$ only with integer multiples. Similarly as previously, given a code state, $|\overline{\Psi}\rangle \in \mathscr{Q}$, and an integer $k \in \mathbb{Z}$ there will be some angle, ϑ_1, such that,

$$e^{ik\vartheta_1} U\left(k\boldsymbol{u}_1\right)|\overline{\Psi}\rangle = \exp\left(ik\left(\vartheta_1 + \hat{O}(\boldsymbol{u}_1)\right)\right)|\overline{\Psi}\rangle = |\overline{\Psi}\rangle\,. \tag{1.73}$$

This means that on the code states, the operator $\hat{O}(\boldsymbol{u}_1)$ can take now several values, given by

$$\begin{aligned} \forall k \in \mathbb{Z}, \quad & O(\boldsymbol{u}_1) + \vartheta_1 = 0 \mod 2\pi/k \\ \Leftrightarrow\ & O(\boldsymbol{u}_1) = -\vartheta_1 \mod 2\pi. \end{aligned}$$

The effect is to discretize this mode. As explained in Ref. [41], one then needs $m = 2k$ lattice generators to fully discretize the remaining k modes.

Using these discrete stabilizers it is also possible to completely freeze some of the modes in a so-called comb state. Hence, one can freeze like this $n-k$ of the modes leaving room for k continuous-variable degrees of freedom.

Summing up, the case where $G = G_0$ and $\ell < n$ ($m = 0$) corresponds to linear oscillator codes defined by ℓ nullifiers, encoding $k = n-\ell$ oscillators. On the other hand, the case $G = L$ and $m = 2n$ ($\ell = 0$) correspond to proper GKP codes described in [41]. Finally there is the case where $G = L$ with $m < 2n$ and all the modes acted on by L are completely frozen. The first two cases are investigated in Chapter 3, the third case was investigated in [43].

1.3. FAULT-TOLERANCE AND UNIVERSALITY

1.3.1. FAULT-TOLERANCE

The principle of fault-tolerance is to design specific protocols to realize the detection and correction of errors according to a quantum error correcting code in such a way that they can work even when their own components are noisy. The notion of fault-tolerance is crucial since encoding quantum information in a quantum error correcting code and frequently performing error correction increases the size of the total system and the number of operations realized on it. When considering every element to be noisy, this increase in size, a priori, increases the number of places where errors can occur and it is therefore non-trivial to design protocols which overall have a net decrease of the error rate when comparing bare, un-encoded systems, and logical encoded ones.

Consider, for example, the process of measuring the syndrome of an error correcting code. In order to measure one of the stabilizer generators, one needs to couple an auxiliary system with all the qubits involved in the measured stabilizer and then measure the auxiliary system. All these interactions can potentially spread errors. It can potentially affect the readout of the syndrome as well as growing the size of the error on the data qubits, which is potentially dangerous. This is why the design of syndrome extraction circuits has to be carefully done with fault-tolerance in mind. Several techniques have been designed to do this for any stabilizer code, see for instance [32, 44–46].

This issue is also crucial when considering that we not only want to store quantum information but also process it. In other words, we want to apply transformations to the physical system which act within the code space as some desired unitary transformations. Of course, any desired transformation of the encoded information can be realized in some way without a guarantee of fault-tolerance. For example, it could be a large circuit which spreads errors substantially or does not tolerate errors on its components. The task is thus to design a set of fault-tolerant gates forming a universal gate set on the encoded level.

One useful concept to realize fault-tolerant logical gates is that of transversality. The idea is that it is sometimes possible to realize logical operations on a code by acting separately on each physical qubit: logical Pauli operators of stabilizer codes are an example. Such gate is automatically fault-tolerant as by acting separately on each qubit, the errors stay localized. There exists a no-go theorem stating that universality through only transversal gates is impossible [47, 48]. Moreover it was shown that topological codes in D spatial dimensions can only implement logical gates from the Dth level of the Clifford hierarchy (see Eq. (1.39)) when using constant-depth geometrically local circuits [49]. Hence researchers have had to find workarounds to these theorems to make universal schemes.

A fault-tolerant scheme for quantum computation should therefore comprise a family of quantum error correcting codes with growing distance, fault-tolerant procedures for state preparation, syndrome extraction, a universal set of logical gates, measurements and an efficient and accurate decoder. Then the ultimate test for the scheme is to exhibit a threshold for a given model of noise. This means that there should be a value of the strength of the noise below which it is possible to reduce the noise on the encoded level to arbitrarily small values by choosing the correct code of the family (the one with large enough distance).

The evaluation of a fault-tolerant scheme then involves a consideration of the overhead in space and time it imposes as compared to the ideal circuit. This overhead will depend on the desired logical accuracy and the model and strength of the noise. The value of the threshold is also very important: To be able to profit from the scheme, one has to be able to build components in the lab with less noise than the threshold which is daunting if the threshold is low.

1.3.2. MAGIC STATES AND THEIR DISTILLATION

It is in general fairly easy to realize fault-tolerantly logical Clifford gates on stabilizer codes. This comes from the fact that Pauli stabilizers interact straightforwardly with Clifford operations. So for many stabilizer codes, a way to implement the full Clifford group fault-tolerantly is known. Then the main difficulty to get to universality is to find a way to implement for example a T gate or a CCZ gate from the third level of the Clifford hierarchy. The concept of magic states and magic state distillation has been devised to fulfill exactly this.

A magic state is a quantum state which can be used to implement a gate on an otherwise unknown state. This can be explained by examining one of the so-called one-bit teleportation circuits [50], represented in Figure 1.5. Consider a unitary gate U which

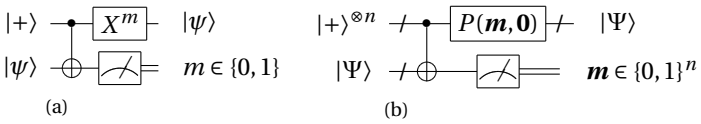


Figure 1.5: (a) One of the circuit called one bit teleportation. Using a CNOT and a measurement one exactly transfers the unknown single-qubit state $|\psi\rangle$ from the bottom wire to the top one. One needs to use the single-bit outcome of the measurement, m, to apply or not a Pauli X correction to the top wire. (b) The same circuit applied in parallel on a n-qubit state $|\Psi\rangle$. The n-bit outcome, $\boldsymbol{m}$, is used to determine the n-qubit Pauli X correction, $P(\boldsymbol{m},\boldsymbol{0})$, see Eq. (1.15).

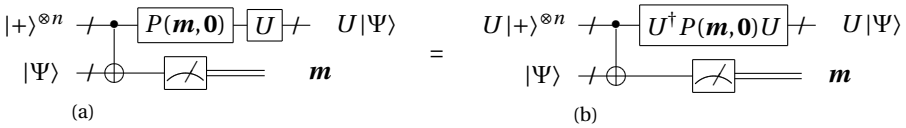


Figure 1.6: (a) Applying the gate U after the one-bit teleportation circuit, evidently produce the state $U|\Psi\rangle$. (b) If the gate U is diagonal in the computational basis, it can be commuted through the CNOTs. The task of implementing U then amounts to preparing the state $U|+\rangle^{\otimes n}$ and applying the correction $U^\dagger P(\boldsymbol{m},\boldsymbol{0})U$. If U is in the kth level of the Clifford hierarchy then by definition the correction is in the $(k-1)$th level, see Eq. (1.39).

is diagonal in the computational basis and belongs to the kth level of the Clifford hierarchy, see Eq. (1.39). Examining the circuit shown in Fig. 1.5b followed by the gate U, one can see that applying U on $|\Psi\rangle$ can be reduced to preparing a specific state $U|+\rangle^{\otimes n}$ and applying a correction in the $(k-1)$th level of the Clifford hierarchy, see Figure 1.6. In particular if U is in the third level of the Clifford hierarchy, e.g. the T gate, this shows

that one can implement universal quantum computation using only Clifford gates and a supply of $T|+\rangle$ states, which are referred to as magic states. The same is true of states like $CCZ|+\rangle^{\otimes 3}$ since CCZ also promotes the Clifford gates to universality.

Considering having access to fault-tolerant constructions for all the Clifford gates with a given code, the problem is now reduced to creating encoded magic states such as

$$|A\rangle = T|+\rangle, \tag{1.74}$$

to get access to universal and fault-tolerant quantum computation. As mentioned earlier, it is always possible to find some non-fault-tolerant way to implement any gate and so it is easy to obtain noisy encoded magic states, e.g. by running the encoding circuit of the code on an un-encoded magic state. Strikingly starting from noisy encoded magic state and using only fault-tolerant Clifford gates it is possible to distill better and better encoded magic states. This is called magic state distillation and several protocols have been devised [51–57].

One type of magic state distillation protocols involves finding a code exhibiting a transversal gate in the third level of the Clifford hierarchy. Consider having a $[[n,k,d]]$ code, $C_{\text{dist.}}$, such that applying transversally the $T^{\otimes n}$ gate on its n physical qubits realizes the gate $T^{\otimes k}$ on its k logical qubits. The general idea to use such a code in a distillation protocols is as follows:

1. Collect n logical qubits encoded each in a base code, C_{base}, with which one can realize all Clifford operations fault-tolerantly.
2. Using Clifford operations concatenate these n logical qubits into the code $C_{\text{dist.}}$ in the $|+\rangle^{\otimes k}$ logical state.
3. Collects n noisy encoded magic states $|A\rangle$ (see Eq. (1.74)).
4. Using the one-qubit version of the circuit shown in Fig. 1.6b use the n noisy magic states to apply with some noise the transversal T gate at the level of $C_{\text{dist.}}$.
5. Using Clifford operations extract the syndrome of $C_{\text{dist.}}$ and post-select on the trivial syndrome.
6. Decode $C_{\text{dist.}}$ to obtain k magic states encoded in C_{base} whose noise was reduced from some p roughly to p^d.

Provided that the quality of the initial magic states is not too low, repeating sufficiently many times the protocol will reach any desired accuracy. Then the amount of resources spent will directly depend on the parameters of the distillation code $[[n,k,d]]$. The efficiency of the protocol is often summarized in just one quantity:

$$\gamma = \frac{\log(n/k)}{\log(d)}, \tag{1.75}$$

since the average number of output distilled magic states at a desired accuracy, ϵ_{out}, per initial noisy magic state is given by $1/O\left(\log(\epsilon_{\text{out}}^{-1})^{\gamma}\right)$. So the smaller γ is, the more efficient the protocol is. Previously conjectured to be at least 1, it has recently been shown that $\gamma < 1$ is achievable [57]. In Chapter 5, Section 1.3.2 we devise a few such protocols.

1.3.3. TECHNIQUES TO GET TO FAULT-TOLERANT UNIVERSALITY

Different techniques have been devised to reach fault-tolerant universal quantum computation. We mention a few here, see also [7, 58–60].

CONCATENATION SCHEMES

The first proofs of the existence of fault-tolerant thresholds were using the principle of code concatenation to build code families and hierarchies of fault-tolerant circuits [61–63]. The noise models considered are independent on each qubit in space and time (or very weakly correlated). The idea of concatenation is to recursively encode the information on several levels of encoding. Then by carefully designing the circuits for the first level, the circuits for the higher levels are directly recursively defined. The thresholds for concatenated schemes are typically fairly low. The overhead they require is poly-logarithmic in the size, N, of the ideal circuit to be implemented (multiplied by $\log^c(N)$).

TOPOLOGICAL SCHEMES

The most popular fault-tolerant schemes are based on 2D topological codes, in particular the one based on the toric code [34, 64]. Their popularity comes from the comparatively high threshold that they offer and the natural geometrical locality that they have. This make them the first candidates in line for implementation. The overheads of these schemes are also poly-logarithmic. For 2D topological codes which have the highest thresholds it is reasonably easy to implement Clifford gates fault-tolerantly, using a mix of transversal gates and code deformation techniques (more about code deformation in Chapter 4). The main difficulty comes from the T gate or CCZ gate from the third level of the Clifford hierarchy for which magic state distillation is used, see Sec.1.3.2.

Another technique for topological codes is to leverage the fact that some 3D topological codes can have transversal gates in the third level of the hierarchy. For example, some 3D color codes with the right boundaries can have a transversal T gate. These $3D$ codes can for example replace the magic state distillation part of the computation to provide magic states to an otherwise 2D Clifford computation. Comparisons of overheads with magic state distillation are still underway. Some of the disadvantages of 3D codes are the overhead coming with 3D geometry and their generally lower thresholds.

These $3D$ codes could also be used for the whole architecture since it is possible to get a universal set of gates on them using a trick called gauge fixing [65]. Interestingly it is also possible by using the measurement based quantum computing approach to realize these $3D$ schemes in a quasi-2D manner where it is sufficient to only have a quasi-2D layer of the 3D code alive at any given time [66, 67]. The thresholds of such schemes are still to be precisely determined but likely to be on the lower sides.

SCHEMES WITH CONSTANT OVERHEAD

All the previous schemes have a poly-logarithmic overhead. The possibility of fault-tolerant schemes with constant overheads has been posed in [68] and recently realized in [69]. One of the main features of these schemes is that they leverage error correcting codes with a constant encoding rate, meaning that k/n goes to a constant with growing n and not to 0. Hence with constant rate codes, increasing the distance of the code also yields more logical qubits which participate in a wholesale effect with reduced cost. The

thresholds of these schemes are also currently under investigation. One of the difficulties with these schemes is the realization of logical gates within the same block of code. They currently use several sub-linear sized blocks and perform gates on one logical qubit of each block at a time. In particular, one desirable feature that this is missing so far is the possibility to realize gates in parallel within a block and therefore also keep the depth of the original circuit. Understanding the fault-tolerant ways to perform gates within a code block is ongoing research.

1.4. ORGANIZATION OF THE THESIS

The rest of this thesis consists in four chapters followed by a conclusion. The chapters cover diverse topics and are organized such that we start closest to the hardware and move gradually up the quantum computing stack.

Chapter 2 presents an experiment realized on a quantum chip whose access was provided by IBM. It is one of the first experiments testing in a real setting the principles of fault-tolerant quantum computation, using the full computational capabilities of a four qubit detecting code. The experimental results are one of the first validations of fault-tolerant designs of circuits as they show an improvement on average over the circuits tested.

Chapter 3 presents a theoretical study of fault-tolerance with quantum oscillators as building blocks. The idea is to consider starting from oscillators in the lab, encoding qubits into them using a first layer of error correction and then using these encoded qubits in a larger qubit error correcting code. Interestingly, we show a numerical threshold in the fault-tolerant regime, which can be interpreted on the level of the qubit code as an error model including leakage errors. We also present a no-go theorem for certain types of oscillator codes which we call linear oscillator codes showing that they cannot protect information.

Chapter 4 presents a theoretical study of code deformation techniques. We present a formalism to easily analyze their fault-tolerance properties and give a prescription about how to perform decoding during the procedure. We illustrate it on known and new code deformation techniques, which have the potential to improve the reliability of these techniques. We also use the new techniques to investigate a curious hybrid scheme mixing topological codes and concatenation.

Chapter 5 presents a new construction of CSS codes, which we call quantum pin codes, providing a generalization of color codes. This code family has the potential to host some codes with constant encoding rate and transversal non-Clifford gates. We define them, present some practical constructions and examples and study their transversality properties. Using these constructions and properties of pin codes, we are able to derive some efficient distillation protocols.

REFERENCES

[1] M. Planck, *On the Law of Distribution of Energy in the Normal Spectrum,* Annalen Phys. **4**, 553 (1901).

[2] A. Aspect, J. Dalibard, and G. Roger, *Experimental Test of Bell's Inequalities Using*

Time-Varying Analyzers, Physical Review Letters **49**, 1804 (1982).

[3] C. Cohen-Tannoudji, B. Diu, and F. Laloë, *Mécanique quantique*, Enseignement des sciences, Vol. I (Hermann, 1973).

[4] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Infomation* (Cambridge Unive. Press, Cambridge, MA, 2000).

[5] P. Kaye, R. Laflamme, and M. Mosca, *An Introduction to Quantum Computing* (Oxford University Press, Inc., New York, NY, USA, 2007).

[6] J. Watrous, *The Theory of Quantum Information* (Cambridge University Press, 2018).

[7] B. M. Terhal, *Quantum error correction for quantum memories,* Rev. Mod. Phys. **87**, 307 (2015).

[8] A. Turing, *On Computable Numbers, with an Application to the Entscheidungsproblem,* Proceedings of the London Mathematical Society Ser. 2 **42**, 220 (1936).

[9] S. Arora and B. Barak, *Computational Complexity: A Modern Approach*, 1st ed. (Cambridge University Press, Cambridge ; New York, 2009).

[10] S. Perifel, *Complexité Algorithmique* (Ellipses Marketing, Paris, 2014).

[11] R. Feynman, *Simulating Physics with Computers,* Int. Jour. of Theor. Phys. **21**, 467 (1982).

[12] D. Deutsch and R. Jozsa, *Rapid Solution of Problems by Quantum Computation,* in *Proceedings of the Royal Society of London*, Vol. A439 (1992) pp. 553–558.

[13] E. Bernstein and U. Vazirani, *Quantum Complexity Theory,* in *Proceedings of the Twenty-fifth Annual ACM Symposium on Theory of Computing*, STOC '93 (ACM, New York, NY, USA, 1993) pp. 11–20, event-place: San Diego, California, USA.

[14] A. C.-C. Yao, *Quantum Circuit Complexity,* in *Proceedings of 34th FOCS* (1993) pp. 352–360.

[15] P. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,* SIAM J. Sci. Statist. Comput. **26**, 1484 (1997).

[16] H. J. Briegel, D. E. Browne, W. Dür, R. Raussendorf, and M. Van den Nest, *Measurement-based quantum computation,* Nature Physics **5**, 19 (2009).

[17] A. Y. Kitaev, A. H. Shen, and M. N. Vyalyi, *Classical and Quantum Computation. Vol. 47 of Graduate Studies in Mathematics.* (American Mathematical Society, Providence, RI, 2002).

[18] P. O. Boykin, T. Mor, M. Pulver, V. Roychowdhury, and F. Vatan, *On Universal and Fault-Tolerant Quantum Computing: A Novel Basis and a New Constructive Proof of Universality for Shor's Basis,* in *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, FOCS '99 (IEEE Computer Society, Washington, DC, USA, 1999) pp. 486–.

[19] D. Gottesman, *The Heisenberg Representation of Quantum Computers,* arXiv:quant-ph/9807006 (1998), arXiv: quant-ph/9807006.

[20] C. Flühmann, T. L. Nguyen, M. Marinelli, V. Negnevitsky, K. Mehta, and J. P. Home, *Encoding a qubit in a trapped-ion mechanical oscillator,* Nature **566**, 513 (2019).

[21] P. Campagne-Ibarcq, A. Eickbusch, S. Touzard, E. Zalys-Geller, N. E. Frattini, V. V. Sivak, P. Reinhold, S. Puri, S. Shankar, R. J. Schoelkopf, L. Frunzio, M. Mirrahimi, and M. H. Devoret, *A stabilized logical quantum bit encoded in grid states of a superconducting cavity,* arXiv:1907.12487 [quant-ph] (2019), arXiv: 1907.12487.

[22] A. Grimm, N. E. Frattini, S. Puri, S. O. Mundhada, S. Touzard, M. Mirrahimi, S. M. Girvin, S. Shankar, and M. H. Devoret, *The Kerr-Cat Qubit: Stabilization, Readout, and Gates,* arXiv:1907.12131 [quant-ph] (2019), arXiv: 1907.12131.

[23] W. Dür, M. Hein, J. I. Cirac, and H.-J. Briegel, *Standard forms of noisy quantum operations via depolarization,* Physical Review A **72**, 052326 (2005).

[24] J. Preskill, *Quantum Computing in the NISQ era and beyond,* Quantum **2**, 79 (2018).

[25] P. W. Shor, *Scheme for reducing decoherence in quantum computer memory,* Phys. Rev. A **52**, R2493 (1995).

[26] A. M. Steane, *Error correcting codes in quantum theory,* Phys. Rev. Lett. **77**, 793 (1996).

[27] C. H. Bennett, D. P. DiVincenzo, J. A. Smolin, and W. K. Wootters, *Mixed-state entanglement and quantum error correction,* Physical Review A **54**, 3824 (1996).

[28] E. Knill, R. Laflamme, and L. Viola, *Theory of Quantum Error Correction for General Noise,* Phys. Rev. Lett. **84**, 2525 (2000).

[29] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes* (North-Holland, Amsterdam, 1981).

[30] A. R. Calderbank and P. W. Shor, *Good quantum error-correcting codes exist,* Physical Review A **54**, 1098 (1996).

[31] A. Steane, *Multiple Particle Interference and Quantum Error Correction,* in *Proceedings of the Royal Society of London*, Vol. A452 (1996) pp. 2551–2577.

[32] D. Gottesman, *Stabilizer codes and quantum error correction*, PhD Thesis, California Institute of Technology (1997).

[33] S. Bravyi, B. M. Terhal, and Bernhard Leemhuis, *Majorana fermion codes,* New Journal of Physics **12**, 083039 (2010).

[34] A. Y. Kitaev, *Fault-tolerant quantum computation by anyons,* Annals of Physics **303**, 2 (2003).

[35] M. H. Freedman and D. A. Meyer, *Projective Plane and Planar Quantum Codes,* Foundations of Computational Mathematics **1**, 325âĂŞ332 (2001).

[36] S. B. Bravyi and A. Y. Kitaev, *Quantum codes on a lattice with boundary,* (1998).

[37] N. Delfosse, P. Iyer, and D. Poulin, *Generalized surface codes and packing of logical qubits,* arXiv:1606.07116 [quant-ph] (2016), arXiv: 1606.07116.

[38] H. Bombin and M. A. Martin-Delgado, *Topological Quantum Distillation,* Physical Review Letters **97**, 180501 (2006).

[39] H. Bombin and M. A. Martin-Delgado, *Exact topological quantum order in $D = 3$ and beyond: Branyons and brane-net condensates,* Physical Review B **75**, 075103 (2007).

[40] A. Kubica and M. E. Beverland, *Universal transversal gates with color codes: A simplified approach,* Phys. Rev. A **91**, 032330 (2015).

[41] D. Gottesman, A. Kitaev, and J. Preskill, *Encoding a qubit in an oscillator,* Phys. Rev. A **64**, 012310 (2001).

[42] N. Bourbaki, *Éléments de mathématique. III. Topologie générale. Groupes additifs de R^n.* (Hermann, 1960).

[43] K. Noh, S. M. Girvin, and L. Jiang, *Encoding an oscillator into many oscillators,* arXiv:1903.12615 [quant-ph] (2019), arXiv: 1903.12615.

[44] D. P. DiVincenzo and P. W. Shor, *Fault-Tolerant Error Correction with Efficient Quantum Codes,* Physical Review Letters **77**, 3260 (1996).

[45] A. M. Steane, *Active stabilization, quantum computation, and quantum state synthesis,* Phys. Rev. Lett. **78**, 2252 (1997).

[46] E. Knill, *Quantum computing with realistically noisy devices,* Nature **434**, 39 (2005).

[47] B. Eastin and E. Knill, *Restrictions on transversal encoded quantum gate sets,* Phys. Rev. Lett. **102**, 110502 (2009).

[48] B. Zeng, A. W. Cross, and I. L. Chuang, *Transversality Versus Universality for Additive Quantum Codes,* IEEE Transactions on Information Theory **57**, 6272 (2011).

[49] S. Bravyi and R. König, *Classification of Topologically Protected Gates for Local Stabilizer Codes,* Phys. Rev. Lett. **110**, 170503 (2013).

[50] X. Zhou, D. W. Leung, and I. L. Chuang, *Methodology for quantum logic gate construction,* Phys. Rev. A **62**, 052316 (2000).

[51] S. Bravyi and A. Kitaev, *Universal quantum computation with ideal Clifford gates and noisy ancillas,* Physical Review A **71**, 022316 (2005).

[52] S. Bravyi and J. Haah, *Magic-state distillation with low overhead,* Physical Review A **86**, 052329 (2012).

[53] C. Jones, *Multilevel distillation of magic states for quantum computing,* Phys. Rev. A **87**, 042305 (2013).

[54] E. T. Campbell and M. Howard, *Unified framework for magic state distillation and multiqubit gate synthesis with reduced resource cost,* Physical Review A **95**, 022316 (2017).

[55] E. T. Campbell and M. Howard, *Unifying Gate Synthesis and Magic State Distillation,* Physical Review Letters **118**, 060501 (2017).

[56] J. Haah and M. B. Hastings, *Codes and Protocols for Distilling T, controlled-S, and Toffoli Gates,* Quantum **2**, 71 (2018).

[57] M. B. Hastings and J. Haah, *Distillation with Sublogarithmic Overhead,* Physical Review Letters **120**, 050504 (2018).

[58] J. Preskill, *Reliable quantum computers,* Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences **454**, 385 (1998).

[59] D. Aharonov, *Noisy quantum computation*, Ph.D. thesis, The Hebrew University, Jerusalem (1999).

[60] E. T. Campbell, B. M. Terhal, and C. Vuillot, *Roads towards fault-tolerant universal quantum computation,* Nature **549**, 172 (2017).

[61] D. Aharonov and M. Ben-Or, *Fault-tolerant Quantum Computation with Constant Error,* in *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97 (ACM, New York, NY, USA, 1997) pp. 176–188, event-place: El Paso, Texas, USA.

[62] E. Knill, R. Laflamme, and W. H. Zurek, *Resilient quantum computation: error models and thresholds,* Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences **454**, 365 (1998).

[63] P. Aliferis, D. Gottesman, and J. Preskill, *Quantum Accuracy Threshold for Concatenated Distance-3 Codes,* Quantum Info. Comput. **6**, 97 (2006).

[64] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, *Topological quantum memory,* Journal of Mathematical Physics **43**, 4452 (2002).

[65] H. Bombín, *Gauge color codes: optimal transversal gates and gauge fixing in topological stabilizer codes,* New Journal of Physics **17**, 083002 (2015).

[66] H. Bombin, *2d quantum computation with 3d topological codes,* arXiv:1810.09571 [quant-ph] (2018), arXiv: 1810.09571.

[67] B. J. Brown, *A fault-tolerant non-Clifford gate for the surface code in two dimensions,* arXiv:1903.11634 [cond-mat, physics:quant-ph] (2019), arXiv: 1903.11634.

[68] D. Gottesman, *Fault-Tolerant Quantum Computation with Constant Overhead,* Quant. Information and Computation **14**, 1338 (2014).

[69] O. Fawzi, A. Grospellier, and A. Leverrier, *Constant Overhead Quantum Fault-Tolerance with Quantum Expander Codes,* in *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)* (2018) pp. 743–754.

2

TESTING QUANTUM FAULT-TOLERANCE

This chapter reports on experiments realized on several IBM 5Q chips which show evidence for the advantage of using error detection and fault-tolerant design of quantum circuits. An average improvement of the task of sampling from states that can be fault-tolerantly prepared in the $[[4,2,2]]$ *code is shown, when using a fault-tolerant technique well suited to the layout of the chip.*

This chapter has been published in Quantum Information & Computation **18**, 11&12, 0949-0964 (2018), [1]. The paper received the IBM Q Best Paper Award 1st place, 2018.

2.1. INTRODUCTION

QUANTUM systems in laboratories around the world are reaching unprecedented level of control and precision for a variety of devices aiming at implementing reliable qubits. Yet, due to the very nature of those devices, errors are still notably present. In order to be able to execute long quantum algorithms, improvements should be made using quantum error correction and fault-tolerant schemes. So-called threshold theorems [2, 3] prove that there exist error rates below which this approach is guaranteed to improve the device performance. However, between those theorems and real experiments there remains a fog of technical details which clouds the actual practicality of error correcting codes.

Already some experiments which demonstrate the usefulness of quantum error correcting codes to protect a quantum memory have been done, e.g. [4–8], but a demonstration of protected computation is still missing.

Inspired by a recent proposal by D. Gottesman [9], we use an IBM 5Q chip to show that error detection can improve some simple sampling tasks, thus turning a tiny portion of fog into blue sky. Closely related to this work, [7] and [8] present experiments based on the same error detecting code, on trapped-ion qubits and on a similar superconducting chip, respectively. Both those works focus on the preparation of few states and only one of the two encoded qubits is handled fault-tolerantly. In the present work an extensive set of fault-tolerant circuits is studied and both logical qubits are protected. Moreover [7] and [8] introduce artificial Pauli errors to study the robustness of their preparation whereas this work takes a higher level approach, treating the experimental set-up as a black box, probing only the intrinsic errors in the system. These considerations make the present work closer to the spirit of [9]. This work is, to the author's knowledge, the first experimental demonstration of error detection and fault-tolerance improving a quantum computation.

The chapter is organized as follows. In Section 2.2 we present the principle of the approach and its specialization to the IBM 5Q chip. In Section 2.3 the experimental results are shown and analyzed.

2.2. DEMONSTRATING FAULT-TOLERANCE

The idea behind fault-tolerance is to devise error correcting codes and the corresponding processes of encoding, correcting, computing or measuring with encoded information such that more errors are corrected than introduced [10]. The difficulty in devising such processes is that they are built out of components that are all faulty, so adding some might do more harm than it can help. Devising, proving and simulating fault-tolerant schemes has been pursued for the last 20 years, e.g. [2, 10–16], but improving and finding better schemes is still important and a subject of ongoing research, e.g. [17–21].

Taking one of these schemes and experimentally demonstrating its fault-tolerance, i.e finding an improvement of performance going from bare to encoded implementation, provides the ultimate validation of the scheme.

2.2.1. GENERAL APPROACH

A description of a general approach to demonstrate fault-tolerant quantum computation is given in [9]; we just briefly recall it here. The idea is to choose a quantum error correcting code $\mathscr{C}$, admitting fault-tolerant circuits for the preparation of some logical states, denoted as $|s_1\rangle, \ldots, |s_m\rangle \in S_{\mathrm{FT}}^{\mathscr{C}}$, as well as for some logical quantum gates, denoted as $U_1, \ldots, U_n \in G_{\mathrm{FT}}^{\mathscr{C}}$. Using these as building blocks one can then randomly draw a state preparation from $S_{\mathrm{FT}}^{\mathscr{C}}$, then draw a sequence of gates from $G_{\mathrm{FT}}^{\mathscr{C}}$ to obtain an encoded fault-tolerant circuit.

More precisely, following the formalism of *rectangles* and *extended rectangles* introduced in [11], one interleaves the logical units (gates in $G_{\mathrm{FT}}^{\mathscr{C}}$, or preparation of states in $S_{\mathrm{FT}}^{\mathscr{C}}$) with a fault-tolerant circuit for error correction. *Rectangles* designate circuits comprising one logical unit preceded by a round of error correction, *extended rectangles* designate circuits comprising one logical unit preceded and followed by a round of error correction. The precise conditions to satisfy in order to be fault-tolerant for distance 3 codes are stated in [11]. This ensures that the whole computation is error free if there is at most one fault per extended rectangle.

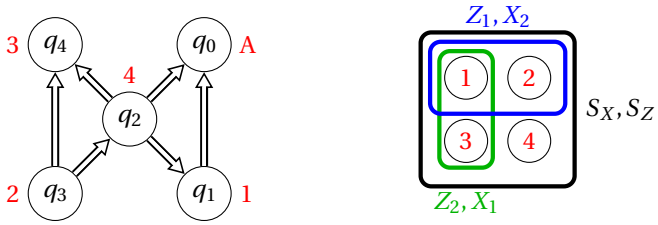
The error model considered in this paper is that the failure of any component can introduce any Pauli error on the qubits acted on by the component. In our case, we cannot repeat fault-tolerant detection of errors, so our circuits will only tolerate a single fault in total.

The encoded circuits then have to be compared to a bare implementation on the physical qubits. Sampling from the final state produced by the circuit in the computational basis and comparing the probability distribution obtained with the expected one gives a simple comparison metric. Note that this supposes that the circuits are small enough or simple enough so that one can classically simulate sampling from the final state efficiently. A successful demonstration of fault-tolerant quantum computation happens if the encoded circuits show better performance than the bare circuits. To be the most convincing, one needs to use the best of the physical qubits as well as the most efficient implementation for the bare circuits.

2.2.2. THE IBM 5Q CHIP AND [[4,2,2]]

In 2016 IBM released a quantum chip with fixed-frequency superconducting transmon qubits, named IBM 5Q. They provide worldwide cloud access to the chip under an initiative called the "IBM Quantum Experience" [22]. The current iteration on which the experiments were done is nicknamed *Raven*. The appendix also presents and compares preliminary results obtained with a previous iteration, *Sparrow*. The chip has five qubits, natively named q_0 to q_4. It features single-qubit Clifford gates, the T gate ($\mathrm{diag}(1, e^{i\pi/4})$) as well as some two-qubit CNOTs with a certain layout represented in Figure 2.1a. This many qubits is the right number to use for a demonstration using the $[[4,2,2]]$ code as discussed in [9].

The $[[4,2,2]]$ code encodes two qubits into four physical qubits. Its code space is stabilized by the all-X and all-Z Pauli operators ($S_X = X \otimes X \otimes X \otimes X$, $S_Z = Z \otimes Z \otimes Z \otimes Z$), together with the logical Pauli operators, they are represented in Figure 2.1b. The logical



(a) Layout (b) [[4,2,2]] Code

Figure 2.1: (a) The connectivity between the qubits in the IBM 5Q Raven chip. Arrows indicate CNOT capabilities where the arrow points towards the target of the CNOT. Red labels indicate our choice of qubit numbering. A is an ancillary qubit used in verification of state preparation. (b) The usual representation layout for the [[4,2,2]] code. The numbered circles represent the qubits and the rectangular boxes represent the support of different Pauli operators: logical Pauli operators Z_1, X_2 are highlighted in blue, logical Pauli operators Z_2, X_1 in green and stabilizers S_X, S_Z in black.

code states are

$$|00\rangle_L = \frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle) \tag{2.1}$$

$$|01\rangle_L = \frac{1}{\sqrt{2}}(|1100\rangle + |0011\rangle) \tag{2.2}$$

$$|10\rangle_L = \frac{1}{\sqrt{2}}(|1010\rangle + |0101\rangle) \tag{2.3}$$

$$|11\rangle_L = \frac{1}{\sqrt{2}}(|1001\rangle + |0110\rangle)\,. \tag{2.4}$$

The code also admits transversal implementations of two Clifford gates, namely $H \otimes H \cdot$ SWAP, where H is the Hadamard gate, see Figure 2.2a, and the controlled phase or CZ gate, see Figure 2.2b. Moreover, if the five qubits have the right connectivity then there

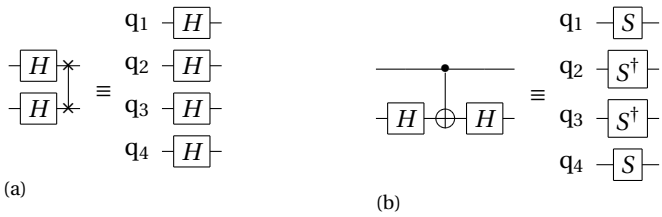
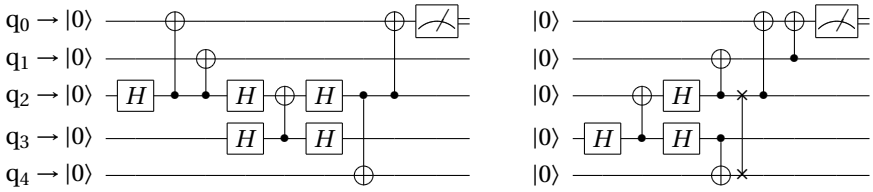


Figure 2.2: (a) Bare and encoded versions of $H \otimes H \cdot$SWAP, where H is the Hadamard gate. In practice the SWAP gate is not done physically on the bare version but just in software, by renumbering the two qubits. (b) Bare and encoded versions of the CZ gate, where $S = \mathrm{diag}(1, i)$.

are fault-tolerant circuits preparing the logical states $|00\rangle_L$, $|0+\rangle_L$ and $(|00\rangle_L + |11\rangle_L)/\sqrt{2}$. For the cat state $|00\rangle_L$, see Equation (2.1), there exists fault-tolerant preparations including one ancillary qubit to check the preparation. One post-selects on a successful preparation. The logical states $|0+\rangle_L$ and $|00\rangle_L + |11\rangle_L$ are both two Bell pairs but with different pairings. They both also have fault-tolerant preparation circuits. In [9], some circuits are

proposed, they can be adapted to the layout of the chip. We tested another preparation for $|00\rangle_L$ inspired by [17], which is substantially shorter so more relevant to this layout.



(a) FTv1 (b) FTv2

Figure 2.3: The two different implementations for preparing the state $|00\rangle_L$: (a) Based on a flagging technique with 5 CNOTs, (b) based on the circular connectivity circuit with 8 CNOTs. In both circuits if the ancillary qubit q_0 is measured as 1, we reject the preparation.

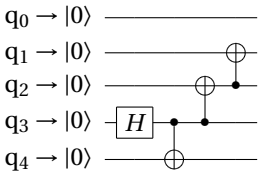


Figure 2.4: A short but non-fault-tolerant circuit to prepare $|00\rangle_L$.

The flagging technique in [17] for the preparation of $|00\rangle_L$ can be implemented with the given CNOT connectivity, see Figure 2.3a. We call this the fault-tolerant version one (FTv1). The technique based on a circular layout proposed by [9] cannot be implemented directly due to the connectivity. A clever introduction of one SWAP gate permits to implement it still fault-tolerantly at the cost of 8 CNOT gates in total, see Figure 2.3b. We call this the fault-tolerant version two (FTv2). We present below, the results for FTv1, which is more appropriate for the layout. For both those circuits one can check that every possible single fault leads to a detectable error, or an error that stabilizes the prepared state. We also tried a non-fault-tolerant one that has the advantage of being short, involving only 3 CNOTs, see Figure 2.4. We call this the non-fault-tolerant version (NFT). For the preparation of $|0+\rangle_L$, we want to create a Bell pair between q_1 and q_3 and between q_2 and q_4. There is a missing connection between q_1 and q_3, but we can do a similar SWAP trick, see the resulting circuit in Figure 2.6a. This circuit is not fault-tolerant as an undetected logical $Z_1 \otimes Z_2 = Z \otimes \mathbb{1} \otimes \mathbb{1} \otimes Z$ can occur if the SWAP gate fails. This is the only possible harmful logical error for this circuit. We'll see below that we only use this circuit in cases were the possible undetected $Z_1 \otimes Z_2$ error does not change the outcome of the sampling test. The preparation for $|00\rangle_L + |11\rangle_L$ can itself be straightforwardly implemented fault-tolerantly, see Figure 2.6b. To summarize, our sets of initial

logical states and gates are

$$\begin{aligned} S_{\mathrm{FT}} &= \left\{ |00\rangle, |0+\rangle, \frac{|00\rangle + |11\rangle}{\sqrt{2}} \right\}, \\ G_{\mathrm{FT}} &= \{X_1, X_2, Z_1, Z_2, H \otimes H \cdot \mathrm{SWAP}, CZ\}. \end{aligned}$$

The $[[4,2,2]]$ code is only an error detecting code, that is, it can detect one Pauli error but cannot correct it. This means that in place of error correction we have to rely on post-selection to remove errors. In other words, we throw away runs where we detect that an error occurred, either from the ancilla measurement checking state preparation, or from the final measurement which indicates, when the parity of the outcomes is odd, that S_Z has value -1.

As mentioned before, we cannot interleave rounds of error detection between state preparation and the gates. That means that the final circuit can only be tolerant to a single fault during the whole computation, except for the specific undetectable $Z_1 \otimes Z_2$ failure mentioned above and when we use the non-fault-tolerant version to prepare $|00\rangle_{\mathrm{L}}$.

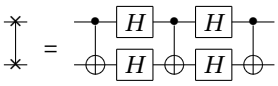


Figure 2.5: Implementation of the SWAP gate.

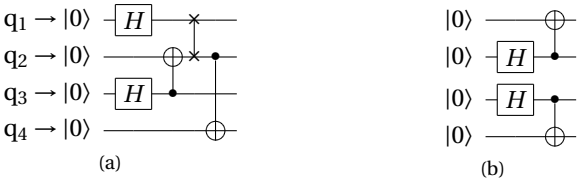


Figure 2.6: (a) Preparation circuits of the logical state $|0+\rangle_{\mathrm{L}}$, with 5 CNOTs. (b) Preparation circuits of the logical Bell state $\left(|00\rangle_{\mathrm{L}} + |11\rangle_{\mathrm{L}}\right)/\sqrt{2}$ with 2 CNOTs.

2.2.3. COMMENTS ON THE TESTED CIRCUITS

Since we work with such a small system, we can exhaustively find and try all the logically equivalent circuits and optimize the bare version for each one. Essentially, we are making it most likely for the bare version of the task to prevail. From the set of states S_{FT}, and the set of gates G_{FT}, one can obtain 20 different stabilizer states. All the states with their most efficient bare preparation circuit, using the native set of gates provided by IBM, are listed in Table 2.1. A brute force approach was used to find them.

Note that adding more gates in the sequence would test states that are already tested with shorter circuits. Therefore those would certainly give worse results as we cannot interleave error detections between gates. Hence we kept only these 20 different preparations. This still extensively probes the computational capabilities of the $[[4,2,2]]$ code.

Initial state	Unitary	Final state	# Instructions
$\|00\rangle$	$\mathbb{1}\otimes\mathbb{1}$	$\|00\rangle$	5
$\|0+\rangle$	$\mathbb{1}\otimes\mathbb{1}$	$\frac{\|00\rangle+\|01\rangle}{\sqrt{2}}$	6
$\|00\rangle$	$\mathbb{1}\otimes X$	$\|01\rangle$	6
$\|00\rangle$	$X\otimes\mathbb{1}$	$\|10\rangle$	6
$\frac{\|00\rangle+\|11\rangle}{\sqrt{2}}$	$\mathbb{1}\otimes\mathbb{1}$	$\frac{\|00\rangle+\|11\rangle}{\sqrt{2}}$	7
$\|0+\rangle$	$\mathbb{1}\otimes Z$	$\frac{\|00\rangle-\|01\rangle}{\sqrt{2}}$	7
$\|00\rangle$	$H\otimes H\cdot\mathrm{SWAP}$	$\frac{\|00\rangle+\|01\rangle+\|10\rangle+\|11\rangle}{2}$	7
$\|0+\rangle$	$X\otimes\mathbb{1}$	$\frac{\|10\rangle+\|11\rangle}{\sqrt{2}}$	7
$\|00\rangle$	$X\otimes X$	$\|11\rangle$	7
$\frac{\|00\rangle+\|11\rangle}{\sqrt{2}}$	$\mathbb{1}\otimes Z$	$\frac{\|00\rangle-\|11\rangle}{\sqrt{2}}$	8
$\frac{\|00\rangle+\|11\rangle}{\sqrt{2}}$	$X\otimes\mathbb{1}$	$\frac{\|10\rangle+\|01\rangle}{\sqrt{2}}$	8
$\|00\rangle$	$\mathbb{1}\otimes Z\cdot H\otimes H\cdot\mathrm{SWAP}$	$\frac{\|00\rangle-\|01\rangle+\|10\rangle-\|11\rangle}{2}$	8
$\|00\rangle$	$Z\otimes\mathbb{1}\cdot H\otimes H\cdot\mathrm{SWAP}$	$\frac{\|00\rangle+\|01\rangle-\|10\rangle-\|11\rangle}{2}$	8
$\|0+\rangle$	$X\otimes Z$	$\frac{\|10\rangle-\|11\rangle}{\sqrt{2}}$	8
$\frac{\|00\rangle+\|11\rangle}{\sqrt{2}}$	$\mathbb{1}\otimes ZX$	$\frac{\|10\rangle-\|01\rangle}{\sqrt{2}}$	9
$\|00\rangle$	$Z\otimes Z\cdot H\otimes H\cdot\mathrm{SWAP}$	$\frac{\|00\rangle-\|01\rangle-\|10\rangle+\|11\rangle}{2}$	9
$\|00\rangle$	$CZ\cdot H\otimes H\cdot\mathrm{SWAP}$	$\frac{\|00\rangle+\|01\rangle+\|10\rangle-\|11\rangle}{2}$	10
$\|00\rangle$	$CZ\cdot\mathbb{1}\otimes Z\cdot H\otimes H\cdot\mathrm{SWAP}$	$\frac{\|00\rangle-\|01\rangle+\|10\rangle+\|11\rangle}{2}$	11
$\|00\rangle$	$CZ\cdot Z\otimes\mathbb{1}\cdot H\otimes H\cdot\mathrm{SWAP}$	$\frac{\|00\rangle+\|01\rangle-\|10\rangle+\|11\rangle}{2}$	11
$\|00\rangle$	$\mathbb{1}\otimes X\cdot CZ\cdot H\otimes H\cdot\mathrm{SWAP}\cdot X\otimes\mathbb{1}$	$\frac{\|00\rangle-\|01\rangle-\|10\rangle-\|11\rangle}{2}$	12

Table 2.1: The list of initial states and unitary circuits and the number of QASM instructions in the bare circuit (including final measurements). The final states are written exclusively in the computational basis since that is how they are measured.

2.3. EXPERIMENTAL RESULTS

The code and data can be found on GitHub [23]: it uses the Python SDK that can be found at [24].

2.3.1. PARAMETERS AND RUNS

Using the IBM chip one can run circuits in batches. For each individual run, 8192 shots are done right one after another in a short time, where each shot outputs 5 bits as measurement outcomes. We consider that the chip is exactly in the same conditions during each run. We consider all the runs to be independent of one another but each to be done in different conditions, so sampling from different output distributions.

For each run, IBM also provides calibration data describing the state of the chip such as: gate error rates, readout error rates, T_1, T_2 and fridge temperature. We give the average values that we observed for our runs in the appendix.

2.3.2. PERFORMANCE METRIC

For each circuit run we want to compare the observed outcome distribution with the ideal one. The ideal distribution is 8192 independent samples from a distribution with four possible outcomes occurring with probabilities p_{00}, p_{01}, p_{10} and p_{11}. The values for p_{ij} can be read from Table 2.1. Since we assume that the conditions stay identical during one run and that the 8192 shots are independent, we observe independent samples from a four-outcome distribution with some different probabilities $\tilde{p}_{00}$, $\tilde{p}_{01}$, $\tilde{p}_{10}$ and $\tilde{p}_{11}$.

We then use the statistical distance as a metric:

$$D = \frac{1}{2}\Big(|p_{00} - \tilde{p}_{00}| + |p_{01} - \tilde{p}_{01}| + |p_{10} - \tilde{p}_{10}| + |p_{11} - \tilde{p}_{11}|\Big).$$

This quantity is estimated for each run by

$$\hat{D} = \frac{1}{2}\left(\left|p_{00} - \frac{n_{00}}{n_{\text{valid}}}\right| + \left|p_{01} - \frac{n_{01}}{n_{\text{valid}}}\right| + \left|p_{10} - \frac{n_{10}}{n_{\text{valid}}}\right| + \left|p_{11} - \frac{n_{11}}{n_{\text{valid}}}\right|\right),$$

where n_{ij} is the number of observation of outcome ij after post-selection and n_{valid} is the number of shots kept after post-selection. This estimator for D is slightly biased except for the case where only one of the p_{ij} is non-zero (because in this case it becomes linear). We use this estimator to keep the analysis simple.

Each of the runs has some different $\tilde{p}_{ij}$ and we have no information about how the $\tilde{p}_{ij}$s vary. Therefore we will assume that there are fluctuations around the mean of $\hat{D}$ following some unknown normal distribution and use this model to compute confidence intervals [25]. This means that the final data points and their confidence intervals don't exactly reflect knowledge of D but only of $\hat{D}$ which we believe is still a valid quantity to characterize the performance of the circuits.

2.3.3. COMPARISONS

We first need to decide on what pair of bare qubits is the best to be compared to the encoded qubits. It is not immediately clear how to choose this given only the calibration numbers. Thus we tried out the six different connected pairs and we found the performance shown in Figure 2.7. There is no clear "best pair" but one can see that the pairs

Figure 2.7: Comparison of the different pairs of qubits to implement the bare circuits together with the number of instructions for each circuit.

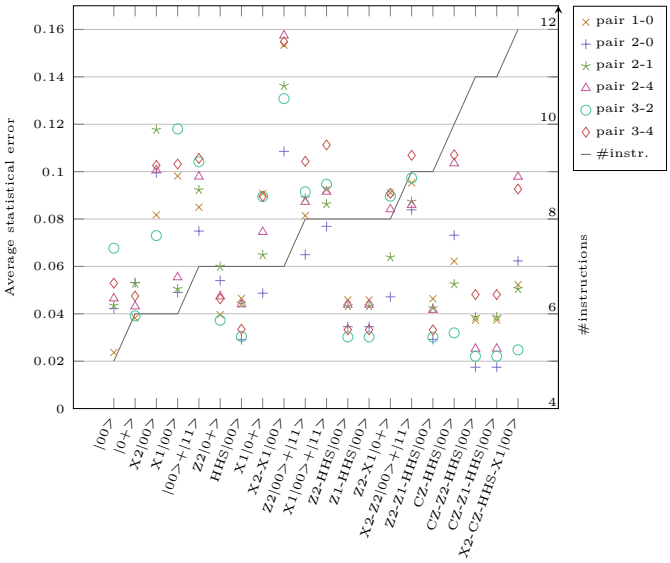
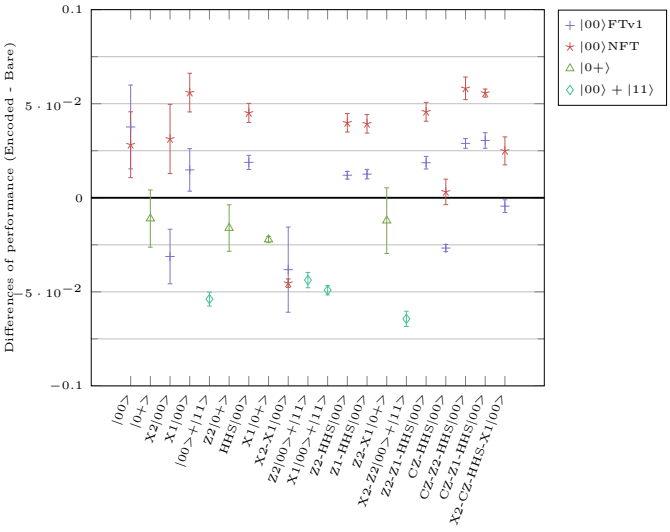


Figure 2.8: Comparing encoded performance with bare qubit pair [2 – 0]. The performance is defined as the statistical distance to the ideal outcome distribution. The figure shows the difference between encoded and bare performance. The error bars show confidence intervals at 99%.



[2 – 0] or [3 – 2] seem to be slightly better. Since we don't have a systematic method for predicting the best pair given the calibration data and the circuit, we will look at average

Implementation	Avg.Perf.($\times 10^{-2}$)	Post-selection ratio
Bare[1-0]	6.72	1.00
Bare[2-0]	5.50	1.00
Bare[2-1]	6.49	1.00
Bare[2-4]	6.98	1.00
Bare[3-2]	6.27	1.00
Bare[3-4]	7.73	1.00
FTv1	4.51	0.65
NFT	6.05	0.71

Table 2.2: Average performance for the different bare and encoded possible implementations with the post-selection ratio (ratio of data kept). The encoded implementations only differ in how they prepare the state $|00\rangle_L$.

performance for each pair over all the circuits. When averaging, we find the pair $[2-0]$ to be the best, see Table 2.2. The second observation that we can make based on Figure 2.7 is that the number of instructions does not explain the performance. It seems that the type of state sampled from is more important. Roughly it is easier to sample from equal superposition of the four computational basis states, than from equal superposition of two, than from just one. This can be understood with the fact that with more states in equal superposition there are less Pauli errors or readout errors that can affect the outcome distribution.

We then go on, comparing the encoded versions of the circuits to the elected best pair in Figure 2.8. The encoded circuits using the preparations $|0+\rangle_L$ and $|00\rangle_L + |11\rangle_L$ are short and fault-tolerant and indeed show better performance than the bare ones. For the different preparations for $|00\rangle_L$, the fault-tolerant version FTv1 is, except in a few cases, better than the non-fault-tolerant one despite being substantially longer. This shows that fault-tolerant design of circuits can be useful. Although they both compare unfavourably to the bare version, except for 4 circuits for FTv1 and only 1 circuit for NFT.

We cannot make an absolute statement as for some circuits the bare version is always better. We also don't want to cherry pick the best version for each circuit since we don't have a systematic way of predicting the best version. The average performance when fixing a preparation for $|00\rangle_L$ is shown in Table 2.2. One can see that FTv1, with 4.51×10^{-2}, is better than the best bare implementation with 5.5×10^{-2}, whereas NFT (6.05×10^{-2}) is worse.

2.4. CALIBRATION DATA AND ADDITIONAL EXPERIMENT

This section presents the calibration data for the experiments, averaged over all the different runs as well as the observed standard deviation. It also shows previous runs on a previous iteration of the chip, called Sparrow, which was showing better performance but has been taken down by IBM. The current iteration is named Raven: the more extensive runs presented in this paper were done on this chip. Note that for the runs on Sparrow, there were only 4 different calibrations and 36 runs for each circuit. For Raven we have more than 100 runs for each circuit and almost a new calibration per run.

chip	Temperature (mK)
Raven	21
Sparrow	19

Table 2.3: Average fridge temperatures for the two chips at the time of the runs in mK.

qubit	T_1 (ţs)	$\sigma[T_1]$	T_2 (ţs)	$\sigma[T_2]$
Q0	46	6.5	40	7.3
Q1	57	6.8	45	6.3
Q2	39	5.1	40	3.2
Q3	40	4.8	57	12.5
Q4	54	9.1	21	1.3

(a) Raven

qubit	T_1 (ţs)	$\sigma[T_1]$	T_2 (ţs)	$\sigma[T_2]$
Q0	49	8.9	39	4.5
Q1	46	11.0	39	13.3
Q2	56	1.7	91	9.9
Q3	47	4.9	62	12.0
Q4	55	6.0	81	13.2

(b) Sparrow

Table 2.4: Average T_1 and T_2 parameters for the runs together with their standard deviation all in ţs.

The two chips are similar, the gate and readout error rates have been slightly improved with Raven but the fridge temperature as well as T_1 and T_2 times became a bit worse. The layout differs only in the orientation of the CNOTs, see Figure 2.9. The circuits run on Sparrow are shown in Figure 2.10a and Figure 2.10. Comparison of the bare and encoded performance between Raven and Sparrow are presented in Figure 2.11 and Figure 2.12. One can see that the previous iteration was performing better.

qubit	gate error (%)	σ[gate error]	readout error (%)	σ[readout error]
Q0	0.1	0.021	4.1	0.63
Q1	0.08	0.035	4.7	0.96
Q2	0.15	0.024	2.7	0.36
Q3	0.15	0.016	6.2	1.11
Q4	0.19	0.026	5.1	1.1

(a) Raven

qubit	gate error (%)	σ[gate error]	readout error (%)	σ[readout error]
Q0	0.23	0.009	1.9	0.22
Q1	0.30	0.099	8.3	2.19
Q2	0.42	0.085	1.5	0.29
Q3	0.52	0.213	12.3	3.60
Q4	0.33	0.027	6.6	0.83

(b) Sparrow

Table 2.5: Average single-qubit gate and readout error rates for the runs together with their standard deviation, all in percent.

pair	gate error (%)	σ[gate error]
1-0	2.2	0.32
2-0	2.3	0.25
2-1	2.7	0.36
2-4	3.9	0.78
3-2	2.2	0.30
3-4	2.7	0.38

(a) Raven

pair	CNOT error (%)	σ[CNOT error]
1-0	4.0	0.74
2-0	3.0	0.24
2-1	4.6	0.86
2-4	3.7	0.34
3-2	6.2	0.90
3-4	5.8	2.05

(b) Sparrow

Table 2.6: Average CNOT gate error rates and their standard deviation for the runs, all in percent.

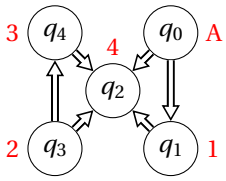


Figure 2.9: Layout of the Sparrow chip.

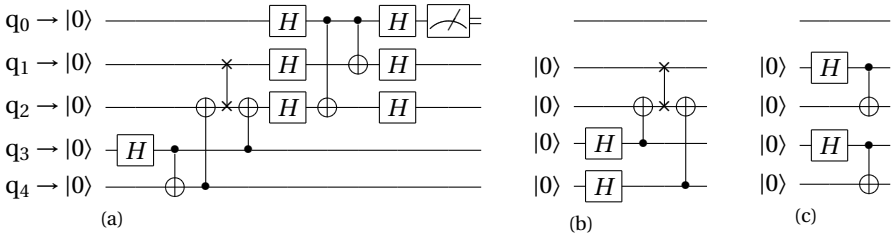


Figure 2.10: (a)The circuit previously implemented on the IBM 5Q Sparrow chip preparing the logical state $|00\rangle_L$. The SWAP gate is implemented via the circuit in Figure 2.5. If the SWAP gate fails, it can introduce Pauli X errors on q_1 and q_2, which would not be detected and which constitute a logical $X_1 \otimes X_2$ error. (b) Preparation circuits of the logical state $|0+\rangle_L$, with 5 CNOTs. If the SWAP gate fails, it can introduce Pauli X errors on q_1 and q_2, which would not be detected and which constitute a logical $X_1 \otimes X_2$ error. (c) Preparation circuits of the logical Bell state $\left(|00\rangle_L + |11\rangle_L\right)/\sqrt{2}$ with 2 CNOTs.

2.5. CONCLUSION

In conclusion, we have shown that already on the IBM 5Q chip one can improve some quantum computation task, namely sampling from a class of states, by using error detection and fault-tolerant design of circuits. This improvement is only on average over 20 different states that the $[[4,2,2]]$ code can fault-tolerantly prepare. We also can see that shorter but non fault-tolerant circuits can be bested by longer but fault-tolerant circuits, showing the usefulness of these. As better and better hardware is developed,

Figure 2.11: Comparison of the performances of the bare circuits between the Raven and Sparrow chips. Except for the four circuits involving a two-qubit gate, Sparrow was showing better performance. This is probably due to improvement of the two-qubit gates but shorter T_1 and T_2 times on Raven, see Table 2.4 and Table 2.6.

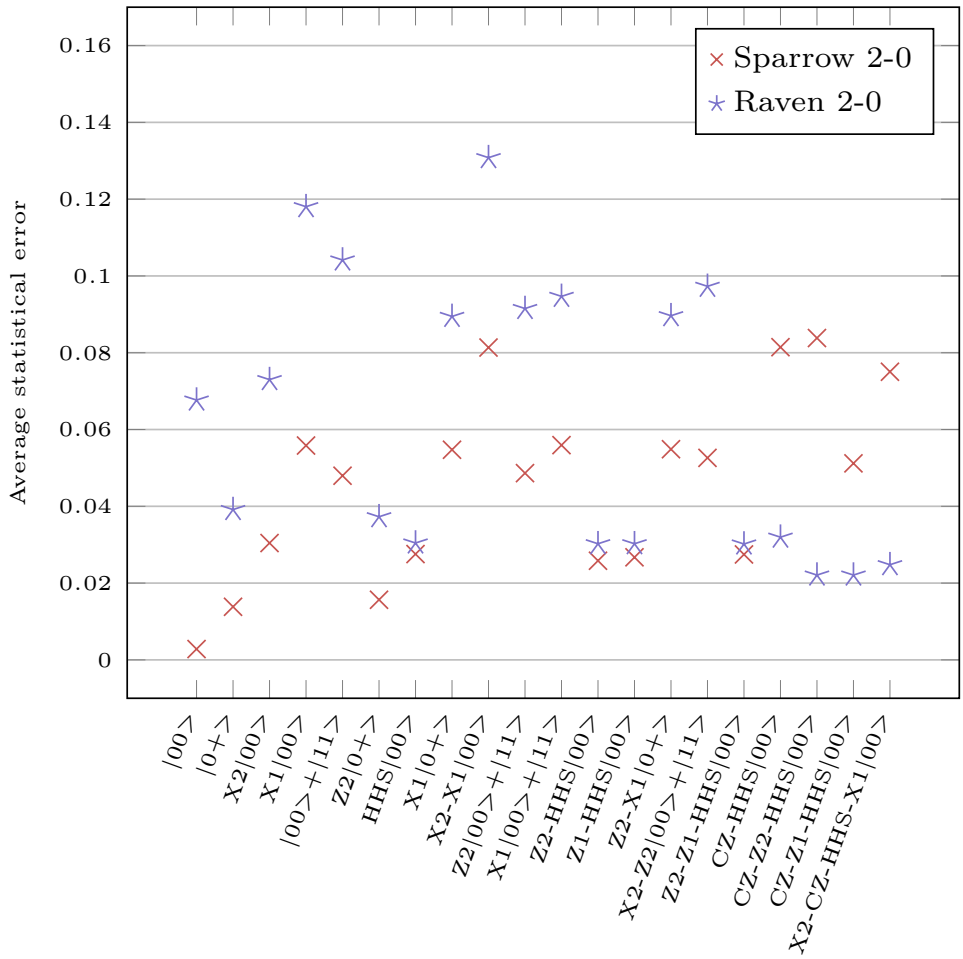
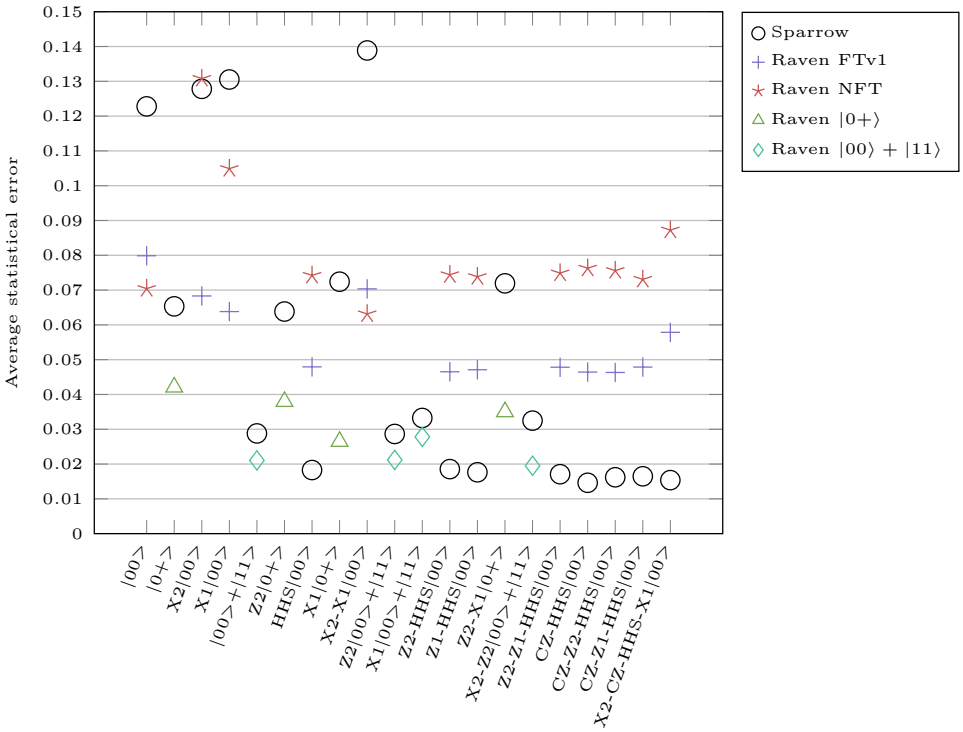


Figure 2.12: Comparing encoded versions on Raven and Sparrow. The $|00\rangle_L$ and $|0+\rangle_L$ preparations were suffering from possible $X_1 X_2$ logical errors.



with more physical qubits, more connectivity and smaller error rates, demonstrations of fault-tolerance will become easier to produce and become more convincing. The set of gates shown to be fault-tolerant in this chapter is very restricted. For the [[4,2,2]] code a few more qubits and connections would be needed to realize fully fault-tolerant circuits with error detection in between logical units. Being able to demonstrate the fault-tolerance of larger gate sets, for example the whole Clifford group for several logical qubits, would be an important milestone towards harnessing universal quantum computation.

REFERENCES

[1] C. Vuillot, *Is error detection helpful on IBM 5q chips ?* Quantum Information and Computation **18**, 0949 (2018).

[2] D. Aharonov and M. Ben-Or, *Fault-tolerant Quantum Computation with Constant Error,* in *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97 (ACM, New York, NY, USA, 1997) pp. 176–188, event-place: El Paso, Texas, USA.

[3] E. Knill, R. Laflamme, and W. H. Zurek, *Resilient quantum computation: error models and thresholds,* Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences **454**, 365 (1998).

[4] E. Knill, R. Laflamme, R. Martinez, and C. Negrevergne, *Benchmarking Quantum Computers: The Five-Qubit Error Correcting Code,* Phys. Rev. Lett. **86**, 5811 (2001).

[5] D. Nigg, M. Müller, E. A. Martinez, P. Schindler, M. Hennrich, T. Monz, M. A. Martin-Delgado, and R. Blatt, *Quantum computations on a topologically encoded qubit,* Science **345**, 302 (2014).

[6] A. D. Córcoles, E. Magesan, S. J. Srinivasan, A. W. Cross, M. Steffen, J. M. Gambetta, and J. M. Chow, *Demonstration of a quantum error detection code using a square lattice of four superconducting qubits,* Nature Communications **6** (2015), 10.1038/ncomms7979.

[7] N. M. Linke, M. Gutierrez, K. A. Landsman, C. Figgatt, S. Debnath, K. R. Brown, and C. Monroe, *Fault-tolerant quantum error detection,* Science Advances **3** (2017), 10.1126/sciadv.1701074.

[8] M. Takita, A. W. Cross, A. D. Córcoles, J. M. Chow, and J. M. Gambetta, *Experimental Demonstration of Fault-Tolerant State Preparation with Superconducting Qubits,* Phys. Rev. Lett. **119**, 180501 (2017).

[9] D. Gottesman, *Quantum fault tolerance in small experiments,* ArXiv e-prints (2016), arXiv: 1610.03507.

[10] P. W. Shor, *Fault-tolerant Quantum Computation,* in *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, FOCS '96 (IEEE Computer Society, Washington, DC, USA, 1996) pp. 56–65.

[11] P. Aliferis, D. Gottesman, and J. Preskill, *Quantum Accuracy Threshold for Concatenated Distance-3 Codes,* Quantum Info. Comput. **6**, 97 (2006).

[12] P. Aliferis, D. Gottesman, and J. Preskill, *Accuracy Threshold for Postselected Quantum Computation,* Quantum Info. Comput. **8**, 181 (2008).

[13] A. W. Cross, D. P. DiVincenzo, and B. M. Terhal, *A Comparative Code Study for Quantum Fault Tolerance,* Quantum Info. and Comput. **9**, 541 (2009).

[14] A. Paetznick and B. W. Reichardt, *Fault-tolerant ancilla preparation and noise threshold lower bounds for the 23-qubit Golay code,* Quantum Information & Computation 12:1034-1080 (2012) **12**, 1034 (2011), arXiv: 1106.2190v2.

[15] Y. Tomita and K. M. Svore, *Low-distance surface codes under realistic quantum noise,* Phys. Rev. A **90**, 062320 (2014).

[16] C. Chamberland, T. Jochym-O'Connor, and R. Laflamme, *Thresholds for Universal Concatenated Quantum Codes,* Phys. Rev. Lett. **117**, 010501 (2016).

[17] R. Chao and B. W. Reichardt, *Quantum error correction with only two extra qubits,* ArXiv e-prints (2017), arXiv: 1705.02329v1.

[18] R. Chao and B. W. Reichardt, *Fault-tolerant quantum computation with few qubits,* ArXiv e-prints (2017), arXiv: 1705.05365.

[19] R. Takagi, T. J. Yoder, and I. L. Chuang, *Error rates and resource overheads of encoded three-qubit gates,* Phys. Rev. A **96**, 042302 (2017).

[20] C. Chamberland, T. Jochym-O'Connor, and R. Laflamme, *Overhead analysis of universal concatenated quantum codes,* Phys. Rev. A **95**, 022313 (2017).

[21] C. Chamberland and M. E. Beverland, *Flag fault-tolerant error correction with arbitrary distance codes,* ArXiv e-prints (2017), arXiv: 1708.02246v2.

[22] *IBM Quantum Experience*.

[23] *Code and data of the experiments*.

[24] *Python SDK*.

[25] R. E. Walpole and R. H. Myers, *Probability and Statistics for Engineers and Scientists,* in *Probability and Statistics for Engineers and Scientists* (Macmillan USA, 1985).

3

QUANTUM ERROR CORRECTION WITH THE TORIC-GKP CODE

This chapter examines the performance of the single-mode Gottesman-Kitaev-Preskill (GKP) code and its concatenation with the toric code for a noise model of Gaussian shifts, or displacement errors. It is shown how one can optimize the tracking of errors in repeated noisy error correction for the GKP code. This is done by examining the maximum-likelihood problem for this setting and its mapping onto a 1D Euclidean path-integral modeling a particle in a random cosine potential. The efficiency of a minimum-energy decoding strategy as a proxy for the path integral evaluation is demonstrated. In the second part of this chapter, the concatenation of the GKP code with the toric code is analyzed and numerically assessed. When toric code measurements and GKP error correction measurements are perfect, using GKP error information the toric code threshold improves from 10% *to* 14%. *When only the GKP error correction measurements are perfect a threshold at* 6% *is observed.*

In the more realistic setting when all error information is noisy, it is shown how to represent the maximum likelihood decoding problem for the toric-GKP code as a 3D compact QED model in the presence of a quenched random gauge field, an extension of the random-plaquette gauge model for the toric code. A new decoder for this problem is presented which shows the existence of a noise threshold at shift-error standard deviation $\sigma_0 \approx 0.243$ *for toric code measurements, data errors and GKP ancilla errors. If the errors only come from having imperfect GKP states, this corresponds to states with just 4 photons or more.*

As a last result, a no-go result for linear oscillator codes encoding oscillators into oscillators is shown. For the Gaussian displacement error model, this proves that encoding corresponds to squeezing the shift errors. This shows that linear oscillator codes are useless for quantum information protection against Gaussian shift errors.

This chapter has been published in Physical Review A **99**, 032344 (2019), [1].

3.1. INTRODUCTION

WITHIN the framework of oscillator or continuous-variable (CV) error correcting codes, one can distinguish mainly two classes of codes. One class generalizes qudit stabilizer codes to encode continuous degrees of freedom into a (larger) CV system [2, 3]. We refer to these codes as *linear oscillator codes*. The other class, first introduced by Gottesman, Preskill and Kitaev (GKP) in Ref. [4], and recently expanded to include many more codes [5, 6], encodes a discrete (finite-dimensional) system into a CV system. Encoding and decoding for the first class of codes falls within the framework of Gaussian quantum information [7], while the second class of codes requires using non-Gaussian states.

In this chapter we propose and analyze a scalable use of the GKP code [4] which encodes a single qubit into an oscillator. An example of such an oscillator is a mode in a high-Q microwave superconducting cavity coupled to superconducting qubits in a circuit-QED set-up. Proposals for preparing a GKP code state in such systems exist [8]. The CNOT gate between two GKP qubits requires about 4 dB of squeezing in both modes and a beam-splitter (see, e.g., Ref. [8]). Such a beam-splitter has been recently implemented between high-Q microwave cavity modes in Ref. [9]. Other possible physical implementations for the GKP code are the motional mode of a trapped-ion qubit [10] or atomic ensembles [11] for measurement-based CV cluster computation [12].

A bosonic code such as the GKP code or the recently-implemented cat code [13] might be used to get a high-quality qubit, but the code does not provide a means to drive error rates down arbitrarily. A *scalable* fault-tolerant architecture can possibly be obtained by concatenating the GKP code with a qubit stabilizer code such as the toric or surface code. A theoretic goal is then to understand how to decode such a toric-GKP code and what is the error threshold of the architecture. Some results on using “analog" error information in concatenating the GKP code with a stabilizer code were obtained in Refs. [14–16]. A concatenation of the GKP code with the surface code was analyzed in Ref. [17] in the channel setting for 2D and 3D surface codes by message-passing (perfect) GKP error information to the surface code decoder. However, this study did not look at error correction when the GKP syndrome is measured inaccurately. Previous work has also studied the performance of the GKP code in comparison with other bosonic codes in a photon loss channel setting, not taking into account the imperfections or processing of repeated rounds of error correction [6]. Other work focused on the effect of photon loss and other sources of error on the preparation of code states [18]. Besides its good performance compared to other bosonic codes, the GKP code is appealing since Clifford gates on the code states use only linear optical elements (including squeezing) [4].

In this chapter we first analyze repeated fault-tolerant quantum error correction for a single GKP qubit, see Section 3.3. Our noise model in this analysis includes errors both on the GKP qubit as well as on the GKP ancilla qubit used in the error correction. We show how decoding this continuous error information in discrete time steps maps onto the evaluation of a stochastic discrete-time Euclidean path integral. We present an efficient minimum-energy decoder which chooses the path which approximately corresponds to a classical trajectory in a disordered potential.

Second, we consider the toric-GKP code in Section 3.4. Assuming that both GKP error correction and toric code correction are noiseless, we show how the use of continuous GKP error information improves the error correction for the toric code (Section 3.4.2).

These results are in correspondence with the previous results in Ref. [17] although our likelihood function is not identical to the one in Ref. [17].

In Section 3.5 we formulate the decoding problem of repeated quantum error correction with the toric-GKP code where both the GKP syndrome and the toric code syndrome contain errors. Since errors on the GKP qubits are intrinsic (getting perfect code states with infinite numbers of photons is unphysical), this is the physically relevant setting. The maximum-likelihood formulation is in terms of a 3D gauge field model with quenched randomness determined by the errors (Section 3.5.2). Then in Section 3.5.3, we show how to re-express this model as a random plaquette gauge model (RPGM) with a $\mathbb{Z}_2$-field coupled to an auxiliary $U(1)$-gauge field. We then use this model to design a computationally-efficient decoder and present numerical results.

Finally, in Section 3.6, we present our general no-go result for the first class of codes, namely the linear oscillator codes. This no-go result is presented as the calculation of the probability distribution of logical errors on the encoded information after perfect maximum-likelihood decoding. The result is in accordance with, but does not directly follow from previous no-go results on Gaussian quantum information in Ref. [19]. The theorem explicitly shows that there are no linear oscillator code families of interest: there is no threshold in σ_0 below which protection of the encoded oscillators against shift errors gets better with increasing code size and the logical noise model is still Gaussian with the same σ_0, and possibly some squeezing of the logical quadratures.

The no-go result also shows that the existence of a threshold for the toric-GKP code is non-trivial. A sufficiently large departure from Gaussian quantum information is necessary to stabilize quantum information. In circuit-QED this departure comes exclusively from the use of the non-linear Josephson junction element.

3.2. GENERAL CONSIDERATIONS

3.2.1. DEFINITIONS AND NOTATIONS

We consider n-mode oscillator codes, which are subspaces in the n-mode Hilbert space $L_2(\mathbb{R}^n)$. Such a Hilbert space can be constructed as a tensor product of n single-particle Hilbert spaces $L_2(\mathbb{R})$ of complex square-integrable functions. It supports n pairs of canonically conjugated coordinate and momentum operators, $\hat{p}_k$ and $\hat{q}_k$, such that $[\hat{q}_k, \hat{p}_l] = i\delta_{kl}$. These operators are used to define the multi-mode exponential shift operators,

$$U(\boldsymbol{e}) \equiv \prod_{k=1}^{n} e^{iu_k\hat{p}_k + iv_k\hat{q}_k}, \quad \boldsymbol{e} \equiv (\boldsymbol{u}, \boldsymbol{v}), \tag{3.1}$$

where $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$ are n-component real vectors. It is easy to check that the product of two such operators satisfies

$$U(\boldsymbol{e})U(\boldsymbol{e}') = U(\boldsymbol{e}+\boldsymbol{e}')e^{i\omega(\boldsymbol{e},\boldsymbol{e}')}, \tag{3.2}$$

with the phase given by the symplectic product $\omega(\boldsymbol{e},\boldsymbol{e}') = \boldsymbol{u}\cdot\boldsymbol{v}' - \boldsymbol{v}\cdot\boldsymbol{u}'$. The set $\mathscr{H}_n$ of all such operators with arbitrary phases is closed under multiplication, it forms an irreducible representation of the Heisenberg group $\mathbb{H}^n$ acting on $L_2(\mathbb{R}^n)$. Just as for the n-qubit Hilbert space and the Pauli group $\mathscr{P}_n$, any operator acting on $L_2(\mathbb{R}^n)$ can be represented as a linear combination of elements of $\mathscr{H}_n$. Furthermore, the product (3.2) of two exponential operators, up to a phase, can be represented in terms of the sum of the

corresponding vectors, $\boldsymbol{e}'' = \boldsymbol{e} + \boldsymbol{e}'$. This map to $\mathbb{R}^{2n}$ is an analogue of the symplectic representation of $\mathscr{P}_n$ used in the theory of quantum codes.

An n-mode GKP code, $\mathscr{Q}$, is a CV stabilizer code defined in terms of an Abelian stabilizer group $\mathscr{S} \subset \mathscr{H}_n$ with elements in the form (3.1), such that $U(\mathbf{0}) \equiv \mathbb{1}$ is the only element in $\mathscr{S}$ proportional to the identity. Namely, the code $\mathscr{Q} \subseteq L_2(\mathbb{R}^n)$ is the common +1-eigenspace of all elements of $\mathscr{S}$,

$$\mathscr{Q} = \{|\psi\rangle \in L_2(\mathbb{R}^n) \mid S|\psi\rangle = |\psi\rangle\,,\ \forall S \in \mathscr{S}\}. \tag{3.3}$$

3

The structure of such Abelian subgroups and the implications for $\mathscr{Q}$ are described in Appendix 1.2.3. In the following, we will assume the representation of such a group in terms of some number r of its members chosen as generators, $\mathscr{S} = \langle S_1, \ldots, S_r\rangle$, $S_j \in \mathscr{H}_n$.

The formalism of qubit stabilizer codes [20, 21] carries over entirely to such CV stabilizer codes and errors from $\mathscr{H}_n$ play the special role played by Pauli errors in the qubit case. Given an error $E \in \mathscr{H}_n$, one can compute its syndrome, $\boldsymbol{q} \equiv \boldsymbol{q}(E)$, whose components are given by the extra phases in the commutation relations with the stabilizer generators, $ES_j = S_j E e^{iq_j}$. The set of errors which commute with all elements of the stabilizer group is called the centralizer $\mathrm{C}(\mathscr{S})$; these errors have a trivial syndrome, $\boldsymbol{q} = \mathbf{0}$. Of these, any error that is a member of the stabilizer group acts trivially on code states, while the remaining errors $L \in \mathrm{C}(\mathscr{S}) \setminus \mathscr{S}$ act non-trivially within the code, they are called *logical operators*. An error $E \in \mathscr{H}_n$ that does not commute with all stabilizer generators has a non-trivial syndrome and it takes $\mathscr{Q}$ into an orthogonal subspace $E\mathscr{Q} \equiv \{E|\psi\rangle \mid |\psi\rangle \in \mathscr{Q}\}$. Two errors that differ by an element of the stabilizer group, $E' = ES$, $S \in \mathscr{S}$, have the same syndrome, and as such, are called mutually *degenerate*. They act identically on the code and are also called *equivalent*. Two errors that differ by a logical operator, $E' = EL$, $L \in \mathrm{C}(\mathscr{S}) \setminus \mathscr{S}$, also have the same syndrome but they act differently on the code. The set of inequivalent logical operators $\mathrm{L}(\mathscr{S})$ is formed by the cosets of $\mathscr{S}$ in the centralizer $\mathrm{C}(\mathscr{S})$. If we ignore the phases, the set of cosets $\mathrm{L}(\mathscr{S})$ actually forms a group, the group of logical operators.

By a slight abuse of notation, and when the global phase is irrelevant, we will often refer to an operator $U(\boldsymbol{e}) \in \mathscr{H}_n$ directly by its symplectic vector component, $\boldsymbol{e} \in \mathbb{R}^{2n}$. For example, we can refer to a logical operator $\boldsymbol{c} \in \mathrm{L}(\mathscr{S})$ when we ought to write $U(\boldsymbol{c}) \in \mathrm{L}(\mathscr{S})$. In accordance with classical codes terminology, we also refer to $\boldsymbol{c} \in \mathrm{L}(\mathscr{S})$ as a *codeword*. Furthermore, for two equivalent errors, $\boldsymbol{e}$ and $\boldsymbol{e}' = \boldsymbol{e} + \boldsymbol{s}$, where $\boldsymbol{s} \in \mathscr{S}$, we use $\boldsymbol{e}' \simeq \boldsymbol{e}$ to denote their equivalence, and $[\boldsymbol{e}]$ to denote the entire equivalence class,

$$[\boldsymbol{e}] = \{\boldsymbol{e} + \boldsymbol{s} : \forall\, \boldsymbol{s} \in \mathscr{S}\}\,. \tag{3.4}$$

Throughout this work we consider the independent Gaussian displacement channel $\mathscr{N}(\rho)$ with standard deviation σ_0:

$$\mathscr{N}(\rho) = \int_{-\infty}^{\infty} \mathrm{d}u \int_{-\infty}^{\infty} \mathrm{d}v\, \mathbb{P}_{\sigma_0}(u)\mathbb{P}_{\sigma_0}(v) e^{iu\hat{p}+iv\hat{q}} \rho\, e^{-iu\hat{p}-iv\hat{q}}, \tag{3.5}$$

where ρ is a single-mode density matrix and $\mathbb{P}_{\sigma_0}(x)$ the Gaussian probability density function with mean zero and variance σ_0^2, i.e. $\mathbb{P}_{\sigma_0}(x) = (2\pi\sigma_0^2)^{-1/2} e^{-x^2/2\sigma_0^2}$. We will refer to σ_0 as the *bare* standard deviation, this is because we will often consider *scaled* observables, e.g. $\hat{P} = 2\alpha\hat{p}$ for which the corresponding effective rescaled standard deviation is

$\sigma = 2\alpha\sigma_0$. Even though this channel may not necessarily be the one which is physically most relevant, it is, like the Pauli error model, a convenient model which allows us to numerically and analytically model approximate GKP code states with finite levels of squeezing, see further motivation in Section 3.3.1.

As a convention we use bold italic symbols, such as $\boldsymbol{u}$, to denote row vectors. We use hatted symbols, such as $\hat{P}, \hat{Q}$, for quantum observables and un-hatted symbols for the corresponding eigenvalues, such as P and Q. We will consider modulo values for real numbers quite often where, for convention, we choose the remainder to be in a symmetrical interval around 0. For example, given $\phi \in \mathbb{R}$, writing $q = \phi \bmod 2\pi$ means that $q \in [-\pi, \pi)$ and $\phi = q + 2\pi k$ for some $k \in \mathbb{N}$. In conventional notation $q := (\phi + \pi) \bmod 2\pi - \pi$ and $k := \lfloor (\phi + \pi)/2\pi \rfloor$. We also denote a range of integers as $[n] \equiv \{1, \ldots, n\}$. We will refer to single-mode q-type errors as displacements of the form $\exp(i\eta\hat{q})$ for some η. Such errors induce shifts in p and are alternatively called shift-in-p errors. Similarly, for p-type errors which induce shifts in q.

3.2.2. MAXIMUM-LIKELIHOOD VS. MINIMUM-ENERGY DECODING

A (classical) binary linear code [22] of length n encoding k bits is a linear space of dimension k formed by binary strings of length n, $\mathscr{C} \subseteq \mathbb{F}_2^n$. For such a code, maximum likelihood (ML) syndrome-based decoding amounts to finding the most likely error which results in the given syndrome. Generally, there are 2^{n-k} distinct syndromes and $|\mathscr{C}| = 2^k$ codewords. It is not hard to find a vector $\boldsymbol{e}$ which produces the correct syndrome; ML decoding can then be done by comparing the probabilities of errors $\mathbb{P}(\boldsymbol{e} + \boldsymbol{c})$, where $\boldsymbol{c} \in \mathscr{C}$ goes over all the codewords. In the simplest case of the binary symmetric channel, the probabilities scale exponentially with error weight which can be thought of as the "energy" associated with the error. Thus, for linear binary codes under the binary symmetric channel, ML decoding is the same as the minimum-weight, or *minimum-energy* (ME) decoding.

Syndrome-based ML decoding for a qubit stabilizer code can be done similarly. The main difference here is the degeneracy: errors that differ by an element of the stabilizer group are equivalent, they can not and need not be distinguished. As a result, the probability $\mathbb{P}(E)$ of an n-qubit Pauli error $E \in \mathscr{P}_n$ needs to be replaced by the total probability to have any error equivalent to E. In the case of Pauli errors which are independent between different qubits, quite generally, this probability can be interpreted as a partition function of certain random-bond Ising model [23, 24]. Exactly which statistical model one gets, depends on the code. For a qubit square-lattice toric code with perfect stabilizer measurements the partition functions are those of 2D Random-Bond Ising model (RBIM). Similarly, for the toric code with repeated noisy measurements, the partition function is that of a random-plaquette gauge model (RPGM) in three dimensions [23], where the "time" dimension enumerates syndrome measurement cycles. More general models are discussed, e.g. in Refs. [24, 25].

Instead of computing the partition functions proportional to the total probabilities of errors in different sectors, one could try finding a single most-likely error compatible with the syndrome. It is the latter method that is usually called the ME decoding for a quantum code. Indeed, in terms of the statistical-mechanical analogy, for ML decoding one needs to minimize the free energy, minus the logarithm of the partition function. In

comparison, for ME decoding, one only looks at a minimum-energy configuration (not necessarily unique); this ignores any entropy associated with degenerate configurations. While the ME technique is strictly less accurate than ML decoding, in practice the difference can be small.

The two approaches are readily extended to GKP codes, both in the *channel model* where perfect stabilizer measurement is assumed, and in the more general *fault-tolerant* (FT) case where repeated measurements are used to offset the stabilizer measurement errors. The latter case can be interpreted in terms of a larger *space-time* code dealing with both the usual quantum errors and the measurement errors [23, 25, 26]. One important aspect is that the quantum errors accumulate over time, while measurement errors in different measurement rounds are independent from each other. This leads to an extended equivalence between combined data-syndrome errors which is similar to degeneracy. The corresponding generators can be constructed, e.g., by starting with a single-oscillator error, followed by measurement errors on all adjacent stabilizer checks that result in zero syndrome, followed by the error which exactly cancels the original error. Because of this cancellation, such an invisible error has no effect and should be counted as a part of the degeneracy group of the larger space-time code.

The following discussion applies to a GKP code in either the channel model or the fault-tolerant model. In both cases we denote as $\mathscr{S} \subseteq \mathscr{H}_n$ the degeneracy group of the code. In the channel model, $\mathscr{S}$ is exactly the stabilizer group, acting on the data oscillators. In the fault-tolerant case, $\mathscr{S}$ is the degeneracy group of the space-time code, acting on both data oscillators as well as ancillary oscillators used to measure the syndrome. Consider a multi-oscillator error, $\boldsymbol{e} \in \mathbb{R}^{2n}$, see Eq. (3.1), and the corresponding probability density $\mathbb{P}(\boldsymbol{e})$. The probability is assumed to have a sharp [exponential or Gaussian, cf. Eq. (3.5)] dependence on the components of $\boldsymbol{e}$; for this reason we can also write

$$\mathbb{P}(\boldsymbol{e}) = \exp\big(-H(\boldsymbol{e})\big), \tag{3.6}$$

where $H(\boldsymbol{e})$ is the dimensionless *energy* associated with the error operator $U(\boldsymbol{e})$. Syndrome-based ML decoding can be formulated as follows. The error $\boldsymbol{e}$ yields the syndrome $\boldsymbol{q}(\boldsymbol{e})$. Given a logical operator $\boldsymbol{c} \in \mathrm{L}(\mathscr{S})$, write as $\mathbb{P}([\boldsymbol{e}+\boldsymbol{c}]|\boldsymbol{q})$ the probability for any error in the class $[\boldsymbol{e}+\boldsymbol{c}]$, see (3.4), conditioned on the syndrome $\boldsymbol{q}$. This probability can be written as

$$\begin{aligned}\mathbb{P}([\boldsymbol{e}+\boldsymbol{c}]|\boldsymbol{q}) &= \int_{\boldsymbol{s}\in\mathscr{S}} \mathrm{d}\boldsymbol{s}\, \mathbb{P}(\boldsymbol{e}+\boldsymbol{c}+\boldsymbol{s}|\boldsymbol{q}) \\ &= \frac{1}{\mathbb{P}(\boldsymbol{q})} \int_{\boldsymbol{s}\in\mathscr{S}} \mathrm{d}\boldsymbol{s}\, \mathbb{P}(\boldsymbol{e}+\boldsymbol{s}+\boldsymbol{c}) \\ &= \frac{1}{\mathbb{P}(\boldsymbol{q})} \int_{\boldsymbol{s}\in\mathscr{S}} \mathrm{d}\boldsymbol{s}\, \mathrm{e}^{-H(\boldsymbol{e}+\boldsymbol{s}+\boldsymbol{c})} \equiv \frac{1}{\mathbb{P}(\boldsymbol{q})} Z_{\boldsymbol{c}}(\boldsymbol{e}),\end{aligned} \tag{3.7}$$

where an appropriate integration measure should be used, and $\mathbb{P}(\boldsymbol{q})$ is the net probability density to obtain the syndrome $\boldsymbol{q} = \boldsymbol{q}(\boldsymbol{e})$. Among all inequivalent codewords $\boldsymbol{c} \in \mathrm{L}(\mathscr{S})$, we select the most likely, i.e., with the largest $Z_{\boldsymbol{c}}(\boldsymbol{e})$. The probability of leaving a logical error $\boldsymbol{c}$ after ML decoding is the net probability of all the errors $\boldsymbol{e}$ for which the sector $[\boldsymbol{e}-\boldsymbol{c}]$ is the most likely, so

$$\mathbb{P}^{(\mathrm{ML})}(\boldsymbol{c}) = \int_{\boldsymbol{e}:\, \forall \boldsymbol{b}\neq(-\boldsymbol{c}),\, Z_{-\boldsymbol{c}}(\boldsymbol{e})>Z_{\boldsymbol{b}}(\boldsymbol{e})} \mathrm{d}\boldsymbol{e}\, \mathbb{P}(\boldsymbol{e}), \tag{3.8}$$

[here we disregarded the contribution from sectors $\boldsymbol{b} \neq (-\boldsymbol{c})$ equiprobable with $(-\boldsymbol{c})$, $Z_{-\boldsymbol{c}}(\boldsymbol{e}) = Z_{\boldsymbol{b}}(\boldsymbol{e})$]. The probability of success of ML decoding can then be expressed as $\mathbb{P}_{\text{succ}}^{(\text{ML})} = \mathbb{P}^{(\text{ML})}(\mathbf{0})$. It is easy to see that any other decoding algorithm gives success probability that is not higher than that of ML decoding. Indeed, a different algorithm would swap some errors $\boldsymbol{e}$ for $\boldsymbol{e}+\boldsymbol{c}$, which may reduce the measure in the corresponding analog of Eq. (3.8).

Furthermore, given an error $\boldsymbol{e}$, the probability $\mathbb{P}(\boldsymbol{q})$ to obtain the syndrome $\boldsymbol{q} = \boldsymbol{q}(\boldsymbol{e})$ can be written as $\mathbb{P}(\boldsymbol{q}) = \int_{\boldsymbol{b}\in \mathrm{L}(\mathscr{S})} \mathrm{d}\boldsymbol{b}\, Z_{\boldsymbol{b}}(\boldsymbol{e})$, using the appropriate integration measure for the logical operators $\boldsymbol{b}$. For this error $\boldsymbol{e}$ we write $\boldsymbol{c}_{\max}(\boldsymbol{e})$ for its corresponding most likely sector,

$$\boldsymbol{c}_{\max}(\boldsymbol{e}) = \underset{\boldsymbol{c}\in \mathrm{L}(\mathscr{S})}{\arg\max}\, Z_{\boldsymbol{c}}(\boldsymbol{e}). \tag{3.9}$$

The probability of a logical error $\boldsymbol{c}$ after ML decoding (3.8) can then be rewritten as an expectation by multiplying and dividing by $\mathbb{P}(\boldsymbol{q})$, changing variables, and re-summing, resulting in

$$\mathbb{P}^{(\text{ML})}(\boldsymbol{c}) = \Big\langle \mathbb{P}\big([\boldsymbol{e}+\boldsymbol{c}_{\max}+\boldsymbol{c}]|\boldsymbol{q}(\boldsymbol{e})\big)\Big\rangle = \int \mathrm{d}\boldsymbol{e}\, \mathbb{P}(\boldsymbol{e})\, \frac{Z_{\boldsymbol{c}_{\max}+\boldsymbol{c}}(\boldsymbol{e})}{\mathbb{P}(\boldsymbol{q})} = \int \mathrm{d}\boldsymbol{e}\, \mathbb{P}(\boldsymbol{e})\, \frac{Z_{\boldsymbol{c}_{\max}+\boldsymbol{c}}(\boldsymbol{e})}{\int \mathrm{d}\boldsymbol{b}\, Z_{\boldsymbol{b}}(\boldsymbol{e})}. \tag{3.10}$$

ML decoding is successful if the most likely error is actually the one that happened, which corresponds to the trivial sector $\boldsymbol{c} \simeq \mathbf{0}$ being dominant over all other sectors $\mathbf{0} \neq \boldsymbol{c} \in \mathrm{L}(\mathscr{S})$. Given the error probability distribution $\mathbb{P}(\boldsymbol{e})$, we say that a sequence of discrete GKP codes of increasing length n is in the *decodable phase* if $\mathbb{P}_{\text{succ}}^{(\text{ML})} \equiv \mathbb{P}^{(\text{ML})}(\mathbf{0}) \to 1$ with $n \to \infty$.

With the definitions (3.6) and (3.7), $Z_{\boldsymbol{c}_{\max}}(\boldsymbol{e})$, can be interpreted as a partition function of a classical model in the presence of quenched randomness determined by the actual error $\boldsymbol{e}$. The partition function $Z_{\boldsymbol{c}_{\max}+\boldsymbol{c}}(\boldsymbol{e})$ differs by an addition of a defect, e.g. a homologically non-trivial domain wall at the locations specified by non-zero components of the codeword $\boldsymbol{c}$. Having already $\boldsymbol{c}_{\max}(\boldsymbol{e}) \neq \mathbf{0}$ means that the disorder, $\boldsymbol{e}$, energetically favors the domain wall $\boldsymbol{c}_{\max}$. In the following, we will also consider the free energy, $F_{\boldsymbol{c}}(\boldsymbol{e})$,

$$F_{\boldsymbol{c}}(\boldsymbol{e}) \equiv -\ln Z_{\boldsymbol{c}_{\max}+\boldsymbol{c}}(\boldsymbol{e}), \tag{3.11}$$

as well as the corresponding average $\langle F_{\boldsymbol{c}}\rangle \equiv \int \mathrm{d}\boldsymbol{e}\, \mathbb{P}(\boldsymbol{e})\, F_{\boldsymbol{c}}(\boldsymbol{e})$. It follows from the Gibbs inequality that below the error-correction threshold for the noise parameters in $\mathbb{P}(\boldsymbol{e})$, the free energy increment $\Delta F_{\boldsymbol{c}} \equiv F_{\boldsymbol{c}}(\boldsymbol{e}) - F_{\mathbf{0}}(\boldsymbol{e})$ associated with a logically-distinct "incorrect" class ($\boldsymbol{c} \neq \mathbf{0}$) necessarily diverges with n for any error, $\boldsymbol{e}$, likely to happen [24]. More precisely, if ML decoding is asymptotically successful with probability one, $\mathbb{P}_{\text{succ}}^{(\text{ML})} \to 1$, the average free energy increment, $\langle \Delta F_{\boldsymbol{c}}\rangle$ associated with any non-trivial codeword $\boldsymbol{c} \neq \mathbf{0}$ must diverge for $n \to \infty$. Such a divergence can be seen as a signature of a phase transition in the corresponding model.

As is the case for the surface codes [23], the partition functions $Z_{\boldsymbol{c}}(\boldsymbol{e})$ are evaluated at a temperature that is not a free parameter but depends on the distribution $\mathbb{P}(\boldsymbol{e})$. For the sake of understanding the physics of the corresponding models, we could relax this, e.g. by additionally rescaling the energy $H(\boldsymbol{e}) \to \beta H(\boldsymbol{e})$, cf. Eq. (3.6), in the definition (3.7) of the partition function $Z_{\boldsymbol{c}}(\boldsymbol{e})$, while keeping the original error probability distribution in

the average (3.10). This amounts to using ML decoder with incorrect input information, thus the corresponding success probability is not expected to increase,

$$\mathbb{P}_{\text{succ}}^{(\text{ML})}(\beta) \leq \mathbb{P}_{\text{succ}}^{(\text{ML})}(\beta = 1) \equiv \mathbb{P}_{\text{succ}}^{(\text{ML})}, \tag{3.12}$$

similar to decoding away from the Nishimori line in the case of qubit stabilizer code [23, 24]. In particular, the limit $\beta \to \infty$ corresponds to ME decoding, where we are choosing the codeword $\boldsymbol{c}$ to minimize the function

$$H_{\boldsymbol{c}}(\boldsymbol{e}) \equiv \min_{\boldsymbol{s} \in \mathscr{S}} H(\boldsymbol{e} + \boldsymbol{c} + \boldsymbol{s}). \tag{3.13}$$

3.3. PROTECTING A SINGLE GKP QUBIT

3.3.1. SET-UP

The single-mode GKP code [4] is a prescription to encode a qubit —a two-dimensional Hilbert space— into the Hilbert space of an oscillator using a discrete subgroup of displacement operators $\mathscr{H}_1$ as the stabilizer group. One chooses the two commuting displacement operators, $S_p = e^{2i\alpha\hat{p}}$ and $S_q = e^{i2\pi\hat{q}/\alpha}$, where $\alpha \neq 0$ is any real number. For this encoded qubit the (logical) Pauli operators are $Z = e^{i\pi\hat{q}/\alpha}$ (with $Z^2 = S_q$) and $X = e^{i\alpha\hat{p}}$ (with $X^2 = S_p$). One can verify that $XZ = -ZX$. The oscillator observables, $\hat{P} = 2\alpha\hat{p}$ and $\hat{Q} = 2\pi\hat{q}/\alpha$, can both take any value $2\pi k$ for $k \in \mathbb{Z}$ on ideal codewords. The codeword $|\overline{0}\rangle$ (respectively $|\overline{1}\rangle$) is distinguished by k being even (respectively odd). The action of phase space translations S_p (respectively S_q) on the eigenvalues of $\hat{Q}$ (respectively $\hat{P}$) is $Q \to Q + 4\pi$ (respectively $P \to P + 4\pi$). The action of X (respectively Z) is $Q \to Q + 2\pi$ (respectively $P \to P + 2\pi$).

A visual representation can be obtained by imagining the variables Q and P as a single torus in phase space with both handles of circumference 2π. In this representation S_p lets Q wind around the handle exactly twice, while X lets Q go around the handle exactly once. A correctable error constitutes a shift in Q by less than half the circumference. In this convenient representation, a logical error thus occurs when the winding number is odd, and no error occurs when the winding number is even. The shifts in P correspond to windings around the other handle of the torus.

We will assume that the oscillator undergoes noise modelled as a Gaussian displacement channel with *bare* standard deviation σ_0, see Eq. (3.5). The effect on the scaled observables $\hat{P}$ and $\hat{Q}$ is to map $P \to P + \epsilon_p$ and $Q \to Q + \epsilon_q$ where ϵ_p and ϵ_q are drawn from Gaussian distributions with *rescaled* variances

$$\sigma_P^2 = 4\alpha^2\sigma_0^2 \text{ and } \sigma_Q^2 = \frac{4\pi^2\sigma_0^2}{\alpha^2}. \tag{3.14}$$

For symmetry reasons, α is chosen to be $\sqrt{\pi}$ and we write $\sigma = \sigma_P = \sigma_Q$. Given perfect measurements of S_p, the error ϵ_q can be corrected if $\left|\epsilon_q \bmod 4\pi\right| < \pi$.

In order to measure stabilizer generators S_p and S_q we consider the fault-tolerant Steane measurement circuits [4] in Fig. 3.1, where encoded $|\overline{+}\rangle$ or $|\overline{0}\rangle$ ancillas, CNOTs and $\hat{q}$ or $\hat{p}$ measurements are used.

For simplicity, we consider the ancilla preparations, the CNOT and the $\hat{q}$ and $\hat{p}$ measurements to be perfect and only add Gaussian displacement channels on the data qubit

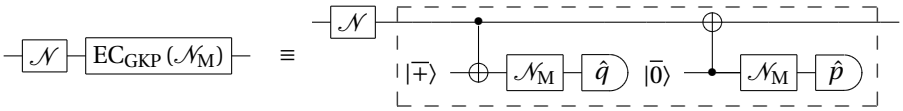


Figure 3.1: A single round of fault-tolerant GKP syndrome measurement for both q and p shifts. Here $|\overline{+}\rangle$ is the $+1$ eigenstate of S_q and X, and $|\overline{0}\rangle$ is the $+1$ eigenstate of S_p and Z. The CNOT gate is the logical CNOT for the GKP code which induces the transformation $q_{\text{target}} \to q_{\text{control}} + q_{\text{target}}$ (while $p_{\text{control}} \to p_{\text{control}} - p_{\text{target}}, q_{\text{control}} \to q_{\text{control}}, p_{\text{target}} \to p_{\text{target}}$). Each measurement is a perfect homodyne measurement of $\hat{q}$ or $\hat{p}$. $\mathcal{N}$ are $\mathcal{N}_{\mathrm{M}}$ are Gaussian displacement channels in Eq. (3.5) which model shift errors on the encoded state in each round of error correction, respectively shift errors in the homodyne measurement.

3

and on the ancilla qubit right before its measurement. Doing this ignores the back propagation of q-type errors to the data due to an imperfect ancilla [27], but if we treat p-type and q-type error correction independently, then this back-propagation does not fundamentally alter the noise model. We will keep the freedom of choosing different standard deviations for the data and the ancilla errors and denote as σ_{M} the *scaled* standard deviation for the ancilla errors.

What is important is that our error model covers dominant sources of imperfections stochastically. Any physically-realistic GKP code state has finite photon number $\overline{n}$ and one reasonable model of such finite-photon GKP state is a coherent superposition of Gaussian displacement errors on a perfect code state, see Eqs. (40),(41) in Ref. [4]. The quality of such an approximate GKP state can be given by an effective squeezing parameter Δ with $\overline{n} \sim \frac{1}{2\Delta^2} - \frac{1}{2}$. Assuming a coherent superposition of Gaussian displacements can be replaced by a Gaussian mixture of displacements on a perfect state we can identify $\Delta^2 = 2\sigma_0^2$. If errors are dominated by such a finite squeezing/finite photon number, we could use $\sigma_0 \sim \frac{1}{\sqrt{2(2\overline{n}+1)}}$ to interpret our numerical data. For example, $\overline{n} = 4$ gives $\sigma_0 \approx 0.236$.

Besides this, photon loss is usually the main source of imperfection and it has been shown that photon loss with rate γ followed by an amplification or pumping step produces the Gaussian displacement channel with $\sigma_0^2 = \frac{\gamma}{1-\gamma}$ [6]. For example, the rate $\gamma = 0.02$ corresponds to $\sigma_0 = 0.14$. Such an amplification step on the GKP data qubit could be added in each step of error correction.

Since the measurement outcomes in Fig. 3.1 are inaccurate, they cannot be used to infer a correction which maps the state back to the code space. In order to perform error correction one has to measure frequently and try to use the record of measurements to stay as close as possible to the code space without incurring logical errors to preserve the codeword. Figure 3.2 shows this repeated measurement protocol for p-type errors (or shift-in-q errors).

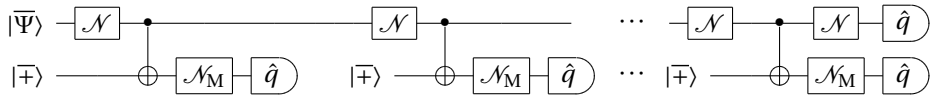


Figure 3.2: Repeated rounds of error correction for the GKP code to detect and keep track of error shifts in $\hat{q}$ followed by a final destructive measurement of the data (modeled as $\mathcal{N}$ followed by a perfect measurement of $\hat{q}$). No explicit corrections based on the measured values are shown.

We will analyse only p-type data errors with scaled standard deviation σ and measurement errors with scaled standard deviation σ_{M}, cf. Eq. (3.14). The analysis for q-type errors would be similar. When considering the realization of a particular shift error we will use the following notation: $\epsilon_t \in \mathbb{R}$ is the shift error occurring on the data before the t^{th} measurement, $\delta_t \in \mathbb{R}$ is the measurement error occuring at the t^{th} step. Furthermore, $q_t \in [-\pi,\pi)$ is the t^{th} measurement outcome for the rescaled variable $\hat{Q}$ and $\phi_t = \epsilon_1 + \cdots + \epsilon_t$ is the cumulative shift on the data. The relations between these quantities are

$$q_t = \phi_t + \delta_t \bmod 2\pi, \qquad \phi_t - \phi_{t-1} = \epsilon_t, \qquad \phi_0 \equiv 0. \tag{3.15}$$

We consider a total of M rounds of GKP measurements indexed by $t \in [M]$. Of these, the last measurement is assumed perfect, incorporating any measurement error into the corresponding shift error. Specifically, we write $\phi_M = \phi_{M-1} + \epsilon_M$, $q_M = \phi_M \bmod 2\pi$, so that $\delta_M = 0$. This last measurement can be thought of as a destructive measurement performed directly on the data without the use of an ancilla, as one would do to retrieve the encoded information. As such, the last data error ϵ_M can equivalently be thought of as the last measurement error on the destructive measurement of the data. Having this last perfect measurement permits to map back to the code space and easily define successful or failed error correction. Specifically, we are trying to determine the parity of k_M in the relation $q_M = \phi_M + 2\pi k_M$; error correction is successful as long as we determined the parity correctly. We denote the set of M measurements as $\boldsymbol{q}$ and M cumulative shift errors as $\boldsymbol{\phi}$.

To get some intuition, imagine that we apply a single round of error correction of Fig. 3.2 and $\mathcal{N}_{\mathrm{M}}$ is the identity channel. The ancilla qubit $|\overline{+}\rangle$ is a uniform sum of delta functions with $Q = q = 0 \bmod 2\pi$, hence we represent the measurement outcome compactly as $q \in [-\pi,\pi)$. An incoming logical X on the data qubit is pushed (through the CNOT) onto the ancilla qubit where it translates q by a full 2π-period, hence logical information is not observed. One corrects a shift of up to π (at most half-a-logical) by shifting Q back by the least amount to make it again equal to 0 $\bmod 2\pi$.

3.3.2. DECODING STRATEGIES

We start by describing the maximum-likelihood strategy. Given the measurement record, one would like to compute the conditional probabilities for different classes of errors which are distinguished by their logical action. In this case of correcting a single qubit against shift errors in q, one has to decide whether there was an X error or there was none. Knowing the details of the error model, namely σ and σ_{M}, one can write down the probability of these two classes. Formally, they are given by

$$\mathbb{P}(0|\boldsymbol{q}) = \int_{I_0} \mathbb{P}(\boldsymbol{\phi}|\boldsymbol{q})\mathrm{d}\boldsymbol{\phi}, \qquad \mathbb{P}(1|\boldsymbol{q}) = \int_{I_1} \mathbb{P}(\boldsymbol{\phi}|\boldsymbol{q})\mathrm{d}\boldsymbol{\phi}, \tag{3.16}$$

where the integration covers all possible realizations of the shift errors described by $\boldsymbol{\phi}$, and I_0 (respectively, I_1) limits the integral to realizations leaving no X error (respectively, leaving an X error). Since the last measurement is assumed perfect, I_0 and I_1 are characterized by $\delta\left(\phi_M - q_M + 2\pi k_M\right)$, with any even k_M in I_0 and any odd k_M for I_1.

In practice, to do decoding for the given measurement history, $\boldsymbol{q}$, one needs to compare the probabilities (3.16). ML decoding algorithm suggests that a logical X correction

is needed if $\mathbb{P}(1|\boldsymbol{q}) > \mathbb{P}(0|\boldsymbol{q})$. Of course, this does not guarantee success in each particular trial. If we take just one measurement round, $M = 1$, which corresponds to measuring the data directly, we get

$$\begin{aligned}\mathbb{P}_{M=1}(0|q_1) &= \sum_{k_1 \text{even}} \int_{-\infty}^{\infty} d\epsilon_1 \, \mathbb{P}(\epsilon_1|q_1)\delta(\epsilon_1 - q_1 + 2k_1\pi) \\ &\propto \sum_{k_1 \text{ even}} \mathbb{P}_\sigma(q_1 - 2\pi k_1),\end{aligned} \tag{3.17}$$

and $\mathbb{P}_{M=1}(1|q_1)$ is given by the complementary sum over odd k_1, which makes the normalization the full sum with k_1 running over all integer values.

To compute these probabilities in general, we apply Bayes' rule:

$$\mathbb{P}(\boldsymbol{\phi}|\boldsymbol{q}) = \frac{\mathbb{P}(\boldsymbol{q}|\boldsymbol{\phi})\mathbb{P}(\boldsymbol{\phi})}{\mathbb{P}(\boldsymbol{q})}. \tag{3.18}$$

Then the probability for some outcome $\boldsymbol{q}$ given data errors $\boldsymbol{\phi}$ can be computed from the measurement error model and the probability for some data error $\boldsymbol{\phi}$ from the data error model. The normalization, $\mathbb{P}(\boldsymbol{q})$, can be computed by integrating the numerator over every $\boldsymbol{\phi}$. Using Eq. (3.15), we have from Eq. (3.18)

$$\begin{aligned}&\mathbb{P}(\boldsymbol{\phi}|\boldsymbol{q})\mathbb{P}(\boldsymbol{q}) \\ &\propto \sum_{\boldsymbol{k}\in\mathbb{Z}^M} \left[\prod_{t=1}^{M-1} \exp\left(-\frac{\left(q_t - \phi_t + 2\pi k_t\right)^2}{2\sigma_{\mathrm{M}}^2}\right) \prod_{t=1}^{M} \exp\left(-\frac{\left(\phi_t - \phi_{t-1}\right)^2}{2\sigma^2}\right) \delta(\phi_M - q_M - 2\pi k_M)\right].\end{aligned} \tag{3.19}$$

Recalling Eq. (3.7), we write the corresponding complementary probabilities (3.16) in terms of partition functions,

$$\mathbb{P}(0|\boldsymbol{q}) = \frac{Z_0(\boldsymbol{q})}{\mathbb{P}(\boldsymbol{q})}, \quad \mathbb{P}(1|\boldsymbol{q}) = \frac{Z_1(\boldsymbol{q})}{\mathbb{P}(\boldsymbol{q})}, \tag{3.20}$$

$$\begin{aligned}Z_c(\boldsymbol{q}) = N^{-1} \int \mathrm{d}\boldsymbol{\phi} \sum_{\boldsymbol{k}\in\mathbb{Z}^{M-1}} &\exp\left(-\sum_{t=1}^{M-1} \frac{\left(q_t - \phi_t + 2\pi k_t\right)^2}{2\sigma_{\mathrm{M}}^2}\right) \exp\left(-\sum_{t=1}^{M} \frac{\left(\phi_t - \phi_{t-1}\right)^2}{2\sigma^2}\right) \\ &\times \sum_{k_M\in\mathbb{Z}} \delta(\phi_M - q_M - 2\pi c - 4\pi k_M), \quad c = 0, 1,\end{aligned} \tag{3.21}$$

with N a normalization constant [1]. In this special case picking $\boldsymbol{e} = \boldsymbol{q}$, for a candidate error is always a valid choice, that is why we can write directly $Z_c(\boldsymbol{q})$.

To compute it, denote as B the symmetric matrix with the components

$$B_{ij} \equiv \frac{1}{2} \frac{\partial^2}{\partial\phi_i \partial\phi_j} R(\boldsymbol{\phi}, \boldsymbol{0}), \quad i, j \in [M-1], \tag{3.22}$$

associated with the first $M-1$ variables ϕ_t in the quadratic form

$$R(\boldsymbol{\phi}, \boldsymbol{q}) = \frac{1}{\sigma_{\mathrm{M}}^2} \sum_{t=1}^{M-1} \left(q_t - \phi_t\right)^2 + \frac{1}{\sigma^2} \sum_{t=1}^{M} \left(\phi_t - \phi_{t-1}\right)^2 \Big|_{\phi_M = q_M}$$

[1] Strictly speaking the normalization constant, N, diverges due to $\mathbb{P}(\boldsymbol{q})$ being an infinite sum of delta peaks. In practice we always compute the quantities $Z_c(\boldsymbol{q})$ with a cutoff and can adjust N such that $1 = Z_0(\boldsymbol{q}) + Z_1(\boldsymbol{q})$.

in the exponent in the integrand of Eq. (3.21). Collecting the remaining terms and completing the square we obtain an M-variable quadratic form $\boldsymbol{q}A\boldsymbol{q}^{\mathrm{T}}$ with the block matrix

$$A = \left(\begin{array}{c|c} \tilde{A} & \boldsymbol{c}^{\mathrm{T}} \\ \hline \boldsymbol{c} & b \end{array}\right), \tag{3.23}$$

expressed in terms of the $(M-1)\times(M-1)$ matrix $\tilde{A}$, row vector $\boldsymbol{c}$, and a scalar b:

$$\tilde{A} = \frac{1}{\sigma_{\mathrm{M}}^2}\mathbb{1} - \frac{1}{\sigma_{\mathrm{M}}^4}B^{-1}, \quad \boldsymbol{c}_i = -\frac{1}{\sigma^2\sigma_{\mathrm{M}}^2}\left[B^{-1}\right]_{M-1,i}, \quad b = \frac{1}{\sigma^2} - \frac{1}{\sigma_{\mathrm{M}}^4}\left[B^{-1}\right]_{M-1,M-1}. \tag{3.24}$$

The evaluation of the Gaussian integrals in Eq. (3.21) therefore gives

$$Z_c(\boldsymbol{q}) = \frac{(2\pi)^{(M-1)/2}}{N(\det B)^{1/2}} \sum_{\boldsymbol{k}=(\tilde{\boldsymbol{k}},c+2m):\, \tilde{\boldsymbol{k}}\in\mathbb{Z}^{M-1},\, m\in\mathbb{Z}} \exp\left(-\frac{1}{2}\left(\boldsymbol{q}+2\pi\boldsymbol{k}\right)A\left(\boldsymbol{q}+2\pi\boldsymbol{k}\right)^{\mathrm{T}}\right), \tag{3.25}$$

where B and A are the symmetric positive-definite matrices given explicitly by Eqs. (3.22) and (3.23). The sums with $c\in\{0,1\}$ can be numerically computed by setting a cut-off K, restricting every k_t to the interval $-K\le k_t\le K$. The number of terms then still exponentially increases with the number of rounds. We used Eq. (3.25) with a cut-off $K=2$ for up to $M=7$ rounds for the results shown in Fig. 3.5 and Fig. 3.6. Intuitively, this cut-off corresponds to only considering events where the measurement shift errors let one wind around the torus at most twice in each round. This is pretty reasonable since these errors follow a Gaussian distribution with small variance.

Generally, a more clever way to calculate the sum in Eq. (3.25) is to express it in terms of a genus-M Riemann theta function [28], and then transform the matrix so that the summation terms can be rearranged in decreasing order, stopping at a desired precision. However, this requires solving a shortest vector problem with the eigenvectors of A and is therefore also conjectured to be computationally difficult [29, 30].

In addition to the formally exact but hard to calculate expressions (3.21), (3.25) for the conditional probabilities, we would like to consider a class of approximate minimum-energy solutions of the corresponding optimization problem. To this end, we define a 2π-periodic potential, $V_\sigma(x) = V_\sigma(x+2\pi)$,

$$\exp\big(-V_\sigma(x)\big) \equiv \sum_{k\in\mathbb{Z}} e^{-(x+2\pi k)^2/2\sigma^2}. \tag{3.26}$$

The periodicity of the sum of the Gaussians implies that one should be able to approximate $V_\sigma(x)$ by its principal Fourier harmonic,

$$V_\sigma(x) \approx A_0 - \beta_V(\sigma)\cos x, \tag{3.27}$$

where β_V is Villain's effective inverse temperature parameter, and the overall shift A_0 is irrelevant. Such a simplified form is exactly the approximation used by Villain [31], but "in reverse." Indeed, for large β, one has [32]

$$e^{\beta\cos x} \approx e^{\beta}\sum_{k\in\mathbb{Z}} \exp\left(-\frac{\beta(x+2\pi k)^2}{2}\right), \tag{3.28}$$

which gives $\beta_V(\sigma) = 1/\sigma^2$, $\sigma \ll 1$.

With the defined periodic potential, the logarithm of the non-singular part of Eq. (3.19) acquires a form of a discrete-time Euclidean action, cf. Eq. (3.6),

$$H(\boldsymbol{\phi};\boldsymbol{q}) \equiv -\log[\mathbb{P}(\boldsymbol{q}|\boldsymbol{\phi})\mathbb{P}(\boldsymbol{\phi})] = \sum_{t=1}^{M} \frac{(\phi_t - \phi_{t-1})^2}{2\sigma^2} + \sum_{t=1}^{M-1} V_{\sigma_{\rm M}}(q_t - \phi_t) + \text{const.} \tag{3.29}$$

With the given values $\phi_0 = 0$ and ϕ_M, the corresponding extremum can be found by solving the equations

$$\phi_{t+1} - 2\phi_t + \phi_{t-1} + \sigma^2 V'_{\sigma_{\rm M}}(q_t - \phi_t) = 0, \quad t \in [M-1], \tag{3.30}$$

where $V'_{\sigma_{\rm M}}(x)$ denotes the derivative of the potential in Eq. (3.26). These equations can be readily solved one-by-one, starting with $\phi_0 = 0$ and some $\phi_1 \equiv \phi$; the boundary condition $\phi_M = q_M + 2\pi k_M$ can be satisfied by scanning over different values of ϕ_1 in a relatively small range around zero, with the global minimum subsequently found by comparing the resulting values of the sum in Eq. (3.29). Then, any even value of k_M corresponds to no logical error, while an odd k_M indicates an X error to be corrected. While such a minimization technique gives the exact ME solution, in practice it is rather slow. Namely, with increasing non-linearity $\sigma^2/\sigma_{\rm M}^2$ and increasing length M of the chain, a small change in ϕ_1 may strongly affect the configuration of the entire chain. Respectively, it is easy to miss an extremum corresponding to the global minimum. This numerical complexity of minimizing Eq. (3.29) is a manifestation of chaotic behavior inherent in the equations (3.30).

Indeed, the problem of minimizing the energy (3.29) can be interpreted as a disordered version of a generalized Frenkel-Kontorova (FK) model [33], where a chain of masses coupled by springs lies in a periodic potential. In our setup, random shifts q_t can be traded for randomness in the initial (unstretched) lengths of the springs, $q_t - q_{t-1}$. The original FK model, with $V_\sigma(x)$ replaced by a harmonic function, is obtained if one uses the Villain approximation "in reverse", see Eq. (3.27). Even in the absence of disorder, the FK model is an example of a minimization problem with multiple competing minima which can be extremely close in energy. The corresponding equations (3.30), viewed as a two-dimensional map $(\phi_{t-1}, \phi_t) \to (\phi_t, \phi_{t+1})$, are a version of the Chirikov-Taylor area-preserving map from a square of size 2π to itself [34], one of the canonical examples of emergent chaos.

For this reason, and also in an attempt to come up with a numerically efficient decoding algorithm, we have designed the following approximate *forward-minimization* technique. For each $t < M$, starting from $t = 1$, given the present value ϕ_{t-1}, one determines the next value ϕ_t such that $\frac{\partial H}{\partial \phi_t}\big|_{\phi_{t+1}=\phi_t} = 0$. Given the syndrome q_t, this implies

$$\phi_t = \arg\min_{\phi} \left[\frac{(\phi - \phi_{t-1})^2}{2} - \sigma^2 V_{\sigma_{\rm M}}(q_t - \phi) \right] \Rightarrow \phi_t = \sigma^2 V'_{\sigma_{\rm M}}(q_t - \phi_t) + \phi_{t-1}. \tag{3.31}$$

At the end, after one obtains ϕ_{M-1}, one chooses a k_M such that $q_M + 2\pi k_M$ is the closest to ϕ_{M-1}. The parity of thus chosen k_M then tells if a logical error happened. This strategy is illustrated in Fig. 3.3.

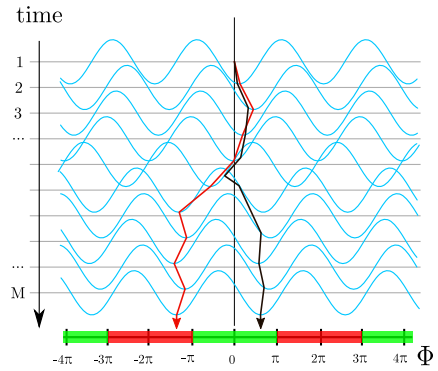


Figure 3.3: (Color online) Sketch of GKP decoding problem and its solution strategy. For each round of error correction $t \in [M]$ one has a Villain potential $V_{\sigma_M}(q_t - \phi_t)$, depicted in blue, with the minimum centered at the measured value q_t. Green and red intervals along the horizontal axis denote the sets I_0 and I_1 in Eq. (3.16), they correspond to realizations leaving no error or an X error, respectively. In a memoryless decoding strategy one decides how to shift the code state after each measurement based on the value of q_t. In a maximum-likelihood decoding procedure one evaluates the path integral in Eq. (3.25). In a minimum-energy decoder one determines an optimal path for the sequence ϕ_t, $t \in [M]$, given the random potential and the quadratic "kinetic energy" term proportional to $(\phi_t - \phi_{t-1})^2$, see Eq. (3.29). Red and black lines show two decoding trajectories which start at the same point but have different winding numbers. Upon a final decoding step, choosing the black trajectory leads to deciding that no logical X error has taken place, since the final value ϕ_M lies in the green region. Choosing the red trajectory leads to deciding that a logical X has taken place since the final value ϕ_M lies in the red region. Examples of actual trajectories are shown in Fig. 3.4.

These equations are certainly different from the exact extremum equations (3.30), and the configuration found by this forward minimization technique necessarily has the energy higher than the exact minimum. On the other hand, empirically, the corresponding energy difference is typically small, much smaller what one gets, if the correct minimum is missed by the formally exact technique based on Eqs. (3.30). Even though this technique is only an approximation, it is fast and is accurate enough in practice. To ensure that the approximation does not hurt the performance of our decoder we can compare it to a rigorous dynamic programming approach.

DYNAMIC PROGRAMMING FOR THE MINIMUM-ENERGY DECODER

We can express the minimum-energy decoder for one GKP qubit as a dynamic programming problem. This allows us to check how well the forward-minimization technique performs. The goal is to minimize the energy from Eq. (3.29), which we write using the Villain approximation in reverse,

$$H[\phi_1,\ldots,\phi_M] = \sum_{t=1}^{M} \frac{(\phi_t - \phi_{t-1})^2}{2\sigma^2} - \sum_{t=1}^{M-1} \frac{\cos(q_t - \phi_t)}{\sigma_M^2}, \quad \phi_0 \equiv 0.$$

We define the partial energy, $H_k[\phi_1,\ldots,\phi_k]$ as the contribution from the first k terms from each sum,

$$H_k[\phi_1,\ldots,\phi_k] = \sum_{t=1}^{k} \frac{(\phi_t - \phi_{t-1})^2}{2\sigma^2} - \frac{\cos(q_t - \phi_t)}{\sigma_M^2},$$

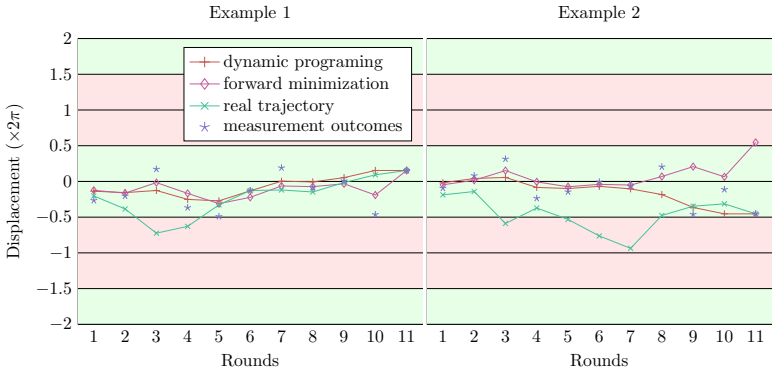


Figure 3.4: (Color online) Two examples of 11 rounds of error correction with data and measurement errors sampled from Gaussian distributions with $\sigma_0 = 0.4$. In the example on the left, forward-minimization and dynamic programming reach the same conclusion, whereas on the right forward-minimization reaches a wrong conclusion. One can compare this data with the sketched trajectories in Fig. 3.3.

and a single-variable function of ϕ_k as

$$M_k(\phi_k) = \min_{\phi_1,\ldots,\phi_{k-1}} \{H_k[\phi_1,\ldots,\phi_k]\}.$$

Then M_k, $k < M$, can be defined recursively by

$$M_1(\phi_1) = \frac{\phi_1^2}{2\sigma^2} - \frac{\cos(q_1-\phi_1)}{\sigma_{\mathrm{M}}^2},$$
$$M_k(\phi_k) = \min_{\phi}\left(M_{k-1}(\phi) + \frac{(\phi_k-\phi)^2}{2\sigma^2}\right) - \frac{\cos(q_k-\phi_k)}{\sigma_{\mathrm{M}}^2}.$$

If one discretizes the values of ϕ_k to a desired precision and restricts them to lie in a reasonable interval, the minimization with M time steps amounts to computing M lists of values of discrete functions $M_k(\phi_k)$. Unlike the minimization technique based on solving Eq. (3.30), there is no accuracy loss at larger M, and much less danger of missing the desired minimum with the present dynamic programming method.

We have compared our forward minimization technique with dynamic programming with a discretization of 200 points per period in a 4-periods window around the last measurement result. The statistics of success or failure of both decoders agree pretty closely while the forward minimization is much faster. In Fig. 3.4 we show two of the obtained realizations for illustration purposes. This comparison shows that the dynamic programming approach has very little advantage while it is substantially slower in its execution.

A very simple decoding strategy that one might also try is to trust every measurement outcome and immediately correct each round. This does not require any memory so we refer to it as the *memoryless* decoder. Intuitively, this method is risky as every round transfers the measurement errors to the data, increasing the variance of the effective error model acting on the data.

3.3.3. NUMERICAL RESULTS

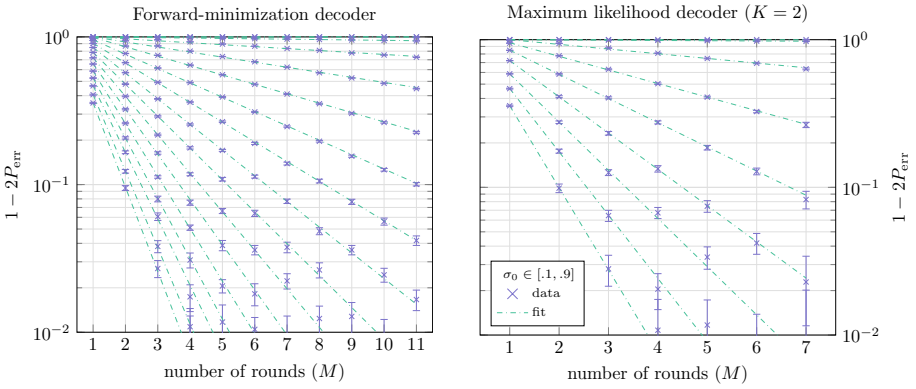


Figure 3.5: (Color online) Some of the numerical results for the forward-minimization and maximum-likelihood decoders (with cut-off $K = 2$). To observe the exponential decay towards 1/2 we plot $1 - 2P_{\text{err}}$ on a log scale for different number of rounds and different bare standard deviation σ_0. Low hundred thousands of trials are performed for each data point and the confidence intervals at 95% are shown. For each σ_0 we fit an exponential decay, the slope gives us an effective logical error rate per round. On both plots the value for σ_0 varies between .1 and .9, on the left by .05 increments, on the right by .1 increments. All effective logical error rates are plotted in Fig. 3.6.

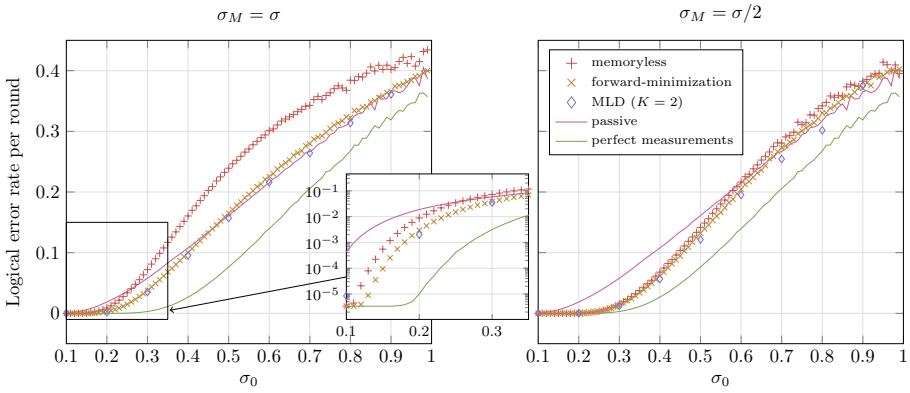


Figure 3.6: (Color online) Plots of the observed effective logical error rate per round for different decoding techniques. In the left plot we have taken the same standard deviation for measurement and data errors. On the right we have taken the measurement standard deviation equal to half that of the data errors. The horizontal axis shows the bare standard deviation σ_0.

We have numerically simulated these different decoders: maximum likelihood with a cut-off, forward-minimization, and memoryless decoding. We have also compared them to the scenario where the measurements are perfect, as well as to the completely *passive* decoder where one lets shift errors happen without performing error correction measurements. For each scenario we considered up to 11 rounds of measurements (M=7 for the maximum likelihood decoder), sampled errors, applied the decoder and gathered

statistics of success or failure of the procedure for different bare standard deviations for the errors $\sigma_0 \in [0.1, 1]$.

In every scenario we observe an exponential decay toward $1/2$ of the probability of logical error as seen in Fig. 3.5. Thus, as expected, there is an eventual loss of logical information for any values of σ and σ_M. The decay can be fitted in order to extract an effective logical error rate per round which we then plotted in Fig. 3.6.

One striking observation is that above a certain bare standard deviation, the measurement outcomes are simply not reliable enough, so that one cannot do substantially better than throwing away the measurements and passively letting errors accumulate. Roughly speaking, this occurs for $\sigma_0 \geq 0.5$ when $\sigma_M = \sigma$ and for $\sigma_0 \geq 0.7$ when $\sigma_M = \sigma/2$. Another observation for $\sigma_M = \sigma$ is that the memoryless technique actually quickly does more harm than the passive approach which forgoes error correction altogether. Finally, we observe that in the range of parameters studied, the forward-minimization technique performs almost as well as the maximum-likelihood decoding, while having the advantage of being much simpler computationally.

3.4. CONCATENATION: TORIC-GKP CODE

3.4.1. SETUP

We consider the following set-up shown in Figure 3.7. We have a 2D lattice of oscillators such that each oscillator encodes a single GKP qubit. In order to error correct these GKP qubits by the repeated application of the circuits in Fig. 3.1, a GKP ancilla qubit oscillator is placed next to each data oscillator, allowing for the execution of these circuits. After each step of GKP error correction, we measure the checks of a surface or toric code: a single error correction cycle for one of the toric-code checks is shown in Fig. 3.8. Note firstly that we omit GKP error correction after each gate in the circuit in Fig. 3.8: the reason is that we assume that these components are noiseless in this set-up so nothing would be gained by adding this. Secondly, the check operators of the toric code are those of the continuous-variable toric code [35] which are commuting operators on the whole oscillator space, see Appendix 3.8. The reason for using these checks is that for the displacements X and Z of a GKP qubit, it only holds that $X = X^{-1}$ and $Z = Z^{-1}$ on the code space. Expressed as displacement operators on two oscillators 1 and 2, it holds that $[X_1 X_2, Z_1 Z_2^{-1}] = 0$. The upshot is that one has to use some inverse CNOTs in the circuit in Fig. 3.8.

In the following, we will denote vertices of the square lattice with letters i, j, k, l, and the directed edges with the corresponding vertex pairs, e.g., $e = (ij)$. Quantities defined on the edges will be considered as vector quantities, e.g. $\hat{p}_{ij} = -\hat{p}_{ji}$ for the momentum operator. The *preferred orientation* is given by the direction of the coordinate axis. For such a lattice vector field, say some field $\boldsymbol{f}$, defined on edges, with components $f_{ij} = -f_{ji}$, we will denote the sum of the vectors from a vertex j as

$$(\nabla \cdot f)_j \equiv \sum_{k \sim j} f_{jk}, \tag{3.32}$$

where the summation is over all vertices k that are neighboring with j. Note that this choice recovers the $\pm$ signs for a X-check (red) in Fig. 3.7, when the quantities f_{jk} are

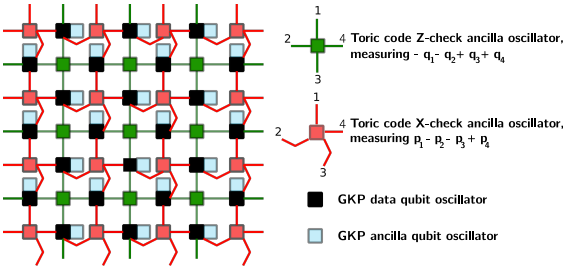


Figure 3.7: (Color online) The two-dimensional lay-out of oscillators, e.g. high-Q cavities, for the toric-GKP code. Shown is a fragment of a surface or toric code lattice. The different ± signs are defined by the orientations as explained in the main text.

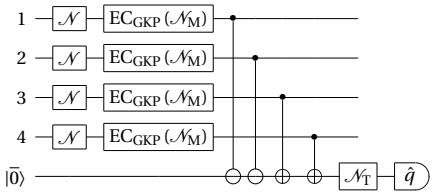


Figure 3.8: A single round of error correction for a Z-check for the toric-GKP code in Fig. 3.7, on oscillators numbered 1 to 4. The GKP error correction unit is given in Fig. 3.1, $|\bar{0}\rangle$ is a +1 eigenstate of Z and S_p. The inverse CNOT which induces the transformation $q_{\text{target}} \to q_{\text{target}} - q_{\text{control}}$ (while $p_{\text{control}} \to p_{\text{control}} + p_{\text{target}}$) is denoted using a $\ominus$ at the target qubit instead of a $\oplus$. A parallel execution of the CNOTs for the X-checks is possible in the toric code.

written in their preferred orientation. The circulation of a vector around a (square) plaquette $p \equiv (ijkl)$ is denoted as

$$(\nabla \times f)_p \equiv f_{ij} + f_{jk} + f_{kl} + f_{li}. \tag{3.33}$$

The preferred orientation for a plaquette, $p = (ijkl)$, is the one for which the closed path $i \to j \to k \to l \to i$ turns counter-clockwise. With this choice Eq. (3.33) recovers ± signs for a Z-check (green) in Fig. 3.7, when the quantities f_{jk} are written in their preferred orientation. With these notations, the vertex A_j (X-type) and the plaquette B_p (Z-type) operators of the toric code in Fig. 3.7 can be denoted as,

$$A_j \equiv e^{i\sqrt{\pi}(\nabla\cdot\hat{p})_j} \text{ and } B_p \equiv e^{i\sqrt{\pi}(\nabla\times\hat{q})_p}. \tag{3.34}$$

3.4.2. NOISELESS MEASUREMENTS & NUMERICAL RESULTS

We first examine the operation of the toric-GKP code in the channel setting. This simplified error model is based on the assumption that there are no measurement errors, i.e. $\mathcal{N}_{\text{M}} = \mathbb{1}$ and $\mathcal{N}_{\text{T}} = \mathbb{1}$ in Fig. 3.8, or equivalently, $\sigma_{\text{M}} = 0$ and $\sigma_{\text{T}} = 0$. In other words, the assumption is that both the GKP syndrome and the toric code syndrome are measured perfectly at every round.

Generally, when two codes are concatenated, it is possible to pass error information of the lower-level code (in this case, the GKP code) to the decoder for the top-level code

(here the toric code). The information that can be passed on is an estimation of the error rate on the underlying GKP qubits based on the outcome of the GKP error correction measurement. Intuitively, if the GKP measurement gives a $q \in [-\pi, \pi]$ which lies at the boundaries of the interval, say, beyond $-\pi/2$ or $\pi/2$, we are less sure that we have corrected this shift correctly [2]. In other words, the logical error rate depends on the measured value of the GKP syndrome, and this conditional error rate can be used in the standard minimum-weight matching decoder for the toric code. If the conditional single-qubit error rate fluctuates throughout the lattice, then one can expect that using this information will substantially benefit the toric code decoder.

We numerically demonstrate that this is the case for the toric-GKP code, reproducing some of the results in Ref. [17]. The threshold of the toric code against independent X and Z errors and without measurement error is about 11% [23]. If we are not using any GKP error information in the toric code decoding, then the threshold for σ_0 is set by the value for which $\mathbb{P}(X) = 11\%$ with $\mathbb{P}(X)$ shown as the green line in Fig. 3.6. We can run a standard minimum-weight matching toric code decoder where qubit X-errors are generated by sampling Gaussian noise with standard deviation σ followed by perfect GKP error correction on every GKP qubit. The left plot in Fig. 3.9 presents our numerical data in this scenario, showing a crossing point at $\sigma_0^c \approx 0.54$.

In order to use the GKP error information, we use Eq. (3.17) for the probability of an X error conditioned on the outcome $q \in [-\pi, \pi)$. Including normalization, this probability reads [3]

$$\mathbb{P}(1|q) = \frac{\sum_{k\in\mathbb{Z}} \mathbb{P}_\sigma(-2\pi + q + 4\pi k)}{\sum_{k\in\mathbb{Z}} \mathbb{P}_\sigma(q + 2\pi k)}. \tag{3.35}$$

To use these expressions we replace $k \in \mathbb{Z}$ by the corresponding sum with a cut-off K, restricting the summation to the interval $-K \leq k \leq K$. This is warranted since the Gaussian weight for large k is small. In the numerics we used the cut-off $K = 3$.

We numerically simulate the following process. For each toric code qubit, (ij), we first generate a shift error ϵ_{ij} according to the Gaussian distribution which leads to a GKP syndrome value $q_{ij} \in [-\pi, \pi)$. Given q_{ij}, we infer a correction which may give rise to an X error on qubit (ij). We evaluate the Z-checks of the toric code given this collection of errors and perform a minimum-weight matching algoritm to pair up the toric code defects. Logical failure is determined when the toric decoder makes a logical X error on any of the two logical qubits of the toric code. To use the information about the logical error rates $\mathbb{P}(1|q_{ij})$, for each qubit (ij), we define a weight:

$$w_{ij} = \log\left[\frac{1 - \mathbb{P}(1|q_{ij})}{\mathbb{P}(1|q_{ij})}\right]. \tag{3.36}$$

Then, we define a new weighted graph $G = (V, E)$, whose vertices, $p \in V$, are plaquette defects from the toric code graph and whose edges constitute the complete graph. Given an edge, $(p, p') \in E$, its weight $\omega_{p,p'}$ is the minimum weight of a path on the dual of the toric code graph connecting the defect plaquettes p and p'. Here, the path weight, $\omega_{p,p'}$, is the

[2] In fact, when one has approximate GKP states with finite photon number for ancilla and data oscillator, there is slightly more information in q than in q cmod 2π but we neglect this extra information here.

[3] Note that this conditional probability distribution is not identical to the likelihood function used in [17]. This might be why [17] does not observe a single cross-over point in their threshold plots.

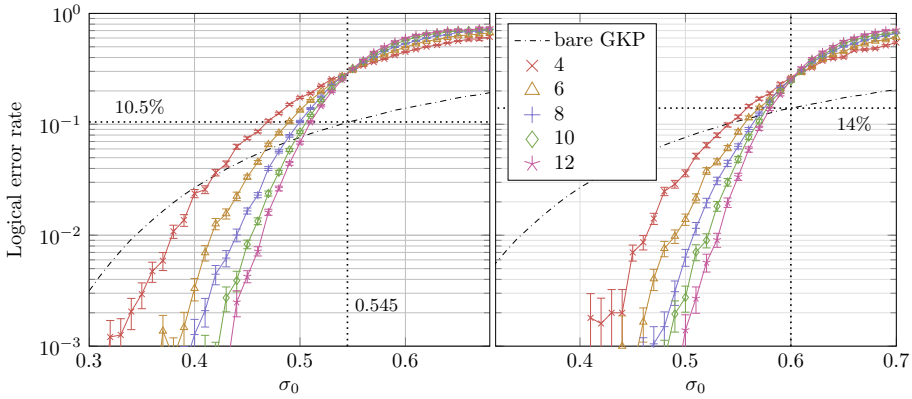


Figure 3.9: (Color online) Threshold comparison between decoding with or without GKP error information. On the left, the simulation only takes the average error rate into account and one obtains a threshold between $\sigma_0 \approx 0.54$ and $\sigma_0 \approx 0.55$ corresponding to $P(X) \approx 10\%$ and $P(X) \approx 10.7\%$, respectively. On the right, the simulation takes the GKP error information into account. In this case, the crossing point is around $\sigma_0^c \approx 0.6$ corresponding to $P(X) \approx 14\%$. The data are labeled by the distance of the toric code. "Bare GKP" is the logical error rate for a single GKP qubit whose errors are processed perfectly without measurement errors (green line in Fig. 3.6).

sum of the weights, w_{ij}, of all edges crossed by the path. Minimum-weight-matching (Blossom) algorithm is then run on this ω-weighted graph G, leading to a matching of defects and thus an inferred X error.

Specifically, we used Dijkstra's algorithm for finding a minimum-weight path in a weighted graph as provided by the Python library `Graph-tools` [36], and the minimum-weight matching algorithm from the C++ library `BlossomV` [37]. The process of sampling from shift errors is repeated many times; the logical error rate plotted in Fig. 3.9 is given by the fraction of runs which result in logical failure over the total number of runs.

3.5. NOISY MEASUREMENTS: 3D SPACE-TIME DECODING

3.5.1. ERROR MODEL

In this section we consider how to use both GKP and toric code error information when both error correction steps are noisy, using repeated syndrome measurements. This is the full error model in Fig. 3.8, which represents one complete QEC cycle. We only consider p-type shift errors (inducing shifts in q), the initial state at $t = 0$ is assumed to be perfect, and the last of M rounds of measurements noise-free, both for the GKP and the toric code ancillas. We will address the question of whether or not there is a *decodable* phase in the space of parameters, such that by increasing the size of the code and the number of measurement cycles, the probability of a logical error can be made arbitrarily small.

To visualize errors of different origin, for the M-times repeated measurement of the toric code on an $L \times L$ square lattice, it is convenient to consider a three-dimensional cubic lattice, with periodic boundary conditions along x and y directions, separated into horizontal layers. Each layer corresponds to a measurement round $t \in [M]$, see Fig. 3.10. In each time layer t, we use the same notations and conventions of directed edges and

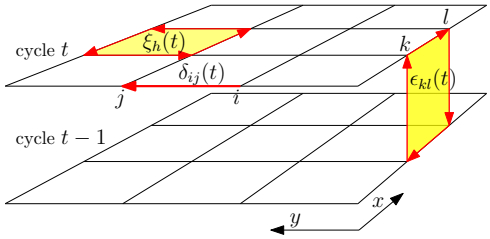


Figure 3.10: Notations for repeated toric-GKP errors. Data oscillators are located on the bonds of the square lattice. A GKP measurement error for the bond $b \equiv (ij,t)$ in the measurement cycle t is denoted as $\delta_{ij}(t) \equiv \delta_b$, while the corresponding measurement error for the toric code generator on horizontal plaquette $p \equiv (h,t)$ is denoted as $\xi_h(t) \equiv \xi_p$. A data qubit error $\epsilon_{kl}(t) \equiv \epsilon_p$ is associated with the vertical plaquette p directly below the bond kl in the layer t.

vector quantities as in Section 3.4.1. Hence we associate a GKP data qubit oscillator with each edge (ij) of the square lattice.

We thus denote the shift occurring on a data oscillator just before the measurement at time t as the shift $\epsilon_{ij}(t)$ (induced by channel $\mathcal{N}$). Since the shifts accumulate, the net shift on the oscillator at bond ij just before the measurement at time t is [cf. Eq. (3.15)]

$$\phi_{ij}(t) \equiv \sum_{t'=1}^{t} \epsilon_{ij}(t'). \tag{3.37}$$

Furthermore, we denote the GKP measurement error of this oscillator at time t as $\delta_{ij}(t)$. This is the shift error on the ancilla inside the corresponding $\mathrm{EC}_{\mathrm{GKP}}$ unit (induced by channel $\mathcal{N}_{\mathrm{M}}$). With these notations, we can write for the GKP syndrome $q_{ij}^{\mathrm{GKP}}(t)$ at time t,

$$q_{ij}^{\mathrm{GKP}}(t) = \delta_{ij}(t) + \phi_{ij}(t) \bmod 2\pi. \tag{3.38}$$

In addition, we have the toric code syndrome. Specifically, we consider the toric code plaquette operators, see Fig. 3.7 and Eq. (3.34). The result of the toric code syndrome measurement on the plaquette $h \equiv (ijkl)$ at time t is

$$q_h^{\mathrm{tor}}(t) = \xi_h(t) + (\nabla \times \phi)_h(t) \bmod 4\pi, \tag{3.39}$$

where the bond vectors $\phi_{ij}(t)$ are the accumulated errors in Eq. (3.37), and $\xi_h(t)$ is the plaquette measurement error (induced by channel $\mathcal{N}_{\mathrm{T}}$). Note that, unlike for the GKP measurements, the syndrome $\boldsymbol{q}^{\mathrm{tor}}$ is measured modulo 4π, since the ancilla starts in the state $|\bar{0}\rangle$.

Since we assume that the measurement errors in the last layer, $t = M$, are absent, we have $\xi_h(M) = \delta_{ij}(M) = 0$. Writing the product of the corresponding probability densities,

we obtain an analog of Eq. (3.29) for the effective energy

$$H(\boldsymbol{\phi};\boldsymbol{q}) = \sum_{t=1}^{M}\sum_{\langle ij\rangle}\frac{\left(\phi_{ij}(t)-\phi_{ij}(t-1)\right)^2}{2\sigma^2} + \sum_{t=1}^{M-1}\sum_{\langle ij\rangle} V_{\sigma_{\mathrm{M}}}\left(q_{ij}^{\mathrm{GKP}}(t)-\phi_{ij}(t)\right) + \sum_{t=1}^{M-1}\sum_{h} V_{\sigma_{\mathrm{T}}/2}\left(\frac{q_h^{\mathrm{tor}}(t)-(\nabla\times\phi)_h(t)}{2}\right), \tag{3.40}$$

which depends on the accumulated field $\boldsymbol{\phi}$ with components $\phi_{ij}(t)$ and on the total measured syndrome $\boldsymbol{q}\equiv\{\boldsymbol{q}^{\mathrm{GKP}},\boldsymbol{q}^{\mathrm{tor}}\}$. Here, $\sum_{\langle ij\rangle}$ indicates a summation over all bonds of the square lattice, the summation over h runs over all square-lattice faces (horizontal), and the structure of the last term accounts for the 4π-periodicity of toric syndrome measurements, see Eq. (3.39).

The energy in Eq. (3.40) defines the conditional probability $\mathbb{P}(\boldsymbol{\phi}|\boldsymbol{q})\propto\exp(-H(\boldsymbol{\phi};\boldsymbol{q}))$, up to a normalization factor. The measurements in the last time-layer, $t=M$, constrain the values $\phi_{ij}(M)$ as follows. From the GKP syndrome we have, as in Sec. 3.3,

$$\phi_{ij}(M) = q_{ij}^{\mathrm{GKP}}(M) - 2\pi k_{ij}, \tag{3.41}$$

while the toric syndrome for each square-lattice face $h=(ijkl)$ gives

$$(\nabla\times\phi)_h(M)\equiv\phi_{ij}(M)+\phi_{jk}(M)+\phi_{kl}(M)+\phi_{li}(M) = q_h^{\mathrm{tor}}(M)-4\pi k_h. \tag{3.42}$$

These equations can be solved to find a last-layer binary candidate error vector $\boldsymbol{b}\in\mathbb{F}_2^{2L^2}$, whose components $b_{ij}\equiv k_{ij} \bmod 2$ give the parity of the integer shifts k_{ij} in Eq. (3.41). Just as for the usual toric code, Eqs. (3.41), (3.42) determine $\boldsymbol{b}$ up to arbitrary cycles on the dual lattice, i.e. X-stabilizers (homologically trivial) and X logical errors (homologically non-trivial) of the toric code. Adding trivial cycles to $\boldsymbol{b}$ gives another equivalent last-layer candidate which should be summed as part of the same sector. A trivial cycle can be written as the gradient of a binary field, $\boldsymbol{b}_{ij}^{\mathrm{triv}} = n_j - n_i$, with $n_i\in\mathbb{F}_2$. To turn $\boldsymbol{b}$ into an inequivalent last-layer candidate error, one should add a homologically non-trivial cycle, $\boldsymbol{c}\in\mathbb{F}_2^{2L^2}$.

We can now write explicitly the partition function $Z_{\boldsymbol{c}}(\boldsymbol{b}|\boldsymbol{q})$, equivalent to Eq. (3.7), which determines the conditional probability, given the measurement outcomes $\boldsymbol{q}$ of the equivalence class of last-layer candidate error $[\boldsymbol{b}+\boldsymbol{c}]$,

$$Z_{\boldsymbol{c}}(\boldsymbol{b}|\boldsymbol{q}) = N'^{-1}\int \mathrm{d}\boldsymbol{\phi}\,\mathrm{e}^{-H(\boldsymbol{\phi};\boldsymbol{q})}\prod_{\langle ij\rangle}\sum_{\{n_i\in\mathbb{F}_2\}}\sum_{m_{ij}\in\mathbb{Z}}\delta\Big(\phi_{ij}(M)-q_{ij}^{\mathrm{GKP}}(M) + 2\pi[b_{ij}+c_{ij}-n_j+n_i+2m_{ij}]\Big). \tag{3.43}$$

For ML decoding, given a $\boldsymbol{b}$ which satisfies Eqs. (3.41) and (3.42), one needs to compare $Z_{\boldsymbol{c}}(\boldsymbol{b}|\boldsymbol{q})$ for different $\boldsymbol{c}\in\mathbb{F}_2^{2L^2}$ which are inequivalent binary codewords of the toric code, i.e. the three homologically non-trivial domain walls on the square lattice or the trivial vector. ML decoding then prescribes that we choose the error $\boldsymbol{b}+\boldsymbol{c}$ as the correction where $\boldsymbol{c}$ has the largest partition function $Z_{\boldsymbol{c}}(\boldsymbol{b}|\boldsymbol{q})$.

3.5.2. EQUIVALENT FORMULATION WITH $U(1)$ SYMMETRY

The partition function in Eq. (3.43) with the Hamiltonian in Eq. (3.40), as a statistical-mechanical model, is not so convenient to analyze, since the components of the syndrome are not independent of each other. So, we will consider an equivalent form of the partition function, that explicitly depends on the data errors $\epsilon_{ij}(t) \equiv \epsilon_p$, the measurement errors $\delta_{ij}(t) \equiv \delta_b$, and the toric code measurement errors $\xi_h(t) \equiv \xi_p$. We group all these errors into one error record $\boldsymbol{e} = \{\boldsymbol{\epsilon}, \boldsymbol{\delta}, \boldsymbol{\xi}\}$. Any error $\boldsymbol{e}'$ that is equivalent to $\boldsymbol{e}$ can be obtained by adding, so to say, stabilizer generators of the space-time code. This can be expressed using a 2π-periodic vector field $\boldsymbol{A}$ whose components are real-valued on horizontal bonds in layers $t \in [M-1]$, and $\{0,\pi\}$-valued on the vertical bonds connecting layers $t-1$ and t for all $t \in [M]$. For horizontal bonds b in layers $t = 0$ and $t = M$, $A_b = 0$. With these notations, we can express the partition function in Eq. (3.7) as

$$H(\boldsymbol{A};\boldsymbol{e}) = \sum_{b\parallel xy} V_{\sigma_M}\big(\delta_b - 2A_b\big) + \sum_{p\parallel xy} V_{\sigma_T/2}\left(\frac{\xi_p}{2} - (\nabla\times A)_p\right) + \sum_{p\perp xy} V_{\sigma/2}\left(\frac{\epsilon_p}{2} + (\nabla\times A)_p\right), \tag{3.44}$$

$$Z_{\boldsymbol{0}}(\boldsymbol{e}) = N''^{-1} \sum_{A_b\in\{0,\pi\}:b\perp xy} \prod_{b\parallel xy} \int_{-\pi}^{\pi} \mathrm{d}A_b \; e^{-H(\boldsymbol{A};\boldsymbol{e})}. \tag{3.45}$$

with some normalization N''. To derive these equations, we determine the shift errors that leave the syndrome record $\{\boldsymbol{q}^{\mathrm{GKP}}, \boldsymbol{q}^{\mathrm{tor}}\}$ unchanged without inducing a logical error. These are called the gauge degrees of freedom and form the stabilizer group for the space-time code. In our case there are five types of gauge degrees of freedom, four discrete and one continuous. The discrete ones are genuine symmetries of the quantum states involved in the code or measurement circuits. Namely, the input state of an ancilla in the GKP measurement circuit in Fig. 3.2 is stabilized by an X operator, whose action is equivalent to a 2π-shift of the corresponding GKP measurement error $\delta_{ij}(t)$. Similarly, application of an $X^2 = S_p$ GKP stabilizer generator to a data qubit or a toric-code ancilla in Fig. 3.8 is equivalent to a 4π-shift of the corresponding error, $\epsilon_{ij}(t)$ or $\xi_h(t)$, respectively. We also have toric code vertex operators A_j [Eq. (3.34)] whose action corresponds to simultaneous 2π-shifts on the four adjacent qubits, $\epsilon_{ij}(t) \to \epsilon_{ij}(t) + 2\pi$. This discrete gauge freedom will be captured by the two-valued field A_b on the vertical bonds.

The only continuous degree of freedom is a space-time one: it corresponds to adding a continuous shift a on a data oscillator at some time step and then canceling it at the next time step, while hiding the shift from the adjacent GKP and toric syndrome measurements by adding the shifts $\pm a$ as necessary on the corresponding ancillas.

When applied to the Gaussian distribution, the discrete local shifts are responsible for forming the Villain potentials (3.26) in the effective Hamiltonian (3.44), where additional rescaling in the last two terms was necessary to account for the 4π-periodicity. The remaining two degrees of freedom are represented by the vertical (discrete) and horizontal (continuous) components of the doubled vector potential $2\boldsymbol{A}$. The scale of the vector potential $\boldsymbol{A}$ was chosen to make easier contact with previous literature on related models. With this choice, adding a π-shift to a component of $\boldsymbol{A}$ correspond to a GKP X-logical, which was a 2π-shift in the previous sections.

Equations (3.44), (3.45) have to be supplemented with the appropriate boundary

conditions to be used in decoding. Given the toric-code codeword $\boldsymbol{c}$, in order to calculate $Z_{\boldsymbol{c}}(\boldsymbol{e})$, corresponding to the sector $[\boldsymbol{e}+\boldsymbol{c}]$, one can add $2\pi\boldsymbol{c}$ to the data error in the top layer, $t = M$. This can be achieved by introducing a fixed non-zero vector potential in this layer, namely $A_b = \pi c_{ij}$ for all top-layer horizontal bonds $b = (ij, t = M)$, instead of zero for the trivial sector.

Equations (3.44), (3.45) look very different from the equivalent form that we first derived, i.e. Eqs. (3.40) and (3.43). A map between these two formulations is given in the next Section.

ALTERNATIVE DERIVATION OF THE $U(1)$-SYMMETRIC MODEL

Here we derive Eqs. (3.44) and (3.45) directly from Eqs. (3.40) and (3.43). First, note that the integration over components of the field $\boldsymbol{\phi}$ in Eq. (3.43) is done in an infinite interval, while the two last terms in the bulk Hamiltonian (3.40) are 2π- and 4π-periodic, respectively. In addition, the background fluxes $(\nabla\times\phi)_h(t)$ in the last term of Eq. (3.40) are explicitly symmetric with respect to the gauge transformation

$$\phi_{ij}(t) \to \phi_{ij}(t) + \alpha_j(t) - \alpha_i(t), \tag{3.46}$$

where $\alpha_j(t) = \alpha_v$ is a real-valued scalar field associated with the vertices $v = (i,t)$ of the cubic lattice. While these are *not* the symmetries of the full Hamiltonian (3.40), we can now render it in a more familiar $U(1)$-symmetric form with a simple change of variables.

For a fixed error $\boldsymbol{e} \equiv \{\boldsymbol{\epsilon},\boldsymbol{\delta},\boldsymbol{\xi}\}$ which corresponds to the syndrome $\boldsymbol{q}$, let us denote the corresponding accumulated error, see (3.37), as $\boldsymbol{\phi}^{(0)}$. We can make a change of integration variables in Eq. (3.43), for all bonds (ij) and layers $1 \le t \le M$, following

$$\phi_{ij}(t) = \phi^{(0)}_{ij}(t) + 2A_{ij}(t) + 4\pi m_{ij}(t) \pm 2\pi \sum_{t'=1}^{t} \big(s_j(t') - s_i(t')\big), \quad -\pi < A_{ij}(t) \le \pi, \tag{3.47}$$

where the field $A_{ij}(t)$ is continuous, $m_{ij}(t) \in \mathbb{Z}$ is integer-valued, and an additional 2π-shift is proportional to the lattice gradient of the binary field $s_i(t) \in \mathbb{F}_2$ accumulated over time. Strictly speaking, the last term is unnecessary as it causes some double counting in the measure. However, the corresponding factor is a constant that is finite on a finite lattice, so it does not cause any trouble. On the other hand, these binary charges simplify the boundary conditions, since we can simply write $s_i(t) + \ldots + s_i(M) = n_i$, see Eq. (3.43), and thus trade the summation over the boundary field n_i for the binary field $s_i(t)$ in the bulk. To complete the derivation, also define $A_{ij}(t=0) = 0$, and take $A_b = \pi s_i(t)$ for the vertical bond b at the square-lattice vertex i, between layers $t-1$ and t for all $t \in [M]$. This gives for the vertical plaquette p at the bond (ij), between the same two layers,

$$\phi_{ij}(t) - \phi_{ij}(t-1) \to \phi^{(0)}_{ij}(t) - \phi^{(0)}_{ij}(t-1) + 2(\nabla\times A)_p \equiv \epsilon_p + 2(\nabla\times A)_p,$$

where the appropriate sign in Eq. (3.47) needs to be chosen to recover the part of the flux that is missing in the first term of Eq. (3.40), and we absorbed the summation over the integer-valued $m_{ij}(t)$ into the definition of the Villain potential $V_{\sigma/2}\big(\epsilon_p/2 + (\nabla\times A)_p\big)$, an even 2π-periodic function of the argument, see Eq. (3.26). In the remaining two terms we use Eqs. (3.38) and (3.39), to recover Eq. (3.44) exactly [4]. Comparing with Eqs. (3.41),

[4] Strictly speaking we get $-(\nabla\times A)_p$ for a vertical p along y. But we can push this sign to the disorder component using the parity of $V_{\sigma/2}$ and use the symmetry of our noise model to ignore this sign.

(3.42), it is also easy to check that for this particular error $\boldsymbol{e}$, one can simply take the vector potential $A_b = 0$ for the top-layer in-plane bonds, $b = (ij, M)$. Similarly, for the three sectors where the error in the top layer is shifted by a non-trivial toric codeword $\mathbf{0} \neq \boldsymbol{c} \in \mathbb{F}_2^{2L^2}$, we can take $A_{ij}(M) = \pi c_{ij}$.

3.5.3. DECODER AND NUMERICAL RESULTS

Maximum likelihood decoding can be done by comparing conditional probabilities in different sectors, see Eq. (3.7). Just as in the case of a single GKP qubit, Gaussian integrations in the relevant partition functions, $Z_{\boldsymbol{c}}(\boldsymbol{b}|\boldsymbol{q})$, in Eq. (3.43), or its equivalent form $Z_{\boldsymbol{c}}(\boldsymbol{e})$, in Eq. (3.45), can be carried through exactly. This would leave expressions similar to Eq. (3.25), with an additional summation over $2L^2$ binary spins. In principle, such expressions can be evaluated using Monte Carlo sampling techniques. In practice, the complexity of such a calculation is expected to be high, because the corresponding coupling matrix is not sparse, just as the matrix A in Eq. (3.25) is not sparse, see Appendix 3.2.2. For this reason, we have not attempted ML decoding for toric-GKP codes.

We have constructed several decoders which approximate ME decoding. The idea is to find a configuration of the field $\boldsymbol{A}$ minimizing the Wilson version of the Hamiltonian (3.44), i.e. with Villain potentials replaced by cosines, by decomposing it into a continuous part (to be guessed or found using a local minimization algorithm), and a binary field which represents frustration, to be found using minimum weight matching. This decomposition relies on the analysis of the Hamiltonian (3.44) in the limit of perfect GKP measurements, $\sigma_{\mathrm{M}} \to 0$.

ME DECODING IN THE LIMIT OF PERFECT GKP MEASUREMENTS

Let us consider what happens with our model (3.44), (3.45) in the limit $\sigma_{\mathrm{M}} \to 0$. First, in this limit all GKP measurement errors δ_b vanish with probability one. Second, the first term in Eq. (3.44) forces $A_b \in \{0, \pi\}$ for all horizontal bonds, the same as we already had for vertical bonds. This forces all plaquette fluxes to take integer values times π. We show here that in this limit we recover a version of the random-plaquette gauge model (RPGM) associated with decoding the usual qubit toric code in the presence of (toric) syndrome measurement errors [38].

Since the limit $\sigma_{\mathrm{M}} \to 0$ makes the vector potential $A_b \in \{0, \pi\}$, and in turn the plaquette flux $B_p \equiv (\nabla \times A)_p$, discrete, one can interpret it as a spin degree of freedom, using the fact that the Villain potential is 2π-periodic as well as even. Indeed, considering for example a vertical plaquette, $p \perp xy$, in Eq. (3.44), one can write,

$$
\begin{aligned}
& V_{\sigma/2}\left(\frac{\epsilon_p}{2} + B_p\right) = -\tau_p(\boldsymbol{e}) e^{iB_p} + \text{const}, \\
& p \perp xy, \quad \tau_p(\boldsymbol{e}) \equiv \frac{1}{2}\left[V_{\sigma/2}\left(\frac{\epsilon_p}{2} - \pi\right) - V_{\sigma/2}\left(\frac{\epsilon_p}{2}\right)\right] \approx \frac{4}{\sigma^2}\cos\left(\frac{\epsilon_p}{2}\right).
\end{aligned} \tag{3.48}
$$

The additive constant has no effect and can be ignored. Similarly for horizontal plaquettes, $p \parallel xy$, where one obtains the weights,

$$
p \parallel xy, \quad \tau_p(\boldsymbol{e}) \equiv \frac{1}{2}\left[V_{\sigma_{\mathrm{T}}/2}\left(\frac{\xi_p}{2} - \pi\right) - V_{\sigma_{\mathrm{T}}/2}\left(\frac{\xi_p}{2}\right)\right] \approx \frac{4}{\sigma_{\mathrm{T}}^2}\cos\left(\frac{\xi_p}{2}\right). \tag{3.49}
$$

Then, if one defines from $\boldsymbol{A}$ some Ising spins, $\boldsymbol{s}$, using for each bond, $s_b \equiv e^{iA_b} \in \{-1, 1\}$, one obtains, in place of Eq. (3.44), a RPGM very similar to that in Ref. [38],

$$H(\boldsymbol{s};\boldsymbol{e}) = -\sum_p \tau_p(\boldsymbol{e}) u_p, \quad u_p \equiv \prod_{b\in p} s_b. \tag{3.50}$$

Unlike in the usual RBGM obtained for the qubit toric code [38] where plaquette weights can take only two values, $\tau_p = \pm J$, here quenched randomness leads to a continuous distribution of the weights. This model is similar to the 2D random-bond Ising model constructed in Sec. 3.4.2 for decoding a toric-GKP code in the channel setting, also in the limit $\sigma_M = 0$. In fact, the weights concerning data errors in Eq. (3.48) (without the cosine approximation), are equivalent to those given by Eqs. (3.35), (3.36). Similar to its counterpart with sign disorder, the phase-diagram of the RPGM with the Hamiltonian (3.50) will show a transition from an ordered to disordered phase as the temperature β^{-1} or the strength of the quenched disorder is increased. If we increase the disorder σ along a line with any fixed ratio $r = \sigma_T/\sigma$, a version of the standard argument [39] shows that no ordered phase can exist beyond the critical value of σ reached along the "Nishimori line," $\beta = 1$ [cf. Eq. (3.12)]. We expect that it is this critical value of σ that is associated with the memory phase transition.

An important quantity that governs the structure of the minimum of the RPGM Hamiltonian (3.50) is *frustration*. We call a cube frustrated when it has an odd number of its boundary plaquettes p with $\tau_p(\boldsymbol{e}) < 0$. Since every spin, s_b, affects the sign of two plaquettes in a cube, for a frustrated cube, no spin configuration on the edges can simultaneously satisfy all the plaquette terms. For the usual toric codes, frustrated cubes can be readily identified by stabilizer defects without referring to a candidate error. For toric-GKP codes one has to be more careful. In the limit $\sigma_M \to 0$, the frustration cannot be read directly from the value of the toric code syndromes but all candidate errors exhibit the same frustration. So picking any candidate error, $\boldsymbol{e}$, finds it. When $\sigma_M \neq 0$, this is no longer the case, the frustration can change between different candidate errors.

We examine the problem of finding the optimum spin configuration, $\boldsymbol{s}$, which minimizes the RPGM Hamiltonian (3.50), given the weights $\tau_p(\boldsymbol{e})$. For a given $\boldsymbol{s}$, we call a plaquette term, $\tau_p(\boldsymbol{e})u_p$, *satisfied* when $\tau_p(\boldsymbol{e})u_p > 0$ and *unsatisfied* when $\tau_p(\boldsymbol{e})u_p < 0$. Necessarily, a frustrated cube is incident to an odd number of unsatisfied plaquettes, in this sense frustrated cubes are the source of unsatisfied plaquettes. Let S be any set of plaquettes such that the frustrated cubes are incident to an odd number of plaquettes in S and the unfrustrated cubes are incident to an even number of plaquettes in S. One has

$$\min_{\boldsymbol{s}} H(\boldsymbol{s};\boldsymbol{e}) = -\sum_p |\tau_p(\boldsymbol{e})| + 2\min_S \sum_{p\in S} |\tau_p(\boldsymbol{e})|.$$

Hence minimum-weight matching on a 3D lattice with vertices representing the frustrated cubes determines the optimal set of unsatisfied plaquettes $S_{\min}$. Given a candidate error $\boldsymbol{e}$ let S_{cand} be the set of plaquettes with $\tau_p(\boldsymbol{e}) < 0$. The candidate error $\boldsymbol{e}$ is now modified using the minimum-weight matching by adding 2π for all plaquettes in $S_{\min}$ as well as adding 2π on all plaquettes in S_{cand} (remember that 4π-shifts are elements of the stabilizer group). This corresponds to an addition of a stabilizer or logical operator to the candidate error $\boldsymbol{e}$, leaving the syndrome unchanged. This modified candidate error is the proposed correction for this decoder.

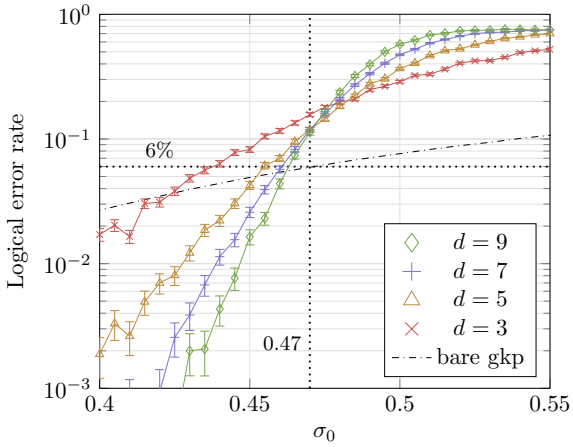


Figure 3.11: (Color online) Numerical results for perfect GKP measurements, with $\sigma = \sigma_{\mathrm{T}}$ and $\sigma_{\mathrm{M}} = 0$. The logical error rate as a function of the bare standard deviation σ_0, see Eq. (3.14), is shown for toric code distances $d = L$ as indicated in the caption. The vertical dotted line indicates the position of the crossing point in the data, a threshold at $\sigma_0 \approx .47$, which corresponds to a logical error probability $p = 6\%$ for single-qubit GKP code with $\sigma_{\mathrm{M}} = 0$. The latter value is recovered as the vertical position on the dash-dotted line which reproduces the green line $\sigma_{\mathrm{M}} = 0$ from Fig. 3.6 (left). The observed threshold is well above $\sigma_0 = 0.41$ which can be superficially expected from the $p = 2.9\%$ threshold for the 2D toric code under phenomenological error model.

We implemented the described decoder to minimize the energy (3.50), with Villain potentials replaced with cosines, see Eqs. (3.48),(3.49). A very simple estimate of the expected performance of the toric-GKP code in this setting is the following. It is known that the threshold for the toric code is about 2.9% under phenomenological noise with independent X and Z errors [38]. If we assume that all errors are due to the logical error on the underlying GKP qubits, see Fig. 3.6, then one can ask what σ_0 leads to a probability for an X error (or equivalently Z) equal to 2.9%. This of course depends on the measurement error σ_{M} as well as the decoding method for the GKP qubit. In case of no measurement error ($\sigma_{\mathrm{M}} = 0$), the error probability can be found by averaging $\mathbb{P}_{M=1}(1|q_1)$, see Eq. (3.17), over the Gaussian error distribution, it is plotted as the green line in Fig. 3.6. An error probability of 2.9% corresponds to $\sigma_0 \approx 0.41$. Our numerical results are shown in Fig. 3.11. One can see a crossing point around $\sigma_0 = .47$ which can be converted to around 6% error rate per round for a GKP qubit with $\sigma_{\mathrm{M}} = 0$, see Fig. 3.6. We can conclude that, similarly to the results in Sec. 3.4.2, the continuous-valued weights $\tau_p(\boldsymbol{e})$, constitute valuable information to the decoder about the likelihood of errors and permit to surpass decoders which do not have access to such information.

DEALING WITH MULTIPLE COMPETING MINIMA

A difficulty in minimizing the Hamiltonian (3.44) is the existence of a large number of competing local minima. This was already the case for a single GKP qubit, which we considered in Sec. 3.3. We saw in the previous section that in the case of perfect GKP measurements, the solution can be found efficiently because the problem is equivalent to a minimum weight matching on a graph. Our approach to minimizing Eq. (3.44) in the

general case will be to decompose the vector potential $A_b \in [-\pi,\pi)$ for horizontal bonds $b=(ij,t)$ into a discrete field $A_b^{(0)} \in \{0,\pi\}$, and an auxiliary continuous field $a_b \in \left[-\frac{\pi}{2},\frac{\pi}{2}\right)$,

$$A_b = A_b^{(0)} + a_b + \delta_b/2. \tag{3.51}$$

For vertical bonds there is no need of such a substitution since the corresponding field is already discrete, see Eq. (3.45) so we set $a_b = 0$ for all $b \perp xy$. The discrete part of the field, $\boldsymbol{A}^{(0)}$, can be used to define Ising spins, $\boldsymbol{s}$, similarily as before, $s_b \equiv e^{iA_b^{(0)}}$. The definition of the weigths, $\tau_p(\boldsymbol{e})$, has to be extended since now they depend also on $\delta_b \neq 0$ and $a_b \neq 0$, or more precisely the residual fluxes $b_p \equiv (\nabla \times a)_p$. Adding these dependencies, the new weights, $\tau_p(\boldsymbol{a};\boldsymbol{e})$, given only in their cosine approximation, read

$$\tau_p(\boldsymbol{a};\boldsymbol{e}) = \frac{4}{\sigma_{\mathrm{T}}^2}\cos\left(\frac{\xi_p}{2} - \frac{1}{2}(\nabla\times\delta)_p - b_p\right), \qquad p\parallel xy, \tag{3.52}$$

$$\tau_p(\boldsymbol{a};\boldsymbol{e}) = \frac{4}{\sigma^2}\cos\left(\frac{\epsilon_p}{2} - \frac{1}{2}\left(\delta_{ij}(t)-\delta_{ij}(t-1)\right) + b_p\right), \quad p\equiv(ij,t)\perp xy. \tag{3.53}$$

Rewriting the Hamiltonian (3.44), using the substitution (3.51) and using the cosine approximation of the Villain potentials gives,

$$H(\boldsymbol{a},\boldsymbol{s};\boldsymbol{e}) = -\frac{1}{\sigma_{\mathrm{M}}^2}\sum_{b\parallel xy}\cos(2a_b) - \sum_p \tau_p(\boldsymbol{a};\boldsymbol{e})u_p. \tag{3.54}$$

Then, the minimization over the gauge field $\boldsymbol{A}$, or equivalently over $\{\boldsymbol{a},\boldsymbol{s}\}$, factors out into minimizing the RPGM similar to Eq. (3.50), and a minimization over the continuous field $\boldsymbol{a}$.

In addition to the candidate error $\boldsymbol{e}$, the RPGM weights now also depend on $\boldsymbol{a}$ via the flux field $\boldsymbol{b}$. Moreover, even with the restriction on the auxiliary vector potentials, $|a_b| \leq \pi/2$, the corresponding fluxes are not so restricted; in particular, both for horizontal and vertical plaquettes one may have $|b_p| = \pi$, sufficient to flip the sign of the RPGM weight, $\tau_p(\boldsymbol{a};\boldsymbol{e})$, and flip the frustration of the two adjacent cubes.

Nevertheless, even though frustration depends on configuration of both the spins and the residual fluxes b_p, increasing the number of variables by the substitution (3.51) does simplify the minimization problem. First the fields a_b are uniquely defined by the fluxes b_p. Indeed, since the gauge fields a_b are only non-zero on the horizontal bonds, and are zero at the bottom layer, $t = 0$, the gauge is fixed. Furthermore, the GKP terms tend to suppress order-reducing fluctuations by favoring small $|a_b|$. Thus, with σ_{M} small compared to σ and σ_{T}, we can hope to find a reasonably good solution just by setting $a_b = 0$.

ACTUAL DECODER ALGORITHMS AND THEIR PERFORMANCE

In order to design a syndrome-based decoder with the starting point $a_b = 0$, we first need to come up with a candidate error $\boldsymbol{e} \equiv \boldsymbol{e}(\boldsymbol{q})$. Since we are not doing the full minimization of the corresponding energy, the method to find $\boldsymbol{e}$ will necessarily affect the performance of the resulting decoder.

ALGORITHM 1:

1. For each plaquette h, starting from $t = M-1$ down to $t = 1$, set the toric code measurement error $\xi_h(t) = q_h^{\text{tor}}(t) - q_h^{\text{tor}}(t+1)$, to suppress the increments of the toric syndrome. This leaves non-zero toric code syndromes only in the first layer, $q_h^{\text{tor}}(t=1)$.

2. Set data errors in the first layer, $\epsilon_{ij}(t=1)$, to move non-zero toric code syndromes to the left (with $ij \| y$), then down along the leftmost column. Due to the boundary conditions, the sum of all toric code syndromes is 0, meaning that after this procedure, all toric code syndromes are removed including the one in the left-bottom plaquette.

3. For each square lattice bond (ij), starting with $t = 1$ up to $t = M-1$, use Eq. (3.38) to set GKP errors $\delta_{ij}(t)$ to suppress the GKP syndromes $q_{ij}^{\text{GKP}}(t)$ (without changing $\phi_{ij}(t)$). This leaves non-zero GKP syndromes only in the top layer, $t = M$, with toric syndromes all zero.

4. Make a gauge transformation on the top layer data errors, $\epsilon_{ij}(t=M) \to \epsilon_{ij}(t=M) + \chi_j - \chi_i$ to move non-zero GKP syndromes to the left (to $x = 0$) and then down (to $y = 0$). The last step works because equations (3.41), (3.42) are satisfied for the updated syndrome; with $q_h^{\text{tor}}(t=M) = 0$ these guarantee that the GKP syndrome is a gradient.

Having determined the candidate error $\boldsymbol{e}$, we calculate the RPGM weights (3.52), (3.53) with $b_p = 0$, and continue with minimum-weight matching decoding described in Sec. 3.5.3.

The numerical results obtained with the described decoder at $\sigma = \sigma_{\text{M}} = \sigma_{\text{T}}$ are shown in Fig. 3.12 (left). Despite the fact that this decoder does not make a particularly good use of the GKP syndrome information, $\boldsymbol{q}^{\text{GKP}}$, and does not even try to find a good candidate error, the logical error rate rapidly goes down with increasing code distance and decreasing σ_0 below the crossing point at $\sigma_0 \approx 0.243$. With the forward-minimization decoder on a single GKP qubit with $\sigma = \sigma_{\text{M}}$, this would correspond to a logical error rate of $p \approx 1.3\%$. If the errors only come from having imperfect GKP states, then this also can be translated to having states with at least 4 photons.

It seems possible that, by making a better use of the GKP syndrome, one should be able to improve this decoder while preserving its computational efficiency. To this end, we tried a *preprocessing* algorithm. The basic idea is, given the syndrome $\boldsymbol{q}^{\text{GKP}}$, to find an initial approximation, $\boldsymbol{e}_0$, for the data errors which would bring back the GKP qubits closer to their code space. Given $\boldsymbol{e}_0$, Algorithm 1 can be used to find an error $\boldsymbol{e}_1$ matching the updated syndrome, after which the RPGM weights can be computed using the full candidate error $\boldsymbol{e}_0 + \boldsymbol{e}_1$. The hope is then that the candidate error found is one for which the first term of the Hamiltonian in Eq (3.54) does not need to be minimized anymore. In particular, we tried using our single-oscillator forward-minimization decoder from Sec. 3.3 as the preprocessing step. Using it directly produced a degradation of performance, seemingly resulting from the fact that the minimization of the RPGM Hamiltonian also tries to optimize the GKP measurement errors δ_b. Our solution was to drop

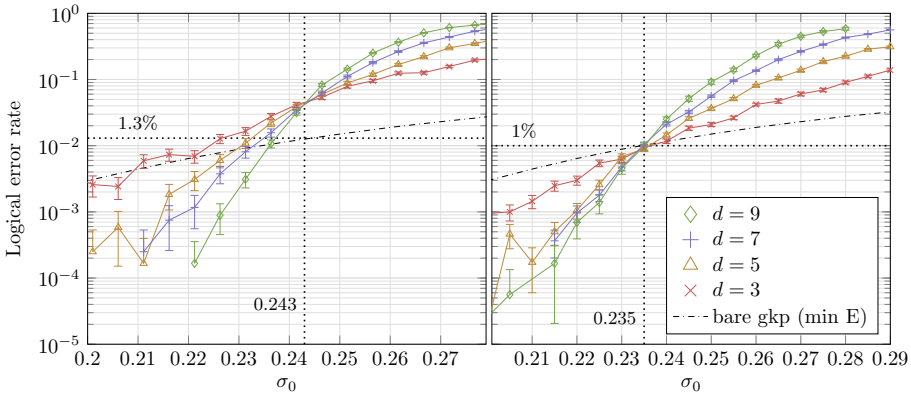


Figure 3.12: (Color online) For both plots the dash-dotted line shows the logical error rate per round for a single GKP qubit corrected with the forward-minimization decoder, with $\sigma = \sigma_T = \sigma_M$, as a function of the bare standard deviation $\sigma_0 = \sigma/2\sqrt{\pi}$. (Left) Numerical results for ALGORITHM 1 without preprocessing of the GKP syndrome information (see text). We observe a crossing point at $\sigma_0 \approx 0.243$ which can be translated to $p = 1.3\%$. (Right) Numerical results for the ALGORITHM 2 with preprocessing of GKP syndrome information (see text). We observe that this decoder improves the logical error rates. On the other hand, the improvement being greater for smaller distances, the crossing point moves left to $\sigma_0 \approx 0.235$ which corresponds to an error rate $p = 1\%$.

the measurement errors from the decomposition of the field in (3.51), which results in RPGM weights identical to those in Sec. 3.5.3. Our second decoder can then be summarized as follows.

ALGORITHM 2:

1. For each bond (ij), use the forward-minimization decoder from $t = 1$ up to $M-1$ to calculate the accumulated data error $\phi_{ij}(t)$, calculate the corresponding $\phi_{ij}(t = M)$, and then go back from $t = M$ to $t = 1$ with a version of the same algorithm, but using previously found values for a more accurate minimization. This gives the data errors $\boldsymbol{\epsilon}_{ij}(t) = \phi_{ij}(t) - \phi_{ij}(t-1)$, which we use to define the error vector $\boldsymbol{e}_0 = \{\mathbf{0}, \boldsymbol{\epsilon}, \mathbf{0}\}$.

2. Calculate the residual syndrome, and run the entire ALGORITHM 1, to calculate the corresponding error $\boldsymbol{e}_1$.

3. Calculate RPGM weights $\tau_p(\boldsymbol{e})$ using Eqs. (3.52), (3.53) with the combined error $\boldsymbol{e} = \boldsymbol{e}_0 + \boldsymbol{e}_1$ and $\boldsymbol{\delta}$ set to zero.

4. Use minimum-weight matching to minimize the RPGM Hamiltonian, and update the error $\boldsymbol{e}$, with the result being the output of the decoder.

The results are shown in Fig. 3.12 (right). One can observe that for each distance, there is an improvement in the encoded logical error rate, compared to the results from Algorithm 1 on the left of Fig. 3.12. One can see, for each curve, a higher *pseudo-threshold*, i.e. the point below which the logical error rate becomes smaller than the physical one,

determined using the logical error rate for a single GKP qubit given by the forward-minimization decoder. However, this improvement is greater for smaller distances, so the overall crossing point is shifted to the left, indicating a smaller threshold. Specifically, the crossing point is at $\sigma_0 \approx 0.235$ which corresponds to $p = 1\%$ for the single GKP qubit with the forward-minimization decoder.

We should note that even though both simple decoders we tried result in finite thresholds, with substantial reduction of the logical error rates with increasing distance below the crossing points, one could expect better performance. Indeed, the toric code with a phenomenological error model shows a threshold at $p = 2.9\%$. The forward-minimization decoder with $\sigma = \sigma_M$ in Sec. 3.3 reaches this error rate at $\sigma_0 \approx 0.28$.

Of course, a direct comparison is technically incorrect: forward-minimization or any other single-oscillator GKP decoder would return highly correlated errors and one cannot expect that the toric code would achieve the same performance. Nevertheless, we expect that adding a minimization step for the continuous part of the potential $\boldsymbol{a}$ in Eq. (3.51) would significantly improve the performance.

3.6. NO-GO RESULT FOR LINEAR OSCILLATOR CODES

We turn to the class of codes defined by continuous subgroups of displacement operators. One can think of linear combinations of position and momentum operators as nullifiers for the code space, i.e. any code state is annihilated by these nullifiers, see Appendix 1.2.3.

It is known that one cannot distill entanglement from Gaussian states by means of purely Gaussian local operations and classical communication [40], [41]. In addition, the authors of Ref. [19] defined a quantity, “entanglement degradation", for any single-mode Gaussian channel, such as the Gaussian displacement channel, and showed that it cannot decrease under Gaussian encoding and decoding. In the setting here we consider any input state which is perfectly encoded into a linear oscillator code. This encoding map, $\mathcal{E}$, is a Gaussian operation as it is a linear transformation of the $\hat{p}$ and $\hat{q}$ variables. After this encoding, the modes go through the Gaussian displacement channel $\mathcal{N}$, see Eq. (3.5). After this, linear combinations of $\hat{p}$ and $\hat{q}$ are measured to give rise to a syndrome $\boldsymbol{q}$. Again, this operation is Gaussian. Our results thus follow the model considered in [19]. However, our results do not require the definition of a new quantity but rather give a description of the logical noise model of Eq. (3.10). Namely, we show in the following theorem, that it can only lead to an effective squeezing of the original Gaussian displacement channel. Hence, whatever protection is gained in one quadrature, is lost in the other quadrature. This is a property of all linear oscillator codes, CSS or non-CSS, i.e. mixing $\hat{p}$ and $\hat{q}$ quadratures or not, with respect to the Gaussian displacement channel. Since the result is a detailed expression of the logical Gaussian displacement channel, it does not immediately follow from earlier no-go results.

Theorem (No-Go)**.** *Let $\mathscr{C}$ be a linear oscillator code on n physical oscillators defined by a set of $n-k$ independent nullifiers, thus encoding k logical oscillators. Let this code undergo independent Gaussian shifts in $\hat{p}$ and $\hat{q}$ of variance σ_0^2 on each of its physical oscillators, followed by a perfect (maximum-likelihood) decoding step. Then the remaining*

logical displacement noise model, Eq. (3.10)*, for logical shift errors* $\boldsymbol{\epsilon} \in \mathbb{R}^{2k}$ *is*

$$\mathbb{P}^{(\mathrm{ML})}(\boldsymbol{\epsilon}) = (2\pi\sigma_0^2)^{-2k} \exp\left(-\frac{1}{2\sigma_0^2}\boldsymbol{\epsilon}\,\Sigma^{-1}\boldsymbol{\epsilon}^T\right),$$

and the eigenvalues of the covariance matrix, Σ^{-1}*, are paired by conjugated logical operators,* $(\lambda_j^p, \lambda_j^q)_{j\in[k]}$*, such that*

$$\forall\, j \in [k],\ \lambda_j^p \lambda_j^q = 1.$$

In particular one has $\det\Sigma^{-1} = 1$.

The proof of this theorem, which is rather lengthy, can be found in Section 3.7. The remaining logical displacement noise is obtained by working out Eq. (3.10), using general properties of linear oscillator codes. The theorem says that for each logical mode, in the basis given by the eigenvectors of Σ, the only effect of the encoding is to squeeze the displacement noise model between the conjugated operators of the mode. The amount of squeezing can depend on code size but it is impossible to reduce the noise in both quadratures.

As a concrete example, we consider the linear oscillator code version of the 2D toric code in Section 3.8 and show that the squeezing depends simply on the ratio L_x/L_y for a $L_x \times L_y$ toric lattice.

3.7. PROOF OF NO-GO THEOREM

For convenience, we rename the $\hat{p}_j$ and $\hat{q}_j$ operators of the oscillators as $\hat{r}_k$, i.e.

$$\forall\, k \in [2n], \quad \hat{r}_k = \begin{cases} \hat{p}_k & \text{when } k \le n \\ \hat{q}_{k-n} & \text{when } k > n. \end{cases}$$

Let $\boldsymbol{g}_j$ be the real vector corresponding to the jth nullifier $\hat{O}(\boldsymbol{g}_j) = \boldsymbol{g}_j \cdot \hat{\boldsymbol{r}}$ of the code, see Appendix 1.2.3. One can extend the set of nullifiers to a full canonical linear transformation given by a real $2n \times 2n$ matrix A defining new variables $\hat{R}_k$, as follows

$$\hat{\boldsymbol{R}} = A\hat{\boldsymbol{r}}^{\mathrm{T}}, \qquad \text{i.e. } \hat{R}_k = \sum_{j=1}^{2n} A_{kj}\hat{r}_j.$$

In order to preserve the commutation relation the condition on the matrix A is that it preserves the symplectic form S. This can be expressed in two ways:

$$ASA^{\mathrm{T}} = S \quad \text{or} \quad A^{\mathrm{T}}SA = S \quad \text{with} \quad S = \begin{pmatrix} 0 & \mathbb{1}_{n\times n} \\ -\mathbb{1}_{n\times n} & 0 \end{pmatrix}. \tag{3.55}$$

The matrix A can be decomposed in blocks

$$A = \begin{matrix} n-k\,\updownarrow \\ k\quad\updownarrow \\ n-k\,\updownarrow \\ k\quad\updownarrow \end{matrix} \overset{\leftarrow 2n \rightarrow}{\begin{pmatrix} G \\ P \\ D \\ Q \end{pmatrix}} = \overset{\leftarrow n \rightarrow \quad \leftarrow n \rightarrow}{\begin{pmatrix} G_p & G_q \\ P_p & P_q \\ D_p & D_q \\ Q_p & Q_q \end{pmatrix}},$$

where the rows of G are the nullifiers. The rows of D represent the corresponding conjugated variables which will be called *pure errors* (they are sometimes referred to as destabilizers). These pure errors give a convenient basis for expressing an error which is compatible with a given syndrome which will be used below. The rows of P resp. Q represent the logical $\hat{p}$ (resp. $\hat{q}$) operators of the code as linear combinations of the original $\hat{p}_i$ and $\hat{q}_j$. The subscript p (respectively q) indicates the $\hat{p}$ part (respectively $\hat{q}$ part) of the operators. Inside the code space the operators $\hat{O}\left(\boldsymbol{g}_j\right)$ only take the value 0. Let's assume that a displacement happens along a pure error direction $\boldsymbol{d}_j$, say $U\left(-\lambda \boldsymbol{d}_j\right)$. Then, measuring $\hat{O}(\boldsymbol{g}_j)$ (equivalently $U\left(\eta \boldsymbol{g}_j\right)$ for all $\eta \in \mathbb{R}$) would give outcome $\lambda \in \mathbb{R}$ called the syndrome, since

$$U(\eta \boldsymbol{g}_j)\left[U(-\lambda \boldsymbol{d}_j)|\Psi\rangle\right] = \mathrm{e}^{i\lambda\eta} U(-\lambda \boldsymbol{d}_j) U(\eta \boldsymbol{g}_j)|\Psi\rangle = \mathrm{e}^{i\lambda\eta}\left[U(-\lambda \boldsymbol{d}_j)|\Psi\rangle\right].$$

Note that the logical operators $\hat{O}(\boldsymbol{p}_k)$ or $\hat{O}(\boldsymbol{q}_\ell)$, or nullifiers, act on the code space without affecting the measurement of $\hat{O}(\boldsymbol{g}_j)$ since they commute with it.

We can express the constraints on the matrix A in Eq. (3.55) in terms of the matrices G, D, P and Q. The blocks which should be equal to zero are shown in blue and the blocks proportional to the identity are shown in green. The first condition gives

$$ASA^{\mathrm{T}} = \begin{pmatrix} GSG^{\mathrm{T}} & GSP^{\mathrm{T}} & GSD^{\mathrm{T}} & GSQ^{\mathrm{T}} \\ PSG^{\mathrm{T}} & PSP^{\mathrm{T}} & PSD^{\mathrm{T}} & PSQ^{\mathrm{T}} \\ DSG^{\mathrm{T}} & DSP^{\mathrm{T}} & DSD^{\mathrm{T}} & DSQ^{\mathrm{T}} \\ QSG^{\mathrm{T}} & QSP^{\mathrm{T}} & QSD^{\mathrm{T}} & QSQ^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} 0 & 0 & \mathbb{1} & 0 \\ 0 & 0 & 0 & \mathbb{1} \\ -\mathbb{1} & 0 & 0 & 0 \\ 0 & -\mathbb{1} & 0 & 0 \end{pmatrix} = S, \tag{3.56}$$

while the second implies

$$\mathbb{1} = A^{\mathrm{T}} S A S^{T} = \begin{pmatrix} D_q^{\mathrm{T}} G_p + Q_q^{\mathrm{T}} P_p - G_q^{\mathrm{T}} D_p - P_q^{\mathrm{T}} Q_p & D_q^{\mathrm{T}} G_q + Q_q^{\mathrm{T}} P_q - G_q^{\mathrm{T}} D_q - P_q^{\mathrm{T}} Q_q \\ -D_p^{\mathrm{T}} G_p - Q_p^{\mathrm{T}} P_p + G_p^{\mathrm{T}} D_p + P_p^{\mathrm{T}} Q_p & -D_p^{\mathrm{T}} G_q - Q_p^{\mathrm{T}} P_q + G_p^{\mathrm{T}} D_q + P_p^{\mathrm{T}} Q_q \end{pmatrix}. \tag{3.57}$$

3.7.1. LOGICAL ERROR MODEL UNDER DISPLACEMENT ERRORS

We consider the Gaussian displacement noise model on every oscillator as in Eq. (3.5). Given a realization of the displacement error as a vector of real amplitudes, $\boldsymbol{e}' \in \mathbb{R}^{2n}$, one can compute the vector of syndromes $\boldsymbol{q} \in \mathbb{R}^{n-k}$, and given $\boldsymbol{q}$, a candidate error with the same syndrome, $\boldsymbol{e} \in \mathbb{R}^{2n}$,

$$\boldsymbol{q} = \boldsymbol{e}' S G^{\mathrm{T}}, \qquad \boldsymbol{e} = -\boldsymbol{q} D.$$

In addition, when any two errors $\boldsymbol{e}$ and $\boldsymbol{e}'$ have the same syndrome $\boldsymbol{q}$, then they can only differ by a stabilizer and a logical operator and one has

$$\exists \boldsymbol{u}_c, \boldsymbol{v}_c \in \mathbb{R}^k, \exists \boldsymbol{a} \in \mathbb{R}^{n-k}, \quad \boldsymbol{e}' = \boldsymbol{e} + \boldsymbol{u}_c P + \boldsymbol{v}_c Q + \boldsymbol{a} G = \boldsymbol{e} + \boldsymbol{c} C + \boldsymbol{a} G.$$

where we have used the notation

$$C := \begin{pmatrix} P \\ Q \end{pmatrix}, \qquad \boldsymbol{c} = \begin{pmatrix} \boldsymbol{u}_c & \boldsymbol{v}_c \end{pmatrix}.$$

We can compute the associated partition function, $Z_{\boldsymbol{c}}(\boldsymbol{e})$ from Eq. (3.7), for the error $\boldsymbol{e}$ and the sector equivalent to $\boldsymbol{c}$,

$$Z_{\boldsymbol{c}}(\boldsymbol{e}) = \left(\frac{1}{\sqrt{2\pi\sigma_0^2}}\right)^n \int \prod_{j=0}^{n-k-1} \mathrm{d}a_j \exp\left(-\frac{(\boldsymbol{c}C + \boldsymbol{a}G + \boldsymbol{e})\,(\boldsymbol{c}C + \boldsymbol{a}G + \boldsymbol{e})^{\mathrm{T}}}{2\sigma_0^2}\right). \tag{3.58}$$

This integral can be evaluated since it is Gaussian, resulting, after some manipulations, in

$$Z_{\boldsymbol{c}}(\boldsymbol{e}) = \mathscr{C}(\boldsymbol{e}) \exp\left(-\frac{1}{2\sigma_0^2}\Big(\boldsymbol{c} - \boldsymbol{\mu}(\boldsymbol{e})\Big)\Sigma^{-1}\Big(\boldsymbol{c} - \boldsymbol{\mu}(\boldsymbol{e})\Big)^{\mathrm{T}}\right), \tag{3.59}$$

where $\mathscr{C}(\boldsymbol{e})$ only depends on $\boldsymbol{e}$. The covariance matrix Σ and the off-set vector, $\boldsymbol{\mu}(\boldsymbol{e})$, are defined as

$$\Sigma^{-1} = C'C'^{\mathrm{T}}, \quad \text{and} \quad \boldsymbol{\mu}(\boldsymbol{e}) = -\Sigma C' \boldsymbol{e}^{\mathrm{T}} \tag{3.60}$$

where

$$C' = C\Pi_{G^\perp}, \quad \text{and} \quad \Pi_{G^\perp} = \mathbb{1} - G^T(GG^T)^{-1}G. \tag{3.61}$$

Remark that $\Pi_{G^\perp}$ is the projector onto $\ker(G)$ along $\mathrm{im}(G^{\mathrm{T}})$ [5]. Indeed it is easy to check that: $\Pi_{G^\perp}^2 = \Pi_{G^\perp}$, $\ker(G) \subset \mathrm{im}(\Pi_{G^\perp})$, $\mathrm{im}(G^{\mathrm{T}}) \subset \ker(\Pi_{G^\perp})$, and the dimensions coincide.

One can see that Eq. (3.59) describes a multivariate Gaussian distribution over the logical variable $\boldsymbol{c}$, hence its maximum is readily given by the mean value, $\boldsymbol{\mu}(\boldsymbol{e})$, see Eq. (3.60). This means that given the error $\boldsymbol{e}$, one can directly express its most likely error class: it is $[\boldsymbol{e} + \boldsymbol{\mu}(\boldsymbol{e})]$. Using this, one can directly compute the probability density of a remaining logical error after ML decoding as given by Eq. (3.10):

$$\boxed{\mathbb{P}^{(\mathrm{ML})}(\boldsymbol{c}) = \int \mathrm{d}\boldsymbol{e}\, \mathbb{P}(\boldsymbol{e}) \frac{Z_{\boldsymbol{\mu}(\boldsymbol{e})+\boldsymbol{c}}(\boldsymbol{e})}{\int \mathrm{d}\boldsymbol{b}\, Z_{\boldsymbol{b}}(\boldsymbol{e})} = \frac{1}{N} \exp\left(-\frac{1}{2\sigma_0^2}\boldsymbol{c}\Sigma^{-1}\boldsymbol{c}^{\mathrm{T}}\right),} \tag{3.62}$$

where N is a normalization constant. All possible dependence on the choice and size of the code is contained in the covariance matrix Σ, leading to some rescaled displacement noise model. Recall that the logical variable vector $\boldsymbol{c} \in \mathbb{R}^{2k}$ represents all k pairs of conjugated logical operators. We will prove in Section 3.7.2 below that for any linear oscillator code, the eigenvalues of Σ^{-1} can be paired between corresponding conjugated logical operators, denoted $(\lambda_j^p, \lambda_j^q)_{j\in[n]}$, and are such that

$$\forall\, j \in [n],\ \lambda_j^p \lambda_j^q = 1. \tag{3.63}$$

This means that the noise remaining after ML decoding on the logical variables is identical to the original physical noise except for a possible squeezing between each logical operator and its conjugated pair.

[5] This is not an orthogonal projection as $\ker(G)$ and $\mathrm{im}(G^{\mathrm{T}})$ are not orthogonal to one another.

3.7.2. EIGENVALUES OF THE COVARIANCE MATRIX

First we remark that C', appearing in $\Sigma^{-1} = C'C'^{\mathrm{T}}$ in Eq. (3.60), corresponds to a valid basis for the logical operators; we call this the spread-out logical basis. This basis is obtained by adding linear combination of stabilizer generators to C. Such addition can be summarized, using Eq. (3.61), by the matrix equation,

$$C' = C - CG^{\mathrm{T}}\left(GG^{\mathrm{T}}\right)^{-1} G = C + \Lambda G,$$

where Λ is a $2k \times (n-k)$ matrix defining the linear combination of stabilizer generators added to the logical operators. One can verify that one can replace C by C' and still satisfy the constraints of Eq. (3.55), if one appropriately redefines the pure errors to be

$$D' = D + \Lambda^{\mathrm{T}} S_{2k}^{\mathrm{T}} C,$$

where S_{2k} is the symplectic form of size $2k \times 2k$. One can also check that this choice of basis is the only choice which enforces the following constraint

$$C'G^{\mathrm{T}} = (C + \Lambda G)\, G^{\mathrm{T}} = 0. \tag{3.64}$$

Indeed, solving for Λ, using the fact that G is full rank and therefore GG^{T} is invertible, gives

$$C' = C - CG^{T}\left(GG^{\mathrm{T}}\right)^{-1} G. \tag{3.65}$$

Now we use this spread-out logical basis to prove Eq. (3.63). We thus consider that we already have chosen the spread-out logical basis, so $C' = C$, and want to get information about the eigenvalues of $\Sigma^{-1} = CC^{\mathrm{T}}$.

We use the diagonal block of Eq. (3.57) as well as the block only about logical operators in Eq. (3.56):

$$\begin{cases} -D_p^{\mathrm{T}} G_q - Q_p^{\mathrm{T}} P_q + G_p^{\mathrm{T}} D_q + P_p^{\mathrm{T}} Q_q = \mathbb{1} & (3.66) \\ D_q^{\mathrm{T}} G_p + Q_q^{\mathrm{T}} P_p - G_q^{\mathrm{T}} D_p - P_q^{\mathrm{T}} Q_p = \mathbb{1} & (3.67) \\ PSQ^{\mathrm{T}} = P_p Q_q^{\mathrm{T}} - P_q Q_p^{\mathrm{T}} = \mathbb{1}, & (3.68) \end{cases}$$

We multiply Eq. (3.66) on the left by P_p and on the right by Q_q^T. We also multiply Eq. (3.67) on the left by P_q and on the right by Q_p^T. Then we take the difference, i.e. we consider

$$P_p(3.66)Q_q^{\mathrm{T}} - P_q(3.67)Q_p^{\mathrm{T}}.$$

By Eq. (3.68), the right hand-side is still the identity, so we have

$$\begin{aligned} \mathbb{1} = \quad & P_p G_p^{\mathrm{T}} D_q Q_q^{\mathrm{T}} + P_q G_q^{\mathrm{T}} D_p Q_p^{\mathrm{T}} \\ - \; & P_q D_q^{\mathrm{T}} G_p Q_p^{\mathrm{T}} - P_p D_p^{\mathrm{T}} G_q Q_q^{\mathrm{T}} \\ + \; & P_p P_p^{\mathrm{T}} Q_q Q_q^{\mathrm{T}} + P_q P_q^{\mathrm{T}} Q_p Q_p^{\mathrm{T}} \\ - \; & P_p Q_p^{\mathrm{T}} P_q Q_q^{\mathrm{T}} - P_q Q_q^{\mathrm{T}} P_p Q_p^{\mathrm{T}} \end{aligned}.$$

Since the logical operators are in their spread-out basis we have the corresponding equations

$$\begin{cases} P_p G_p^{\mathrm{T}} + P_q G_q^{\mathrm{T}} = 0 \\ G_p Q_p^{\mathrm{T}} + G_q Q_q^{\mathrm{T}} = 0, \end{cases}$$

and can write

$$\begin{aligned} \mathbb{1} = & P_q \left(D_q^{\mathrm{T}} G_q - G_q^{\mathrm{T}} D_q \right) Q_q^{\mathrm{T}} \\ & + P_p \left(D_p^{\mathrm{T}} G_p - G_p^{\mathrm{T}} D_p \right) Q_p^{\mathrm{T}} \\ & + P_p \left(P_p^{\mathrm{T}} Q_q - Q_p^{\mathrm{T}} P_q \right) Q_q^{\mathrm{T}} \\ & + P_q \left(P_q^{\mathrm{T}} Q_p - Q_q^{\mathrm{T}} P_p \right) Q_p^{\mathrm{T}}. \end{aligned}$$

One can now recognize that the off-diagonal terms in Eq. (3.57) can be used to make an equation only about logical variables, using

$$\begin{cases} D_q^{\mathrm{T}} G_q - G_q^{\mathrm{T}} D_q = P_q^{\mathrm{T}} Q_q - Q_q^{\mathrm{T}} P_q \\ D_p^{\mathrm{T}} G_p - G_p^{\mathrm{T}} D_p = P_p^{\mathrm{T}} Q_p - Q_p^{\mathrm{T}} P_p. \end{cases}$$

Hence we have the following matrix identity

$$\mathbb{1} = P P^{\mathrm{T}} Q Q^{\mathrm{T}} - P Q^{\mathrm{T}} P Q^{\mathrm{T}} \tag{3.69}$$

In the next section, 3.7.3, we show that there always exists a logical basis which is both spread-out as well as orthogonal. Hence for this new basis, Eq. (3.64) is satisfied, as well as

$$\tilde{P}\tilde{P}^{\mathrm{T}} = \mathrm{Diag}\left(\lambda_j^p\right), \quad \tilde{Q}\tilde{Q}^{\mathrm{T}} = \mathrm{Diag}\left(\lambda_j^q\right), \quad \text{and} \quad \tilde{P}\tilde{Q}^{\mathrm{T}} = 0. \tag{3.70}$$

Therefore deriving Eq. (3.69) in this basis yields

$$\mathbb{1} = \tilde{P}\tilde{P}^{\mathrm{T}}\tilde{Q}\tilde{Q}^{\mathrm{T}} = \mathrm{Diag}\left(\lambda_j^p \lambda_j^q\right), \tag{3.71}$$

and therefore

$$\forall\, j \in [k],\ \lambda_j^p \lambda_j^q = 1. \tag{3.72}$$

Lastly recall that starting with this choice of basis, one has

$$\Sigma^{-1} = \tilde{C}\tilde{C}^{\mathrm{T}} = \begin{pmatrix} \tilde{P}\tilde{P}^{\mathrm{T}} & \tilde{P}\tilde{Q}^{\mathrm{T}} \\ \tilde{Q}\tilde{P}^{\mathrm{T}} & \tilde{Q}\tilde{Q}^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} \mathrm{Diag}\left(\lambda_j^p\right) & 0 \\ 0 & \mathrm{Diag}\left(\lambda_j^q\right) \end{pmatrix}. \tag{3.73}$$

□

3.7.3. EXISTENCE OF A SPREAD-OUT & ORTHOGONAL LOGICAL OPERATOR BASIS

We start from a given stabilizer matrix G and denote the rows by $\{\boldsymbol{g}_1,\ldots,\boldsymbol{g}_{n-k}\}$. We showed above that one can always find a logical basis, P and Q with rows $\{\boldsymbol{p}_1,\ldots,\boldsymbol{p}_k\}$ and $\{\boldsymbol{q}_1,\ldots,\boldsymbol{q}_k\}$, which is orthogonal to the stabilizer generators, i.e

$$\mathscr{L}_k = \text{Span}\{\boldsymbol{p}_1,\ldots,\boldsymbol{p}_k,\boldsymbol{q}_1,\ldots,\boldsymbol{q}_k\} \perp \mathscr{G} = \text{Span}\{\boldsymbol{g}_1,\ldots,\boldsymbol{g}_{n-k}\}.$$

We want to construct a new symplectic and orthogonal basis, $\{\tilde{\boldsymbol{p}}_1,\ldots,\tilde{\boldsymbol{p}}_k,\tilde{\boldsymbol{q}}_1,\ldots,\tilde{\boldsymbol{q}}_k\}$, for the $2k$-dimensional space $\mathscr{L}_k$. One can first find an orthogonal basis for $\mathscr{L}_k$ by the Gram-Schmidt process for the regular inner product, denote it $\{\boldsymbol{c}_1^k,\ldots,\boldsymbol{c}_{2k}^k\}$. We will now construct a new basis pair by pair, decreasing at each step the dimension of the space $\mathscr{L}_k$ by 2. We can choose

$$\tilde{\boldsymbol{p}}_1 = \boldsymbol{c}_1^k.$$

There has to exist a conjugated pair, $\tilde{\boldsymbol{q}}_1$, in the space spanned by $\{\boldsymbol{c}_2^k,\ldots,\boldsymbol{c}_{2k}^k\}$. Indeed, all stabilizer generators, pure errors as well as $\boldsymbol{c}_1^k$ have a trivial symplectic product with $\tilde{\boldsymbol{p}}_1$, hence any conjugated $\tilde{\boldsymbol{q}}_1 \in \text{Span}\{\boldsymbol{c}_2^k,\ldots,\boldsymbol{c}_{2k}^k\}$. So the following equation

$$\boldsymbol{\lambda} O_k S \tilde{\boldsymbol{p}}_1^{\mathrm{T}} = -1,$$

where O_k are the basis vectors $\{\boldsymbol{c}_2^k,\ldots,\boldsymbol{c}_{2k}^k\}$ stacked in rows, has a solution for $\boldsymbol{\lambda}$. Then we can choose

$$\tilde{\boldsymbol{q}}_1 = \boldsymbol{\lambda} O_k.$$

Note that the created pair is indeed conjugated for the symplectic inner product as well as orthogonal. Moreover they are composed only of linear combinations of the original $\boldsymbol{p}_i$s and $\boldsymbol{q}_j$s, so they do have trivial symplectic product with stabilizer generators and pure errors and they are orthogonal to the stabilizer generators. Now we define the $2(k-1)$-dimensional subspace $\mathscr{L}_{k-1}$, as

$$\begin{aligned}\mathscr{L}_{k-1} &= \text{Span}\left\{\boldsymbol{c}_2^k\left(\mathbb{1}-\Pi_{\tilde{\boldsymbol{q}}_1}\right),\ldots,\boldsymbol{c}_{2k}^k\left(\mathbb{1}-\Pi_{\tilde{\boldsymbol{q}}_1}\right)\right\}\\ &= \text{Span}\left\{\boldsymbol{c}_1^{k-1},\ldots,\boldsymbol{c}_{2(k-1)}^{k-1}\right\},\end{aligned}$$

where $\Pi_{\tilde{\boldsymbol{q}}_1}$ is the orthogonal projector onto $\tilde{\boldsymbol{q}}_1$, and the $\boldsymbol{c}_j^{k-1}$ form a new orthogonal basis for this space. We can then repeat the procedure until we reach $\mathscr{L}_0 = \{0\}$. To summarize we have constructed a new symplectic and orthogonal basis $\tilde{C} = \begin{pmatrix}\tilde{P}\\ \tilde{Q}\end{pmatrix}$ obeying Eq. (3.70).

3.8. CONTINUOUS-VARIABLE TORIC-CODE

In this section we examine the continuous-variable toric code [35, 42] as an example of a continuous-variable topological code. Since the toric code is a homological code, it is easy to convert it from a $\mathbb{Z}_2$-code to an $\mathbb{R}$-code using orientation to add appropriate minus signs to the stabilizer checks.

The stabilizer checks are shown on the left in Fig. 3.13. Since the toric code is a CSS code we will always have the orthogonality condition $PQ^{\mathrm{T}} = 0$. The vectors $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$

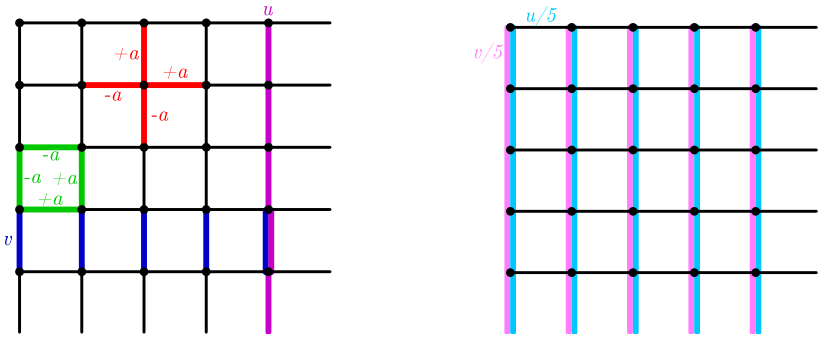


Figure 3.13: In both figures periodic boundary conditions are assumed. (Left) Example of stabilizer checks and logical operators for the distance-5 continuous-variable toric-code. The stabilizer checks are also shown in Fig. 3.8. The support of the logical $\boldsymbol{p_1}$ (in dark blue) and $\boldsymbol{q}_1$ (in dark purple) of one of the encoded oscillators is depicted. Shifts of strength v on the support of $\boldsymbol{p}_1$ is a logical $\boldsymbol{p}$-shift of strength v of the first oscillator. (Right) The support of the spread-out version of the logical operators $\boldsymbol{p}_1$ and $\boldsymbol{q}_1$. Both operators have the same support (in light blue and light purple) and one can verify that these logical operators are orthogonal to the stabilizer checks as well as commuting with them. A logical $\boldsymbol{p}$-shift of strength v on the first oscillator is now realized by applying $v/5$ on the support of the spread-out $\boldsymbol{p}_1$. In general, if the torus had dimension $L_x \times L_y$, the spread-out $\boldsymbol{p}_1$ would have a shift rescaling of $\frac{1}{L_x}$ over the whole lattice while $\boldsymbol{q}_1$ would a shift rescaling of $\frac{1}{L_y}$. At the same time $\boldsymbol{p}_2$ (resp. $\boldsymbol{q}_2$) would have rescaling $\frac{1}{L_y}$ (resp. $\frac{1}{L_x}$).

are the two rows of P while the vectors $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ are the two rows of Q. On the left, one sees the usual string-like $\boldsymbol{p}_1$ and $\boldsymbol{q}_1$. The spread-out basis for the code can be computed and is shown on the right of Fig. 3.13. The name *spread-out basis* comes from the fact that these logical operators have support over the full lattice: the continuous-variable stabilizer checks have been used to spread out or distribute their support over the lattice.

Computing CC^T, Eq. (3.73), with C being the spread-out logical operators in the general case of different dimensions L_x and L_y gives directly a diagonal matrix:

$$CC^T = \text{diag}\left(\lambda_1^p, \lambda_2^p, \lambda_1^q, \lambda_2^q\right) = \text{diag}\left(\boldsymbol{p}_1^2, \boldsymbol{p}_2^2, \boldsymbol{q}_1^2, \boldsymbol{q}_2^2\right), \tag{3.74}$$

with

$$\lambda_1^p = \boldsymbol{p}_1^2 = \sum_{e_x} \frac{1}{L_y^2} = \frac{L_x L_y}{L_y^2} = \frac{L_x}{L_y}, \qquad \lambda_2^p = \boldsymbol{p}_2^2 = \sum_{e_y} \frac{1}{L_x^2} = \frac{L_x L_y}{L_x^2} = \frac{L_y}{L_x},$$
$$\lambda_1^q = \boldsymbol{q}_1^2 = \sum_{e_x} \frac{1}{L_x^2} = \frac{L_x L_y}{L_x^2} = \frac{L_y}{L_x}, \qquad \lambda_2^q = \boldsymbol{q}_2^2 = \sum_{e_y} \frac{1}{L_y^2} = \frac{L_x L_y}{L_y^2} = \frac{L_x}{L_y}.$$

One sees that it is possible to choose the squeezing amount by choosing the ratio of the dimensions L_x/L_y. In particular, with Eq. (3.62), one sees that the first encoded oscillator experiences a Gaussian displacement noise model with variance $\sigma_p^2 = \frac{L_y}{L_x}\sigma_0^2$ for $\hat{p}$ and variance $\sigma_q^2 = \frac{L_x}{L_y}\sigma_0^2$ for $\hat{q}$. For the second oscillator the quadrature squeezing goes in the other direction.

3.9. DISCUSSION

In this chapter we have presented the first strides in tackling the problem of error correction and decoding for the toric-GKP code. Interestingly, the decoding problem maps onto a new class of physical continuous-variable models with quenched-disorder, going beyond the random plaquette gauge model corresponding to toric code decoding [38]. We have presented an efficient minimum-energy decoding method for a single GKP oscillator and an efficient decoder for the toric-GKP code. We have also presented a combination of these two decoders for the toric-GKP code which improves the achieved logical error rates but in greater proportion for the small distances, hence achieving better pseudo-thresholds but a slightly worse threshold. It would be interesting to design a better decoder to improve this threshold.

An interesting open problem is to study the phase diagram of the toric-GKP code numerically. An example of such a numerical study for 3D color codes is [43].

Future studies could look at the question of decoding coherent errors and/or correcting both p and q-shifts simultaneously. Another question is whether it is possible to handle more realistic noise models, e.g. consider a model of repeated photon loss [6] for a single GKP oscillator. All such different error models will have a particular path integral representation and the idea of choosing an energy-minimizing path can be examined.

A variant of the toric-GKP code is the toric-rotor code. This is the concatenation of a rotor space with integer $\hat{n}$ and 2π-periodic $\hat{\varphi}$ with a rotor toric code whose nullifiers are linear combinations of $\hat{n}$s and $\hat{\varphi}$s of four rotors [44]. When one uses a two-dimensional rotor subspace such as a cat code [13] or just a transmon qubit, one could still express the proper toric code checks in the entire rotor space and examine when a memory phase is present. Note that the difference in such analysis versus the usual toric code analysis is that in this model the effect of leakage errors is automatically included.

A possible realization of a surface code variant of the toric-GKP code is an array of superconducting 2D or 3D resonators. For a code such as Surface-17, this would require $2 \times 17 = 34$ (coplanar) microwave resonators. One can compare this to the transmon + resonators lay-out for the regular surface code [45], [46] in which each CNOT gate is mediated via a coupling bus, hence one uses $8 \times 4 = 32$ such bus resonators for Surface-17.

REFERENCES

[1] C. Vuillot, H. Asasi, Y. Wang, L. P. Pryadko, and B. M. Terhal, *Quantum error correction with the toric Gottesman-Kitaev-Preskill code,* Physical Review A **99**, 032344 (2019).

[2] S. Braunstein, *Error correction for continuous quantum variables,* Phys. Rev. Lett. **80**, 4084 (1998).

[3] S. Lloyd and J.-J. Slotine, *Analog Quantum Error Correction,* Phys. Rev. Lett. **80**, 4088 (1998).

[4] D. Gottesman, A. Kitaev, and J. Preskill, *Encoding a qubit in an oscillator,* Phys. Rev. A **64**, 012310 (2001).

[5] M. H. Michael, M. Silveri, R. T. Brierley, V. V. Albert, J. Salmilehto, L. Jiang, and S. M. Girvin, *New Class of Quantum Error-Correcting Codes for a Bosonic Mode,* Phys. Rev. X **6**, 031006 (2016).

[6] V. V. Albert, K. Noh, K. Duivenvoorden, D. J. Young, R. T. Brierley, P. Reinhold, C. Vuillot, L. Li, C. Shen, S. M. Girvin, B. M. Terhal, and L. Jiang, *Performance and structure of single-mode bosonic codes,* Physical Review A **97**, 032346 (2018).

[7] C. Weedbrook, S. Pirandola, R. García-Patrón, N. J. Cerf, T. C. Ralph, J. H. Shapiro, and S. Lloyd, *Gaussian quantum information,* Rev. Mod. Phys. **84**, 621 (2012).

[8] B. M. Terhal and D. Weigand, *Encoding a qubit into a cavity mode in circuit QED using phase estimation,* Phys. Rev. A **93**, 012315 (2016).

[9] Y. Y. Gao, B. J. Lester, Y. Zhang, C. Wang, S. Rosenblum, L. Frunzio, L. Jiang, S. M. Girvin, and R. J. Schoelkopf, *Programmable Interference between Two Microwave Quantum Memories,* Phys. Rev. X **8**, 021073 (2018).

[10] C. Flühmann, V. Negnevitsky, M. Marinelli, and J. P. Home, *Sequential Modular Position and Momentum Measurements of a Trapped Ion Mechanical Oscillator,* Phys. Rev. X **8**, 021001 (2018).

[11] K. R. Motes, B. Q. Baragiola, A. Gilchrist, and N. C. Menicucci, *Encoding qubits into oscillators with atomic ensembles and squeezed light,* \pra **95**, 053819 (2017).

[12] N. C. Menicucci, *Fault-tolerant measurement-based quantum computing with continuous-variable cluster states,* Phys. Rev. Lett. **112**, 120504 (2014).

[13] N. Ofek, A. Petrenko, R. Heeres, P. Reinhold, Z. Leghtas, B. Vlastakis, Y. Liu, L. Frunzio, S. M. Girvin, L. Jiang, M. Mirrahimi, M. H. Devoret, and R. J. Schoelkopf, *Extending the lifetime of a quantum bit with error correction in superconducting circuits,* Nature **536**, 441 (2016).

[14] K. Fukui, A. Tomita, and A. Okamoto, *Analog Quantum Error Correction with Encoding a Qubit into an Oscillator,* Physical Review Letters **119**, 180507 (2017).

[15] Y. Wang, *Quantum Error Correction with the GKP Code and Concatenation with Stabilizer Codes,* Master's thesis, RWTH Aachen University (2017).

[16] L. P. Pryadko, H. Asasi, M. Mulligan, and A. A. Kovalev, *Maximum likelihood decoding threshold for oscillator quantum error correcting codes* (2017).

[17] K. Fukui, A. Tomita, A. Okamoto, and K. Fujii, *High-Threshold Fault-Tolerant Quantum Computation with Analog Quantum Error Correction,* Phys. Rev. X **8**, 021054 (2018).

[18] K. Duivenvoorden, B. M. Terhal, and D. Weigand, *Single-mode displacement sensor,* Phys. Rev. A **95**, 012305 (2017).

[19] J. Niset, J. Fiuráek, and N. J. Cerf, *No-Go Theorem for Gaussian Quantum Error Correction,* Phys. Rev. Lett. **102**, 120501 (2009).

[20] D. Gottesman, *Stabilizer codes and quantum error correction*, PhD Thesis, California Institute of Technology (1997).

[21] E. Hostens, J. Dehaene, and B. De Moor, *Stabilizer states and Clifford operations for systems of arbitrary dimensions and modular arithmetic*, Phys. Rev. A **71**, 042315 (2005).

[22] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes* (North-Holland, Amsterdam, 1981).

[23] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, *Topological quantum memory*, Journal of Mathematical Physics **43**, 4452 (2002).

[24] A. A. Kovalev and L. P. Pryadko, *Spin Glass Reflection of the Decoding Transition for Quantum Error Correcting Codes*, Quantum Info. Comput. **15**, 825 (2015).

[25] I. Dumer, A. A. Kovalev, and L. P. Pryadko, *Thresholds for Correcting Errors, Erasures, and Faulty Syndrome Measurements in Degenerate Quantum Codes*, Phys. Rev. Lett. **115**, 050502 (2015).

[26] A. J. Landahl, J. T. Anderson, and P. R. Rice, *Fault-tolerant quantum computing with color codes*, (2011), arXiv:1108.5738 [quant-ph] .

[27] S. Glancy and E. Knill, *Error analysis for encoding a qubit in an oscillator*, Phys. Rev. A **73**, 012325 (2006).

[28] B. Deconinck, *Multidimensional Theta Functions*, published: NIST digital library of mathematical functions.

[29] B. Deconinck, M. Heil, Alexander Bobenko, M. van Hoeij, and M. Schmies, *Computing Riemann theta functions*, Math. Comp. **73**, 1417 (2004).

[30] J. Frauendiener, C. Jaber, and C. Klein, *Efficient computation of multidimensional theta functions*, (2017).

[31] J. Villain, *Theory of one- and two-dimensional magnets with an easy magnetization plane. II. The planar, classical, two-dimensional magnet*, Journal de Physique **36**, 581 (1975).

[32] W. Janke and H. Kleinert, *How good is the Villain approximation?* Nuclear Physics B **270**, 135 (1986).

[33] O. M. Braun and Y. S. Kivshar, *Nonlinear dynamics of the Frenkel-Kontorova model*, Physics Reports **306**, 1 (1998).

[34] Y. S. Kivshar, H. Benner, and O. M. Braun, *Nonlinear models for the dynamics of topological defects in solids. Nonlinear Science at the Dawn of the 21st Century.*, Lecture Notes in Physics, Vol. 542 (Springer, 2008).

[35] B. M. Terhal, *Quantum error correction for quantum memories*, Rev. Mod. Phys. **87**, 307 (2015).

[36] T. P. Peixoto, *The graph-tool Python library* (2014).

[37] J. Edmonds, *Paths, trees, and flowers,* Canadian Journal of mathematics **17**, 449 (1965).

[38] C. Wang, J. Harrington, and J. Preskill, *Confinement-Higgs transition in a disordered gauge theory and the accuracy threshold for quantum memory,* Annals of Physics **303**, 31 (2003).

[39] H. Nishimori, *Statistical Physics of Spin Glasses and Information Processing: An Introduction* (Clarendon Press, Oxford, 2001).

[40] G. Giedke and J. Ignacio Cirac, *Characterization of Gaussian operations and distillation of Gaussian states,* Phys. Rev. A **66**, 032316 (2002).

[41] J. Eisert, S. Scheel, and M. B. Plenio, *Distilling Gaussian States with Gaussian Operations is Impossible,* Phys. Rev. Lett. **89**, 137903 (2002).

[42] J. Zhang, C. Xie, K. Peng, and P. van Loock, *Anyon statistics with continuous variables,* Phys. Rev. A **78**, 052121 (2008).

[43] A. Kubica, M. E. Beverland, F. Brandão, J. Preskill, and K. M. Svore, *Three-Dimensional Color Code Thresholds via Statistical-Mechanical Mapping,* Phys. Rev. Lett. **120**, 180501 (2018).

[44] V. V. Albert, S. Pascazio, and M. H. Devoret, *General phase spaces: from discrete variables to rotor and continuum limits,* Journal of Physics A: Mathematical and Theoretical **50**, 504002 (2017).

[45] D. P. DiVincenzo, *Fault-tolerant architectures for superconducting qubits,* Physica Scripta Volume T **137**, 014020 (2009).

[46] R. Versluis, S. Poletto, N. Khammassi, B. Tarasinski, N. Haider, D. J. Michalak, A. Bruno, K. Bertels, and L. DiCarlo, *Scalable Quantum Circuit and Control for a Superconducting Surface Code,* Physical Review Applied **8**, 034021 (2017).

4

CODE DEFORMATION TECHNIQUES

The large-scale execution of quantum algorithms requires basic quantum operations to be implemented fault-tolerantly. The most popular technique for accomplishing this, using the devices that can be realised in the near term, uses stabilizer codes which can be embedded in a planar layout. The set of fault-tolerant operations which can be executed in these systems using unitary gates is typically very limited. This has driven the development of measurement-based schemes for performing logical operations in these codes, known as lattice surgery and code deformation. In parallel, gauge fixing has emerged as a measurement-based method for performing universal gate sets in subsystem stabilizer codes. In this chapter, it is shown that lattice surgery and code deformation can be expressed as special cases of gauge fixing, permitting a simple and rigorous test for fault-tolerance together with simple guiding principles for the implementation of these operations. The accuracy of this method is demonstrated with examples based on the surface code and color code, some of which are novel.

Lingling Lao, developed the rotation procedure presented in Figure 4.1. Christophe Vuillot and supervisor, Barbara Terhal, developed all the rest of the material for this chapter. Parts of it are published in the New Journal of Physics **21**, 033028 (2019) [1], and were written with Lingling Lao and Ben Criger.

4.1. INTRODUCTION

QUANTUM computers can implement algorithms which are much faster than their classical counterparts, with exponential speedup for problems such as prime factorisation [2], and polynomial speedup for many others [3]. The main obstacle to constructing a large-scale quantum computer is decoherence, which partially randomizes quantum states and operations. Although state-of-the-art coherence times are now appreciably longer than gate times [4, 5], they remain too short for useful quantum computation.

To counter the effect of decoherence on quantum states which are stored or manipulated imperfectly, we can encode logical qubit states into several physical qubits, and perform non-destructive multi-qubit measurements of the resulting system to extract information about which errors have occurred, called the *syndrome*. The spaces of multi-qubit states used to encode these logical states are called *quantum error-correcting codes*, and their ability to correct errors is measured by the *distance d*, which is the number of independent errors (or error *weight*) necessary to alter the state of the logical qubits without being detected. In order to use one of these codes in practice, it is also necessary to account for the effect of decoherence on operations. For example, a syndrome measurement may involve a sequence of entangling gates, and the error caused by a faulty gate on a small set of qubits in the beginning of the circuit may propagate onto many qubits, producing a high-weight error, increasing the likelihood of a logical error. Measurement results can also be corrupted by decoherence, so syndrome extraction often has to be repeated. In order to prevent error propagation during repeated measurement, syndrome extraction circuits must be designed such that a small number of faults (from imperfect gates or memory errors on data qubits) will result in a small number of errors on the physical qubits, which can be corrected using noisy syndromes. Given a family of codes of different distances, we can determine a *threshold error rate*, the rate beneath which codes with higher distance produce lower logical error probabilities.

Several such families of quantum error-correcting codes have been developed, including concatenated codes [6, 7], subsystem codes such as Bacon-Shor codes [8], and 2D topological codes. The most prominent 2D topological codes are surface codes [9] derived from Kitaev's toric code [10]. Another notable example of topological code is the color code [11]. 2D topological codes can be implemented using entangling gates which are local in two dimensions, allowing fault-tolerance in near-term devices which have limited connectivity. In addition, 2D topological codes generally have high fault-tolerant memory thresholds, with the surface code having the highest known at $\sim 1\%$ [12].

These advantages come at a cost, however. While 2D color codes permit certain single-qubit logical operations to be implemented *transversally*, the surface code does not have any. In addition, the constraint that computation be carried out in a single plane does not permit two-qubit physical gates to be carried out between physical qubits in different code blocks, precluding the two-qubit gates which, in principle, can be carried out transversally.

These two restrictions have led to the design of measurement-based protocols for performing single- and two-qubit logical gates by making gradual changes to the underlying stabilizer code. Measurement-based protocols that implement single-qubit gates

are typically called *code deformation* [13], and protocols that involve multiple logical qubits are usually called *lattice surgery* [14]. A separate measurement-based technique, called *gauge fixing* [15], can be applied to *subsystem codes*, which have operators which can be added to or removed from the stabilizer group as desired, the so-called *gauge operators*. During gauge fixing, the stabilizer generators of the subsystem code remain unchanged, and can be used to detect and correct errors; so decoding is unaffected by gauge fixing. This is in contrast to code deformation and lattice surgery, where it is not *a priori* clear which measurement results to incorporate into decoding, or how to process them. Recently, many different code deformation and lattice surgery techniques have been devised, most of which use tailor-made analysis or decoding techniques, see e.g. [16–23].

In this chapter, we phrase existing lattice surgery and code deformation protocols as special cases of gauge fixing, showing that the underlying subsystem code dictates the fault-tolerance properties of the protocol. This perspective can simplify the analysis of new measurement-based protocols, provided that they are based on stabilizer codes whose distances can be easily calculated. Also, knowing the stabilizer of the underlying subsystem code results in clear guidelines for decoding using the measurement results produced by such a protocol.

The remainder of this chapter is organised as follows. In Section 4.2, we review the ideas behind code deformation and lattice surgery. In Section 4.3, we review the formalism of gauge fixing. Following this, in Section 4.4, we formulate lattice surgery and code deformation operations as gauge fixing, demonstrating that fault-tolerant code deformation protocols are in fact based on high-distance subsystem codes. We also show this explicitly using both well-known and novel protocols. We conclude and discuss potential future research in Section 4.6.

In all figures in this chapter, qubits are located on the vertices of the drawn lattice. We refer to the local generators of the stabilizer group of the codes as *stabilizers* or *checks*. In surface code figures, black regions signify X-stabilizers and light grey regions Z-stabilizers, with no stabilizers measured on white plaquettes. In color code figures, colored faces signify both X- and Z-stabilizers unless otherwise specified. Shaded regions are a visual aid to identify colored boundaries.

4.2. CODE DEFORMATION AND LATTICE SURGERY

4.2.1. CODE DEFORMATION

Code deformation is a technique to convert one code into another by making a series of changes to the set of stabilizer generators to be measured in each round of error correction.

Typically, these protocols use ancillae prepared in entangled and/or encoded states as a resource. Also, a typical code deformation sequence proceeds gradually, first expanding the code into a large intermediate code by entangling the original code block with the ancillae, then disentangling some of the qubits (which may include some or all of the original data qubits), producing a final code which can then be used for further computation. The initial and final code may differ in their logical operators, in which case the deformation may perform a logical operation. Also, the initial and final code

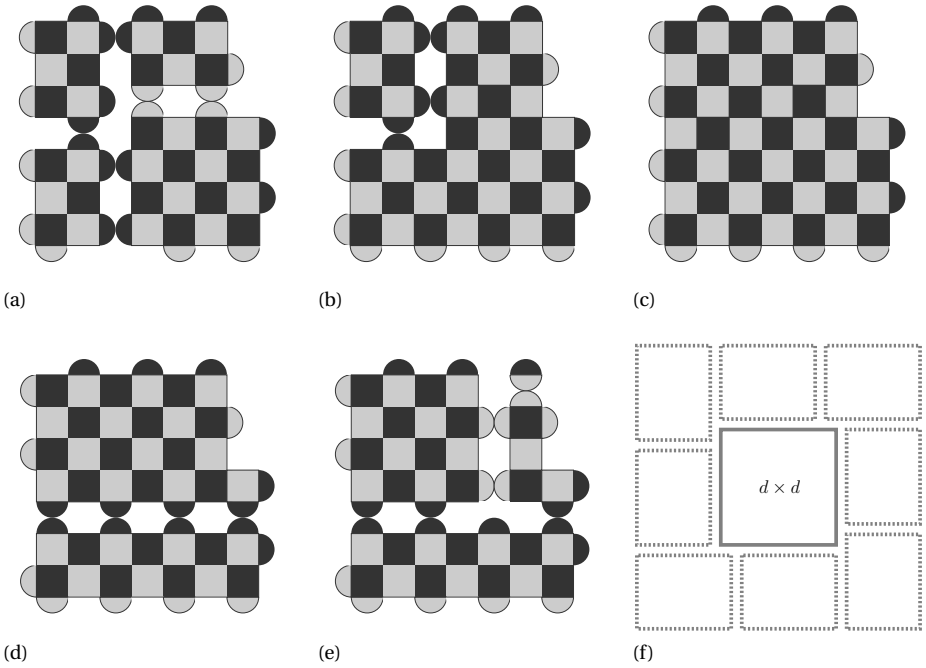
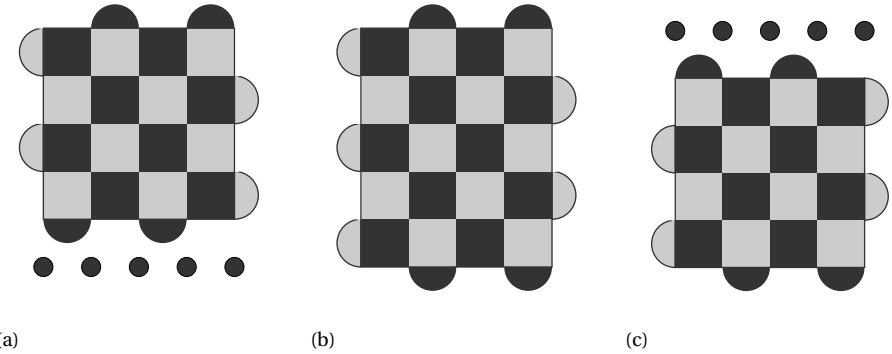


Figure 4.1: Fault-tolerant procedure for rotating a surface code by 90° and reflecting it about the x axis (see [14, Figure 10] for the corresponding protocol using smooth/rough boundaries). (a) Initial layout where the 5×5 lattice is to be rotated, the three 3×4 patches are ancillas in fixed states, fully specified by the stabilizers shown. (b) Intermediate lattice, this step is required to expand the lattice fault-tolerantly. (c) Fully expanded lattice. (d) and (e) Splitting operations performed to shrink the lattice. (f) By using the two steps from (a) to (c) at the same time on all corners, one can grow a lattice from distance d to $3d-4$. The surrounding ancillary patches have $(d-2) \times (d-1)$ qubits each.

may differ in their position or orientation within a larger quantum computer.

For example, consider the proposed fault-tolerant procedure for lattice rotation of surface codes shown in Figure 4.1, similar to the one presented in [24]. One can see five steps which gradually modify the surface code patch starting at the bottom right of Figure 4.1a and ending at the top left of Figure 4.1e in a different orientation. First, three ancillary patches are prepared in fixed states, and placed near the upper left corner of the target patch. Then, the patch undergoes a two-step growing operation, followed by a two-step shrinking operation. Advancing one step is done by measuring the operators corresponding to the new stabilizers, some of which anti-commute with the old ones. Measurement of these new stabilizers will return ± 1 values at random. This means that additional corrections, unrelated to errors that may have occurred, are needed in order to enter the new code space (the mutual $+1$-eigenspace of all new stabilizers). Moreover, to account for noisy operations, one must simultaneously perform error correction. After one is confident that the encoded state is in the new code space, one can proceed to the next step.



(a) (b) (c)

Figure 4.2: A procedure to flip a lattice using code deformation. (a) The lattice to be flipped, and the physical qubits prepared in $|+\rangle$ states. (b) The flip operation is realised by merging the original lattice with the physical qubits below. (c) Subsequently measuring the physical qubits at the top in the X basis finishes the flip operation.

In Section 4.4, we will demonstrate that, following these five steps, one can fault-tolerantly protect the logical information at all times with a distance-5 code. We also show that the distance would be reduced to 3 if one were to omit step (b), going directly from (a) to (c), as one would do when directly adapting the established surface code rotation method from [14] to rotated surface codes.

This lattice rotation followed by the lattice flip in Figure 4.2 is useful for performing a transversal Hadamard gate. The transversal Hadamard gate on a surface code patch, performed by applying a Hadamard gate on each qubit, interchanges X and Z plaquettes. This code transformation can be undone by a lattice rotation, followed by a lattice flip. Moreover, part of this rotation procedure can be used to grow a code with distance d to a code with distance $(3d-4)$ in two steps by simultaneously growing all corners, see Figure 4.1f.

This type of code deformation does not, in itself, perform logical operations, but can be used to move patches of code or to convert between codes where different gates are transversal [18]. Other code deformation procedures such as moving holes or twists do perform unitary logical Clifford operations [20, 25, 26]. In the next section, we present another similar procedure which executes a logical measurement.

4.2.2. LATTICE SURGERY

Lattice surgery is a particular measurement-based procedure that acts non-trivially on logical information. By going through two steps of deformation, it implements a joint measurement of logical operators, typically $\overline{X}_1\overline{X}_2$ or $\overline{Z}_1\overline{Z}_2$, where $\overline{X}_j$ and $\overline{Z}_j$ denote the logical operators of the logical qubit j. We will focus on the $\overline{Z}_1\overline{Z}_2$ measurement and review the protocol used for the surface code [14, 17].

Consider two patches of $L \times L$ rotated surface code, as in Figure 4.3a. Each has a $\overline{Z}$ along the boundary which faces the other patch. In the *merge* step, one measures the intermediary Z-plaquettes (in pink in Figure 4.3b). These plaquettes are such that the

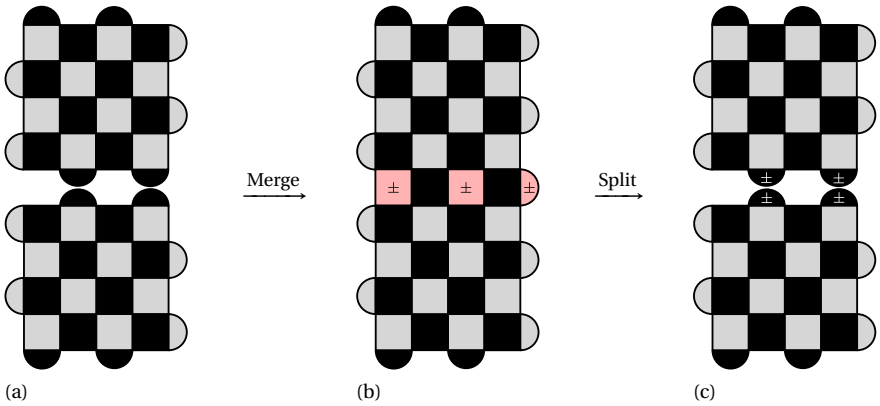


Figure 4.3: Lattice surgery for the rotated surface code. Grey plaquettes show Z-stabilizers, black plaquettes represent X-stabilizers. A '±' label indicates a random sign for the corresponding plaquette in the stabilizer group. (a) Initial layout, two rotated surface codes. (b) The merged lattice, which is a surface code with random ± signs on the newly-measured (red) plaquettes. (c) The *split* lattices, in which the original stabilizers are measured again. Random ± signs are produced on the boundary X-stabilizers.

product of all outcomes is the outcome of the $\overline{Z}_1\overline{Z}_2$ measurement, but any subset of these outcomes produces a random result when multiplied together. This ensures that the only non-stabilizer operator whose eigenvalue can be inferred from these measurements is $\overline{Z}_1\overline{Z}_2$. These measurements do not commute with the weight-2 X stabilizers at the joint boundary (in Figure 4.3a). The Gottesman-Knill theorem [27] prescribes how to update the stabilizer after such measurements, namely we only retain elements in the original stabilizer group which do commute with the newly measured stabilizers. This implies that the code becomes a $2L \times L$ patch of surface code, apart from some minus signs on the newly-measured Z-checks. This merge step is very similar to the rotation presented before, except that some logical information is gained in the process and the additional corrections which fix the state into the new code space may involve one of the original logical operators (when the number of intermediary plaquettes with -1 eigenvalues is odd). To finish the protocol, the original code space must be restored by performing a *splitting* operation, measuring the original stabilizers of the two separate patches instead of the intermediary Z-plaquettes. Those Z-plaquettes, as in the merge step, anticommute with the boundary X-stabilizers, and will be removed from the stabilizer group. Their product, equal to $\overline{Z}_1\overline{Z}_2$, does commute, and will remain as a stabilizer of the final state. In addition, the boundary X-plaquettes will have random ± signs which are perfectly correlated between facing pairs. Therefore, one can eliminate these ± signs by applying some of the former stabilizers (those supported on the intermediary Z-plaquettes).

One can check (see the algebraic proof in Section 4.5) that depending on the outcome (± 1) of the logical $\overline{Z}_1\overline{Z}_2$ measurement, the merge and split operations, respectively $M_{\pm}$

and $S_\pm$ can be expressed as

$$M_+ = |\overline{0}\rangle\langle\overline{00}| + |\overline{1}\rangle\langle\overline{11}|, \qquad S_+ = |\overline{00}\rangle\langle\overline{0}| + |\overline{11}\rangle\langle\overline{1}|, \tag{4.1}$$

$$M_- = |\overline{0}\rangle\langle\overline{01}| + |\overline{1}\rangle\langle\overline{10}|, \qquad S_- = |\overline{01}\rangle\langle\overline{0}| + |\overline{10}\rangle\langle\overline{1}|. \tag{4.2}$$

They are related to the projections, $P_\pm$, onto the ±1 eigenspace of $\overline{Z}_1\overline{Z}_2$ by composition:

$$P_+ = S_+ \circ M_+, \qquad P_- = S_- \circ M_-.$$

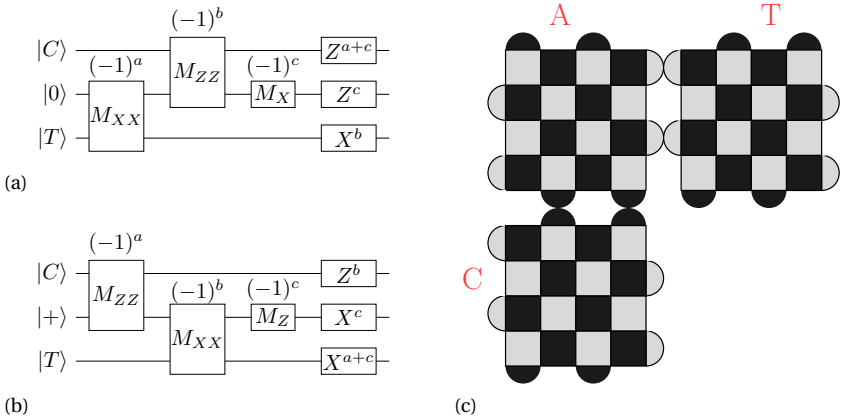


Figure 4.4: (a) & (b) Two equivalent measurement-based circuits for the CNOT gate. (c) The qubit layout for a CNOT gate between two surface-code qubits. C is the control qubit, T is the target qubit, and A is a logical ancilla.

In particular, lattice surgery allows us to implement the measurement-based CNOT gate [28] in a 2D layout with only local operations as shown in Figure 4.4. We note that a more general set of operations which can be implemented by lattice surgery can be constructed using the relation between the merge and split operations considered here and the three-legged nodes of the ZX-calculus [29]. For the purposes of this chapter, however, we will limit our discussion to CNOT gates.

4.3. GAUGE FIXING

Gauge fixing [15] is an approach which has been used to implement universal fault-tolerant gate sets in *subsystem codes* [30]. A subsystem code is equivalent to a stabilizer code in which some of the logical qubits are not used to carry any logical information. These logical qubits are called *gauge* qubits and they can be acted on or measured without disturbing the states of the other logical qubits, which are used to store and process quantum information. Then, one way to formally define a subsystem code, C, is to define a subgroup of the Pauli group, called the *gauge group* $\mathcal{G}$, containing all the Pauli stabilizers as well as the Pauli operators defining the gauge qubits. This subgroup is non-Abelian as it contains anti-commuting Pauli operator pairs which represent the gauge

qubit logical operators. The stabilizer group, $\mathcal{S}$, can be derived from $\mathcal{G}$ as its center, denoted Z(·), i.e. containing all elements in $\mathcal{G}$ which mutually commute

$$\mathcal{S} = \mathrm{Z}(\mathcal{G}) = \mathcal{C}(\mathcal{G}) \cap \mathcal{G}, \tag{4.3}$$

where $\mathcal{C}(\mathcal{G})$ denotes the centralizer of $\mathcal{G}$ in the Pauli group, i.e. all elements in the Pauli group with commute with all elements in $\mathcal{G}$. Elements in $\mathcal{G}$ which are not in $\mathcal{S}$ are the Pauli operators acting non-trivially on the gauge qubits: this is the set of gauge operators $\mathcal{L}_g$

$$\mathcal{L}_g = \mathcal{G} \setminus \mathcal{S}. \tag{4.4}$$

The codespace C is defined by the stabilizer group $\mathcal{S}$, with an additional equivalence between code states if they differ only by a gauge operator

$$C = \left\{ |\psi\rangle \mid \forall S \in \mathcal{S}\ S|\psi\rangle = |\psi\rangle \right\}, \qquad |\psi_1\rangle \sim |\psi_2\rangle \Leftrightarrow \exists G \in \mathcal{L}_g,\ |\psi_1\rangle = G|\psi_2\rangle\,. \tag{4.5}$$

Following this, one can define operators for the actual logical qubits which by definition are elements in $\mathcal{C}(\mathcal{S}) \setminus \mathcal{S}$. If these operators act trivially on the gauge qubits, we call these *bare* logical operators. Bare logical operators can be multiplied by elements in $\mathcal{L}_g$ to become *dressed* logical operators which also act on the gauge qubits. We can write

$$\mathcal{L}_{\text{bare}} = \mathcal{C}(\mathcal{G}) \setminus \mathcal{G}, \qquad \mathcal{L}_{\text{dressed}} = \mathcal{C}(\mathcal{S}) \setminus \mathcal{S}. \tag{4.6}$$

Note that with this definition we have, $\mathcal{L}_{\text{bare}} \subset \mathcal{L}_{\text{dressed}}$. The distance of the subsystem code C is the smallest weight of any of its logical operators,

$$d_C = \min_{\ell \in \mathcal{L}_{\text{dressed}}} \mathrm{wt}(\ell). \tag{4.7}$$

One advantage of subsystem codes is that to measure stabilizers, one is free to measure any set of checks in the gauge group as long as this set generates the stabilizer group. By measuring elements in the full gauge group, one can put the gauge qubits in specific states, permitting different sets of transversal logical gates. This act of putting the gauge qubits in a specific state is called *gauge fixing*. The idea is to measure a commuting subset of gauge operators (all the Z-type gauge operators, for example), obtaining ± 1 outcomes and applying the anticommuting, or *conjugate* partner operator (an X-type gauge operator in the example), wherever a -1 outcome has been obtained. In the example, this would fix all gauge qubits to the $|0\rangle$ state. While the gauge is fixed in this way, the Z-type gauge operators become elements of the stabilizer group, so $\mathcal{S}$ is augmented to some larger Abelian subgroup of $\mathcal{G}$.

4.4. FAULT-TOLERANCE ANALYSIS WITH GAUGE FIXING

In this section, we show how both code deformation and lattice surgery can be viewed as gauge fixing operations and therefore, one can use gauge fixing to analyze the fault-tolerance of these operations.

We consider the QEC codes before and after a deformation step, denoted as C_{old} and C_{new}, with stabilizer groups $\mathcal{S}_{\text{old}}$ and $\mathcal{S}_{\text{new}}$, respectively. Both codes are fully defined on the same set of qubits. The logical operators of each code are defined as

$$\mathcal{L}_{\text{old}} = \mathcal{C}(\mathcal{S}_{\text{old}}) \setminus \mathcal{S}_{\text{old}}\,, \qquad \mathcal{L}_{\text{new}} = \mathcal{C}(\mathcal{S}_{\text{new}}) \setminus \mathcal{S}_{\text{new}}.$$

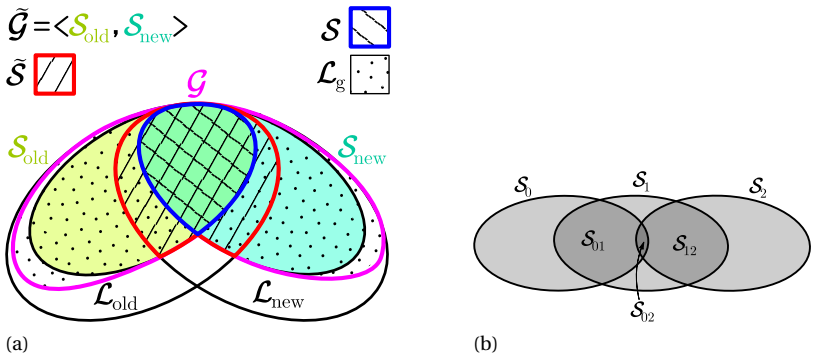


Figure 4.5: Venn diagrams depicting the relations between the different sets of Pauli operators concerning the gauge group $\mathcal{G}$ of interest, see main text. (a) For one step, the yellow-green set represents the old stabilizer group, $\mathcal{S}_{\text{old}}$, and the blue-green set the new group, $\mathcal{S}_{\text{new}}$. Both are surrounded by the logical operators, $\mathcal{L}_{\text{old}}$ and $\mathcal{L}_{\text{new}}$ respectively. The gauge group generated by both, $\tilde{\mathcal{G}} = \langle \mathcal{S}_{\text{old}}, \mathcal{S}_{\text{new}} \rangle$, has $\tilde{\mathcal{S}}$ as its center, shown by the down-left-dashed region. The gauge group of interest, $\mathcal{G}$, is outlined in purple and has $\mathcal{S}$, in the down-right-dashed region as its center. The set of gauge operators defining the gauge qubits, $\mathcal{L}_g$, is the dotted region. When switching from $\mathcal{S}_{\text{old}}$ to $\mathcal{S}_{\text{new}}$ one fixes the gauge for the elements in the blue-green dotted region $\mathcal{M}_{\text{fix}} = \tilde{\mathcal{G}} \setminus \mathcal{S}_{\text{old}}$. (b) One possible scenario for two successive steps of deformation. Doing it in two steps, i.e. from $\mathcal{S}_0 \to \mathcal{S}_1$, and then from $\mathcal{S}_1 \to \mathcal{S}_2$ permits to use successively the stabilizer groups $\mathcal{S}_{01}$ and then $\mathcal{S}_{12}$ for error correction. Skipping the intermediary steps, one can only use $\mathcal{S}_{02}$ which might offer less protection.

The intuition we follow is to see the two stabilizer codes as two different gauges of the same subsystem code. The first step, then, is to define a joint subsystem code, $\tilde{C}$, whose gauge group, $\tilde{\mathcal{G}}$, is generated by both $\mathcal{S}_{\text{old}}$ and $\mathcal{S}_{\text{new}}$,

$$\tilde{\mathcal{G}} = \langle \mathcal{S}_{\text{old}}, \mathcal{S}_{\text{new}} \rangle.$$

The generated group, $\tilde{\mathcal{G}}$, is not necessarily Abelian, since it contains elements of $\mathcal{S}_{\text{old}}$ which may anti-commute with some elements of $\mathcal{S}_{\text{new}}$.

The stabilizer group, $\tilde{\mathcal{S}}$, defined as in Eq. (4.3), can be characterised as follows: Elements in the center of $\tilde{\mathcal{G}}$ also have to be in the centralisers of $\mathcal{S}_{\text{old}}$ and $\mathcal{S}_{\text{new}}$. Moreover, being in both centralisers and in $\tilde{\mathcal{G}}$ is sufficient to be in the center, or

$$\tilde{\mathcal{S}} = \mathcal{C}(\mathcal{S}_{\text{old}}) \cap \mathcal{C}(\mathcal{S}_{\text{new}}) \cap \tilde{\mathcal{G}}.$$

See Figure 4.5a for a representation of $\tilde{\mathcal{S}}$ as a Venn diagram. Note that, in addition to containing $\mathcal{S}_{\text{old}} \cap \mathcal{S}_{\text{new}}$, $\tilde{\mathcal{S}}$ can also contain some logical operators from either $\mathcal{L}_{\text{old}}$ or $\mathcal{L}_{\text{new}}$. This is the case for the merge operation of lattice surgery where the logical $\overline{Z}_1\overline{Z}_2 \in \mathcal{L}_{\text{old}}$ but also $\overline{Z}_1\overline{Z}_2 \in \mathcal{S}_{\text{new}}$ and therefore $\overline{Z}_1\overline{Z}_2 \in \tilde{\mathcal{S}}$. Similarly, for the split operation $\overline{Z}_1\overline{Z}_2 \in \mathcal{L}_{\text{new}}$ but also in $\mathcal{S}_{\text{old}}$ and therefore in $\tilde{\mathcal{S}}$.

As defined above, this subsystem code $\tilde{C}$ indeed admits $\mathcal{S}_{\text{old}}$ and $\mathcal{S}_{\text{new}}$ as two distinct Abelian subgroups of $\tilde{\mathcal{G}}$. Therefore the codes $\mathcal{S}_{\text{old}}$ and $\mathcal{S}_{\text{new}}$ correspond to fixing two different sets of states for the gauge qubits of $\tilde{\mathcal{G}}$. However, for this to function as a subsystem code, one would have to be stabilized at all times by $\tilde{\mathcal{S}}$ and thus be able to measure all values of the stabilizers of $\tilde{\mathcal{S}}$.

This is not the necessarily the case when $\tilde{\mathcal{S}}$ contains some elements of $\mathcal{L}_{\text{old}}$ or $\mathcal{L}_{\text{new}}$, and we have to further modify $\tilde{\mathcal{G}}$ to a gauge group $\mathcal{G}$ whose center is solely

$$\mathrm{Z}(\mathcal{G}) = \mathcal{S} = \mathcal{S}_{\text{old}} \cap \mathcal{S}_{\text{new}}.$$

How do we obtain $\mathcal{G}$ from $\tilde{\mathcal{G}}$? This new gauge group, $\mathcal{G}$ will be generated by $\mathcal{S}_{\text{old}}$ and $\mathcal{S}_{\text{new}}$ in addition to (anti-commuting) conjugate partners of elements in the sets $\mathcal{M}_{\text{prep}} = \mathcal{S}_{\text{old}} \cap \mathcal{L}_{\text{new}}$ and $\mathcal{M}_{\text{meas}} = \mathcal{S}_{\text{new}} \cap \mathcal{L}_{\text{old}}$. More precisely, one views $\mathcal{M}_{\text{prep}}$ as a subset of $\mathcal{L}_{\text{new}}$, and for each independent logical operator contained in $\mathcal{M}_{\text{prep}}$ adds a chosen conjugated partner within $\mathcal{L}_{\text{new}}$. One operates similarly for $\mathcal{M}_{\text{meas}}$ by viewing it as a subset of $\mathcal{L}_{\text{old}}$. If we then consider the center of $\mathcal{G}$, we see that all elements in $\mathcal{M}_{\text{prep}}$ and $\mathcal{M}_{\text{meas}}$ are excluded from it since they anti-commute with some elements in $\mathcal{G}$. This means that the center of $\mathcal{G}$ is reduced to $\mathrm{Z}(\mathcal{G}) = \mathcal{S}_{\text{old}} \cap \mathcal{S}_{\text{new}}$ as desired.

The names $\mathcal{M}_{\text{prep}}$ and $\mathcal{M}_{\text{meas}}$ are chosen to represent their respective roles in the deformation procedure. In such a procedure one starts from a system encoded in C_{old}, i.e. stabilized by $\mathcal{S}_{\text{old}}$, and then one measures the new stabilizers, $\mathcal{S}_{\text{new}}$. When $\mathcal{S}_{\text{new}}$ contains some elements of $\mathcal{L}_{\text{old}}$, then in general these elements will not stabilize the state of the system, since it can be in any logical state at the beginning of the procedure. Measuring these operators will return information about the logical state and cannot return information about errors. Thus, by switching to $\mathcal{S}_{\text{new}}$ one also performs a logical measurement of the elements in $\mathcal{M}_{\text{meas}}$.

It is also possible for $\mathcal{S}_{\text{old}}$ to contain some elements of $\mathcal{L}_{\text{new}}$. In that case, the state of the system is initially stabilized by these elements, and remains so, since we only measure operators commuting with them. In this sense, the deformation procedure will prepare the logical +1 state of elements in $\mathcal{M}_{\text{prep}}$.

We denote the code underlying the code deformation step as C. Its gauge group, $\mathcal{G}$, is represented as a Venn diagram in Figure 4.5a. Thus the deformation operation that transforms C_{old} into C_{new} is realized by switching what gauge to fix of the code C: in one gauge one obtains C_{old}, the other gauge gives C_{new}. Since the deformation step can also transform logical information, what gauge elements are fixed is subtle. Namely, note that in this gauge fixing of C to either code C_{old} or C_{new} the gauge elements in $\mathcal{G} \setminus \tilde{\mathcal{G}}$ will never be fixed. Said differently, only the elements of $\mathcal{L}_g$ which are in the blue-green dotted region in Fig. 4.5 will be fixed, one can also view these as elements of $\mathcal{M}_{\text{fix}} \equiv \tilde{\mathcal{G}} \setminus \mathcal{S}_{\text{old}}$.

4.4.1. FAULT-TOLERANCE OF CODE DEFORMATION

Given an underlying subsystem deformation code C, one can ensure the fault-tolerance of a code deformation operation by checking three criteria:

1. **Code distance:** The distance of the subsystem code, C, must be large enough for the desired protection. Ideally it matches the distances of C_{old} and C_{new} so the degree of protection is not reduced during the deformation step.

2. **Error correction:** The error correction procedure follows that of the subsystem code C through the code deformation step.

3. **Gauge fixing:** To fix the gauge, one has to use operators exclusively from $\mathcal{L}_g = \mathcal{G} \setminus \mathcal{S}$.

More specifically, criterion 2 means that to perform error correction, one has to reconstruct from the measurements of $\mathscr{S}_{\text{new}}$ the syndrome given by $\mathscr{S}$. Importantly, criteria 2 and 3 demonstrate that the processes of error correction and that of gauge fixing are two separate processes with different functionality. Both processes require the application of Pauli operators (in hardware or in software) to make sure that stabilizer measurements are corrected to have outcome +1. The error correction process does this to correct for errors, while the gauge-fixing process does this to move from C_{old} to C_{new}.

This description holds for one step of deformation, so that for each step in a sequence of deformations one has to examine the corresponding subsystem code C and its distance. Depending on the sequence, Figure 4.5b illustrates why skipping steps could lead to poor distance and poor protection against errors. This discussion also assumes that stabilizer measurements are perfect; the effect of noisy stabilizer measurements is considered in the following section.

NOISY MEASUREMENTS

When one considers noisy syndrome measurements, one needs to ensure that both the stabilizer outcomes and the state of the gauge qubits can be learned reliably. For 2D stabilizer codes such as the surface code this is simply done by repeating the measurements. To process this repeated measurement information for the surface code, one no longer uses the syndrome but the *difference syndrome*: the difference syndrome is marked as non-trivial (we say that a *defect* is present) only when the syndrome value changes from the previous round of measurement. This difference syndrome or defect gives information about both qubit errors as well as measurement errors. Switching to the difference syndrome point of view can be understood algebraically as follows:

$$\begin{aligned} G\boldsymbol{e}_0 + \boldsymbol{d}_0 &= \boldsymbol{s}_0 & G\boldsymbol{e}_0 + \boldsymbol{d}_0 &= \boldsymbol{s}_0 = \boldsymbol{\Delta s}_0 \\ G(\boldsymbol{e}_0 + \boldsymbol{e}_1) + \boldsymbol{d}_1 &= \boldsymbol{s}_1 & G\boldsymbol{e}_1 + \boldsymbol{d}_1 - \boldsymbol{d}_0 &= \boldsymbol{s}_1 - \boldsymbol{s}_0 = \boldsymbol{\Delta s}_1 \\ &\vdots & &\vdots \\ G\left(\sum_{t=0}^{T} \boldsymbol{e}_t\right) + \boldsymbol{d}_T &= \boldsymbol{s}_T & G\boldsymbol{e}_T + \boldsymbol{d}_T - \boldsymbol{d}_{T-1} &= \boldsymbol{s}_T - \boldsymbol{s}_{T-1} = \boldsymbol{\Delta s}_T \end{aligned} \tag{4.8}$$

where G is the stabilizer generating matrix of the code, $\boldsymbol{e}_t$ is the vector of errors occurring at time t on data qubits, $\boldsymbol{d}_t$ the vector of errors on the syndrome readout at time t, $\boldsymbol{s}_t$ the vector of syndrome values observed at time t and $\boldsymbol{\Delta s}_t$ the difference syndrome at time t. The left column in (4.8) describes the physical evolution of the system where in every round the syndrome measurement gives information about the cumulative errors in time. The right column in (4.8) describes rewriting the same information in terms of the difference syndrome, which yields equations that are local in both space and time. It can therefore be reinterpreted as a *space-time* error correcting code and the decoding procedure is more easily understood in this picture.

We can see how this picture changes when doing a code deformation at time T_d. At T_d one obtains the syndrome for the code $\mathscr{S}_{\text{new}}$. For those elements in $\mathscr{S}_{\text{new}}$ which are in $\mathscr{S}$, we know what this syndrome should have been if no measurement or data errors had occurred since the previous QEC round which measured the stabilizers of $\mathscr{S}_{\text{old}}$. Therefore, we can place defects when the found syndrome changes from what it was expected

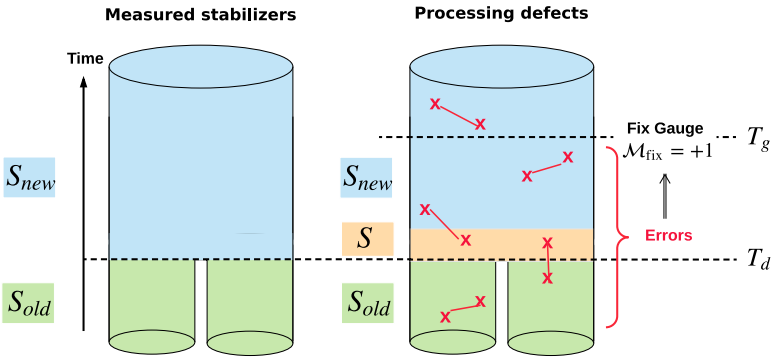


Figure 4.6: Schematic drawing of a code deformation procedure with repeated noisy measurements, with time increasing upwards. T_d designates the time step at which the code deformation (the switch from measuring the checks of $\mathcal{S}_{\text{old}}$ to those of $\mathcal{S}_{\text{new}}$) is performed. T_g is the time at which one is confident enough about the state of the gauge qubits, taking into account errors, to fix their states. This means that, after T_g, another logical computation can be performed. (Right) The first round of measurement of $\mathcal{S}_{\text{new}}$ at time T_d does not have a previous value to compare to in order to construct a difference syndrome, i.e. one can only construct defects for $\mathcal{S}$. Immediately after this step, one can derive the difference syndrome of the full $\mathcal{S}_{\text{new}}$, placing defects accordingly. Using defects before and after T_d, one processes error information to infer the value of the gauge operators in $\mathcal{M}_{\text{fix}}$ at time T_g, thus fixing the gauge at T_g.

to be based on the last round of measurements with $\mathcal{S}_{\text{old}}$. $\mathcal{S}_{\text{new}}$ also contains a subset of elements in $\mathcal{L}_g$, namely the blue-green dotted region $\mathcal{M}_{\text{fix}}$ in Fig. 4.5a. Some of these elements are also contained in $\mathcal{L}_{\text{old}}$ (down-right-dashed area in Fig. 4.5a), i.e. they are elements of $\mathcal{M}_{\text{meas}}$. The eigenvalues of these elements in $\mathcal{M}_{\text{meas}}$ depend on the logical state and are therefore not a proper syndrome for S_{old}. So only after one more round of QEC with $\mathcal{S}_{\text{new}}$, one can mark whether the syndrome for these elements in $\mathcal{M}_{\text{meas}}$ changes, and either place a defect or not. In addition, the eigenvalues of the gauge operators in the remaining blue-green dotted region take random ± 1 eigenvalues (since they anti-commute with some elements in $\mathcal{S}_{\text{old}}$): for these checks, like for the elements in $\mathcal{M}_{\text{meas}}$, there is no previous record to construct a difference syndrome right away. Again, only after one round of QEC with $\mathcal{S}_{\text{new}}$ one can again mark whether the syndrome changed, placing a defect for an element or not. In processing these new syndromes of $\mathcal{S}_{\text{new}}$ to do error correction, we should also allow them to be matched with virtual defects placed beyond the past-time boundary T_d. For example, a measurement error in the first step when the syndrome is randomly $+1$ or -1, followed by many rounds without measurement error, produces a single defect and should be interpreted as the first measurement being incorrect.

This can be summarized algebraically as follows. We denote as G^{old} the generating matrix for the stabilizer before the code deformation, G^{new} for after, as well as G the generating matrix for the stabilizer of the associated subsystem code. By definition of G, there exist two matrices, Λ^{old} and Λ^{new} obeying

$$G = \Lambda^{\text{old}} G^{\text{old}} = \Lambda^{\text{new}} G^{\text{new}}. \tag{4.9}$$

Using this identity we can construct the local space-time code across the code deforma-

tion time step at time T_d as follows

$$
\begin{aligned}
&\vdots && \vdots \\
&G^{\text{old}}\left(\sum_{t=0}^{T_d-1} \boldsymbol{e}_t\right) + \boldsymbol{d}_{T_d-1} = \boldsymbol{s}_{T_d-1} && G^{\text{old}}\boldsymbol{e}_{T_d-1} + \boldsymbol{d}_{T_d-1} - \boldsymbol{d}_{T_d-2} = \boldsymbol{s}_{T_d-1} - \boldsymbol{s}_{T_d-2} = \boldsymbol{\Delta s}_{T_d-1} \\
&G^{\text{new}}\left(\sum_{t=0}^{T_d} \boldsymbol{e}_t\right) + \boldsymbol{d}_{T_d} = \boldsymbol{s}_{T_d} && G\boldsymbol{e}_{T_d} + \Lambda^{\text{new}}\boldsymbol{d}_{T_d} - \Lambda^{\text{old}}\boldsymbol{d}_{T_d-1} = \Lambda^{\text{new}}\boldsymbol{s}_{T_d} - \Lambda^{\text{old}}\boldsymbol{s}_{T_d-1} = \tilde{\boldsymbol{\Delta}}\boldsymbol{s}_{T_d} \\
&G^{\text{new}}\left(\sum_{t=0}^{T_d+1} \boldsymbol{e}_t\right) + \boldsymbol{d}_{T_d+1} = \boldsymbol{s}_{T_d+1} && G^{\text{new}}\boldsymbol{e}_{T_d+1} + \boldsymbol{d}_{T_d+1} - \boldsymbol{d}_{T_d} = \boldsymbol{s}_{T_d+1} - \boldsymbol{s}_{T_d} = \boldsymbol{\Delta s}_{T_d+1} \\
&\vdots && \vdots
\end{aligned}
\tag{4.10}
$$

Given the syndromes $\Delta \boldsymbol{s}_t$ of the constructed local space time code, any suitable decoder, minimum-weight matching for the surface code for example, can be used to decode (see Fig. 4.6), to infer some errors as they have occurred in a window of time before and after T_g and T_d (one may use a sliding window as in [24]). Let us then imagine that by matching defects in a window which goes beyond a so-called gauge-fixing time T_g, one infers a set of measurement and data errors. These errors are projected forwards to the time-slice T_g and they are used to do three things. The first two concern correctly interpreting the measured syndrome $\boldsymbol{s}_{T_d}$ at the time of the deformation. In more detail the first one is to correct the value of elements in M_{meas} (if any), so that the logical measurement has been completed and properly interpreted. The second is to determine or fix the gauge, i.e. determine the outcome of elements $\mathscr{M}_{\text{fix}}$ in the blue-green dotted region of Fig. 4.5. As we have argued, these gauge values may be ± 1 at random and hence Pauli gauge-fixing corrections can be added in software to make the outcomes all $+1$ if one wishes to work with the frame where all elements in $\mathscr{S}_{\text{new}}$ have $+1$ eigenvalue. These Pauli gauge-fixing corrections are not error corrections and any set of Pauli operators can be chosen as long as they solely fix the values of the elements in $\mathscr{M}_{\text{fix}}$. Thirdly, the projected errors provide the usual update of the Pauli frame for the code $\mathscr{S}$, so together with the gauge-fixing corrections, for the code $\mathscr{S}_{\text{new}}$. The whole procedure is represented schematically in Fig. 4.6; at time T_g, the code deformation step is finished.

Note that, after T_d, the elements in $\mathscr{M}_{\text{prep}}$ are no longer measured, but their fixed values before the code deformation now represent logical states prepared by code deformation.

Typically, for 2D stabilizer codes, the time window between T_g and T_d needs be of size $O(d)$ in order to fix the gauge, where d is the distance of code C. In some cases, the measurements contain enough redundant information about the gauge operators so that T_g can be equal to T_d. For example, this is the case when performing the logical measurement of a patch of code by measuring every single qubit in the Z basis. This is also the case for the logical measurement step of the plain surgery technique explained below. Such measurement procedures, where a single-round of measurements is enough to gather enough information about the syndrome are called single-shot. This concept exists already for regular error correction, outside of a code deformation procedure. If one can perform error correction reliably given a single round syndrome mea-

surements, this is called single-shot error correction. In the context of a code deformation procedure, one needs to be able to learn the gauge information reliably to be single-shot.

In the remainder of this section, we apply this formalism to the code deformation and lattice surgery operations discussed earlier.

4.4.2. CODE DEFORMATION EXAMPLES

GROW OPERATIONS

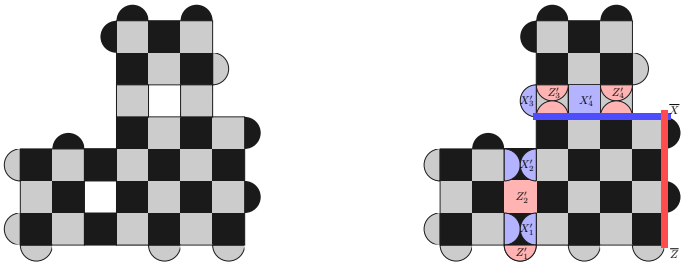
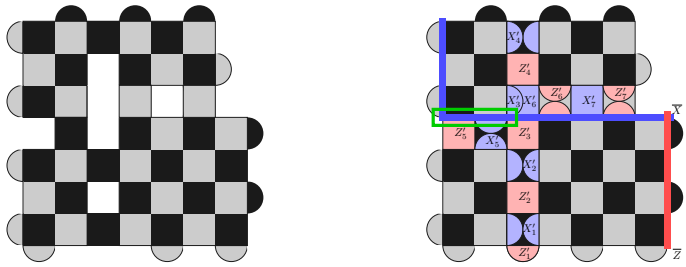


Figure 4.7: Description of the subsystem code, C, which holds during the first step of the grow operation depicted in 4.1a, 4.1b. (a) Generators for the stabilizer group, $\mathcal{S}$, of C. (b) Generators for the whole gauge group $\mathcal{G}$ of C. Highlighted in red and blue, respectively, are gauge operators, elements of $\mathcal{L}_g$, of Z-type and X-type, respectively. The logical operators, $\overline{X}, \overline{Z} \in \mathcal{L}_{\text{bare}}$, are also represented in brighter colours.

Gauge fixing, when applied to the growing operations of Figure 4.1 and Figure 4.2, reveals an underlying subsystem code with a small number of widely-spaced holes and large boundaries, resulting in a high distance. The stabilizer group, $\mathcal{S}$, as well as the gauge operators, $\mathcal{L}_g$, for the subsystem code C which governs the deformation from Figure 4.1a to Figure 4.1b, are shown in Figure 4.7.

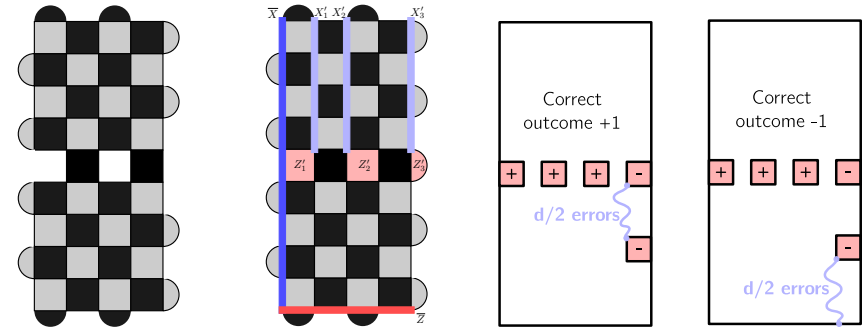
In all figures of this chapter, light blue and light red patches individually represent X-type and Z-type gauge operators, and bright blue and bright red qubit chains are $\overline{X}$ and $\overline{Z}$ operators respectively. The grow operation is changing the gauge from one in which the gauge operators not overlapping between the initially separate patches are fixed, denoted as $\{X_1', X_2', Z_3', Z_4'\}$ in Figure 4.7b, to one in which the overlapping ones are fixed, denoted as $\{Z_1', Z_2', X_3', X_4'\}$ in Figure 4.7b. The distance of C is still 5, matching the distance of the initial code.

Now consider what happens if we would go directly from Figure 4.1a to Figure 4.1c. The stabilizers and the gauge operators for this operation are shown in Figure 4.8. Similarly, one fixes the gauge going from separate patches to a single patch. The distance of the subsystem code for this operation is only 3. Indeed one of the minimum-weight dressed logical operators is the $\overline{Z}$ on the qubits in the green box in Figure 4.8b. That means that, in order to preserve the code distance, one should perform the intermediary step.



(a) (b)

Figure 4.8: The operators of the subsystem code for the one-step grow operation from Fig. 1a to Fig. 1c, skipping Fig. 1b: (a) The stabilizers which generate $\mathscr{S}$ and (b) the whole gauge group, $\mathscr{G}$, with highlighted gauge operators and logical operators.



(a) (b) (c) (d)

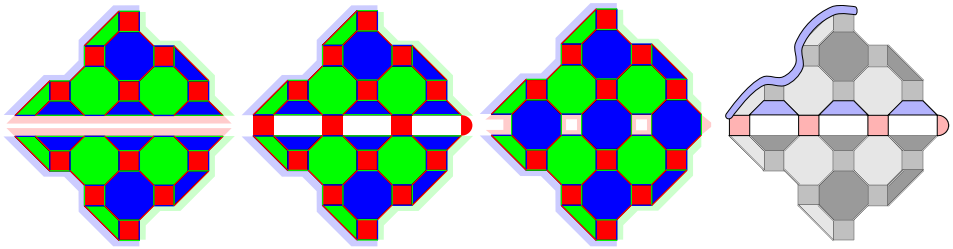
Figure 4.9: The operators of the subsystem code, C, for the joint measurement $\overline{ZZ}$. (a) The generators of stabilizer group $\mathscr{S}$. (b) The highlighted operators are either gauge operators in $\mathscr{L}_g$ or logical operators in $\mathscr{L}_{\text{bare}}$. We start in the gauge where the products $X_1'X_2'$ and $X_2'X_3'$ are fixed, and end in the gauge where Z_1', Z_2', and Z_3' are fixed. The distance of the subsystem code is 5, since one can construct a logical $\overline{X}$ with this weight by multiplying it with X gauge operators. (c) & (d) Two different scenarios with errors of weight $d/2$ with the same observed measurements.

THE MERGING AND SPLITTING OPERATIONS

In this section, we interpret the joint measurement of $\overline{ZZ}$ by lattice surgery in Figure 4.3b as gauge fixing. The stabilizer group $\mathscr{S}$ is generated by all the stabilizers in Figure 4.9a. The gauge operators, $\mathscr{L}_g$, of the gauge group are given by three representatives of the logical X of the top patch and the intermediary Z plaquettes that anti-commute with them. They are denoted as $\langle X_1', Z_1', X_2', Z_2', X_3', Z_3' \rangle$ in Figure 4.9b. Representatives of the bare logical operators, $\overline{X}, \overline{Z} \in \mathscr{L}_{\text{bare}}$, are the logical Z of the bottom patch and the logical X of the merged patch (joining the very top to the very bottom), see Figure 4.9b. The merge and split operations are realised by fixing some gauge operators of $\mathscr{L}_g$, resulting in new codes C_{merged} or C_{split}, respectively. Note that the weight of $\overline{X}$ of the subsystem

code, C, is only d and not $2d$ which is the distance for X of the merged code. Indeed, by using the gauge operators like X_1' and stabilizers, one can construct a dressed logical X of weight d. Another way of seeing this is by realizing that one cannot distinguish between two errors of weight $d/2$ depicted in Figures 4.9c and 4.9d. In the first one, the logical measurement outcome is -1 and there is a string of $d/2$ X-errors from the bottom to the middle of the bottom patch. In the second one the logical measurement outcome is $+1$ and there is a string of $d/2$ X-errors from the middle of the bottom patch and the middle (changing the observed logical measurement outcome to -1). Note also that when performing the splitting operation, one wants to correct the -1 outcomes for some of the intermediary X stabilizers. They are gauge operators equivalent to, say $X_1'X_2'$. They have to be corrected using the Z gauge operators, say Z_1' in this case. Otherwise one would introduce a logical Z error.

An almost identical analysis can be done for the color code version of lattice surgery [17], as well as hybrid between a surface code patch and a color code patch [19]. The idea is very similar since color codes also have logical operators along their colored boundaries, see the presentation of color codes in Section 1.2.3 of Chapter 1. Figure 4.10 shows the different stabilizer and gauge operators involved in color code surgery, when measuring $\overline{Z_1} \otimes \overline{Z_2}$. One can observe a very similar structure of the gauge operators where the logical X operator of one of the two patches is added to the gauge group of the deformation.



(a) Initial layout (b) Z-stab. after merging (c) X-stab. after merging (d) X (blue) and Z (red) gauge operators

Figure 4.10: (a) Two patches of color codes with facing red boundaries ready for the merging phase of lattice surgery to measure say $Z \otimes Z$. Qubits are located at the vertices and each colored face hosts both a X and Z stabilizer. (b) The Z stabilizers after the merging step on each colored face. (c) The X stabilizers after the merging step on each colored face. (d) Highlighted in red (Z) and blue (X) are the gauge operators of the code deformation procedure not in the stabilizer. The X logical operator of the top patch has to be added to the gauge group. The distance is still that of a single patch because of this.

Interfacing triangular 2D color codes and tetrahedral 3D color codes using these techniques is also very similar. This is interesting since a tetrahedral 3D color code supports a transversal T gate which, once added to Clifford gates, forms a universal set. For this we consider 3D tetrahedral color codes with a string-like Z logical and a membrane like X logical. The main feature to notice is that the four faces of a tetrahedral 3D color code have the same structure as that of a triangular 2D color code. This permits to either perform surgery along an edge or along a face, see Figure 4.11. The (Z) surgery along an edge has again the same protocol and characteristics. For doing surgery along one face,

it is simpler to have the 2D patch to match exactly one face of the 3D patch and then perform joint $Z \otimes Z$ measurements on pairs of matching qubits. One difference with the edge surgery is that now learning the result of the logical $\overline{Z_1} \otimes \overline{Z_2}$ measurement can be done in a single-shot way similar as reading out a single 2D patch. Note that another way of transferring a logical state between a 3D and 2D color code is to perform a so-called "dimensional jump" [31]. The rough idea behind this is to measure everything except one of the four faces of the tetrahedron hosting the code.

But all of these techniques are made less interesting by the fact that one can perform a CNOT directly transversally between the 3D and 2D patches. In the same configuration as for the face surgery, with a 2D patch matching one of the faces of the 3D patch, performing CNOTs between pairs of matching qubits, the control on the 3D side and target on the 2D side realize a logical CNOT in the same direction.

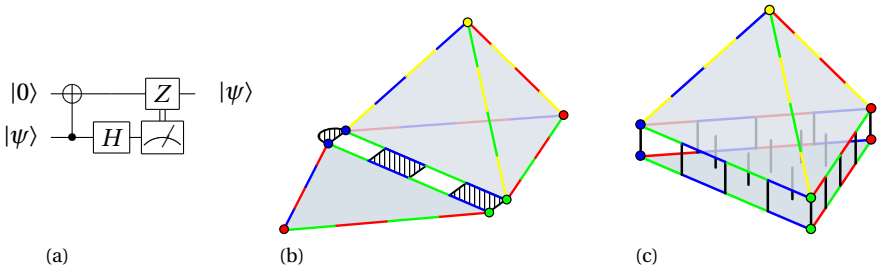


Figure 4.11: (a) One of the "one bit teleportation" circuits that can be used to teleport a magic state from a 3D color code to a 2D one. Realizing the CNOT via measurements requires to do a $Z \otimes Z$ measurement. Several configurations are possible to do the measurement between a triangular 2D color code with a tetrahedral 3D color code. (b) Interfacing them via an edge, the procedure is that of lattice surgery. (c) Interfacing them via a face, one can either jointly measure pairs of qubits to perform a similar operation to lattice surgery or actually directly perform transversal CNOTs between the two faces.

PLAIN SURGERY

We now introduce a new technique with the same goal as lattice surgery, namely performing joint measurements of logical operators, but following a different procedure. One of the less elegant parts of jointly measuring logical operators using lattice surgery (including when measuring jointly more than two logical qubits such as in [21, 22]) is that the outcome of the measurement is learned without any redundancy, imposing a repetition in time for the measurement to be reliable. The difference between lattice surgery and the new procedure, *plain surgery*, will be that the logical measurement is performed with redundancy, so that this part of the protocol can be made more robust to noise.

The idea is to separate the merging and logical measurement of lattice surgery into two distinct steps. The first step deforms the two separated blocks into a single code block where the joint logical operators can be measured redundantly. Possible configurations for surface codes and color codes with the possibility of a joint measurement of two logical qubits with redundancy are shown in Figure 4.12. Since this first step merges the codes, but leaves the logical information unchanged, we call it a *plain* merge. In the

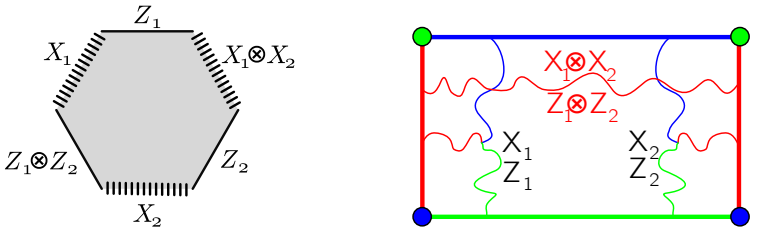


Figure 4.12: (a) The boundary configuration for a hexagonal surface code encoding two qubits. Access to the joint logical Paulis is possible by measuring near the corresponding boundary (but away from the complementary joint logical boundary in order to not also measure single logical Paulis). (b) The boundary configuration of a rectangular color code suited for a joint measurement of $X_1 \otimes X_2$ or $Z_1 \otimes Z_2$. The two logical qubits are associated with string nets either attached to the left or right red boundary. Red strings going from the left boundary to the right boundary are the support for the logical $Z_1 \otimes Z_2$ and $X_1 \otimes X_2$. So measuring all red edges in the X or Z basis permits a joint measurement of the two logical qubits.

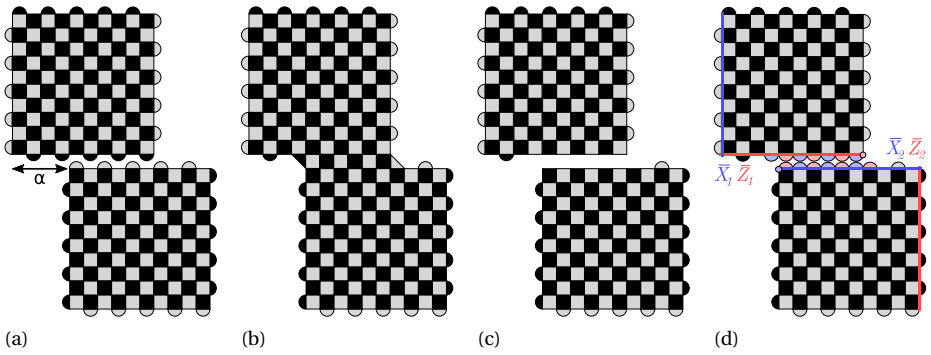


Figure 4.13: (a) & (b) The qubit layouts before and after the plain merge operation. The two patches are offset by a fraction $\alpha \in [0,1]$ of their distance, represented by the double arrow. The number of logical qubits is kept constant during this merge operation. (c) The stabilizers of the subsystem code. (d) The gauge operators and logical operators of the subsystem code. One can see that the distance is guaranteed by the offset between the two blocks. The distance of the separate surface codes is 11, and the distance of the subsystem code is 4.

second step, for the surface code version, we measure the desired logical operator similar to the standard logical measurement of a surface code block. The standard logical Z measurement goes as follows: measure each qubit in the Z basis. Use this information to reconstruct the Z-stabilizer outcomes and correct the X errors using this syndrome. Then reconstruct any Z logical measurement from the single-qubit measurements to give the logical measurement outcome. At this last step since the X errors have been corrected, all the reconstructed Z logical measurment will give the same answer. The same principle can be applied when measuring each qubit in a large region which contains several representatives of a logical operator but not necessarily extending to the full patch. For the color code version a joint measurement of all the red edges is what permits to measure the joint logical. A final deformation step can be used to return to

the original code space.

Looking at each step in more details we can apply the prescription of Section 4.4. The layout for the plain merge operation for the surface code is shown in Figure 4.13a. The patches are placed with an overlap of $(1-\alpha)d$, the X-boundary of one facing the Z-boundary of the other. Then they are merged into a single patch with 3 X-boundaries and 3 Z-boundaries, so two logical qubits, see Figure 4.12a. Logical operators far away from the interface are left unchanged, and the logical information is untouched. When looking at the subsystem code for this deformation, shown in Figure 4.13d, one can see that the distance is guaranteed by the offset αd between the two patches.

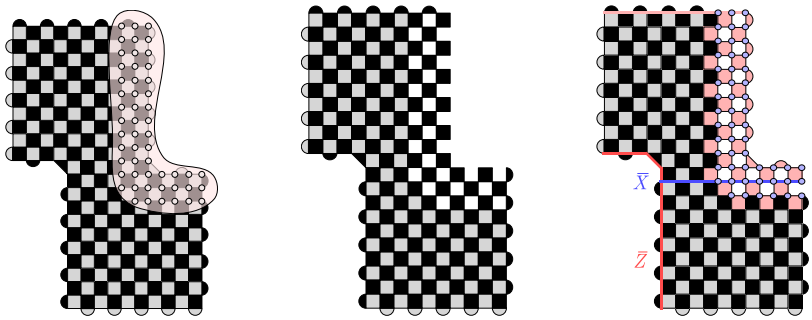
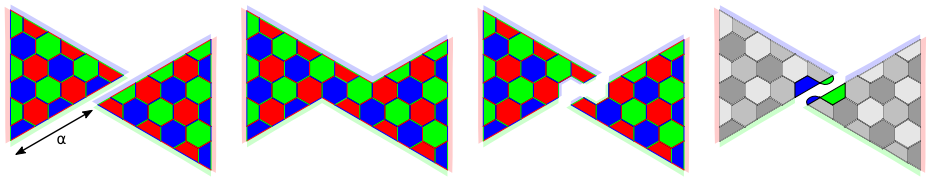


Figure 4.14: (a) The layout where each qubit in the region highlighted are to be measured in the X basis. (b) The stabilizers of the underlying subsystem code C. (c) The gauge operators (in pink) and logical operators of the code. One can see that the distance is guaranteed by the amount of overlap between the two blocks. The distance of the subsystem code is 4.

Then, in this new code, the logical operator $\overline{X}_1 \otimes \overline{X}_2$ is given by a string starting from the top boundary of the top patch and ending on the right boundary of the bottom patch. So, by measuring qubits in the X basis in a region away from the third X-boundary, one can learn $\overline{X}_1 \otimes \overline{X}_2$ but not $\overline{X}_1$ or $\overline{X}_2$. This measurement procedure is depicted in Figure 4.14. One can check that the associated subsystem code has a distance of at least half the overlap between the patches, $(1-\alpha)d/2$. The amount of redundancy in the measurement is also $(1-\alpha)d/2$, which makes this procedure costly in qubit overhead but as we show in [1], it offers a better threshold than the standard lattice surgery technique.

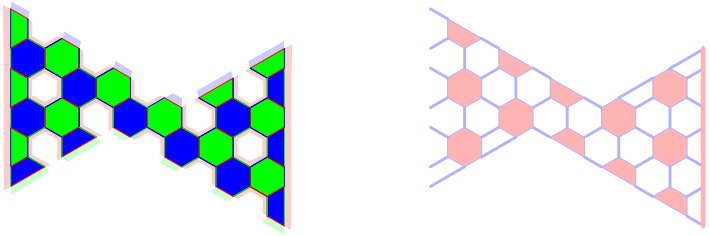
This plain surgery technique is actually more suited to color code patches. This comes from the fact that on rectangular patches of color code with the correct colored boundaries, the joint logical operators, $\overline{X_1} \otimes \overline{X_2}$ or $\overline{Z_1} \otimes \overline{Z_2}$, can be measured redundantly using all the qubits of the patch without destroying the encoded state. This is in contrast with the technique presented above for the surface code where the logical measurement can only involve a subset of the qubits in order to not learn too much logical information. The correct boundary configuration is represented in Figure 4.12b. It consist in two opposite boundaries of the same color (here red) and one of each of the two other colors (here blue and green). In this configuration, the two logical qubits can be associated with string nets attached to either the left or the right red boundary. In this configuration, all red strings joining the left and right boundaries support the joint logical

operators $\overline{X_1} \otimes \overline{X_2}$ or $\overline{Z_1} \otimes \overline{Z_2}$. Hence measuring all red edges in the X basis allows to learn the logical $\overline{X_1} \otimes \overline{X_2}$ without learning any more logical information. Here measuring a red edge in the X basis means measuring the $X \otimes X$ operator of the two qubits forming the edge. Similarly, measuring the red edges in the Z basis allows to learn $\overline{Z_1} \otimes \overline{Z_2}$.



(a) (b) (c) (d)

Figure 4.15: (a) & (b) Layout before and after the plain merge of two triangular 2D color code patches. They are offset by a fraction $\alpha \in [0,1]$ of their distance, represented by the double arrow. Each red blue and green face hosts both a Z and a X stabilizer. The distance in both configurations is 7. (c) The stabilizer generators of the subsystem code associated with the deformation. (d) The purely gauge operators of the subsystem code are represented by the colored faces. They host both Z and X gauge operators. The logical operators are unchanged but can be dressed so that the distance of the subsystem code is reduced to 5.



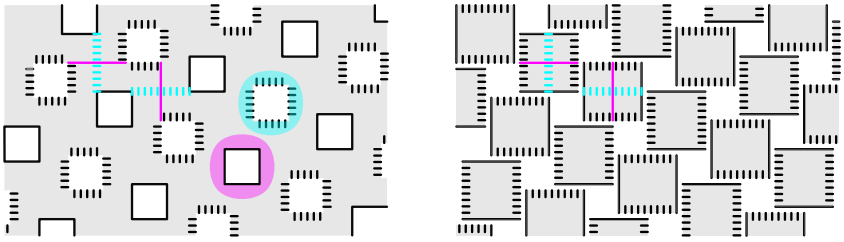
(a) (b)

Figure 4.16: Performing a $X \otimes X$ measurement on a retangular color code patch. (a) The Z-stabilizers during the measurement process are only the blue and green faces, while the X-stabilizers are still the same red, green and blue faces (see Fig. 4.15b). (b) The gauge operators with the color convention of light red for Z and light blue for X. With the original color code colors they correspond to red edges for the X gauge operators and red faces for the Z gauge operators with the addition of the all Z on one of the red boundaries (e.g. the rightmost one).

Similarly to the surface code case, it is possible get to the correct boundary conditions starting from disjoint patches of triangular color code using a plain merge. The plain merge procedure is described in Figure 4.15. The two patches of color codes are placed in a slightly crooked bow-tie pattern. The mismatch of the two patches guarantees a distance for the merging procedure while the size of the overlap dictates the amount of redundancy in the measurement afterwards and the protection during the measurement. One can remark that applying the unfolding procedure [32] of the color code to Figure 4.15a readily yields a configuration similar to Figure 4.13a; although it is not as straightforward between Figure 4.15b and Figure 4.13b. Once merged, the joint logical operator $\overline{X_1} \otimes \overline{X_2}$ can be measured by measuring each red edge in the X basis.

Analyzing this with our prescription we can see that the distance is guaranteed by size of the overlap, i.e. $(1-\alpha)d$, since now a string of Xs from the blue to the green boundary represent a dressed logical. Figure 4.16 shows the Z-stabilizers and the gauge operators of the relevant subsystem code.

TOPOLOGICAL CONCATENATION



(a) (b)

Figure 4.17: (a) Layout for topological concatenation of the surface code. Grayed out surfaces represent surface code bulk. X-type and Z-type holes, represented by trails of parallel segments and solid lines respectively, are placed at the vertices of two square lattices (slightly rotated in the picture) dual to one another. The logical basis is chosen to be geometrically local and two logical qubits are highlighted by their X (blue trails of parallel segments) and Z (pink solid lines) logical operators. This basis is such that measuring every qubits in the X basis around a X-type hole (blue region) measures jointly the logical $X_1X_2X_3X_4$ of the four adjacent logical qubits. Similarly for the Z basis around a Z-type hole (pink region). This allows to measure in a single-shot way the stabilizers of a higher level of concatenated surface code. (b) Interpretation of the chosen logical basis as the plain merge of many single logical qubit surface code patches, i.e. performing a plain merge of these patches yields (a). The logical operators of the same two logical qubits are highlighted.

Using the insight of the possible single-shot measurement within the 2D plane of joint logical operators of several surface code or color code patches, it is possible to imagine realizing something curious which could be called *topological concatenation*. The idea is to concatenate the surface code with the surface code, or the color code with the color code, while staying in the 2D plane by using the type of single-shot logical measurement presented above for the checks of the higher level code.

This mixes two approaches usually made distinct: concatenated codes and topological codes. On the one hand, the idea of concatenated codes is to start with some small code and obtain larger codes with larger distances by successive concatenations of the original code with itself. With this technique the distance grows exponentially with the number of concatenation levels which is very efficient but so does the number of qubits and the size of the stabilizer checks. On the other hand, toplogical codes are codes where the qubits are placed on some geometrical object, e.g. a 2D surface, and the stabilizer checks act only locally with respect to the metric of the surface. Then growing the surface is what increases the distance, but enlarging the surface does not change the local checks so they stay the same size. For topological codes on Euclidean lattices a bound is known, relating the number of physical qubits the dimension and the distance [33], for 2D it says

$$d \leq r n^{\frac{1}{2}}, \tag{4.11}$$

where d is the distance of the code, n the number of physical qubits and r is such that every stabilizer generator can be enclosed in an area of size r^2. This directly bounds the overhead of any topological code in 2D, which we are going to denote as c and define as follows

$$c = \frac{n}{d^2}. \tag{4.12}$$

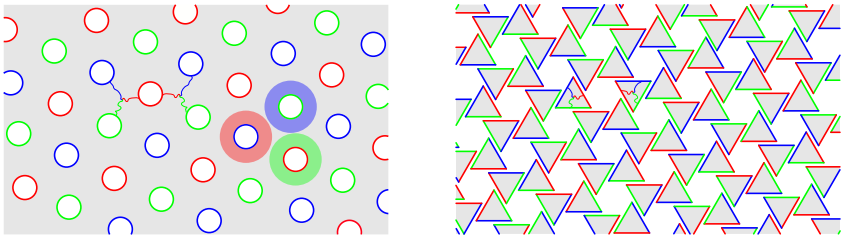
Consider a code with parameters $[[cd^2, 1, d]]$, concatenated k times with itself. It yields a code with parameters $[[c^{k+1}d^{2(k+1)}, 1, d^{k+1}]]$. So this means that if the initial constant, c, was smaller than 1, then further concatenations exponentially decrease the overhead. This seems at first to break the bound in Eq. (4.11) but of course the size of some of the checks also increases exponentially and there is no a priori guarantee that the resulting code can be made local in 2D even if the original code was. This has been explored for example in [34] where the authors proved a threshold for a 2D local implementation of the concatenation of Steane's $[[7,1,3]]$ code (the smallest triangular color code). To achieve this, some circuits, involving moving qubits around with SWAP gates, are carefully designed to perform error correction. Using the techniques developed for plain surgery above we try to get a different approach to realize a similar concatenation scheme where the error correction procedure resembles more that of topological codes.

Consider first the surface code. Figure 4.17 presents a possible layout to concatenate it once with itself using plain surgery. The idea is to start from several patches as in Figure 4.17b and plain merge them together (never plit them back again) to obtain Figure 4.17a. Some holes of X-type and Z-type are formed by this procedure. Then any X-logical going around a X-type hole represent the joint X-logical operator of the four patches around the hole. In other words, if one fills the holes with large stabilizers, one obtains the concatenated code. The rotated surface code, that we used in this chapter, has a constant $c = 1$ so that no real benefit is gained from this. However, an extension of the bound in Eq. (4.11) involving the number of encoded qubits was also found, [35]. It says that

$$kd^2 \leq n/\tilde{c}, \tag{4.13}$$

where k is the number of logical qubits, d the distance of the code, n the number of physical qubits and $\tilde{c}$ some constant. When considering puncturing holes in a big sheet of surface code the best known configuration has $\tilde{c} = 1$. It uses mixed X-type and Z-type boundaries and was proposed in [36]. Figure 4.17 shows an alternative configuration of holes without mixed boundaries (but still both types) also achieving $\tilde{c} = 1$. Note that in order to maintain the distance after the plain merge of all the patches the offset between the patches needs to be at least $\alpha \geq 1/4$ in order for the holes to not introduce lower distance logical operators.

For color codes the constant c can be smaller than 1. For example, multiple copies of 2D triangular color codes based on the 4.8.8 lattice (as in Fig. 4.10) have $c = 1/2$ and the hexagonal lattice (as in Fig. 4.15) have $c = 3/4$ [37]. The setup for the hexagonal color code version is represented in Figure 4.18. The principle is the same as for the surface code version. Some triangular color code patches are disposed as in Figure 4.18b and plain merged together once to obtain Figure 4.18a. Then the colored holes formed can be seen as higher level stabilizers for the concatenated code. We first consider the stabilizer code obtained when these holes are directly included as stabilizers. In order to see if



(a) (b)

Figure 4.18: (a) Layout for topological concatenation of the 2D color code. Grayed out surfaces represent color code bulk. Red, green and blue holes are placed on the faces of a (tri-colored) hexagonal lattice. The logical basis is chosen to be geometrically local and two logical qubits are highlighted by their string net. This basis is such that measuring every red edge in the X (or Z) basis around a blue hole (red region) measures jointly the logical $X_1X_2X_3X_4X_5X_6$ (or $Z_1Z_2Z_3Z_4Z_5Z_6$) of the six adjacent logical qubits. Similarly for blue edges around a green hole (blue region) or green edges around a red hole (green region). This allows one to measure, in a single shot way, the stabilizers of a higher level of concatenated color code. (b) Interpretation of the chosen logical basis as the plain merge of many single logical qubit color code patches. The two string nets of the same logical qubits are highlighted.

one can benefit from the overhead reduction of the concatenation one has to be careful to evaluate the distance of the code obtained. In the limit of full offset of the triangular patches, i.e $\alpha = 1$ from Figure 4.15a, the scheme correspond exactly to the concatenation of the codes. The perimeter of each hole is of length $6d$ where d is the distance of one triangular patch. This shows that it is possible to obtain the reduced overhead in a planar layout but with large stabilizers. Decreasing the offset α shrinks the holes. In the other limit of zero offset, i.e. $\alpha = 0$ from Figure 4.15a, the higher level checks actually become of the same size of the lower level ones, but the overall distance is reduced by a factor 2. An easy way to see this is to see that a red edge in the higher level of concatenation corresponds to two triangular patches. When two triangular patches are plain merged with zero offset ($\alpha = 0$), the distance of the joint logical operators is the same as that of a single one. So forming the logical operator with red edges in the higher level will have a weight divided by two compared to the actual concatenated code. In general with an offset of $\alpha \in [0,1]$, the overall distance is multiplied by a factor $(1+\alpha)/2$. This gives the following parameters for the code concatenated k times with an offset of α

$$[[N,K,D]] = \left[\left[c^{k+1}d^{2(k+1)}, 1, \left(\frac{1+\alpha}{2}\right)^k d^{k+1}\right]\right]. \tag{4.14}$$

These parameters are that of the stabilizer code with stabilizers where the holes are. The biggest ones, from the higher level of concatenation have a perimeter of $6(\alpha d)^k$. Hence in the bound (4.11), one should have $r \propto (\alpha d)^k$.

Measuring stabilizers with very large weight is not realistic, so we would like to use the measurement techniques of plain surgery to measure these high weight stabilizers using only small weight measurements. We know it is not possible to find a way to measure all the checks of the concatenated code in Eq. (4.14) by doing only local measurements on the scale of the lowest level triangles as this would violate the bound (4.11).

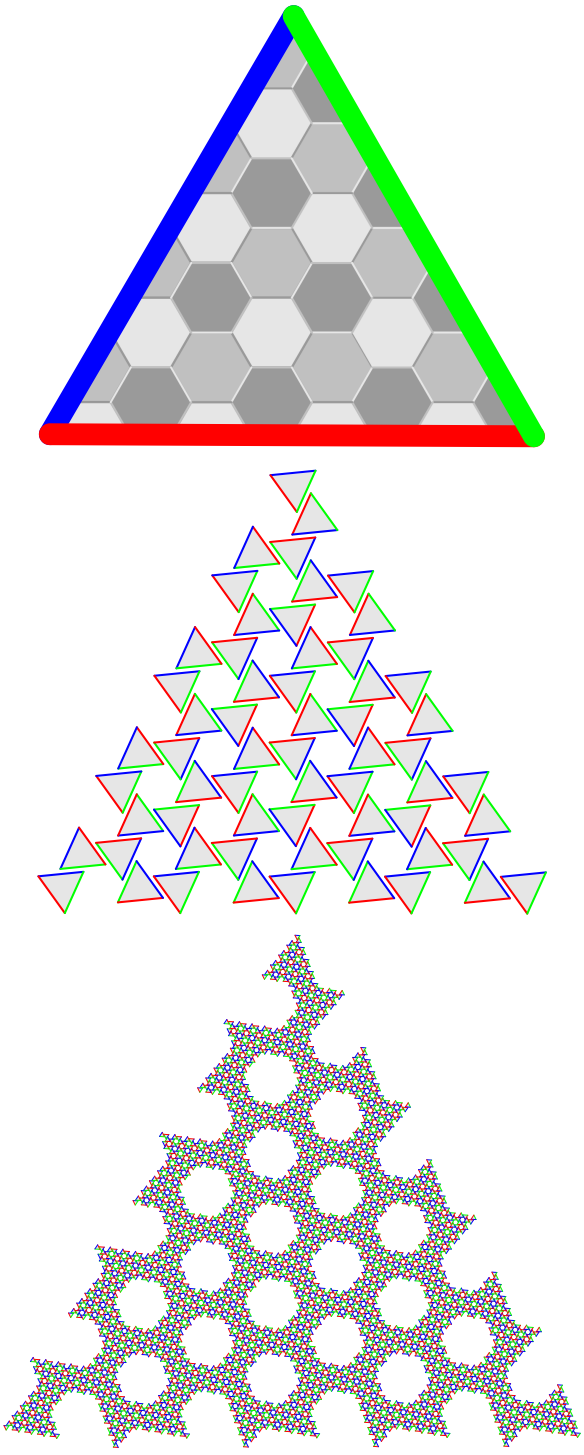


Figure 4.19: Zero, one and two levels of concatenation (before plain merge) with the $d = 9$ hexagonal color code with an offset of 5. Details of the lowest level are grayed out and not shown in level one and two.

And indeed the measurement scheme we find is such that the measurements of the checks of different levels of concatenation cannot be conducted at the same time.

The first thing to note is that with full offset, $\alpha = 1$, the distance is the best possible but it is not possible to use the measuring technique of plain surgery. Indeed there is no region with constant width around the holes to measure colored edges in, see Figure 4.18a. On the opposite case, with zero offset, $\alpha = 0$, the holes can be measured directly as stabilizers but the distance is reduced so that there is no longer an advantage for the concatenation. So α needs to be chosen the largest possible for better overhead advantage, and smallest possible for better measurement reliability. The overhead of the overall scheme, $C = N/D^2$ is given by

$$C = \left(\frac{4c}{(1+\alpha)^2}\right)^{k+1}. \tag{4.15}$$

For the overhead to decrease with the number of concatenations, we have to choose

$$\alpha > 2\sqrt{c} - 1, \tag{4.16}$$

For the 4.8.8 color code, which has $c = 1/2$, this implies a choice of $\alpha > \sqrt{2}-1 \simeq 0.414$. For the hexagonal color code, which has $c = 3/4$, this implies a choice of $\alpha > \sqrt{3}-1 \simeq 0.732$. Figure 4.19 represents level zero one and two (before plain merge) of the concatenation of the distance 9 hexagonal color code with an offset of 5 (this is not within the prescribed value for α but was easier to draw).

The idea of the scheme is then to perform error correction of the lowest level, as one would with standard color code patches, on a short time scale. Then on larger and larger time scales one would perform a measurement of the higher level stabilizers by measuring colored edges in full regions around holes. The reliability of these measurements depends on the size of the region in which edges are measured.

How this scheme performs in practice is still an open question. In particular it would be interesting to compare, for a target logical error rate, its overhead with that of a standard triangular color code. It would also be particularly interesting to study its threshold to prove that it is non-zero and compare it to schemes such as [34].

4.5. LOGICAL OPERATION OF A CODE DEFORMATION

In this section we show how to look at what logical operation is realized by a code deformation. We take the example of lattice surgery with the surface code and prove that it indeed realizes the desired operation at the logical level. In this section, we denote the set of physical qubits as Q. For any subset of k qubits, $s = \{j_1, \ldots, j_k\} \subset \mathsf{Q}$, we denote the operator composed of a Pauli Z resp. X on each qubit in s as $Z(s)$, resp. $X(s)$, i.e.

$$Z(s) = Z_{j_1} \otimes \cdots \otimes Z_{j_k}, \qquad X(s) = X_{j_1} \otimes \cdots \otimes X_{j_k}.$$

4.5.1. MERGE OPERATION

The setting for the merge operation is drawn in Figure 4.3a. The starting code, C_{split}, with stabilizer $\mathscr{S}_{\text{split}}$, consists of two adjacent $L \times L$ patches of rotated surface code with

the opposite boundaries being supports for their $\overline{Z}$ operators. We label the upper logical qubit as 1 and the lower qubit as 2. The new code, C_{merged}, with stabilizer $\mathcal{S}_{\text{merged}}$, consists of only one $2L \times L$ patch of rotated surface code.

We define the subsystem code, C, and its gauge group, $\mathcal{G}$, as specified in Section 4.4, see Figure 4.9. Notably, we exclude from the center of $\tilde{\mathcal{G}}$ the logical operator $\overline{Z}_1\overline{Z}_2 \in \mathcal{S}_{\text{merged}}$. We therefore add $\overline{X}_1$ to $\tilde{\mathcal{G}}$ to form $\mathcal{G}$, and so have $\overline{X}_1 \in \mathcal{L}_g$. Call $\mathcal{I}$ the set of intermediary plaquettes (red plaquettes in Figure 4.3a) to be measured to perform the merge operation. For $p \in \mathcal{I}$ we have $Z(p) \in \mathcal{L}_g$, these are the gauge operators to be fixed by the merge operation. For each $p \in \mathcal{I}$, one measures the operator $Z(p)$ and let its outcome be m_p.

To explain the action of the merge operation at the logical level, we first prove that this operation transforms code states of the two original $L \times L$ patches of surface code into code states of the $2L \times L$ patch surface code with some X errors. To accomplish this, we use the standard prescription from the Gottesman-Knill theorem [27]. It is straightforward to see that the original Z checks stay unchanged, and the newly-measured checks, the $p \in \mathcal{I}$, are added, with sign m_p. The original X checks all commute with the new intermediary Z checks except for the two-body boundary checks between the two patches, which are also part of $\mathcal{L}_g$. Those boundary checks can be merged in pairs in order to commute with the new Z checks. The situation is then the same as depicted in Figure 4.3b.

The product of all measurement outcomes gives the desired outcome for the $\overline{Z}_1\overline{Z}_2$ measurement, we denote it as

$$m_L = \prod_{p \in \mathcal{I}} m_p.$$

Then one fixes the gauge by applying the conjugate X-gauge operators to the $Z(p)$ with $m_p = -1$. Let's call c_{m_L} the set of qubits involved in this fixing operation. Note that when $m_L = +1$ then the correction is equivalent to a stabilizer in $\mathcal{S}_{\text{split}}$ whereas when $m_L = -1$, the correction is equivalent to $\overline{X}_1$. Then, the full merge operation at the physical qubit level is easily written as

$$X(c_{m_L}) \cdot \left(\prod_{p \in \mathcal{I}} \frac{\mathbb{1} + (-1)^{m_p} Z(p)}{2} \right) = \left(\prod_{p \in \mathcal{I}} \frac{\mathbb{1} + Z(p)}{2} \right) \cdot X(c_{m_L}).$$

Due to the definition of $X(c_{m_L})$, commuting it through the Z projections eliminates the $(-1)^{m_p}$ terms.

To determine the logical operation realised by this procedure, we use encoding isometries of C_{split} and C_{merged}, called E_{split} and E_{merged}, respectively. These isometries map unencoded logical states to code states in the full physical Hilbert space. Since C_{split} contains two logical qubits and C_{merged} contains only one, the isometries have the following signatures:

$$E_{\text{split}} : \mathbb{C}^2 \otimes \mathbb{C}^2 \to \mathbb{C}^{2Q}, \qquad E_{\text{merged}} : \mathbb{C}^2 \to \mathbb{C}^{2Q}.$$

Let $\tilde{M}_{m_L}$ be the operation on the logical level, which can be expressed as

$$\begin{aligned}\tilde{M}_{m_L} &: \mathbb{C}^2 \otimes \mathbb{C}^2 \to \mathbb{C}^2, \\ \tilde{M}_{m_L} &= \left(E_{\text{merged}}\right)^\dagger \cdot \left(\prod_{p \in \mathscr{I}} \frac{\mathbb{1} + Z(p)}{2}\right) \cdot X(c_{m_L}) \cdot E_{\text{split}}.\end{aligned} \tag{4.17}$$

An important fact about encoding isometries E is that, if S is a stabilizer of the code and $\overline{L}$ a representative for the logical operator L, then

$$S \cdot E = E, \tag{4.18}$$

$$\overline{L} \cdot E = E \cdot L, \tag{4.19}$$

where L is the corresponding physical operator. This means that $\tilde{M}_{m_L}$, defined in Eq. (4.17), simplifies to

$$\tilde{M}_{m_L} = \left(E_{\text{merged}}\right)^\dagger \cdot E_{\text{split}} \cdot X_1^{(1-m_L)/2}. \tag{4.20}$$

To show this, we use the fact that for all $p \in \mathscr{I}$, $Z(p)$ is a stabilizer of C_{merged} and the correction $X(c_+)$ is in $\mathscr{S}_{\text{split}}$ whereas $X(c_-)$ is a representative of $\overline{X}_1$ in C_{split}.

To show that the operation $\tilde{M}_{m_L}$ is equal to M_{m_L}, as defined in Eq. (4.1) and Eq. (4.2), one can analyse how $\tilde{M}_{m_L}$ acts on the computational basis, i.e. we track how it transforms the stabilizers of those states. For example, the state $|00\rangle$ is stabilized by Z_1 and Z_2, this means that

$$\begin{aligned}\tilde{M}_+ |00\rangle &= \left(E_{\text{merged}}\right)^\dagger \cdot E_{\text{split}} |00\rangle \\ &= \left(E_{\text{merged}}\right)^\dagger \cdot E_{\text{split}} \cdot Z_1 |00\rangle \\ &= \left(E_{\text{merged}}\right)^\dagger \cdot \overline{Z}_1 \cdot E_{\text{split}} |00\rangle \\ &= Z \cdot \left(E_{\text{merged}}\right)^\dagger \cdot E_{\text{split}} |00\rangle \\ &= Z \cdot \tilde{M}_+ |00\rangle ,\end{aligned}$$

and therefore $\tilde{M}_+ |00\rangle$ is stabilized by Z. Here, we have used the properties of the encoding isometries and the fact that a representative $\overline{Z}_1$ for C_{split} is also a representative $\overline{Z}$ for C_{merged}. Doing the same with the other stabilizer, Z_2, also yields Z as a stabilizer (so $Z_1 Z_2$ yields the identity). One can also verify that $\tilde{M}_+ |00\rangle$ is not stabilized by $-Z$ by reversing the previous equalities and therefore $\langle Z \rangle$ is the full stabilizer group of $\tilde{M}_+ |00\rangle$. Looking now at $\tilde{M}_- |00\rangle$ one can see that Z_2 also yields Z but Z_1 will yield $-Z$, indeed

$$\begin{aligned}\tilde{M}_- |00\rangle &= \left(E_{\text{merged}}\right)^\dagger \cdot E_{\text{split}} \cdot X_1 |00\rangle \\ &= \left(E_{\text{merged}}\right)^\dagger \cdot E_{\text{split}} \cdot X_1 \cdot Z_1 |00\rangle \\ &= -\left(E_{\text{merged}}\right)^\dagger \cdot \overline{Z}_1 \cdot E_{\text{split}} \cdot X_1 |00\rangle \\ &= -Z \cdot \left(E_{\text{merged}}\right)^\dagger \cdot E_{\text{split}} \cdot X_1 |00\rangle \\ &= -Z \cdot \tilde{M}_- |00\rangle .\end{aligned}$$

Hence, $\tilde{M}_- |00\rangle$ is both stabilized by Z and $-Z$, and is therefore the null vector. In other words, the state $|00\rangle$ will never give an outcome -1 for m_L, which is what we expect. The

$\mathscr{S}$	State	$\tilde{M}_+$ $\mathscr{S}$	State	$\tilde{M}_-$ $\mathscr{S}$	State
$\langle Z_1, Z_2\rangle$	$\|00\rangle$	$\langle Z\rangle$	$\|0\rangle$	$\langle Z, -Z\rangle$	0
$\langle Z_1, -Z_2\rangle$	$\|01\rangle$	$\langle Z, -Z\rangle$	0	$\langle Z\rangle$	$\|0\rangle$
$\langle -Z_1, Z_2\rangle$	$\|10\rangle$	$\langle -Z, Z\rangle$	0	$\langle -Z\rangle$	$\|1\rangle$
$\langle -Z_1, -Z_2\rangle$	$\|11\rangle$	$\langle -Z\rangle$	$\|1\rangle$	$\langle -Z, Z\rangle$	0

Table 4.1: How $\tilde{M}_\pm$ transforms the computational basis states characterised by their stabilizer group.

full results (shown in Table 4.1) indicate that

$$\tilde{M}_+ = \alpha_+ |0\rangle\langle 00| + \beta_+ |1\rangle\langle 11|$$
$$\tilde{M}_- = \alpha_- |0\rangle\langle 01| + \beta_- |1\rangle\langle 10|,$$

for some non-zero complex numbers $\alpha_\pm$ and $\beta_\pm$. To complete the proof, we verify that there are no relative phases or amplitude differences between $\alpha_\pm$ and $\beta_\pm$. To see that, one can look at the action of $\tilde{M}_{m_L}$ on the Bell states. For $\tilde{M}_+$ we look at the Bell state $(|00\rangle + |11\rangle)/\sqrt{2}$, stabilized by $\langle X_1X_2, Z_1Z_2\rangle$ and for $\tilde{M}_-$ the Bell state $(|01\rangle + |10\rangle)/\sqrt{2}$ stabilized by $\langle X_1X_2, -Z_1Z_2\rangle$. The important fact is that a representative $\overline{X}_1\overline{X}_2$ for C_{split} is also a representative of $\overline{X}$ for C_{merged}. That is to say

$$\tilde{M}_+ \frac{|00\rangle + |11\rangle}{\sqrt{2}} = \gamma_+ \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$
$$\tilde{M}_- \frac{|01\rangle + |10\rangle}{\sqrt{2}} = \gamma_- \frac{|0\rangle + |1\rangle}{\sqrt{2}},$$

for some non-zero complex numbers $\gamma_\pm$. By linearity of $\tilde{M}_{m_L}$ we can conclude that $\alpha_+ = \beta_+ = \gamma_+$ and that $\alpha_- = \beta_- = \gamma_-$. In conclusion, we have shown that $\tilde{M}_{m_L} \propto M_{m_L}$, meaning that it performs the desired logical operation.

4.5.2. SPLIT OPERATION

For the Z-split operation one reverses the roles of C_{split} and C_{merged}. The starting point is the same as shown in Figure 4.3b, without $\pm$ terms in the middle. Then, in order to split the patch, one has to split each four-body X stabilizer in the middle row into a pair of two-body X stabilizers. Those stabilizers are shown with $\pm$ signs on Figure 4.3c. They commute with everything except for the central row of Z-plaquettes. One can see that measuring them will remove those Z-plaquettes from the stabilizer group, but keep the product of all those plaquettes, the logical $\overline{Z}_1\overline{Z}_2$ of the two separate patches. Note that it is sufficient to measure only the top (or bottom) row of two-body X-checks as the bottom (or top) one is then the product of those and the previous four-body X-checks. This also means that the outcomes of those two-body checks are perfectly correlated between facing pairs. Letting $\mathscr{I}$ be the set of the top row of those checks and $m_p = \pm 1$ the measurement outcome of the two-body plaquette p, the operation performed is then

$$\prod_{p\in\mathscr{I}} \frac{\mathbb{1} + (-1)^{m_p} X(p)}{2}.$$

Then, to correct to standard surface codes with no remaining minus signs, one has to apply some of the previous Z-plaquettes that were removed from the stabilizer, correcting the correlated facing X-checks. Labeling the set of qubits affected by the correction c, one has

$$Z(c)\cdot\prod_{p\in\mathscr{I}}\frac{\mathbb{1}+(-1)^{m_p}X(p)}{2}=\prod_{p\in\mathscr{I}}\frac{\mathbb{1}+X(p)}{2}\cdot Z(c).$$

This operation corresponds to S_+, defined in Eq. (4.1). If one wants to implement S_-, defined in Eq. (4.2), then one has to additionally apply a logical representative of X on the first patch, $\overline{X}_1$. The choice of one or the other version is conditioned by the previous m_L outcome that we received during the merging step. Then, to show that this performs the correct logical operation, we analyse

$$\tilde{S}_{m_L}=\left(E_{\text{split}}\right)^\dagger\cdot\overline{X}_1^{\frac{1-m_L}{2}}\cdot\prod_{p\in\mathscr{I}}\frac{\mathbb{1}+X(p)}{2}\cdot Z(c)\cdot E_{\text{merged}},$$

which, using the properties of the encoding isometries, $\tilde{S}_{m_L}$ simplifies to

$$\tilde{S}_{m_L}=X_1^{\frac{1-m_L}{2}}\cdot\left(E_{\text{split}}\right)^\dagger\cdot E_{\text{merged}}. \tag{4.21}$$

At this point, recalling Eq. (4.20), we can see that

$$\tilde{S}_\pm=\left(\tilde{M}_\pm\right)^\dagger=(M_\pm)^\dagger=S_\pm,$$

which concludes the proof of correctness for the split operation. Note that it was crucial to apply the intermediary Z-plaquettes (in $\mathscr{L}_g$) as the correction. If we had instead applied a string of Z-flips between the faulty X-plaquettes, the correction would not be absorbed in the encoding map of C_{merged} and moreover would anti-commute with any representative $\overline{X}$ of C_{merged} or $\overline{X}_1\overline{X}_2$ of C_{split} and therefore flip the phase between the $|0\rangle$ and $|1\rangle$ states.

4.5.3. GENERAL CODE DEFORMATION OPERATION

Having analyzed the merge and the split of lattice surgery we can see that the overall structure will always be the same for any code deformation: the operation on the logical level is given by

$$L_{\text{prep}}\cdot(E_{\text{new}})^\dagger\cdot E_{\text{old}}\cdot L_{\text{meas}}(m),$$

where E_{old} and E_{new} are the encoding isometries of the codes before and after the deformation, L_{prep} is a conjugate Pauli operator to some elements in $\mathscr{M}_{\text{prep}}$ (see Section 4.4) depending on what preparation one wants to achieve and $L_{\text{meas}}(m)$ is a conjugate Pauli operator to some elements in $\mathscr{M}_{\text{meas}}$ (see Section 4.4) depending on what measurement outcomes where observed (denoted m). Overall the operation will always directly depend on the relation between the logical operators of the old versus the new code. Note that it can be that one step of deformation seems trivial on the logical level if it just deforms the logical operators. But several such steps can gradually deform operators in an overall non-trivial way. This is the case for example when braiding holes around one another in the surface code. One needs to analyze several steps together to see that overall the logical operators have been deformed in a non trivial way.

4.6. DISCUSSION

We have illustrated how to describe current measurement-based operations in 2D topological quantum computing using the gauge fixing technique. We have shown that, by using the formalism of gauge fixing, the fault tolerance analysis of these code deformation and lattice surgery protocols is considerably simplified, their error correction and gauge fixing schemes also become clear. We have also presented a new code deformation technique, plain surgery, and showed potential applications for it. Although this gauge fixing formalism does not provide direct guidelines on how to design code deformation protocols for a desired logical operation, it does provide an easy way to check the fault-tolerance of protocols and search for new ones via iterations of trial and error.

Moreover, this formalism applies not only to 2D topological codes, but more generally to any stabilizer code. In the general case (non-topological codes), the analysis of fault-tolerance in the presence of measurement errors becomes more involved, in particular with respect to how much repetition is really needed, see for example [38, 39]. We leave for future work how to obtain general and simple criteria for fault-tolerance.

REFERENCES

[1] C. Vuillot, L. Lao, B. Criger, C. G. Almudéver, K. Bertels, and B. M. Terhal, *Code deformation and lattice surgery are gauge fixing,* New Journal of Physics **21**, 033028 (2019).

[2] P. W. Shor, *Algorithms for quantum computation: Discrete logarithms and factoring,* in *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on* (IEEE, 1994) pp. 124–134.

[3] S. Jordan, *Quantum algorithm zoo* (2011).

[4] D. Ristè, S. Poletto, M.-Z. Huang, A. Bruno, V. Vesterinen, O.-P. Saira, and L. DiCarlo, *Detecting bit-flip errors in a logical qubit using stabilizer measurements,* Nature communications **6**, 6983 (2015).

[5] J. Kelly, R. Barends, A. Fowler, A. Megrant, E. Jeffrey, T. White, D. Sank, J. Mutus, B. Campbell, Y. Chen, and others, *State preservation by repetitive error detection in a superconducting quantum circuit,* Nature **519**, 66 (2015).

[6] A. M. Steane, *Error correcting codes in quantum theory,* Phys. Rev. Lett. **77**, 793 (1996).

[7] E. Knill and R. Laflamme, *Concatenated quantum codes,* arXiv:9608012 (1996).

[8] D. Bacon, *Operator quantum error-correcting subsystems for self-correcting quantum memories,* Phys. Rev. A **73**, 012340 (2006).

[9] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, *Surface codes: Towards practical large-scale quantum computation,* Phys. Rev. A **86**, 032324 (2012).

[10] A. Y. Kitaev, *Fault-tolerant quantum computation by anyons,* Annals of Physics **303**, 2 (2003).

[11] H. Bombin and M. A. Martin-Delgado, *Topological Quantum Distillation,* Physical Review Letters **97**, 180501 (2006).

[12] D. S. Wang, A. G. Fowler, and L. C. Hollenberg, *Surface code quantum computing with error rates over 1%,* Phys. Rev. A **83**, 020302 (2011).

[13] H. Bombín and M. A. Martin-Delgado, *Quantum measurements and gates by code deformation,* Journal of Physics A: Mathematical and Theoretical **42**, 095302 (2009).

[14] C. Horsman, A. G. Fowler, S. Devitt, and R. Van Meter, *Surface code quantum computing by lattice surgery,* New Journal of Physics **14**, 123011 (2012).

[15] A. Paetznick and B. W. Reichardt, *Universal fault-tolerant quantum computation with only transversal gates and error correction,* Phys. Rev. Lett. **111**, 090505 (2013).

[16] H. Bombin, *Clifford gates by code deformation,* New Journal of Physics **13**, 043005 (2011).

[17] A. J. Landahl and C. Ryan-Anderson, *Quantum computing by color-code lattice surgery,* arXiv:1407.5103 (2014).

[18] S. Bravyi, *Fault-tolerant quantum computing by code deformation,* QIP Tutorial (2016).

[19] H. Poulsen Nautrup, N. Friis, and H. J. Briegel, *Fault-tolerant interface between quantum memories and quantum processors,* Nature Communications **8**, 1321 (2017).

[20] B. J. Brown, K. Laubscher, M. S. Kesselring, and J. R. Wootton, *Poking Holes and Cutting Corners to Achieve Clifford Gates with the Surface Code,* Phys. Rev. X **7**, 021029 (2017).

[21] D. Litinski and F. v. Oppen, *Lattice Surgery with a Twist: Simplifying Clifford Gates of Surface Codes,* Quantum **2**, 62 (2018).

[22] A. G. Fowler and C. Gidney, *Low overhead quantum computation using lattice surgery,* arXiv:1808.06709 (2018).

[23] M. Vasmer and D. E. Browne, *Universal Quantum Computing with 3d Surface Codes,* arXiv:1801.04255 (2018).

[24] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, *Topological quantum memory,* Journal of Mathematical Physics **43**, 4452 (2002).

[25] R. Raussendorf and J. Harrington, *Fault-Tolerant Quantum Computation with High Threshold in Two Dimensions,* Phys. Rev. Lett. **98**, 190504 (2007).

[26] H. Bombín, *Topological Order with a Twist: Ising Anyons from an Abelian Model,* Phys. Rev. Lett. **105**, 030403 (2010).

[27] D. Gottesman, *The Heisenberg Representation of Quantum Computers,* arXiv:quant-ph/9807006 (1998), arXiv: quant-ph/9807006.

[28] D. Gottesman, *Fault-Tolerant Quantum Computation with Higher-Dimensional Systems,* arXiv:9802007 (1998).

[29] N. de Beaudrap and D. Horsman, *The ZX calculus is a language for surface code lattice surgery,* arXiv:1704.08670 (2017).

[30] D. Poulin, *Stabilizer formalism for operator quantum error correction,* Phys. Rev. Lett. **95**, 230504 (2005).

[31] H. Bombín, *Dimensional jump in quantum error correction,* New Journal of Physics **18**, 043038 (2016).

[32] A. Kubica, B. Yoshida, and F. Pastawski, *Unfolding the color code,* New Journal of Physics **17**, 083026 (2015).

[33] S. Bravyi and B. Terhal, *A no-go theorem for a two-dimensional self-correcting quantum memory based on stabilizer codes,* New Journal of Physics **11**, 043029 (2009).

[34] K. M. Svore, D. P. DiVincenzo, and B. M. Terhal, *Noise Threshold for a Fault-Tolerant Two-Dimensional Lattice Architecture,* Quantum Info. and Comput. **7**, 297 (2007).

[35] S. Bravyi, D. Poulin, and B. M. Terhal, *Tradeoffs for Reliable Quantum Information Storage in 2d Systems,* Phys. Rev. Lett. **104**, 050503 (2010).

[36] N. Delfosse, P. Iyer, and D. Poulin, *Generalized surface codes and packing of logical qubits,* arXiv:1606.07116 [quant-ph] (2016), arXiv: 1606.07116.

[37] A. J. Landahl, J. T. Anderson, and P. R. Rice, *Fault-tolerant quantum computing with color codes,* (2011), arXiv:1108.5738 [quant-ph] .

[38] E. T. Campbell, *A theory of single-shot error correction for adversarial noise,* Quantum Science and Technology **4**, 025006 (2019).

[39] O. Fawzi, A. Grospellier, and A. Leverrier, *Constant Overhead Quantum Fault-Tolerance with Quantum Expander Codes,* in *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)* (2018) pp. 743–754.

5

QUANTUM PIN CODES

A class of quantum CSS codes, quantum pin codes, are introduced. Quantum pin codes are a generalization of quantum color codes and Reed-Muller codes. A lot of the structure and properties of color codes carries over to pin codes. Pin codes have gauge operators, an unfolding procedure and their stabilizers form multi-orthogonal spaces. This last feature makes them interesting for devising magic-state distillation protocols. Examples of these codes and their properties are studied.

At the time of submission of this thesis the following chapter is under review by IEEE Transactions on Information Theory. The preprint can be found on the arXiv:1906.11394 [1]

5.1. INTRODUCTION

THE REALIZATION of a fault-tolerant universal quantum computer is a tremendous challenge. At each level of the architecture, from the hardware implementation up to the quantum software, there are difficult problems that need to be overcome. Hovering in the middle of the stack, quantum error correcting codes influence both hardware design and software compilation. They play a major role not only in mitigating noise and faulty operations but also in devising protocols to distill the necessary resources granting universality to an error corrected quantum computer [2]. The study and design of quantum error correcting codes is therefore one of the major tasks to be undertaken on the way to universal quantum computation.

A well-studied class of quantum error correcting codes are Calderbank-Shor-Steane codes (CSS codes) [3, 4], which are a kind of stabilizer quantum codes [5, 6]. The advantage of CSS codes is their close connection to linear codes which have been studied in classical coding theory. A CSS code can be constructed by combining two binary linear codes. Roughly speaking, one code performs parity checks in the Pauli X-basis and the other performs parity checks in the Pauli Z-basis. Not any two binary linear codes can be used: it is necessary that any two pairs of code words from each code space have to have even overlap. Common classical linear code constructions, e.g. random constructions, do not sit well with this restriction and can therefore not be applied to construct CSS codes. Several families of CSS codes have been devised based on geometrical, homological or algebraic constructions [7–18], however, it is still open which parameters can be achieved.

Besides being able to protect quantum information, quantum error correcting codes must also allow for some mechanism to process the encoded information without lifting the protection. It is always possible to find some operations realizing a desired action on the encoded information but these operations may spread errors in the system. One should restrict oneself to fault-tolerant operations which do not spread errors. For instance, acting separately on each qubit of a code cannot spread single qubit errors to multi-qubit errors. This is called a transversal gate, but not any code admits such gates. More generally, for many codes in the CSS code family it is possible to fault-tolerantly implement Clifford operations, which are all unitary operations preserving Pauli operators under conjugation. Clifford operations by themselves do not form a universal gate set. Several techniques to obtain a universal gate set, by supplementing the non-Clifford T gate to Cliffords for example, have been devised [19, 20], among which magic state distillation is currently the most promising candidate.

In this work we introduce a new class of CSS codes, which we call *quantum pin codes*. These codes form a large family while at the same time having structured stabilizer generators, namely they form multi-orthogonal spaces. This structure is necessary for codes to admit transversal phase gates and it can be leveraged to obtain codes that can be used within magic state distillation protocols. Moreover the construction of pin codes differs substantially from previous approaches making it an interesting space to explore further.

In Section 5.2, after introducing some notations and terminology, we define quantum pin codes, explain their relation to quantum color codes and give some concrete approaches to construct them. In Section 5.3, we discuss the conditions for transversal implementation of phase gates on a CSS code. In Section 5.4, we investigate the prop-

erties of pin codes. Finally in Section 5.5, we study concrete examples of pin codes obtained from Coxeter groups and chain complexes as well as applications for magic state distillation.

5.2. PIN CODES

5.2.1. TERMINOLOGY AND FORMALISM

Consider $D+1$ finite, disjoint sets, $(C_0,\dots,C_D)$ which we call *levels*. The elements in each of the levels are called *pins*. If a pin c is contained in a level C_j then j is called the *rank* of c. Since all the C_j are disjoint each pin has a unique rank.

Consider a $(D+1)$-ary relation on the $D+1$ levels $C_0,\dots,C_D$, that is to say a subset of their Cartesian product $F \subset C_0 \times \cdots \times C_D$. The tuples in the relation F will be called *flags*.

A subset of the ranks, $t \subset \{0,\dots,D\}$, is called a *type*. We will consider tuples of pins coming from a subset of the levels selected by a type t and call them *collections* of pins of type t. A collection of pins of type $t = \{j_1,\dots,j_k\}$, is therefore an element $s \in C_{j_1} \times \cdots \times C_{j_k}$. Note that we can interchangeably view a collection of pins as a tuple or a set as long as no two pins come from the same level in the set.

We now define specific subsets of flags, called pinned sets, using projections.

Definition 1 (Projection of type t)**.** *Given a set of flags F and a type $t = \{j_1,\dots,j_k\}$, the projection, Π_t, is defined as the natural Cartesian product projection acting on the flags, F*

$$\begin{aligned} \Pi_t : \quad F \quad &\to C_{j_1} \times \cdots \times C_{j_k} \\ (c_0,\dots,c_D) \quad &\mapsto (c_{j_1},\dots,c_{j_k}). \end{aligned}$$

Note that the projection of empty type, $\Pi_\emptyset$, is also well defined: for any $f \in F$ we have $\Pi_\emptyset(f) = ()$.

Definition 2 (Pinned set)**.** *Let F be a set of flags, s be a collection of pins of type t and Π_t be the corresponding projection as defined above. We define the pinned set of type t and collection of pins s, $P_t(s)$, as the preimage of s under the projection Π_t,*

$$P_t(s) = \Pi_t^{-1}(s) \subset F.$$

In words: a pinned set is the set of flags whose projection of a given type t yields a given collection of pins, s. A definition of a pinned set which is equivalent to the one given above is

$$P_t(c_{j_1},\dots,c_{j_k}) = F \cap C_0 \times \cdots \times \{c_{j_1}\} \times \cdots \times \{c_{j_k}\} \times \cdots \times C_D, \tag{5.1}$$

where $\{c_{j_i}\}$ denotes the set containing the single element c_{j_i}. The pinned set with respect to the empty type is none other than the full set of flags, F. For convenience, we will refer to a pinned set defined by a collection with k pins as a *k-pinned set*.

If one wants to form a mental image one can imagine a pin-board with pins of different colors on it with one color for each level. Then the flags can be represented by cords each attached to one pin of each level, see Fig. 5.1 as an example.

The structure of pinned sets layed out above is such that they intersect and decompose nicely. This is captured by the following two propositions.

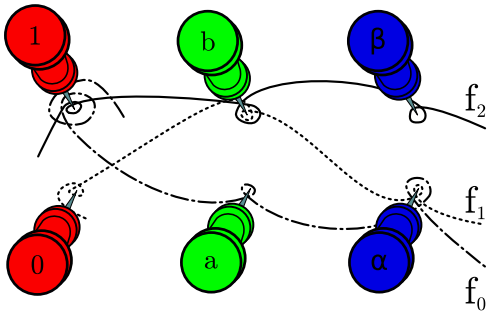


Figure 5.1: Illustration of three levels (red, green and blue) each containing two pins and a relation containing three flags (f_0, f_1 and f_2) symbolized by cords attached to the pins. The pinned set $P_{\text{green}}(b)$ is composed of the flags f_1 and f_2. The pinned set $P_{(\text{red,blue})}(1,\alpha)$ only contains the flag f_0.

Proposition 1 (Intersection of pinned sets)**.** *Let s_1 and s_2 be two collections of pins of types t_1 and t_2 respectively. Then the intersection of the two pinned sets $P_{t_1}(s_1)$ and $P_{t_2}(s_2)$ is either empty or a pinned set of type $t_1 \cup t_2$ characterized by the collection of pins $s_1 \cup s_2$,*

$$P_{t_1}(s_1) \cap P_{t_2}(s_2) = \begin{cases} P_{t_1 \cup t_2}(s_1 \cup s_2) & \text{if } |s_1 \cup s_2| = |t_1 \cup t_2| \\ \emptyset & \text{otherwise.} \end{cases}$$

Proof. The proof follows directly from the alternative characterization of pinned sets given in Eq. (5.1). □

Proposition 2 (Pinned set decomposition)**.** *Let s be a type-t collection of pins and let t' be a type containing t, i.e. $t' \supset t$. The pinned set $P_t(s)$ is partitioned into some number, say m, of pinned sets, each characterized by a type-t' collection of pins containing s, i.e. $s'_j \supset s$,*

$$P_t(s) = \bigsqcup_{j=1}^{m} P_{t'}(s'_j).$$

Proof. Let s be a type-t collection of pins and let t' be a type such that $t' \supset t$. Define the following set of collections of pins

$$S = \Pi_{t'}(P_t(s)),$$

then it is the case that

$$P_t(s) = \bigsqcup_{s' \in S} P_{t'}(s').$$

Indeed, since $t' \supset t$ and $\forall s' \in S$, $s' \supset s$ we have that $\forall s' \in S$, $P_{t'}(s') \subset P_t(s)$ showing the right to left inclusion. The left to right inclusion follows from the definition of S. Finally the fact that the union is disjoint follows from Prop. 1. □

5.2.2. DEFINITION OF AN (x,z)-PIN CODE

Equipped with the notions presented in the previous section, we now construct quantum codes. They are defined by a choice of flags F and two natural integers x and z

which fulfill the condition $x+z \leq D$. The flags are identified with qubits, so that $n = |F|$. The X-stabilizer generators are defined by all the x-pinned sets and the Z-stabilizer generators by the z-pinned sets in the following way. For each x-pinned set we define the Pauli operator acting as the Pauli-X operator on the qubits corresponding to the flags in the pinned set and as the identity on the rest; we then add it to the generating set of the X-stabilizers. We do similarly for the z-pinned sets to form the generating set of the Z-stabilizers. In order to ensure the correct commutation relations between the X- and Z-stabilizers it is sufficient to enforce the following condition on the relation F.

Definition 3 (Pin code relation)**.** *A $(D+1)$-ary relation, F, is a* pin code relation *if all D-pinned sets have even cardinality.*

This property is sufficient for the overlap between two pinned sets to be even every time they are pinned by a small enough number of pins. This is summarized in the following proposition.

Proposition 3 (Even overlap)**.** *Let F be a $(D+1)$-ary pin code relation as per Definition 3. Let x and z be two natural integers such that,*

$$x+z \leq D.$$

Then, the intersection between any x-pinned set and any z-pinned set has even cardinality.

Proof. By Proposition 1, the intersection of a x-pinned set and a z-pinned set is either empty or a pinned set with at most $x+z \leq D$ pins. In turn by Proposition 2, the intersection can be partitionned into D-pinned sets which all have even cardinality since the relation F is a pin code relation. □

Using a pin code relation we can therefore define a CSS code as follows.

Definition 4 ((x,z)-pin code)**.** *Given a pin code relation, F, on $(D+1)$ levels and two natural integers $(x,z) \in \mathbb{N}^2$, such that $x+z \leq D$, we define the corresponding (x,z)-pin code by associating the elements of F with qubits, all the x-pinned sets with X-stabilizer generators and all the z-pinned sets with Z-stabilizer generators. The defined code is a valid CSS code.*

A first remark is that the choice of $x = 0$ (or $z = 0$) is not particularly interesting since in this case there is a single X-stabilizer (or Z-stabilizer) acting on all the qubits.

A second remark is that in the strict case, where $x+z < D$, the code will contain many small logical operators which are naturally identified as gauge operators. This is explained in details in Sec. 5.4.3.

Before explaining how to construct pin code relations we show that quantum color codes are a subclass of pin codes.

5.2.3. RELATION TO QUANTUM COLOR CODES

The formalism and definitions above can be viewed as a generalization of quantum color codes without boundaries [12, 21, 22]. However it is also possible to integrate the notion

of boundaries into the general framework of pin codes but we delay these considerations to Sec. 5.4.5. A D-dimensional color code is defined by a homogeneous, D-dimensional, simplicial complex, triangulating a D-manifold, whose vertices are $(D+1)$-colorable. A D-dimensional, simplicial complex is called homogeneous if every simplex of dimension less than D is a face of some D-simplex. The (Poincaré) dual of a simplicial complex as defined above is sometimes called a colex [21], it consists in a tessellation where the vertices are $(D+1)$-valent and the D-cells are $(D+1)$-colorable. In the original simplicial complex, the qubits are identified with the D-simplices and one chooses two natural integers, $(\tilde{x},\tilde{z}) \in \mathbb{N}^2$ with $\tilde{x}+\tilde{z} \leq D-2$, to define the X- and Z-checks using the $\tilde{x}$- and $\tilde{z}$-simplices in the following way. Each X-stabilizer generator is identified with a $\tilde{x}$-simplex which acts as Pauli-X on all D-simplices in which it is contained. Similarly, each Z-stabilizer generator operate on all the D-simplices as Pauli-Z in which the corresponding $\tilde{z}$-simplex is contained, respectively.

This definition can be restated in the language of pin codes: the $D+1$ levels, $C_0,\dots,C_D$, are indexed by the $D+1$ colors and each level contains the vertices of a given color. The flags, F, are defined using the D-simplices (each containing $D+1$ vertices); this defines a relation, $F \subset C_0 \times \cdots \times C_D$ thanks to the colorability condition as each D-simplex will not contain two vertices of the same color. One can further check that the relation F is a pin code relation as stated in Def. 3. Indeed, D-pinned sets correspond to $(D-1)$-simplices which are contained in exactly two D-simplices since the simplicial-complex triangulates a D-manifold without boundaries. This shows that it is a pin code relation. Then subsets of the $(D+1)$ colors correspond to types and any k-simplex corresponds directly to a collection of pins of type given by the $(k+1)$ different colors of the $(k+1)$ vertices of the k-simplex. The corresponding $(k+1)$-pinned set contains all the D-simplices containing the original k-simplex. As such all the non-empty $(k+1)$-pinned sets are given by all the collection of pins and type corresponding to all the k-simplices. With these consideration, we see that choosing $x=\tilde{x}+1$ and $z=\tilde{z}+1$, the corresponding (x,z)-pin code is the same as the original color code.

An example for $D=2$, based on the hexagonal color code, is shown in figure 5.2a. To summarize: to go from color codes to pin codes one just forgets the geometry, keeping only the $(D+1)$-ary relation given by the $(D+1)$-colored D-simplices.

Importantly, pin codes are more general, as there are pin code relations which are not derived from these specific simplicial complexes. In the next section after recalling a concrete color code construction we give two more general constructions of pin code relations.

5.2.4. CONSTRUCTING PIN CODES

COLOR CODES FROM TILINGS

In [21], the authors explain how to obtain a colex from any tiled D-manifold. The idea is to successively inflate the $(D-1)$-cells, $(D-2)$-cells, …, 0-cells into D-cells. The dual of the tiling obtained is then a $(D+1)$-colorable triangulation of the D-manifold, see Appendix A of [21]. This can also be understood directly, without inflating the cells, as follows: Separate all the cells into $(D+1)$ levels, $C_0,\dots,C_D$, according to their dimensions, i.e. C_j contains all the j-cells. We can now define a $(D+1)$-ary relation on the cells via the incidence relation. Two cells of different dimension are incident if and only if one

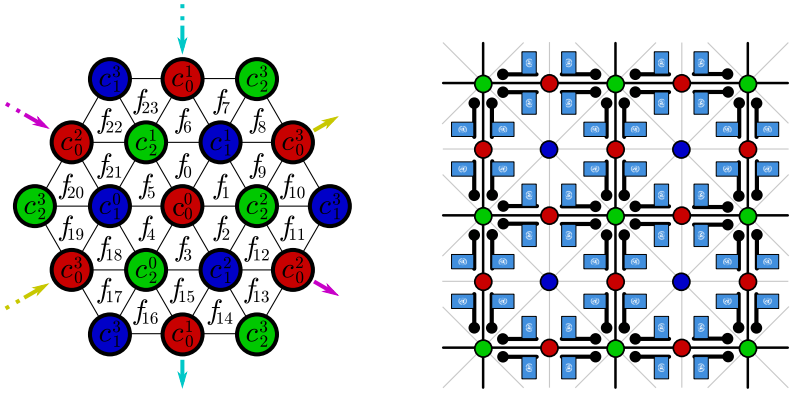


Figure 5.2: (a) An example of a set of flags based on the triangular lattice: $F = \{f_0, \ldots, f_{23}\}$ from $D+1 = 3$ levels: $C_0 = \{c_0^0, \ldots, c_0^3\}$ in red, $C_1 = \{c_1^0, \ldots, c_1^3\}$ in blue, $C_2 = \{c_2^0, \ldots, c_2^3\}$ in green. The figure wraps around itself according to the arrows so that there are no boundaries, and the surface obtained is a torus. (b) Schematic representation of the flags of the square lattice and the corresponding pins. Each triple of incident vertex (green pin), edge (red pin) and face (blue pin) is a flag. They are symbolized in the picture as actual flags put closest to the elements in the triple (vertex, edge, face) they stand for. The corresponding color code is the well known 4.8.8 color code: there are eight flags around each vertex, four around each edge and eight around each face.

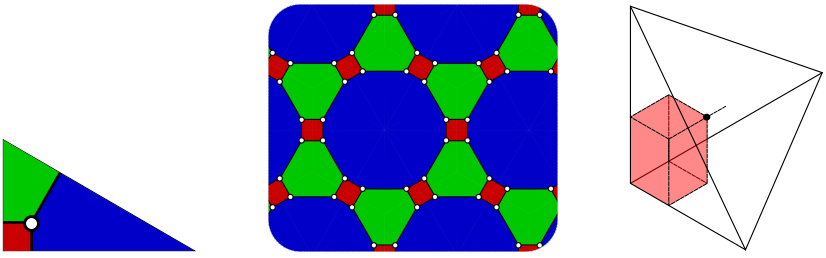


Figure 5.3: (a) The fundamental triangle of the Wythoff construction. A vertex (white cirlce) is placed in the middle and three edges are drawn to the boundary. The three regions are colored red, green and blue. (b) Reflecting the fundamental triangle along its sides creates a three colored lattice. In this case the 4.6.12 lattice. Note that we can obtain the red/blue/green shrunk lattice by moving the vertex into the red/blue/green corner, see Sec. 5.4.2. (c) The Wythoff construction can be generalized to higher dimensions as well. This example shows the vertex put into the middle of a 3D simplex, a tetrahedron, and one of the four corner cells colored in red (see main text for more information).

is a sub-cell of the other. An element of this relation, i.e. a $(D+1)$-tuple containing a 0-cell (a vertex), a 1-cell (an edge), etc…up to a D-cell, is called a flag. See for example Fig. 5.2b for a representation of the flags of the square lattice. The flag relation obtained this way is the same as the one after going through the inflating procedure and it is a pin code relation.

A similar way to construct $(D+1)$-colorable tessellations in D dimensions directly is

to use the Wythoff construction. The construction is quite general, but for simplicity, let us start on the 2D Euclidean plane. Consider a right-angled triangle and draw a point into its interior. From this point we draw three lines, each intersecting a boundary edge at a right angle (see Fig. 5.3a). This creates three regions in the triangle which we assign three different colors. We can now reflect the triangle along its boundary edges. The internal points of the original and the reflected triangles become the vertices of a uniform tiling. The faces of the tiling are colored by the three colors and by construction no two faces of the same color are adjacent (see Fig. 5.3b). If the angles of the triangle are $2\pi/r$, $2\pi/s$ and $2\pi/l$ then the result will be a $r.s.l$-tiling, meaning that the three faces around a vertex will have r, s and l number of sides.

This idea readily generalizes to higher dimensions by placing a vertex into a D-dimensional simplex and drawing lines to the mid-point of the $D-1$-dimensional faces of the simplex (see Fig. 5.3c for the case $D=3$). The faces of the simplex are simplices themselves, so this process can be iterated until $D=2$. Reflecting along the faces of the D-simplex gives rise to a uniform tiling of the D-dimensional space with D-cells being colored by $D+1$ colors and no two cells of the same color sharing a $D-1$-dimensional face. The number of vertices of the D-cells is then determined by the orbit of the reflections along all but one of the sides.

The color codes from regular tilings can be obtained this way, for example, in 2D Euclidean space, the hexagonal, 4.8.8. or 4.6.12. color codes or more generally both Euclidean and hyperbolic tilings in any dimension.

The classification of what initial simplex can be used, also called fundamental domain, in order to tile the spherical, Euclidean or hyperbolic spaces amounts to studying the groups of symmetries of the tilings. These groups of symmetries are also called Coxeter groups.

COXETER GROUP APPROACH

In this section we present how to obtain pin code relations directly from Coxeter groups [23, 24], or more generally from finite groups which are generated by elements with even order. A Coxeter group is a finitely presented group with reflections as generators, denoted as

$$G = \langle a_0, \ldots, a_D | a_0^2 = \cdots = a_D^2 = \left(a_i a_j\right)^{k_{ij}} = r_k = \cdots = 1 \rangle,$$

where the k_{ij} define the relations between the generators and r_k are some optional additional relations between generators and 1 is the trivial element. Define the subgroups, H_j for $j \in \{0, \ldots, D\}$, as

$$H_j = \left\langle \{a_0, \ldots, a_D\} \setminus \{a_j\} \right\rangle.$$

Define the levels, C_j, as the sets of left cosets for each H_j, i.e.

$$C_j = \left\{ g H_j \mid g \in G \right\}.$$

The cosets of a subgroup always form a partition of the full group. So for every $j \in \{0, \ldots, D\}$, a group element $g \in G$ uniquely defines a coset $c_j \in C_j$ such that $g \in c_j$. Hence, each group element defines a $(D+1)$-tuple of cosets, $(c_0, \ldots, c_D) \in C_0 \times \cdots \times C_D$. Taking the set of all such tuples defines a $(D+1)$-ary relation on the cosets, $F \subset C_0 \times \cdots \times C_D$,

which is a pin code relation. The fact that this F is a pin code relation can be verified by the following argument. A k-pinned set here corresponds to the intersection of k different cosets with respect to k different subgroups H_j. It always holds that the intersection of several cosets is either empty or is a coset with respect to the intersection of the subgroups of the original cosets. Hence non-empty k-pinned sets are cosets with respect to a subgroup, $H_{j_1,\dots,j_k}$,

$$H_{j_1,\dots,j_k} = \bigcap_{i=1}^{k} H_{j_i}.$$

Each subgroup H_{j_i} is generated by all generators of G except one, this means that $H_{j_1,\dots,j_k}$ contains at least a subgroup generated by $D-k+1$ of the generators,

$$H_{j_1,\dots,j_k} \supseteq \langle \{a_0,\dots,a_D\} \setminus \{a_{j_1},\dots,a_{j_k}\} \rangle. \tag{5.2}$$

In particular the D-pinned sets are cosets with respect to a subgroup which contains $\langle a_j \rangle$ for some j, which has even order since a_j is a reflection. Therefore D-pinned sets have even order. In well behaved cases the containment in Eq. (5.2) will actually be an equality but this is not guaranteed depending on the relations between generators.

In the case where the Coxeter group describes the symmetries of a tiling this is equivalent to the Wythoff construction described above. But one also obtains more general pin codes when considering Coxeter groups not defining tilings or more general finite groups with generators of even order.

In Sec. 5.5.1 we explore in more details the construction of pin codes from 3D hyperbolic Coxeter groups and give some explicit examples.

CHAIN COMPLEX APPROACH

Another way of obtaining a pin code relation is from $\mathbb{F}_2$ chain complexes of length $D+1$. These algebraic objects are composed of $(D+1)$ vector spaces over $\mathbb{F}_2$, say $\mathscr{C}_0,\dots,\mathscr{C}_D$, together with D linear maps called boundary maps, $\partial_j : \mathscr{C}_j \to \mathscr{C}_{j-1}$, which are such that

$$\forall\, j \in \{0,\dots,D-1\}, \partial_j \circ \partial_{j+1} = 0. \tag{5.3}$$

For example the tiling of a D-manifold can be seen as a chain complex, taking the j-cells as a basis for the $\mathscr{C}_j$ vector space and the natural boundary map. We have shown how to get a pin code relation from such a tiling by taking its flags, but it can as well be obtained from any $\mathbb{F}_2$ chain complex.

The construction works as follows: choose a basis set C_j for each vector space $\mathscr{C}_j$. The C_j basis sets are the levels and the basis elements the pins. Then use the boundary map, ∂, to define binary relations, $R_{j,j+1} \subset C_j \times C_{j+1}$, where $(c_j, c_{j+1}) \in R_{j,j+1}$ if c_j appears in the decomposition of $\partial(c_{j+1})$ over the basis set C_j. Then the relation $F \subset C_0 \times \cdots \times C_D$ is defined as follows

$$F = \left\{ (c_0,\dots,c_D) \mid \forall\, j,\ (c_j, c_{j+1}) \in R_{j,j+1} \right\}.$$

The relation F obtained like this is almost a pin code relation. All the pinned sets of type $t = \{0,\dots,D\} \setminus \{j\}$ with $0 < j < D$ have even cardinality since their size is given by

the number of paths between the pin c_{j+1} and the pin c_{j-1} which has to be even by the property of the boundary map ∂ given in Eq. (5.3). For pinned sets of type $t = \{1,\dots,D\}$ or $t = \{0,\dots,D-1\}$ it is not generally the case that they have even cardinality. However, this can be easily fixed by adding at most two pins: the idea is then to add one rank-0 pin, b_0, in the level C_0 and add all pairs $(b_0, c^\star)$ such that

$$\left|\{c_0 \mid (c_0, c^\star) \in R_{0,1}\}\right| = 1 \pmod 2,$$

to the new relation $R_{0,1}$. Then do the same for the level D, adding b_D in C_D. After this modification the resulting flag relation F is a pin code relation.

Note that this way of obtaining a quantum code from any $\mathbb{F}_2$ chain complex is fundamentally different from the usual homological code construction. In the homological code construction one chooses one of the levels, say C_j, and identifies its elements with qubits. Then the Z-stabilizer generators are given by the boundary of the elements in C_{j+1} and the X-stabilizer generators by the coboundary of the elements in C_{j-1}. These are different from the flags and pinned sets used to define a pin code.

In Sec. 5.5.2 we give some explicit pin codes constructed from chain complexes.

5

5.2.5. REMARKS

While some flag relations F obtained from Coxeter groups can be equivalently viewed as coming from some $\mathbb{F}_2$ chain complex, the converse does not necessarily hold. Indeed not every multi-ary relation can be decomposed into a sequence of binary relations, the hexagonal lattice depicted in Fig. 5.2a is an example of such a relation which cannot be decomposed this way. The other way around, not all flag relations obtained from an $\mathbb{F}_2$ chain complex can be seen as coming from a Coxeter group as in general they would lack the regular structure required.

Depending on the pin code relation, F, it can happen that some pinned sets can in fact be safely split when defining the stabilizers. That is to say, one can separate them into several disjoint sets of flags defining each an independent stabilizer still commuting with the rest of the stabilizers. For example this is the case for Coxeter groups for which (5.2) is strict, i.e.

$$\begin{aligned}\langle \{a_{i_1},\dots,a_{i_s}\}\rangle \cap \langle \{a_{j_1},\dots,a_{j_t}\}\rangle \supsetneq \\ \langle \{a_{i_1},\dots,a_{i_s}\} \cap \{a_{j_1},\dots,a_{j_t}\}\rangle.\end{aligned} \tag{5.4}$$

In this case the cosets with respect to the first group can be further split into cosets with respect to the second one without harming the commutation relations. Groups generated by reflections for which (5.2) is always an equality are called C-groups [25]. If the stabilizers are still defined as whole pinned sets, in cases where they could be split, then these smaller sets of qubits would be logical operators which would be detrimental to the overall distance of the code.

REED-MULLER CODE-WORDS AS PINNED SETS

Reed-Muller codes can be simply expressed using specific pin code relations. One can define the Reed-Muller code, $\mathscr{RM}(r,m) \subset \mathbb{F}_2^m$, as follows. Define

$$k = \sum_{j=0}^{r} \binom{m}{j},$$

k gives the number of multilinear monomials of degree at most r over m variables. So given k coefficients, $c \in \mathbb{F}_2^k$, we can define a binary polynomial over m variables, $p_c \in \mathbb{F}_2[x_1, \ldots, x_m]$, as

$$p_c = \sum_{\substack{S \subset \{1,\ldots,m\} \\ |S| \leq r}} c_S \prod_{j \in S} x_j.$$

Such a polynomial, when evaluated, gives binary numbers and that for each 2^m possible assignments of the variables. So we associate to any polynomial a binary vector of length 2^m. Then the code $\mathscr{RM}(r,m)$ is generated by all the binary vectors associated with the polynomials of degree at most r over m variables:

$$\mathscr{RM}(r,m) = \left\{ \left(p_c(x)\right)_{x \in \mathbb{F}_2^m} : c \in \mathbb{F}_2^k \right\}.$$

We now show that $\mathscr{RM}(r,m)$ is generated by the pinned sets with r pins of a certain pin code relation F. Consider m levels, each containing two pins,

$$\forall\, j \in \{0,\ldots,m-1\},\ C_j = \{0,1\},$$

and the complete m-ary relation, $F = C_0 \times \cdots \times C_{m-1}$, see also the left of figure 5.10. Consider now a r-pinned set defined, with type $t = \{j_1, \ldots, j_r\}$, and pins $b = (b_{j_1}, \ldots, b_{j_r})$. One can check that a flag, $f \in F = \mathbb{F}_2^m$, belongs to $P_t(b)$ if and only if the following degree r polynomial, $p_{t,b}$, evaluates to 1,

$$p_{t,b} = \prod_{\substack{j \in t \\ b_j = 1}} x_j \prod_{\substack{k \in t \\ b_k = 0}} (1 - x_k).$$

So the pinned sets with r pins generate the following code,

$$\mathscr{P}(r,m) = \left\langle (p_{t,b}(x))_{x \in \mathbb{F}_2^m} : t \subset \{0,\ldots,m-1\}, |t| = r, b \in \mathbb{F}_2^m \right\rangle.$$

Then we just have to check that these generate all polynomials of degree at most r. By definition they generate polynomials constituted of a product of r elements being either x_j or $(1 - x_j)$. Let us suppose, for some $\ell \leq r$, they can generate all such product with only ℓ terms. Then we can contruct all product with only $\ell - 1$ terms, q, as follows

$$q = q \cdot (1 - x_j) + q \cdot x_j,$$

where x_j is a variable that does not appear in q. It follows by induction that they generate all degree at most r polynomials and so

$$\mathscr{RM}(r,m) = \mathscr{P}(r,m).$$

Another way to see this is to use the decomposition property of pinned sets (Proposition 2) and generate the lower-degree monomials directly with pinned sets with less pins and decompose these pinned sets into disjoint unions of r-pinned sets.

5.3. TRANSVERSAL GATES AND MAGIC STATE DISTILLATION

In this section we discuss what structure is desirable for CSS codes to admit transversal phase gates of different levels of the Clifford hierarchy. The presentation here is close in spirit to that of [26, 27]. It is included here to set terminology and for self-containment purposes.

Given ℓ binary row vectors, $\boldsymbol{v}^1, \ldots, \boldsymbol{v}^\ell \in \mathbb{F}_2^n$, we denote their element-wise product as $\boldsymbol{v}^1 \wedge \cdots \wedge \boldsymbol{v}^\ell$. Its j^{th} entry is given by

$$\left[\bigwedge_{m=1}^{\ell} \boldsymbol{v}^m\right]_j = \left[\boldsymbol{v}^1 \wedge \boldsymbol{v}^2 \wedge \cdots \wedge \boldsymbol{v}^\ell\right]_j = v_j^1 v_j^2 \cdots v_j^\ell.$$

The Hamming weight of a binary vector $\boldsymbol{v}$ is denoted as $|\boldsymbol{v}|$, it is given by the sum of its entries. We also define the notions of multi-even and multi-orthogonal spaces:

Definition 5 (Multi-even space)**.** *Given an integer, $\ell \in \mathbb{N}$, a subspace $\mathscr{C} \subset \mathbb{F}_2^n$, is called ℓ-even if all vectors in $\mathscr{C}$ have Hamming weight divisible by 2^ℓ:*

$$\forall\, \boldsymbol{v} \in \mathscr{C},\ |\boldsymbol{v}| = 0 \pmod{2^\ell}.$$

An equivalent characterization is that for any integer $s \in \{1, \ldots, \ell\}$ and any s-tuple of vectors, $(\boldsymbol{v}^1, \ldots, \boldsymbol{v}^s) \in \mathscr{C}^s$, it holds that

$$\left|\boldsymbol{v}^1 \wedge \cdots \wedge \boldsymbol{v}^s\right| = 0 \pmod{2^{\ell-s+1}}.$$

Definition 6 (Multi-orthogonal space)**.** *Given an integer, $\ell \in \mathbb{N}$, a subspace $\mathscr{C} \subset \mathbb{F}_2^n$, is called ℓ-orthogonal if for any ℓ-tuple of vectors, $\left(\boldsymbol{v}^1, \ldots, \boldsymbol{v}^\ell\right) \in \mathscr{C}^\ell$,*

$$\left|\boldsymbol{v}^1 \wedge \cdots \wedge \boldsymbol{v}^\ell\right| = 0 \pmod{2}.$$

Binary addition is denoted by $\oplus$, and the following identity can be used to convert binary addition to regular integer addition

$$\bigoplus_{m=1}^{r} \boldsymbol{w}^m = \sum_{s=1}^{r} (-2)^{s-1} \sum_{1 \le m_1 < \cdots < m_s \le r} \bigwedge_{i=1}^{s} \boldsymbol{w}^{m_i}. \tag{5.5}$$

Using this identity one can show the equivalence of the two characterizations of an ℓ-even space given in Def. 5, and that it is enough to verify the second characterization on a generating set of the space. Similarly it is enough to verify that a space is ℓ-orthogonal on a generating set.

The single-qubit phase gates are denoted as

$$R_\ell = \begin{pmatrix} 1 & 0 \\ 0 & \omega_\ell \end{pmatrix}, \qquad \omega_\ell = \mathrm{e}^{i\frac{2\pi}{2^\ell}}, \tag{5.6}$$

where ω_ℓ is the 2^ℓth root of unity. For instance $R_1 = Z$, $R_2 = S$ and $R_3 = T$ in the usual notations.

5.3.1. WEIGHTED POLYNOMIALS AND TRANSVERSAL GATES

Given k qubits and some integer ℓ, we consider quantum gates, U_{F_ℓ}, acting diagonally on the computational basis, such that for $\boldsymbol{x} \in \mathbb{F}_2^k$,

$$U_{F_\ell} |\boldsymbol{x}\rangle = \omega_\ell^{F_\ell(\boldsymbol{x})} |\boldsymbol{x}\rangle ,$$

where F_ℓ is a so-called weighted polynomial of the form

$$F_\ell(\boldsymbol{x}) = \sum_{s=1}^{\ell} 2^{s-1} \sum_{m_1<\cdots<m_s} \alpha_{m_1\ldots m_s} \cdot x_{m_1} \cdots x_{m_s}, \tag{5.7}$$

with coefficients $\alpha_{m_1\ldots m_s}$ in $\mathbb{F}_{2^\ell}$. Any such gate U_{F_ℓ} belongs to the ℓth level of the Clifford hierarchy [28]. Examples, and generating sets for $\ell = 3$, are given in Table 5.1.

Table 5.1: Weighted polynomials corresponding to some well known gates, for $\ell = 3$, arranged according number of qubits involved and level of the Clifford hierarchy.

$U_{F_3} \leftrightarrow F_3(\boldsymbol{x})$	1st level	2nd level	3rd level
1 qubit	$Z \leftrightarrow 4x_1$	$S \leftrightarrow 2x_1$	$T \leftrightarrow x_1$
2 qubits	-	$CZ \leftrightarrow 4x_1x_2$	$CS \leftrightarrow 2x_1x_2$
3 qubits	-	-	$CCZ \leftrightarrow 4x_1x_2x_3$

The goal is to implement such a gate on the logical level of a quantum error correcting code by the transversal application of R_ℓ gates, defined in Eq. (5.6). Given an $[[n,k,d]]$ quantum CSS code, define G as the $r \times n$ matrix whose rows describe a generating set of the X-stabilizers of the code, and define L as the $k \times n$ matrix whose rows describe a basis for the X-logical operators. The code state in this basis corresponding to $\boldsymbol{x} \in \mathbb{F}_2^k$ can then be expressed as

$$|\overline{\boldsymbol{x}}\rangle = \frac{1}{\sqrt{2^r}} \sum_{\boldsymbol{y}\in\mathbb{F}_2^r} |\boldsymbol{x}L \oplus \boldsymbol{y}G\rangle .$$

Applying transversally the gate R_ℓ on this code state, $|\overline{\boldsymbol{x}}\rangle$, yields

$$R_\ell^{\otimes n} |\overline{\boldsymbol{x}}\rangle = \frac{1}{\sqrt{2^r}} \sum_{\boldsymbol{y}\in\mathbb{F}_2^r} \omega_\ell^{|\boldsymbol{x}L\oplus\boldsymbol{y}G|} |\boldsymbol{x}L \oplus \boldsymbol{y}G\rangle . \tag{5.8}$$

Using Eq. (5.5), the power of ω_ℓ above can be rewritten in three parts as

$$|\boldsymbol{x}L \oplus \boldsymbol{y}G| = F_\ell(\boldsymbol{x}) + F'_\ell(\boldsymbol{y}) + F''_\ell(\boldsymbol{x},\boldsymbol{y}), \tag{5.9}$$

where we defined

$$F_\ell(\boldsymbol{x}) = |\boldsymbol{x}L| \tag{5.10}$$
$$F'_\ell(\boldsymbol{y}) = |\boldsymbol{y}G| \tag{5.11}$$
$$F''_\ell(\boldsymbol{x},\boldsymbol{y}) = -2\,|\boldsymbol{x}L \wedge \boldsymbol{y}G| \tag{5.12}$$

Using again Eq. (5.5), one can express these three parts as weighted polynomials whose coefficients are given by the different overlap between X-logical operator generators, between X-stabilizer generators or between both,

$$F_\ell(\boldsymbol{x}) = \sum_{s=1}^{\ell} (-2)^{s-1} \sum_{1\le m_i\le k} \left| \bigwedge_{i=1}^{s} \boldsymbol{L}^{m_i} \right| \prod_{i=1}^{s} x_{m_i}, \tag{5.13}$$

$$F'_\ell(\boldsymbol{y}) = \sum_{t=1}^{\ell} (-2)^{t-1} \sum_{1\le n_j\le r} \left| \bigwedge_{j=1}^{t} \boldsymbol{G}^{n_j} \right| \prod_{j=1}^{t} y_{n_j}, \tag{5.14}$$

$$F''_\ell(\boldsymbol{x},\boldsymbol{y}) = \sum_{\substack{s+t=2\\ s\ge 1,\, t\ge 1}}^{\ell} (-2)^{s+t-1} \sum_{\substack{1\le m_i\le k\\ 1\le n_j\le r}} \left| \bigwedge_{i=1}^{s} \boldsymbol{L}^{m_i} \bigwedge_{j=1}^{t} \boldsymbol{G}^{n_j} \right| \prod_{i=1}^{s} x_{m_i} \prod_{j=1}^{t} y_{n_j}. \tag{5.15}$$

Provided that it is possible to cancel the action of $F'_\ell(\boldsymbol{y})$ and $F''_\ell(\boldsymbol{x},\boldsymbol{y})$ then the resulting operation would correspond to the gate U_{F_ℓ} on the logical qubits of the code. The following two properties of CSS codes are designed to get rid of these two unwanted parts.

Proposition 4 (Exact transversality)**.** *Let $\mathscr{C}$ be a CSS code, given an integer ℓ, the code $\mathscr{C}$ allows for the transversal application of R_ℓ if the following conditions hold:*

(i) The X-stabilizers form an ℓ-even space.

(ii) Element-wise products of a X-logical operator and a X-stabilizer always have Hamming weight divisible by $2^{\ell-1}$.

The gate performed at the logical level is then given by the weighted polynomial in Eq. (5.10).

Indeed the two conditions above exactly give

$$F'_\ell(\boldsymbol{y}) = F''_\ell(\boldsymbol{x},\boldsymbol{y}) = 0 \pmod{2^\ell},$$

which precisely enforce that the actions of $F'_\ell(\boldsymbol{y})$ and $F''_\ell(\boldsymbol{x},\boldsymbol{y})$ are trivial. We can also settle for a weaker condition under which the unwanted part is not trivial but belongs to the $(\ell-1)$th level of the Clifford hierarchy. We refer to this weaker condition as *quasi-transversality*.

Proposition 5 (Quasi-transversality)**.** *Let $\mathscr{C}$ be a $[[n,k,d]]$ CSS code, with $r\times n$ generating matrix, G, for its X-stabilizers and $k\times n$ generating matrix, L, for its X-logical operators. Given an integer ℓ, the code $\mathscr{C}$ allows for the transversal application of R_ℓ up to a $(\ell-1)$th-level Clifford correction if the following conditions hold:*

(i) The X-stabilizers form an ℓ-orthogonal space.

(ii) For any choice of $s\ge 1$ X-logical operators and $t\ge 1$ X-stabilizers with $s+t\le\ell$

$$\left| \bigwedge_{i=1}^{s} \boldsymbol{L}^{m_i} \bigwedge_{j=1}^{t} \boldsymbol{G}^{n_j} \right| = 0 \pmod 2.$$

The gate performed at the logical level after correction is then given by the weighted polynomial in Eq. (5.10).

One can check that under these conditions it follows that,

$$\omega_\ell^{F'_\ell(\boldsymbol{y})+F''_\ell(\boldsymbol{x},\boldsymbol{y})} = \omega_\ell^{2\tilde{F}_{\ell-1}(\boldsymbol{x},\boldsymbol{y})} = \omega_{\ell-1}^{\tilde{F}_{\ell-1}(\boldsymbol{x},\boldsymbol{y})}, \tag{5.16}$$

where $\tilde{F}_{\ell-1}$ is a properly weighted polynomial which defines a $(\ell-1)$th-level Clifford correction to be applied. Indeed (i) enforces that all coefficients $\left|\bigwedge_{j=1}^{t} \boldsymbol{G}^{n_j}\right|$ in Eq. (5.14) are divisible by 2 and (ii) that all coefficients $\left|\bigwedge_{i=1}^{s} \boldsymbol{L}^{m_i} \bigwedge_{j=1}^{t} \boldsymbol{G}^{n_j}\right|$ in Eq. (5.15) also are divisible by 2. Hence we can pull out a factor 2 in front of everything while keeping the correct prefactor in front of each monomial. The exact correction is given by the conjugation of $U_{\tilde{F}_{\ell-1}}$ by a decoding circuit for $\mathscr{C}$, see [26]. Note that we can also define intermediate conditions so that the correction belongs to the $(\ell-q)$th level of the Clifford hierarchy for some $1 \le q \le \ell$. Here we have just defined the two extreme ones, for which the correction belongs either to the 0th level or the $(\ell-1)$th level of the Clifford hierarchy.

5.4. PROPERTIES OF QUANTUM PIN CODES

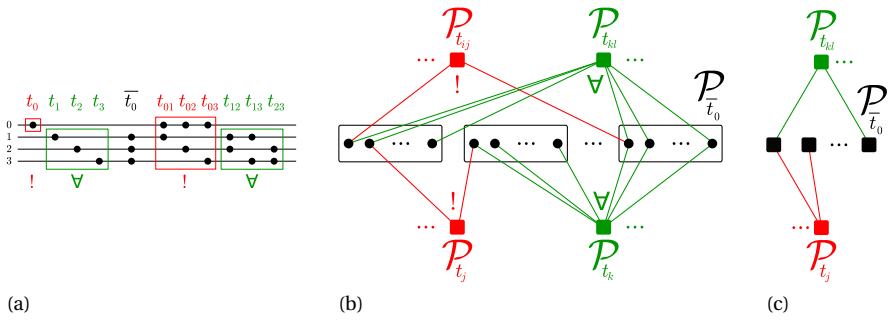


Figure 5.4: (a) Schematic representation of the different types for the X- and Z-stabilizers in the case of $D=3$. The different types are classified according to their possible intersection with the complementary type to t_0. For some, the intersection, if not empty, is a unique flag, they are labeled by the symbol "!". For the others, the intersection, if not empty, is the full pinned set of type $\overline{t_0}$, they are labeled by the symbol "∀". (b) The chain complex corresponding to the pin code, highlighting the intersections between the different sorts of X- and Z-stabilizers and pinned sets of type $\overline{t_0}$. (c) The t_0-shrunk chain complex derived from the pin code (see main text).

In this section we examine the properties of pin codes. Since their definition is fairly general, their properties depend on the precise choice of pin code relations F. We stay as general as possible and state precisely when the pin code relations need to be restricted.

5.4.1. CODE PARAMETERS AND BASIC PROPERTIES

First we investigate the LDPC (Low Density Parity Check) property. A code family is LDPC if it has stabilizer checks of constant weight and each of its qubits is acted upon by a constant number of checks. For pin codes, both properties depend on the relation F,

but it is fairly easy to construct LDPC families. For instance, pin codes based on Coxeter groups with fixed relations between generators and one growing compactifying relation are LDPC, see Sec. 5.5.1. As another example, pin codes from chain complexes with fixed length $D+1$, sparse boundary map and growing dimension of the levels are LDPC as well.

Let us examine a simple example: choose some $D \in \mathbb{N}$, a set, C, of size $2m$ for some $m \in \mathbb{N}$ and the complete relation on $D+1$ copies of C: $F = C^{D+1}$. One can easily verify that the relation F is a pin code relation as C has even cardinality. The number of flags is $n_q = |F| = (2m)^{D+1}$ and the number of x- and z-pinned sets are $n_x = \binom{D+1}{x} \times (2m)^x$ and $n_z = \binom{D+1}{z} \times (2m)^z$. If one considers growing m, then the code would not be LDPC, but more strikingly the ratio of number of stabilizer checks to number of qubits would go to zero. This illustrates that for a fixed D the complete relation is a poor choice, leading to very high rate and very low distance. To get interesting codes, one either needs to vary D, or find some other relations with a number of flags growing significantly slower than the complete relation.

Concerning logical operators, we first note that they have even weight.

Proposition 6 (Logical operators have even weight)**.** *Let F be a pin code relation on $D+1$ levels and let $\mathscr{C}$ be the associated (x,z)-pin code for $(x,z) \in \{1,\dots,D\}^2$ with $x+z \leq D$. Then the X- and Z-logical operators of $\mathscr{C}$ have even weight.*

Proof. Let the set $L \subset F$ represent a X-logical operator, and let t be a type of size z. Consider the set, S, of collections of pins given by the projection of type t of the set L,

$$S = \Pi_t(L).$$

For every $s \in S$, the pinned set $P_t(s)$ corresponds to a Z-stabilizer and therefore has an even intersection with L. Pinned sets of the same type but defined by two different collections of pins are necessarily disjoint. Hence, every element in L appears in exactly one of the pinned sets $P_t(s)$ for some $s \in S$ and so the cardinal of L is even. The proof for Z-logical operators is the same. □

One can also prove the following general lower bound on the distance of pin codes.

Proposition 7 (Distance at least 4)**.** *Let F be a pin code relation on $D+1$ levels and let $\mathscr{C}$ be the associated (x,z)-pin code for $(x,z) \in \{1,\dots,D\}^2$ with $x+z \leq D$. Then the distance of $\mathscr{C}$ is at least 4.*

Proof. Using the fact that the distance has to be even from Prop. 6, we just need to verify that it cannot be 2. Let f_1 and f_2 be any two flags in F, they must differ on at least one level $j \in \{0,\dots,D\}$. Pick a type t_x of size x containing j and define the collection of pins $s_x = \Pi_{t_x}(f_1)$ of type t_x. Then the pinned set $P_{t_x}(s_x)$, defining a X-stabilizer, contains f_1 but not f_2, hence $\{f_1, f_2\}$ cannot define a Z-logical operator. Similarly by defining the type t_z of size z containing j and the collection of pins $s_z = \Pi_{t_z}(f_1)$, the pinned set $P_{t_z}(s_z)$ is a witness that $\{f_1, f_2\}$ cannot be a X-logical operator. □

We conjecture that better lower bounds can be obtained when taking into account the exact size of the type for the stabilizers as larger types can differentiate more flags. The proof above is optimal only for $x = z = 1$. In order to get odd weight logical operators, one has to introduce free pins, see Sec. 5.4.5. Note that in the presence of free pins, the proof above does not hold anymore.

5.4.2. COLORED LOGICALS AND UNFOLDING

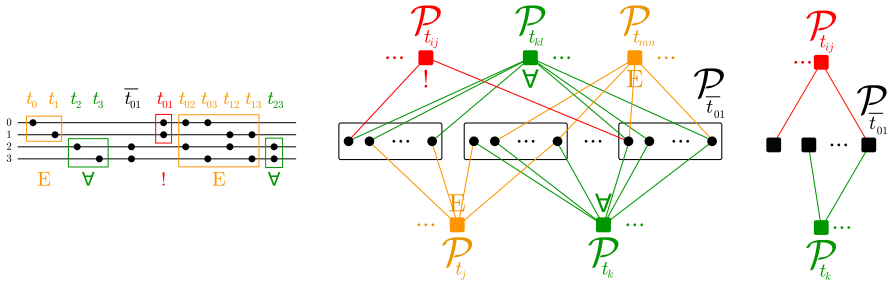


Figure 5.5: (a) Schematic representation of the different types classified according to their possible intersection with the complementary type to t_{01}. The ones with unique intersection are labeled with "!", the ones with even intersection with "E" and the one with full containement with "∀". (b) The chain complex corresponding to the pin code, highlighting the intersections between the different sorts of X- and Z-stabilizers and pinned sets of type $\overline{t_{01}}$. (c) The t_{01}-shrunk (co)chain complex derived from the pin code (see main text).

The structure of the logical operators of color codes is understood as colored string-nets or membrane nets [21] and this structure is directly linked to an unfolding procedure existing for color codes [29, 30]. This structure remains for all pin codes, so we recast it here as a first step to gain more insight in the structure of the logical operators.

The general idea is to group qubits into sets with even overlap with almost all except for one sort of stabilizer. The chosen sort of stabilizer which has odd overlap with the groups of qubits corresponds to the stabilizers defined by pinned sets of a given type. Logical operators built out of these groups of qubits then only depend on the structure of the one type of stabilizer selected. Repeating this for different choices of type of stabilizer fully covers all logical operators in the case of color codes.

Consider a pin code relation, $F \subset C_0 \times \cdots \times C_D$, and the associated (x,z)-pin code. Define the complement of a type, t, denoted as $\overline{t}$:

$$\overline{t} = \{0,\ldots,D\} \setminus t.$$

The intersection between a pinned set of type t and a pinned set of type $\overline{t}$ is either empty or it contains exactly one flag. Furthermore, for any other type with the same number of pins as t, the corresponding pinned sets have necessarily even overlap with pinned sets of type $\overline{t}$, see Figs. 5.4a and 5.5a for visual representations of this. This means that grouping flags according to pinned set of the complementary type $\overline{t}$ can single out logical operators only having to ensure commutation with pinned sets of type t. For our code, X-stabilizers are generated by x-pin sets, which come in $\binom{D+1}{x}$ different types. Take one such type, t_x, and group the qubits according to pinned sets of type $\overline{t_x}$. Now the Z-stabilizers are generated by z-pinned sets, which come in $\binom{D+1}{z}$ different types. Some of these types, we denote them as $t_z^{\text{inc.}}$, are fully included in $\overline{t}$, which means that pinned sets of such type fully contain any group of qubits they intersect. The other types only partially intersect with the groups of qubits. The situation is schematized in Fig. 5.4b for $D = 3$, $x = 1$ and $z = 2$. From these considerations, one can construct a chain complex

for which the homology gives candidate Z-logical operators. Take the pinned sets of type t_x, for the level 0, the pinned sets of type $\overline{t_x}$ for the level 1, and the pinned sets of types $t_z^{\text{inc.}\cdot}$ for level 2 and the boundary map is given by the overlaps of these sets. This is represented in Fig. 5.4c, we call it the t_x-shrunk chain complex. Then one can check that an element of the homology of this chain complex can be lifted to a potential Z-logical operator for the pin code. Indeed it would commute with all the X-stabilizer, by homology for the stabilizers of type t_x and by construction for the other X-stabilizers. It would also not be simply generated by Z-stabilizers of type $t_z^{\text{inc.}\cdot}$ by homology, and one would have to check for the other Z types. So it is a valid (potentially trivial) Z-logical operator.

The same procedure can be done for each of the X types. Symmetrically, the same can be done for the X-logical with the Z types, and this is represented in Fig. 5.5 in the case $D = 3$, $x = 1$ and $z = 2$.

Given a type t, the chain complexes constructed like this are called t-shrunk lattices in the case of color codes [21]. For color codes obtained from the Wythoff construction described in Sec. 5.2.4, the construction of the t-shrunk lattice is fairly direct. First move the vertex from the middle of the fundamental simplex to the corner corresponding to the first rank in the type t, then focus on the opposite face: a simplex of dimension one less which now looks exactly like the beginning of the procedure but in a lower dimension. Recursively exhaust all the ranks of t in this way by each time adding a vertex in the middle of the current simplex and moving it to the corresponding corner.

These shrunk lattices are the basis for the unfolding procedure proved for color codes in all dimensions in [29]. This procedure establishes a local unitary equivalence between a color code and the union of the homological codes on the shrunk lattices corresponding to all the different types for X-stabilizers except one. The local unitary acts separately on groups of qubits defined by the X-stabilizer generators of the type that is not used to produce one of the shrunk lattices. The proof of the existence of the local unitary relies on the analysis of the so-called overlap groups of stabilizers restricted to the support of the X-stabilizer generators aforementioned and the corresponding groups of qubits in the shrunk lattices. The global structure is still present for general pin codes, but for the proof to hold it suffice to require that the linear dependency between the generators within the overlap groups in the pin code is such that the number of independent generators agrees with the number of independent generators in the corresponding shrunk lattices as an additional assumption.

These shrunk lattices are also the basis for some color code decoders [31–34] but these decoders rely on a lifting procedure from the shrunk lattices to the color code lattice which seems intrinsically geometric as it consists in finding a surface filling inside a boundary. So it is at this point unclear how to leverage this structure in order to decode general pin codes.

5.4.3. GAUGE PIN CODES

In this section we define gauge pin codes from a pin code relation. Gauge pin codes can also be viewed as a generalization of gauge color codes [20].

A gauge code, or subsystem code, is a code defined by a so-called gauge group instead of a stabilizer group [11, 35]. For stabilizer codes, the code states are eponymously

stabilized by the stabilizer group which is an Abelian subgroup of the group of Pauli operators. For gauge codes, the gauge group is not Abelian and hence all gauge operators cannot share a common +1-eigenspace. In this case the code states are stabilized by the center of the gauge group. Gauge operators not in the center of the gauge group commute with its center. Hence they qualify as logical operators in the case of a stabilizer code but they are not used to encode information in the case of a gauge code.

Take a pin code relation F and two positive integers x and z such that $x+z<D$. The associated pin code has its X-stabilizer generators defined by all the x-pinned sets and its Z-stabilizers generators defined by all the z-pinned sets. Since the relation F is a pin code relation, by Prop. 3 any $(D-x)$-pinned set has an even intersection with any x-pinned set. So all the $(D-x)$-pinned sets correspond to some Z-logical operators. On top of that, they generate all the Z-stabilizers. Indeed using Prop. 2 and the fact that $D-x>z$ one shows that the z-pinned sets decompose into disjoint $(D-x)$-pinned sets. As such $(D-x)$-pinned set define naturally Z-gauge operators which can be measured individually and whose outcomes can be recombined to reconstruct the value of the Z-stabilizers defined by z-pinned sets. Symmetrically, the same happens for $(D-z)$-pinned sets which have even overlap with z-pinned sets and generate x-pinned sets and therefore can be viewed as X-gauge operators.

In conclusion, given a pin code relation F and two natural integers x and z such that $x+z<D$; one defines the corresponding gauge pin code with X-gauge operators defined by the $(D-z)$-pinned sets and Z-gauge operators by the $(D-x)$-pinned sets. One can check that these operators do not all commute since $(D-x)+(D-z)>D$. The center of this gauge group, i.e. the stabilizer group, is defined by the x-pinned sets as X-stabilizer generators and z-pinned sets as Z-stabilizer generators. Note that it is not guaranteed that the number of logical qubits in a (x,z)-gauge pin code is the same as the number of logical qubits in the $(x,D-x)$-pin code obtained from the same relation F.

The error correction procedure for a gauge code with only fully X-type or fully Z-type gauge operators is conveniently performed in two parts. In one part, one measures the X-gauge operators, reconstructs the syndrome for the X-stabilizers and uses it to correct Z-errors. In the other part, one measures the Z-gauge operators, reconstructs the syndrome for the Z-stabilizers and uses it to correct X-errors.

The advantages of this procedure in the case of gauge pin codes are two-fold. First, the weight of the gauge generators, i.e. the number of qubits involved in each generator, is reduced compared to the weight of the stabilizer generators making their measurement easier and less error prone. Second, the record of gauge operator measurements contains the information of the stabilizer measurements with redundancy. To understand this redundancy consider a x-pinned set and define $k=(D-z)-x$. This is the number of additional levels to pin in order to decompose the x-pinned set into $(D-z)$-pinned sets. There are $\binom{D+1-x}{k}$ different ways to choose these additional levels to pin and therefore that many different ways to reconstruct the x-pinned set. This redundancy permits a more robust syndrome extraction procedure which can even become in some cases single-shot, meaning that the syndrome measurements do not have to be repeated to reliably decode [36].

5.4.4. TRANSVERSALITY

Here we examine pin codes with regards to the transversality and quasi-transversality of Prop. 4 and Prop. 5. Nicely, x-pinned sets always have some multi-orthogonality property.

Proposition 8 (Multi-orthogonality of pinned sets)**.** *Let F be a $(D+1)$-ary pin code relation. For any $x \in \{1,\dots,D\}$, the x-pinned sets seen as binary vectors in $\mathbb{F}_2^F$ generate a ℓ-orthogonal space with $\ell = \lfloor D/x \rfloor$.*

Proof. Given $x \in \{1,\dots,D\}$, by Prop. 1, the intersection of $\lfloor D/x \rfloor$ (or less) x-pinned sets is either empty or a pinned set with at most D pins, hence it has even weight for a pin code relation. □

Interestingly it is also not too difficult to find pin code relations for which the 1-pin sets are D-even. For example, using a chain complex whose boundary map have even row and column weights and is regular enough will typically suffice.

One could also hope for the second part of Proposition 5 to always hold. Unfortunately it holds only partially in general.

Proposition 9 (X-logical intersection with X-stabilizers)**.** *Let F be a $(D+1)$-ary pin code relation, and consider the associated (x,z)-pin code for $x \in \{1,\dots,D\}$ and $z = D-x$. Then for any one X-logical operator, $\boldsymbol{L}$, and k X-stabilizer generators, $\boldsymbol{G}^j$, with $k \leq \lfloor D/x \rfloor - 1$,*

$$\left|\boldsymbol{L} \wedge \boldsymbol{G}^1 \wedge \cdots \wedge \boldsymbol{G}^k\right| = 0 \pmod 2.$$

Proof. Indeed, using Prop. 1, the overlap between $\lfloor D/x \rfloor - 1$ (or less) different x-pinned sets is either empty or a pinned set with at most $D - x = z$ pins. Hence by Prop. 2, it can be decomposed into z-pinned sets, i.e. Z-stabilizers which have even overlap with X-logicals by definition. □

Overlaps involving more than one X-logical operator do not have such guarantees in general. Focusing on the case $\ell = 3$, given the two propositions above the only problematic conditions are the ones of type

$$\left|\boldsymbol{L}^j \wedge \boldsymbol{L}^k \wedge \boldsymbol{G}^\ell\right| = 0 \pmod 2. \tag{5.17}$$

In order for these terms to hold, one has to have that the intersection of two X-logical operators is always a Z-logical operator. This is the case for example for Euclidean color codes but it is not the case for every pin code. This means that we have to find subfamilies of pin codes in order to get transversal phase gates.

5.4.5. BOUNDARIES AND FREE PINS

The geometrical notion of colored boundaries existing for color codes can also be generalized to pin codes. The way to do this is to introduce a specific type of pins which will be called *free pins*.

Consider the chain complex approach to building pin code relations presented in Sec. 5.2.4. In this construction, it is sometimes necessary to add a rank-0 pin b_0 (in the

level C_0) or a rank-D pin b_D (in the level C_D) in order to ensure that the relation F is a pin code relation. The new pin b_0 is linked to all the rank-1 pins which previously were linked to an odd number of rank-0 pins. So even if the initial boundary relation is sparse, the number of connections to b_0 may be large. As such the 1-pinned set pinned by this new pin b_0 potentially contains a large number of flags. To keep the size of the 1-pinned sets under control it is then preferable to not allow to pin b_0 alone. That is why we then call b_0 a free pin. Any of the $D+1$ levels can contain free pins, the chain complex construction potentially puts one in C_0 and one in C_D. The rule for a larger collection of pins is that if it contains at least one non-free pin then it can define a valid pinned set, but if it is composed of only free pins then it is disregarded. Finally consider when a flag is only composed of free pins, in that case this flag will not enter any valid pinned sets. Hence such flags must also be discarded. This is summarized in the following definition.

Definition 7 (Pin code with free pins)**.** *Let F be a pin code relation defined on $D+1$ levels of pins. Let some of the pins be labeled as free pins. Let x and z be two natural integers such that $x+z \leq D$. The associated (x,z)-pin code is defined as follows: The elements of F containing at least one non-free pin are associated with qubits. All the x-pinned sets defined by a collection of pins containing at least one non-free pin are associated with X-stabilizer generators. All the z-pinned sets defined by a collection of pins containing at least one non-free pin are associated with Z-stabilizer generators.*

As examples we give a representation of Steane's $[[7,1,3]]$ code and the $[[4,2,2]]$ code as $(1,1)$-pin codes with free pins in Figure 5.6.

Figure 5.6: (Left) Representation of Steane's $[[7,1,3]]$ code as a $(1,1)$-pin code from a chain complex with free pins. There are three levels represented by the colors red green and blue. The free pins are represented between parentheses. There are 8 flags but one is composed only of free pins hence only 7 qubits. There are three non-free pins defining three 1-pinned sets for both X- and Z-stabilizers. (Right) Representation of the $[[4,2,2]]$ code as a $(1,1)$-pin code with free pins. There are four flags, and a single non-free pin defining the X- and Z-stabilizer both containing the four flags.

Introducing free pins can allow the code to escape the even-weight logical proposition, Prop. 6. This can be interesting as then different logical gates could be implemented which require the logical operators to have odd support. One idea to introduce free pins in every level could be to consider boundary map matrices which are almost sparse except for a small number of row or columns which could be dense. The basis element corresponding to these would then be labeled as free pins in the construction of the pin code relation.

Note that in the presence of free pins, the proof of Prop. 6 can only be reproduced when at least one level selected by the chosen type t does not contain any free pin. So as long as at least one level does not contain any free pin, it still holds that all logical

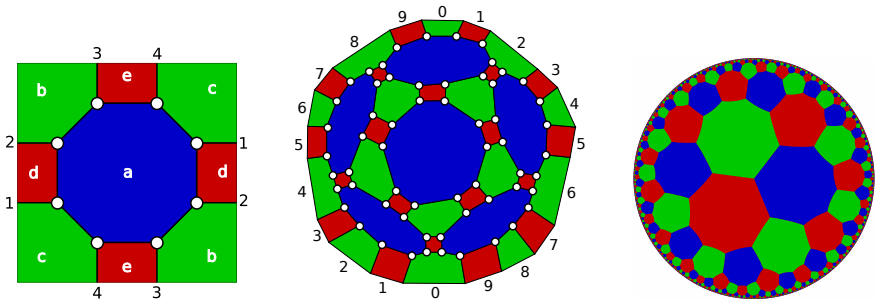


Figure 5.7: (a) [[8,2,2]] color code on the projective plane based on octahedral symmetry, which is topologically a sphere where opposite points are identified. (b) [[60,2,6]] color code on the projective plane based dodecahedral symmetry. (c) 3-colored hyperbolic tiling with edges and vertices forming a 3-valent graph.

operators have even weight. When all levels contain at least one free pin then the code may contain odd weight logical operators.

The notion of free pins carries over straightforwardly to gauge pin codes.

5.5. EXAMPLES AND APPLICATIONS

5.5.1. COXETER GROUPS, HYPERBOLIC COLOR CODES

In Section 5.2.4 we discussed the construction of pin codes from tilings and Coxeter groups. Well-known examples of such code families are color codes on Euclidean tilings such as the hexagonal tiling in 2D and the bitruncated cubic honeycomb in 3D. Using the Wythoff construction we can construct tilings which fulfill the right pin code condition and therefore have the correct colorability for defining a color code.

Besides the known Euclidean examples we can consider tilings of more exotic spaces. For the projective plane there exist two tilings based on the Wythoff construction: The first is based on the symmetry group of an octahedron. It is an [[8,2,2]]-code where the check generators correspond to one octagon, two red squares and two green squares, see Figure 5.7a. Note that this code does not quite fit the pin code definition because it contains distinct qubits which would be described by the same flag, for example (d, c, a) on edge 1. This degeneracy explains why it escapes Prop. 7. The second is based on the icosahedral symmetry group, which gives a a [[60,2,6]]-code with checks given by 6 decagons (blue), 10 hexagons (green) and 15 squares (red), see Figure 5.7b.

Color codes based on two-dimensional hyperbolic tilings were first considered in [37] where 3-colorability and 3-valence was postulated (see Figure 5.7c for an example). The Wythoff construction of Section 5.2.4 allows us to obtain color codes from arbitrary regular tilings of closed hyperbolic surfaces. To define a family of closed surfaces one needs to compactify the infinite lattice as explained in [38]. There are infinitely many regular tilings of 2D hyperbolic space. The lowest weight achievable with our construction is 4.8.10, meaning that checks are squares, octagons and dodecagons. The smallest code in this family is [[120,10,6]] based on a non-orientable hyperbolic surface (cf. Ta-

ble 3.1 in [39]). Another small example is a [[160,20,8]] code with stabilizer checks of weight 4 and 10 based on a 4.10.10 tiling of an orientable hyperbolic surface of genus 10.

Using the construction outlined in Section 5.2.4 we can consider any D-dimensional hyperbolic reflection group and obtain a tiling which is $D+1$-colorable and which has a $D+1$-valent graph. In particular, we can consider hyperbolic tilings in 3D which are 4-colorable. There exist four regular hyperbolic tilings in 3D of which two are self-dual tilings and two related by duality. The self-dual ones are a tiling by dodecahedra, denoted $\{5,3,5\}$, and one by icosahedra, denoted $\{3,5,3\}$. The other are a tiling by cubes $\{4,3,5\}$ and its dual $\{5,3,4\}$. All of these give rise to codes with maximum stabilizer weight 120. Here we will focus on the $\{5,3,5\}$-tiling, which is the unique self-dual tiling of space by dodecahedra where five dodecahedra are placed around an edge. Performing the Wythoff construction on a family of closed manifolds, all equipped with a $\{5,3,5\}$-tiling yields a code family where checks are of weight 20 and 120. The weight of the stabilizer is given by the order of the subgroup of the full reflection group which is generated by all except for one of the generators. The smallest example is a $[[7200,5526,4]]$ code.

5.5.2. PIN CODES FROM CHAIN COMPLEXES

In Sec. 5.2.4 we showed how from any $\mathbb{F}_2$ chain complex one can construct a pin code relation. In this section we explore some specific examples of chain complexes and the corresponding pin codes.

One way to obtain arbitrary length chain complexes is to repeatedly use the hypergraph product with a classical code. The hypergraph product was introduced in [16] as a way to turn any two classical codes into a quantum code. This product can be viewed as the tensor product of chain complexes, which takes two length-2 chain complexes to a length-3 chain complex. More generally the product of a length-k_1 and length-k_2 chain complexes yields a length-(k_1+k_2-1) chain complex. This generalization and its characteristics has been studied in the context of homological codes [17, 40, 41]. We consider here the approach of [41] but look at the resulting chain complexes from the point of view of pin codes.

The idea goes as follows: consider $\mathscr{A}$, a $\mathbb{F}_2$ chain complex of length k, characterized by $\mathbb{F}_2$-vector spaces $(\mathscr{A}_j)_{0\le j\le k-1}$ and $k-1$ boundary maps $\partial_j^{\mathscr{A}} : \mathscr{A}_j \to \mathscr{A}_{j-1}$, obeying Eq. (5.3). We now take the product with a chain complex of length 2. Note that any two vector spaces, $\mathscr{B}_1$ and $\mathscr{B}_0$ and any linear map between them $\partial^{\mathscr{B}} : \mathscr{B}_1 \to \mathscr{B}_0$ defines a length-2 chain complex. The product, $\mathscr{C} = \mathscr{A} \otimes \mathscr{B}$, is defined by $k+1$ vector spaces $\mathscr{C}_j$ for $0 \le j \le k$,

$$\mathscr{C}_j = (\mathscr{B}_1 \otimes \mathscr{A}_{j-1}) \oplus (\mathscr{B}_0 \otimes \mathscr{A}_j),$$

with the convention that $\mathscr{A}_{-1}$ and $\mathscr{A}_k$ are both the zero vector spaces. And the k boundary maps, $\partial_j^{\mathscr{C}} : \mathscr{C}_j \to \mathscr{C}_{j-1}$, are defined as

$$\begin{aligned} &\forall\, u = v \oplus w \in (\mathscr{B}_1 \otimes \mathscr{A}_{j-1}) \oplus (\mathscr{B}_0 \otimes \mathscr{A}_j) \\ &\partial_j^{\mathscr{C}}(u) = (\mathbb{1}_{\mathscr{B}_1} \otimes \partial_{j-1}^{\mathscr{A}} + \partial^{\mathscr{B}} \otimes \mathbb{1}_{\mathscr{A}_{j-1}})(v) + (\mathbb{1}_{\mathscr{B}_0} \otimes \partial_j^{\mathscr{A}})(w). \end{aligned}$$

One straightforwardly checks that the $\partial_j^{\mathscr{C}}$ are valid boundary maps, i.e. obeying Eq. (5.3).

Repeatedly taking the product with a length-2 chain complex therefore increases the length of the resulting chain complex each time by one. Moreover any binary matrix de-

fines a valid $\mathbb{F}_2$ chain complex of length 2 so this approach allows to explore numerically many pin codes.

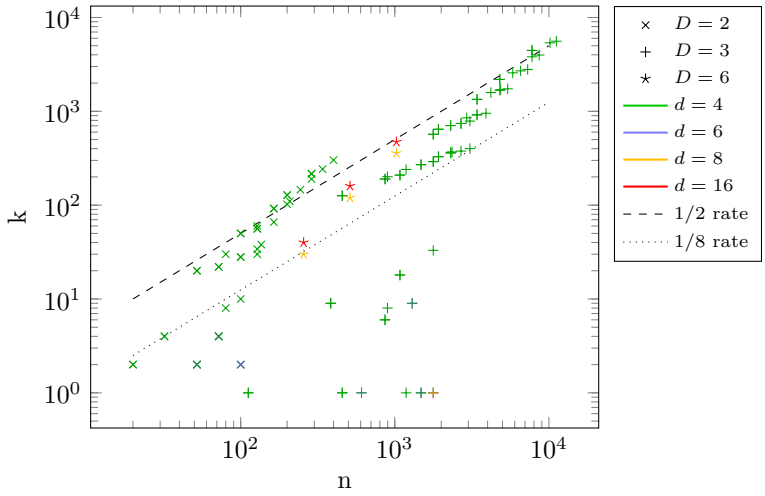


Figure 5.8: Plot of the $[[n, k, d]]$ parameters of pin codes from chain complexes described in this section. One application of the hypergraph product on 3×4 binary matrices yields the $D = 2$ pin codes represented by '×' and two applications the $D = 3$ pin codes represented by '+'. The $D = 6$ pin codes described in Table. 5.2 are represented by stars. The colors indicate the distance of the codes. The dashed and dotted lines represent the rates $k/n = 1/2$ and $k/n = 1/8$, respectively.

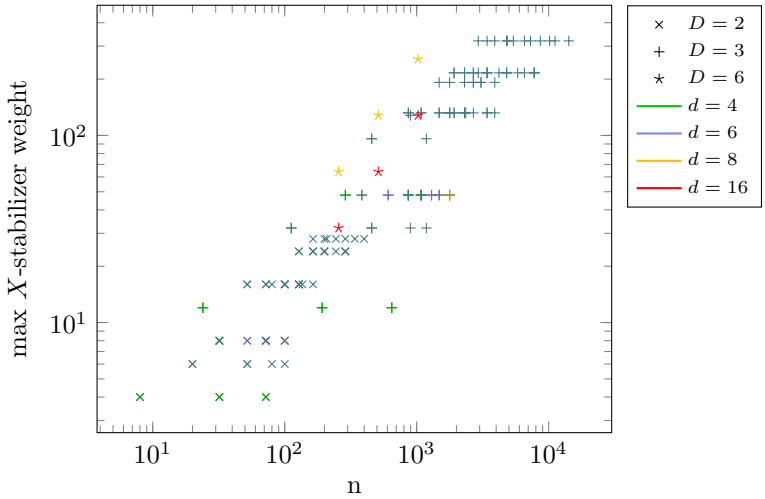


Figure 5.9: Plot of the maximum X-stabilizer weight of the pin codes from chain complexes described in this section.

We have looked at small binary matrices, up to 3×4, and their hypergraph product to

form pin code relations with $D = 2$ and $D = 3$. We plot in Fig. 5.8 the code parameters obtained $[[n, k, d]]$. Strinkingly these codes seem to show a general trend of high encoding rate k/n for a small distance. Indeed most of them are around $1/2$ rate but just distance 4 which is the lower bound guaranteed by Prop. 7. A few of them reach distance 6 or 8 but for significantly smaller rates. The codes yielding no logical qubits are not displayed in this plot. Note that this procedure is far from generating all chain complex of a given length.

We have also looked at a few pin code relations for $D = 6$ using small even size levels and the complete relation for F. Three notable examples are presented in Table. 5.2. When writing $2^{\times 6} \times 4$ we mean that 6 of the levels contain each 2 pins and the last one contains 4. Since we use the complete relation, the number of flags and the size of the pinned sets are easily computed as a product of the size of some levels. The number of logical qubits is computed numerically and the distance is upper bounded and we believe it is tight.

Table 5.2: Parameters of some $D = 6$ pin codes using the complete relation described by the size of the $D+1 = 7$ levels.

(x, z)	$2^{\times 6} \times 4$	$2^{\times 5} \times 4^{\times 2}$	$2^{\times 4} \times 4^{\times 3}$
$(2, 4)$	$[[256, 30, 8]]$	$[[512, 120, 8]]$	$[[1024, 358, 8]]$
$(3, 3)$	$[[256, 40, 16]]$	$[[512, 160, 16]]$	$[[1024, 472, 16]]$

We also represent the maximum weight of the X-stabilizers for these codes in Fig. 5.9. When checking for transversal phase gates for $\ell = 3$, most of the codes examined above do not satisfy (5.17) and therefore one cannot apply transversal R_3 on most of these codes.

5.5.3. PUNCTURING TRIPLY-EVEN SPACES

If pin codes in general are not guaranteed to fulfill all the requirements of transversality or quasi-transversality of Prop. 4 or Prop. 5, their stabilizers always form multi-orthogonal spaces, see Prop. 8. This is directly useful as multi-orthogonal spaces together with puncturing techniques can be used to construct codes fulfilling Prop. 5 (or Prop. 4 if the space is multi-even), as explained for example in [27]. We focus here on triply-even spaces and tri-orthogonal spaces. The idea goes as follows: take a binary matrix, G, whose rows generate a tri-orthogonal space, using Gaussian elimination it is always possible to put the matrix in the following form:

$$G = \begin{matrix} \\ k \updownarrow \\ r \updownarrow \end{matrix} \begin{matrix} \leftarrow k \rightarrow & \leftarrow n \rightarrow \\ \end{matrix} \begin{pmatrix} \mathbb{1} & G_1 \\ 0 & G_0 \end{pmatrix}. \tag{5.18}$$

To obtain this form one just performs row operations as well as column permutations and different column permutations will yield different G_0 and G_1. Then choosing the rows of G_0 as X-stabilizer generators and the rows of G_1 as X-logical operators (this fully specifies the Z-stabilizers and Z-logicals) yields a code fulfilling Prop. 5. Moreover the

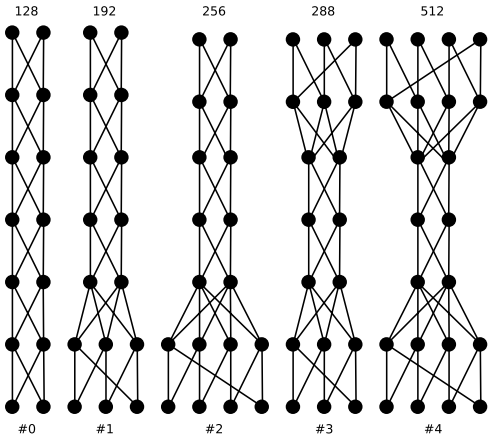


Figure 5.10: Some possible variations on Reed-Muller codes seen as pin codes on a chain complexes, by modifying the ends of the chain complex, the left most chain complex is used to define Reed-Muller codes. Example of dimension 6 are shown here, making variations on $\mathscr{RM}(2,7)$. The number of flags is written above each chain complex and an identification number is given below.

logical gate obtained is the transversal $T^\dagger$ so directly usable in a T-gate distillation protocol. It distills n magic states into k ones of better quality which depends on the distance of the code that has to be computed independently.

In [27], the authors use Reed-Muller codes, $\mathscr{RM}(r,m)$, to obtain initial tri-orthogonal spaces (even triply-even). Viewed as a pin code, $\mathscr{RM}(r,m)$ is a very simple chain complex, represented on the left of figure 5.10, see also section 5.2.5. This chain complex can be modified in several ways to obtain different pin codes. We tried different modifications in the case $D=6$, they are represented in figure 5.10. For all of them the 2-pinned sets generate a triply-even space.

We tried to randomly puncture the pin codes obtained from these chain complexes; similarly to [27] but without deploying the more advanced techniques. We were able to find a few interesting codes this way, see Table 5.3, which can be used to distill T magic states, see Chapter 1, Section 1.3.2. The obtained parameters γ, see Eq. (1.75), are similar but do not improve on the small examples found in [27].

Table 5.3: Some triorthogonal codes found by randomly puncturing the pin codes represented in figure 5.10.

code #	initial n	punctured code: $[[n,k,d]]$	$\gamma = \frac{\ln(n/k)}{\ln d}$
0	128	$[[116,12,4]]$	1.64
1	192	$[[175,17,4]]$	1.68
2	256	$[[236,20,4]]$	1.78
3	288	$[[261,27,4]]$	1.64
4	512	$[[466,46,4]]$	1.67

Table 5.4: Alternative construction of CSS codes with transversal T implementing some circuit of CCZ gates on the logical level.

$D=5, x=2$	$D=8, x=3$
$[[64,15,4]]$	$[[512,84,8]]$
$[[96,23,4]]$	$[[768,126,8]]$
$[[128,31,4]]$	$[[1024,168,8]]$
$[[144,35,4]]$	$[[1152,188,8]]$
$[[256,63,4]]$	$[[2048,332,8]]$

5.5.4. LOGICAL CIRCUITS OF CCZS

It is also possible to use the property of multi-orthogonality of pinned sets on a given pin code relation in a slightly different way. The construction proposed for Reed-Muller codes in [42] can be directly adapted to general pin code relations. This construction is the following: given a pin code relation on $D+1$ levels and a positive integer x, choose the $(x-1)$-pinned sets as X-stabilizer generators and impose the x-pinned sets to be the X-logical operators. This is enough to completely characterize a CSS code. Then by choosing carefully the parameters D and x, one can obtain a code guaranteed to satisfy Prop. 5 where the logical operation realized belongs to some level ℓ of the Clifford hierarchy. Roughly x has to be small enough so that the conditions in Prop. 5 hold, but large enough for the logical operation described by the weighted polynomial in Eq. (5.10) to be in the ℓth level of the Clifford hierarchy. When taken together these constraints become

$$x = \frac{D+1}{\ell}. \tag{5.19}$$

We can for example adapt the pin code relations presented in Fig. 5.10 to have the correct dimension D by inserting or removing levels of size 2 in the middle of the chain complexes and look at what code parameters they give. These parameters are compiled in Table. 5.4, for $D=5$ we remove the middle level and for $D=8$ we add two levels of size 2 compared to Fig. 5.10. All these codes support a transversal T and up to a Clifford correction the logical operation implemented is some circuit of CCZs characterized by which triple of X-logical operators have an odd overlap. This could be used in specific compilation techniques or by converting this resource to some more conventional T gate resources using results such as [43].

5.6. DISCUSSION

Quantum pin codes form a large family of CSS codes which we have just begun to explore. These codes can be viewed as a generalization of quantum color codes and the notions of boundaries, colored logical operators and shrunk lattices all generalize to pin codes. Pin codes also have a gauge code version with potentially similar advantage as the gauge color codes. The main property of pin codes is that their X- and Z-stabilizers form multi-orthogonal spaces. We have presented two concrete ways of constructing pin codes and numerically explored some examples. Several aspects of pin codes merit

further studying.

First, it would be interesting to find restricted families with good parameters and constant-weight stabilizer generators. Exploring other finite groups with even-order generators, other families of sparse chain complexes or finding other constructions of pin code relations altogether would help figuring out the achievable parameters for pin codes.

A second aspect concerns logical operators. Understanding if some conditions on the pin code relation F can make the logical operators fulfill the second condition of Prop. 4 or Prop. 5 would help in the design of codes with transversal gates. Also, logical operators and boundaries of 2D color codes have a richer structure than the colored logicals and boundaries that we have explored. It would be interesting to generalize the construction in [44] to pin codes with $D = 2$, as well as for larger D. Moreover, the structure of colored logicals plays a key role in decoding color codes [31–34]. Understanding if it can help in finding efficient decoders for more general pin codes is a natural question.

Finally, more extensively exploring tri-orthogonal spaces obtained from pin code relations and puncturing them to obtain good T distillation protocols as well as using them as the basis for T-to-CCZ or other protocols seems worth trying as distilling magic states will constitute a sizable fraction of any fault-tolerant quantum computation.

5

REFERENCES

[1] C. Vuillot and N. P. Breuckmann, *Quantum Pin Codes*, arXiv:1906.11394 [quant-ph] (2019), arXiv: 1906.11394.

[2] E. T. Campbell, B. M. Terhal, and C. Vuillot, *Roads towards fault-tolerant universal quantum computation*, Nature **549**, 172 (2017).

[3] A. R. Calderbank and P. W. Shor, *Good quantum error-correcting codes exist*, Physical Review A **54**, 1098 (1996).

[4] Steane Andrew, *Multiple-particle interference and quantum error correction*, Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences **452**, 2551 (1996).

[5] D. Gottesman, *Stabilizer codes and quantum error correction*, PhD Thesis, California Institute of Technology (1997).

[6] A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane, *Quantum error correction via codes over GF(4)*, in *Proceedings of IEEE International Symposium on Information Theory* (1997) pp. 292–.

[7] M. Grassl, T. Beth, and T. Pellizzari, *Codes for the quantum erasure channel*, Physical Review A **56**, 33 (1997).

[8] M. H. Freedman and D. A. Meyer, *Projective Plane and Planar Quantum Codes*, Foundations of Computational Mathematics **1**, 325âĂŞ332 (2001).

[9] M. H. Freedman, D. A. Meyer, and F. Luo, *Z_2-systolic freedom and quantum codes*, in *Computational Mathematics* (Chapman and Hall/CRC, 2002) pp. 287–320.

[10] A. Y. Kitaev, *Fault-tolerant quantum computation by anyons,* Annals of Physics **303**, 2 (2003).

[11] D. Bacon, *Operator quantum error-correcting subsystems for self-correcting quantum memories,* Phys. Rev. A **73**, 012340 (2006).

[12] H. Bombin and M. A. Martin-Delgado, *Topological Quantum Distillation,* Physical Review Letters **97**, 180501 (2006).

[13] A. A. Kovalev and L. P. Pryadko, *Quantum Kronecker sum-product low-density parity-check codes with finite rate,* Phys. Rev. A **88**, 012311 (2013).

[14] A. Couvreur, N. Delfosse, and G. Zémor, *A Construction of Quantum LDPC Codes From Cayley Graphs,* IEEE Transactions on Information Theory **59**, 6087 (2013).

[15] L. Guth and A. Lubotzky, *Quantum error correcting codes and 4-dimensional arithmetic hyperbolic manifolds,* Journal of Mathematical Physics **55**, 082202 (2014).

[16] J. Tillich and G. Zémor, *Quantum LDPC Codes With Positive Rate and Minimum Distance Proportional to the Square Root of the Blocklength,* IEEE Transactions on Information Theory **60**, 1193 (2014).

[17] B. Audoux and A. Couvreur, *On tensor products of CSS codes,* Annales de lInstitut Henri Poincaré D **6**, 239 (2019).

[18] A. Leverrier, J. P. Tillich, and G. Zémor, *Quantum Expander Codes,* in *2015 IEEE 56th Annual Symposium on Foundations of Computer Science* (2015) pp. 810–824.

[19] S. Bravyi and A. Kitaev, *Universal quantum computation with ideal Clifford gates and noisy ancillas,* Physical Review A **71**, 022316 (2005).

[20] H. Bombín, *Gauge color codes: optimal transversal gates and gauge fixing in topological stabilizer codes,* New Journal of Physics **17**, 083002 (2015).

[21] H. Bombin and M. A. Martin-Delgado, *Exact topological quantum order in $D = 3$ and beyond: Branyons and brane-net condensates,* Physical Review B **75**, 075103 (2007).

[22] A. Kubica and M. E. Beverland, *Universal transversal gates with color codes: A simplified approach,* Phys. Rev. A **91**, 032330 (2015).

[23] H. S. M. Coxeter, *Regular polytopes*, 3rd ed. (Dover Publications, New York, 1973) open Library ID: OL5436086M.

[24] M. W. Davis, *The Geometry and Topology of Coxeter Groups. (LMS-32) (London Mathematical Society Monographs)* (Princeton University Press, 2007) open Library ID: OL11182953M.

[25] P. McMullen and E. Schulte, *Abstract Regular Polytopes by Peter McMullen,* (2002).

[26] E. T. Campbell and M. Howard, *Unified framework for magic state distillation and multiqubit gate synthesis with reduced resource cost,* Physical Review A **95**, 022316 (2017).

[27] J. Haah and M. B. Hastings, *Codes and Protocols for Distilling T, controlled-S, and Toffoli Gates,* Quantum **2**, 71 (2018).

[28] S. X. Cui, D. Gottesman, and A. Krishna, *Diagonal gates in the Clifford hierarchy,* Physical Review A **95** (2017), 10.1103/PhysRevA.95.012329.

[29] A. Kubica, B. Yoshida, and F. Pastawski, *Unfolding the color code,* New Journal of Physics **17**, 083026 (2015).

[30] A. Bhagoji and P. Sarvepalli, *Equivalence of 2d color codes (without translational symmetry) to surface codes,* in *2015 IEEE International Symposium on Information Theory (ISIT)* (2015) pp. 1109–1113.

[31] N. Delfosse, *Decoding color codes by projection onto surface codes,* Physical Review A **89**, 012317 (2014).

[32] A. B. Aloshious and P. K. Sarvepalli, *Projecting three-dimensional color codes onto three-dimensional toric codes,* Physical Review A **98**, 012302 (2018).

[33] A. Kubica, M. E. Beverland, F. Brandão, J. Preskill, and K. M. Svore, *Three-Dimensional Color Code Thresholds via Statistical-Mechanical Mapping,* Phys. Rev. Lett. **120**, 180501 (2018).

[34] A. Kubica and N. Delfosse, *Efficient color code decoders in $d \geq 2$ dimensions from toric code decoders,* arXiv:1905.07393 [quant-ph] (2019), arXiv: 1905.07393.

[35] D. Poulin, *Stabilizer formalism for operator quantum error correction,* Phys. Rev. Lett. **95**, 230504 (2005).

[36] H. Bombín, *Single-shot fault-tolerant quantum error correction,* Phys. Rev. X **5**, 031043 (2015).

[37] N. Delfosse, *Tradeoffs for reliable quantum information storage in surface codes and color codes,* in *2013 IEEE International Symposium on Information Theory* (2013) pp. 917–921.

[38] N. P. Breuckmann and B. M. Terhal, *Constructions and Noise Threshold of Hyperbolic Surface Codes,* IEEE Transactions on Information Theory **62**, 3731 (2016).

[39] N. P. Breuckmann, *Homological Quantum Codes Beyond the Toric Code*, PhD Thesis, RWTH Aachen University (2017).

[40] E. T. Campbell, *A theory of single-shot error correction for adversarial noise,* Quantum Science and Technology **4**, 025006 (2019).

[41] W. Zeng and L. P. Pryadko, *Higher-dimensional quantum hypergraph-product codes,* arXiv:1810.01519 [math-ph, physics:quant-ph] (2018), arXiv: 1810.01519.

[42] N. Rengaswamy, R. Calderbank, and H. D. Pfister, *Synthesizing Stabilizer Codes that Support Physical T and $T^\dagger$ Gates*, in preparation (2019).

[43] M. Beverland, E. Campbell, M. Howard, and V. Kliuchnikov, *Lower bounds on the non-Clifford resources for quantum computations*, arXiv:1904.01124 [quant-ph] (2019), arXiv: 1904.01124.

[44] M. S. Kesselring, F. Pastawski, J. Eisert, and B. J. Brown, *The boundaries and twist defects of the color code and their applications to topological quantum computation*, Quantum **2**, 101 (2018).

6

CONCLUSION

There are still many challenges left in the field of quantum error correction and fault-tolerant universal quantum computation. They span the whole spectrum from experimental implementations and hardware improvements to theoretical fault-tolerant schemes for distillation of magic states, better error correcting codes for storing and processing information. Decisive progress on all these fronts appears to be necessary in order to reach a point where universal quantum computation becomes feasible. In this thesis we have presented several results going in this direction.

First we presented an experiment demonstrating a proof of concept for fault-tolerant quantum computation. The experiment was based on a four qubit error detecting code with a fairly large set of fault tolerant logical gates. The usage of error detection and fault-tolerant circuits showed an average improvement of the sampling task tested.

Second, we presented and analyzed a scheme, the toric-GKP code, to use quantum continuous variable degrees of freedom to encode discrete quantum information in a scalable way. We show that if one can get enough control to implement a bosonic code such as the GKP code and then concatenate it with a qubit code, this can significantly improve the performance of the overall scheme. Remarkably we numerically observe a threshold even when all syndrome measurements are noisy, which can also be interpreted as a threshold against leakage errors. We also exhibited the statistical physics model representing the behavior of the maximum likelihood decoder for this scheme which allowed us to design a practical decoder but would also give access to theoretical bounds on the performance when studied further. We also prove a no-go theorem for protecting quantum continuous variable degrees of freedom using linear oscillator codes highlighting the benefits of discretization. This is a first step in the direction of better integration of hardware and error correcting codes to improve the overall performance of error correction.

Third, we presented a formalism to analyze code deformation techniques and illustrated it on previously known and new code deformation processes. Code deformation techniques seem inevitable in most current proposed architectures of quantum computers. This formalism allows to evaluate them, also providing a prescription about how

decoding should be performed during these procedures. In addition, the new technique we introduce, namely plain surgery, has the potential to improve the reliability of these operations. It also allows us to investigate a hybrid scheme between topological codes and concatenation.

Fourth, we presented a new family of CSS codes exhibiting some multi-orthogonality properties that we call quantum pin codes. They can be viewed as a generalization of color codes in which we remove the geometrical and topological structure but keep the otherwise transversality friendly properties. This is a first step towards codes with transversal non-Clifford gates and high encoding rates and we are indeed able to present some small codes with transversal T gates or circuits of CCZ gates.

All these fronts have room for expansion. As control over the hardware becomes more precise and the number of simultaneously controlled systems grows, test and optimization of fault-tolerant circuits should be conducted. Interactions between different layers in the architecture of a quantum computer should be better understood and taken advantage of. Questions about the limits of quantum error correcting codes in terms of encoding rate, distance and fault-tolerant implementation of universal gate sets are still open and solving them has a great potential. Ultimately, actively error corrected universal quantum computers constitute some dynamical phases of matter and we would like to understand what they are made of, how they behave and how they are constructed.

ACKNOWLEDGEMENTS

A large portion of science happens via interactions between people. Exchanging ideas and learning from people is one of the most rewarding part of research and certainly drove a lot of the projects for this thesis. It is difficult to be exhaustive when trying to enumerate all the exchanges and their protagonists but know that I certainly value each and everyone of them.

I would like to first extend my full gratitude to my supervisor, Barbara Terhal. I feel particularly fortunate to have met and worked with you. Under your supervision I felt, at the same time, free to explore my own ideas and guided through challenging and exciting research. You have never ceased to inspire me and will continue to be a role model for me for years to come.

I would like to thank warmly Carlo Beenakker, Leo DiCarlo, Robert König, Anthony Leverrier, Lieven Vandersypen and Ronald de Wolf, for accepting to be my thesis committee.

I consider myself very lucky to have been able to collaborate during this thesis with amazing people and hope to continue on many more projects to come. Thank you Nikolas for teaching me so much and being overall my big brother in PhD. Thank you Leonid for all the discussions, insights and tenacity to get through this very long paper with Barbara and I. Thank you Lingling for all the Monday meetings. I'm very happy that we pushed this project to its conclusion. Thank you Ben as much for the physics as for the political perspectives. Thank you Earl for all the discussions, and all the great conferences I attended that you organized. Thank you Ani for rooting for always more classical coding theory perspectives. Thank you Victor for teaching me and getting me interested in bosonic codes.

The time working on this thesis was split in half in two geographical locations, Aachen and Delft. I would like to thank all my colleagues and friends from both these places. From Aachen, many thanks to Stefano for being the best roommate I could hope for. Thank you Kasper for sharing an office with me and for the many interesting discussions at the whiteboard. Thank you David for having me tutor for the quantum information course, it was a very formative experience. Thank you Fabian for all the interesting discussions ranging from physics to politics; lunch is never the same without your inputs to the conversation. Thanks to everyone that came and participated in the journal club. Thanks to Alessandro, Ananda, Cica, Joel, Manuel, Suzanne for the good coffee and great atmosphere at the institute.

From Delft, many thanks to Joel and Radha for all the discussions about science, philosophy and much more, as well as all the dinners, it was truly lovely. Thank you Xiaotong for the science, movies and games. Thanks to Yang, Daniel and Jonathan for the bosonic research. Thank you Francesco for taking on the journal club. Thank you Jonas and Kenneth for the science and roll-playing action. Thanks to Helen and Jenny for the expert administrative support in Aachen and Delft.

Also thanks to the many colleagues and friends met at conferences or workshops, including, Aleks, Antoine, Benjamin, Benji, Chris, Daniel, Giacomo, Juani, Markus, Michael (×2), Narayanan, Nicolas, Niel, Sam, Simon, Vivien.

I would also like to thank the IBM Quantum Experience team for providing, early on, extensive access to their chips and support on how to interface with them. I should also mention that the views expressed in this thesis are mine and do not reflect the official policy or position of IBM or the IBM Quantum Experience team.

During my undergraduate studies I had the great chance of working with many amazing people who kindly hosted me for research internships which made me discover and appreciate the field of quantum information. Many thanks to Erwan Faou, Fabio Nobile, Zhengfeng Ji, Debbie Leung, Christina Kraus, Mikhail Baranov and Thomas Vidick. I have very fond memories of these internships and I learned a lot from each of you. I would also like to thank the great teachers who taught me about science and advised me about academia, notably Iordanis Kerenidis, David Pichardie, Luc Bougé and Dominique Lavenier.

Thanks to my very good friends from Rennes, Gaspard, Mathias, Olivier, Pierre, Simon, Thibault, Yannick for the good times, discussions, coding contests and much more!

Thanks to my familly for all their support, Maman, Papa, Carole, Olivier, I have been far away for a few years now and I happily expect to be able to see you much more often from now on.

Finally thousand thanks to Lisa: Living with you is everything I hoped for and I'm confident in the future knowing that I'll be at your side and you at mine. Some extra thanks to my future daughter for not being too impatient and having waited to be born after I finish writing these words.

CURRICULUM VITÆ

Christophe VUILLOT

17-11-1990 Born in Clamart, France.

EDUCATION

2005–2008 High School
Lycée Marie Curie, Sceaux, France
Baccalauréat, mention très bien (2008)

2008–2010 Preparatory classes, MPSI/MP*
Lycée Henri IV, Paris, France
Admitted at ENS Rennes through national competitive exam (2010)

2010–2012 Bachelor in Computer Science and Physics
Computer Science, ENS Rennes, Rennes, France (2010–2011)
Physics, Université Rennes 1, Rennes, France (2011–2012)

2011–2015 Master in Computer Science and Physics
1st year in Computer Science, ENS Rennes, Rennes, France (2011–2012)
1st year in Physics, ENS Cachan, Cachan, France (2013–2014)
Master of Computer Science, Université Paris Diderot, Paris, France (2015)

AWARDS

2010–2015 Normalien, ENS Rennes
Four year scholarship awarded through national competitive exams.

2018 IBM Q Best Paper Award, 1st place
Highest-impact scientific papers by a master student, PhD student or postdoctoral researcher using the IBM Q Experience and Qiskit as tools to achieve the presented results.

LIST OF PUBLICATIONS

JOURNAL ARTICLES

6. **C. Vuillot**[1], L. Lao[1], B. Criger, C. G. Almudéver, K. Bertels and B. M. Terhal, *Code deformation and lattice surgery are gauge fixing*, New Journal of Physics **21**, 033028 (2019).

5. **C. Vuillot**, H. Asasi, Y. Wang, L. P. Pryadko and B. M. Terhal, *Quantum error correction with the toric Gottesman-Kitaev-Preskill code*, Physical Review A **99**, 032344 (2019).

4. V. V. Albert, K. Noh, K. Duivenvoorden, D. J. Young, R. T. Brierley, P. Reinhold, **C. Vuillot**, L. Li, C. Shen, D. M. Girvin, B. M. Terhal and L. Jiang, *Performance and structure of single-mode bosonic codes*, Physical Review A **97**, 032346 (2018).

3. **C. Vuillot**, *Is error detection helpful on IBM 5Q chips ?*, Quantum Information & Computation **18**, 11&12, 0949-0964 (2018).

2. N. Breuckmann, **C. Vuillot**, E. Campbell, A. Krishna and B. M. Terhal, *Hyperbolic and semi-hyperbolic surface codes for quantum storage*, Quantum Science and Technology **2**, 035007 (2017).

1. E. Campbell, B. M. Terhal and **C. Vuillot**, *Roads towards fault-tolerant universal quantum computation*, Nature **549**, 172–179 (2017).

PREPRINTS

2. **C. Vuillot** and N. Breuckmann, *Quantum Pin Codes*, under review by IEEE Transactions on Information Theory, arXiv:1906.11394, (2019).

1. E. Faou, F. Nobile and **C. Vuillot**, *Sparse spectral approximations for computing polynomial functionals*, arXiv:1207.3728, (2012).

[1] Equal contributions