High-dimensional permutation-invariant quantum error-correcting codes

MSc Thesis

written by

Vladyslav Visnevskyi

under the supervision of **Dr. Māris Ozols** and **Dr. Arghavan Safavi-Naini**, and submitted in partial fulfillment of the requirements for the degree of

MSc in Physics and Astronomy, Theory Track

at the *Universiteit van Amsterdam*.

Date of submission:
July 29, 2025

Examiner:
Prof. Dr. Kareljan Schoutens

QuSoft
Research Center for Quantum Software

University of Amsterdam

CWI
Research institute for mathematics & computer science in the Netherlands

Abstract

We study properties of Permutationally-Invariant (PI) quantum error-correcting codes [PR04]. We first derive the Knill-Laflamme (KL) conditions for qudit PI codes, based on the idea that erasure and deletion errors are equivalent when acting on permutationally-symmetric states [AAB24]. We then conduct a numerical study of solutions to KL equations for qubit PI codes, and conjecture, based on numerical evidence, an upper bound on the code distance d_C of qubit PI codes in terms of their block length n: $d_C \leq \sqrt{12n-3}/3$. Moreover, we also conjecture that qubit PI codes with real coefficients can achieve this upper bound, and we highlight the symmetries in the codewords of such real qubit PI codes. Finally, we construct our own family of qudit PI codes that encode a single logical qudit with local dimension d into physical qudits with local dimension $q \geq d$. The codespace of these qudit PI codes can be represented by a discrete subspace of a $(q-1)$-dimensional simplex. We then compare our code construction to other qudit PI code families.

Acknowledgments

First of all, I would like to thank my supervisors for their invaluable guidance. Thank you Māris for always being readily available to answer my questions, for the many hours spent discussing ideas and solutions, and for all the personal support. I'm deeply grateful for your encouragement and the confidence you've helped me build along the way. Thank you Arghavan for all your physical insight into the problems, for all the meaningful ideas, and for your willingness to help with issues extending beyond the thesis itself. I also want to thank Jiri Minař for all your feedback and ideas, as well as for finding time for our regular meetings, even while outside Amsterdam. I want to thank Liam Bond for teaching me so much about Julia, and for all the time spent debugging. I want to thank Nicolas Resch for all the valuable discussions on classical error correction. I also want to thank Dmitry Grinko for organizing the Quantum Error Correction journal club together with Nicolas, and for recommending me to attend the Les Houches FTQC summit. Even though this project was my first introduction to the field of quantum error correction, I have learnt so much. It was truly a pleasure and a privilege to do my thesis in such a wonderful team.

Contents

1 Introduction

In recent years, qudit quantum computation has been getting more attention, both in theoretical and experimental research [WHSK20]. For many physical systems, including photons [Mil09], superconducting systems [CNHM03], trapped ions [BW08], neutral atoms [SWM10], and spin qubits [VE19], more than two physical states, such as hyperfine electronic levels of the atoms and ions or polarization states of photons, are typically available for local access and control. Qudit systems can utilize these additional states, unlike qubit systems. This allows qudit quantum computers to operate more efficiently by reducing the elementary gate complexity of important gates like the Toffolli or the Hadamard [KNXSF20; RRG07]. Moreover, qudit quantum computers generally have a better lower bound for unitary synthesis [Nie05]. Qudit systems also have many advantages in quantum communication, in part due to their higher noise-resilience [CDBO19].

Quantum error correction (QEC) is a necessary ingredient for fault tolerant quantum computing, which aims to recover quantum information after errors are introduced from the environment into the computational system. QEC achieves this by encoding logical qubits (qudits) into so called "codewords", which are states in a larger space of physical qubits (qudits), with errors acting on this larger physical space. Together, the encoding map and the space it encodes into is called a *code*. The logical space is protected from errors since in encoded form it contains a certain level of redundancy, which allows it to be recoverable even after being corrupted.

Numerous studies have explored error correction for the qubit model of quantum computation, along with fault-tolerant properties of qubit codes. Some families of quantum codes can be easily generalized to qudits. However, for the majority of quantum codes, a generalization of their properties to the qudit model is not so obvious. For instance: even extending the qubit Clifford group C_n^2 to C_n^d of n qudits is a highly non-trivial task [HDD05], and, in general, with the number of group elements growing as $O(d^{n^2}\prod_{i=1}^{n}(d^{2i}-1))$, finding transversal gates [Got24] for qudit codes becomes more challenging as d increases. Many newer papers focus on valuable extensions of quantum codes to the qudit model [UG24; Bro+24], establishing thus basic building blocks for fault-tolerant qudit quantum computing.

In this thesis we explore an interesting class of quantum codes, called Permutationally Invariant (PI) codes, which utilize permutation symmetry to generate codewords. PI codes are not stabilizer codes [Got97] by the conventional definition, yet they could be defined as "quantum codes stabilized by the symmetric group". This exotic family of codes was first introduced in [PR04], where the idea stemmed from studying codes stabilized by a non-abelian group. PI codes have remained relatively unexplored for a long time after first being introduced, but in recent years this code family turned out to have many useful properties. For instance, PI codes are excellent candidates for practical implementation of fault-tolerant quantum channels, and for error-correction on quantum platforms with large connectivity [AAB24; OB24]. From the physics perspective, they are useful in fully-connected systems for easy implementation of global transversal gates. Interestingly, PI codewords also naturally appear as the ground states of the Heisenberg ferromagnetic model [Ouy14; AAB24]. This family of non-additive codes [Got97] is generally flexible to correct for various error types,

offering protection against arbitrary t-qubit errors, deletion and erasure errors, amplitude damping, absorption and emission errors, phase decoherence, and many more [Ouy14; Ouy17; AB24]. Moreover, the shortest known quantum codes to correct t deletions come from this family, with block length n, i.e. number of physical encoding qubits, scaling as $n = O(t^2)$ with error weight t [AAB24]. Additionally, PI family contains codes with exotic transversal gates that contribute to the most efficient single qubit universal gate set [KT23], a result interesting to explore further in the case of qudits.

Some existing works [Ouy17; AB24] describe certain qudit subfamilies of PI codes, however, to the best of our knowledge, no complete qudit generalizations for the PI code family have been made so far. We extend the PI formalism to qudits, derive general necessary and sufficient conditions for qudit PI codes to be error-correcting (Section 2), and study some of their properties. We then perform a numerical study of solutions for the KL conditions of qubit PI quantum error-correcting codes (see Section 3) and conjecture, based on numerical evidence, an upper bound on the code distance d_C of arbitrary qubit PI quantum error-correcting codes: $d_C \leq d_{min} = \sqrt{12n-3}/3$. We also claim that qubit PI codes with real coefficients can achieve a code distance scaling that is as good as of qubit PI codes with complex coefficients. Moreover, based on numerical evidence we conjecture that real minimal qubit PI QEC have certain symmetries in their codewords that we call *mirror* and *phase-flip* symmetries (see equation (36)). A table summarizing the code properties of all PI codes reviewed in this thesis is given below, see Table 5. To ensure reproducibility of our numerical findings, all the code is openly accessible on our GitHub repository [VB].

We then construct our own *qudit* PI code family, using codewords that could be graphically organized as points of a simplex (see Section 4). This construction works for encoding a qudit of arbitrary logical local dimension d into ququits of physical local dimension $q \geq d$. Lastly we compare our *qudit-to-ququit* PI QEC codes to existing *qudit-to-qubit* PI QEC codes [Ouy17] (Section 6).

1.1 Notation and basic definitions

Throughout the thesis, we often use the term **local dimension** to describe the arity of a single qudit. For instance, a qubit has local dimension 2, a qutrit has local dimension 3, and so on. We also refer to encoded states as **physical** states, and to computational states before encoding (and after decoding) as **logical** states. Note that in this thesis we don't consider the fault-tolerant regime (see [Got24] for definition of fault-tolerance), so we assume errors happen only on encoded, i.e. physical states. Specifically, we assume errors happen **only** after encoding and/or before decoding. Physical and logical states have physical and logical local dimension, respectively. We often denote the physical local dimension with a q and refer to physical quantum units as qu**q**its. Similarly, we denote logical local dimension with a d of logical qu**d**its. Additionally, if not specified otherwise, whenever we denote a quantum state with an $|x\rangle$ we assume it is a **basis state** of the computational basis, i.e. $|x\rangle$ is a basis vector of $(\mathbb{C}^d)^{\otimes n}$ for some arbitrary n and d. The following notation and definitions are general, for both logical and physical states:

- $\lambda \overset{q}{\vdash} n := \{\lambda_0, \lambda_1, \ldots, \lambda_{q-1}\}$ s.t. $\sum_{i=0}^{q-1} \lambda_i = n$ – means λ is a partition of n into q parts, and $\sum_{\lambda \overset{q}{\vdash} n}$ denotes a sum ranging over all such partitions for given n, q. Note that the λ_i are *not* sorted in our definition for the partition;

- $l(\lambda)$ – length of the partition λ, e.g. for $\lambda \overset{q}{\vdash} n$ we have $l(\lambda) = q$;

- $\binom{n}{\lambda} := \frac{n!}{\lambda_0!\lambda_1!\ldots\lambda_{l(\lambda)-1}!}$ – multinomial coefficient, defined for a partition λ of $n \in \mathbb{N}$;

- $\cancel{x_i}$ – the string x with the entry x_i deleted, i.e. for a size n string $x = (x_1, \ldots x_{i-1}, x_i, x_{i+1}, \ldots, x_n)$ we have $\cancel{x_i} := (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$;

- $N_i(x)$ – number of entries "i" in string x, e.g. $N_2(021230) = 2$;

- $w_q(x) = \{N_0(x), N_1(x), \ldots, N_{q-1}(x)\}$ – weight of a string x with alphabet $\{0, 1, \ldots, q-1\}$ of size q (or equivalently, weight of a quqit computational basis state $|x\rangle$ defined via this string), defined as a partition listing how many times each entry "i" occurs in x for $i \in \{0, 1, \ldots, q-1\}$;

- $X_\lambda^n = \{x \in \{0, 1, \ldots, l(\lambda)-1\}^n \mid w_{l(\lambda)}(x) = \lambda\}$ – the set of all strings x of size n with alphabet $\{0, 1, \ldots, l(\lambda)-1\}$ that have weight λ;

- $[d] = \{0, 1, \ldots, d-1\}$ – the set of all integers from 0 to $d-1$;

- We use the upright font "i" to denote the imaginary number, for example: $e^{\mathrm{i}\pi}$.

Definition 1.1. *A Dicke state $|D_\lambda^n\rangle$ is a normalized and permutationally invariant state given by a linear combination over all n-quqit states of weight λ:*

$$|D_\lambda^n\rangle = \frac{1}{\sqrt{\binom{n}{\lambda}}} \sum_{x=(x_1,\ldots,x_n)\in X_\lambda^n} |x_1\rangle \otimes |x_2\rangle \otimes \cdots \otimes |x_n\rangle \tag{1}$$

Since a Dicke state is itself a superposition over states of some fixed weight λ, we can also refer to the λ index of the Dicke state as its *weight*. Note that Dicke states of different weight belonging to the same Hilbert space (i.e. same n) are *orthogonal*, $\langle D_\lambda^n | D_{\lambda'}^n \rangle = \delta_{\lambda,\lambda'}$, and together they form the complete basis for the *symmetric subspace*.

Definition 1.2. *A qudit PI code $\mathcal{C}$ encoding a single logical qudit of local dimension d into n physical quqits of local dimension q, is a vector space spanned by the basis set of codewords of the form*

$$|c_i\rangle = \sum_{\lambda \overset{q}{\vdash} n} \alpha_{i,\lambda} |D_\lambda^n\rangle , \tag{2}$$

where $i \in \{0, 1, \ldots, d-1\}$ are the codeword indices and $\alpha_{i,\lambda} \in \mathbb{C}$.

- We say an error E has *weight t* if it acts non-trivially on ***at most*** t quqits of the system. Note to be careful and not confuse this with the weight of a string x or a computational quantum state $|x\rangle$ defined via this string.

- For a quantum code $\mathcal{C}$ encoding k logical qudits into n physical quqits, we refer to n as the *block length* of this code. Note that k is then usually called *message length*, but throughout this thesis our message length is always 1, since we always encode a single qubit/qudit in the codes we are working with.

1.2 Preliminaries

In QEC, from the physics perspective, we consider the most general case of some open system, where errors are introduced as some unwanted interaction with the environment. As any interaction, its action on the quantum state can be described by a quantum channel, which we then call an *error channel*. In Kraus form [NC10], we can write the error channel $\mathscr{E}$ as:

$$\mathscr{E}(\rho) = \sum_k E_k \rho E_k^\dagger \tag{3}$$

where ρ is the quantum state, and E_k are Kraus operators of the error channel, which satisfy:

$$\sum_k E_k^\dagger E_k = \mathbb{1} \tag{4}$$

The Kraus operators E_k linearly span a space, which we refer to as the *error set* and denote as $\mathscr{E}$, just like the error channel. In this paper, we consider linear spans over $\mathbb{C}$, but it is generally possible to define error sets over arbitrary fields. Any element $E \in \mathscr{E}$ of the error set we call an *error*. We say an error has *weight* t if it acts non-trivially on *at most* t quqits. Notice that by our definition an error is not necessarily a Kraus operator, but some general linear operator, yet physically any error would be a Kraus operator in any one of the possible Kraus decompositions (which is not unique) of a channel. Technically, we then should have defined $\mathscr{E}$ as a *convex* linear span of E_k, but mathematically this rescaling plays no difference when studying quantum codes, so we keep the definition more general. Note, however, that the physical probability of an error E_k from a Kraus set occurring on a quantum state ρ is $\mathrm{Tr}(E_k \rho E_k^\dagger)$, so that the final quantum state after an error E_k occurs is:

$$E_k : \rho \to \frac{E_k \rho E_k^\dagger}{\mathrm{Tr}(E_k \rho E_k^\dagger)}. \tag{5}$$

Keep in mind that when considering actual error channels that can physically occur, the error set would instead be the set of Kraus operators in any of the possible different Kraus representations of the error channel. Choosing different Kraus representations would correspond to choosing different error sets. The general definitions and theorems of QEC, described below, still apply in this special case.

A quantum code (QC) in general can be defined via its codewords, which together span a codespace. For a quantum error-correcting code (QECC) we also specify which error set it can correct. The definition is as follows:

Definition 1.3. *A quantum error-correcting code $\mathcal{C}$ correcting errors from the set $\mathscr{E}$ is a vector space $\mathcal{C}$, called codespace, equipped with a partial isometry $U : \mathcal{H}_L \to \mathcal{H}_P$, called*

encoder, which satisfies $\mathcal{C} = \text{Image}(U)$*, and has the following property:* $\exists$ *a quantum channel* $\mathscr{D} : End(\mathcal{H}_G) \to End(\mathcal{H}_L)$*, called decoder, such that* $\forall E \in \mathscr{E}$, $\forall |\psi\rangle \in \mathcal{H}_L$

$$\mathscr{D}(EU |\psi\rangle \langle\psi| U^\dagger E^\dagger) = p(E) |\psi\rangle \langle\psi| \tag{6}$$

where $\mathcal{H}_L$ *is the logical Hilbert space,* $\mathcal{H}_P$ *is the physical Hilbert space, and* $E \in \mathscr{E}$ *are acting from the physical Hilbert space* $\mathcal{H}_P$ *to some corrupted Hilbert space* $\mathcal{H}_G$*, i.e.* $E : \mathcal{H}_P \to \mathcal{H}_G$*.*

In the above definition, the coefficient $p(E)$ appears since E is a general linear map, as discussed previously, and therefore the final state might not be normalized properly. Again, in the case of physical errors expressed as Kraus operators, the coefficients $p(E)$ would be related to the probabilities of the errors E happening. Specifically, there exists a choice of a Kraus set for the error channel such that these will be *exactly* the probabilities of the errors, and for other choices of the Kraus set the probabilities will be related to actual probabilities via the unitary transform transforming the corresponding Kraus sets (see sections 2.4.2 and 2.5.2 in [Got24] for more intuition on this). Notice how it's not trivial that the coefficients $p(E)$ depend only on the error E and not on the state ψ, that is, generally we would expect to see $p(E, |\psi\rangle)$ instead. However, one can show that they indeed do not depend on the state $|\psi\rangle$ [Got24]. Intuitively, we don't want the probabilities of the errors to depend on the codewords they are acting on, otherwise we might end up in some convex combination (mixed state) of logical codewords after decoding.

Note also that we usually omit the encoder U when discussing a quantum error correcting code $\mathcal{C}$ correcting an error set $\mathscr{E}$, because any two different partial isometries $U, U' : \mathcal{H}_L \to \mathcal{H}_P$ with $\text{Image}(U) = \text{Image}(U')$ are necessarily related via some unitary $V : \mathcal{H}_L \to \mathcal{H}_L$ as $U' = UV$, so all encoders for a codespace $\mathcal{C}$ are unitarily-equivalent, therefore the corresponding decoders would be unitarily-equivalent too, so by omitting the encoder we don't lose any information about the structure of the code and of the error set. Explicitly considering the encoder and decoder would be relevant only when discussing the complexity of using a QECC in a circuit, but not when discussing the code's properties. However, the same codespace can be used to correct different error sets, and in general: *there is no notion of a unique largest error set* ***correctable*** *for a given codespace.* Therefore it is important to include the error set in the definition of a quantum error-correcting code, and we will only omit it when the error-set is easily determined from context, or if it is a default set of arbitrary weight-t errors. Thus, we will often denote a QECC with a quantum code $\mathcal{C}$ correcting errors from $\mathscr{E}$ as $(\mathcal{C}, \mathscr{E})$.

An important property of any quantum error correcting code $(\mathcal{C}, \mathscr{E})$ is linearity [Got24]:

Theorem 1.1. *If a quantum code* $\mathcal{C}$ *can correct some errors* E, F*, it can also correct* $\alpha E + \beta F$*, for* $\alpha, \beta \in \mathbb{C}$*.*

Proof. See proof in the Appendix (A). ■

So then our earlier definition of an error set $\mathscr{E}$ as a linear span of Kraus operators is justified: if a code can correct the Kraus operators, it can correct any error from the error set. This also implies an important corollary:

Corollary 1.1. *If the set of correctable errors $\mathscr{E}$ of a QECC includes all weight-t n-fold tensor products of (qudit) Pauli operators and identity, this QECC can correct arbitrary weight-t errors.*

Proof. This result follows simply by linearity, since any general linear matrix having support on t quqits can be decomposed as a linear combination of t-fold tensor products of Pauli matrices and identity. This applies to both the qubit and the qudit case, except in the qudit case we would have a qudit generalization of the (unitary) Pauli matrices. ■

For a quantum code $\mathcal{C}$ to be error correcting on a set $\mathscr{E}$, it must satisfy certain necessary and sufficient quantum error-correction conditions, called the Knill-Laflamme (KL) conditions [Got24; KL97]:

Theorem 1.2. *$(\mathcal{C}, \mathscr{E})$ is a quantum error-correcting code iff $\forall \left|\psi\right\rangle, \left|\phi\right\rangle \in \mathcal{C}, \forall E_a, E_b \in \mathscr{E}$:*

$$\left\langle\psi\right| E_a^\dagger E_b \left|\phi\right\rangle = C_{ab} \left\langle\psi|\phi\right\rangle \tag{7}$$

where $C_{ab} \in \mathbb{C}$ is zero whenever $p(E_a) = 0$ or $p(E_b) = 0$.

Proof. See proof of Theorem 2.7 in [Got24]. ■

The intuition to these conditions is that orthogonal codewords should stay orthogonal even after errors are introduced (in order to be detectable and distinguishable), and the errors themselves, being general linear maps, when restricted to the codespace should act as unitaries, in order to be reversible.

Definition 1.4. *The distance of a QECC $(\mathcal{C}, \mathscr{E})$ is the minimum weight d of an error F such that:*

$$\left\langle\psi\right| F \left|\phi\right\rangle \neq p(F) \left\langle\psi|\phi\right\rangle \tag{8}$$

With that we know, for instance, that if a QECC can correct all weight-t errors, it must have a distance of $d \geq 2t + 1$.

Similarly to an error-correcting code, a quantum code can also be *error-detecting*. Rather, for a given codespace $\mathcal{C}$, correction or detection depends on which function one assigns it to execute, with respect to the error set. That is, for a codespace $\mathcal{C}$, we can assign a correctable error set, and a *detectable* error set. We say that a quantum code is error-detecting according to the following definition:

Definition 1.5. *A quantum code $\mathcal{C}$ is error-detecting with respect to the error set $\mathscr{E}$ if for the projector Π on the codespace $\mathcal{C}$ the following condition holds $\forall E \in \mathscr{E}, \left|\psi\right\rangle \in \mathcal{C}$:*

$$\Pi E \left|\psi\right\rangle = q(E) \left|\psi\right\rangle, \tag{9}$$

where again the coefficient $q(E) \in \mathbb{C}$ is related to the probability of the error E occuring.

Here, again, it turns out that the coefficient $q(E)$ depends on the error only [Got24]. The idea is that a measurement using this projector Π should either return the initial state, or detect that an error happened, but never return a different codeword belonging to the space $\mathcal{C}$. We can formulate conditions for error-detection, similar to the KL conditions [Got24]:

Theorem 1.3. *$(\mathcal{C}, \mathscr{E})$ is a quantum error-detecting code iff $\forall \left|\psi\right\rangle, \left|\phi\right\rangle \in \mathcal{C}, \forall E \in \mathscr{E}$:*

$$\left\langle\psi\right| E \left|\phi\right\rangle = q(E) \left\langle\psi|\phi\right\rangle \tag{10}$$

Proof. See proof of Theorem 2.9 in [Got24]. ■

Clearly, this implies that a quantum code with distance d can detect arbitrary errors of weight $d-1$. The main conceptual difference between correction and detection for a given codespace $\mathcal{C}$, is that unlike with the correctable error set, *there does exist a complete set of* ***detectable*** *errors $\mathscr{E}_D$ for a given codespace $\mathcal{C}$*:

$$\mathscr{E}_D = \{E \text{ s.t. } \left\langle\psi\right| E \left|\phi\right\rangle = q(E) \left\langle\psi|\phi\right\rangle \ \forall \left|\psi\right\rangle, \left|\phi\right\rangle \in \mathcal{C}\}. \tag{11}$$

This difference comes from the definition of correctable and detectable errors: recall that an error set $\mathscr{E}$ (i.e. error channel) is *correctable* if $\forall E_a, E_b \in \mathscr{E} : \left\langle\psi\right| E_a^\dagger E_b \left|\phi\right\rangle = C_{ab} \left\langle\psi|\phi\right\rangle$, while at the same time it is *detectable* if $\forall E \in \mathscr{E} : \left\langle\psi\right| E \left|\phi\right\rangle = q(E) \left\langle\psi|\phi\right\rangle$. Notice how there can be many choices of E_a, E_b that equal to the same product $E_a^\dagger E_b$ (we could, for instance, easily insert a resolution of identity $E_a^\dagger U^\dagger U E_b$ for an arbitrary unitary U, to get a different correctable error set). At the same time, the choice for the set $\{E | E \in \mathscr{E}\}$ is fixed through $\left\langle\psi\right| E \left|\phi\right\rangle = q(E) \left\langle\psi|\phi\right\rangle$, as this fixes the eigenspace for E to be the set of all codewords.
Importantly, this implies the that for a given code $\mathcal{C}$, if $\mathscr{E}_C$ is some set of correctable errors, and $\mathscr{E}_D$ is the complete set of detectable errors for $\mathcal{C}$, then we can reformulate the KL conditions as:

$$\mathscr{E}_C^2 \subseteq \mathscr{E}_D. \tag{12}$$

Note that we write $\mathscr{E}_C^2$ instead of $\mathscr{E}_C^\dagger \mathscr{E}_C$, because we know for sure that if $E \in \mathscr{E}_C$, then $E^\dagger \in \mathscr{E}_C$, by just taking hermitian conjugate of the equation in the KL conditions.

Now, using this form of the KL conditions, we can formulate an important equivalence for quantum error-correcting codes. First, we define erasure errors:

Definition 1.6. *A single quqit erasure error is an error which maps all the q internal states of the quqit it is acting on to some local state $\left|s\right\rangle$ orthogonal to all the q internal states of the quqit. Then, by measuring the presence of a state $\left|s\right\rangle$ within our physical system, we are able to determine exactly which quqit got erased, gaining classical information about the position of the erasure. A multi-quqit erasure error is just a tensor product of erasure errors on different quqits.*

Then, using this definition, we can formulate the following theorem:

Theorem 1.4. *A QECC with distance d can correct $d-1$ erasure errors.*

Proof. See proof in the Appendix (A). ■

Reformulating this statement using the notion of code distance, we see that Theorem 1.4 trivially implies an important equivalence:

Corollary 1.2. *A QECC can correct arbitrary weight-t errors if and only if it can correct $2t$ erasure errors.*

2 Knill-Laflamme conditions for qudit PI codes

In this section, we use an important equivalence between erasure and deletion errors that is present in PI codes, extending the deletion channel construction for PI quantum codes given in [AAB24] from qubits to qudits. Using this, we then derive the necessary and sufficient conditions for arbitrary qudit PI quantum codes to be error-correcting.

2.1 Equivalence of erasure and deletion errors on PI codes

Definition 2.1. *A weight-t deletion (trace) map* Del_I *on an index set* $I = \{i_1, i_2, \ldots, i_t\}$ *with* $i_1 < i_2 < \cdots < i_t$ *is defined as a channel that traces out the quqits with indices from* I*:*

$$\mathrm{Del}_I(\rho) = \mathrm{Tr}_{i_1} \circ \mathrm{Tr}_{i_2} \circ \cdots \circ \mathrm{Tr}_{i_t}(\rho). \tag{13}$$

Note that a more standard notation for Del_I is Tr_I. We can now define the deletion channel:

Definition 2.2. *A t-deletion channel* Del_t^n *on n quqits is defined as a convex linear combination of all weight-t deletion maps* Del_I *with an index set of size t, i.e. with* $|I| = t$*:*

$$\mathrm{Del}_t^n(\rho) = \sum_{I:|I|=t} p(I)\mathrm{Del}_I(\rho) \tag{14}$$

where $p(I)$ *is a probability distribution.*

We usually say that t deletion errors happened when a t-deletion channel acts on a state, and that if a QECC can recover the state after a t-deletion channel, it can correct up to t deletion errors.

Next, we introduce an argument similar to the one made in [AAB24].

Theorem 2.1. *A PI QECC can correct arbitrary weight-t errors if and only if it can correct 2t deletion errors.*

Proof. Notice how for PI codes, arbitrary permutations of the index set leave the codewords invariant, since we defined the codewords to be invariant under arbitrary permutations of the physical quqits. Therefore, all weight-t deletion maps Del_I act the same on permutation invariant states ρ, irrespective of the choice for the index set I. Thus, when restricting the deletion map to the symmetric manifold, all the $\mathrm{Del}_I(\rho)$ in the convex sum are equal, so then $\mathrm{Del}_t^n(\rho)$ is equivalent to acting with $\mathrm{Del}_I(\rho)$ on some fixed index set I with $|I| = t$, so we can treat the general t-deletion channel as if we are deleting on known locations, which gives us classical information "for free". Therefore, when acting on PI codes, deletion errors are equivalent to erasure errors.

Now, importantly, as established earlier, a QECC can correct arbitrary weight-t errors iff it can correct $2t$ erasures. Since deletion and erasure errors are equivalent for PI codes, this concludes the proof. ■

This theorem makes the KL conditions much simpler to write for PI codes, if we use the Kraus set for the $2t$-deletion channel instead of weight-t Pauli matrices.

2.2 Kraus set for a quqit quantum deletion channel

Aydin, Alekseyev, and Barg [AAB24] define a Kraus set for an arbitrary t-deletion channel and use this to derive the KL conditions for qubit PI codes. We generalize this construction to qudit PI codes.

In order to derive the Kraus decomposition, we have to make a couple of auxiliary definitions. In spirit of [SO21], we want to define an operator with the following properties: for an index set $I = \{i_1, i_2, \ldots, i_t\} \subseteq \{1, \ldots, n\}$ of size t with $i_1 < i_2 < \cdots < i_t$, and a t-quqit computational basis state $\langle x|$ with $x = x_1 x_2 \ldots x_t \in \{0, 1, \ldots, q-1\}^t$, we have the operator $A^n_{I,x}$ acting on n quqits that has the following tensor product structure:

$$A^n_{I,x} = A_{I,1} \otimes A_{I,2} \otimes \cdots \otimes A_{I,n}, \text{ where each } A_{I,j} \text{ is}$$

$$A_{I,j} = \begin{cases} \langle x_k|, & j = i_k \in I; \\ \mathbb{1}_q, & j \notin I. \end{cases} \tag{15}$$

For example, $A_{I,i_3} = \langle x_3|$, and $A_{I,k} = \mathbb{1}_q$ if $k \notin I$. This description gives a good idea of what we actually want to define. To give a somewhat informal definition for $A^n_{I,x}$, we have:

Definition 2.3. *Let $I \subseteq \{1, \ldots, n\}$ and $x \in \{0, 1, \ldots, q-1\}^{|I|}$. We define $A^n_{I,x} : \mathbb{C}^{q^n} \to \mathbb{C}^{q^{n-|I|}}$ as the operator that projects quqits located at positions I to the standard basis state $\langle x|$:*

$$A^n_{I,x} := \langle x|_I \otimes \mathbb{1}_{\{1,\ldots,n\}\setminus I} \tag{16}$$

where terms in the tensor product need to be rearranged appropriately. In this particular instance, by $\mathbb{1}_{\{1,\ldots,n\}\setminus I}$ we mean an identity of size $q * (n - |I|)$ that is acting on appropriately rearranged positions.

Similarly to the qubit case [SO21], for quqits this operator also has a decomposition:

Lemma 2.1.

$$A^n_{I,x} = A^{n-t+1}_{i_1,x_1} \circ A^{n-t+2}_{i_2,x_2} \circ \cdots \circ A^n_{i_t,x_t} \tag{17}$$

Proof. Straightforward computation, see proof in Appendix (A). ■

Now we observe a Kraus decomposition:

Lemma 2.2. *We have the following Kraus decomposition for the t-deletion channel:*

$$\mathrm{Del}^n_t(\rho) = \sum_{x \in \{0,\ldots,q-1\}^t;\ I:|I|=t} p(I) A^n_{I,x} \rho A^{n\dagger}_{I,x} \tag{18}$$

with $\sqrt{p(I)}A^n_{I,x}$ – the Kraus operators.

Proof. A simple calculation gives:

$$\mathrm{Del}_I(\rho) = \mathrm{Tr}_{i_1} \circ \mathrm{Tr}_{i_2} \circ \cdots \circ \mathrm{Tr}_{i_t}(\rho) = \sum_{x \in \{0,\ldots,q-1\}^t} A^{n-t+1}_{i_1,x_1} \circ \cdots \circ A^n_{i_t,x_t} \rho A^{n\dagger}_{i_t,x_t} \circ \cdots \circ A^{n-t+1\dagger}_{i_1,|x_1\rangle} =$$

$$= \sum_{x \in \{0,\ldots,q-1\}^t} A^n_{I,x} \rho A^{n\dagger}_{I,x},$$

which, when introducing a convex combination over Del_I, easily implies the statement of the lemma for the deletion channel. ■

Although $\sqrt{p(I)}A^n_{I,x}$ are the actual Kraus operators, we will often refer to the normalized $A^n_{I,x}$ as the Kraus operators.

Definition 2.4. *A k-type deletion operator applied to the i-th quqit in an n-quqit system is the operator:*

$$\chi_{i,k} = A^n_{i,k} \tag{19}$$

So that the action of $\chi_{i,k}$ is:

$$\chi_{i,k} |x\rangle = \langle k|x_i\rangle |\cancel{x_i}\rangle$$

Note that the index n is unnecessary in $\chi_{i,k}$ since the index set is always of size 1.

Lemma 2.3. *The action of a k-deletion $\chi_{i,k}$ on a Dicke state $|D^n_\lambda\rangle$ is:*

$$\chi_{i,k} |D^n_\lambda\rangle = \sqrt{\frac{\binom{n-1}{\lambda-e_k}}{\binom{n}{\lambda}}} |D^{n-1}_{\lambda-e_k}\rangle\,, \tag{20}$$

where $\lambda - e_k = \{\lambda_0, \ldots, \lambda_k - 1, \ldots, \lambda_{q-1}\}$ for e_k the k-th standard basis vector.

Proof.

$$\chi_{i,k} |D^n_\lambda\rangle = \frac{1}{\sqrt{\binom{n}{\lambda}}} \sum_{x=(x_1,\ldots,x_n)\in X^n_\lambda} \langle k|x_i\rangle |x_1\rangle \otimes \cdots \otimes |x_{i-1}\rangle \otimes |x_{i+1}\rangle \otimes \cdots \otimes |x_n\rangle =$$

$$= \frac{1}{\sqrt{\binom{n}{\lambda}}} \sum_{x=(x_1,\ldots,x_{n-1})\in X^{n-1}_{\lambda-e_k}} |x_1\rangle \otimes |x_2\rangle \otimes \cdots \otimes |x_{n-1}\rangle = \frac{\sqrt{\binom{n-1}{\lambda-e_k}}}{\sqrt{\binom{n}{\lambda}}} |D^{n-1}_{\lambda-e_k}\rangle$$

where, by convention, we say that the Dicke state is zero if the partition denoting this Dicke state contains any negative entries. ■

Note that, evidently, when acting with any k-type deletion on a Dicke state, the index i in $\chi_{i,k}$ is irrelevant. Since i denotes a particular quqit which is being deleted and Dicke states are permutation-symmetric, the outcome of a k-deletion is the same for all indices i.
Since we will be mostly discussing errors happening on PI and Dicke states, it is convenient to remove the indexation in i when discussing the k-type deletions, and denote them as χ_k instead.

Lemma 2.4. *For a Dicke state $|D^n_\lambda\rangle$, $\forall a \in \{1, \ldots, n\}$:*

$$(\chi_k)^a |D^n_\lambda\rangle := \underbrace{\chi_k \circ \cdots \circ \chi_k}_{a} |D^n_\lambda\rangle = \sqrt{\frac{\binom{n-a}{\{\lambda_0,\ldots,\lambda_k-a,\ldots,\lambda_{q-1}\}}}{\binom{n}{\lambda}}} |D^{n-a}_{\{\lambda_0,\ldots,\lambda_k-a,\ldots,\lambda_{q-1}\}}\rangle \tag{21}$$

Proof. Follows easily from Lemma 2.3 by induction. ■

Lemma 2.5. *Deletion operators commute when acting on Dicke states.*

Proof. Consider a k-type deletion χ_k and a l-type deletion χ_l, where without loss of generality we have $k < l$ then:

$$\chi_k \circ \chi_l \left|D^n_\lambda\right\rangle = \sqrt{\frac{\binom{n-2}{\lambda-e_l-e_k}}{\binom{n}{\lambda}}} D^{n-2}_{\lambda-e_l-e_k} = \sqrt{\frac{\binom{n-2}{\lambda-e_k-e_l}}{\binom{n}{\lambda}}} D^{n-2}_{\lambda-e_k-e_l} = \chi_l \circ \chi_k \left|D^n_\lambda\right\rangle. \qquad \blacksquare$$

From the lemmas, it is clear that a general Kraus operator $A^n_{I,\langle x|}$ decomposes into a product of k-type deletions (for different k) when acting on Dicke states. For a quqit t-deletion channel acting on states with local dimension q, this decomposition has the following form:

$$(\chi_{q-1})^{\mu_{q-1}} \circ (\chi_{q-2})^{\mu_{q-2}} \circ \cdots \circ (\chi_1)^{\mu_1} \circ (\chi_0)^{\mu_0} \;\Big|\; \mu \overset{q}{\vdash} t \tag{22}$$

Where from now on we will label:

$$E^q_\mu = (\chi_{q-1})^{\mu_{q-1}} \circ \cdots \circ (\chi_0)^{\mu_0} \tag{23}$$

And further on we call E^q_μ *deletion errors*. Therefore, when acting on Dicke states we can write the Kraus set for a quqit t-deletion channel acting on states with local dimension q as:

$$\mathscr{E}^q_t = \{E^q_\mu \mid \mu \overset{q}{\vdash} t\} \tag{24}$$

Lemma 2.6. *For a Dicke state $\left|D^n_\lambda\right\rangle$ of local dimension q, and for $\lambda \overset{q}{\vdash} n, \mu \overset{q}{\vdash} t, \lambda - \mu \geq 0$:*

$$E^q_\mu \left|D^n_\lambda\right\rangle = \sqrt{\frac{\binom{n-t}{\lambda-\mu}}{\binom{n}{\lambda}}} \left|D^{n-t}_{\lambda-\mu}\right\rangle \tag{25}$$

where $\lambda - \mu$ denotes the entry-wise difference of λ and μ, and $\lambda - \mu \geq 0$ means that each entry in the resulting vector is non-negative. The action of deletion errors on Dicke states gives zero if $\lambda - \mu \geq 0$ doesn't hold.

Proof. Straightforward, see Appendix (A). $\blacksquare$

2.3 Necessary and sufficient error-correction conditions

Using the previous lemmas, it is now straightforward to derive the KL conditions for PI codes to correct arbitrary weight-t errors. As a result of Theorem 2.1, we do this using the Kraus operators for the $2t$ deletion channel, i.e. we use deletion errors from the Kraus set in equation (24). We use the codewords defined as in Definition 1.2.

Theorem 2.2. *The KL conditions for qudit PI codes encoding a single logical qudit of local dimension d into n physical quqits of local dimension q are:*

$$\textit{C1:}\ \sum_{\lambda \overset{q}{\vdash} n} \bar{\alpha}_{i,\lambda}\alpha_{j,\lambda} = \delta_{i,j}, \quad \forall i,j \in \{0,\ldots,d-1\} \tag{26}$$

$$\textit{C2:}\ \sum_{\lambda \overset{q}{\vdash} n} \bar{\alpha}_{i,\lambda}\alpha_{j,\lambda-\mu+\nu} \frac{\binom{n-2t}{\lambda-\mu}}{\sqrt{\binom{n}{\lambda}\binom{n}{\lambda-\mu+\nu}}} = 0, \quad \forall i \neq j \in \{0,\ldots,d-1\},\ \forall \mu,\nu \overset{q}{\vdash} 2t, n \geq 2t \tag{27}$$

$$\textit{C3:}\ \sum_{\lambda \overset{q}{\vdash} n} (\bar{\alpha}_{i,\lambda}\alpha_{i,\lambda-\mu+\nu} - \bar{\alpha}_{j,\lambda}\alpha_{j,\lambda-\mu+\nu}) \frac{\binom{n-2t}{\lambda-\mu}}{\sqrt{\binom{n}{\lambda}\binom{n}{\lambda-\mu+\nu}}} = 0, \quad \forall i,j \in \{0,\ldots,d-1\}, \forall \mu,\nu \overset{q}{\vdash} 2t, n \geq 2t \tag{28}$$

where we define $\frac{\binom{n-2t}{\lambda-\mu}}{\sqrt{\binom{n}{\lambda}\binom{n}{\lambda-\mu+\nu}}}$ *to be zero if any of the multinomial coefficients contain factorials of negative numbers (so that it is zero when* $\lambda-\mu$ *contains negative entries).*

Proof.
C1. This is just the orthonormality condition:

$$\langle c_i|c_j\rangle = \sum_{\lambda,\lambda' \overset{q}{\vdash} n} \bar{\alpha}_{i,\lambda}\alpha_{j,\lambda'} \underbrace{\langle D^n_\lambda | D^n_{\lambda'}\rangle}_{\delta_{\lambda\lambda'}} = \sum_{\lambda \overset{q}{\vdash} n} \bar{\alpha}_{i,\lambda}\alpha_{j,\lambda} = \delta_{i,j}, \;\; \forall i,j \in \{0,\ldots,d-1\}$$

C2. This condition we derive from the Knill-Laflamme condition $\langle c_i|\, E^{q\dagger}_\mu E^q_\nu\, |c_j\rangle = 0$ for $i \neq j$, where we have $\mu,\nu \overset{q}{\vdash} 2t$ since we are working with the $2t$-deletion channel:

$$\langle c_i|\, E^{q\dagger}_\mu E^q_\nu\, |c_j\rangle = 0 \Rightarrow \sum_{\lambda,\lambda' \overset{q}{\vdash} n} \bar{\alpha}_{i,\lambda}\alpha_{j,\lambda'} \langle D^n_\lambda | E^{q\dagger}_\mu E^q_\nu | D^n_{\lambda'}\rangle = \sum_{\lambda,\lambda' \overset{q}{\vdash} n} \bar{\alpha}_{i,\lambda}\alpha_{j,\lambda'} \sqrt{\frac{\binom{n-2t}{\lambda-\mu}\binom{n-2t}{\lambda'-\nu}}{\binom{n}{\lambda}\binom{n}{\lambda'}}} \underbrace{\langle D^{n-2t}_{\lambda-\mu} | D^{n-2t}_{\lambda'-\nu}\rangle}_{\delta_{\lambda-\mu+\nu,\lambda'}} =$$

$$= \sum_{\lambda \overset{q}{\vdash} n} \bar{\alpha}_{i,\lambda}\alpha_{j,\lambda-\mu+\nu} \frac{\binom{n-2t}{\lambda-\mu}}{\sqrt{\binom{n}{\lambda}\binom{n}{\lambda-\mu+\nu}}} = 0, \forall i \neq j | i,j \in \{0,\ldots,d-1\}, \; \forall \mu,\nu \overset{q}{\vdash} 2t, n \geq 2t$$

Note that only entries with $\lambda-\mu \geq 0$ remain non-zero, since Dicke states which don't satisfy this get deleted (i.e. turn to zero) by the deletion errors.
C3. This condition we derive from the Knill-Laflamme condition $\langle c_i|\, E^{q\dagger}_\mu E^q_\nu\, |c_i\rangle = \langle c_j|\, E^{q\dagger}_\mu E^q_\nu\, |c_j\rangle$, where again $\mu,\nu \overset{q}{\vdash} 2t$:

$$\langle c_i|\, E^{q\dagger}_\mu E^q_\nu\, |c_i\rangle - \langle c_j|\, E^{q\dagger}_\mu E^q_\nu\, |c_j\rangle = 0 \Rightarrow \sum_{\lambda \overset{q}{\vdash} n} (\bar{\alpha}_{i,\lambda}\alpha_{i,\lambda-\mu+\nu} - \bar{\alpha}_{j,\lambda}\alpha_{j,\lambda-\mu+\nu}) \frac{\binom{n-2t}{\lambda-\mu}}{\sqrt{\binom{n}{\lambda}\binom{n}{\lambda-\mu+\nu}}} = 0,$$

$\forall i,j \in \{0,\ldots,d-1\}, \; \forall \mu,\nu \overset{q}{\vdash} 2t, n \geq 2t.$ ∎

3 Numerical study of qubit PI code solutions

Before constructing qudit PI codes in later sections, we have to understand the structure and properties of qubit PI codes first. In this section, we report on a comprehensive numerical study of qubit PI code solutions that we conducted, conjecture certain PI code properties based on numerical evidence, and provide numerical results and plots, as well as code references, for future study. We also give a brief introduction into a numerical method that we have used throughout our study, called Homotopy Continuation.

3.1 Homotopy continuation

In this subsection, we provide a brief introduction to a numerical algebraic geometry method known as Homotopy Continuation. This introduction is based on the crash course on Homotopy Continuation for the HomotopyContinuation Julia package [BT17; BT], which we used in our numerical study.

Nonlinear systems, in particular systems of polynomial equations, in the most general case have no universal analytical solution. Instead, numerical methods can be used to provide practical solutions. One such powerful approach is the *homotopy continuation method*, which solves a complex system by continuously deforming a simpler one whose solutions are already known.

Basic idea

Suppose we want to find all complex solutions ($x_i \in \mathbb{C}$) to a system of m polynomial equations with n variables $x = (x_1, x_2, \ldots, x_n)$:

$$f(x) = \begin{cases} f_1(x_1, \ldots, x_n) = 0 \\ \vdots \\ f_m(x_1, \ldots, x_n) = 0 \end{cases}$$

The key idea in homotopy continuation is to construct a *homotopy* $H(x, t)$ (a continuous path deformation) from our target system $f(x) = 0$ to an easier system $g(x) = 0$ that we know solutions for, and that has at least as many solutions as $f(x) = 0$. This is done via a "time" parameter $t \in [0, 1]$, and $H(x, t)$ must satisfy:

- $H(x, 1) = g(x)$;
- $H(x, 0) = f(x)$;
- $\forall t \in (0, 1]$, $H(x, t)$ has the same number of isolated zeros in the variables $x = (x_1, \ldots, x_n)$ (29)

The numerical method traces the paths of solutions of $H(x, t) = 0$ as t moves from 1 to 0. Since $g(x) = 0$ has at least as many solutions as $f(x) = 0$ and we choose the homotopy to be continuous and to satisfy the conditions in (29), each solution of $g(x) = 0$ gives rise to a path that leads to a solution of $f(x) = 0$. After following all the paths, we find all the solutions of

$f(x) = 0$. Note that there are infinitely many possible easy systems $g(x) = 0$ that we can choose, and infinitely many homotopies, yet there are certain standard choices we can always make that guarantee a solution, as discussed below with the *total degree homotopy*.

Path tracking, and a simple example

Consider the problem of solving a single-variable polynomial equation:

$$f(x) = x^2 - 5 = 0$$

Its solutions are $x = \pm\sqrt{5}$, but imagine we don't know that. We construct a homotopy using a simple start system, say:

$$g(x) = x^2 - 1 = 0 \quad \Rightarrow \quad \text{solutions: } x = \pm 1$$

Define the homotopy:

$$H(x, t) = t(x^2 - 1) + (1 - t)(x^2 - 5) = x^2 + 4t - 5$$

As t decreases from 1 to 0, the solutions of $H(x, t) = 0$ trace continuous paths from ± 1 to $\pm\sqrt{5}$. The homotopy continuation method follows these paths numerically. The most popular way of doing this is through the so-called *predictor-corrector method*: to find the path $x(t)$ we differentiate $H(x, t)$ with respect to t; since we know the initial solution $x(1)$ for $H(x, 1) = g(x) = 0$, now the path is governed by an ODE with some boundary condition, which we solve by a suitable numerical method (predictor), such as the Runge-Kutta method (RK4) [AP98], and then correct the prediction via Newton's method (corrector) [AP98], see Figure 1.

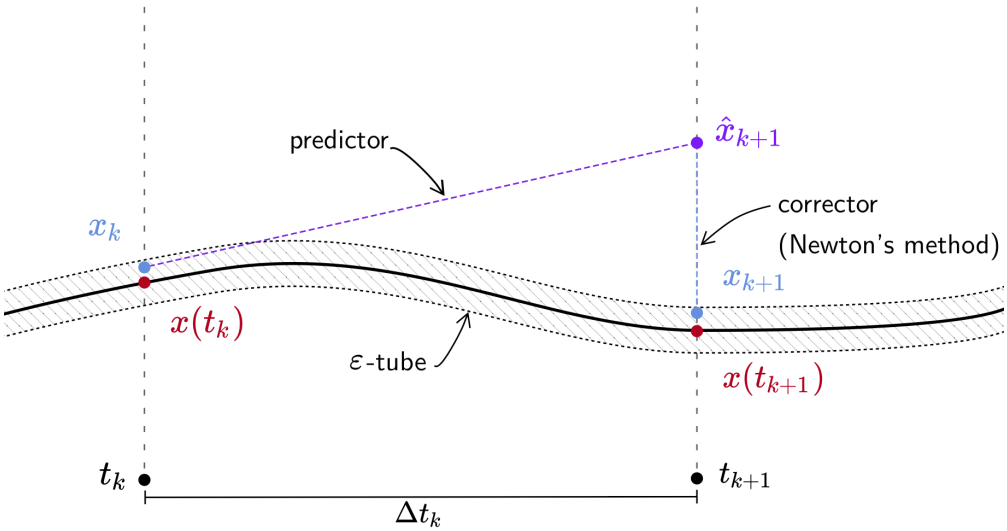


Figure 1: Predictor-corrector method for path tracking in homotopy continuation, showcased graphically. Image credit: [BT].

Choosing start systems and homotopies

Homotopy continuation is especially useful in multivariate systems of polynomial equations, where analytical solutions are rare. For instance, consider a system of two equations with two variables:

$$f(x,y) = \begin{cases} f_1(x,y) = x^2y^2 + x^2 + y^2 - 1 = 0 \\ f_2(x,y) = x^3 - y + 1 = 0 \end{cases} \tag{30}$$

The polynomial f_1 has degree 4, and the polynomial f_2 has degree 3. According to the Bezout's theorem [Ful74], such a system has *at most* $4 \cdot 3 = 12$ isolated solutions (isolated zeros). The standard choice for a start system $g(x,y)$ in this case would be:

$$g(x,y) = \begin{cases} g_1(x,y) = x^4 - 1 = 0 \\ g_2(x,y) = y^3 - 1 = 0 \end{cases} \tag{31}$$

The 12 solutions of $g(x,y)$ are much easier to compute:

$$(x,y) = (\omega_4^{k_1}, \omega_3^{k_2}),\ k_1 \in \{0,1,2,3\},\ k_2 \in \{0,1,2\}$$

where $\omega_q = e^{2\pi \mathrm{i}/q}$. We then construct the homotopy:

$$H(x,y,t) = tg(x,y) + (1-t)f(x,y) \tag{32}$$

and numerically trace the paths of each solution from $t = 1$ to $t = 0$. This approach is an example of a *total degree homotopy*.

In general, the number of isolated solutions of a polynomial system in $\mathbb{C}^n$ is bounded by Bézout's theorem: for a system of n polynomial equations in n variables, where each polynomial f_i has degree d_i for $i \in \{1, \ldots, n\}$, the maximum number of isolated solutions is

$$D = d_1 \cdot d_2 \cdots d_n.$$

We call systems of equations *square systems* if they have the same number of variables and equations. The homotopy choice to the above example (30) of a square system of 2 equations in 2 variables easily generalizes to an arbitrary square polynomial system of n equations in n variables. Specifically: we choose g to have exactly D solutions (maximum allowed by Bezout's theorem) which are all tuples of roots of unity, with each equation having the degree $d_1, d_2, \ldots, d_n$ respectively as in (31), and we construct the homotopy like in equation (32). This is called the *total degree homotopy*.

We call a homotopy *optimal* if $H(x,1)$ has the same number of solutions as the target $H(x,0)$. And while the total degree homotopy may not always be optimal, it always exists for square polynomial systems [BT17].

Homotopy continuation can also help to solve overdetermined systems (systems with more equations than variables), and underdetermined systems (systems with more variables than equations). For the former, the standard approach would be to artificially add solutions in order to make the overdetermined system into a square system, and then remove them (by substituting back the original variables into the artificial ones); for the latter, the standard

approach would be to find a suitable easy underdetermined start system that also has infinitely many solutions, by removing constraints from a square system, and deform these continuously to the target system via a suitable homotopy. Note that unlike for square and overdetermined systems, underdetermined systems don't guarantee that the family of solutions found is the full infinite family of solutions, so this should be used more as a "find instance" type of numerical methods.

Advantages

Homotopy continuation has several advantages over other numerical methods (for example over zero search through gradient descent):

- It can compute *all* isolated complex solutions, since in Homotopy Continuation all of the solutions of the start system are homotopic to all of the solutions of the target system, with high precision (via some precision parameter of solving ODE's numerically) tracking different paths can numerically find all solutions of the target system if the system is not underdetermined. Compare this to Gradient Descent zero search, which finds different solutions randomly, by initializing random starting points, with no precision parameter or other guarantees of finding all solutions.
- It is highly parallelizable, as each path can be tracked independently.
- It applies to square (same number of equations and variables), overdetermined (more equations), and partially to underdetermined (more variables) systems.

3.2 Codeword symmetries, scaling, and other properties of minimal PI codes

In this section we report on the results of our numerical study of *minimal* qubit PI codes (see Definition 3.1). Based on numerical evidence, we conjecture a block length (see Section 1.1 for definition) scaling $\mathbf{n(t) = n_{min}(t) = 3t^2 + 3t + 1}$ with the error weight t, as well as certain codeword symmetries that *all* minimal qubit PI codes must satisfy.

Block length scaling

First, let us write down the special case of the necessary and sufficient error-correction condition (i.e. KL conditions) for qubit PI codes. That is, we take our results from Theorem 2.2 and specialize to the case of $q = d = 2$, to get KL equivalent conditions to the ones in [AAB24]:

$$\text{C1: } \sum_{j=0}^{n} \bar{\alpha}_j \beta_j = 0, \;\; \sum_{j=0}^{n} |\alpha_j|^2 = \sum_{j=0}^{n} |\beta_j|^2 = 1, \tag{33}$$

$$\text{C2: } \sum_{j=0}^{n} \bar{\alpha}_j \beta_{j-a+b} \frac{\binom{n-2t}{j-a}}{\sqrt{\binom{n}{j}\binom{n}{j-a+b}}} = 0, \;\; \forall a, b \in \mathbb{N},\ 0 \leq a, b \leq 2t, \tag{34}$$

$$\text{C3: } \sum_{j=0}^{n} (\bar{\alpha}_j \alpha_{j-a+b} - \bar{\beta}_j \beta_{j-a+b}) \frac{\binom{n-2t}{j-a}}{\sqrt{\binom{n}{j}\binom{n}{j-a+b}}} = 0, \;\; \forall a, b \in \mathbb{N},\ 0 \leq a, b \leq 2t. \tag{35}$$

where n is the number of physical qubits we use to encode, t is the number of arbitrary errors we want to correct (which for PI codes means $2t$ deletion errors), and comparing with Theorem 2.2, for simplicity we denoted the coefficients of the 0-th codeword as $\alpha_j := \alpha_{0,\{n-j,j\}}$, and the coefficients of the 1-st codeword as $\beta_j := \alpha_{1,\{n-j,j\}}$. We also use $a := \mu_1$ for $\mu = (\mu_1, \mu_2) = (2t - a, a)$, and similarly $b := \nu_1$ for $\nu = (\nu_1, \nu_2) = (2t - b, b)$. Recall that we defined binomial coefficients to be equal to zero if any of the entries of the binomial coefficient is negative.

We study numerical solutions to the KL equations (33) for qubit PI codes with complex coefficients, and from our numerical results, for a fixed weight t of errors we want to correct we numerically find the lowest number n of physical qubits such that solutions exist. Recall from Section 1.1 that we refer to the number of physical encoding qubits of the code as the *block length*. We say that a block length is *minimal* when for a fixed number of errors (i.e. error weight) t we have that for any length below the minimal block length solutions to KL equations (33) don't exist.

Definition 3.1. *For PI codes encoding n physical qubits into one logical qubit and correcting arbitrary errors of weight t, we say that the block length n is* ***minimal*** *if there exist no PI codes of block length smaller than n correcting t errors. We denote the minimal block length for a given number of errors t as $n_{min}(t)$. We call the subfamily of PI codes that for any $t \in \mathbb{N}$ have block length $n_{min}(t)$* ***minimal PI codes****.*

In other words, our first task is as follows:

Task 3.1. *For a fixed error weight t, to determine the minimal block length scaling $n_{min}(t)$ for arbitrary PI codes with complex coefficients, we numerically look for the instances of n when solutions to the qubit PI KL conditions* (33) *exist.*

Method 1. Our primary method for solving this task utilizes the numerical method introcuded in Section 3.1, the *Homotopy Continuation*. After fixing n, t, we simply input the KL equations (33) as a polynomial system, with codeword coefficients α_j, β_j as its variables, $j \in [n + 1]$. If

for a pair (n, t) Homotopy Continuation doesn't find any solutions, they don't exist for this pair. For every value t we try multiple values of n until we find the first (and smallest) one for which a solution exists. For every next iteration of t, we choose the starting value for n as $n_{start} = n_{min}(t-1)$. See Table 1 for numerical results.

t =	1	2	3	4	5
n =	7	19	37	61	91

Table 1: First five numerical values, found using Homotopy Continuation, for the smallest block length n for a given t such that solutions to KL equations (33) exist.

It is clear from system (33) that for any fixed n, t there are $2n+2$ variables α_j, β_j, $j \in [n+1]$. Naïvely, we might think that there are $2(2t+1)^2+3$ equations in the system (33) simply by counting, therefore for the first $n_{min}(1) = 7$ found numerically we should then expect an overdetermined system of 21 equations with 16 variables. Yet Homotopy Continuation finds infinitely many solutions for $(n, t) = (7, 1)$, while for any smaller values of n it finds no solutions. Thus the system must be underdetermined to have infinitely many solutions, so we conclude that some of the equations in the system are **algebraically dependent** (meaning we can recover some of the polynomial equations in the system using algebraic combinations of other polynomial equations). Numerically, we observe the following:

Conjecture 3.1. *For any fixed error weight t and block length $n = n_{min}(t)$ the family of minimal qubit PI quantum error-correcting codes with parameters (n, t) is **infinite**. Moreover, for any $(n_{min}(t), t)$ it suffices to fix $t+1$ variables in the first codeword coefficients or $t+1$ variables in the second codeword coefficients to make the number of solutions to the KL equations with these variables fixed **finite**.*

For example: in the case of $t = 1$, $n = n_{min}(1) = 7$ if we fix any two α_i, α_j **or** any two β_i, β_j for different $i \neq j \in \{0, 1, \ldots, 7\}$, the system of equations in the remaining variables stops being underdetermined, and Homotopy Continuation either finds a finite set of solutions, or no solutions.

Method 2. As a secondary method, to solve this task, for fixed n, t we consider a vector with its entries being the differences $(LHS - RHS)$ between the left and right hand side of the KL equations. We then define the cost function as the norm of this vector squared, and treating this cost as a function of the codeword coefficients α_j, β_j, $j \in [n+1]$, we look for the minimum of this cost function using *Gradient Descent*. Whenever for a pair (n, t) the minimum of the cost function is sufficiently close to zero (in our case we set the threshold to $\epsilon \simeq 10^{-15}$) and the fraction between the value of the cost function at $cost[n_{min}(t) - 1]$ and $cost[n_{min}(t)]$ is sufficiently big (we set the threshold to $\theta \simeq 10^5$), we say that a solution exists. For every value t we try multiple values of n until we find the first (and smallest) one for which a solution exists. For every next iteration of t, we choose the starting value for n as $n_{start} = n_{min}(t-1)$. See Table 2 for numerical results.

t =	1	2	3	4	5
n =	7	19	37	61	91
iters =	100000				
reps =	256				

Table 2: First five numerical values, found using Gradient Descent, for the smallest block length n for a given t such that solutions to KL equations (33) exist. The threshold we set for existence of solutions, i.e. for the cost function to be considered zero, is $\epsilon \simeq 10^{-15}$ throughout. The threshold we set for the jump in the cost function between the minimum and the value before, i.e. the threshold for the fraction $cost[n_{min}(t) - 1]/cost[n_{min}(t)]$, is $\theta \simeq 10^5$. "Iters" is the number of iterations for the gradient descent, i.e. number of steps it takes for reaching the minimum. "Reps" is the number of different starting points we choose when looking for the minimum. Clearly these numerical results from the second Method coincide with the results of the first Method.

There exist explicit (i.e. with analytically defined coefficients) qubit PI codes with block length scaling of $n = O(t^2)$ [AAB24; Ouy14]. For instance: qubit PI codes by Aydin et. al. [AAB24] have block length scaling $n(t) = 4t^2 + 2t + 1$. Therefore the full family of minimal qubit PI codes cannot have a block length scaling worse than quadratic in t, moreover the scaling must be at least as good as for Aydin qubit PI codes. Generally, for t the desired error weight, good QECC have code distance $d_C = 2t + 1$ scaling as a root of block length n: $d_C = O(n^{1/p}), p \in \mathbb{N}$. So we also expect $n(t)$ to scale polynomially with t for minimal qubit PI codes. Now, from the numerical results of both of our models, we can already build a function $n_{min}(t)$ for minimal qubit PI codes, since we actually can reconstruct a polynomial function with scaling $\leq O(t^2)$ by at most three points $n(t)$ that perfectly passes through all the points $n(t)$ computed numerically (Tables 1, 2), and has polynomial scaling as expected:

Conjecture 3.2. *Minimal qubit PI codes have block length scaling $n_{min}(t) = 3t^2 + 3t + 1$. That is: all qubit PI QEC codes have an upper bound on their code distance, $d_C \leq \sqrt{12n - 3}/3$, that is saturated in minimal PI codes.*

This, in particular, implies that there are **no** good qLDPC [BE21] qubit PI codes, which is quite a strong statement. We also notice another interesting detail: there exist qubit PI codes with real coefficients that also have block length scaling of $3t^2 + 3t + 1$, for example a family of qubit PI codes by Ruskai et. al. [PR04], that we discuss later on in Section 6. Therefore we can make an important conclusion: *qubit PI codes with **complex** coefficients have no benefits in terms of code distance scaling compared to qubit PI codes with **real** coefficients.*

Codeword symmetries and continuity

As mentioned earlier as a conclusion to Conjecture 3.2, minimal qubit PI codes with real coefficients are as good as minimal qubit PI codes with complex coefficients in terms of their distance (or block length) scaling. Due to this, all numerical results that follow until the rest of this section we obtained for real minimal qubit PI codes, since they are also faster to obtain numerically.

For all real minimal qubit PI code solutions found numerically, we observe the following symmetry in the codeword coefficients (33):

$$\begin{gathered}\beta_i = (-1)^i \alpha_{n-i} \\ \text{or} \\ \beta_i = (-1)^{i+1} \alpha_{n-i},\end{gathered} \tag{36}$$

where $n = n_{min}(t)$ is the block length for an error weight t, and $i \in [n+1]$. We call the $|\beta_i| = |\alpha_{n-i}|$ symmetry "*mirror symmetry*", and the $\text{sgn}(\beta_i) = (-1)^{i(+1)}\,\text{sgn}(\alpha_{n-i})$ "*phase-flip symmetry*". Note that the Aydin code family [AAB24], the Ruskai code family [PR04], and the Ouyang code family [Ouy14] all have these symmetries as well. For minimal qubit PI code solutions with real coefficients, this symmetry is showcased in Figures 2, 3, 4, 5.

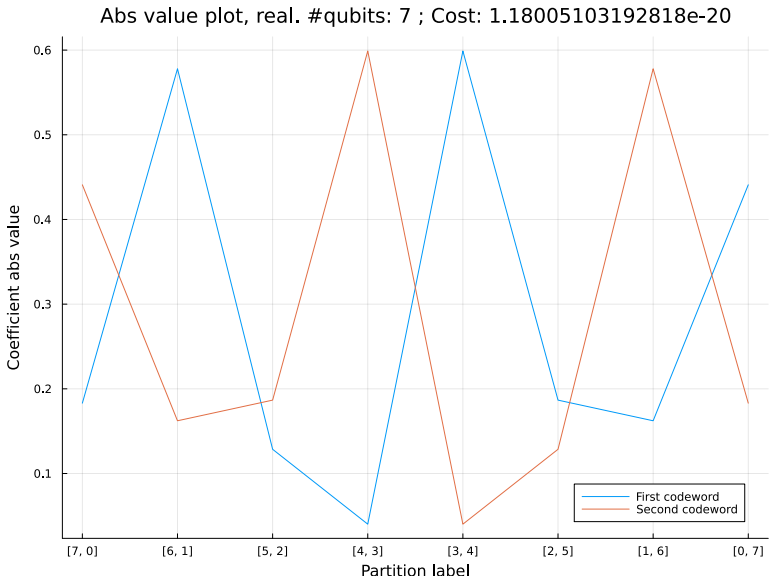


Figure 2: Plot of the absolute value of the codeword coefficients of one of the solutions in the real minimal qubit PI code family for n=7, t=1. It is easy to see that the absolute values of the codeword coefficients have mirror symmetry, with respect to reversing the partition labels.

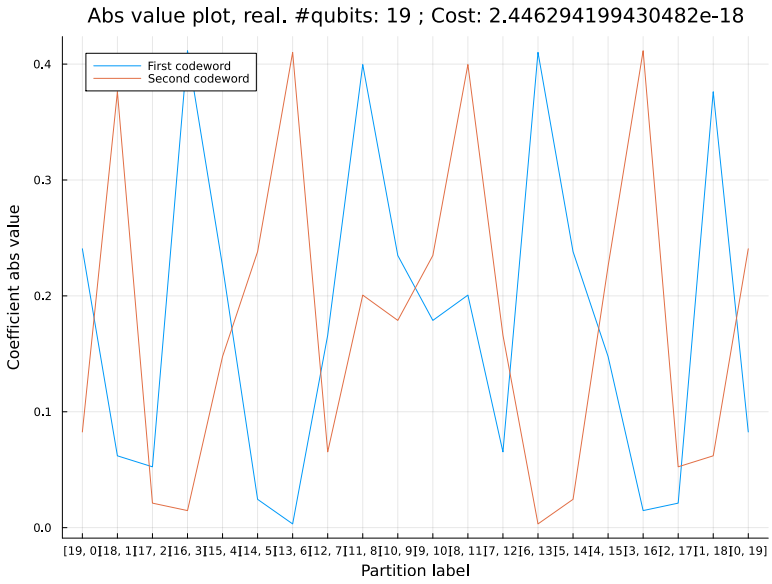


Figure 3: Plot of the absolute value of the codeword coefficients of one of the solutions in the real minimal qubit PI code family for n=19, t=2. It is evident from the plot that the solution has mirror symmetry.

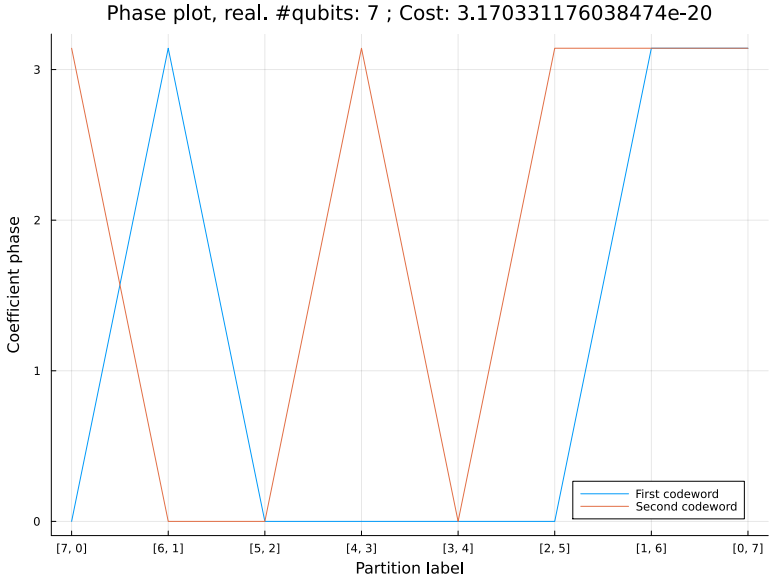


Figure 4: Plot of the phase of the codeword coefficients of one of the solutions in the real minimal qubit PI code family for n=7, t=1. The phase is in radians. It is easy to see that compared to codeword 1, the phase of the coefficients in codeword 2, reflected with respect to reversing the partition labels, is the same as that of the coefficients in codeword 1, except the sign flips every step. This showcases the phase-flip symmetry.

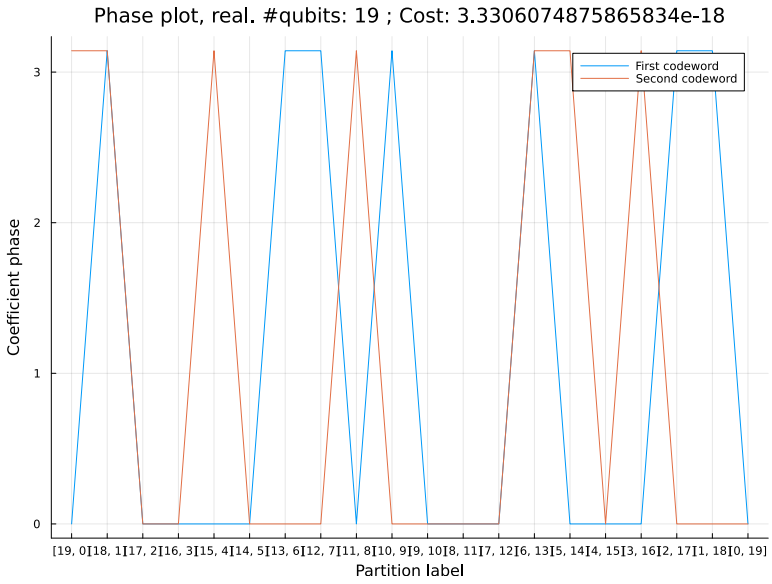


Figure 5: Plot of the phase of the codeword coefficients of one of the solutions in the real minimal qubit PI code family for n=19, t=2. The phase is in radians. Again, similar to the previousFigure, this plot showcases the phase-flip symmetry.

Based on this numerical evidence, we make the following conjecture:

Conjecture 3.3. *All minimal qubit PI codes with real coefficients exhibit both the* ***mirror*** *and the* ***phase-flip*** *symmetries* (36) *in their codewords.*

Note that this implies that minimal PI codes with real coefficients are actually defined by just the $n+1$ coefficients of the first codeword, i.e. they have $n+1$ variables, since due to the symmetries we can recover the values of the second codeword uniquely from the first one.

We also further explore the implications of Conjecture 3.1. Specifically: by fixing $t+1$ variables in the first codeword of a real minimal PI code we always find finitely many solutions, which means there are no more degrees of freedom (except for some possible discrete symmetries). By using either Homotopy Continuation or Gradient Descent, for a fixed $n_{min}(t), t$ we can choose some set of $t+1$ variables to fix, and vary the values we fix this set to in order to see which values give a finite set of solutions, and which values give the empty set. In this way we can construct the boundaries for each variable of the codeword, and approximately reconstruct how the full space of real minimal PI code solutions looks like for fixed n, t. Let us explain this on an example.
Consider a real minimal PI code with block length and error weight $(n, t) = (7, 1)$. We know this code will have mirror and phase-flip symmetry in its codewords (36), and that it is defined by 8 coefficients of its first codeword: α_i, $i \in \{0, \ldots, 7\}$. Let's say we want to fix $t+1 = 2$ coefficients, say α_0 and α_6, both to 0; does the system of KL equations (33) associated to this minimal PI code still have solutions? That is, can such a code be error-correcting? Turns out that for this particular choice of a variable fix, solutions **do exist**, as we numerically

verify through Homotopy Continuation (it finds finitely many solutions). What if we shift $(\alpha_0, \alpha_6) = (0, 0)$ by 0.02 in either direction? If we keep going at some point we will hit a boundary, after which shifting this fixed value further will lead to no solutions. In this way we can visualize the projection of the full space of solutions on the coordinate plane (α_0, α_6). This is shown on Figure 6. **Importantly:** this plot (as well as other similar plots) were built using Julia, which has 1-based indexing in arrays instead of the usual 0-based indexing, so it was convenient numerically to shift the index of all our coefficients by 1 (i.e. α_0, α_6 would turn into α_1, α_7, and $i \in \{1, \ldots, 8\}$ instead of $\{0, \ldots, 7\}$).

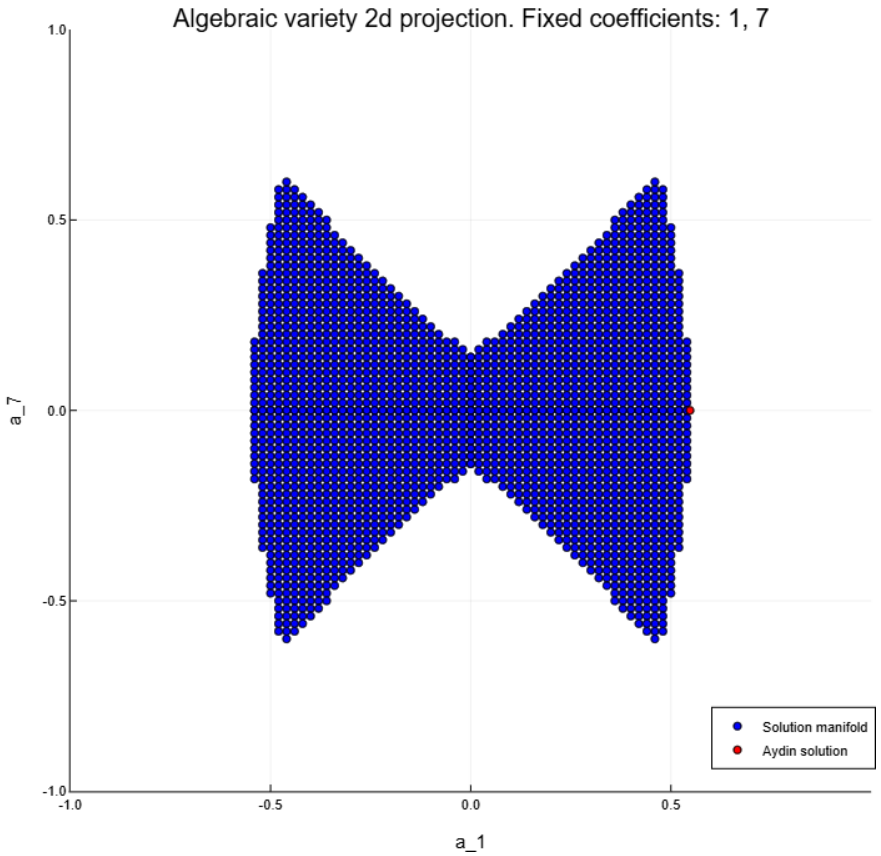


Figure 6: For block length and error weight $(n, t) = (7, 1)$, thisFigure shows for which values of α_0, α_6 coefficients real minimal PI codes are error correcting (KL solutions exist, blue dot), and for which they are not error correcting (KL solutions don't exist, no blue dot). Note that on the figure we say "Fixed coefficients $1, 7$" instead of $0, 6$, because these plots were built using Julia, which has 1-based indexing in arrays instead of the usual 0-based indexing. The "Aydin solution" labels the plotted coefficients value for qubit PI codes by Aydin et. al. [AAB24] at $(n, t) = (7, 1)$.

We can make the shift in fixed values of α_0, α_6 finer, from 0.02 to 0.001, see Figure 7. We conduct other even finer tests numerically, and since there are no "holes" in the final picture showcasing the solution space projection, this suggests that the projection is continuous.

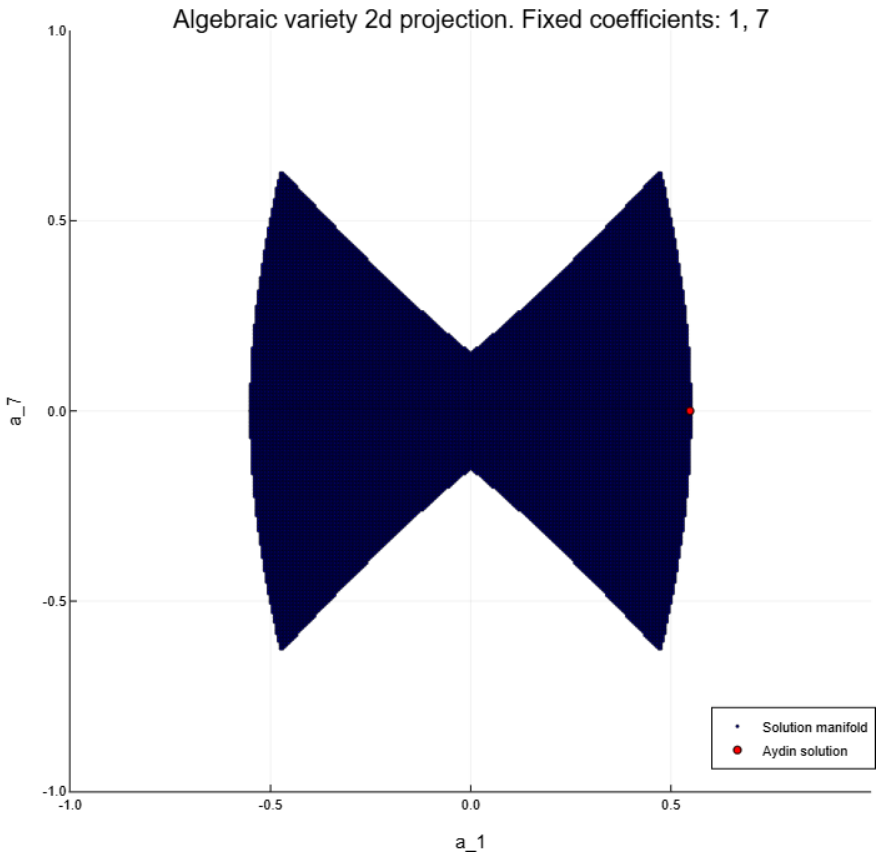
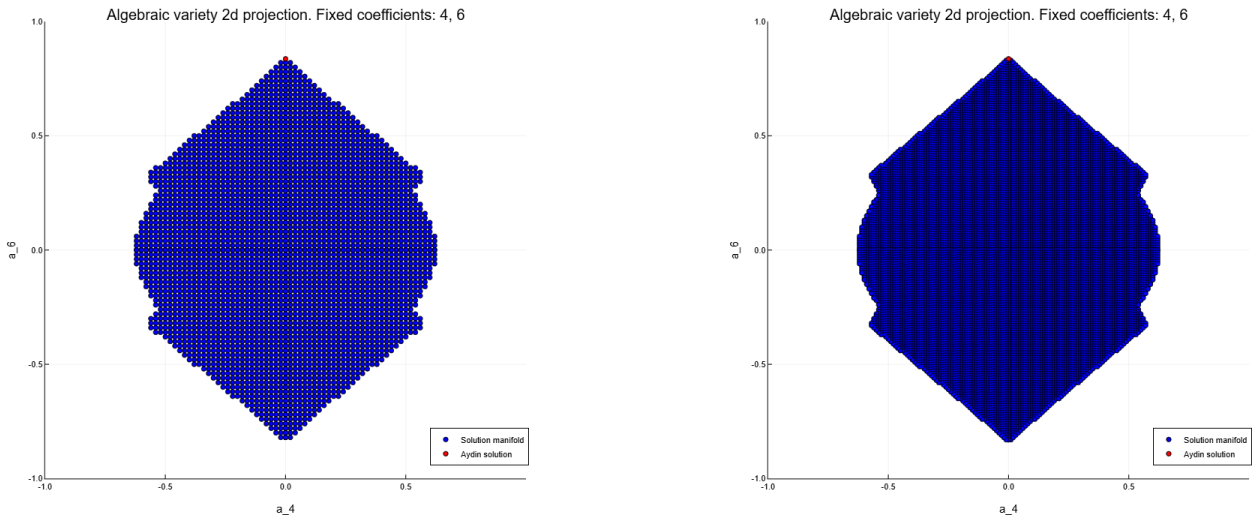


Figure 7: This figure showcases the same space as Figure 6, but with finer shift, and thus finer details. Note how existence of finer solutions suggests that the solution space is continuous.

We construct similar solution space projections for other pairs of fixed coefficients, in fact for $n = 7, t = 1$ we construct **all** projections, as shown on the Figure 9. All these projections exhibit similar behaviour of containing all finer solutions inbetween coarser ones as we make the shift finer, see Figure 8. We also see this for 3-dimensional projections of $n = 19, t = 2$. In other words, this suggests that all solution space projections are continuous, therefore the solution space itself must be continuous. We therefore refer to it as a solution manifold, or more formally an **algebraic variety**. We can thus formulate a stronger version of Conjecture 3.1 for **real** minimal qubit PI codes:

Conjecture 3.4. *For any fixed error weight t and block length $n = n_{min}(t)$ the family of **real** minimal qubit PI quantum error-correcting codes with parameters (n, t) is a continuous algebraic variety (manifold) of dimension $t + 1$.*



(a) Coarse. (b) Fine.

Figure 8: This figure showcases how a projection of the solution space starts looking more continuous as we go to finer and finer shifts for fixing coefficients.

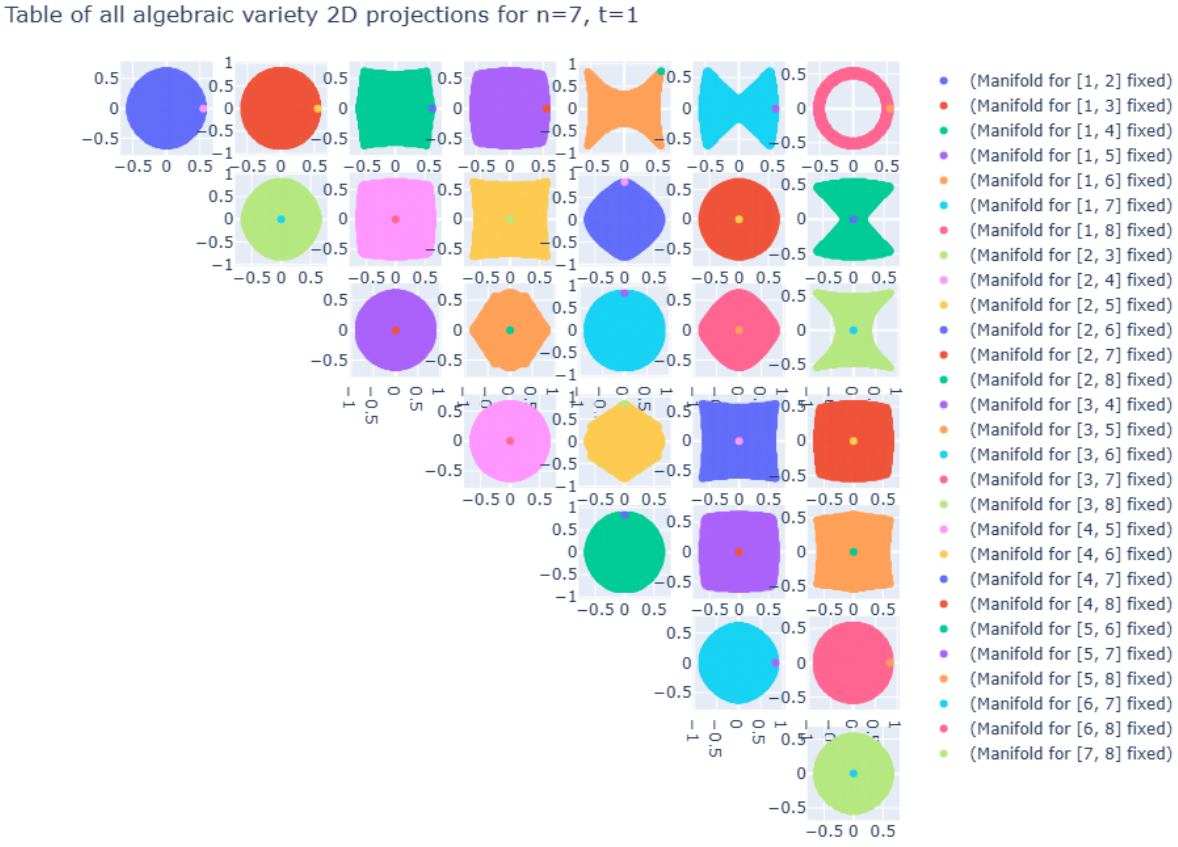


Figure 9: Table of all solution space projections on coordinate planes defined by all possible pairs of codeword coefficients α_i, α_j for $i \neq j \in \{0, \ldots, 7\}$ of real minimal qubit PI codes for $(n, t) = (7, 1)$. Note again that all the numbers on the figure that denote which coefficients are being fixed are shifted by 1 due to the array indexing in Julia being 1-based.

As a good sanity check, we can actually easily see that the symmetry of "flipping along the diagonal" present in Figure 9 is a consequence of the mirror symmetry of qubit PI codes. KL conditions (33) naturally have the swap symmetry of exchanging the first and second codewords, the system stays invariant. This means that whenever there is a solution for certain choice of fixing $\alpha_i = x, \alpha_j = y$ for any $i, j \in [n+1]$, there must exist a solution for fixing $\beta_i = x, \beta_j = y$, and vice versa. In other words, for any $(n_{min}(t), t)$:

$$\forall i, j \in [n+1] : (\exists \text{ solution for } \alpha_i = x, \alpha_j = y) \Leftrightarrow (\exists \text{ solution for } \beta_i = x, \beta_j = y). \quad (37)$$

Notice now that the mirror symmetry (36) implies the following:

$$\forall i, j \in [n+1] : (\exists \text{ solution for } \beta_i = x, \beta_j = y) \Leftrightarrow (\exists \text{ solution for } \alpha_{n-i} = x, \alpha_{n-j} = y). \quad (38)$$

So then finally:

$$\forall i, j \in [n+1] : (\exists \text{ solution for } \alpha_i = x, \alpha_j = y) \Leftrightarrow (\exists \text{ solution for } \alpha_{n-i} = x, \alpha_{n-j} = y), \quad (39)$$

which is exactly the symmetry of "flipping along the diagonal" from Figure 9.

Additional observations

To finish this section, for future reference we make a couple of remarks about other numerical observations we have made.

For some choices of fixed coefficients it is actually possible to find analytical solutions from approximate numerical ones, if enough of the other non-fixed coefficients are close to zero. For example, for $(n, t) = (7, 1)$ fixing α_0, α_7 to the top right point of the hollow disc in Figure 10 we can find analytical values for this code solution:

$$\alpha_0 = \alpha_7 = \frac{\sqrt{15}}{10}, \ \alpha_2 = -\alpha_5 = \frac{\sqrt{35}}{10}, \ \alpha_1 = \alpha_3 = \alpha_4 = \alpha_6 = 0, \quad (40)$$

which we find by knowing that $\alpha_1 = \alpha_3 = \alpha_4 = \alpha_6 = 0$ (from numerics, approximately), using $\alpha_0 = \alpha_7$ (by definition of this point), also using the normalization condition, and using $\alpha_2 = -\alpha_5$ (approximately, from numerics). In this way, one might think that building codes around cases where there are many zero codeword coefficients is beneficial, since it presumably becomes possible to find analytical solutions in the general case of n, t. For instance, the Aydin solution from Figure 11 is the furthest away from the center of its projection compared to **all** other points on other projections. Since everything is normalized, such a point would carry more "weight" and leave less possibilities for what values other coefficients can take (that is to say: other coefficients will be closer to zero if we choose to fix a point far from the center). Indeed: in this case it turns out that the top right edge point on Figure 11 (Aydin point) gives a solution with all other coefficients of the first codeword zero except for α_0, α_5. However, when we try to generalize this approach to $(n, t) = (19, 2)$ this idea already fails here at $t = 2$. The reason is that numerical evidence suggests that there are no real minimal qubit PI error correcting codes for $n = 19, t = 2$ that have more than half of its first codeword coefficients equal to zero (i.e. no more than 10 out of 20 are zero).

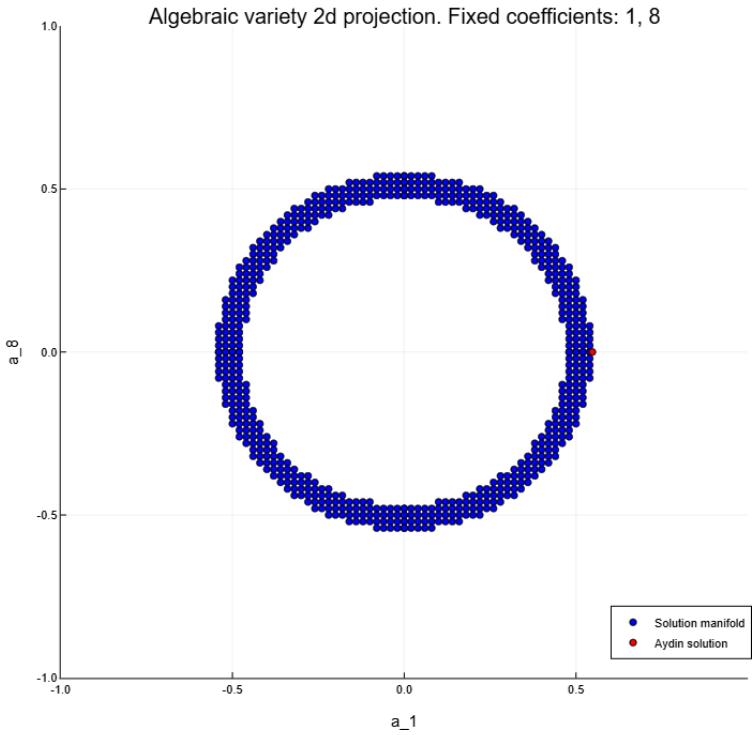


Figure 10: Block length and error weight $(n, t) = (7, 1)$, α_0, α_7 coefficients fixed to different values.

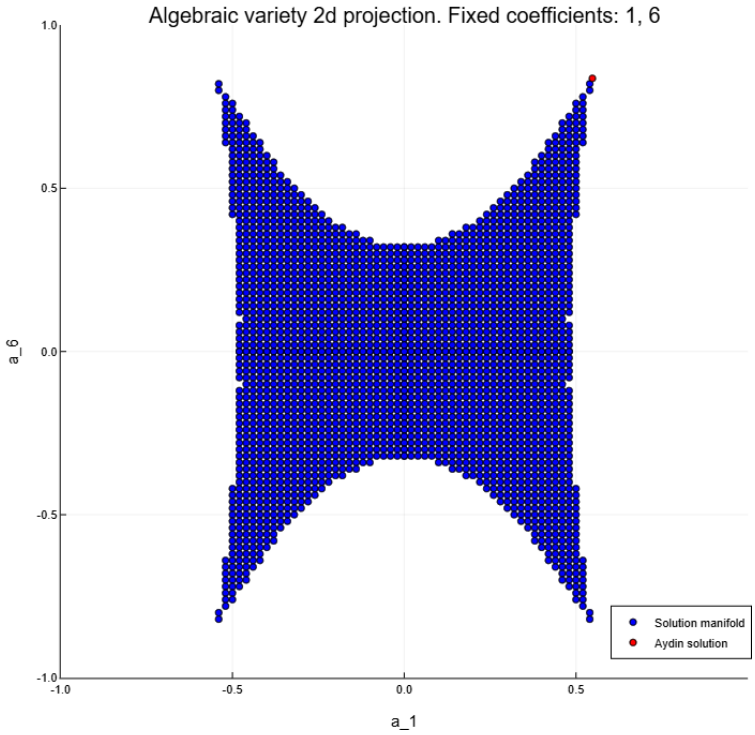


Figure 11: Block length and error weight $(n, t) = (7, 1)$, α_0, α_5 coefficients fixed to different values. Here at the Aydin solution point, only α_0, α_5 are nonzero, all other coefficients are zero due to construction in [AAB24] at $(n, t) = (7, 1)$.

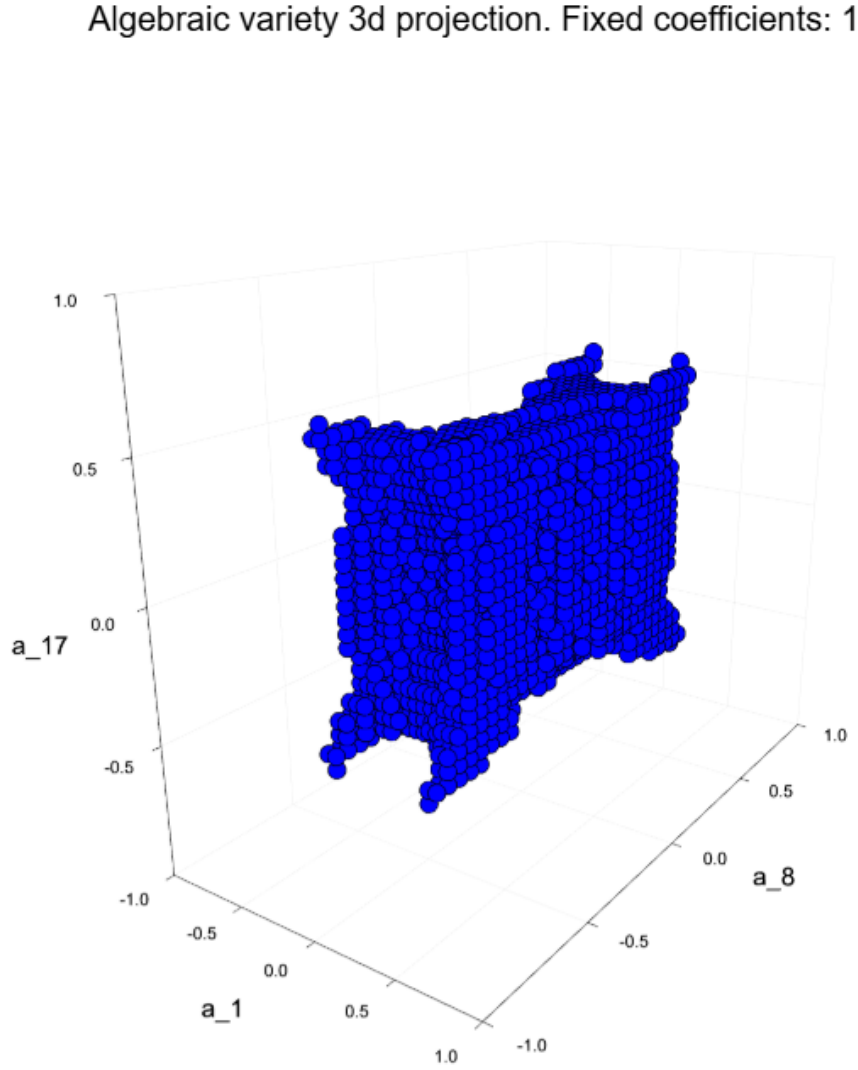


Figure 12: Example of a 3D projection of the real minimal PI code algebraic variety. Block length and error weight $(n, t) = (19, 2)$, $\alpha_0, \alpha_7, \alpha_{16}$ coefficients fixed to various different values.

All the html versions of the plots (with the point data) along with the Julia code used to obtain these numerical results can be found on GitHub here: [VB].

4 Qudit PI codes from points on simplices

In this section we take a qubit code construction that is close-to-optimal (according to our Conjecture 3.2) in block-length scaling $n(t)$, and thus in minimal distance scaling, and generalize it to qudits. The qubit code construction, which we base our qudit codes on, is by Aydin et. al., described in [AAB24]. We hope to achieve a construction that is also close-to-optimal in terms of block length scaling.

4.1 Aydin et. al. code construction

As usual, consider a block length (i.e. number of physical encoding qubits) n, and an error weight t (i.e. number of errors that we want to tolerate). As is standard for PI codes, we are defining the codewords in terms of a linear combination over Dicke states (See Definitions 1.1, 1.2). In this construction, to create more "space" between different terms, we want the weights of our Dicke states to be scaled by a parameter g, that way every weight is g apart. The parameter m defines how many terms we will have in a codeword, i.e. over how many different Dicke states the superposition will be in the definition of a codeword. Finally, a is a parameter akin to the one seen in the qubit version of the KL equations for PI codes 33, i.e. it is representing an action of a deletion error. The full code construction is as follows:

$$\begin{aligned} |c_0\rangle &= \sum_{\substack{l \text{ even} \\ 0\le l\le m}} f(l)\,|D^n_{gl}\rangle + \sum_{\substack{l \text{ odd} \\ 0\le l\le m}} f(l)\,|D^n_{n-gl}\rangle\,, \\ |c_1\rangle &= \sum_{\substack{l \text{ odd} \\ 0\le l\le m}} f(l)\,|D^n_{gl}\rangle + \epsilon \sum_{\substack{l \text{ even} \\ 0\le l\le m}} f(l)\,|D^n_{n-gl}\rangle\,, \end{aligned} \tag{41}$$

where $n, g, m, t \in \mathbb{N} : 0 \le 2t \le 2m < n/g$, and ϵ is just an orthogonality coefficient $\epsilon = \pm 1$.

Here $f(l)$ are the codeword coefficients, and they must satisfy certain conditions, derived when acting with KL equations directly on this code construction. Moreover, the KL conditions imply certain constraints on the block length n, specifically to be error-correcting we must have:

$$n = 2gm + \delta + 1,\ \delta \ge 2t, g \ge 2t, \tag{42}$$

where δ is a shift parameter, the nature of which is easier explained a bit later in the intuition Section 4.4. For full derivation see [AAB24]. Additionally, conditions that $f(l)$ must satisfy are:

$$\sum_{l=0}^{m} (-1)^l \frac{f(l)^2}{\binom{n}{gl}} \left[\binom{n-2t}{gl-a} - \binom{n-2t}{gl-2t+a} \right] = 0, \tag{43}$$

which is derived from C3 of the KL equations 33, and $f(l)$ should be normalized:

$$\sum_{l=0}^{m} f(l)^2 = 1, \tag{44}$$

which is derived from C1 of the KL equations 33.

One possible choice that Aydin et. al. make for the codeword coefficients (this choice is *not* unique!) is as follows:

$$f(l) = \sqrt{\binom{n/(2g)}{m} \frac{n-2gm}{g(m+1)} \frac{\binom{m}{l}}{\binom{n/g-l}{m+1}}} \tag{45}$$

4.2 Intuition behind the Aydin et. al. qubit PI code construction

In this section we present a convenient and intuitive way that we invented to represent PI codes by Aydin et. al. 4.1. A useful graphical interpretation of this PI code family is representing the basis vectors in the support of the codespace, i.e. the Dicke states, as points of a 1-dimensional simplex. Specifically: since for $q = d = 2$ each Dicke state is in one-to-one correspondence to a partition of n into $q = 2$ parts, we can consider a simplex $\mathcal{S} = \{\lambda = (\lambda_0, \lambda_1) \in \mathbb{R}^2 | \lambda_0 + \lambda_1 = n\}$ (note that here λ is not a partition since the parameters are continuous), with a discrete set of points $(gl, n-gl) \in \mathcal{S}$ and $(n-gl, gl) \in \mathcal{S}$ corresponding to the Dicke states on which the codewords in (41) are supported, for $l \in \{0, 1, \ldots, m\}$. This simplex is demonstrated in Figure 13. Everywhere, g is just a scaling parameter used to increase the code distance, or intuitively, increase the spacing between the discrete set of points in the simplex to account for the shifts caused by deletion errors, and prevent overlap between the Dicke states, after their weight gets shifted due to deletion errors.

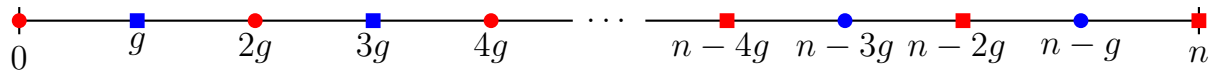


Figure 13: Simplex $\mathcal{S}$ as a graphical interpretation of the codespace of the Aydin et. al. family of PI codes. Here the points denote the entry λ_1 of the Dicke state partitions, which is sufficient since λ_0 can always be recovered as $\lambda_0 = n - \lambda_1$. Circles • correspond to terms in the support of the c_0 codeword, and squares ■ to terms that are in the support of the c_1 codeword. *Red* shapes correspond to terms with *even* l, and *blue* to terms with *odd* l.

4.3 Our code construction

Now, inspired by Aydin's qubit PI code construction from Section 4.1, we present our construction of qudit PI codes. Firstly, note that in our construction we generally use complex coefficients. Consider a block length n, an error weight t, a logical local dimension d and a physical local dimension $q \geq d$. Again, we are defining the codewords in terms of a linear combination over Dicke states (See Definitions 1.1, 1.2), but the weight of the Dicke states is now a partition, compared to the qubit case where it was a natural number. We also want the weights of our Dicke states to be scaled by a parameter g to, informally speaking, increase the distance between weights. $\vec{l}$ now will be the iterator for different terms in a codeword, basically each term (Dicke state) will have a different $\vec{l}$ associated to it. The space of all such $\vec{l}$ we call a region $\mathcal{R}$. To ensure orthogonality between terms, we introduce the Fourier phases

through the q-th root of unity ω_q, defined below. Finally, μ is a partition akin to the one seen in the qudit version of the KL equations for PI codes in Theorem 2.2, i.e. it is representing an action of a deletion error. The full code construction is as follows:

$$|c_i\rangle = \sum_{j=0}^{q-1} \omega_q^{ij} \sum_{\substack{\vec{l} \equiv i+j \bmod q \\ \vec{l} \in \mathcal{R}}} f(\vec{l}) \, |D^n_{\lambda^{g\vec{l}j}}\rangle \tag{46}$$

where we use the notation:

$$\begin{aligned} &\omega_q = e^{2\pi \mathrm{i}/q}, \ 2 < q \in \mathbb{N}\,; \\ &\lambda^{g\vec{l}j} = (gl_0, gl_1, \ldots, gl_{j-1}, n - (\sum_{k=0}^{q-2} gl_k), gl_j, \ldots, gl_{q-2}), \end{aligned} \tag{47}$$

where $\mathcal{R}$ is a discrete (finite) bounded space to which all the vectors $\vec{l}$ belong to, and for convenience we also denote:

$$\sum_{k=0}^{q-2} gl_k = S_{g\vec{l}} \tag{48}$$

In our case, to be error-correcting we must have: $n, g, b, t \in \mathbb{Z}; i, j \in [d]$ such that $0 \leq 2t \leq b < n/g$, where:

$$n = gb + \delta + 1, \ \delta \geq 2t, g \geq 2t, \tag{49}$$

and b is a constant determined by the region $\mathcal{R}$. Specifically the region $\mathcal{R}$ must satisfy:

$$\forall \vec{l}, \vec{l'} \in \mathcal{R}, \forall k \in [q-1] : l_0 + l_1 + \cdots + l_{k-1} + l'_k + l_k + \cdots + l_{q-2} \leq b \tag{50}$$

Additionally, again by applying condition C3 from the KL equations 2.2, for $\mu \overset{q}{\vdash} 2t$, $f(\vec{l})$ must satisfy:

$$\sum_{r=0}^{q-1} \sum_{\substack{\vec{l} \equiv r \bmod q \\ \vec{l} \in \mathcal{R}}} \frac{|f(\vec{l})|^2}{\binom{n}{\lambda^{g\vec{l}r}}} \left[\binom{n-2t}{\lambda^{g\vec{l}(r-i)} - \mu} - \binom{n-2t}{\lambda^{g\vec{l}(r-j)} - \mu} \right] = 0 \tag{51}$$

We also require that $f(\vec{l})$ is normalized, from condition C1 of KL equations 2.2:

$$\sum_{\vec{l} \in \mathcal{R}} |f(\vec{l})|^2 = 1 \tag{52}$$

and since we assume all the $f(\vec{l})$ that fall outside of the equation (51) to be zero, we might as well restrict the region $\mathcal{R}$ further to only contain vectors that are entrywise congruent modulo q, that is if $\vec{l} = (l_0, l_1, \ldots, l_{q-2}) \in \mathcal{R}$, it must satisfy:

$$l_0 \equiv l_1 \equiv \cdots \equiv l_{q-2} \bmod q \tag{53}$$

This code construction requires $q \geq d$, which is a reasonable requirement from the hardware perspective, since we want to utilize the extra degrees of freedom in local dimension q of our physical systems when building codes. Specifically: for a given q, all the codewords in the constructions for $d \leq q$ are a subset of the codewords in the code construction for $d = q$, that is, the function which maps logical basis states to the codewords for the case $d \leq q$ is simply a restriction with respect to the domain of the function in the case of $d = q$.

4.4 Intuition behind our code construction

We use a similar interpretation to the one in Section 4.2 when constructing a qudit generalization of Aydin codes. We will now have Dicke states corresponding to partitions of n into q parts where $q > 2$, and we will similarly graphically represent the codespace as points on a simplex of dimension $q - 1$. Specifically, to make this showcase easier and to be able to explain the intuition a bit better, here we consider qutrit codes $q = d = 3$ with a very simple region $\mathcal{R}$ that we call the *Mitsubishi region* (origin of this name would be obvious later), defined as $l_i \leq b/3 \; \forall i \in \{0, 1\}$, where we set $\delta = 0$ for simplicity of this showcase (note that this is then not a valid error-correcting code since we must have $\delta \geq 2t$). Recall that b is a parameter defining n, which unlike g or δ is not trivially dependent on the error weight t, but which will effectively determine the scaling $n(t)$. This parameter b would essentially be responsible for making the subset of the simplex where the codewords are defined larger. Clearly, we can represent the region $\mathcal{R}$, defined above, as showcased in Figure 14.

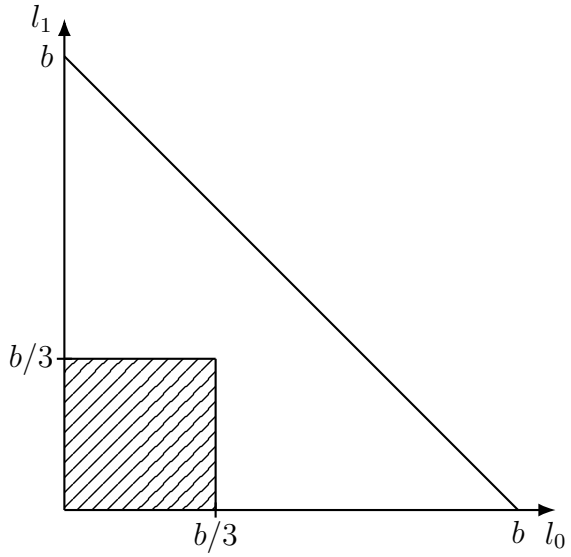


Figure 14: Mitsubishi region (origin of this name would be obvious later), represented by the shaded square in this figure.

Now in order to construct the full codespace as a set of points of types $(gl_0, gl_1, n - gl_0 - gl_1)$, $(gl_0, n - gl_0 - gl_1, gl_1)$, and $(n - gl_0 - gl_1, gl_0, gl_1)$ on the simplex $\mathcal{S} = \{\lambda = (\lambda_0, \lambda_1, \lambda_2) \in \mathbb{R}^3 | \lambda_0 + \lambda_1 + \lambda_2 = n\}$ we notice that those three types of points correspond to the shaded part of the region $\mathcal{R}$ on three different coordinate projections: to (λ_0, λ_1), (λ_0, λ_2), and (λ_1, λ_2) planes respectively. From these projections we can reconstruct the full codespace, as shown in Figure 15.

We then have a simplex of dimension 2, with the discrete set of points on it as described above, and with three vertices corresponding to the cases $\lambda = (n, 0, 0)$, $\lambda = (0, n, 0)$, and $\lambda = (0, 0, n)$ respectively, denoted by $j = 0, j = 1, j = 2$, as described in the code construction in Section 4.3. Just like in the construction by Aydin et. al., the main idea is to have alternating parity in the patterns of the simplex in the support of the codewords, to ensure that there is no overlap between codewords. This is showcased in Figure 16.

Finally, adding δ to this region (remember that previously we set it to zero for simplicity) shifts the subsets of the simplex, creating more space to ensure no overlap occurs "in the middle of the simplex" when introducing deletion errors. This action is illustrated in Figure 17.

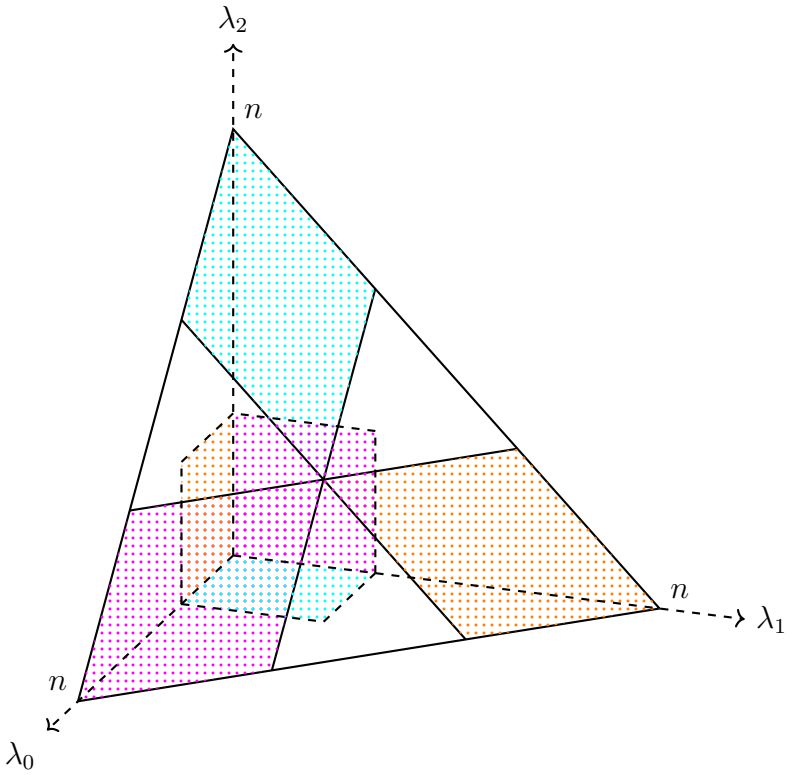
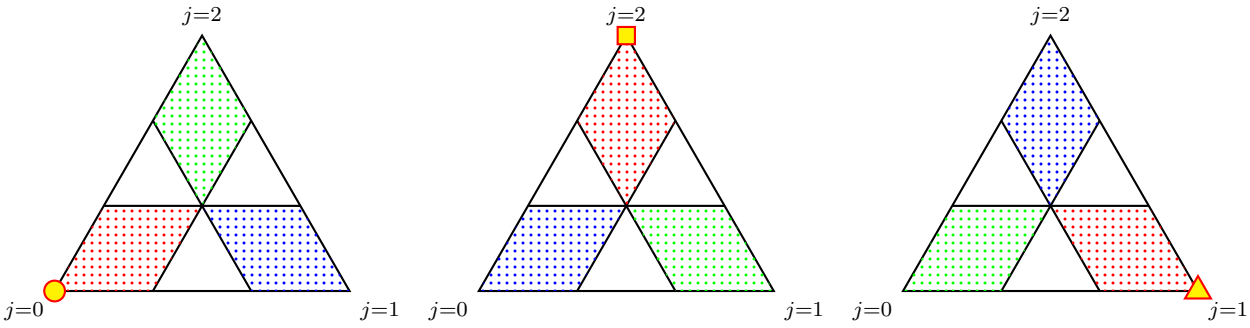


Figure 15: Reconstructing the codespace as the discrete subset of the simplex $\mathcal{S}$ from the region $\mathcal{R}$. Colors of the subsets correspond to their respective shaded square (i.e. their region $\mathcal{R}$) on the corresponding axial plane.



c_0 codeword c_1 codeword c_2 codeword

Figure 16: Graphical interpretation of our code construction (simplicial codes) for qutrit codes, $q = d = 3$. The region $\mathcal{R}$ for these codes is described above. Note that each dot on these pictures corresponds to some unique partition $\lambda = (\lambda_0, \lambda_1, \lambda_2)$ such that no dot overlaps with another one. In order to not clutter the image, we divided it into 3 parts, each corresponding to their respective codeword support. *Red* shapes correspond to terms with $\vec{l} \equiv 0 \mod q$, *blue* to terms with $\vec{l} \equiv 1 \mod q$, and *green* to terms with $\vec{l} \equiv 2 \mod q$. The yellow circle, square, and triangle denote which of the vertices of the simplex belong to the c_0, c_1, and c_2 codewords respectively. Each codeword is supported on only one of the three vertices generally.

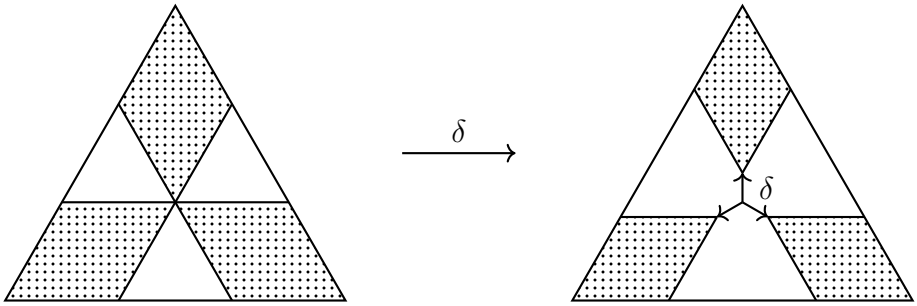


Figure 17: Action of adding the shift δ, illustrated graphically (shift happens from the center).

4.5 Proof that our qudit codes are error-correcting

To show the correctness of the construction given in Section 4.3, we must show that under the assumptions of this construction the equations given in Theorem 2.2 are satisfied, so that our codes are error-correcting.

Theorem 4.1. *The qudit PI codes described in Section 4.3 are error-correcting.*

Note: this theorem holds only if all the conditions for the parameters of these codes (Section 4.3) hold.

Proof.

C1

By definition of simplicial codes given in equation (46), we have:

$$\alpha_{i,\lambda} = \sum_{j=0}^{q-1} \omega_q^{ij} \sum_{\substack{\vec{l} \equiv i+j \bmod q \\ \vec{l} \in \mathcal{R}}} f(\vec{l}) \delta_{\lambda, \lambda^{g\vec{l}_j}} \tag{54}$$

First, it is easy to see that within our construction the codewords are normalized, since $\forall i \in [d]$:

$$\begin{aligned}
\sum_{\lambda \vdash_n^q} \bar{\alpha}_{i,\lambda} \alpha_{i,\lambda} &= \sum_{\lambda \vdash_n^q} \sum_{j,j'=0}^{q-1} \omega_q^{-ij} \omega_q^{ij'} \sum_{\substack{\vec{l} \equiv i+j \bmod q \\ \vec{l} \in \mathcal{R}}} \sum_{\substack{\vec{l'} \equiv i+j' \bmod q \\ \vec{l'} \in \mathcal{R}}} \bar{f}(\vec{l}) f(\vec{l'}) \delta_{\lambda, \lambda^{g\vec{l}_j}} \delta_{\lambda, \lambda^{g\vec{l'}_{j'}}} = \\
&= \sum_{\lambda \vdash_n^q} \sum_{j,j'=0}^{q-1} \omega_q^{-ij} \omega_q^{ij'} \sum_{\substack{\vec{l} \equiv i+j \bmod q \\ \vec{l} \in \mathcal{R}}} \sum_{\substack{\vec{l'} \equiv i+j' \bmod q \\ \vec{l'} \in \mathcal{R}}} \bar{f}(\vec{l}) f(\vec{l'}) \delta_{\lambda, \lambda^{g\vec{l}_j}} \delta_{\lambda^{g\vec{l}_j}, \lambda^{g\vec{l'}_{j'}}} \delta_{j,j'} \delta_{\vec{l},\vec{l'}} = \\
&= \sum_{j=0}^{q-1} \sum_{\substack{\vec{l} \equiv i+j \bmod q \\ \vec{l} \in \mathcal{R}}} \bar{f}(\vec{l}) f(\vec{l}) = \sum_{\vec{l} \in \mathcal{R}} |f(\vec{l})|^2 = 1
\end{aligned} \tag{55}$$

where the last equality holds by assumption from equation (52).

Next, just like in Aydin's family of PI codes, to simplify solving the equations in Theorem 2.2, our intuition when constructing a family of qudit PI codes was to impose a stronger constraint on codewords than orthogonality, namely to make the codewords have disjoint support in the symmetric subspace. That is, our construction has the following property:

$$\bar{\alpha}_{i,\lambda} \alpha_{j,\lambda} = 0, \ \forall i \neq j \in [d] \tag{56}$$

from which $\sum_{\lambda \vdash_n^q} \bar{\alpha}_{i,\lambda} \alpha_{j,\lambda} = 0, \ \forall i \neq j \in [d]$ would trivially follow. To show equalities (56), we have:

$$\begin{aligned}
\bar{\alpha}_{i,\lambda} \alpha_{j,\lambda} &= \sum_{r,s=0}^{q-1} \omega_q^{-ir} \omega_q^{js} \sum_{\substack{\vec{l} \equiv i+r \bmod q \\ \vec{l} \in \mathcal{R}}} \sum_{\substack{\vec{l'} \equiv j+s \bmod q \\ \vec{l'} \in \mathcal{R}}} \bar{f}(\vec{l}) f(\vec{l'}) \delta_{\lambda, \lambda^{g\vec{l}_r}} \delta_{\lambda, \lambda^{g\vec{l'}_s}} = \\
&= \sum_{r,s=0}^{q-1} \omega_q^{-ir} \omega_q^{js} \sum_{\substack{\vec{l} \equiv i+r \bmod q \\ \vec{l} \in \mathcal{R}}} \sum_{\substack{\vec{l'} \equiv j+s \bmod q \\ \vec{l'} \in \mathcal{R}}} \bar{f}(\vec{l}) f(\vec{l'}) \delta_{\lambda, \lambda^{g\vec{l}_r}} \delta_{\lambda^{g\vec{l}_r}, \lambda^{g\vec{l'}_s}}
\end{aligned} \tag{57}$$

• for terms with $r = s$:

$$\delta_{\lambda^{g\vec{l}_r}, \lambda^{g\vec{l'}_s}} = \delta_{\vec{l},\vec{l'}} \tag{58}$$

and since for these terms

$$\left.\begin{aligned} l_0 \equiv l_1 \equiv \cdots \equiv l_{q-2} \equiv i + r \bmod q \\ l'_0 \equiv l'_1 \equiv \cdots \equiv l'_{q-2} \equiv j + r \bmod q \end{aligned}\right\}, i \neq j \Rightarrow \vec{l} \neq \vec{l'} \tag{59}$$

thus: $\delta_{\lambda^{g\vec{l}r},\lambda^{g\vec{l}'r}} = \delta_{\vec{l},\vec{l}'} = 0$

• for terms with $r \neq s$:

$$\delta_{\lambda^{g\vec{l}r},\lambda^{g\vec{l}'s}} = \delta_{l_0,l'_0}\delta_{l_1,l'_1}\dots\delta_{l_{r-1},l'_{r-1}}\delta_{(n-S_{g\vec{l}}),gl'_r}\delta_{l_r,l'_{r+1}}\dots\delta_{l_{s-2},l'_{s-1}}\delta_{gl_{s-1},(n-S_{g\vec{l}'})}\delta_{l_s,l'_s}\dots\delta_{l_{q-2},l'_{q-2}} \quad (60)$$

but due to equation (50), we have:

$$gl'_r + S_{g\vec{l}} \leq gb = n - \delta - 1 < n \Rightarrow \delta_{(n-S_{g\vec{l}}),gl'_r} = 0 \quad (61)$$

thus: $\delta_{\lambda^{g\vec{l}r},\lambda^{g\vec{l}'s}} = 0$, which concludes the proof of C1. □

C2

Consider some $\mu, \nu \overset{q}{\vdash} 2t$. In the following, we show that for all λ, μ, ν and all $i \neq j$ we have $\bar{\alpha}_{i,\lambda}\alpha_{j,\lambda-\mu+\nu} = 0$, which is a stronger condition than required for C2 to hold. Without loss of generality, here we can consider only those λ that satisfy $\lambda - \mu \geq 0$, since due to proof of C2 of Theorem 2.2 the terms with λ that don't satisfy this are zero. Then we have for all $i \neq j$:

$$\bar{\alpha}_{i,\lambda}\alpha_{j,\lambda-\mu+\nu} = \sum_{r,s=0}^{q-1}\omega_q^{-ir}\omega_q^{js}\sum_{\substack{\vec{l}\equiv i+r \bmod q\\ \vec{l}\in\mathcal{R}}}\sum_{\substack{\vec{l}'\equiv j+s \bmod q\\ \vec{l}'\in\mathcal{R}}}\bar{f}(\vec{l})f(\vec{l}')\delta_{\lambda,\lambda^{g\vec{l}r}}\delta_{\lambda-\mu+\nu,\lambda^{g\vec{l}'s}} =$$

$$= \sum_{r,s=0}^{q-1}\omega_q^{-ir}\omega_q^{js}\sum_{\substack{\vec{l}\equiv i+r \bmod q\\ \vec{l}\in\mathcal{R}}}\sum_{\substack{\vec{l}'\equiv j+s \bmod q\\ \vec{l}'\in\mathcal{R}}}\bar{f}(\vec{l})f(\vec{l}')\delta_{\lambda,\lambda^{g\vec{l}r}}\delta_{\lambda^{g\vec{l}r}-\mu+\nu,\lambda^{g\vec{l}'s}} \quad (62)$$

• for terms with $r \neq s$:

$$\delta_{\lambda^{g\vec{l}r}-\mu+\nu,\lambda^{g\vec{l}'s}} = \delta_{gl_0-\mu_0+\nu_0,gl'_0}\delta_{gl_1-\mu_1+\nu_1,gl'_1}\dots\delta_{gl_{r-1}-\mu_{r-1}+\nu_{r-1},gl'_{r-1}}\delta_{(n-\mu_r+\nu_r-S_{g\vec{l}}),gl'_r}\delta_{gl_r-\mu_{r+1}+\nu_{r+1},gl'_{r+1}}\dots$$
$$\dots\delta_{gl_{s-2}-\mu_{s-1}+\nu_{s-1},gl'_{s-1}}\delta_{gl_{s-1}-\mu_s+\nu_s,(n-S_{g\vec{l}'})}\delta_{gl_s-\mu_{s+1}+\nu_{s+1},gl'_s}\dots\delta_{gl_{q-2}-\mu_{q-1}+\nu_{q-1},gl'_{q-2}} \quad (63)$$

but due to equation (50), we have:

$$gl'_r + \mu_r - \nu_r + S_{g\vec{l}} \leq gb + \mu_r - \nu_r = n + \mu_r - \nu_r - \delta - 1 \leq n + 2t - \delta - 1 \leq n - 1 < n \Rightarrow$$
$$\Rightarrow \delta_{(n-\mu_r+\nu_r-S_{g\vec{l}}),gl'_r} = 0 \quad (64)$$

thus: $\delta_{\lambda^{g\vec{l}r}-\mu+\nu,\lambda^{g\vec{l}'s}} = 0$

• for terms with $r = s$:

$$\delta_{\lambda^{g\vec{l}r}-\mu+\nu,\lambda^{g\vec{l}'r}} = \left(\prod_{k=0}^{r-1}\delta_{gl_k-\mu_k+\nu_k,gl'_k}\right)\delta_{(n-\mu_r+\nu_r-S_{g\vec{l}}),(n-S_{g\vec{l}'})}\left(\prod_{k=r}^{q-2}\delta_{gl_k-\mu_{k+1}+\nu_{k+1},gl'_k}\right) \quad (65)$$

where $l_k \equiv i + r \bmod q$ and $l'_k \equiv j + r \bmod q$ with $i \neq j$. Since $g \geq 2t$ and $|\nu_k - \mu_k| \leq 2t$, to satisfy $gl_k + \nu_k - \mu_k = gl'_k$ we must have $g = 2t$, since otherwise:

$$|l'_k - l_k| \geq 1, g > 2t \Rightarrow |\nu_k - \mu_k| > 2t \text{ – contradiction.} \quad (66)$$

thus for $g = 2t$:

$$\forall k \in [q] : \nu_k - \mu_k = \pm 2t \Rightarrow \forall k \in [q-1] : l'_k = l_k \pm 1 \tag{67}$$

Without loss of generality, let's assume $\nu_k - \mu_k = 2t$, which implies $l'_k = l_k + 1$ and therefore $l'_k - l_k = j - i = 1$. But then for $q > 2$, we have that for any other $u \in [q]$ such that $u \neq k,\ u \neq r$ the following holds: $\nu_u - \mu_u \neq 2t$. This then implies that $j - i \neq 1$, since there is a corresponding inequality $l'_u - l_u \neq 1$ or $l'_{u-1} - l_{u-1} \neq 1$. This clearly leads to a contradiction. Therefore the condition $gl_k + \nu_k - \mu_k = gl'_k$ or $gl_k + \nu_{k+1} - \mu_{k+1} = gl'_k$ cannot be simultaneously satisfied for all $k \in [q-1]$, therefore we must have that $\delta_{\lambda^{g\vec{l}r} - \mu + \nu, \lambda^{g\vec{l'}r}} = 0$, which concludes the proof of C2 for any $q > 2$. For $q = 2$, see [AAB24]. □

C3

Again, consider $\mu, \nu \in 2t$ and, without loss of generality, consider only those λ that satisfy $\lambda - \mu \geq 0$. We have:

$$\begin{aligned}\bar{\alpha}_{i,\lambda}\alpha_{i,\lambda-\mu+\nu} &= \sum_{r,s=0}^{q-1} \omega_q^{-ir}\omega_q^{is} \sum_{\substack{\vec{l} \equiv i+r \bmod q \\ \vec{l} \in \mathcal{R}}} \sum_{\substack{\vec{l'} \equiv i+s \bmod q \\ \vec{l'} \in \mathcal{R}}} \bar{f}(\vec{l}) f(\vec{l'}) \delta_{\lambda, \lambda^{g\vec{l}r}} \delta_{\lambda-\mu+\nu, \lambda^{g\vec{l'}s}} = \\ &= \sum_{r,s=0}^{q-1} \omega_q^{i(s-r)} \sum_{\substack{\vec{l} \equiv i+r \bmod q \\ \vec{l} \in \mathcal{R}}} \sum_{\substack{\vec{l'} \equiv i+s \bmod q \\ \vec{l'} \in \mathcal{R}}} \bar{f}(\vec{l}) f(\vec{l'}) \delta_{\lambda, \lambda^{g\vec{l}r}} \delta_{\lambda^{g\vec{l}r} - \mu + \nu, \lambda^{g\vec{l'}s}}\end{aligned} \tag{68}$$

• for terms with $r \neq s$:

$$\begin{aligned}\delta_{\lambda^{g\vec{l}r} - \mu + \nu, \lambda^{g\vec{l'}s}} = \delta_{gl_0 - \mu_0 + \nu_0, gl'_0} \delta_{gl_1 - \mu_1 + \nu_1, gl'_1} \cdots \delta_{gl_{r-1} - \mu_{r-1} + \nu_{r-1}, gl'_{r-1}} \delta_{(n - \mu_r + \nu_r - S_{g\vec{l}}), gl'_r} \delta_{gl_r - \mu_{r+1} + \nu_{r+1}, gl'_{r+1}} \cdots \\ \cdots \delta_{gl_{s-2} - \mu_{s-1} + \nu_{s-1}, gl'_{s-1}} \delta_{gl_{s-1} - \mu_s + \nu_s, (n - S_{g\vec{l'}})} \delta_{gl_s - \mu_{s+1} + \nu_{s+1}, gl'_s} \cdots \delta_{gl_{q-2} - \mu_{q-1} + \nu_{q-1}, gl'_{q-2}}\end{aligned} \tag{69}$$

but again due to equation (50), we have:

$$\begin{aligned}gl'_r + \mu_r - \nu_r + S_{g\vec{l}} \leq gb + \mu_r - \nu_r = n + \mu_r - \nu_r - \delta - 1 \leq n + 2t - \delta - 1 \leq n - 1 < n \Rightarrow \\ \Rightarrow \delta_{(n - \mu_r + \nu_r - S_{g\vec{l}}), gl'_r} = 0\end{aligned} \tag{70}$$

thus: $\delta_{\lambda^{g\vec{l}r} - \mu + \nu, \lambda^{g\vec{l'}s}} = 0$

• for terms with $r = s$:

$$\delta_{\lambda^{g\vec{l}r} - \mu + \nu, \lambda^{g\vec{l'}r}} = \left(\prod_{k=0}^{r-1} \delta_{gl_k - \mu_k + \nu_k, gl'_k} \right) \delta_{(n - \mu_r + \nu_r - S_{g\vec{l}}), (n - S_{g\vec{l'}})} \left(\prod_{k=r}^{q-2} \delta_{gl_k - \mu_{k+1} + \nu_{k+1}, gl'_k} \right) \tag{71}$$

where $l_k \equiv i + r \bmod q$ and $l'_k \equiv i + r \bmod q$. So the only terms that survive must satisfy $gl_k + \nu_k - \mu_k = gl'_k$ for all $k \in [q-1]$. Since $g \geq 2t$, $q \geq 2$, $l_k \equiv l'_k \bmod q$, and $|\nu_k - \mu_k| \leq 2t$,

to satisfy $gl_k + \nu_k - \mu_k = gl'_k$ we must have $\nu_k = \mu_k$ and $l_k = l'_k$ for all $k \in [q-1]$. Therefore only terms with $\mu = \nu$ and $\vec{l} = \vec{l'}$ survive, thus:

$$\bar{\alpha}_{i,\lambda}\alpha_{i,\lambda-\mu+\nu} = |\alpha_{i,\lambda}|^2 = \sum_{r=0}^{q-1} \sum_{\substack{\vec{l}\equiv i+r \bmod q \\ \vec{l}\in\mathcal{R}}} |f(\vec{l})|^2 \delta_{\lambda,\lambda^{g\vec{l}r}} \tag{72}$$

Now consider the full expression for C3 from Theorem 2.2 and substitute the expression from equation (72). We have:

$$\sum_{\substack{q \\ \lambda\vdash n}} (\bar{\alpha}_{i,\lambda}\alpha_{i,\lambda-\mu+\nu} - \bar{\alpha}_{j,\lambda}\alpha_{j,\lambda-\mu+\nu}) \frac{\binom{n-2t}{\lambda-\mu}}{\sqrt{\binom{n}{\lambda}\binom{n}{\lambda-\mu+\nu}}} = \sum_{\substack{q \\ \lambda\vdash n}} \frac{\binom{n-2t}{\lambda-\mu}}{\binom{n}{\lambda}} (|\alpha_{i,\lambda}|^2 - |\alpha_{j,\lambda}|^2) =$$

$$= \sum_{\substack{q \\ \lambda\vdash n}} \frac{\binom{n-2t}{\lambda-\mu}}{\binom{n}{\lambda}} \left(\sum_{r=0}^{q-1} \sum_{\substack{\vec{l}\equiv i+r \bmod q \\ \vec{l}\in\mathcal{R}}} |f(\vec{l})|^2 \delta_{\lambda,\lambda^{g\vec{l}r}} - \sum_{r'=0}^{q-1} \sum_{\substack{\vec{l'}\equiv j+r' \bmod q \\ \vec{l'}\in\mathcal{R}}} |f(\vec{l'})|^2 \delta_{\lambda,\lambda^{g\vec{l'}r'}} \right) := C \tag{73}$$

Now since the sum is over the same domain, we can collect the terms in equation (73) in such a way that $j + r' \equiv i + r \mod q$, i.e. set $r' \equiv i + r - j \mod q$. Then we have:

$$C = \sum_{\substack{q \\ \lambda\vdash n}} \frac{\binom{n-2t}{\lambda-\mu}}{\binom{n}{\lambda}} \sum_{r=0}^{q-1} \sum_{\substack{\vec{l}\equiv i+r \bmod q \\ \vec{l}\in\mathcal{R}}} |f(\vec{l})|^2 (\delta_{\lambda,\lambda^{g\vec{l}r}} - \delta_{\lambda,\lambda^{g\vec{l}(i+r-j)}}) =$$

$$= \sum_{r=0}^{q-1} \sum_{\substack{\vec{l}\equiv i+r \bmod q \\ \vec{l}\in\mathcal{R}}} |f(\vec{l})|^2 \left(\frac{\binom{n-2t}{\lambda^{g\vec{l}r}-\mu}}{\binom{n}{\lambda^{g\vec{l}r}}} - \frac{\binom{n-2t}{\lambda^{g\vec{l}(i+r-j)}-\mu}}{\binom{n}{\lambda^{g\vec{l}(i+r-j)}}} \right) \tag{74}$$

Now, by relabeling the summation index $i + r \to r$, and rearranging the terms, we get:

$$C = \sum_{r=0}^{q-1} \sum_{\substack{\vec{l}\equiv r \bmod q \\ \vec{l}\in\mathcal{R}}} |f(\vec{l})|^2 \left(\frac{\binom{n-2t}{\lambda^{g\vec{l}(r-i)}-\mu}}{\binom{n}{\lambda^{g\vec{l}(r-i)}}} - \frac{\binom{n-2t}{\lambda^{g\vec{l}(r-j)}-\mu}}{\binom{n}{\lambda^{g\vec{l}(r-j)}}} \right) \tag{75}$$

Finally, notice that multinomial coefficients are invariant under permutations of indices in their defining partition, therefore:

$$\binom{n}{\lambda^{g\vec{l}(r-i)}} = \binom{n}{\lambda^{g\vec{l}(r-j)}} = \binom{n}{\lambda^{g\vec{l}r}} \tag{76}$$

so the expression simplifies further to:

$$C = \sum_{r=0}^{q-1} \sum_{\substack{\vec{l}\equiv r \bmod q \\ \vec{l}\in\mathcal{R}}} \frac{|f(\vec{l})|^2}{\binom{n}{\lambda^{g\vec{l}r}}} \left[\binom{n-2t}{\lambda^{g\vec{l}(r-i)}-\mu} - \binom{n-2t}{\lambda^{g\vec{l}(r-j)}-\mu} \right] = 0 \tag{77}$$

where the above equation (77) is zero due to assumption in equation (51).

□

Since C1, C2, and C3 are satisfied, this concludes the full proof.

■

Note that since $\lambda^{g\vec{l}(r-j)} = \sigma_{r-i,r-j} \cdot \lambda^{g\vec{l}(r-i)}$ and since permutations acting on the full multinomial coefficients leave them invariant, we also have the following equality:

$$\binom{n-2t}{\lambda^{g\vec{l}(r-j)} - \mu} = \binom{n-2t}{\lambda^{g\vec{l}(r-i)} - \sigma_{r-i,r-j} \cdot \mu} \tag{78}$$

5 Optimal region $\mathcal{R}$ for our qudit PI codes

Notice that for a given t the equations (51) $\forall i,j \in [d]; \mu \overset{q}{\vdash} 2t$ become a linear system of equations in (non-negative) variables $|f(\vec{l})|^2$ once we specify the region $\mathcal{R}$ and find the minimal distance scaling $n(t)$. Without fixing $n(t)$ and the region $\mathcal{R}$, while much simpler than the system from Theorem 2.2, this is generally still a difficult non-linear system of equations to solve. If we can find a general approach to give the best region $\mathcal{R}$ (in terms of giving the biggest scaling of the code distance $d_C(n)$, i.e. smallest $n(t)$), then finding $f(\vec{l})$ reduces to solving a linear program (51). In this section we construct a good approximation to the optimal region $\mathcal{R}$, specifically we define a region $\mathcal{R}$ that approaches the optimal region $\mathcal{R}$ as t grows.

5.1 Boundary regions

Notice how equation (50) already puts a significant constraint on what regions $\mathcal{R}$ we can choose:

$$\forall \vec{l}, \vec{l'} \in \mathcal{R}, \forall k \in [q-1] : l_0 + l_1 + \cdots + l_{k-1} + l'_k + l_k + \cdots + l_{q-2} \leq b$$

Now, for any given region $\mathcal{R}$, it is defined by coordinate values l_i of vectors $\vec{l}$ belonging to it. Consider a coordinate value of a vector $\vec{l}$ of this region that is greater than any other coordinate value of any other vector in this region. That is, this is a maximal coordinate value along the boundary of $\mathcal{R}$. We will denote this value as l_{max}, i.e. l_{max} satisfies:

$$\forall \vec{l} \in \mathcal{R}\ \forall k \in [q-1] : l_k \leq l_{max} \tag{79}$$

Since $\vec{l}, \vec{l'}$ in equation (50) are two independent vectors belonging to the same region, we have $l'_k \leq l_{max}$, and actually since equation (50) is supposed to hold for any $\vec{l'} \in \mathcal{R}$, then there always exists a k for which we can choose $\vec{l'}$ to contain l_{max} as its k-th coordinate, i.e. we can choose $\vec{l'}$ that satisfies $l'_k = l_{max}$. So then the condition in equation (50) is equivalent to:

$$\forall \vec{l} \in \mathcal{R} : \begin{cases} l_0 + l_1 + \cdots + l_{q-2} + l_{max} \leq b \\ \forall k \in [q-1] : l_k \leq l_{max} \end{cases} \tag{80}$$

From this simplified form of the region $\mathcal{R}$ constraints in equation (80) we can already see that l_{max} and b would be the only two parameters defining the region. Moreover, it's clear that we can easily rescale the equations by b, so then b would really only be responsible for the size of the region, while l_{max} would be defining its shape. From these equations we can already put some constraints on l_{max} in terms of b, that would define the boundary regions.

Upper bound:

It is easy to see that l_{max} can't be larger than $b/2$. To prove this, assume $l_{max} > b/2$, then $\exists \vec{l} \in \mathcal{R}$ that has l_{max} as one of its entries, therefore for this $\vec{l}$ we have:

$$l_0 + l_1 + \cdots + l_{q-2} + l_{lmax} > 2l_{max} > b$$

so that equation (80) doesn't hold, which is a contradiction. Thus the upper bound is: $l_{max} \leq b/2$.
For $l_{max} = b/2$, the constraint is simply:

$$l_0 + l_1 + \cdots + l_{q-2} \leq b/2 \tag{81}$$

thus the region $\mathcal{R}$ is just a simplex of dimension $q-1$. For $q = 3$, $\mathcal{R}$ with $l_{max} = b/2$ looks like in Figure 18.

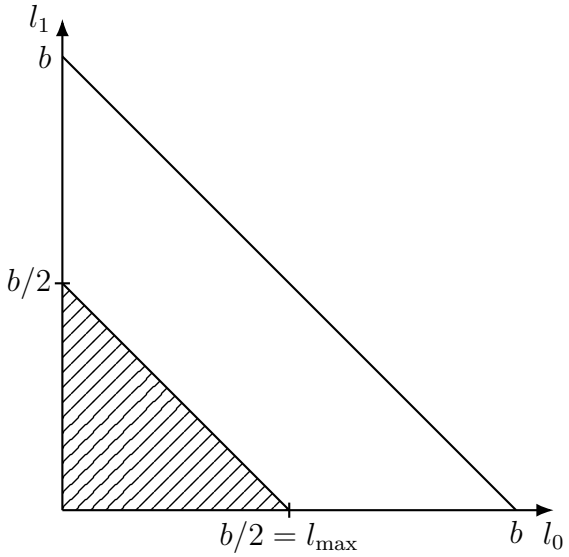


Figure 18: region $\mathcal{R}$ with $l_{max} = b/2$ and $q = 3$, shown as the shaded part of the figure.

And the codespace formed from this region (constructed via the same method as described in Section 4.4) is shown on Figure 19. We refer to this $\mathcal{R}$ as the *Sierpinski region*.

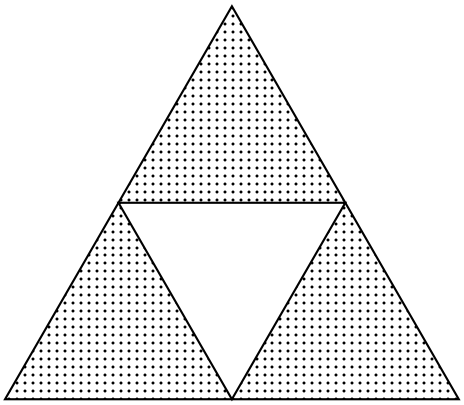


Figure 19: Codespace formed from the Sierpinski region for $q = 3$, shown as the dotted subset of the simplex.

Lower bound:

If we make l_{max} less or equal to b/q, then the first inequality in equation (80) is automatically satisfied:

$$l_{max} \leq b/q \Rightarrow l_0 + l_1 + \cdots + l_{q-2} + l_{lmax} \leq q \cdot l_{max} \leq q \cdot b/q = b$$

Thus for any values of $l_{max} \leq b/q$ the region $\mathcal{R}$ is a hypercube with edges of length l_{max}. Since the region defined by $l_{max} = b/q$ contains all the other regions with smaller values of l_{max}, we might as well set this as the lower bound for l_{max}, as considering smaller regions definitely won't give better scaling for $n(t)$. So the lower bound is: $l_{max} \geq b/q$.

Notice how we have previously seen the region $\mathcal{R}$ with $l_{max} = b/q$ for the case of $q = 3$ in Figure 14. The codespace formed from this region is shown in Figure 20. We refer to this $\mathcal{R}$ as the *Mitsubishi region*.

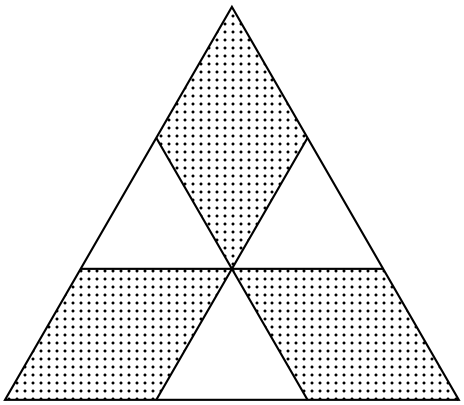


Figure 20: Codespace formed from the Mitsubishi region for $q = 3$, shown as the dotted subset of the simplex.

All other $\mathcal{R}$ interpolate between the Mitsubishi and the Sierpinski regions as we change l_{max} from b/q to $b/2$.

In the next two sections we solve an **auxiliary problem** of finding the region $\mathcal{R}$ which, for a fixed b, has the highest volume, i.e. we find the optimal ratio of l_{max}/b that achieves the highest volume of $\mathcal{R}$. Note that we maximize the volume of the region $\mathcal{R}$, not of the full simplex, since by maximizing $\mathcal{R}$ we also maximize the volume of the *subset* of the simplex that *represents our codespace*. We say that this optimal ratio achieves the highest "volume density", i.e. for any b this choice of l_{max}, recovered from the ratio, produces the highest volume of $\mathcal{R}$ compared to other values of l_{max}.

We then argue in Section 5.4 why the region $\mathcal{R}$ found as a solution to this auxiliary problem is a good approximation to the optimal $\mathcal{R}$ giving the smallest scaling $n(t)$.

5.2 Optimizing volume density for $q = 3$

Let us first consider a special case of this auxiliary problem for $q = 3$.

Problem: we aim to find l_{max} that, for a fixed b, maximizes the area of the region $\mathcal{R}$, in the case of $q = 3$.

Solution:
We have that the simplex and its region $\mathcal{R}$ have dimension $D = q - 1 = 2$. Notice how the first inequality in (80) defines a half-plane cut by a line in coordinates l_i, defined by the equation:

$$l_0 + l_1 \leq b - l_{max}, \tag{82}$$

and the rest of the inequalities in (80) define a square:

$$\begin{cases} l_0 \leq l_{max} \\ l_1 \leq l_{max} \end{cases}, \tag{83}$$

therefore any region $\mathcal{R}$ for $q = 3$ is given by a square cut by a line.

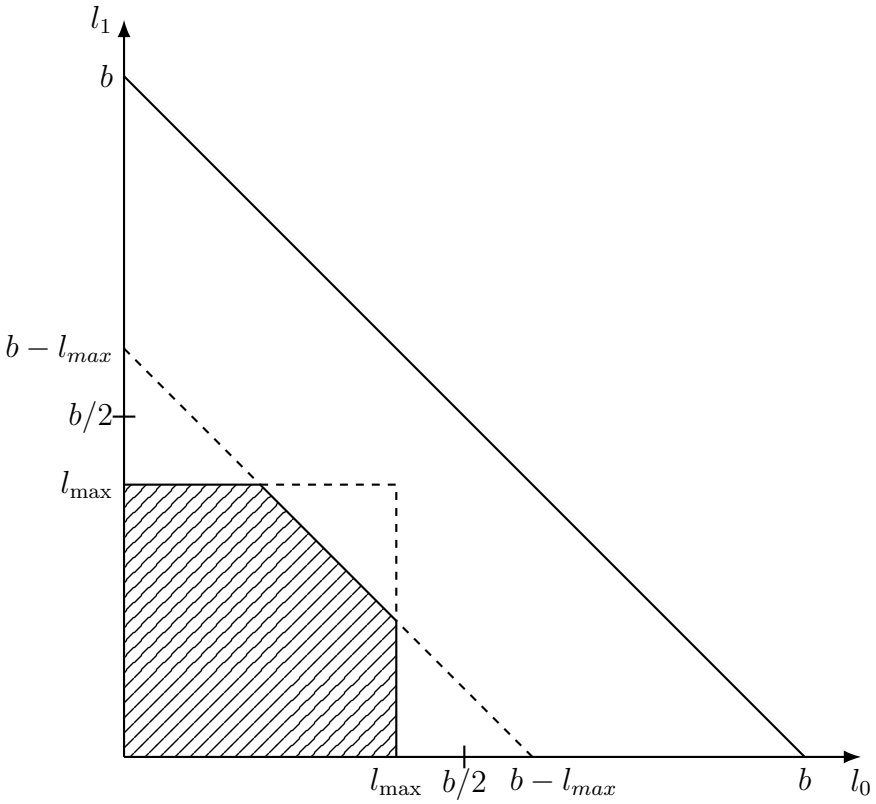
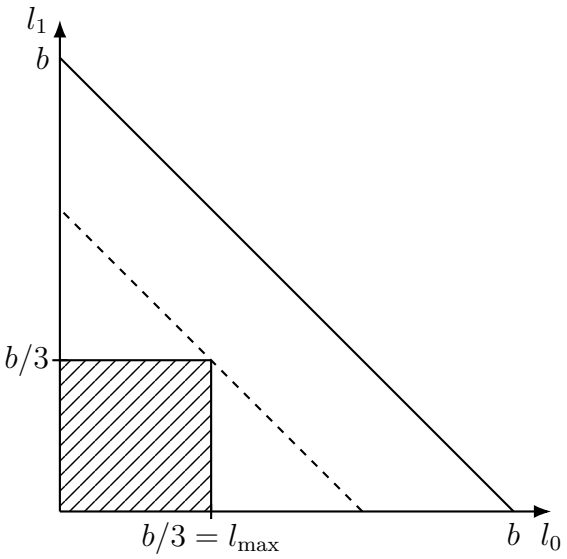
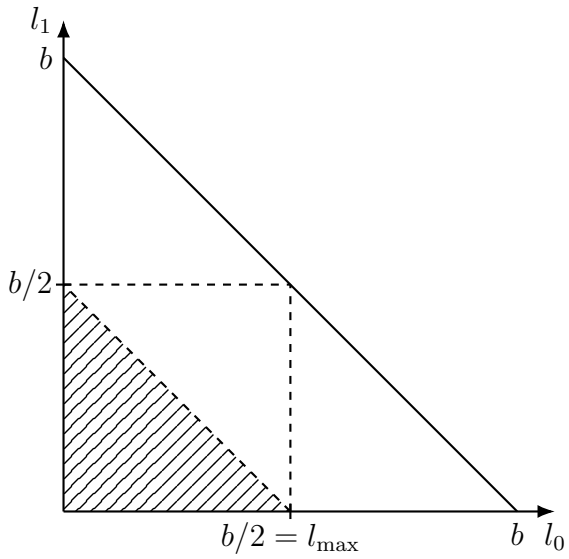


Figure 21: Any region $\mathcal{R}$ for $q = 3$ is a square at origin with edge length l_{max}, cut by a line crossing the axis at $b - l_{max}$.

Notice that the larger l_{max} is, the larger fraction of the square is being cut by the line. Indeed, for $l_{max} = b/q = b/3$ the line touches the square, and for $l_{max} = b/2$ the line cuts the square along the diagonal.



Mitsubishi region.



Sierpinski region.

Figure 22: Boundary regions presented as squares cut by lines.

For $l_{max} \in [b/3; b/2]$, we can compute the area of the shaded region in Figure 21 by simply subtracting the area of the cut triangle from the area of the square:

$$V = l_{max}^2 - (l_{max} - (b - 2l_{max}))^2 \cdot \frac{1}{2} = l_{max}^2 - (3l_{max} - b)^2 \cdot \frac{1}{2}, \tag{84}$$

$V(l_{max})$ is concave on $[b/3; b/2]$ (it's generally concave), so then to compute the maximum we have:

$$\frac{dV}{dl_{max}} = 2l_{max} - 3(3l_{max} - b) = 3b - 7l_{max} = 0 \Rightarrow l_{max} = \frac{3}{7}b \tag{85}$$

■

5.3 Optimizing volume density for arbitrary q

Problem: we aim to find l_{max} that, for a fixed b, maximizes the area of the region $\mathcal{R}$.

Solution:
For arbitrary q, it is easy to see that the $D := q - 1$-dimensional system (80) defines a hypercube cut by a hyperplane:

$$\forall \vec{l} \in \mathcal{R} : \begin{cases} l_0 + l_1 + \cdots + l_{q-2} \leq b - l_{max} \\ \forall k \in [q-1] : l_k \leq l_{max} \end{cases},$$

so first we have to compute the volume of this region. It turns out that this is not as simple as in the $q = 3$ case: in general, computing the volume of convex polytopes is as hard as computing the permanent of a matrix, i.e. this problem is #**P**-hard [DF88]. Even in the case of a hypercube cut by a single hyperplane, this problem is algorithmically difficult [Kha89]. However, there exist analytic expressions for this problem, which in certain special cases

simplify drastically. Since our equation for the hyperplane is quite simple, fortunately the volume formula simplifies in our case as well.

We can compute the volume of a hypercube cut by a hyperplane using the following formula [CK22; BS79; Pol13]:

$$\text{vol}([0,1]^D \cap H^+) = \sum_{\mathbf{v} \in F^0 \cap H^+} \frac{(-1)^{|0_\mathbf{v}|} h(\mathbf{v})^D}{D! \prod_{t=1}^D a_t}, \tag{86}$$

where:

- $D = q - 1$;
- The hypercube $[0,1]^D$ has edges of length 1;
- F^0 is the set of all vertices of the hypercube;
- $|0_\mathbf{v}|$ is the number of zeros in the entries of $\mathbf{v}$;
- The half-space H^+ clipped by the hyperplane is generally given by:

$$H^+ := \{\mathbf{l} \,|\, h(\mathbf{l}) := \mathbf{a} \cdot \mathbf{l} + r = a_0 l_0 + a_1 l_1 + \cdots + a_{D-1} l_{D-1} + r \geq 0\}, \tag{87}$$

  where, in our case, $r = \frac{b}{l_{max}} - 1$ and $\mathbf{a} = (\underbrace{-1, \ldots, -1}_{D})$.

The reason why $r = \frac{b}{l_{max}} - 1$ instead of $b - l_{max}$ like in the system (80) is because we have rescaled the hypercube by l_{max} to $[0,1]^D$ in order to fit the volume formula, so we had to rescale the hyperplane as well. The volume formula simplifies further after the substitutions:

$$\text{vol}([0,1]^D \cap H^+) = \sum_{\mathbf{v} \in F^0 \cap H^+} \frac{(-1)^{|0_\mathbf{v}|} h(\mathbf{v})^D}{D! \, (-1)^D} = \sum_{\mathbf{v} \in F^0 \cap H^+} (-1)^{|1_\mathbf{v}|} \frac{h(\mathbf{v})^D}{D!},$$

where $|1_\mathbf{v}|$ is the number of ones in the entries of $\mathbf{v}$. Note that $h(\mathbf{v})$ also simplifies for vertices:

$$h(\mathbf{v}) = \frac{b}{l_{max}} - 1 - |1_\mathbf{v}| \tag{88}$$

Now, to find the volume of the hypercube of edge size l_{max} cut by a hyperplane, we rescale the system back by l_{max}, to get:

$$V = \sum_{\mathbf{v} \in F^0 \cap H^+} (-1)^{|1_\mathbf{v}|} \frac{h(\mathbf{v})^D}{D!} l_{max}^D = \sum_{\mathbf{v} \in F^0 \cap H^+} (-1)^{|1_\mathbf{v}|} \frac{(b - (1 + |1_\mathbf{v}|) l_{max})^D}{D!} \tag{89}$$

Notice that vertices $\mathbf{v}$ of the hypercube that also belong to the half-space H^+ must satisfy $h(\mathbf{v}) = \frac{b}{l_{max}} - 1 - |1_\mathbf{v}| \geq 0$, i.e. $|1_\mathbf{v}| \leq \frac{b}{l_{max}} - 1$. Additionally, there are multiple terms in the above expression that have different $\mathbf{v}$ but same $|1_\mathbf{v}|$, specifically there are exactly $\binom{D}{|1_\mathbf{v}|}$ such terms for every $|1_\mathbf{v}|$. So we can further simplify (89) to:

$$V = \sum_{w \in \{0,1,\ldots,\lfloor r \rfloor\}} (-1)^w \frac{(b - (1 + w) l_{max})^D}{D!} \binom{D}{w} \tag{90}$$

where, again, $r = \frac{b}{l_{max}} - 1$ and $l_{max} \in [b/q, b/2]$. Therefore in this special case we have a closed form formula for the volume of a convex polytope. Now, to find l_{max} that maximizes

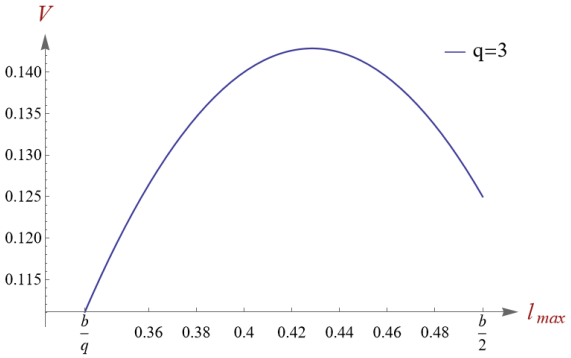
the volume, we again solve to find all extrema:

$$\frac{dV}{dl_{max}} = \sum_{w\in\{0,1,\dots,\lfloor r\rfloor\}} (-1)^{1+w}(1+w)\frac{(b-(1+w)l_{max})^{D-1}}{(D-1)!}\binom{D}{w} = 0. \tag{91}$$

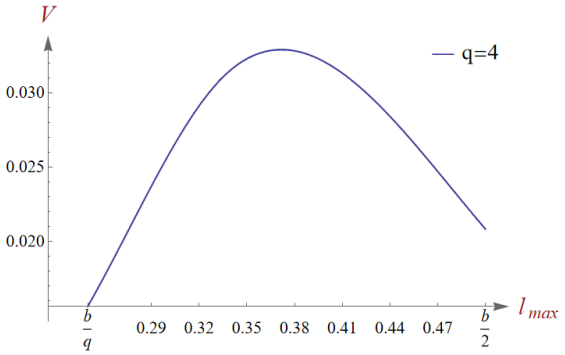
The first couple of solutions found numerically are shown in Table 3. $V(l_{max})$ is plotted in range $l_{max} \in [b/q, b/2]$ for several values of q in Figure 23

$D=2$ $(q=3)$: $\frac{l_{max}}{b} = \frac{3}{7}$	$D=3$ $(q=4)$: $\frac{l_{max}}{b} = \frac{1}{23}(11-\sqrt{6})$	$D=4$ $(q=5)$: $\frac{l_{max}}{b} = \frac{1}{3}$

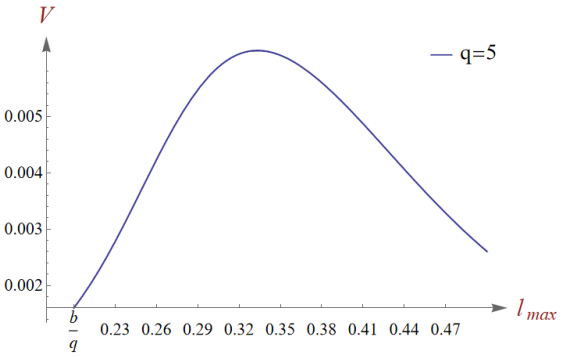
Table 3: Optimal l_{max} for $q = 3, 4, 5$.



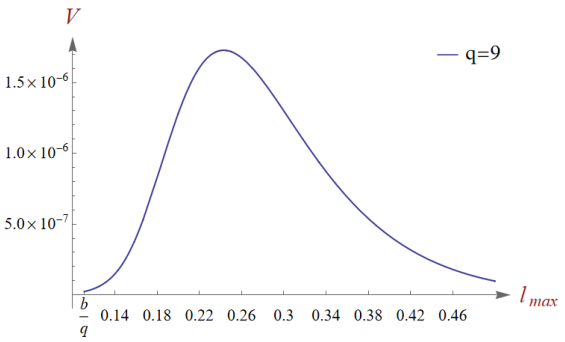
(a) $q = 3$



(b) $q = 4$



(c) $q = 5$



(d) $q = 9$

Figure 23: Plots of $V(l_{max})$ for various values of q. Notice that for arbitrary values of q, in range $l_{max} \in [b/q, b/2]$ it is apparent that $V(l_{max})$ always has one maximum, which is the solution we find when solving equation (91).

■

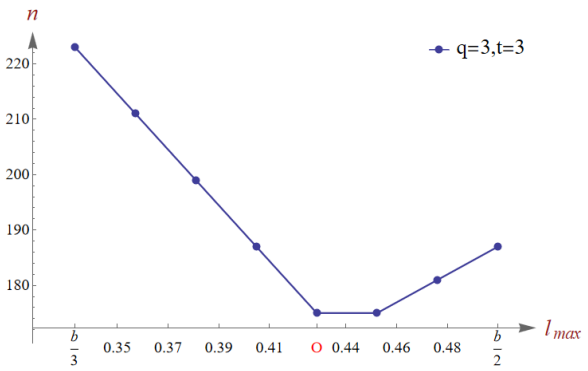
5.4 Justifying the approximation

In Sections 5.2 and 5.3 we described the optimal region in terms of giving the highest volume density. But notice that in the system of equations (51) for fixed n (so for fixed b as well) the more possible different vectors $\vec{l}$ the region contains, the more terms does each equation have. That is, the number of variables grows with the number of possible vectors $\vec{l}$ in the region. And the number of vectors clearly grows with the volume of the region $\mathcal{R}$!

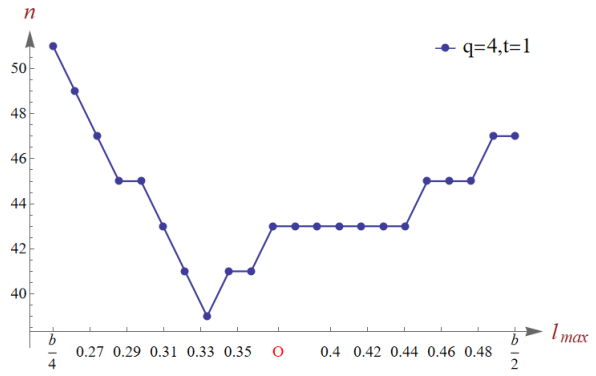
Based on this idea, we claim that regions $\mathcal{R}$ that have the highest volume density are a good approximation to the true regions that achieve the smallest scaling of $n(t)$.

Conjecture 5.1. *The region $\mathcal{R}$ found through solving the problem of optimal volume density, in the limit of $t \to \infty$ for all $q \in \mathbb{N}$ achieves the smallest possible scaling of $n(t)$ for our qudit PI code construction compared to all other choices of the region $\mathcal{R}$. Moreover, it steadily approaches the true optimal model for all finite values of t as t grows.*

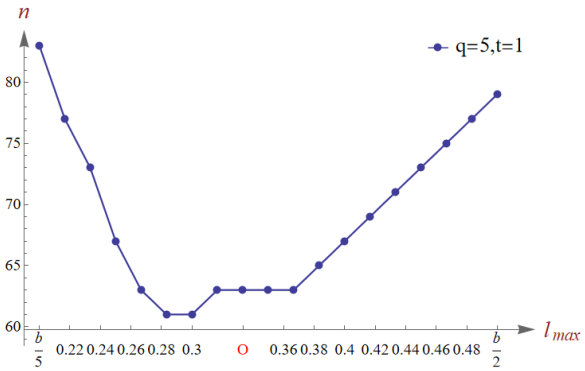
Of course, if there was a guarantee that all the equations in the system were linearly independent, this claim would obviously be true: since the number of equations in the system (51) is a function of q, d, t only, and n is a function of t and $\mathcal{R}$, there is clearly some minimal amount of variables that we have to keep non-zero for the linear program (LP) to be solvable. That is, for fixed t and fixed $\mathcal{R}$, there is a lower limit on the amount of $f(\vec{l})$ that have to be non-zero, therefore a lower limit on the volume of the region, which for a fixed l_{max} scales only with n. This means that we can achieve lower n (that still keeps the LP solvable) by choosing the right l_{max} to maximize the volume density. Unfortunately, it is non-trivial whether the equations in the system (51) are linearly independent. Nevertheless, since the coefficients in front of the variables are complicated expressions with binomial coefficients, intuitively there should be minimal linear dependence across different terms and equations. We check this conjecture numerically, and numerical evidence suggests that our assumption is correct (see Figure 24).



(a) $n(l_{max})$ at $q = 3, t = 3$, step $= 1/42$



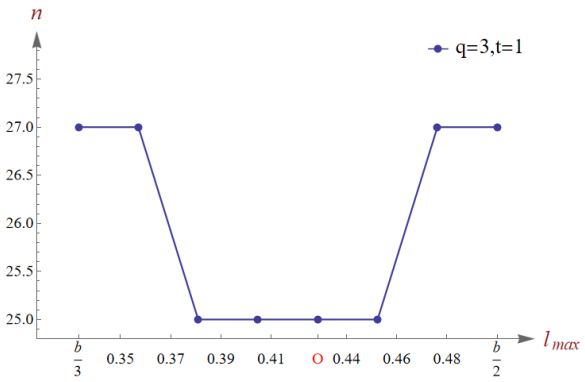
(b) $n(l_{max})$ at $q = 4, t = 1$, step $= 1/84$



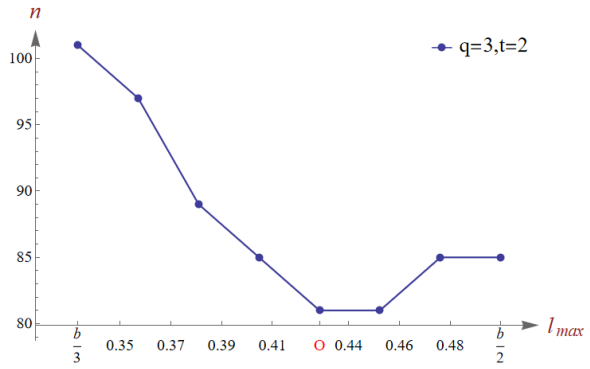
(c) $n(l_{max})$ at $q = 5, t = 1$, step $= 1/60$

Figure 24: Plots showing true values of minimal n for fixed q, t as function of l_{max}. "Step" is a parameter determining the accuracy, specifically: it is the step in l_{max} the numerical program makes when building the plot. Predicted optimal values of l_{max} from the continuous volume density problem are marked as O. As evident from the figures, we do get the expected dependence of a convex function $n(l_{max})$ with its minimum located inbetween the boundary values of b/q and $b/2$, so the opposite dependence to $V(l_{max})$. Moreover, we see that the predicted optimal values either coincide or are close to true values, as expected. The reason why they don't coincide exactly is due to the fact that: a) the equations in (51) might not be all linearly independent; b) we have a continuous region modelling a discrete one, i.e. we assume some continuous limit convergence (explained later below).

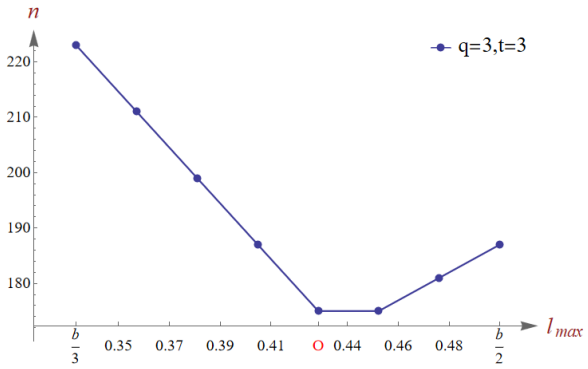
There is another more fundamental reason why we call the highest volume-density regions $\mathcal{R}$ a "good approximation" to true optimal regions. This has to do with the fact that in Sections 5.2 and 5.3 we were solving a continuous optimization problem, while the initial problem of finding an optimal region for achieving minimal scaling of $n(t)$ is discrete. In this sense, largest volume density achieved by choosing a certain l_{max} in the continuous case might not give as many terms as some other l_{max} in the discrete case. Nevertheless, the discrete case approaches the continuous limit as t grows, since the number of equations grows with t, thus requiring more variables to keep the LP solvable. Indeed, this is something that we observe numerically in Figure 25.



(a) $n(l_{max})$ at $q = 3, t = 1$



(b) $n(l_{max})$ at $q = 3, t = 2$



(c) $n(l_{max})$ at $q = 3, t = 3$

Figure 25: Plots showing true values of minimal n for fixed q, t as function of l_{max}. Each one has step $= 1/42$. Predicted optimal values of l_{max} from the continuous volume density problem are marked as O. From the figures we see that indeed as t grows, the true optimal l_{max} values converge to the predicted l_{max} values, while other values of n which were close to optimal for small t diverge from optimal as t grows. Other cases for $q > 3$ exhibit similar behaviour where, for instance in the $q = 4$ case, at $t = 2$ already $l_{max} = 1/3$ stops being the true optimal value and $\frac{1}{23}(11 - \sqrt{6})$ (i.e. predicted) becomes the true optimal value.

6 Comparing our qudit PI codes to other PI code families

In this section we aim to benchmark our qudit PI code construction, described in Section 4, against a qudit generalization of qubit PI codes by Ruskai et. al. [PR04]. Ruskai qubit PI codes have optimal block length scaling of $n(t) = 3t^2 + 3t + 1$, which makes a qudit generalization of such codes a good first candidate to benchmark our codes against. We also compare these codes to a family of qudit-to-qubit PI code family by Ouyang [Ouy17], encoding a logical qudit into physical qubits.

6.1 Qudit generalization of Ruskai qubit PI codes

Before introducing the qudit PI codes that we use as a reference to benchmark our codes 4, as a preliminary, we first introduce the qubit PI code construction by Ruskai et. al. [PR04].

Definition 6.1. *The qubit PI code family by Ruskai et. al. encoding one logical qubit into n physical qubits is defined as:*

$$\begin{aligned} |c_0\rangle &= \sum_{k=0}^{n} a_k \, |D_k^n\rangle \\ |c_1\rangle &= \sum_{k=0}^{n} b_k \, |D_k^n\rangle \end{aligned}, \tag{92}$$

where the codeword coefficients a_k, b_k satisfy the following conditions:

1. *$b_k = a_{n-k}$, that is: $|c_1\rangle = (\bigotimes_{j=1}^{n} X) \, |c_0\rangle$;*
2. *c_0 has terms with even k only, while c_1 has terms with odd k only, that is $a_{2m+1} = 0$ and $b_{2m} = 0$, where $m = \{0, 1, \ldots, \lfloor n/2 \rfloor\}$, which is equivalent to: $(\bigotimes_{j=1}^{n} Z) \, |c_i\rangle = (-1)^i \, |c_i\rangle$.*

Note that together the conditions from Definition 6.1 imply that <u>n must be odd</u>, since we have $b_k = a_{n-k}$ that must be non-zero (or zero) simultaneously, therefore for even $n - k$ (i.e. non-zero a_{n-k}) to have non-zero b_k we must have k odd, thus $n = n - k + k$ is odd.

Now, we can easily generalize this construction to qudits. The main idea would be to use the qudit version of the Pauli X and Z matrices for the qudit version of the code conditions in Definition 6.1. Specifically, we would use the Sylvester shift (X) and clock (Z) matrices [Syl09].

Definition 6.2. *For $2 \leq q \in \mathbb{N}$, the qudit generalization of Pauli X and Z matrices, also known as Sylvester shift and clock matrices respectively, are defined as the following $q \times q$*

matrices:

$$X = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 0 \end{pmatrix}$$
$$Z = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & \omega_q & 0 & \dots & 0 \\ 0 & 0 & \omega_q^2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \omega_q^{q-1} \end{pmatrix} \tag{93}$$

where $\omega_q = e^{2\pi \mathrm{i}/q}$.

We can define the qudit Ruskai PI code analogously to the Definition 6.1.

Definition 6.3. *The qudit Ruskai PI code* $\mathcal{C}$ *encoding one logical qudit of local dimension* d *into* n *physical quqits of local dimension* q*, with* $d \leq q$ *is defined as:*

$$\mathcal{C} = \{|c_i\rangle \mid \forall i \in [d] : |c_i\rangle = \sum_{\lambda \overset{q}{\vdash} n} a_{i,\lambda} |D_\lambda^n\rangle\}, \tag{94}$$

where the codeword coefficients $a_{i,\lambda}$ *must satisfy the constraints implied by the following conditions:*

1. $\forall i \in [d]$: $|c_{i+1}\rangle = (\bigotimes_{j=1}^n X) |c_i\rangle$;
2. $\forall i \in [d]$: $(\bigotimes_{j=1}^n Z) |c_i\rangle = \omega_q^i |c_i\rangle$.

The conditions from Definition 6.3 indeed imply certain restrictions on the codeword coefficients, summarized in Lemma 6.1. Before we formulate this lemma, let us define the following function:

Definition 6.4. *Modular weight of a partition* λ *of length* $l(\lambda) = q$*, where* $q \in \mathbb{N}$*, denoted as* $\mathrm{mw}(\lambda)$*, is defined as a weighted sum modulo* q *of the entries of the partition, with weights being the indices of their respective entries:*

$$\mathrm{mw}(\lambda) = (0\lambda_0 + 1\lambda_1 + 2\lambda_2 + \cdots + (q-1)\lambda_{q-1}) \mod q = \sum_{i=0}^{q-1} i\lambda_i \mod q, \;\; q := l(\lambda) \tag{95}$$

Now we can formulate the lemma using this definition:

Lemma 6.1. *For block length* $n \in \mathbb{N}$*, for all* $i \in [d]$*, and for all* $\lambda \overset{q}{\vdash} n$ *the coefficients* $a_{i,\lambda}$ *of the codewords* $|c_i\rangle$ *of the qudit Ruskai PI code are non-zero only if the modular weight of the corresponding partition,* $\mathrm{mw}(\lambda)$*, is equal to the corresponding index* i*, that is:*

$$\mathrm{mw}(\lambda) \neq i \Rightarrow a_{i,\lambda} = 0 \tag{96}$$

moreover, the block length n *of the qudit Ruskai PI code must satisfy:* $n \equiv 1 \mod q$.

Proof. This is evident from the action of the qudit X and Z on the Dicke states:

$$\begin{aligned}
&1.\ \bigotimes_{j=1}^{n} X : |D^n_\lambda\rangle \mapsto |D^n_{\lambda'}\rangle\,, \text{ where:} \\
&\lambda = \{\lambda_0, \lambda_1, \lambda_2, \ldots, \lambda_{q-1}\} \mapsto \{\lambda_{q-1}, \lambda_0, \lambda_1, \ldots, \lambda_{q-2}\} = \lambda'; \\
&2.\ \bigotimes_{j=1}^{n} Z : |D^n_\lambda\rangle \mapsto 1^{\lambda_0}\omega_q^{\lambda_1}\omega_q^{2\lambda_2}\omega_q^{3\lambda_3}\ldots\omega_q^{(q-1)\lambda_{q-1}}\,|D^n_\lambda\rangle = \omega_q^{\mathrm{mw}(\lambda)}\,|D^n_\lambda\rangle\,,
\end{aligned} \tag{97}$$

where from the second equation it is clear that to satisfy the second condition from Definition 6.3, for $|c_i\rangle$ we must have $\omega_q^{\mathrm{mw}(\lambda)} = \omega_q^i$ for all partitions $\lambda \overset{q}{\vdash} n$. This is only possible if the coefficients $a_{i,\lambda}$ that correspond to partitions λ that don't have modular weight i, $\mathrm{mw}(\lambda) \neq i$, are zero. This concludes the proof of the first statement of the lemma.

As for the second statement: notice that due to the first condition from Definition 6.3 all non-zero coefficients $a_{i,\lambda}$ are determined by the coefficients of the first codeword $a_{0,\lambda}$. Let us give a more convenient notation for the partition λ' obtained from λ by acting with an n-fold tensor product of X (equation (97)):

$$\begin{aligned}
&s_i(\lambda) = s_i(\{\lambda_0, \lambda_1, \lambda_2, \ldots, \lambda_{q-1}\}) := \{\lambda_{q-i}, \lambda_{q-i+1}, \ldots, \lambda_{q-1}, \lambda_0, \lambda_1, \ldots, \lambda_{q-i-1}\}, \\
&\text{that is, } \bigotimes_{j=1}^{n} X^i : |D^n_\lambda\rangle \mapsto |D^n_{s_i(\lambda)}\rangle
\end{aligned} \tag{98}$$

For any partition λ of such a non-zero coefficient $a_{0,\lambda}$, equations (97) together with conditions from Definition 6.3 imply the following:

$$\begin{aligned}
&i = 0 : \mathrm{mw}(\lambda) \equiv \sum_{j=0}^{q-1} j\lambda_j \equiv 0 \mod q, \\
&i = 1 : a_{1,s_1(\lambda)} = a_{0,\lambda}, \mathrm{mw}(s_1(\lambda)) \equiv \sum_{j=0}^{q-1} j\lambda_{j-1} \equiv 1 \mod q, \\
&\vdots \\
&i = q-1 : a_{q-1,s_{q-1}(\lambda)} = a_{0,\lambda}, \mathrm{mw}(s_{q-1}(\lambda)) \equiv \sum_{j=0}^{q-1} j\lambda_{j-q+1} \equiv q-1 \mod q,
\end{aligned} \tag{99}$$

where when we write λ_{j-k} we mean mod q, so $\lambda_{j-k} := \lambda_{(j-k) \bmod q}$. Finally, notice that:

$$\begin{aligned}
1 \equiv \mathrm{mw}(s_1(\lambda)) = \sum_{j=0}^{q-1} j\lambda_{j-1} = \sum_{j=0}^{q-1} j\lambda_j + \sum_{j=0}^{q-2} \lambda_j - (q-1)\lambda_{q-1} \equiv \\
\equiv 0 + n - \lambda_{q-1} - (q-1)\lambda_{q-1} = n - q\lambda_{q-1} \equiv n \Rightarrow n \equiv 1,
\end{aligned} \tag{100}$$

which concludes the proof.

■

6.2 Numerical benchmark of our PI codes for $q = 3$.

Intuitively, since Ruskai qudit PI code family was generalized from Ruskai qubit codes with optimal (for all qubit PI codes) block length scaling of $n(t) = 3t^2+3t+1$, we expect these qudit codes to have good block length scaling as well. Moreover, in many aspects this construction is similar to our codes (in the sense that codewords have certain parity properties), yet it is less restrictive. That is why we chose this family of codes as a first benchmark to our code construction from Section 4. We focus on the smallest case of $q = 3$.

In their paper Yingkai Ouyang [Ouy17] gives an explicit construction of qudit-to-qubit PI codes (i.e. from a logical qudit of arbitrary local dimension d to physical qubits of local dimension $q = 2$) with block length scaling of $n(t) = O(t^2)$. In other words, the scaling is always quadratic with t regardless of the logical local dimension d. Therefore, having more degrees of freedom and a larger local Hilbert space in the qudit-to-quqit case, i.e. $q > 2$, we expect our codes to perform as well, if not better than qudit-to-qubit codes. Specifically, for $q = 3$ we expect $n(t) \leq O(t^2)$ for both our code construction, and Ruskai qudit PI codes from Section 6.1.

Numerical approach and results

The way we determine the block length scaling $n(t)$ is analogous to the method we used in Section 3.2: for t being the weight of correctable errors, good QECC have code distance $d_C = 2t + 1$ scaling as a root of block length n: $d_C = O(n^{1/p}), p \in \mathbb{N}$. Thus, we expect $n(t)$ to scale polynomially with t.

Recall from Section 5 that for a fixed weight t of errors we want to correct, block length n, and region R defined by the parameter l_{max} (equation (80)), finding the codewords explicitly for our qudit PI codes reduces to a linear program (equation (51)). Recall also from Section 4.3 that n is defined as:

$$n = 2bt + 2t + 1 \tag{101}$$

We fix l_{max}/b to $l_{max}/b = 3/7$, which is the optimal value according to our results from Section 5. To find the block length scaling n with t of our qudit PI codes 4, for every value of t we try to solve the linear program for all possible values of n until a solution exists. As the starting point we always pick n to be the smallest n for which a solution exists for the previous value of t. This numerical method always finds the set of solutions (or declares an empty set) precisely, so there is no approximation, and for every value of t we can numerically find the smallest possible n exactly.

Finally, to find the scaling of $n(t)$ from a finite set of numerical values, we use the fact that we expect the scaling to be polynomial. A polynomial of degree p is uniquely determined by $p + 1$ points, so we try to fit our numerical data until we find the smallest p that contains all the points. Moreover, we see that n is actually determined by b (for fixed t), so to make the degree lower we actually work with numerical data for b instead, see Table 4. Even from the first five values of $b(t)$ (actually even 4 is sufficient), we are able to determine that b scales quadratically with t. Specifically, by the first 3 values for b and t, we can scaling the

equation to be:

$$b(t) = \frac{t^2}{2} + \frac{13t}{2} + 4, \tag{102}$$

t =	1	2	3	4	5
b =	11	19	28	38	49
n =	25	81	175	313	501

Table 4: First five numerical values found for the smallest b for a given t through solving a linear program. The values of n are reconstructed from b and t.

and it is easy to check that indeed the point b we find numerically for $t = 4$ (from Table 4) also belongs to the parabola, i.e. the value is perfectly predicted by the scaling in equation (102). This scaling also holds true for values beyond $t = 4$. From $b(t)$, we can easily get $n(t)$ as:

$$n(t) = t^3 + 13t + 10t + 1. \tag{103}$$

As evident from equation (103), we see that our expectation was wrong, since $n(t) = O(t^3) > O(t^2)$ for $q = 3$, performing worse than a family of qudit-to-qubit PI codes [Ouy17].

This unexpected result may either indicate that one of the restrictions in our code construction was too strong, or it might hint at a more interesting fundamental result that *high dimensional deletion errors may be hard to correct*. Of course the latter would be a much stronger implication with interesting new directions to explore. Specifically, if the latter is true, this means that the growth of degrees of freedom associated with choosing a higher local dimension is not as significant as the growth of difficulty of high dimensional deletion errors, compared to qubit deletions. In order to probe further which one of the problems is true, we have to benchmark our qudit PI codes against a family of quqit-to-qudit PI codes that we believe to have good scaling. Specifically, as mentioned earlier, we want to benchmark our codes against the PI qudit Ruskai codes.

Unfortunately, despite our best efforts of optimization, finding solutions to KL conditions 2.2 restricted to the qudit Ruskai PI code family 6.3 numerically via gradient descent takes a long time for $q > 2, t > 2$, more than 12 CPU hours for a single check of a fixed value n vs t. We save this task for future work, when we port our code and get access to run it on a cluster.

7 Discussion

In this thesis we have conducted a comprehensive analytical and numerical study of Permutationally-Invariant Quantum Error-Correcting codes (PI QEC codes, Definition 1.2).

We have derived general quantum error-correcting conditions, also known as Knill-Laflamme (KL) conditions, for qudit PI codes (Section 2) of arbitrary physical and logical local dimension, starting from the idea inspired by Aydin et. al. [AAB24] that for PI codes deletion errors are equivalent to erasure errors (see Section 2.1).

We have then performed a numerical study of solutions for the KL conditions of qubit PI quantum error-correcting codes that achieve the smallest possible block length scaling (see Section 1.1 and Definition 3.1) $n(t) = n_{min}(t)$ with the error weight t. In more common terms: such qubit PI codes have the most optimal possible scaling of their code distance compared to all other families of qubit PI QEC codes. We call qubit PI codes with this optimal block length scaling *minimal PI codes* (Definition 3.1). In this numerical study, we have used Gradient Descent methods, as well as a numerical algebraic geometry method called *Homotopy Continuation* (Section 3.1). We conjecture, based on numerical evidence, that $\mathbf{n_{min}(t) = 3t^2 + 3t + 1}$ (i.e. the qubit PI code distance is $d_C \leq d_{min} = \sqrt{12n-3}/3$), and that minimal qubit PI QEC with real coefficients have the same block-length scaling as those with complex coefficients. Moreover, we also conjecture, based on numerical evidence, that real minimal qubit PI QEC have certain symmetries in their codewords that we call *mirror* and *phase-flip* symmetries (see equation (36)). A table summarizing the code properties of all PI codes reviewed in this thesis is given below, see Table 5. To ensure reproducibility of our numerical findings, all the code is openly accessible on our GitHub repository [VB].

We then construct our own *qudit* PI code family, using codewords that could be graphically organized as points of a simplex (see Section 4). This construction works for encoding a qudit of arbitrary logical local dimension d into ququits of physical local dimension $q \geq d$. We optimize the region (subset) $\mathcal{R}$ of the simplex $\mathcal{S}$, which is the region defining the codespace of our qudit PI QEC codes, in order to achieve the smallest possible scaling of the block length $n(t)$ with the error weight t for our code construction (see Section 5). Last but not least, we compare our *qudit-to-ququit* PI QEC codes to existing *qudit-to-qubit* PI QEC codes [Ouy17], and unexpectedly find that the performance in the case $q > 2$ is worse than in the case $q = 2$. This result might suggest that *high-dimensional deletion errors are difficult to correct*. Specifically, in the sense that the leverage given by increase in degrees of freedom that a larger local Hilbert space can provide is not enough to account for the increasing difficulty of deletion errors in this high-dimensional Hilbert space. This hypothesis needs further investigation in future studies.

PI Code	d	q	**Block length scaling** $n(t)$
Minimal codes (Def. 3.1)	$d = 2$	$q = 2$	$n(t) = 3t^2 + 3t + 1$
Aydin code (Sec. 4.1, [AAB24])	$d = 2$	$q = 2$	$n(t) = 4t^2 + 2t + 1$
Ruskai code (Def. 6.1, [PR04])	$d = 2$	$q = 2$	$n(t) = 3t^2 + 3t + 1$
Ruskai qudit code (Def. 6.3)	$d \in \mathbb{N}$	$q \geq d$	Unknown
Ouyang code (Example 6.3 in [Ouy17])	$d \in \mathbb{N}$	$q = 2$	$n(t) = (2t + 1)^2(d - 1)$
Our qudit code (Section 4.3)	$d \in \mathbb{N}$	$q \geq d$	For $q = 3$: $n(t) = t^3 + 13t + 10t + 1$

Table 5: Table summarizing the properties of codes reviewed in this thesis. Here d is the logical local dimension, and q is the physical local dimension. Note that each of the PI codes in this table encodes a single qubit/qudit, and all qubit-to-qubit PI codes in this table have mirror and phase-flip symmetries.

7.1 Direct improvements to the results of this thesis

Some of the problems presented in this work were not explored to their full extent. The following are several straightforward extensions and improvements that can be made to the results of this thesis.

- Finish numerical benchmark for our codes: finding solutions to qudit Ruskai PI codes for $q = 3$ takes a long time computationally, see Section 6.2. However this task is highly parallelizable, so solving this on a cluster should be a relatively easy and quick task. Finishing this benchmark would provide further insight into whether high-dimensional deletion errors are diffucult to correct (see Section6.2 for more details on what we mean by this), or if our qudit code construction is suboptimal. We plan to finish this task in the nearest future.

- Improve our qudit PI code construction: when proving C2 in Section 4.5, specifically in the $r = s$ case, we see that $g \geq 2t$ is actually not a strict requirement, it could potentially be smaller than $2t$. This weaker condition would make the error-correction analysis more difficult. Specifically, the product of the coefficients is not automatically zero, but the sum over all coefficients should be, i.e. codewords don't have disjoint support anymore after introducing μ, ν if $g < 2t$. One could try to see what is the

smallest value g could take in terms of t, so that in C1 codewords still have disjoint support, while in C2 they satisfy orthogonality without disjoint support.

- Extend numerical results to higher values: in order to be more certain in the conjectures we make throughout the thesis. For instance, one could provide better numerical evidence for the conjectures about minimal qubit PI codes in Section 3. Or, one could find better numerical verification for the conjecture in Section 5 that justifies the use of regions $\mathcal{R}$ having optimal volume density for getting close to optimal qudit PI codes within our construction.

7.2 Ideas for future work

Several results from this thesis can lead to interesting new directions in the study of permutationally-invariant quantum error-correcting codes. Here we list some ideas possible to explore in future works.

- One of the major results of this thesis is the conjecture we make (supported by numerical evidence) that *no qubit PI QEC codes can have better block length scaling with error weight than minimal PI codes, i.e. the best scaling that can be achieved is* $n_{min} = 3t^2 + 3t + 1$, see Section 3.2. This conjecture has some strong implications, for example that good qLDPC PI QEC codes *don't exist*. It would be a major development in the study of PI codes to *analytically* prove this lower bound on block length.

- Another idea introduced in this thesis as a result of our efforts, is that *high dimensional deletion errors are difficult to correct*, see Section 6.2 for more details on what we mean by "difficult". There is not enough numerical evidence yet to trust this hypothesis, however studying this question in future work can prove useful. For example, in case the hypothesis is true, this suggests that using the extra degrees of freedom in physical local dimension is better for measuring syndromes, i.e. identifying when an erasure happened, since utilizing these extra degrees of freedom in computation would introduce high-dimensional deletion errors into the system. In the event where high-dimensional errors are inevitable, we can treat all high-dimensional Paulis acting non-trivially on more then two levels as erasure errors, since they take the system outside the two levels that we use for the qubit.

- Interestingly, the block length scaling $n(t)$ of minimal qubit codes in the Ruskai family [PR04] is the same as the scaling of general minimal qubit PI codes 3.1, specifically, $n(t) = 3t^2 + 3t + 1$. This suggests that the minimal PI code family might be equivalent to the Ruskai code family, in the sense of unitary similarity of the corresponding KL solutions. To show this, one has to find a unitary that converts between the two, such that this unitary doesn't change the KL conditions. Therefore, such a unitary has to commute with the deletion errors to preserve the KL conditions, so in the computational basis it has to separate into a tensor product of a unitary that's acting on the qubits being deleted, and a unitary acting on the rest of the qubits. Since indexing doesn't

matter for PI codes, we just say the first $2t$ are being deleted, so our unitary should look something like this in the computational basis:

$$U = U_{q^{2t}} \otimes U_{q^{n-2t}} \tag{104}$$

At the same time, we know that this unitary should preserve the symmetric subspace, therefore in the Schur basis (see [BCH06] on what a Schur basis and transform is) it should decompose as:

$$T_S U T_S^\dagger = U_{d_{Sym}} \oplus U_{q^n - d_{Sym}}. \tag{105}$$

where d_{Sym} is the size of the symmetric subspace, and T_S is the Schur transform. This is actually equivalent to just saying that in the Schur basis:

$$U_{d_{Sym}} = U_{2t} \oplus U_{d_{Sym}-2t}. \tag{106}$$

One could explore this idea further. If this is true and two families are equivalent, this would help simplify analytical proofs of lower bounds for PI codes, as one would have to show them for the Ruskai code family instead of the general case.

7.3 Further open problems

Finally, here we list a number of other open questions in the study of PI codes, which would be interesting to address. These are namely:

- Syndrome measurements for PI codes;
- Encoding and decoding for PI codes;
- Find qudit PI codes that implement a group that strictly contains the qudit Clifford group (inspired by [KT23]);
- Study how well PI codes perform in the fault-tolerant regime;
- Benchmark experimantally how well PI codes perform against deletion errors.

References

[AAB24] Arda Aydin, Max A Alekseyev, and Alexander Barg. "A family of permutationally invariant quantum codes". In: *Quantum* 8 (2024), p. 1321 (cit. on pp. 1, 3, 4, 10, 11, 19, 21, 22, 25, 29, 31, 39, 57, 58).

[AB24] Arda Aydin and Alexander Barg. "Class of codes correcting absorptions and emissions". In: *arXiv preprint arXiv:2410.03562* (2024) (cit. on p. 4).

[AP98] Uri M Ascher and Linda R Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations*. SIAM, 1998 (cit. on p. 16).

[BCH06] Dave Bacon, Isaac L. Chuang, and Aram W. Harrow. "Efficient Quantum Circuits for Schur and Clebsch-Gordan Transforms". In: *Physical Review Letters* 97.17 (Oct. 2006). DOI: 10.1103/physrevlett.97.170502. URL: http://dx.doi.org/10.1103/PhysRevLett.97.170502 (cit. on p. 60).

[BE21] Nikolas P. Breuckmann and Jens Niklas Eberhardt. "Quantum Low-Density Parity-Check Codes". In: *PRX Quantum* 2.4 (Oct. 2021). DOI: 10.1103/prxquantum.2.040101. URL: http://dx.doi.org/10.1103/PRXQuantum.2.040101 (cit. on p. 21).

[Bro+24] Benjamin L Brock, Shraddha Singh, Alec Eickbusch, Volodymyr V Sivak, Andy Z Ding, Luigi Frunzio, Steven M Girvin, and Michel H Devoret. "Quantum Error Correction of Qudits Beyond Break-even". In: *arXiv preprint arXiv:2409.15065* (2024) (cit. on p. 3).

[BS79] D. L. Barrow and P. W. Smith. "Spline Notation Applied to a Volume Problem". In: *The American Mathematical Monthly* 86.1 (1979), pp. 50–51. DOI: 10.1080/00029890.1979.11994730. eprint: https://doi.org/10.1080/00029890.1979.11994730. URL: https://doi.org/10.1080/00029890.1979.11994730 (cit. on p. 47).

[BT] Paul Breiding and Sascha Timme. *The basics of the theory and techniques behind HomotopyContinuation.jl*. URL: https://www.juliahomotopycontinuation.org/guides/introduction (cit. on pp. 15, 16).

[BT17] Paul Breiding and Sascha Timme. "HomotopyContinuation.jl - a package for solving systems of polynomial equations in Julia". In: *CoRR* abs/1711.10911 (2017). arXiv: 1711.10911. URL: http://arxiv.org/abs/1711.10911 (cit. on pp. 15, 17).

[BW08] Rainer Blatt and David Wineland. "Entangled states of trapped atomic ions". In: *Nature* 453.7198 (2008), pp. 1008–1015 (cit. on p. 3).

[CDBO19] Daniele Cozzolino, Beatrice Da Lio, Davide Bacco, and Leif Katsuo Oxenløwe. "High-dimensional quantum communication: benefits, progress, and future challenges". In: *Advanced Quantum Technologies* 2.12 (2019), p. 1900038 (cit. on p. 3).

[CK22] Yunhi Cho and Seonhwa Kim. *Volume of Hypercubes Clipped by Hyperplanes and Combinatorial Identities*. 2022. arXiv: 1512.07768 [math.CO]. URL: https://arxiv.org/abs/1512.07768 (cit. on p. 47).

[CNHM03] Irinel Chiorescu, Y Nakamura, CJP Ma Harmans, and JE Mooij. "Coherent quantum dynamics of a superconducting flux qubit". In: *Science* 299.5614 (2003), pp. 1869–1871 (cit. on p. 3).

[DF88] M. E. Dyer and A. M. Frieze. "On the Complexity of Computing the Volume of a Polyhedron". In: *SIAM Journal on Computing* 17.5 (1988), pp. 967–974. DOI: 10.1137/0217060. eprint: https://doi.org/10.1137/0217060. URL: https://doi.org/10.1137/0217060 (cit. on p. 46).

[Ful74] W. Fulton. *Algebraic Curves: An Introduction to Algebraic Geometry*. Math Lecture Notes Series. Addison-Wesley Longman, Incorporated, 1974. URL: https://books.google.nl/books?id=jqsY0QEACAAJ (cit. on p. 17).

[Got24] Daniel Gottesman. "Surviving as a quantum computer in a classical world". In: *Textbook manuscript preprint* (2024). URL: https://www.cs.umd.edu/class/spring2024/cmsc858G/QECCbook-2024-ch1-15.pdf (cit. on pp. 3, 4, 7–9, 65).

[Got97] Daniel Gottesman. *Stabilizer Codes and Quantum Error Correction*. 1997. arXiv: quant-ph/9705052 [quant-ph]. URL: https://arxiv.org/abs/quant-ph/9705052 (cit. on p. 3).

[HDD05] Erik Hostens, Jeroen Dehaene, and Bart De Moor. "Stabilizer states and Clifford operations for systems of arbitrary dimensions and modular arithmetic". In: *Physical Review A—Atomic, Molecular, and Optical Physics* 71.4 (2005), p. 042315 (cit. on p. 3).

[Kha89] Leonid Genrikhovich Khachiyan. "The problem of calculating the volume of a polyhedron is enumerably hard". In: *Russian Mathematical Surveys* 44.3 (1989), pp. 199–200. URL: https://doi.org/10.1070/RM1989v044n03ABEH002136 (cit. on p. 46).

[KL97] Emanuel Knill and Raymond Laflamme. "Theory of quantum error-correcting codes". In: *Physical Review A* 55.2 (1997), p. 900 (cit. on p. 8).

[KNXSF20] Evgeniy O Kiktenko, Anastasiia S Nikolaeva, Peng Xu, Georgy V Shlyapnikov, and Arkady K Fedorov. "Scalable quantum computing with qudits on a graph". In: *Physical Review A* 101.2 (2020), p. 022304 (cit. on p. 3).

[KT23] Eric Kubischta and Ian Teixeira. "Family of Quantum Codes with Exotic Transversal Gates". In: *Physical Review Letters* 131.24 (2023), p. 240601 (cit. on pp. 4, 60).

[Mil09] GJ Milburn. "Photons as qubits". In: *Physica Scripta* 2009.T137 (2009), p. 014003 (cit. on p. 3).

[NC10] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010 (cit. on p. 6).

[Nie05] Michael A Nielsen. "A geometric approach to quantum circuit lower bounds". In: *arXiv preprint quant-ph/0502070* (2005) (cit. on p. 3).

[OB24] Y Ouyang and GK Brennen. "Finite-round quantum error correction on symmetric quantum sensors (2024)". In: *arXiv preprint arXiv:2212.06285* (2024) (cit. on p. 3).

[Ouy14] Yingkai Ouyang. "Permutation-invariant quantum codes". In: *Physical Review A* 90.6 (2014), p. 062317 (cit. on pp. 3, 4, 21, 22).

[Ouy17] Yingkai Ouyang. "Permutation-invariant qudit codes from polynomials". In: *Linear Algebra and its Applications* 532 (2017), pp. 43–59 (cit. on pp. 4, 52, 55–58).

[Pol13] Georg Polya. "Berechnung eines bestimmten Integrals". In: *Mathematische Annalen* 74.2 (1913), pp. 204–212. URL: `https://link.springer.com/article/10.1007/BF01456040#citeas` (cit. on p. 47).

[PR04] Harriet Pollatsek and Mary Beth Ruskai. "Permutationally invariant codes for quantum error correction". In: *Linear algebra and its applications* 392 (2004), pp. 255–288 (cit. on pp. 1, 3, 21, 22, 52, 58, 59).

[RRG07] TC Ralph, KJ Resch, and Alexei Gilchrist. "Efficient Toffoli gates using qudits". In: *Physical Review A—Atomic, Molecular, and Optical Physics* 75.2 (2007), p. 022313 (cit. on p. 3).

[SO21] Taro Shibayama and Yingkai Ouyang. "The equivalence between correctability of deletions and insertions of separable states in quantum codes". In: *2021 IEEE Information Theory Workshop (ITW)*. IEEE. 2021, pp. 1–6 (cit. on p. 11).

[SWM10] Mark Saffman, Thad G Walker, and Klaus Mølmer. "Quantum information with Rydberg atoms". In: *Reviews of modern physics* 82.3 (2010), pp. 2313–2363 (cit. on p. 3).

[Syl09] James Joseph Sylvester. *The Collected Mathematical Papers of James Joseph Sylvester*. Cambridge University Press, 1909 (cit. on p. 52).

[UG24] Robert Frederik Uy and Dorian A Gangloff. "Qudit-based quantum error-correcting codes from irreducible representations of SU (d)". In: *arXiv preprint arXiv:2410.02407* (2024) (cit. on p. 3).

[VB] Vladyslav Visnevskyi and Liam Bond. *A numerical study of qubit and qudit Permutationally-Invariant quantum error-correcting codes.* URL: `https://github.com/Techtonick/PI-codes` (cit. on pp. 4, 30, 57).

[VE19] Lieven MK Vandersypen and Mark A Eriksson. "Quantum computing with semiconductor spins". In: *Physics Today* 72.8 (2019), pp. 38–45 (cit. on p. 3).

[WHSK20] Yuchen Wang, Zixuan Hu, Barry C Sanders, and Sabre Kais. "Qudits and high-dimensional quantum computing". In: *Frontiers in Physics* 8 (2020), p. 589504 (cit. on p. 3).

A Proofs of some theorems and lemmas

Proof of Theorem 1.1:

Proof. We outline the idea behind the proof here, for full proof, see Theorem 2.4 in [Got24]. The basic idea is to just straightforwardly use the Definition 1.3 of quantum error-correcting codes. $\mathcal{C}$ has an encoding partial isometry U and a decoding map $\mathscr{D}$. The general idea is as follows: take the decoder map $\mathscr{D}$ associated with the code $\mathcal{C}$, purify the decoder map to a unitary V so that we are working with unitary maps only (we can then trace out the unnecessary part, and we can always purify due to Stinespring dilation). Then, we have that QECC $(\mathcal{C}, \mathscr{E})$ corrects $E, F \in \mathscr{E}$, so then:

$$V(E(U|\psi\rangle)_G \otimes |0\rangle_D) = c_E |\psi\rangle_L \otimes |s_E\rangle_{D'}, \tag{107}$$

$$V(F(U|\psi\rangle)_G \otimes |0\rangle_D) = c_F |\psi\rangle_L \otimes |s_F\rangle_{D'}, \tag{108}$$

where recall that we denoted $\mathcal{H}_G$ as the corrupted Hilbert space after acting with errors, $\mathcal{H}_D$ here is the space of auxiliary qudits we add to purify the decoder, and $\mathcal{H}_{D'}$ is the space we trace out at the end to get back to the initial logical Hilbert space $\mathcal{H}_L$ we started from. Now just by linearity:

$$V[(\alpha E + \beta F)|\bar{\psi}\rangle_G \otimes |0\rangle_D] = \alpha c_E |\psi\rangle_L \otimes |s_E\rangle_{D'} + \beta c_F |\psi\rangle_L \otimes |s_F\rangle_{D'} = \\ = |\psi\rangle_L \otimes (\alpha c_E |s_E\rangle_{D'} + \beta c_F |s_F\rangle_{D'}), \tag{109}$$

where after tracing out D', the error coefficient near the final qudit state $|\psi\rangle_L$ turns out to be of the desired form according to the definition, see Theorem 2.4 in [Got24] for further details. ■

Proof of Theorem 1.4:

Proof. We use an argument inspired by [Got24]: suppose we construct a set of QECC's which all have the same encoder, and thus the same codespace, but each code is associated with a different error-set, depending on where the erasures will take place. Therefore, each code would also have its own decoder. When decoding, we now have classical information telling us exactly on which quqits the erasures took place, so we can choose an appropriate decoder based on the actual error set, and use it. We label the actual error set as $\mathscr{E}_A$, with support on the set A of at most t quqits. Then in order for this code to be error-correcting, it must satisfy the KL conditions with $\mathscr{E}_A$ as the correctable set of errors. But, notice that we actually have $\mathscr{E}_A^2 = \mathscr{E}_A$, since the product of two errors with support on A is still an error with support on A. In a way, using this classical information we have "localized" our correctable error set. Now, for a distance d code, all errors of weight $d-1$ are detectable, so when $t \leq d-1$ we have that $\mathscr{E}_A \subseteq \mathscr{E}_D$, so then $\mathscr{E}_A^2 = \mathscr{E}_A \subseteq \mathscr{E}_D$, therefore $\mathscr{E}_A^2 \subseteq \mathscr{E}_D$ and the KL conditions are satisfied for up to $t \leq d-1$ erasure errors, which concludes the proof. ■

Proof of Lemma 2.1:

Proof.

$$A^{n-t+1}_{i_1,x_1} \circ A^{n-t+2}_{i_2,x_2} \circ \cdots \circ A^{n}_{i_t,x_t} = A^{n-t+1}_{i_1,x_1} \circ A^{n-t+2}_{i_2,x_2} \circ \cdots \circ \left(\langle x_t |_{i_t} \otimes \mathbb{1}_{\{1,\ldots,n\}\setminus\{i_t\}} \right) =$$

$$= \left(\langle x_1 |_{i_1} \otimes \mathbb{1}_{\{1,\ldots,n\}\setminus\{i_1,i_2,\ldots,i_t\}} \right) \circ \left(\langle x_2 |_{i_2} \otimes \mathbb{1}_{\{1,\ldots,n\}\setminus\{i_2,\ldots,i_t\}} \right) \circ \cdots \circ \left(\langle x_t |_{i_t} \otimes \mathbb{1}_{\{1,\ldots,n\}\setminus\{i_t\}} \right) =$$

$$= \langle x |_I \otimes \mathbb{1}_{\{1,\ldots,n\}\setminus I} = A^n_{I,x} \tag{110}$$

■

Proof of Lemma 2.6:

Proof.

$$E^q_\mu \, |D^n_\lambda\rangle = (\chi_{q-1})^{\mu_{q-1}} \circ \cdots \circ (\chi_0)^{\mu_0} \, |D^n_\lambda\rangle = \sqrt{\frac{\binom{n-\mu_0-\mu_1-\cdots-\mu_{q-1}}{\{\lambda_0-\mu_0,\lambda_1-\mu_1,\ldots,\lambda_{q-1}-\mu_{q-1}\}}}{\binom{n}{\lambda}}} \, |D^{n-a}_{\{\lambda_0-\mu_0,\ldots,\lambda_{q-1}-\mu_{q-1}\}}\rangle =$$

$$= \sqrt{\frac{\binom{n-t}{\lambda-\mu}}{\binom{n}{\lambda}}} \, |D^{n-t}_{\lambda-\mu}\rangle \tag{111}$$

■