Complexité du décodage des codes stabilisateurs quantiques

Hardness of decoding stabilizer codes

par

Pavithran Iyer Sridharan

Mémoire présenté au département de physique
en vue de l'obtention du grade de maître ès sciences (M.Sc.)

FACULTÉ des SCIENCES
UNIVERSITÉ de SHERBROOKE

Sherbrooke, Québec, Canada, 18 juillet 2014

Le 18 Juillet 2014

le jury a accepté le mémoire de Monsieur Pavithran Iyer Sridharan dans sa version finale.

Membres du jury

Professeur David Poulin
Directeur de recherche
Département de physique

Professeur Alexandre Blais
Membre interne
Département de physique

Professeur André-Marie Tremblay
Président rapporteur
Département de physique

À mes parents et amis ...

To my parents and friends ...

Sommaire

Ce mémoire porte sur l'étude de la complexité du problème du décodage des codes stabilisateurs quantiques. Les trois premiers chapitres introduisent les notions nécessaires pour comprendre notre résultat principal. D'abord, nous rappelons les bases de la théorie de la complexité et illustrons les concepts qui s'y rattachent à l'aide d'exemples tirés de la physique. Ensuite, nous expliquons le problème du décodage des codes correcteurs classiques. Nous considérons les codes linéaires sur le canal binaire symétrique et nous discutons du célèbre résultat de McEliece *et al.* [1].

Dans le troisième chapitre, nous étudions le problème de la communication quantique sur des canaux de Pauli. Dans ce chapitre, nous introduisons le formalisme des codes stabilisateur pour étudier la correction d'erreur quantique et mettons en évidence le concept de dégénérescence. Le problème de décodage des codes stabilisateurs quantiques négligeant la dégénérescence est appelé «*quantum maximum likelihood decoding*»(QMLD). Il a été démontré que ce problème est NP-complet par Min Hseiu Heish *et al.*, dans [2]. Nous nous concentrons sur la stratégie optimale de décodage, appelée «*degenerate quantum maximum likelihood decoding*»(DQMLD), qui prend en compte la présence de la dégénérescence et nous mettons en évidence quelques instances pour lesquelles les performances de ces deux méthodes diffèrent drastiquement.

La contribution principale de ce mémoire est de prouver que DQMLD est considérablement plus difficile que ce que les résultats précédents indiquaient. Dans le dernier chapitre, nous présentons notre résultat principal (Thm. 5.1.1), établissant que DQMLD est #P-complet. Pour le prouver, nous démontrons que le problème de l'évaluation de l'énumérateur de poids d'un code linéaire, qui est #P-complet, se réduit au problème DQMLD. Le résultat principal de ce mémoire est présenté sour forme d'article dans [3] et est présentement considéré pour publication dans *IEEE Transactions on Information Theory*. Nous montrons également que, sous certaines conditions, les résultats de QMLD et DQMLD coïncident. Il s'agit d'une amélioration par rapport aux résultat obtenus dans [4, 5].

Summary

This thesis deals with the study of computational complexity of decoding stabilizer codes. The first three chapters contain all the necessary background to understand the main result of this thesis. First, we explain the necessary notions in computational complexity, introducing P , NP , #P classes of problems, along with some examples intended for physicists. Then, we explain the decoding problem in classical error correction, for linear codes on the binary symmetric channel and discuss the celebrated result of Mcleicee et al., in [1].

In the third chapter, we study the problem of quantum communication, over Pauli channels. Here, using the stabilizer formalism, we discuss the concept of degenerate errors. The decoding problem for stabilizer codes, which simply neglects the presence of degenerate errors, is called *quantum maximum likelihood decoding* (QMLD) and it was shown to be NP-complete, by Min Hseiu Heish et al., in [2]. We focus on the problem of optimal decoding, called *degenerate quantum maximum likelihood decoding* (DQMLD), which accounts for the presence of degenerate errors. We will highlight some instances of stabilizer codes, where the presence of degenerate errors causes drastic variations between the performances of DQMLD and QMLD.

The main contribution of this thesis is to demonstrate that the optimal decoding problem for stabilizer codes is much harder than what the previous results had anticipated. In the last chapter, we present our own result (in Thm. 5.1.1), establishing that the optimal decoding problem for stabilizer codes, is #P-complete. To prove this, we demonstrate that the problem of evaluating the weight enumerator of a binary linear code, which is #P-complete, can be reduced (in polynomial time) to the DQMLD problem, see (Sec. 5.1). Our principal result is also presented as an article in [3], which is currently under review for publication in *IEEE Transactions on Information Theory*.

In addition to the main result, we also show that under certain conditions, the outputs of DQMLD and QMLD always agree. We consider the conditions developed by us to be an improvement over the ones in [4, 5].

Acknowledgements

I would like to thank my research supervisor, Pr. David Poulin, for several reasons. First, I want to thank my supervisor for facilitating my admission in his research team and his patient guidance on the various concepts in quantum error correction, that I have learnt in the due course of my masters program. Second, I want to thank my supervisor for helping me to greatly improve my way of writing and presentation. Lastly, I thank my supervisor very much for bearing the inconveniences that resulted due to my handling of the various unconventional administrative difficulties, which were involved with my masters program.

I am grateful to Pr. Patrick Fournier for his extensive help with which I found it easy to complete the formal requirements of the graduate program, at my home institution. Thanks to his great efforts, I never felt that my poor proficiency in French was a barrier to my comprehension of the graduate courses. I owe a big thank you to Dominique Parisé for her kind support and for patiently fulfilling all of my logistic requirements, at my home institution. I also want to thank Guillaume Duclos-Cianci for useful discussions and careful reading of the manuscript containing our main result of this thesis.

Eventually, I would like to highlight the encouraging support, given by my family and my dear friends from India, without which, it would have been impossible for me to reach the end of my masters program.

Contents

Chapter 1

Introduction

The last decades have witnessed a rapid growth in information technology, with the advent of telecommunications. Much of the great developments would be inconceivable without transmitting large amounts of digital information over very long distances. However, an important question has kept the scientists and engineers busy till today – How do we communicate reliably with the presence of noise in the communication channel ? Noise, or errors, can be of various types, depending upon their source. In practice, communication errors can refer to any disturbances of the communication channel, they can range from atmospheric disturbances like thunder to human adversaries like enemy spies. In the case of digital communication, the role of noise can be modelled as an operation that cause some bits in a long binary sequence to unwontedly *flip* (change). All types of errors in a communication channel are studied under a unified framework which specifies a few protocols the sender and the receiver must adopt, in order to achieve reliable communication. In the 1940's, the famous american mathematician Richard Hamming pioneered in developing a framework for correcting errors on digital data received from a communication channel and this framework is now studied by the name of *Error correcting codes*. The first error correcting code[1] to be discovered was called the *Hamming code*, see [7, 6].

We will describe the theory of error correction to non-experts with a simple example. See (Chap. 3) for a formal and detailed discussion. Suppose a message sequence 11011 is to be is sent from one end of a channel to another. In a scenario, the receiver receives a sequence 11001, but is unaware of what sequence was actually transmitted by the sender. If the former

[1] Historically, error correcting codes were not invented to achieve reliable communication across a channel, since telecommunication technology was not developed until the decade of the 1990s. They were discovered for the purpose of correcting errors that occur during the storage of output processed by mainframe computers. See [6] for details.

and latter sequences correspond to different messages, there can be a serious problem in the receiver's understanding of the sender. A method of error correction can be described as follows. We will *repeat the message bits a sufficient number of times so that the majority of bits are unaffected*. In our example, the sender will send 111111000111111 instead of 11011, where every bit is now repeated three times. This sequence prepared by the sender is called an *encoded sequence*. Due to the presence errors in the communication channel, the receiver may not receive the same encoded sequence as the one which was sent. The sequence which the receiver gets will be *corrupted* – it will differ from the encoded sequence at some unknown location. The task of the receiver, which is referred to as *decoding*, is to determine the encoded sequence from the received corrupted sequence. Again in our example, the receiver will look at every block of three bits, to conclude the corresponding value of the message bit, using a simple majority vote. Now, even if one of the 15 bits in the encoded sequence is changed from 0 to 1 or vice versa, the majority of bits in every block (of three bits) will retain their original value and therefore the single bit flip will not affect the inference of the receiver. What we have described is one possible method of error correction.

But adding *redundant bits* certainly comes at a cost. Firstly, the relevant communication technology must be scaled to handle 15 bits in a sequence as opposed to five, yet only a $5-$bit message is transmitted, indicating a *transmission rate* of $5/15$ or 33% as opposed to 100% in the unencoded case. Carrying forward this idea, if one aims to communicate with a channel that causes four bits (at unknown locations) to flip, then we must repeat every bit in the message nine times, thereby sending 45 bits to transmit a $5-$bit message, the transmission rate will be 11%. For various reasons, one prefers to adopt an encoding technique with a higher transmission rate. However, this may simply indicate that naively repeating each bit of the message many times is not the best method of error correction. The central problem in error correction still remains – what is the *best way* of encoding a message such that errors during transmission do not change the receiver's inference of the message ?

In his seminal paper [8] that led to the birth of information theory, the american mathematician Claude Shannon in 1964 developed much of the formalism that is used today to assess the usefulness of an encoding technique. In [8] Shannon also provided an upper bound on the maximum rate of transmission that can ever be achieved with any encoding technique. However, his proof only partially solved the problem – he did not provide a specific construction or description of an encoding technique which will achieve the maximum possible rate of transmission for a given channel. The explicit construction of an encoding technique was indeed essential to formulate a technique that the receiver must adopt to infer the original encoded sequence from the received sequence.

Thirty years after this discovery of Shannon, an american mathematician Robert McEliece,

along with Elwyn Berlekamp and Van Tilborg in 1978, demonstrated in [1] that the types of encoding techniques that achieve the maximum possible transmission rate according to Shannon, cannot possibly be supplemented by an *efficient* decoding technique. In other words, the corresponding *decoding problem* (inferring the transmitted sequence from the received data) is a theoretically intractable problem in computer science. This was a huge negative result[2] which seemed to imply that though there are encoding techniques which will have the maximum possible transmission rate, they cannot be used in any applications. In (Chap. 3) we will develop the necessary formalism of error correction (while referring to standard texts in classical information theory and error correction such as [10, 11, 12, 13, 8]) in order to understand the result of McEliece et al. in [1, 14]. The result in [1] connects the study of computational complexity of problems in computer science to the decoding problem of error correction.

To understand the result in [1] completely, one must be familiar with the classification of problems in computer science into various classes based on their *computational hardness*. Such a classification was developed in 1965-1972 by various eminent computer scientists, see [15, 16] and for the sake of completeness, it is introduced in (Chap. 2). See any standard text on computational complexity such as [17, 18, 19] for a deeper study. It is worthwhile to mention that a lot of the formalism developed to study the computational complexity of problems in theoretical computer science, have been very useful in understanding fundamental problems in physics too, see (Ex. 2.2.2, 2.4.3 and [20]) for some famous examples. The purpose of (Chap. 3) besides introducing various terms in classical coding theory, is to convey the message that in spirit, classical error correction is an intractable problem. At the end of (Chap. 3), we formally close the discussion on classical error correction and consider a new problem in the forthcoming chapters, that concerns communication using quantum information, which brings us to the topic of *quantum error correction*.

In 1984 the american physicist Richard Feynman conjectured that since any conceivable simulation of quantum mechanics using classical computers is extremely tedious, the quantum mechanical degrees of freedom of a particle can be manipulated to perform useful computations. The system consisting of some particles whose quantum mechanical properties can be exploited to perform useful computations, is called a *quantum computer*. For a general reading, see a compilation of lectures by Feynman on this subject in [21] and an introductory text by Scott Aaranson in [22]. For a detailed study, see [23, 24]. Analogous

[2]Exactly in between these two important discoveries, an american computer scientist Robert Gallager in 1962 published in [9] his PhD thesis, a type of codes which have a transmission rate that is *almost* equal to the maximum possible rate any code can achieve as per Shannon's claims. Furthermore the same article, Gallager also provided an efficient algorithm which can be used to decode these types of codes. Surprisingly, his discovery was largely ignored by the research community for the next 30 years.

to a bit, a unit of quantum information is called a *qubit* and its quantum state is given by $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, the (spin) state of a simple spin-half particle, see any standard reference in quantum mechanics like [25].

In the next few years after Feynman, some of these ideas were developed leading to the first milestone in *Quantum Computation*, set by Peter Shor in 1994. He proposed in [26] an algorithm to factorize a positive integer, by suitably manipulating qubits and adhering to the laws of quantum mechanics. This famous algorithm goes by the name of *Shor's factorization algorithm* and to everyone's surprise, this algorithm is still far more efficient that any integer factorization algorithm known to computer scientists[3]. This popularized the prospect of quantum computation techniques greatly. Unlike its classical counterpart, developing a technology to realize a quantum computer in a laboratory is a quixotic project, see [27, 28, 29]. The biggest barrier for the experimental realization of quantum computation techniques is that quantum information is very sensitive to noise. In a classical channel, a bit can be changed to 0 if it was initially 1 and changed to 1 if it was initially 0. In the case of a qubit, the rules of quantum mechanics allows for any transformation on the qubit given by a unitary matrix. If a qubit is sent as $|0\rangle$, the receiver can find the qubit to be in one amongst an infinite set of possible states each resulting from applying some unitary matrix to $|0\rangle$. There is an even more serious problem if one resorts to repeating the qubit multiple times – the *no cloning* principle of quantum mechanics prevents this possibility, see [23]. For these reasons, many believed that it was impossible to perform quantum error correction. However, it was again Peter Shor[4] who provided a framework to perform error correction on quantum information. This was the biggest breakthrough in quantum error correction, Peter Shor provided an encoding where each qubit is expressed as a highly entangled state of nine qubits. This encoding technique is popularly known as the *Shor's nine qubit code*, cf. (Sec. 4.8). Many proposals have been made to perform quantum error correction and to incorporate techniques with quantum computing. For a compendium of the experimental research hitherto, cf. Chapter 7 of [23] and the references therein.

Though it was possible in principle to encode a set of data qubits and send them across a channel, the whole formalism was particularly messy and tedious. In 1997, Daniel Gottesman in his PhD thesis in [30] provided a formalism called the *stabilizer formalism* of quantum error correction, which provided a greatly simplified understanding of the operations on quantum states in the error correction procedure. The stabilizer formalism is by far the most widely studied methods of performing quantum error correction, see [23, 31, 32]. This

[3] In fact, integer factorization is the only instance of a problem, to this day, where a quantum algorithm offers an exponential speedup in runtime as compared to any known classical algorithm. See [17, 23] for details.

[4] Calderbank and Steane also independently provided a protocol for quantum error correction, shortly after Peter Shor see [23] for details.

formalism was particularly useful for quantum error correction because the entire process of error correction was formulated with no direct reference to the particular structure of the encoded sequence of qubits. In (Chap. 4) we will introduce this formalism and describe the central problem of quantum error correction.

The striking distinction between quantum error correction and classical error correction, can be pointed out very well through the stabilizer formalism. Consider the state $|0\rangle$ transmitted across a channel, suppose the error during transmission is Z given by $Z = \mathsf{diag}(1, -1)$. Since $Z|0\rangle = |0\rangle$, the state will simply pass through the channel as though nothing happened to it. That a non identity operation on the qubit still results in the same transformation as an identity operation, is a peculiar feature of quantum noise which is absent in classical noise. It seems like a blessing in disguise for error correction applications. Such types of errors are called *degenerate errors*, see also [24, 23, 2, 31]. However, it turns out that if one makes changes in the decoding strategy by acknowledging the presence of degenerate errors, this will cause the decoding algorithm for quantum codes to be fundamentally different from its classical counterpart. In (Chap. 4) we will formulate a *optimal* decoding strategy for stabilizer codes, which accounts for degenerate errors in the channel. Subsequently, we will also explain how decrease errors affect the decoding strategy. See also [33, 34, 35, 36].

At this point, a very important question emerges. *How hard is the problem of decoding a quantum error correcting code ?* This question was partially answered by Min Hseiu Heish and Francois Le Gall, who showed in 2011 in [2] that quantum error correction ignoring the presence of degenerate errors is a problem which is equally as hard as the classical error correction problem.

This brings us to the question at the core of this thesis. *Is optimal decoding of a quantum error correcting code harder than decoding a classical error correcting code ?*

The main result derived in this thesis answers the above question by proving that performing optimal quantum error correction, i.e, decoding while accounting for degenerate errors, is much harder than classical error correction or decoding while ignoring degenerate errors. This proof is detailed in (Chap. 5) and also presented in the article in [3]. A consequence of our result can be stated in the following appealing manner. An algorithm that can perform optimal quantum error correction, is so powerful that it can accomplish all the computation tasks of an infinite number of classical error correction algorithms.

Lastly, we have a discussion in (Chap. 6) on the scope of further research on the computational complexity of quantum error correction problems.

Chapter 2

Computational Complexity

Problems in computer science are classified based on the time and space resources required to solve them. There are two issues to be overcome to accomplish such a classification – a precise model of computation needs to be defined to specify the time taken for an operation in this model and then any classification must not depend strongly on the machine details of the computer used. The first issue can be circumvented in many ways, however the most widely used method is using the model of computation proposed by the british computer scientist Alan Turing in 1936 (see Turing's original paper in chapter 1 of [37]). This model, called the *Turing machine* model of computation, provides a framework in which every conceivable algorithm can be broken into specific components of this model and thereby its runtime can be specified as a multiple of a unit of time defined in this model. Historically, all the notions discussed in the first half of this chapter were first introduced by Richard Karp in [15, 16] with reference to the Turing model of computation, however we will instead introduce the concepts from a more algorithmic point of view.

In order to overlook hardware details, we choose to consider only the asymptotic scaling of the runtime (and correspondingly the space) with the size of the problem. The size of a problem is simply the number of bits used to represent the input of the problem, to the solver, and is usually represented as n. Hence the runtime can be asymptotically bounded above by $f(n)$, for some function of n. For retaining the information about asymptotic behaviour of a function alone, we will find the order-notation in computer science, defined below, particularly useful in this chapter and further on too. In what follows, we will discuss a notation to compare the asymptotic behaviour of two functions, $f(n)$ and $g(n)$. In the first case, let us suppose that, as n becomes sufficiently large, the growth of $f(n)$ is not faster than the growth of $g(n)$. Such a case is formally represented by the relation $f(n) = \mathcal{O}(g(n))$, or, equivalently, by $g(n) = \Omega(f(n))$. Lastly, if it turns out that both $f(n)$ and $g(n)$ have similar

growth rates, we denote this case by the relation $g(n) = \Theta(f(n))$. The following definition summarizes the order-notation which we will use.

Definition 2.0.1 *Order notations in computer science*
Let $\mathbb{R}_+ \in [0, \infty)$ be the set of non negative numbers, $n \in \mathbb{R}_+$ and $f, g : \mathbb{R}_+ \to \mathbb{R}_+$ be any two functions. We say that

1. *$f(n) = \mathcal{O}(g(n))$, when $\exists\, n_0 \in \mathbb{R}_+$ such that $\forall\, n > n_0$, $\exists\, c \in \mathbb{R}_+$, $f(n) \leq c\, g(n)$.*
2. *$f(n) = \Omega(g(n))$, if and only if $g(n) = \mathcal{O}(f(n))$, see [38] for this terminology.*
3. *$f(n) = \Theta(g(n))$, when $f(n) = \mathcal{O}(g(n))$ and $g(n) = \mathcal{O}(f(n))$, i.e, $\exists\, c_1, c_2, n_0 \in \mathbb{R}_+$ such that $c_1\, g(n) \leq f(n) \leq c_2\, g(n)$, $\forall\, n \geq n_0$.*

To understand the first two notations in the above definition, consider the problem of multiplying two matrices, each having n bits. The multiplication operation takes time that is at most n^3, whereas it can be improved to $n^{2.38}$ by some special techniques, cf. [39]. Nevertheless, we can say that the best known algorithm for multiplying two matrices will have a runtime that is $\mathcal{O}(n^3)$, implying that the scaling in runtime for large n will not be any more than n^3. It is also correct to say that the best known matrix multiplication algorithm takes time $\Omega(n)$ (or even $\Omega(n^2)$) because the best known technique, in fact, runs in a time that is proportional to $n^{2.38}$. To understand the last notation in (Def. 2.0.1), consider the problem of adding two sequences. The best known algorithm for adding two $n-$bit sequences, runs in $\Theta(n)$ time, implying that as n becomes *sufficiently* large, the scaling of the run time of this algorithm, is linear in n. Lastly, for those familiar with array sorting algorithms in computer science (see in [40]), a more subtle example can be provided. Consider the problem of sorting an array of size n using two algorithms, the Merge sort and the Heap sort. The former is a $\mathcal{O}(n \log_2 n)$ time algorithm whereas the latter is a $\Theta(n \log_2 n)$ algorithm.

In this chapter, we will introduce a classification of problems based on their asymptotic runtimes. The reader is a priori not required to have any background in computational complexity to read this chapter. All definitions and the central ideas behind the explanations to follow have been borrowed from standard texts in computational complexity theory such as [17, 18, 19, 41]. We have also provided examples of physical problems (in statistical mechanics of spin systems) to show the relevance of computation complexity in physics. For an interesting and verbose reading in computational complexity, see [42]. The layout of this chapter is as follows. In the first section (Sec. 2.1), we will only deal with the formal classification of decision problems into classes P, NP and NP-complete. In the subsequent

section (Sec. 2.3), we will turn to counting problems, formally introducing the classes #P and #P-complete and discuss some examples.

2.1 Decision problems

In this section, we formalize the notion behind *solving a problem*. Throughout this thesis we will present a problem in the format of an input and an output. See (Ex. 2.2.1, 2.2.2) for examples. A large number of problems in computer science can be formulated as a *decision problem*[1]. A *decision problem* is one for which the output can be one of two possible values, denoted by "0" and "1" or "yes" and "no". Consider the problem of adding two $n-$bit sequences. The problem can be presented as a series of n decision problems where each one involves determining the value of each successive bit of the sum. A similar strategy can be used to convert any problem to a series of *polynomially many* decision problems. Hence, instead of analyzing the general class of all possible problems, we will confine ourselves to decision problems alone. As remarked, we will not loose any generality.

2.1.1 Computational complexity of decision problems

For a decision problem whose input size is n, the runtime can be expressed as $\mathcal{O}(f(n))$ for some function $f(n)$. Considering one unit of time consumed in reading each bit of the input, we can claim that the smallest runtime of any problem that reads the entire input is n. Extending this idea, we classify all decision problems, which can be solved in time that scales *polynomially* with the size of the problem, i.e $f(n) = \mathcal{O}\left(n^k\right)$ for some $k \in \mathbb{N}$, to lie in the *complexity class P*. Problems in this class are said to be *efficiently solvable*.

Definition 2.1.1 *Complexity Class P*
A problem L with input of size n, is in P if there exists an algorithm which can produce the output of L, in time that is $\mathcal{O}\left(n^k\right)$ *for some constant k that is independent of n.*

Existence of a polynomial time algorithm for the problem is a sufficient condition for it to lie in P. Hence, proofs for containment of a problem in P is often formulated by explicitly providing an algorithm for the problem that runs in polynomial time. A famous example of a problem in P which has a great relevance in statistical physics is the problem of finding the ground state configuration of the simple two-dimensional Ising model on a planar lattice

[1] The reason behind such a formulation involves the details of the *Turing machine model*. See [19, 17] for details.

with ferromagnetic couplings, cf. [43]. Yet another example is that of computing the partition function (see Ex. 2.4.2) of the two dimensional Ising model on a planar lattice, cf. [43]. A proof for containment of the latter problem in P is an extension of the result of Lars Onsager (in [44]) – an analytic expression to compute the partition function, cf. [45]. Physicists often refer to problems in P that frequently occur in computations, as *exactly solvable problems*.

It is completely reasonable to ask why a *polynomial* scaling is a preferred criteria in the classification of problems. This is because it is the smallest order of scaling which is preserved under composition. Suppose we had classified all problem which have logarithmic time solutions, in a class like P, then a composition of two such problems will not be in this class (product of two functions, each having a logarithmic scaling, does not itself have a logarithmic scaling). This would not be a convenient classification for various reasons, one of them being that we cannot call a method to solve a problem within an algorithm for another problem of the same class. However, a polynomial multiplied to itself, a constant number of times, still yields a polynomial. An exponential scaling also follows this rule, but it will correspond to a bigger class than P.

Clearly there are decision problems for whom a polynomial time solution is not (yet) known, i.e, it is not immediately clear if they belong to P. Of these, there are problems for whom any claim for a solution can be *verified* in polynomial time. A claim for a solution is simply one of two values from the set $\{0, 1\}$. However, this claim must be supported by an *evidence*, *witness* or a *certificate*. When a decision problem is derived from a real world problem, this certificate is sometimes the output of the real problem itself, which is used by the solver to output the decision. For instance, see (Exs. 2.2.1 and 2.2.2). Such problems for which any claim for a solution can be supplemented with a *efficiently verifiable certificate*, is classified in the class NP.

Definition 2.1.2 *Complexity class NP (Non-Deterministic Polynomial time)*
A decision problem L is said to be in NP if and only if there exists a polynomial $p(t)$ and polynomial time algorithm for a problem V_L, described as follows.

1. *For every input to L corresponding to the output 1, denoted by x, there exists some sequence y of $p(|x|)$ bits such that the output of V_L on input (x, y), is 1.*
2. *For every input to L corresponding to the output 0, denoted by x, for every sequence y of $p(|x|)$ bits, the output of V_L on input (x, y), is 0.*

where $|x|$ denotes the number of bits in the sequence x, V_L is called the verifier for L and y is called the certificate to the input x.

For instance, consider the problem of factorizing an integer, denoted by M. An NP formulation of the problem will be – Given three positive integers M , M_1 and M_2, is there any factor of M that is greater than M_1 as well as less than M_2 ? Any affirmative answer to this decision problem must be supplemented with a certificate, that is the particular factor of M, which is in between M_1 and M_2. A polynomial time verifier must simply check whether the certificate is indeed a factor of M, less than M_1 and greater than M_2.

Historically, this is not the original definition of NP as in [15]. In its original form, NP was defined using the concept of *non-deterministic Turing machines*[2], see [17, 18, 19]. The above definition is sometimes referred to as a *verifier based definition of NP* and it is particularly useful in proving the membership of problems in NP as well as illustrating the relevance of computational complexity in real world problems. There is also a subtlety in (Def. 2.1.2). For inputs on which a NP problem outputs 0, a polynomial time verifiable certificate need not exist. Whether all NP problems have a polynomial time verifiable certificate for each of its inputs that result in the output "0", is an open problem, see [41, 17]. For an example of a NP problem, see (Ex. 2.2.2).

According to (Def. 2.1.2) of the class NP, it may seem that there is no restriction on the running time of an algorithm for any problem in NP. However, this is not true, there is always an algorithm for every NP problem that runs in time $2^{\mathcal{O}\left(n^k\right)}$, for some constant k, independent of n. An algorithm with this runtime is often called a *brute-force solution*, see [40]. However, the exponential runtime of the brute force technique strongly relies on the fact that a polynomial time verifier exists for a NP problem. In this technique, for each input to the NP problem, one can enumerate all possible sequences of a (given) polynomial length and assume each one to be a potential certificate for an affirmative answer to the problem and test the validity of the certificate using the polynomial time verifier. The runtime of algorithms that adopt such a brute force technique will scale as the number of different binary sequences of length of n, i.e, exponential in n. Therefore, problems in NP must be in the class of problems for which there exist an algorithm whose runtime scales exponentially in the size of the input, the latter class is called EXP, cf. [17].

Clearly, the class of problems in P also lie in NP. To show that a problem lies in NP, it suffices to show that there exists a certificate, corresponding to every affirmative output, which can be verified in polynomial time. It is worth remarking here that the question of whether NP is also contained in P, or equivalently, whether $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$, is one of the most

[2]In a nutshell, a Turing machine is a model of a computer that performs a sequence of certain elementary operations, given by a set of rules. In the case of a *deterministic* Turing machine, at each time step, there is only one rule that decides the next elementary operation. Whereas, in the case of a *non-deterministic* Turing machine, at each time step, *any one out of a set of possible rules can be chosen*, to decide the next elementary operation. NP contains all problems that can be solved in polynomial time, using a non-deterministic Turing machine.

celebrated open problems in computational complexity [17, 18, 42].

It is often difficult to show the *non-existence* of a polynomial time algorithm for a decision problem. Clearly, a reason for the non-existence cannot be that no such polynomial time algorithm is *yet* known. Often in computer science, for such problems that do not have readily known algorithms, a theoretical concept of an *oracle* is used. An *oracle* for a complexity class $\mathcal{C}$ is an algorithm that can solve any problem in the complexity class $\mathcal{C}$. However, this notion strongly relies on the fact that it is possible to express instances of a particular problem of interest as instances of another problem (about which much is already known). A problem L can be termed "at least as hard as" another problem L', if the solution for L can be used with polynomial preprocessing, to obtain a solution for L'. In formal terms, we say that L' can be *reduced* to solve L, where a reduction is precisely defined below.

Definition 2.1.3 *Polynomial time reduction or Karp reduction [19]*
A problem L is said to be polynomial time reducible, or Karp reducible, to another problem L', denoted by $L \prec_P L'$, if there is a polynomial time computable function f, such that for all inputs to L, denoted by x, the corresponding output of L is the same as the output of L' with input $f(x)$.

The above reduction is named after the famous computer scientist Richard Karp, in [15]. Here, the reduction function can be viewed as a preprocessor on the input. If $L \prec_P L'$, then every input to L, denoted by x, can be preprocessed in polynomial time to $f(x)$ and then fed as an input into the algorithm for solving L'. The corresponding decision output will coincide with any algorithm which can directly solve L. Informally[3], we can say that the task of designing an algorithm for L can be *reduced* to the task of designing an algorithm for L'. For concrete examples of reductions, look into [17, 18, 19]. On a technical note, when L is Karp reducible to L', the oracle for L' is used only once, to decide the problem L. However, one could instead think of an algorithm for deciding L that can access the oracle for L' multiple (polynomial many) times. This is called a *Turing reduction*. If L is *Turing reducible* to L', this means that an oracle that can solve L' can be used as a subroutine inside a polynomial time algorithm to provide a solution for L. Unless stated otherwise, when talking about reduction between decision problems, we will implicitly refer to Karp reductions. Consider the following useful remark.

Lemma 2.1.1 *Reduction is transitive. [19]*
Let L, L', L'' be any three decision problems. If $L \prec_P L'$ and $L' \prec_P L''$, then $L \prec_P L''$.

[3]The terminology can be misleading if one thinks of "reduction" as an artificial simplification of the problem itself. When we say "L is reduced to L'", it does not mean that problem L' is an easier version of the problem L.

We will now consider a particular class of problems in NP such that, if a polynomial time algorithm can be invented for any one problem in this class, then every problem in NP can also be provided with a polynomial time algorithm. Such a class is said to be *complete* for NP and hence termed as NP-complete. Below, we have a formal definition of NP-complete, using the idea of Karp reduction in (Def. 2.1.3), cf. [17].

Definition 2.1.4 *Complexity class NP-complete*
A decision problem L is said to belong to the class NP-complete if and only if $L \in$ NP and every problem in NP is Karp reducible to L according to (Def. 2.1.3), i.e, $L' \prec_P L$, $\forall\, L' \in$ NP.

This class of problems was first introduced by Richard Karp in 1921, see [15]. He termed the decision problems in the class NP-complete as *intractable* problems and provided a list of 21 such problems in his article [15]. The formulation of a complete class of problems for NP is very useful in determining which of the problems in NP must be of interest to both theoretical as well as algorithmic aspects of study. There is a general agreement that if there is no known sub exponential time algorithm for a NP problem, then the problem is very likely to be NP-complete. There are a lot of examples for NP-complete problems, few of which are mentioned in (Sec. 2.2). Intuitively, if the solution to a problem can be used to solve an NP-complete problem, then we expect it to be at least as *difficult* as any NP problem, and consequently be in NP-complete itself. Below is a theorem to formalize this idea.

Lemma 2.1.2 *Proof technique for NP-complete−ness results*
Let L , L' be two problems, $L \in$ NP-complete and $L' \in$ NP. If $L \prec_P L'$, then $L' \in$ NP-complete.

Proof : By (Def. 2.1.4), every $A \prec_P L, \forall\, A \in$ NP. Using (Lemma. 2.1.1), $A \prec_P L$ and $L \prec_P L'$ imply $A \prec L', \forall\, A \in$ NP. By (Def. 2.1.4), $L' \in$ NP-complete . □

The above property highlights an important technique used to prove that a given problem is in NP-complete. Notice that it is a far easier criterion that the definition in (Def. 2.1.4) for a problem to be in NP-complete because the latter involves testing the polynomial time reducibility of every problem in NP to a particular problem of interest. Whereas following the above technique, we can pick an NP-complete problem which seems close in context to a particular problem of interest and demonstrate a polynomial time reduction (Def. 2.1.3) from the chosen NP-complete problem.

The class NP-complete can also be interpreted as the set of hardest problems in the class NP. However, there are also decision problems *outside* (it is not known if they are contained

in) NP, yet with the property that an oracle which can solve such a problem can be used to solve any problem in the class NP. Such problems are classified in the class NP-Hard.

Definition 2.1.5 *Complexity class NP-Hard*
A problem L is said to belong to the class NP-Hard if and only if every problem in NP is Karp reducible to L, i.e, $L' \prec_P L\,,\ \forall\, L' \in NP.$

Note that problems in NP-Hard need not even be decision problems. Often NP-complete decision problems can have NP-Hard counterparts, for instance, the ground state computation problem for the Ising anti-ferromagnet, in (Ex. 2.2.2). Sometimes, a decision problem itself can be in NP-Hard if every affirmative output does not have an efficiently verifiable certificate. Clearly, NP-complete $\subseteq$ NP-Hard. Typically, to show that a particular problem is NP-complete, we first show that the problem is in NP and then, show that the problem is in NP-Hard as well. For concrete examples, refer to [17, 19, 1, 43, 2].

2.2 Examples of hard decision problems

In the previous section, we have introduced various terminologies in classifying decision problems into complexity classes P , NP and NP-complete. Here we will see a few important examples of NP-complete problems and also illustrate some of the techniques introduced above, with the specific example.

Example 2.2.1 *3 Satisfiability problem, cf. [17, 19, 41].*
3-SAT, or the 3-Satisfyability problem, is the archetype example of a NP-complete problem. Briefly, 3-SAT is the problem of assigning binary values to boolean variables $x_1, x_2, \ldots, x_n$ *that satisfies a given* boolean formula ϕ. *The* boolean formula *in reference is an expression of n variables denoted by* $x_1, x_2, \ldots, x_n$ *consisting of the logical "AND" and logical "OR" operations between the variables, denoted by* $\wedge$ *and* $\vee$ *respectively. The "value" of the expression is simply the evaluated result of the boolean expression for given values of the boolean variables, which is again one of* $\{0, 1\}$.

The definition of the problem is standardized by imposing the boolean expression to be in the canonical form, *cf. [17], i.e,*

$$\phi(x_1, \ldots, x_n) = \phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_k\,, \quad \textit{where} \quad \phi_i = x_i \vee x_j \vee x_k\,. \tag{2.1}$$

The formal definition of the 3-SAT problem is as follows.

Definition 2.2.1 *3-SAT*

Input: *A boolean formula ϕ of n variables $x_1, \ldots, x_n$ in the canonical form as in (Eq. 2.1).*

Output: *Does there exists an assignment of binary values for the variables $\{x_i\}_{i=1}^n$ such that $\phi(x_1, x_2, \ldots, x_n) = 1$?*

The 3-SAT problem is one of the very few cases of a NP-complete problem where it can be explicitly demonstrated that a polynomial solution for this problem implies a polynomial time solution for any NP problem in the Turing machine model. This proof was given independently by two computer scientists, Stephen Arthur Cook and Leonid Levin, in 1933, in [46, 47]. For details, see [17].

Example 2.2.2 *Ground state of an Ising Anti-Ferromagnet, see [43, 48, 45] .*

This is a problem of determining the ground state of a spin glass model *on a* $2D$ *plannar lattice, in the presence of an external magnetic field (see P5 in [48]).*

In 1912 the american mathematician Ernst Ising devised a simple spin model to study the theory of phase transitions in statistical physics. His model explained how magnetic properties of a material change with temperature. This spin model is named after its inventor, as the Ising model. *There are several ways in which one can generalize the model to describe the behaviour of many more physical systems using this model. Determining the ground state of an Ising model is one of the most fundamental problems in the study of condensed matter systems, see [43] for a quick review. A generalization of the Ising model presented here is called the* 2D Ising Anti-Ferromagnet *(IAF). Physicists have found it extremely strenuous to compute the ground state configuration of this type of spin model. Not surprisingly, it turns out that determining the ground state is an NP-Hard problem.*

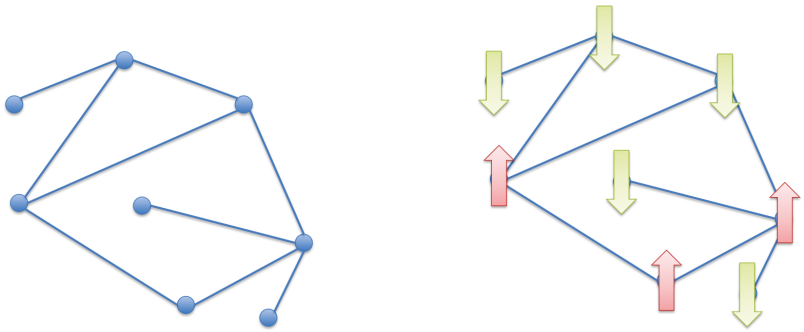
The IAF is specified using a planar graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as follows. Each vertex of $\mathcal{G}$ is identified with a spin$-1/2$ particle and every edge of $\mathcal{G}$ indicates an anti-ferromagnetic type interaction between the particles that correspond to the vertices incident on that edge. We will assume that the graph has n vertices, labelled $1, \ldots, n$. The spin at each vertex is specified by a number that is ± 1.

Any state of the IAF is specified by a configuration of the spin at each vertex of $\mathcal{G}$. We will use the variable S_i to denote the configuration of spin at vertex i, so $S_i \in \{+1, -1\}$. An assignment of ± 1 values to every spin variable in $\{S_i\}_{i=1}^n$ indicates a particular spin configuration of the IAF. In order for the Ising model to describe magnetic interactions, for a simplistic analysis, we will assume the external magnetic field to be perpendicular to the plane of the lattice, whose strength is given by B. The Hamiltonian describing the IAF model is specified by

$$H(\mathcal{G}) = \sum_{(i,j) \in \mathcal{E}} S_i S_j - B \sum_{i \in \mathcal{V}} S_i \, . \tag{2.2}$$

The energy of a particular configuration of the model is computed by simply substituting the values of the spin at each site, into the above Hamiltonian. See (Fig. 2.1b) for an example. The ground state of the model is nothing but the configuration with the least possible energy. Note that the Hamiltonian in (Eq. 2.2) provides all the information that is needed to compute its ground state configuration.

The following figure describes an IAF, defined using a planar graph.



(a) **(b)**

Figure 2.1 (Fig. 2.1a) shows a IAF model on a graph. In (Fig. 2.1b) we have a particular configuration of spins, specified by assigning ± 1 values to each vertex of the above square lattice. The energy of the spin configuration is (Eq. 2.2) is $1 + 2B$.

In the problem of determining the ground state of the IAF model, the planar graph defining the IAF model is provided along with the magnetic field. The task is to determine the ground state configuration (an assignment of ± 1 values at each vertex) of the IAF using the Hamiltonian in (Eq. 2.2). An equivalent decision version can be formulated as follows, see [48] for a similar formulation.

Definition 2.2.2 *2D Ground state of an Ising Anti-Ferromagnet (GIAF)*

Input: *A planar graph $\mathcal{G}$ with vertices labelled as $\{1, \ldots, n\}$, magnetic field strength B and a constant Δ.*

Output: *Is there an assignment to each variable in $\{S_i\}_{i=0}^{n}$ such that the Hamiltonian in (Eq. 2.2) becomes $H(\mathcal{G}) \leq \Delta$?*

GIAF as defined above was shown to be NP-complete by Barahona in [48] by showing a polynomial time Karp reduction from the NP-complete problem of finding the existence of an independent set in a graph*, see also [15] for a description of the latter problem. The containment of GIAF in NP can be seen easily. The certificate to the GIAF problem is just the explicit configuration of all*

the spins at every vertex of the underlying planar graph, see for instance in (Fig. 2.1b). A verifier must simply substitute each assignment of the spin variable (given by the configuration) into the Hamiltonian of the model in (Eq. 2.2) and verify whether indeed the energy of the model is less than Δ *(given as input to GIAF). The verifier runs in polynomial time. GIAF is also considered as the canonical NP-complete problem for physicists, like 3-SAT for computer scientists. Several problems in condensed matter physics are shown to be NP-complete by adopting a reduction from GIAF. An equivalent problem, of determining the ground state configuration of a* 2*D Ising spin glass on a non-planar graph fields, is also known to be NP-complete, cf. [45]. There are also different variants of GIAF which are known to belong to other classes of decision problems, which we shall not consider here*[4]*. For a detailed study of spin models and their relevance in studying error correction and known results about hardness of problems involving these models, refer to [50, 43].*

2.3 Complexity of counting

In the previous sections, we have studied the classification of decision problems into various complexity classes. In the following section, we will consider some problems which are different from decision problems. There is a class of problems which involve counting the number of solutions to some decision problem. Therefore, problems of this type output a positive integer instead of just 0 or 1, unlike decision problems. Such problems are called *counting problems*, cf. [17, 51, 52]. They were first introduced by Lesslie Valiant in 1979, cf. [53]. In this section, we will study counting problems, whose corresponding decision problem is in NP. Clearly, the number of certificates corresponding to affirmative answers to an NP problem, can be exponentially large. For instance, consider the problem of computing the number of configurations of the Ising Antiferromagnetic model (in Ex. 2.2.2), which have a total energy that is less than a given number Δ. A polynomial time algorithm to verify the exact number of such configurations, cannot simply exist, because there can be up to 2^n such configurations (where n is the number of spins in the model).

Therefore, a polynomial time certificate to verify the exact number of configurations of energy at most Δ for any given Δ cannot simply exist. In this sense a problem of counting the number of solutions to an NP problem need not necessarily be in NP itself.

The counting version of a NP problem L (with a verifier V_L) takes the same input as the L, whereas the output of the counting problem is the number of different certificates (see Def. 2.1.2) for which the output of the verifier is "1". The class of such counting problems,

[4] A variant of GIAF which allows for 3−body interactions is known to be complete for the quantum complexity class QMA, the quantum analogue of the complexity class NP, cf. [49].

where each problem involves counting the number of solution to some NP problem, is called #P (to be read as "sharp-P"). If the output of a #P problem is anything greater than one, it immediately indicates that the corresponding NP problem must output 1. Hence the counting version of an NP-complete problem is clearly NP-Hard, according to our discussion in (Sec. 2.1.1). The interesting question is whether the counting version of a NP problem is distinctively harder than the NP problem itself. One way to motivate the hardness of a counting problem is to show that there exists a hierarchy of infinitely many classes of problems, between NP and the class #P. This is one of the celebrated results in computational complexity, shown by Senusoke Toda in 1991, cf. [54].

Consider a Turing reduction from a problem P_1 to another problem P_2, which we mentioned in (Sec. 2.1, see the discussion following Def. 2.1.3). The reduction implies that a polynomial time algorithm for P_1 can have the algorithm for P_2 as a subroutine. The class of all problems which can be decided by a polynomial time algorithm, with access to an oracle for P_2 is denoted by P^{P_2} and it follows that $P_1 \in \mathsf{P}^{P_2}$. Combining this idea with the definition of NP-complete problems in (Def. 2.1.4) we find that $\mathsf{P}^{\mathsf{NP}} = \mathsf{P}^{\text{3-SAT}}$. That is, given access to a subroutine for solving 3-SAT, we can design a polynomial time algorithm for any problem in NP. Similarly, the class of problems in $\mathsf{NP}^{\mathcal{C}}$, for any complexity class $\mathcal{C}$, consists of all problems such that any certificate to a problem can be verified in polynomial time, by an algorithm that has access to an oracle for a complete problem in $\mathcal{C}$ (like 3-SAT is a complete problem for NP). Notice that $\mathsf{NP}^{\mathcal{C}}$, for any complexity class $\mathcal{C}$, is still a class of decision problems and moreover it contains NP itself.

Using this idea, one can provide a straightforward extension of the class NP, denoted by Σ_2^{P} in the literature (see [17, 18]) which is defined as $\Sigma_2^{\mathsf{P}} \stackrel{\text{def}}{=} \mathsf{NP}^{\mathsf{NP}}$. Similarly, we can define a complexity class $\Sigma_i^{\mathsf{P}} = \mathsf{NP}^{\Sigma_{i-1}^{\mathsf{P}}}$, for each successive value of i, starting from 2 onwards. Clearly, this will form an infinite hierarchy of decision problems where $\Sigma_i^{\mathsf{P}} \in \Sigma_{i+1}^{\mathsf{P}}$ for all $i \geq 2$. Now, it may seem that any decision problem can be placed in one of the infinite hierarchy of classes. Following the discovery of the class #P by Lesslie Valiant in 1978, Senusuike Toda in 1991 showed a result to the contrary in [54] by demonstrating that

$$\mathsf{P}^{\#\mathsf{P}} = \mathsf{NP} \cup \mathsf{NP}^{\mathsf{NP}} \cup \mathsf{NP}^{\mathsf{NP}^{\mathsf{NP}}} \ldots . \tag{2.3}$$

Informally, an algorithm with access to a NP oracle will have to call itself recursively, infinitely many times, in order to solve some problems in #P. This greatly highlighted the power of a #P oracle, distinguishing it from an NP oracle. We will now study the complexity class #P and problems that are complete for this class.

2.3.1 Classifying counting problems

In this section, we will describe formal notions behind the classification of counting problems. Firstly, a counting problem is defined as follows, cf. [17, 51].

Definition 2.3.1 *Counting Problem corresponding to a decision problem*
Let L be some decision problem and V_L be its verifier, according to the definition in (Def. 2.1.2). A counting problem P corresponding to the decision problem L is defined as follows.

Input: *Binary sequence x.*

Output: *Number of certificates for x that correspond to the output of V_L being 1. That is, number of elements in the set $\{y : V_L(x, y) = 1\}$.*

The problem of computing the number of configurations of a spin model in (Ex. 2.2.2) having a given total energy, is an example of a counting problem. In that case, x in the above definition will denote the input to GIAF – the graph of the model as well as the target energy, denoted by Δ. Then, y in the above definition will denote any spin configuration which serves as a certificate, i.e, whose energy is less than Δ. The verifier is simply an algorithm that computes the energy of the spin configuration, and checks that it is less that Δ.

The solution space of a counting problem is therefore the set of all non negative integers. For a decision problem in NP, the corresponding counting problem is classified in the class #P which is formally defined as below. See also [17].

Definition 2.3.2 *Complexity class #P (to be read as "sharp-p")*
A counting problem P is said to be in the class #P if there exists a decision problem L in NP with a polynomial time verifier V_L and a polynomial $p(t)$ such that for every input to the problem P, denoted by x, the output of P equal to the number of elements in the set $\{y \in \mathbb{Z}_2^{p(|x|)} : V_L(x, y) = 1\}$.

Typically the counting problem corresponding to a decision problem L is labelled as #L. But we will avoid this labelling as it may introduce some confusion with the notation #P, where the latter denotes a complexity class.

Note that the solution to any problem in #P cannot always be verified in polynomial time. This is because the number of solutions is not always bounded by a polynomial in the size of the input. The canonical example of a #P problem is that of counting all solutions to 3-SAT, defined in (Ex. 2.2.1). There can however be counting problems in #P which have a polynomial time algorithm. These problems are classified along with all problems (those

which are not necessarily of counting type) whose output can be computed in polynomial time. Such problems for which a polynomial time algorithm is known to exist, are classified into the class FP, defined as follows.

Definition 2.3.3 *Complexity class of efficiently computable problems, FP.*
A problem P is said to be in FP if for every input to the problem, denoted by x, there exists a polynomial time algorithm to compute the output of P on input x.

The complexity class FP can be seen as a straightforward generalization of the complexity class P. Clearly problems in P are also in FP. The output of problems in FP is not restricted to $\{0,1\}$, unlike for problems in P. But nevertheless a problem in FP can be solved by polynomially many queries to a problem in P. For details, see [17].

The problem of computing the energy of the IAF model (of Ex. 2.2.2), in the absence of the magnetic field, will be a candidate problem in FP, cf. [43]. It may not be surprising that problems in P have counting versions which are in FP, in fact, the counting version of the IAF problem (of Ex. 2.2.2), which is the problem of computing the partition function of this IAF model (similar to Ex. 2.4.2), is in FP too, cf. [43]. What is surprising is that there are problems in P whose counting versions are strongly believed not to be in FP. A famous example is one of counting the number of *simple cycles in a graph*. The problem of determining the existence of a simple cycle, is in P – it can be decided in linear (in the number of edges plus vertices in the graph) time using a *breadth-first search* on a graph, see [17, 41] for details. However no polynomial time algorithm to count the number of simple cycles in a graph is yet known.

Like in the case of decision problems (in Def. 2.1.3), we have a notion of reduction to compare the relative hardness of two problems in #P. The reduction that is used between two counting problems is called a *Turing reduction*. In this type of reduction, the number of solutions to a NP problem is counted using the number of solutions to polynomially many instances of another NP problem. Formally, a counting reduction is defined using the function class FP, as below. See [17] for a similar definition.

Definition 2.3.4 *Counting reduction*
Let P_1, P_2 be two counting problems. The function P_1 is Turing reducible to P_2, if for every input to P_1, denoted by x, we have $P_1(x) \in FP^{P_2}$.

That is, $f(x)$ can be computed in polynomial time with at most polynomially many queries to an oracle that can compute $g(x)$, for any x. Furthermore, if $f(x) = g(x)$ for all x, the reduction is called a *parsimonious reduction* [52, 51, 17]. Subsequently, analogous to NP-complete, the

above reduction characterizes a set of hardest counting problems in #P, called #P-complete. This class is formally defined as follows.

Definition 2.3.5 *Complexity class #P-complete*
A counting function f is in #P-complete iff f is in #P and $g \in FP^f, \forall\, g \in \#P$.

A NP-complete problem (with respect to Karp reductions) will have a #P-complete analogue, whenever the Karp reduction map in (Def. 2.1.3), is a one-one function (parsimonious reduction) as well. The following remark summarizes this property.

Remark 2.3.1 *A decision problem R is in NP-complete under parsimonious reductions if and only if #R is in #P-complete under parsimonious reductions.*

See [55, 56] for a proof of the above remark.

2.4 Examples of hard counting problems

In this section, we will provide some examples of problems that are complete for #P, including a fundamental problem in statistical physics of spin systems. The canonical example for a #P-complete problem is # 3-SAT, defined in (Ex. 2.4.1). The most celebrated example is one proposed by Valiant in 1978 [53], which is of computing the permanent of a $n \times n$ matrix with binary entries, see (Ex. 2.4.2).

Example 2.4.1 *Counting solutions to the 3−satisfiability problem or # 3-SAT*

This is a straightforward counting version of the NP-complete problem 3-SAT, in (Ex. 2.2.1). Its formal definition goes as follows.

Definition 2.4.1 *Counting satisfiability problem or # 3-SAT*

Input: *A boolean formula ϕ of n variables $x_1, \ldots, x_n$ in the conjunctive normal form, with each claus having exactly three variables.*

Output: *Number of elements in the set $\{v \in \mathbb{Z}_2^n : \phi(v_1, \ldots, v_n) = 1\}$.*

Note that each element of the above set is a solution to a 3-SAT problem (with the same input) in (Ex. 2.2.1). Hence this problem is in #P. Furthermore, the (Karp-)reduction used to prove the NP-complete−ness of the 3-SAT problem is also parsimonious, implying that # 3-SAT ∈ #P-complete, see [17]. Like 3-SAT is a canonical complete problem for NP, the # 3-SAT is often considered as the canonical #P-complete problem.

Example 2.4.2 *The Zero-One Permanent problem or PERM-01*
This is the archetype example for a #P-complete problem, and moreover, the counting complexity class itself was introduced by Lesslie Valiant in 1979 in [53] with this problem. It is one of computing the permanent of a binary matrix. Roughly, the permanent of a matrix can be written in the Laplace expansion [57] just like a determinant, however, all signs in the expansion (irrespective of even and odd permutations) must be positive. In other words, for a $n \times n$ matrix A, its permanent denoted by $\mathsf{Perm}(A)$, is given by the formula (see [51, 52])

$$\mathsf{Perm}(A) = \sum_{\pi \in S_n} \prod_{i=1}^{n} A_{i,\pi_i} \ , \tag{2.4}$$

where S_n is the group of all permutations on n indices and each element π of S_n is a vector of n components, such that each of its components is an integer between 1 and n, i.e, $1 \leq \pi_i \leq n$, $\pi_i \neq \pi_j$, $\forall i \neq j \in [1, n]$. This problem was proved to be #P-complete by Valiant in 1978, in [53].

Example 2.4.3 *Partition function of a Spin glass*

This is the problem of computing the partition function of a spin model which is quite similar to the IAF model introduced in (Eq. 2.2.2). This model is again defined using a graph $\mathcal{G} \stackrel{def}{=} (\mathcal{V}, \mathcal{E})$ whose vertices are labelled $1, \ldots, n$. An Ising spin$-1/2$ particle is identified with every vertex of $\mathcal{G}$ and the particles which interact correspond to vertices that are connected by an edge in the graph. As before we will identify the spin variables $\{S_i\}_{i=1}^{n}$ each of whose value is either $+1$ or -1. However, the key difference is the following. The interaction between every pair of particles (vertices) can be one of the two types, anti-ferromagnetic or ferromagnetic. The type of interaction between two particles can be specified on the graph by identifying a number to each edge of the graph, that is either -1 or $+1$. This model is called an Ising spin glass (*ISG*) *and is described by the following Hamiltonian where B is the magnetic field strength, like in (Eq. 2.2).*

$$H(\mathcal{G}) = -\sum_{(i,j) \in \mathcal{E}} J_{ij} S_i S_j - B \sum_{i \in \mathcal{V}} S_i \tag{2.5}$$

The graph $\mathcal{G}$ that specifies the spin-glass can be described by a $n \times n$ symmetric matrix (which is quite like the adjacency matrix of $\mathcal{G}$), denoted by $\mathcal{J}$. The (i, j) entry of the matrix $\mathcal{J}$ is J_{ij} and it

specifies the type of interaction between particles i, j of the spin model.

In (Ex. 2.2.2) we assumed the spin model to be at zero temperature, implying that the most stable configuration of spins in the model is given by the ground state configuration of the system. However, at any finite temperature T, this need not be true. The most stable configure is the configuration with minimum free energy*, denoted by $F(T)$. The free energy of the ISG is defined as.*

$$F(T) \stackrel{def}{=} -\ln \mathcal{Z}(\beta) \ . \tag{2.6}$$

where $\mathcal{Z}$ is the partition function *of the model defined by*

$$\mathcal{Z}(\mathcal{J},\, B,\, \beta) \stackrel{def}{=} \sum_{\{S_i\}_{i=1}^n \in \{+1,-1\}^n} \exp(-\beta\, H(\{S_i\}_{i=1}^n,\, \mathcal{J},\, B)) \ , \tag{2.7}$$

where $\beta = 1/k_B T$ is a measure of inverse temperature, B is the magnetic field strength and H is the Hamiltonian of the model defined in (Eq. 2.2). The sum in the above expression is over all possible configurations of spins on the vertices of the graph $\mathcal{G}$. Each configuration is specified by a list of n numbers each of which are either $+1$ or -1.

Notice that the set of n^2 coupling constants in $\mathcal{J}$, the magnetic field strength B and the inverse temperature β completely specify the partition function of the spin glass model. Finally, the problem of computing the partition function of the ISG is formulated as below.

Definition 2.4.2 *Partition function of the Ising Spin glass (SGZ)*

Input: *A symmetric $n \times n$ binary matrix J and two positive real numbers B, β.*

Output: *The numeric value of $\mathcal{Z}(\beta)$ according to the definition in (Eq. 2.7).*

The partition function of the spin model not only gives its free energy but also gives the probability of the system to assume various configurations. It provides viable information required to compute various thermodynamic and statistical properties of the spin model, see [20] for details. The expression for $\mathcal{Z}(\beta)$ in (Eq. 2.4.2) can be expanded as the series

$$\mathcal{Z}(\beta) = \sum_{\epsilon=-2|\mathcal{E}|}^{2|\mathcal{E}|} A_\epsilon\, e^{-\beta\,\epsilon} \ , \tag{2.8}$$

where each coefficient A_ϵ is the number of configurations of the spin model, whose energy (computed using the Hamiltonian in Eq. 2.5) is ϵ. Therefore, the computation of A_ϵ for any ϵ is clearly a counting problem and moreover, it is the counting version of an NP-complete problem similar in nature to (Def. 2.2.2), cf. [48]. Since the model can have at most $2|\mathcal{E}|$ different energies, the series in (Eq. 2.8)

is finite with at most $2|\mathcal{E}|$ *non zero terms. Therefore computing the value of* $\mathcal{Z}$*, i.e,* SGZ *for a given temperature is as hard as computing each coefficient* A_ϵ *in (Eq. 2.8) for given values of* ϵ *and the task of computing* $\mathcal{Z}$ *according to (Eq. 2.7) is intrinsically a counting problem.*

It was first demonstrated by Jerrum and Sinclair in 1990 in [58] that SGZ *(according to Eq. 2.4.2) is in* #P-complete*, by demonstrating a reduction from the* 3-SAT *problem in (Def. 2.4.1), see also [17]. See [59, 60] for different proofs of* #P-complete *—ness of* SGZ *and* #P-complete *—ness of problems that are similar in nature to* SGZ*.*

2.5 Summary

In this chapter, we have presented terminologies that are used to claim that a certain problem is computationally hard. Problems were of two major types, the decision type and the counting type. In (Sec. 2.1) we classified the problems of decision type for whom a polynomial time algorithm is known, into the class P, defined in (Def. 2.1.1). This was part of a larger class of problems for whom a polynomial time verification is known to exist, yet the existence of an algorithm to solve the problem itself is unknown. This larger class of problem was called NP, defined in (Def. 2.1.2). Besides membership, we introduced a framework of comparing the hardness of two problems, using the notion of Karp-reductions, defined in (Def. 2.1.3). This led to the introduction of complexity classes NP-Hard and NP-complete, defined in (Def. 2.1.4), the later corresponding to hardest problems in NP. Importantly, we introduced a technique in (Lemma. 2.1.2) to show that a given problem is NP-complete.

We introduced a class of problems #P, each of which can be formulated as counting the number of solutions to an NP problem in (Def. 2.3.2). Similar to the decision case, we introduced a framework for comparing two counting problems through the notion of counting reductions in (Def. 2.3.4). This gave rise to the class #P-complete, of hardest counting problems in #P. Here too we provided a technique in (Def. 2.3.5) to show that a given problem is in #P-complete. We have also provided a few examples of problems in NP-complete and #P-complete classes. We will close with presenting the following diagram showing the various complexity classes discussed in this chapter and their containments[5].

[5]One cannot strictly compare #P with complexity classes of decision problems like P, NP, but the intention of the figure is to show that a #P oracle can solve any problem in NP. In the strict sense, one can replace #P by PP (see [54, 53, 51] for its definition) in the figure.

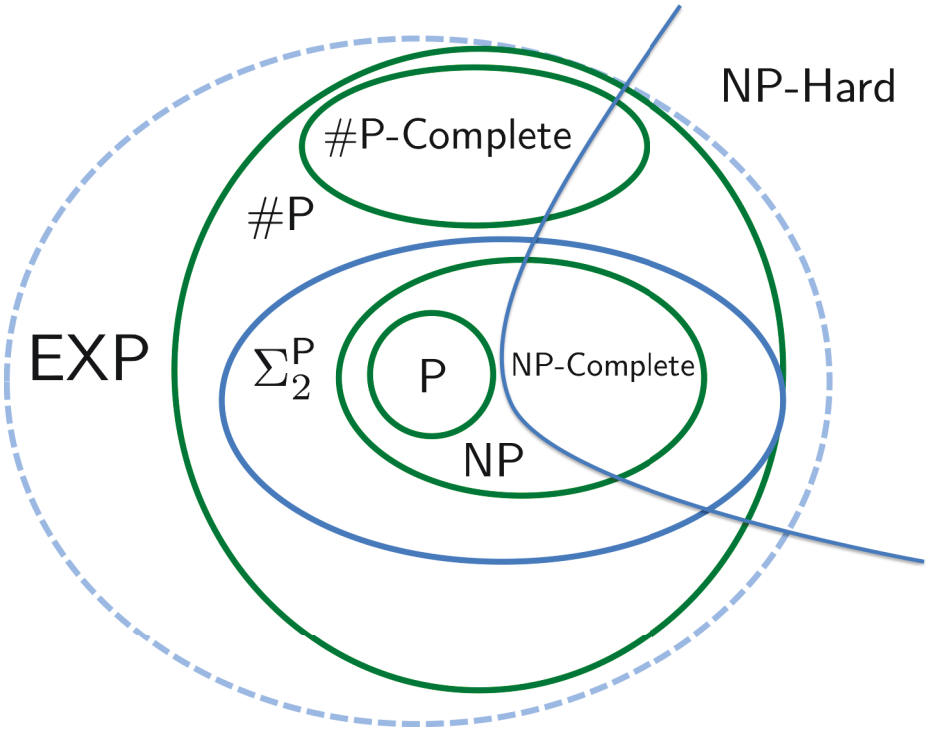


Figure 2.2 The figure shows various complexity classes and those which are contained in them. Note that the size of the shapes corresponding to classes does not signify any information about the actual sizes of the corresponding complexity classes, in terms of the number of problems known to be contained in them. Similar diagrams can be found in [17, 51, 19, 18]. Some of the classes labelled here, like EXP, have not been introduced in this chapter, however they are shown for the sake of completeness. For the interested reader, the only two complexity classes which are known to be clearly separated from one another are P and EXP, see [42] for a nice survey.

To appreciate this chapter, one can take home the following message. Easy problems in computer science are classified as P, hard problems as NP-complete and further incredibly hard problems as #P-complete.

Chapter 3

Classical Error correction

In this chapter, we will introduce all the necessary formalism in classical error correction that will be used throughout the rest of this thesis. This chapter is structured as follows. In (Sec. 3.1), we will introduce some basic terminologies in error correction. In (Sec. 3.2) we will specify a particular probabilistic model for errors on a communication channel. In (Sec. 3.3), we will specialize to *linear codes*, which are one of the most widely used error correcting codes. In (Sec. 3.4), we will formulate the error correction problem precisely. Finally, in (Sec. 3.5) we will discuss the computational intractability of the error correction problem as it was proposed in [1]. In (Sec. 3.6) we will introduce some more problems, which are of great importance to error correction and that are harder than the error correction problem itself.

3.1 An error correcting code

When a message is transmitted across a noisy channel, some bits in the message might be flipped. As a result of this, the error in the channel can cause a serious misunderstanding of the transmitted sequence, by the receiver. We will aim at avoiding such misunderstandings by carefully choosing what to transmit across the channel. For instance, if the sequences transferred between the sender and receiver are restricted to the following set,

$$\mathcal{C} \stackrel{\text{def}}{=} \{00000000, 01111000, 10110100, 11001100, 11010010, 10101010, 01100110, 00011110, \\ 11100001, 10011001, 01010101, 00101101, 00110011, 01001011, 10000111, 11111111\} \quad (3.1)$$

then it turns out that even if one of the bits may be flipped during transmission, one can devise an algorithm to retrieve the actual transmitted sequence. Therefore, we say that the message is now *protected* from any error that can cause one of the bits to flip.

The set of sequences which are transmitted across a channel is called an *error correcting code* and the sequences themselves are called *codewords*. The method of assigning a codeword to every message is known as an *encoding* map or an *encoder*. Of course, the codeword must be ultimately mapped back to a message. This is achieved by a method that resembles the reverse of encoding and we will call it an *unencoding* map or an *unencoder*. We will directly deal with errors that occur on the codewords, so, encoding and unencoding are not important for our analysis, we have mentioned these steps for completeness.

The *Hamming weight* of a sequence b, denoted by $\mathsf{wt}(b)$, is equal to the number of "1"'s in b. Two sequences b_1, b_2 match exactly if and only if their bitwise addition yields the *trivial sequence*, i.e, the sequence whose Hamming weight is 0. This concept is used to quantify the difference between two sequences, by the notion of their *Hamming distance*, denoted by $d(b_1, b_2)$, which is simply equal to the Hamming weight of their bitwise sum, i.e, $d(b_1, b_2) = \mathsf{wt}(b_1 + b_2)$. Informally, the Hamming distance between two sequences is the number of positions at which one sequence must be modified to match the other sequence.

An error correcting code is classified based on the number of messages it can protect, the number of bits in a codeword of the error correcting code and the smallest Hamming distance between any pair of codewords.

Definition 3.1.1 *Parameters of an Error Correcting Code, cf. [12, 61, 10].*
An error correcting code $\mathcal{C}$ with M codewords is characterized by the following parameters.

1. *Block length, denoted by n, is the number of bits in a codeword. (Same for all codewords.)*
2. *Code dimension, denoted by k, where $k \stackrel{def}{=} \lfloor \log_2 M \rfloor$.*
3. *Distance, denoted by d, is the minimum Hamming distance between two codewords of $\mathcal{C}$.*
4. *Rate of a code, denoted by R, where $R \stackrel{def}{=} k/n$, with n, k as defined above.*

In future, we will refer to $\mathcal{C}$ as a (n, k, d) code and in situations where d is unknown, we will simply refer to $\mathcal{C}$ as a (n, k) code.

For instance, the set of sequences in (Eq. 3.1) constitute a $(8, 4)$ error correcting code. Furthermore, after comparing the Hamming distance between every pair of codewords, we find that the distance of this code is 4. In other words, we have a $(8, 4, 4)$ error correcting code.

3.2 The error model

A bit flip error exchanges 0 and 1 in a codeword. This enables us to consider the error as a binary sequence itself, denoted by e, which simply adds to the transmitted codeword, so that, at positions where e has a "1", the corresponding codeword suffers from a bit flip error. For instance, the abstract operation of flipping the bits at positions $1, 3, 5, 7$, in a $10-$bit codeword, can be achieved by adding 1010101000 to the codeword. Hence if a sequence r is received, then $r = c + e$ where c is the transmitted codeword and e is an error. The number of bits flipped by the error is simply the Hamming weight of e.

When all errors are equally likely, there is no way of determining what codeword was actually sent. The first step in error correction is to specify an ordering on the errors, according to their *likelihood* of occurrence. This is formally done with the help of an *error model*, which simply assigns a probability to every possible error operation on the codewords.

One common assumption for bit flip errors is that every bit is flipped independently and the probability of a bit flip is identical for all bits. More precisely, the probability of a weight$-t$ error is simply the probability of t, independent and equally likely, weight-1 errors. For instance, the probability of the weight-2 error, $e = 00110$, is given by $(1-p) \times (1-p) \times p \times p \times (1-p) = p^2(1-p)^3$. These assumptions specify a error model called the *Binary Symmetric Channel (BSC)*, defined as follows, cf. [12, 10, 61, 62].

Definition 3.2.1 *Binary symmetric channel (BSC)*
On a binary symmetric channel, $\mathsf{Prob}(e) = p^{\mathsf{wt}(e)}(1-p)^{n-\mathsf{wt}(e)}$. *$p$ is the probability of a single bit flip error and it is often called the error rate per bit.*

The BSC is a widely studied error model, cf. [10, 61], for the different types of error models.

3.3 Linear codes

Suppose the sender prepares a codeword and transfers it across a channel. If the received sequence does not match any codeword, then the receiver can be certain that there was an error during transmission. A straightforward method to determine if a $n-$bit sequence is indeed a codeword of a (n, k) error correcting code, is to compare the sequence with each codeword. But this method will soon become infeasible if it the number of codewords scale exponentially with n, in other words, if $k = \Omega(n)$, which, in fact, is often the case, cf. [61, 10].

So, the encoding must be carefully designed so that the codewords can be characterized succinctly. A careful observation of the sequences in (Eq. 3.1) will suggest that when the following matrix is multiplied with each codeword, the result is always the trivial sequence.

$$\mathsf{H} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.2}$$

$$\mathsf{H} \cdot c = 0 \ , \ \forall\, c \in \mathcal{C}$$

where $\mathcal{C}$ is the code whose codewords are specified in (Eq. 3.1). Conversely, one can further verify that any binary sequence which multiplies with H to give the trivial sequence, is an element of $\mathcal{C}$. Every row of the above matrix can be interpreted as a set of *parity* constraints on the bits of the codeword. For instance, the first row implies that a sequence is in the code if and only if its bits at positions $2, 3, 4, 5$ have an even parity. Mathematically, we can say that the codewords must form the null space of the above matrix. Because the null space of a matrix forms a vector space, it follows that a code defined by such parity constraints is also a vector space. Such codes are called *Linear codes*, defined as follows, cf. [63, 13, 10].

Definition 3.3.1 *Linear Code*
A (n, k, d) linear error correcting code, denoted by $\mathcal{C}$, is a $k-$dimensional subspace of $\mathbb{Z}_2^n$, where the distance of $\mathcal{C}$, defined according to (Def. 3.1.1), is given by d.

Henceforth, we will focus our study on linear codes. Subsequently, for any linear code, we can associate a matrix which contains all the parity constraints that each codeword must satisfy. This matrix is called the *parity check matrix* and it is defined as follows, cf. [10, 63, 62].

Definition 3.3.2 *Parity check matrix*
For a (n, k) linear code $\mathcal{C}$, a parity check matrix, denoted by H, is a $(n-k) \times n$ binary matrix with the property $H \cdot c = 0\,,\ \forall\, c \in \mathcal{C}$. In other words, $\mathcal{C} = \{x \in \mathbb{Z}_2^n : H \cdot x = 0\}$.

So, if a received $n-$bit sequence does not satisfy the $(n-k)$ parity check constraints of the corresponding linear code, then an error has certainly occurred during transmission.

Since a linear code is also a vector space, it can be expressed as a span of a basis set. The basis set is represented in the form of a matrix, called the *generator matrix*, defined below.

Definition 3.3.3 *Generator matrix*
For a (n, k) linear code $\mathcal{C}$ whose basis vectors are $b_1, b_2, \ldots, b_k$, a generator matrix for $\mathcal{C}$, denoted by G, is a $k \times n$ matrix whose rows are $b_1, b_2, \ldots, b_k$.

For instance, the linear code in (Eq. 3.1) has the following generator matrix.

$$\mathsf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix} \tag{3.3}$$

Indeed, one can verify that every codeword of the example in (Eq. 3.1) can generated by the rows of the above matrix, for instance, the fifth codeword can be expressed as $11010010 = 10000111 + 01001011 + 00011110$ and the sixth codeword can be expressed as $10101010 = 10000111 + 00101101$. Formally, if γ is some $k-$bit sequence, then it corresponds to a unique codeword of a linear code, given by $\gamma \cdot \mathsf{G}$, where G is the generator matrix of the code. Since the rows of the generator matrix are codewords themselves,

$$\mathsf{G} \cdot \mathsf{H}^T = 0\,. \tag{3.4}$$

A particular code can have many generator matrices (as well as parity check matrices), however this freedom identifies any two codes as *equivalent* if the rows of their corresponding generator matrices (or parity check matrices) are related by a change of basis, cf. [10, 2].

Note that given a binary matrix, there is no intrinsic property of the matrix that deems it as a parity check matrix or a generator matrix. In fact, in (Def. 3.3.2), H can also be regarded as a generator matrix of a $(n, n-k)$ linear code which is denoted by $\mathcal{C}^{\perp}$. Furthermore, the

parity check matrix for $\mathcal{C}^{\perp}$ will be G as a consequence of the orthogonality property in (Eq. 3.4). It follows that every codeword of $\mathcal{C}$ is orthogonal to every codeword of $\mathcal{C}^{\perp}$. The code $\mathcal{C}^{\perp}$ is called the *dual* of $\mathcal{C}$, or simply, the *dual code*, and defined as follows, see also [10, 13, 63].

Definition 3.3.4 *Dual code*
Let $\mathcal{C}$ be a (n,k) linear code whose parity check matrix is H and generator matrix is G. A $(n, n-k)$ binary linear code is said the dual of $\mathcal{C}$, denoted by $\mathcal{C}^{\perp}$, if the parity check matrix of $\mathcal{C}^{\perp}$ is G and the generator matrix of $\mathcal{C}^{\perp}$ is H. In other words,

$$\mathcal{C}^{\perp} \stackrel{def}{=} \{b \in \mathbb{Z}_2^n \,:\, b \cdot c = 0\,,\ \forall\, c \in \mathcal{C}\}. \tag{3.5}$$

The rows of a parity check matrix H need not always be orthogonal to one another, however, when the rows are orthogonal, it means that the rows are themselves codewords of the linear code $\mathcal{C}$. In other words, the dual of the code $\mathcal{C}^{\perp}$ (spanned by the rows of H) will itself be in $\mathcal{C}$, i.e, $\mathcal{C}^{\perp} \subseteq \mathcal{C}$. A linear code that satisfies this property, is called a *weakly self dual code*.

Recall that to know the distance of a code (cf. Def. 3.1.1), we must know the Hamming distance between every pair of codewords, in other words, the Hamming weight of the sum of every pair of codewords. However, since a linear code posses all the properties of a vector space, the sum of its codewords, must also be a codeword. Then, it follows that the distance of the linear code is simply the minimum Hamming weight of a non-trivial codeword.

Let us summarize below all the formal notions we have learnt so far, about linear codes.

Lemma 3.3.1 *Properties of a linear code*
Let $\mathcal{C}$ be some (n,k,d) linear code. The following statements about $\mathcal{C}$ are true.

1. *$\mathcal{C}$ is a $k-$dimensional subspace of $\mathbb{Z}_2^n$, with a code distance d, cf. (Def. 3.1.1).*
2. *The smallest Hamming weight of a non-trivial codeword of $\mathcal{C}$, is d.*
3. *$\mathcal{C} = \{m \cdot G \,:\, m \in \mathbb{Z}_2^k\}$ where G is the $k \times n$ generator matrix for $\mathcal{C}$, cf. (Def. 3.3.3).*
4. *$\mathcal{C} = \{b \in \mathbb{Z}_2^n \,:\, H \cdot b = 0\}$ where H is the $(n-k) \times n$ parity check matrix for $\mathcal{C}$. Furthermore, $G \cdot H^T = 0$, cf. (Def. 3.3.2).*
5. *$\mathcal{C} = \{b \in \mathbb{Z}_2^n \,:\, b \cdot c = 0\,,\ \forall\, c \in \mathcal{C}^{\perp}\}$ where $\mathcal{C}^{\perp}$ is the dual code to $\mathcal{C}$. The parity check matrix and generator matrix for $\mathcal{C}^{\perp}$ are G and H, respectively, cf. (Def. 3.3.4).*

For a proof, refer to any of the standard texts on linear codes, such as [61, 10, 63, 1, 12, 63].

3.4 Decoding Linear codes

When a codeword undergoes an error while it is transmitted, the received sequence might not match any codeword. This is true for any error on the codewords in (Eq. 3.1), that affects three bits or less,. The process of inferring the transmitted codeword from the received sequence, is referred to as *decoding*. Therefore, a *decoder* must input the received sequence r and output a codeword c (desirably, the transmitted codeword). Because $r = c + e$, where e is the error that occurs during transmission, a decoder can be designed to just output e itself. Then, the transmitted codeword can easily be retrieved by adding e to the received sequence, i.e, $r + e = c + e + e = c$. Clearly, decoding is the most important step in correcting the error on the transmitted codeword. In this section, we will devise a strategy that can be used to perform decoding when the probability of errors is given by the binary symmetric channel.

The first step in error correction is to determine whether there is actually any error. This step is formally called an *error detection step*. The facility of describing a linear code by its parity check matrix greatly simplifies the error detection step. All we need to do it to test if the received sequence r obeys $\mathsf{H} \cdot r = 0$. For instance, take the example of the linear code in (Eq. 3.1) whose parity check matrix is given in (Eq. 3.2). Suppose the sender transmits some codeword across the BSC channel, resulting in the received sequence r, given by

$$r = 10001100\ . \tag{3.6}$$

The receiver immediately knows the presence of an error, observing that $\mathsf{H} \cdot r = 1011 \neq 0000$. In general, the received sequence r and the transmitted sequence c are related by $r = c + e$ and a nontrivial value of the product $\mathsf{H} \cdot r$, indicates the presence of an error. Since we know that c must satisfy $\mathsf{H} \cdot c = 0$, it must be that

$$\mathsf{H} \cdot e = 1011\ . \tag{3.7}$$

The sequence 1011 is called the *error syndrome*. In general, an *error syndrome* can be defined for any error and a linear code, as follows.

Definition 3.4.1 *Error Syndrome*
The error syndrome for a $n-$bit binary sequence e and a (n, k) binary linear code $\mathcal{C}$ with parity check matrix H is defined as the $(n-k)-$bit sequence s, where

$$s \stackrel{def}{=} H \cdot e\ . \tag{3.8}$$

In particular, the bits of the syndrome indicate the rows of the parity check matrix (correspondingly, the parity constraints) that are not satisfied by the error sequence.

Let us revisit (Eq. 3.8), which is also called the *syndrome equation*. Our aim is to determine the sequence e that satisfies the above relation. However, we cannot simply invert (Eq. 3.8) to find e as there will be no unique solution. Still, one need not doubt the significance of s in the correction strategy, it certainly gives some nontrivial information about the error. For the parity check matrix in (Eq. 3.2) and the received sequence 10001100, we observe that

$$s = 1011\ . \tag{3.9}$$

This narrows down the potential error to any binary sequence that violates all but the second parity constraint of H (in Eq. 3.2). But this condition alone cannot lead us to a unique error sequence – there are 16 sequences with this property, given as follows.

$$\begin{aligned} &01000000, 00111000, 11110100, 10001100, 10010010, 11101010, 00100110, 01011110, \\ &10100001, 11011001, 00010101, 01101101, 01110011, 00001011, 11000111, 10111111 \end{aligned} \tag{3.10}$$

In general, these constraints that evolve from observing the syndrome are obviously underdetermined. Indeed, observe that if e_0 is a particular solution to the syndrome equation, so is $e_0 + c$, for every codeword c. Here is where the error model provides a crucial information to choose the most likely error from the set of all solutions to the syndrome equation.

We are in a position to formulate the decoding problem. Suppose the probability of errors is given by the binary symmetric channel, i.e, $\mathsf{Prob}(e) = p^{\mathsf{wt}(e)}(1-p)^{n-\mathsf{wt}(e)}$. For $0 \leq p \leq 1/2$, $\mathsf{Prob}(e)$ is a strictly decreasing function of $\mathsf{wt}(e)$ and so, the most likely error must have the lowest Hamming weight. Therefore, for a syndrome s, of all the errors that satisfy the syndrome equation, a decoder must select the error which has the lowest Hamming weight, as the actual error in transmission. This strategy is called *minimum weight decoding* (MWD) and summarized as follows, cf. [63, 13, 1].

Definition 3.4.2 *Minimum weight syndrome decoding (MWD)*

Input: *A $(n-k) \times n$ parity check matrix H and a $(n-k)-$bit syndrome s.*

Output: *A $n-$bit sequence e such that $s = H \cdot e$ and $\mathsf{wt}(e) = \min\{\mathsf{wt}(e') : H \cdot e' = s\}$.*

For the code in (Eq. 3.1) with parity check matrix in (Eq. 3.2), we note that the possible errors with the syndrome 1011, have weights $\{1,3,5,3,3,5,3,5,5,3,5,7\}$ and it turns out that the

lowest weight error is

$$e = 01000000\ , \tag{3.11}$$

indicating that the most likely error that could have caused the syndrome 1011 is the flip of the second bit. Indeed, adding e to the received sequence gives the codeword 11001100. An obvious difficulty with the minimum weight decoding strategy is that the number of errors whose weights must be compared, to find the minimum, is *exponential in k* (which is effectively exponential in n too, when the linear code has a constant rate).

3.5 Complexity of the decoding problem

In the minimum weight decoding strategy, due to the exponential size of the solution set to the syndrome equation (cf. Eq. 3.8), there is no straightforward efficient technique to determine the minimum weight error, for a given syndrome and a parity check matrix. Now, compare the MWD problem to the NP-complete problems that we discussed in (Sec. 2.2).

In 1978, Robert McEliece, Elwin Berlekamp and Henk van Tilborg showed that indeed MWD is a NP-complete problem, cf. [1]. So, one is unlikely to find a polynomial time algorithm for MWD. From the discussion on computational complexity in (Sec. 2.1), we know that in order to show that MWD is intractable, it must first be formulated as a decision problem. One way to do this is the following. Instead of asking for the minimum weight solution to (Eq. 3.8), we can ask for the existence of a solution, whose weight is less than ω, where $\omega \geq 0$ is a fixed constant and an input to the decision problem, defined as below, cf. [1, 14, 64].

Definition 3.5.1 *ErrorWeights*

Input: *A* $(n-k) \times n$ *parity check matrix* H, $(n-k)-$*bit syndrome* s *and* $\omega \in [0, n]$.

Output: *Is there any* $n-$*bit sequence* e *such that* $H \cdot e = s$ *and* $wt(e) \leq \omega$ *?*

If an algorithm can decide ErrorWeights, we could simply run the algorithm for $\omega = 0, 1, 2, \ldots$ until the first affirmative answer is obtained, which will indicate the minimum weight of an error for the input syndrome. It only remains to actually find the bits in that particular error sequence. This can be done using n sequential queries to the ErrorWeights problem. Hence, in principle, ErrorWeights is as hard as MWD. Next, we must determine the complexity class of ErrorWeights. The following theorem now summarizes the result of McEliece et al. in [1].

Theorem 3.5.1 *Computational complexity of decoding a linear code, cf. [1].*
The computational complexity of the ErrorWeights problem in (Def. 3.5.1) can be summarized by

$$\textit{ErrorWeights} \in \textit{NP-complete} . \tag{3.12}$$

The original proof of the above theorem, in [1], involves a Karp reduction (cf. Def. 2.1.3) from the NP-complete problem called *Three dimensional matching* to ErrorWeights. A similar proof is also provided in [14, 64]. Following this, there have also been several other reduction proofs for (Thm. 3.5.1). The decoding problem ErrorWeights most naturally resembles a famous NP-complete problem called the *Binary Integer Programming* problem, cf. [65]. This suggests a straightforward Karp reduction from the *Binary Integer Programming* problem. Typically reduction proofs involve a lot of technical details which will divert the focus of this thesis. So, we have chosen to avoid the rederivation of the exact reduction proofs.

Let us consider a special case of the above ErrorWeights problem, when $s = 0$. This problem is of deciding the existence of a codeword in $\mathcal{C}$ of a given weight, formulated below.

Definition 3.5.2 *CodeWeights*

Input: *A* $(n-k) \times n$ *parity check matrix* H *and* $\omega \in [0, n]$.

Output: *Is there any* $n-$*bit sequence* c *such that* $H \cdot c = 0$ *and* $\mathsf{wt}(c) = \omega$ *?*

An intractability result for this problem too, was derived in [1], stated as follows.

Theorem 3.5.2 *Computational complexity of codeword membership problem, proposed in [1].*
The computational complexity of the CodeWeights problem in (Def. 3.5.2) can be summarized by

$$\textit{CodeWeights} \in \textit{NP-complete} . \tag{3.13}$$

In [1], a polynomial time Karp reduction from the NP-complete problem *Three dimensional matching* to CodeWeights is derived, to prove the above theorem. For a similar proof, see [14].

Remember that the distance of a linear code is simply the least weight of a codeword. One can query an oracle for CodeWeights with $\omega = 1, 2, \ldots$ until the first affirmative answer, which is indeed the distance of the input linear code. But, technically, a CodeWeights oracle reveals more information that what is really required to compute the distance. More precisely, the phrase "$\mathsf{wt}(c) = \omega$" in CodeWeights can be replaced by a weaker condition: "$\mathsf{wt}(c) \leq \omega$",

cf. [1, 64]. Computing the distance of a linear code was a celebrated open problem, until 1997, when it was finally shown to be NP-complete, by Alexander Vardy, cf. [64].

Codes, which do not have a parity check matrix are called *non-linear codes*. They are less favourable for error correction purposes and far less used than linear codes. A decoding problem like MWD can be analogously defined for non-linear codes too, which is called a *Minimum distance decoding* (MDD), cf. [12, 62, 13]. However, due to the absence of a parity check matrix, there is no clear error detection step for these codes, unlike for linear codes. Furthermore, the specification of a non-linear code requires the explicit listing of all its codewords. Therefore, the complexity of MDD is not immediately interesting to analyze.

3.6 Decoding performance and associated hard problems

Let us analyze the *performance* of the MWD strategy in (Def. 3.4.2). The decoder is completely deterministic, however the errors are not. In (Sec. 3.4), though the minimum weight error for the observed syndrome 1011 is given by 01000000, other errors could potentially have occurred, like the error 11110000 which has a probability $p^4(1-p)^4$. Given the syndrome 1011, regardless of which of the 16 errors (in Eq. 3.10) actually occurred, the MWD algorithm will always output 01000000, since it is the lowest weight sequence for the given syndrome.

If the actual error in transmission is indeed 11110000, then we have a scenario where the minimum weight decoder simply *fails*. Certainly, the probability of such a failure is exactly the probability that any error, other than the minimum weight error, occurs during transmission. For a given syndrome s, corresponding to the minimum weight error e_s, the probability of failure of MWD, denoted by P_s, is given by

$$P_s = \sum_{c \in \mathcal{C}} p^{\mathsf{wt}(c+e_s)}(1-p)^{n-\mathsf{wt}(c+e_s)} - p^{\mathsf{wt}(e_s)}(1-p)^{n-\mathsf{wt}(e_s)} \,. \tag{3.14}$$

A small value of P_s for a given syndrome indicates that the output of the decoder is very likely to be the actual error that occurred in transmission. But, if P_s is large for a given syndrome, then there is a high chance that the output of MWD is not the actual error sequence. On averaging P_s over all syndromes, we find

$$\bar{P} = \sum_{s \in \mathbb{Z}_2^{n-k}} P_s = \sum_{s \in \mathbb{Z}_2^{n-k}} \sum_{c \in \mathcal{C}} p^{\mathsf{wt}(c+e)}(1-p)^{n-\mathsf{wt}(c+e)} - \sum_{s \in \mathbb{Z}_2^{n-k}} p^{\mathsf{wt}(e_s)}(1-p)^{n-\mathsf{wt}(e_s)} \tag{3.15}$$

$$= \sum_{i=0}^{n} \binom{n}{i} p^i (1-p)^{n-i} - \sum_{s \in \mathbb{Z}_2^{n-k}} p^{\mathsf{wt}(e_s)}(1-p)^{n-\mathsf{wt}(e_s)} \tag{3.16}$$

$$= 1 - \sum_{i=0}^{n} w_i p^i (1-p)^{n-i} \tag{3.17}$$

where w_i indicates the number of syndromes for which the minimum weight of an error is i. For a given linear code $\mathcal{C}$ with a parity check matrix H, all errors with a particular syndrome s, can be represented by the set

$$\{e + c : c \in \mathcal{C}\} \tag{3.18}$$

where $\mathsf{H} \cdot e = s$. The above set is called a *coset* of a linear code, corresponding to the syndrome s. The minimum weight element of a coset is called a *coset leader*, and so, the polynomial in (Eq. 3.16) is referred in the literature [11, 63, 13, 66] by the name: *coset leader weight enumerator*. For a given linear code, if one were to compute all the coefficients in $\{w_i(s)\}_{i=0}^{n}$, then one can easily compute $\bar{P}$ in (Eq. 3.17). Furthermore, the problem of computing the coefficients $w_i(s)$ for any i, is precisely the counting version of the ErrorWeights (cf. Def. 3.5.1) problem. Therefore, it is legitimate to suppose that the problem of computing all the coefficients in $\{w(s)_i\}_{i=0}^{n}$ is certainly as hard as MWD, if not harder. Since the latter problem is NP-complete by (Thm. 3.5.1) and moreover, under parsimonious reductions (cf. [14, 67]), it turns out that the problem of computing the coefficients in $\{w(s)_i\}_{i=0}^{n}$ is complete for #P, cf. [52, 51].

Let us first discuss an upper bound on $\bar{P}$ and show that for codes with a large distance, $\bar{P}$ is sufficiently low. If t bits in a codeword are flipped, resulting in a received sequence which is closer (in Hamming distance) to a codeword that is different from the transmitted codeword, the MWD strategy can potentially fail. In order to avoid such scenarios, we must ensure that two sequences that result from t bit flips on a pair of codewords, never match each other. So, the Hamming distance between any two codewords must be larger than $2t$. Therefore, on a code with Hamming distance d, the MWD strategy will never fail whenever the weight of the actual error that occurs, is at most $(d-1)/2$. In other words, the number of errors of weight i, that lead to successful decoding by MWD, is simply the number of all possible sequences of weight i, for each $0 \leq i \leq (d-1)/2$, cf. [12, 10, 62, 61]. Hence,

$$\bar{P} \leq 1 - \sum_{i=0}^{(d-1)/2} \binom{n}{i} p^i (1-p)^{n-i} \,. \tag{3.19}$$

Using the inequalities on binomial coefficients presented in [68], we find that the expression on the right side of the above inequality is a monotonically decreasing function of d, for $0 \leq p \leq d/n$. Therefore a minimum weight decoder works well for codes with large distance, equivalently, if a code has large distance, then MWD will have a small failure probability. However, the converse is not true. There exist codes with a small distance, for which MWD

nonetheless has a small failure probability, cf. (Chapter 13 of [69]).

An *undetectable error* is an error which results in the zero syndrome because in such cases, we cannot be sure that an error has occurred. In what follows, we will mention the computational complexity of the exact computation of $\bar{P}_s$ in (Eq. 3.14), for the special case $s = 0$. This problem is precisely the counting version of the decision problem CodeWeights (that is also NP-complete, according to Thm. 3.5.2). All errors that result in $s = 0$ are exactly those sequences that are codewords. The probability of all the codewords is precisely the probability of observing an undetectable error, denoted by P_{ue}, i.e,

$$P_{ue} = \sum_{c \in \mathcal{C}} \mathsf{Prob}(c) = \sum_{c \in \mathcal{C}} p^{\mathsf{wt}(c)}(1-p)^{n-\mathsf{wt}(c)} \Rightarrow \sum_{i=0}^{n} A_i p^i (1-p)^{n-i} \,, \tag{3.20}$$

where A_i is the number of codewords in $\mathcal{C}$ whose Hamming weight is i. If we know the coefficients in $\{A_i\}_{i=0}^{n}$, it will be simple to compute the probability of an undetected error for a code. These coefficients can be collectively described by identifying them with a polynomial of degree n, defined as follows, cf. [13].

Definition 3.6.1 *Weight enumerator polynomial*
Let S be a set of $n-$bit sequences. The weight enumerator polynomial of S, denoted by $WE_S(x)$ is a degree n polynomial defined as follows.

$$WE_S(x) = \sum_{s \in S} x^{wt(s)} = \sum_{i=0}^{n} we_i(S) x^i \tag{3.21}$$

where each coefficient $we_i(S)$ denotes the number of sequences in S of Hamming weight i.

Consequently, the probability of an undetected error in (Eq. 3.20) can be expressed as

$$P_{ue} = (1-p)^n \left[\mathsf{WE}_{\mathcal{C}} \left(\frac{p}{1-p} \right) - 1 \right] \tag{3.22}$$

The problem of computing the weight enumerator coefficients of a linear code can be formulated as follows, cf. [52, 51].

Definition 3.6.2 *Weight enumerator problem: $we_j(\mathcal{C})$*

Input: *A $(n-k) \times k$ parity check matrix H, an integer j between 0 and n.*

Output: *Number of elements in the set $\{c \in \mathbb{Z}_n^2 : H \cdot c = 0\,,\ wt(c) = j\}$.*

Note that for any given i, any element from the above set is a valid solution to CodeWeights. Hence the above problem is clearly in #P according to (Def. 2.3.2). It turns out that this problem is also #P-complete. In fact, an analogous problem defined for any coset of a code, which would be applicable in (Eq. 3.14), is #P-complete too, cf. [52, 51, 67]. The below theorem summarizes the hardness of computing the weight enumerator of a linear code.

Lemma 3.6.1 *Computational complexity of computing the weight enumerator of a linear code*
The computational complexity of determining the number of codewords of Hamming weight w, in a linear code described by a parity check matrix H, according to (Def. 3.6.2), can be summarized by

$$we_w(\mathcal{C}) \in \#P\text{-complete} \ . \tag{3.23}$$

There are two ways to prove this theorem. The first method, which is used in [14], results from the observation that the reduction employed in the intractability result for CodeWeights in (Thm. 3.5.2) is indeed parsimonious. Hence as a consequence of (Lemma. 2.3.1), the weight enumerator problem is #P-complete. The second method, shown in [52], involves a polynomial time Turing reduction (see Def. 2.3.4) from the classic #P-complete problem PERM-01 (discussed in Ex. 2.4.2), to the weight enumerator problem.

3.7 Summary and outlook

Let us summarize the complexity results that we have discussed so far, as follows.

Lemma 3.7.1 *Hardness issues in classical error correction*
For a (n, k) binary linear code $\mathcal{C}$ specified by the parity check matrix H, a $(n-k)-$bit syndrome s and a number i between 0 and n, the following statements are true.

1. *The problem of determining the lowest weight $n-$bit sequence e satisfying the syndrome equation (in Eq. 3.8), also called MWD (cf. Def. 3.4.2), is NP-complete, cf. (Thm. 3.5.1).*
2. *The problem of determining whether $\mathcal{C}$ has a codeword of weight i, also called CodeWeights (cf. Def. 3.5.2), is NP-complete, cf. (Thm. 3.5.2).*
3. *The problem of determining the distance of $\mathcal{C}$ is NP-complete, cf. [64].*
4. *The problem of determining the number of errors of weight i that satisfy the syndrome equation is in #P-complete, cf. [52].*

5. *The problem of determining the number of codewords in $\mathcal{C}$ of a weight i, also called WE (cf. Def. 3.6.1), is #P-complete, cf. (Thm. 3.6.1).*

One of the main reasons why the decoding problem for linear codes is in NP-complete, is that we have not assumed any particular structure on the linear code. However, the reason behind searching for a decoding strategy for a general linear code is motivated by the results of Shannon in his seminal paper on information theory. He gave a probabilistic proof to show that if one were to randomly choose a linear code, then its rate is very likely to be close to the capacity of the channel, cf. [8]. Despite this advantage, the intractability result which we have summarized in the above lemma, shows that the possibility of there existing a generic efficient decoder for any linear code, is very unlikely.

In [70], the authors analyze the complexity of the minimum weight decoding algorithm for linear codes, when a pre-processing is permitted on the code. This is simply the MWD problem, where the parity check matrix is provided beforehand so that any essential features that are useful for decoding, can be stored before commencing the decoding procedure itself. Their result is rather surprising – the decoding problem remains NP-complete even if the preprocessing is allowed to extend for an infinite time, cf. also [71].

On the other hand, there are also several families of linear codes, for whom an efficient minimum weight decoding algorithm is known (cf. the review in [72]). Some examples include *convolution codes*, cf. [73] and *polar codes*, cf. [74]. One important family of codes for which an efficient decoding algorithm exists[1], were discovered by Robert Gallager in 1962, cf. [9, 75]. These are called *Low-density Parity-check* (LDPC) codes because they can be described using a parity check matrix that is *sparse*, cf. [76, 77, 9]. Since they possess all the desirable features that are expected of an error correcting code, many applications today have adopted LDPC codes as industry standards. Recently, a generalized version of LDPC codes, called *spatially coupled LDPC codes* have been discovered to achieve capacity over the BSC, cf. [78], with an efficient decoding algorithm (whose threshold matches with that of MWD, cf. [79]). Like the polar codes, spatially coupled LDPC codes have been proven to achieve capacity over the BSC. However, the desirable properties of the last two types codes mentioned here, are only observed at block lengths that are too large for practical applications and hence they are not as widely applied as general (irregular) LDPC codes.

We will close the discussion on classical error correction by stating that, to our knowledge, the only two classes of linear codes (with a nontrivial structure), for which the minimum weight decoding problem is known to remain NP-complete, are *Reed Solomon Codes*, cf. [80]

[1] However, it is not strictly equivalent minimum weight decoding, but it has a success probability that is equal to that of minimum weight decoding, in the asymptotic sense.

and LDPC codes, cf. [81]. The complexity of the minimum weight decoding problem for many other known families of codes remains a vast open problem, cf. [71, 82].

Chapter 4

Quantum Codes

Despite the mathematically convenient structure which the linear codes possess, it is strongly believed that they cannot have an efficient general purpose decoder. In this chapter, we will turn to the study of reliable communication of quantum information. We will study the central problem in quantum error correction – understanding how quantum information must be protected against noise. This problem encapsulates the most serious challenge to experimental realizations of the quantum computing techniques.

This chapter is structured as follows. In (Sec. 4.1) we will introduce some formal terminologies involving general quantum codes. In (Sec. 4.2) we will discuss a model to study the effect of errors in a quantum communication process and subsequently in (Sec. 4.3), we will provide a probabilistic model for errors on a quantum channel. In (Sec. 4.4), we will introduce the stabilizer formalism of quantum error correction, introduced by Daniel Gottesman in 1997, which will be used extensively in the rest of this thesis. Meanwhile, in (Sec. 4.2.1) we will introduce the *symplectic representation*, which provides an alternative language to study quantum error correction (compatible with the stabilizer formalism). In (Sec. 4.5), we will develop a strategy for correcting errors on a quantum error correcting code and subsequently discuss the computational complexity of the underlying decoding problem for stabilizer codes (Sec. 4.5.1). In (Sec. 4.6) we will highlight a striking non-classical feature which is native to errors on a quantum channel, that does not have any classical analogue. This is the concept of *degenerate errors*. Then, in (Sec. 4.6) we will discuss how degeneracy affects the decoding strategy and formulate the correct or *optimal* decoding strategy for stabilizer codes. Finally in (Sec. 4.8) we will illustrate all of the concepts discussed so far in this chapter, using the famous example of the *Shor's* $9-$*qubit code*, which also happens to be the first quantum error correcting code to be invented, in 1995, named after Peter Shor.

4.1 A quantum error correcting code

Quantum information is encoded in quantum states. The unit of quantum information is called a *qubit*, represented by the quantum state of a simple two level system (Ex. spin$-1/2$),

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \ , \tag{4.1}$$

where α, β are complex numbers and the orthogonal basis $\{|0\rangle, |1\rangle\}$ is called the *computational basis*. Similarly a $n-$qubit state can be expressed as a linear combination of vectors in the basis $\{|v\rangle\}_{v \in \mathbb{Z}_2^n}$, cf. [23, 24]. A quantum code on $n-$qubits is any subspace of the vector space of all possible $n-$qubit states. We will define a quantum code as follows.

Definition 4.1.1 $[[n,k]]$ *Quantum Code*
A $[[n,k]]$ *Quantum Code, denoted by* $\mathcal{Q}$*, is a* 2^k-*dimensional subspace of the* $\mathcal{H}_2^n$*. All the* $n-$*qubit states in* $\mathcal{Q}$ *are called codewords.*

For instance, the basis vectors for a $[[5,1]]$ quantum code (cf. [83]) are given as follows.

$$\begin{aligned}
|\bar{0}\rangle &= -|00000\rangle + |01111\rangle - |10011\rangle + |11100\rangle + |00110\rangle + |01001\rangle + |10101\rangle + |11010\rangle \\
|\bar{1}\rangle &= -|11111\rangle + |10000\rangle + |01100\rangle - |00011\rangle + |11001\rangle + |10110\rangle - |01010\rangle - |00101\rangle
\end{aligned} \tag{4.2}$$

Remember that the distance of a classical code is also the the least number of bits which must be flipped in a codeword, to obtain another codeword. We will adopt an analogue of this definition to understand the notion of a distance for quantum codes. However, first we must characterize the nature of errors on $n-$qubit states, in quantum mechanics.

4.2 Errors on quantum codes

An error operation is simply any quantum evolution of the codeword, while it is being transmitted. Physically speaking, a quantum system cannot be held in perfect isolation, implying that it will interact with its environment. Therefore, every quantum evolution occurs on the combined state of the codeword *plus* its environment. Let $|\psi\rangle$ be a single qubit codeword (Eq. 4.1). With a slight loss of generality, (cf. Chap. 3 of [24]), we can assume that

the composite state of the codeword plus its environment is $|\psi\rangle \otimes |0\rangle_E$, where $|0\rangle_E$ is the $m-$qubit state of the environment. The quantum evolution of the composite state is given by

$$U|\psi\rangle \otimes |0\rangle_E = \alpha|0\rangle \otimes |e_{00}\rangle + \alpha|1\rangle \otimes |e_{01}\rangle + \beta|0\rangle \otimes |e_{10}\rangle + \beta|1\rangle \otimes |e_{11}\rangle \tag{4.3}$$

for some states $|e_{ij}\rangle$, that denote the evolved states of the environment. Note that the states in $\{|e_{ij}\rangle\}$ are not mutually orthogonal. Therefore, in general, $|\psi\rangle$ undergoes a transformation dictated by a 2×2 matrix, conditioned by the state of the environment. The Pauli matrices (or Pauli errors), denoted by $\mathbb{1}$, X, Y, Z, whose representations are given by

$$\mathbb{1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \ , \ X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \ , \ Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \ , \ Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \ , \tag{4.4}$$

form a basis to express every 2×2 matrix. We can express the (Eq. 4.3) as

$$\begin{aligned} U|\psi\rangle \otimes |0\rangle_E &= \frac{\mathbb{1} + Z}{2}|\psi\rangle \otimes |e_{00}\rangle + \frac{X - iY}{2}|\psi\rangle \otimes |e_{01}\rangle + \frac{X + iY}{2}|\psi\rangle \otimes |e_{10}\rangle + \frac{\mathbb{1} - Z}{2}|\psi\rangle \otimes |e_{11}\rangle \\ &= |\psi\rangle \otimes |e_{\mathbb{1}}\rangle + (X|\psi\rangle) \otimes |e_X\rangle + (Y|\psi\rangle) \otimes |e_Y\rangle + (Z|\psi\rangle) \otimes |e_Z\rangle \ , \end{aligned} \tag{4.5}$$

where the states $|e_{\mathbb{1}}\rangle$, $|e_X\rangle$, $|e_Y\rangle$, $|e_Z\rangle$ are defined in terms of $|e_{ij}\rangle$, as follows, cf. also [24].

$$\begin{aligned} |e_{\mathbb{1}}\rangle &= \frac{1}{2}\left(|e_{00}\rangle + |e_{11}\rangle\right) \ , \ |e_X\rangle = \frac{1}{2}\left(|e_{01}\rangle + |e_{10}\rangle\right) \\ |e_Y\rangle &= \frac{i}{2}\left(|e_{10}\rangle - |e_{01}\rangle\right) \ , \ |e_Z\rangle = \frac{1}{2}\left(|e_{00}\rangle - |e_{11}\rangle\right) \end{aligned}$$

It is easy to observe that the states defined above are not mutually orthogonal. Similarly, while considering the evolution of a $n-$qubit codeword $|\psi\rangle$ plus its $m-$qubit environment $|0\rangle_E$, the codeword undergoes a transformation dictated by some $2^n \times 2^n$ matrix. A particular basis for expressing any $2^n \times 2^n$ matrix is the set of $n-$fold tensor products of Pauli matrices. Formally, this basis set is known as the *Pauli group*, defined as follows.

Definition 4.2.1 *Pauli Group*
The Pauli group $\mathcal{G}_n$ is a matrix group generated by all $n-$fold tensor products of Pauli matrices.

$$\mathcal{G}_n \stackrel{def}{=} \{\epsilon P_1 \otimes \cdots \otimes P_n \ : \ \epsilon \in \{\pm 1, \pm i\} \, , \ P_i \in \{\mathbb{I}, X, Y, Z\}\} \tag{4.6}$$

The evolution of $|\psi\rangle \otimes |0\rangle_E$ can be expressed as

$$U|\psi\rangle \otimes |0\rangle_E = \sum_a (E_a|\psi\rangle) \otimes |e_a\rangle \tag{4.7}$$

where a labels each element of the Pauli group, denoted by E_a, and $\{|e_a\rangle\}$ are evolved states of the environment. Since the receiver does not have access to the environment of the quantum system, the received state is simply a partial trace of the complete density matrix associated to the state in (Eq. 4.7). This partial trace will consist of a sum of terms like $E_{a'}|\psi\rangle\langle\psi|E_{a''}\langle e_{a'}|e_{a''}\rangle$, where $E_{a'}$, $E_{a''}$ are Pauli errors. With a slight loss of generality, one can assume that the states of the environment are orthogonal, i.e, $\langle e_{a'}|e_{a''}\rangle \propto \delta_{a',a''}$, and interpret the above expression by saying that one of the Pauli operators E_a must have occurred on the codeword (with a probability $\langle e_a|e_a\rangle$), while it was being transmitted[1]. Knill and Laflamme showed, in [84], that if an error can be expressed as a linear sum of a finite number of basis operators (as in Eq. 4.7), then it suffices to be able to correct for only those basis operations (i.e, Pauli errors). For a similar reasoning, see [85].

Henceforth, we will consider the direct action of $n-$fold tensor products of Pauli matrices on the $n-$qubit states of the codewords, but we must bear in mind that this action is simply an effective action that results from a general operator (in Eq. 4.7) acting on a larger dimensional space – containing the codeword and its environment. Before devising a strategy to correct Pauli errors, we must learn more about their action on the codewords. Pauli matrices perform the most fundamental operations on a $1-$qubit state (cf. Eq. 4.1),

$$X|\psi\rangle = \alpha|1\rangle + \beta|0\rangle\,,\; Y|\psi\rangle = i\alpha|1\rangle - i\beta|0\rangle\,,\; Z|\psi\rangle = \alpha|0\rangle - \beta|1\rangle\,. \tag{4.8}$$

The Pauli matrix X is called a *bit flip* error, i.e, it exchanges $|0\rangle$ state and $|1\rangle$ states. The Pauli matrix Z is called a *phase flip* error, i.e, it *flips* the relative sign between $|0\rangle$ state and $|1\rangle$ states. The third matrix Y is simply a combination of a phase flip followed by a bit flip, i.e, $Y = iXZ$.

Similarly, when a $n-$fold tensor product of Pauli matrices acts on a $n-$qubit state, every element of the tensor product acts on the corresponding qubit of the state, i.e,

$$X \otimes Z \otimes \mathbb{1} \otimes Z|1110\rangle = X|1\rangle \otimes Z|1\rangle \otimes \mathbb{1}|1\rangle \otimes Z|0\rangle = -|0110\rangle\,. \tag{4.9}$$

[1] This is our first simplification, if the evolved states of the environment are orthogonal, then there is some physical process to distinguish between them.

The *weight* of a $n-$qubit Pauli operator P, denoted by $\mathsf{wt}(P)$, is the number of non-identity single-qubit Pauli operators in the tensor product form of P. For instance,

$$\mathsf{wt}(X \otimes Z \otimes \mathbb{1} \otimes \mathbb{1}) = 2\,,\ \mathsf{wt}(Z \otimes Y \otimes \mathbb{1} \otimes \mathbb{1} \otimes Z) = 3\,,\ \mathsf{wt}(\mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1}) = 0. \tag{4.10}$$

Note that every pair of Pauli operators in $\mathcal{G}_n$ either mutually commute or anti-commute.

Though the Pauli group (as per Def. 4.2.1) contains errors which differ from each other by a scalar in $\{\pm 1, \pm i\}$, this distinction is often unimportant for error detection or correction because these scalars appear as a global phase on the received $n-$qubit state. Since all the information about the received state is obtained throughout quantum measurements, for all purposes in error correction, we can ignore these phases. We will define an *effective Pauli group*, by identifying elements that are related by a scalar in $\{\pm 1, \pm i\}$, cf. [30].

Definition 4.2.2 *Effective Pauli group*
The effective Pauli group, denoted by $\overline{\mathcal{G}}_n$, is a group formed by identifying all errors the Pauli group $\mathcal{G}_n$, cf. (Def. 4.2.1), which are related by a scalar in $\{-1, +i, -i\}$, i.e,

$$\overline{\mathcal{G}}_n \simeq \mathcal{G}_n / \{+1, -1, +i, -i\}\ . \tag{4.11}$$

Each equivalence class of the quotient group has a unique representative in $\overline{\mathcal{G}}_n$, which is the element of the equivalence class corresponding to the phase $+1$.

The effective Pauli group has certain nice properties, which are key to some of the quantum error correction strategies we will discuss in this chapter. Note that $\overline{\mathcal{G}}_n$ is a finite group, with 4^n elements. Every Pauli error in $\overline{\mathcal{G}}_n$ is an unitary and a Hermitian operator.

4.2.1 Symplectic representation

One common method to compare various properties of Pauli errors and with properties of binary sequences, is studied using the *symplectic representation*, c.f [30, 23, 2, 4, 5]. Note that any Pauli operator in $\overline{\mathcal{G}}_n$, denoted by M, can be expressed as the product

$$\mathsf{M} = \mathsf{M}_X \cdot \mathsf{M}_Z \tag{4.12}$$
$$\mathsf{M}_X = (\mathsf{A}_1 \otimes \cdots \otimes \mathsf{A}_n)\ ,\ \mathsf{M}_Z = (\mathsf{B}_1 \otimes \cdots \otimes \mathsf{B}_n)\ ,\ \ \mathsf{A}_i \in \{Z, \mathbb{I}\}\ ,\ \ \mathsf{B}_j \in \{X, \mathbb{1}\}\ ,\ \ 1 \leq i, j \leq n$$

We will represent the $X-$type and the $Z-$type factors (above), separately, as two $n-$bit sequences. For the $X-$type part denoted by M_X, we will assign a $n-$bit sequence denoted

by $\eta_X(\mathsf{M})$ whose n bits are $\eta_X(\mathsf{A}_1)$, $\eta_X(\mathsf{A}_2)$, $\ldots$, $\eta_X(\mathsf{A}_n)$ such that $\eta_X(\mathsf{A}_i) = 1$ if and only if $\mathsf{A}_i = X$ and 0 otherwise. Similarly, to derive the binary sequence for the $Z-$type factor of the Pauli error denoted by M_Z, we will define a $n-$bit sequence denoted by $\eta_Z(\mathsf{M})$ whose bits are by $\eta_Z(\mathsf{B}_1)$, $\eta_Z(\mathsf{B}_2)$, $\ldots$, $\eta_Z(\mathsf{B}_n)$ such that $\eta_Z(\mathsf{B}_i) = 1$ if and only if $\mathsf{B}_i = Z$ and 0 otherwise. Therefore the *symplectic representation* of a Pauli error M, expressed in the form of (Eq. 4.12), is the $2n-$bit sequence expressed in the form

$$\eta(\mathsf{M}) \stackrel{\text{def}}{=} (\eta_X(\mathsf{M}) \,|\, \eta_Z(\mathsf{M})) \,. \tag{4.13}$$

For instance, the Pauli error $Z \otimes X \otimes \mathbb{1} \otimes Y \otimes Z \otimes \mathbb{1}$ is represented by $(010100\,|\,100110)$. Informally speaking, when a Pauli error is expressed in the form in (Eq. 4.12), its symplectic representation is a $2n-$bit sequence with "0"s at positions corresponding to the identity operator in the tensor product form and "1"'s elsewhere.

Let us denote two Pauli operators in $\mathcal{G}_n$, by E and F, expressed as a product of $X-$type and $Z-$type operators as in (Eq. 4.12), i.e, $\mathsf{E} = \mathsf{E}_X \cdot \mathsf{E}_Z$ and $\mathsf{F} = \mathsf{F}_X \cdot \mathsf{F}_Z$. We find that

$$\begin{aligned}
\mathsf{E} \cdot \mathsf{F} &= \mathsf{E}_X \cdot \mathsf{E}_Z \cdot \mathsf{F}_X \cdot \mathsf{F}_Z \\
&= (-1)^{\eta(\mathsf{E}_Z)\cdot\eta(\mathsf{F}_X)}\; \mathsf{E}_X \cdot \mathsf{F}_X \cdot \mathsf{E}_Z \cdot \mathsf{F}_Z \\
&= (-1)^{\eta(\mathsf{E}_Z)\cdot\eta(\mathsf{F}_X)} (-1)^{\eta(\mathsf{E}_X)\cdot\,\eta(\mathsf{F}_Z)}\; \mathsf{F}_X \cdot \mathsf{F}_Z \cdot \mathsf{E}_X \cdot \mathsf{E}_Z \\
&= (-1)^{\eta(\mathsf{E}_Z)\cdot\eta(\mathsf{F}_X)+\eta(\mathsf{E}_X)\cdot\eta(\mathsf{F}_Z)}\; \mathsf{F} \cdot \mathsf{E}
\end{aligned} \tag{4.14}$$

where $\eta(\mathsf{E}_Z) \cdot \eta(\mathsf{F}_X)$ (or, $\eta(\mathsf{E}_X) \cdot \eta(\mathsf{F}_Z)$) is simply the total number of exchanges of Pauli X and Z matrices while exchanging the $n-$fold tensor products E_Z and F_X (E_X and F_Z). Therefore, E will commute with F if and only if the corresponding symplectic representations of E, F denoted by $(a|b)$ and $(c|d)$ respectively, satisfy $a \cdot d + b \cdot c = 0$, cf. (Eq. 4.14). In other words, we say that the *symplectic product* of $(a|b)$ with $(c|d)$ must be 0. The symplectic product between two vectors $(a|b)$ and $(c|d)$ is denoted using "$\odot$" and it is defined as

$$(a|b) \odot (c|d) \stackrel{\text{def}}{=} a \cdot d + b \cdot c \,. \tag{4.15}$$

This enables us to identify a commutation relation between Pauli errors with an orthogonality relation between their symplectic representations.

Let the symplectic representation of a Pauli error E be denoted by $\eta(E)$. Note that the weight of E is no more the Hamming weight of $\eta(E)$ due to the fact the a Y operation gives rise to two "1"'s in $\eta(E)$. In fact, the Hamming weight of $\eta(E)$ is equal to the number X and Z type errors in E, while a $Y-$type error being counted as both a $X-$type as well as a $Z-$type error. The Hamming weight of $\eta(E)$ is instead called the *symplectic weight* of E,

which we will denote by $\mathrm{wt}_2(E)$. This measure of weight will be particularly useful in our analysis of the quantum error correction problem, c.f (Def. 4.3.3), [2, 4, 5].

4.3 Error Model

In the previous chapter, we saw that an error model plays a crucial role in error correction, by specifying an ordering on the errors, according to their probabilities, so that a decoding strategy can choose an error with the highest probability. In quantum setting too, we must first discuss how different Pauli errors can be assigned probabilities and then design a decoding strategy. Due to the different types of Pauli errors, we have several options of assigning probabilities to errors, cf. [23], however, we will consider some common simplifications.

Our first assumption of the quantum channel, is that the effective action of a general unitary evolution on the statistical mixture of the system plus its environment, can be modelled as the action of a Pauli operator (in $\overline{\mathcal{G}}_n$) on the $n-$qubit codeword. We will impose further restrictions on the quantum channel. Like in the classical case, we will assume that the probability of a $n-$qubit Pauli operator is simply the probability of n independent, single-qubit Pauli operators, each of which affect a distinct qubit of the codeword. Since non-trivial errors can be of types X, Y, Z, each of them can have different probabilities. With these assumptions, we will define an *independent Pauli channel*, as follows, cf. [24, 31].

Definition 4.3.1 *Independent Pauli Channel*

On an independent Pauli channel, the probability of a Pauli error, denoted by $E = \bigotimes_{i=1}^{n} E_i$, *is*

$$Prob(E) = \prod_{i=1}^{n} q_{i,E_i} \,, \tag{4.16}$$

where the $q_{i,E_i} \in [0,1]$ *is the probability of the single-qubit Pauli error* E_i, *on the* i^{th} *qubit, and*

$$q_{i,\mathbb{1}} + q_{i,X} + q_{i,Y} + q_{i,Z} = 1 \,, \ \forall\, 1 \leq i \leq n \,. \tag{4.17}$$

The constant q_{i,E_i} *is referred to as the qubit-noise-rate for error* E_i *on qubit* i.

As an aside, note that an independent Pauli channel can be completely specified by $4n$ values of noise-rates, each of which can be specified to an exponential accuracy, using $\mathcal{O}(n)$ bits.

On an independent Pauli channel, the probability vectors $q_i \stackrel{\text{def}}{=} (q_{i,\mathbb{1}}, q_{i,X}, q_{i,Y}, q_{i,Z})$ can be different for different qubits, however when the probability vectors are all identical, the independent Pauli channel is characterized as an *independent and identically distributed* (IID) Pauli channel. A straightforward simplification of the IID Pauli channel is made by assuming that whenever an error occurs on a qubit, it is equally likely to be one of X, Y, Z. This assumption specifies a *depolarizing channel*, which is defined as follows, cf. [24, 23].

Definition 4.3.2 *Depolarizing channel*
On a depolarizing channel, the probability of a $n-$qubit Pauli error E is given by

$$Prob(E) = \left(\frac{p}{3}\right)^{wt(E)} \times (1-p)^{n-wt(E)} \tag{4.18}$$

where $wt(E)$ is the weight of E (cf. 4.10) and $p \in [0,1]$, known as the depolarizing noise rate.

It is easy to observe that the above probability distribution is normalized, i.e,

$$\sum_{E \in \mathcal{G}_n} \left(\frac{p}{3}\right)^{\mathsf{wt}(E)} \times (1-p)^{n-\mathsf{wt}(E)} = \sum_{i=0}^{n} \binom{n}{i} 3^i \left(\frac{p}{3}\right)^i (1-p)^{n-i} = 1\,, \tag{4.19}$$

where the last equality follows from a basic property of binomial coefficients. The key observation in the above derivation is the fact that all Pauli operators of the same weight, have equal probability. Moreover, there are $3^i \binom{n}{i}$ Pauli operators of weight i. So, in (Eq. 4.19), we transformed a sum over 4^n Pauli operators, into a sum over $n+1$ distinct probabilities.

One can interpret the depolarizing channel by saying that for every qubit, the information in the state is completely lost with certain probability $(4p/3)$ and remains intact otherwise[2].

Alternatively, one can consider a simplification of the IID Pauli channel by assuming that each qubit is first affected by a X-type error, then by a Z-type error. So, a Y-type error occurs only when both a $Z-$type as well as an $X-$type error affect a qubit. With this assumption, we will define an *independent $X-Z$ channel*, as follows.

Definition 4.3.3 *Independent $X-Z$ channel*
On an independent $X-Z$ channel, the probability of a $n-$qubit Pauli error E is given by

$$Prob(E) = \left(\frac{p}{2}\right)^{wt_2(E)} \times \left(1-\frac{p}{2}\right)^{2n-wt_2(E)} \tag{4.20}$$

[2]Denoting the density matrix associated to the codeword as ρ, the action of the depolarizing channel with depolarizing noise rate p, is specified as $\rho \mapsto (1-p)\rho + p/3\,(X\rho X + Y\rho Y + Z\rho Z)$, which can be rewritten using the identity $\rho + X\rho X + Y\rho Y + Z\rho Z = \mathbb{1}/2$, as: $\rho \mapsto (1 - 4p/3)\rho + 4p/3 \cdot \mathbb{1}/2$.

where $\mathsf{wt}_2(E)$ is the symplectic weight of E, cf. (Sec. 4.2.1) and $p \in [0,1]$, which we will simply refer to as the qubit noise rate.

The independent $X-Z$ channel will play an important role in our complexity analysis of a decoding problem for stabilizer codes, in (Chap. 5).

One key feature of both the depolarizing channel and the independent $X-Z$ channel is that the probability of an error depends only on its weight (either wt or wt_2), much like their classical counterpart, the binary symmetric channel, cf. (Def. 3.2.1). Notice also that $\mathsf{Prob}(E)$ monotonically decreases with $\mathsf{wt}(E)$ (or $\mathsf{wt}_2(E)$) for $p \in [0, 1/2]$, implying that in such a range, *a minimum weight error* has the maximum probability. This is again similar to the reasoning behind the decoding algorithm for linear codes, in (Sec. 3.4).

4.4 Stabilizer formalism

When a codeword of a $[[n,k]]$ code $\mathcal{Q}$ is transmitted across a channel, the receiver will be certain that an error has occurred if the received $n-$qubit state is no more a codeword of $\mathcal{Q}$. Similar to the classical case, we require an efficient method to verify if any $n-$qubit state is a codeword. There is a class[3] of quantum codes whose codewords can be characterized using a few measurements. More precisely, there are $[[n,k]]$ quantum codes which can be described as *a common eigenspace of a set of commuting Pauli errors*, which is also an Abelian subgroup of the Pauli group, which is called a *stabilizer subgroup*. For instance, the $5-$qubit code presented in (Eq. 4.2) can be expressed as the $+1$ eigenspace of the following set of commuting Pauli errors, cf. [23].

$$XZZXI\,,\ IXZZX\,,\ XIXZZ\,,\ ZXIXZ \tag{4.21}$$

In 1998, Daniel Gottesman in his PhD thesis, cf. [30], showed that one can take advantage of this property in quantum error correction. Such a codes, that can be described as a common $+1$ eigenspace of a stabilizer subgroup of the Pauli group, is called a *stabilizer codes*. Following is a formal definition of a stabilizer code, cf. [23, 31, 32].

[3] These are definitely not the most general quantum codes, for instance, there are $[[n,k]]$ quantum codes which can be expressed as a common eigenspace of a commuting set of operators which are not necessarily Pauli errors, cf. [86] and also [87]. Nonetheless, stabilizer codes are by far the most widely studied codes.

Definition 4.4.1 *Stabilizer codes*
A $[[n,k]]$ stabilizer code is a $[[n,k]]$ quantum code that can be described as the common +1 eigen space of an Abelian and Hermitian subgroup of $\mathcal{G}_n$, called the stabilizer subgroup. For a stabilizer code $\mathcal{Q}$, its stabilizer subgroup is denoted by $\mathcal{S}$ and it is defined as follows.

$$\mathcal{S} = \{S \in \overline{\mathcal{G}}_n \,:\, S|\psi\rangle = |\psi\rangle \,,\ \forall\, |\psi\rangle \in \mathcal{Q}\} \tag{4.22}$$

The elements of the stabilizer subgroup are called stabilizers of $\mathcal{Q}$. Equivalently, for a stabilizer subgroup $\mathcal{S}$, the corresponding stabilizer code $\mathcal{Q}$ is defined as follows.

$$\mathcal{Q} = \{|\psi\rangle \in \mathcal{H}_2^n \,:\, S|\psi\rangle = |\psi\rangle \,,\ \forall S \in \mathcal{S}\}. \tag{4.23}$$

Stabilizer codes are by far the most widely studied quantum codes, c.f [23, 31, 32]. It follows that the stabilizer subgroup of a $[[n,k]]$ quantum code has 2^{n-k} stabilizers and therefore it can be described using $(n-k)$ generators. So, if codeword of a $[[n,k]]$ stabilizer code is transmitted across a channel, the receiver must simply check if the received $n-$qubit state is indeed a $+1$ eigenstate of all the generators of the corresponding stabilizer group. First, let us derive a convenient basis to express Pauli errors, in which we can study the effect of various errors on the codewords of a stabilizer code.

4.4.1 Operator basis

Let $\mathcal{Q}$ denote a $[[n,k]]$ stabilizer code, with stabilizer generators $\{S_1, \ldots, S_{n-k}\}$ and $|\psi\rangle$ be some codeword of $\mathcal{Q}$. By definition, the action of all stabilizers on $|\psi\rangle$ is trivial. From basic linear algebra, we know that the action of any operator that commutes with all the stabilizer generators, on $|\psi\rangle$, must not alter its eigenvalue, i.e, $S_i \cdot N|\psi\rangle = N|\psi\rangle$ for every N such that $[S_i, N] = 0$. In other words, N must take a codeword of $\mathcal{Q}$ to another codeword of $\mathcal{Q}$ (not necessarily the same one). Mathematically, we denote the set of all the Pauli operators which commute with every stabilizer generator, as the *normalizer*[4] *of the stabilizer subgroup in the Pauli group*, denoted by $\mathcal{N}(\mathcal{S})$. Clearly, $\mathcal{S} \subseteq \mathcal{N}(\mathcal{S})$ because $\mathcal{S}$ is Abelian. Since $\mathcal{S}$ has $n-k$ independent generators, the total number of Pauli operators in $\mathcal{N}(\mathcal{S})$ is simply $4^n \Big/ 2^{n-k} = 2^{n+k}$. Furthermore, $\mathcal{N}(\mathcal{S})$ is also a subgroup of the Pauli group which has $n+k$ independent generators. One can derive a choice of generators for $\mathcal{N}(\mathcal{S})$ such that, of the $n+k$ generators, $n-k$ of them are stabilizer generators.

[4] Strictly speaking, it must be called the *Centralizer* of the stabilizer subgroup in the Pauli group, denoted by $\mathcal{Z}(\mathcal{S})$. Due to the commutation anti commutation duality of Pauli errors, it turns out that $\mathcal{Z}(\mathcal{S}) = \mathcal{N}(\mathcal{S})$.

Hence a normalizer certainly leaves $\mathcal{Q}$ globally invariant, i.e, it simply performs a permutation of codewords (including the identity permutation). An error that performs a nontrivial permutation (i.e, a permutation that is not equivalent to the identity permutation) of the codewords can be multiplied with any stabilizer, yet yielding the same nontrivial permutation. Therefore, we can identify a subset of errors in $\mathcal{N}(\mathcal{S})$, denoted by $\mathcal{L}$, each of which perform *distinct* permutations of codewords in $\mathcal{Q}$, so that every normalizer is simply a product of an error in $\mathcal{L}$ with a stabilizer. Since the elements of $\mathcal{L}$ perform logical operations on the codewords, they are called *logical errors*. Mathematically, we can express

$$\mathcal{L} \simeq \mathcal{N}(\mathcal{S})/\mathcal{S} \ . \tag{4.24}$$

For instance, $X^{\otimes 5}$, $Z^{\otimes 5}$ are logical errors for the $5-$qubit code, in (Eq. 4.2) and

$$X \otimes X \otimes X \otimes X \otimes X|\bar{0}\rangle = |\bar{1}\rangle \ , \ X \otimes X \otimes X \otimes X \otimes X|\bar{1}\rangle = |\bar{0}\rangle \tag{4.25}$$

$$Z \otimes Z \otimes Z \otimes Z \otimes Z|\bar{0}\rangle = |\bar{0}\rangle \ , \ Z \otimes Z \otimes Z \otimes Z \otimes Z|\bar{1}\rangle = -|\bar{1}\rangle \ . \tag{4.26}$$

Since $\mathcal{N}(\mathcal{S})$ has 2^{n+k} operators and $\mathcal{S}$ has 2^{n-k} stabilizers, the quotient relation in (Eq. 4.24) implies that $\mathcal{L}$ has 2^{2k} operators. Furthermore, $\mathcal{L}$ can be described using $2\,k$ generators, they are called *logical generators*, denoted by $\{\bar{X}_i, \bar{Z}_i\}_{i=1}^{k}$. Due to the commutation anti-commutation duality of the Pauli group, it turns out that the logical generators can be arranged into k mutually anti-commuting pairs, i.e, $\{\bar{X}_i, \bar{Z}_i\} = 0$, for every i between 1 and k. Therefore, the generators of $\mathcal{N}(\mathcal{S})$ are simply $\{S_1, \ldots, S_{n-k}, \bar{Z}_1, \bar{X}_1, \ldots, \bar{Z}_k, \bar{X}_k\}$.

Besides the operators in $\mathcal{N}(\mathcal{S})$, there are those which do not commute with some stabilizer generator. In other words, the action of such an operator on $|\psi\rangle$, necessarily yields a state that is a not a valid codeword. These errors are called *pure errors*, denoted by the set $\mathcal{T}$. Again, due to the commutation anti-commutation duality of Pauli errors, for every pure error T, there is at least one stabilizer generator S_i such that $S_i \cdot T|\psi\rangle = -|\psi\rangle$. Furthermore, action of the product of a pure error with any normalizer, on $|\psi\rangle$, will still yield a state that is not a valid codeword. In other words, we can express every Pauli operator as a product of a pure error with a normalizer. Mathematically, we can express

$$\mathcal{T} \simeq \overline{\mathcal{G}}_n/\mathcal{N}(\mathcal{S}) \ . \tag{4.27}$$

It follows that $\mathcal{T}$ consists of (mutually commuting) Pauli operators, each of which anti-commute with a unique set of stabilizer generators. The number of elements in $\mathcal{T}$ is $4^n \Big/ 2^{n+k} = 2^{n-k}$ and we will denote the $(n-k)$ generators of $\mathcal{T}$ by $\{T_1, \ldots, T_{n-k}\}$. Furthermore, one can choose these generators such that $\{S_i, T_i\} = 0$ and $[S_i, T_j] = 0$, for all $i \neq j$. For instance,

$X \otimes \mathbb{1}^{\otimes 3} \otimes X$ and $\mathbb{1} \otimes X \otimes X \otimes \mathbb{1}^{\otimes 2}$ are pure errors for the $5-$qubit code in (Eq. 4.2).

The following lemma summarizes the above discussion by providing a canonical generating set for each $\mathcal{T}$, $\mathcal{S}$ and $\mathcal{L}$, cf. also [2, 23, 30].

Lemma 4.4.1 *Canonical generators of the Pauli group*
Let $\mathcal{S}$ be a stabilizer group for a $[[n,k]]$ stabilizer code. There is a canonical generating set for each $\mathcal{S}$, $\mathcal{L}$, $\mathcal{T}$ (cf. Eqs. 4.24, 4.27), denoted by $\{S_i\}_{i=1}^{n-k}$, $\{\bar{X}_i, \bar{Z}_i\}_{i=1}^{k}$ and $\{T_i\}_{i=1}^{n-k}$ respectively, such that

$$[S_i\,,\,S_j] = [T_i\,,\,T_j] = [S_i\,,\,\bar{X}_i] = [S_i\,,\,\bar{Z}_i] = [T_i\,,\,\bar{X}_i] = [T_i\,,\,\bar{Z}_i] = 0\,,\ \forall\ i,j\,, \tag{4.28}$$

$$[S_i\,,\,T_j] = [\bar{Z}_i\,,\,\bar{X}_j] = 0\,,\ \forall\ i \neq j\,, \tag{4.29}$$

$$\{\bar{Z}_i\,,\,\bar{X}_i\} = \{S_i\,,\,T_i\} = 0\,. \tag{4.30}$$

Therefore, we have an operator basis for the Pauli group given by $\overline{\mathcal{G}}_n \simeq \mathcal{T} \times \mathcal{N}(\mathcal{S}) = \mathcal{T} \times \mathcal{L} \times \mathcal{S}$, so, every Pauli error E is a product of errors in the respective sets $\mathcal{S}$, $\mathcal{L}$, $\mathcal{T}$, i.e,

$$E = T \cdot L \cdot S \qquad (\text{where } T \in \mathcal{N},\, L \in \mathcal{L} \text{ and } S \in \mathcal{S})\,. \tag{4.31}$$

Decomposition in this basis will be particularly useful to formulate the decoding problem and it has also been recalled in [34, 2, 4, 5, 3]. We will often refer to T, S, L as the support of E in the sets $\mathcal{T}$, $\mathcal{S}$, $\mathcal{L}$ respectively.

As a result of the above decomposition, one can view the probability of an error, defined according to some Pauli channel, as a joint distribution over its supports in the respective sets $\mathcal{S}, \mathcal{T}, \mathcal{L}$, as $\mathsf{Prob}(E) = \mathsf{Prob}(T,\, L,\, S)$. Extending this notion, one can also define marginal probabilities $\mathsf{Prob}(L)$, $\mathsf{Prob}(T)$ by

$$\mathsf{Prob}(L) = \sum_{S \in \mathcal{S}} \sum_{T \in \mathcal{T}} \mathsf{Prob}(T,\, L,\, S) \tag{4.32}$$

$$\mathsf{Prob}(T) = \sum_{S \in \mathcal{S}} \sum_{L \in \mathcal{L}} \mathsf{Prob}(T,\, L,\, S) \tag{4.33}$$

where $\mathsf{Prob}(L)$ should be interpreted as the probability of a Pauli error to perform the logical operation L and $\mathsf{Prob}(T)$ as the probability of a Pauli error to perform the pure error operation T, on the codewords of $\mathcal{Q}$.

Recall that the distance of a quantum code is simply the minimum number of single qubit operations that are required to transform any codeword to a different codeword. Since the effect of errors in $\mathcal{N}(\mathcal{S})\backslash\mathcal{S}$ on a codeword necessarily results in a different codeword,

the distance of a stabilizer code is simply the smallest weight of an error in this set, i.e,

$$d = \min_{E \in \mathcal{N}(\mathcal{S}) \setminus \mathcal{S}} \mathsf{wt}(E) \ . \tag{4.34}$$

We can specify a $[[n,k]]$ code whose distance is d, as a $[[n,k,d]]$ stabilizer code. For instance, the distance of the $5-$qubit code in (Eq. 4.2) is 3, so it can be described as a $[[5,1,3]]$ code.

There is an alternate language to study the stabilizer formalism. This is developed using the symplectic representation of Pauli operators, which we discussed in (Sec. 4.2.1). Remember that the commutation relation between two Pauli operators translates to an orthogonality condition between their corresponding symplectic representations, cf. (Eq. 4.14). Using this, we can discover several analogies between stabilizer codes and linear codes (which we discussed in Sec. 3.3). In particular, it turns out that a $[[n,k]]$ stabilizer code can be specified using $(n+k)$ parity constraints.

Let H denote a $(n-k) \times 2n$ matrix, whose each row contains the symplectic representation of a stabilizer generator, cf. (Eq. 4.13). Since all the stabilizer generators mutually commute, all the rows of H will be mutually orthogonal (w.r.t the symplectic product, cf. 4.15). Consequently, we can interpret H to be the parity check matrix of a self-dual code (cf. Def. 3.5). For this reason, stabilizer codes are sometimes called *weakly self-dual codes*, c.f [30]. The symplectic representation of any error in $\mathcal{N}(\mathcal{S})$ will be also orthogonal to every row of H. It follows that, H serves as a parity check matrix for the *symplectic code* corresponding to $\mathcal{N}(\mathcal{S})$, i.e, $\{(u|v) \in \mathbb{Z}_2^{2n} : \mathsf{H} \odot (u|v) = 0\}$. Similarly, a $(n+k) \times 2n$ matrix whose rows contain the symplectic representations of the generators of $\mathcal{N}(\mathcal{S})$, will serve as a parity check matrix for the symplectic code corresponding to $\mathcal{S}$. This analogy forms the crucial part of the study of quantum error correction problems in [2, 4, 5, 30].

4.4.2 Degenerate errors

Suppose $|\psi\rangle$ is a codeword of some $[[n,k]]$ quantum code $\mathcal{Q}$ which is transmitted across a channel, and the error while transmission is some non-trivial stabilizer, denoted by S. In this case, the qubits of $|\psi\rangle$ are certainly affected in a non-trivial manner, but surprisingly the received state $S|\psi\rangle$ is the same as the transmitted codeword $|\psi\rangle$ itself, by the definition of the stabilizers, cf. (Eq. 4.22). For instance, if a codeword of the $5-$qubit code, presented in (Eq. 4.2), is affected by the error $Z \otimes X \otimes \mathbb{1} \otimes X \otimes Z$, then despite error acting non-trivially on each qubit, it leaves the codeword unchanged. This is a special property of errors on a quantum channel that cannot be compared to errors on a classical channel. Recall that on a classical channel, the only type of error which leaves every codeword unchanged is

the trivial error, or equivalently, the trivial sequence. It means that even though a quantum channel can perform non-trivial operations on the qubits of the codeword, there can be some situations where the net result of the operation does not affect the codeword at all.

Take any two Pauli errors which are equal up to a stabilizer, i.e, $E,\ E' \in \overline{\mathcal{G}}_n$ such that

$$E' = E \cdot S \tag{4.35}$$

for any stabilizer $S \in \mathcal{S}$. As a result of the above discussion, we will notice that even though E and E' perform completely different operations on each qubit of a $n-$qubit state, the action of E and E' are nevertheless exactly the same on every codeword of $\mathcal{Q}$. Errors such as $E,\ E'$ that have the same action on every codeword of $\mathcal{Q}$ are called *degenerate errors*. If E is decomposed in the operator basis (cf. Eq. 4.31) as $E = T \cdot L \cdot S$, then, all the 2^{n-k} errors in the set $\{T \cdot L \cdot S' : S' \in \mathcal{S}\}$ are degenerate errors. Since all the errors in this set perform the same logical operation on every codeword of the stabilizer code, as E, we will refer to this set of degenerate errors as the *logical class* of E. It follows that, if any error in the logical class of E, occurs during transmission, causing a codeword $|\psi\rangle$ to be received as $|\phi\rangle$, then, in order to map $|\phi\rangle$ back to $|\psi\rangle$, we can apply any error in the logical class of E, not necessarily the actual error that occurred during transmission.

We have not yet formulated a strategy for error correction, so we cannot precisely tell what role degeneracy plays in quantum error correction. However, it might seem that degenerate errors are good for quantum error correction purposes simply because they need not be corrected separately. This is only partially true. We will elaborate on the effect of degenerate errors in quantum error correction, in (Sec. 4.6).

4.5 Correcting errors on a stabilizer code

The scenario for quantum error correction is the following. A codeword of a $[[n, k]]$ stabilizer code $\mathcal{Q}$ is sent across an quantum channel and it results in the received state $|\phi\rangle$. Strictly speaking, the receiver does not have complete knowledge of the quantum state $|\phi\rangle$, but can only make measurements on $|\phi\rangle$. The problem of error correction is to map $|\phi\rangle$ back to the original encoded state $|\psi\rangle$, using some measurements on $|\phi\rangle$. This is equivalent to inferring a Pauli error E that results in $|\phi\rangle$ starting from the transmitted codeword $|\psi\rangle$, cf [23]. Since any Pauli error can be decomposed into the product $E = T \cdot L \cdot S$ where $T \in \mathcal{T}$, $L \in \mathcal{L}$, $S \in \mathcal{S}$ (cf. Eq. 4.31), it suffices for the receiver to determine the T, L, S factors in this decomposition. In order to deduce a method to determine the various errors in the decomposition of E, we

will recall the effect of errors in each class $\mathcal{S}$, $\mathcal{L}$ and $\mathcal{T}$, on a codeword of the $[[n,k]]$ stabilizer code, from (Sec. 4.4.1).

Let us suppose that the error E, which occurred during transmission, contains a nontrivial support in $\mathcal{T}$, i.e, $T \neq \mathbb{1}$ in (Eq. 4.31). In this case, at least one of the stabilizer generators will anti commute with E. It turns out that E anti-commutes with a stabilizer generator S_i, if and only if the received $n-$qubit state $|\phi\rangle$ is a -1 eigenstate of S_i. Thus, upon measurement of each stabilizer generator on $|\phi\rangle$, we must obtain a result $+1(-1)$ indicating that the respective error has commuted (anti-commuted) with the corresponding stabilizer generator, i.e,

$$S_j|\phi\rangle = S_j \cdot E|\psi\rangle = \begin{cases} E \cdot S_j|\psi\rangle & = E|\psi\rangle \quad = +|\phi\rangle \quad \text{if } E \cdot S_j = S_j \cdot E \\ -E \cdot S_j|\psi\rangle & = E|\psi\rangle \quad = -|\phi\rangle \quad \text{if } E \cdot S_j = -S_j \cdot E \end{cases} . \tag{4.36}$$

We say that an error is *detected* if the received $n-$qubit state is a -1 eigenstate of any of the stabilizer generators. To formalize this notion, the outcomes of measurement of all the stabilizer generators is stored as a $(n-k)-$bit sequence, like in (Def. 3.4.1). This $(n-k)-$bit sequence is called the *error syndrome* and it can be combined with (Eq. 4.36) as follows.

Definition 4.5.1 *Error Syndrome*
Let $\mathcal{Q}$ be a $[[n,k]]$ stabilizer code whose generators are expressed in the canonical form (cf. Lemma 4.4.1) and E be some $n-$qubit Pauli error that results in the received $n-$qubit state $|\phi\rangle$. The error syndrome of E, denoted by $s(E)$, is a $(n-k)-$bit vector whose i^{th} bit is defined to be 0 if the result of measuring S_i on $|\phi\rangle$ is $+1$ (in other words, $[E, S_i] = 0$) and 1 otherwise.

Hence an error has certainly occurred if the observed syndrome differs from the all zero sequence. Conversely, if the syndrome is the all zero sequence, the received state will indeed be a codeword, however we cannot be certain that it will match the transmitted codeword. In other words, since all the errors in $\mathcal{L}$, $\mathcal{S}$ commute with the stabilizer generators, the error syndrome of E is completely independent of S and T (support of E in $\mathcal{S}$ and $\mathcal{L}$ respectively).

Like in the classical case, the error syndrome conveys more information that merely indicating the presence of an error. Since each generator of $\mathcal{T}$ anti commutes with a unique stabilizer generator (c.f Lemma. 4.4.1), each element of $\mathcal{T}$ anti commutes with a unique set of stabilizer generators. If the i^{th} bit of the observed error syndrome of a $n-$qubit state $|\phi\rangle$ is 1, it indicates that the result of measuring S_i on $|\phi\rangle$ is -1. So, any error that results in $|\phi\rangle$ must have its support in $\mathcal{T}$ as T_i. Therefore, if the bits of the observed syndrome are $s_1 s_2 \ldots s_{n-k}$,

then the support of the corresponding error in $\mathcal{T}$, denoted by T in (Eq. 4.31), must be

$$T_s = \prod_{i=1}^{n-k} T_i^{s_i} \ . \tag{4.37}$$

Therefore, $s(E)$ completely fixes the support of the Pauli error in $\mathcal{T}$. In fact, it only fixes T.

Let us discuss how we must determine the support of the error E in $\mathcal{L}$ and $\mathcal{S}$ respectively. Since the error syndrome of E does not reveal any information about S and L, we can only say that E is one of the 2^{n+k} errors in $T_s \cdot \mathcal{N}(\mathcal{S})$ where T_s is fixed by the syndrome, c.f (Eq. 4.37). However, different errors in $T_s \cdot \mathcal{N}(\mathcal{S})$ can lead to different inferences for the transmitted codeword. This must not be a striking difficulty because it is also observed in the classical case, cf. (Sec. 3.4), where the error syndrome can only tell that the actually error in transmission is one of the 2^k errors in a coset of the linear code.

At this stage, we must employ a *best guess* for the support of E in $\mathcal{L}$ and $\mathcal{S}$. The strategy of decoding is similar to the minimum weight decoding strategy for linear codes (cf. Def. 3.4.2) – given a syndrome s and a Pauli channel, we will find the error in the coset $T_s \cdot \mathcal{N}(\mathcal{S})$ that has the maximum probability. Therefore, adopting the operator decomposition in (Eq. 4.31), the decoding strategy must result in the error, denoted by $E_D(s)$, which satisfies

$$E_D(s) = T_s \cdot \mathsf{ArgMax}_{L \in \mathcal{L}, S \in \mathcal{S}} \, \mathsf{Prob}(L \, , \, S \mid T_s) \ , \tag{4.38}$$

where the conditional probability is defined by

$$\mathsf{Prob}(L, S \mid T_s) = \frac{\mathsf{Prob}(L, S, T_s)}{\mathsf{Prob}(T_s)} \tag{4.39}$$

and the marginal $\mathsf{Prob}(T_s)$ is defined in (Eq. 4.33).

When we think of decoding as a computational problem, a technical difficulty must be addressed for the problem to be interesting. While decoding over a Pauli channel (cf. Def. 4.3.1), suppose the maximum probability in (Eq. 4.38) is a *close* tie. Surely, there cannot be a polynomial time decoding algorithm which aims at comparing probabilities of errors, to an accuracy that cannot be afforded by a polynomial number of bits. So, we will characterize the complexity of the decoding problem, based on a gap that exists between the probability of the most likely error and any other error, for the given syndrome. We can define the decoding problem as before, but we will tolerate that whenever this gap in the problem is less than a certain threshold (which we will refer to as a *promise gap*), the decoder is allowed to output any error in $\mathcal{N}(\mathcal{S})$. Finally, we can formally define the decoding problem, called *Quantum Maxmimum Likelihood Decoding* or (QMLD), cf. [2, 4, 5], as follows.

Definition 4.5.2 *Quantum Maximum Likelihood Decoding (QMLD)*
Input: *The generators of a $[[n,k]]$ stabilizer code with stabilizer group $\mathcal{S}$, an independent Pauli channel to specify $\mathsf{Prob}(E)$ for all $E \in \overline{\mathcal{G}}_n$, an error syndrome $s \in \mathbb{Z}_2^{n-k}$ and a promise gap Δ.*

Promise: *There exists a couple $S^\star \in \mathcal{S}$, $L^\star \in \mathcal{L}$ such that for all $L \in \mathcal{L}$, $S \in \mathcal{S}$,*

$$\mathsf{Prob}(L^\star, S^\star | T_s) - \mathsf{Prob}(L, S \mid T_s) \geq \Delta \mathsf{Prob}(L^\star, S^\star \mid T_s) \ , \quad (L,S) \neq (L^\star, S^\star) \tag{4.40}$$

Output: *If the promise is satisfied, output $L^\star S^\star$. Otherwise output any error in $\mathcal{N}(\mathcal{S})$.*

For the special cases of the depolarizing channel and the independent $X-Z$ channel (cf. Def. 4.3.2, 4.3.3), the search for an error with maximum probability is formally equivalent to the search for an error of minimum weight (with two possible notions of weight). Consequently, QMLD is also known as a *minimum-weight decoding* strategy and the promise gap in the above definition is irrelevant. Therefore, while analyzing the computational complexity QMLD on the depolarizing or the independent $X-Z$ channel, one can set Δ to 0, cf. [2, 4, 5].

4.5.1 Computational complexity of QMLD

In the previous section, we saw that the minimum weight decoding problem for stabilizer codes, also called QMLD, closely resembles the classical minimum weight decoding problem in (Sec. 3.4). In particular, both the decoding strategies involve an unstructured search over an exponentially large set. Comparing the nature of QMLD with some of the examples for NP-complete problems which we saw in (Sec. 2.2) gives an indication that QMLD must be an intractable problem. Moreover, the resemblance of QMLD with the classical decoding problem through the symplectic representation (cf. Sec. 4.2.1), strengthens our suspicion.

In 2011, Min-Hsiu Hsieh and Francois Le Gall proved that indeed QMLD is an NP-complete problem, cf. [2]. Their main result can be summarized by the following theorem.

Theorem 4.5.1 *Computational complexity of QMLD, cf. [2]*
The computational complexity of determining a Pauli error that has the lowest symplectic weight and a given syndrome, called QMLD (cf. Def. 4.5.2), can be summarized by

$$\mathsf{QMLD} \in \mathsf{NP\text{-}complete} \ . \tag{4.41}$$

The authors in [2] showed a polynomial time Karp reduction (cf. Def. 2.1.3) from the NP-complete classical decoding problem, ErrorWeights (cf. Def. 3.5.1), to QMLD. The essential

ingredient of the proof is the symplectic representation of Pauli errors which we have discussed in (Sec. 4.2.1). In [2] the authors have formulated the QMLD problem in the symplectic representation and then used the minimum weight decoding strategy (MWD) for linear codes which we discussed in (Def. 3.4.2). That QMLD $\in$ NP can be easily established with the help of the stabilizer formalism – one needs to check at most $(n-k)$ commutation relations to conclude whether an error indeed has the given syndrome. We will omit a restatement of the reduction proof since it involves quite a lot of technical manipulations. See also [4, 5] for some variants of QMLD, which are also shown to be intractable.

4.6 Decoding with degenerate errors

Suppose the error that actually occurred during transmission is $E = T_s \cdot L \cdot S$ but the minimum weight error is $E \cdot S'$ for some stabilizer S', then the output of QMLD will be technically incorrect, i.e it is not quite the actual error which occurred in transmission. But applying this incorrect output will return the received $n-$qubit state to the same codeword as if we had guessed the right error. In this sense, the QMLD strategy effectively succeeds. E and $E \cdot S'$ are degenerate errors, cf. (Eq. 4.35). Therefore, even though we did not account for the presence of degenerate errors in the design of QMLD, our negligence seems to affect the decoding strategy in a positive way. But this is only partially true.

For a particular syndrome s, we can decompose the set of all candidate errors into 2^{2k} logical classes, where each class can be identified with a logical error L_i, as (cf. Sec. 4.4.2)

$$T_s \cdot L_i \cdot \mathcal{S} = \{T_s \cdot L_i \cdot S' : S' \in \mathcal{S}\} \tag{4.42}$$

and furthermore, all errors in a logical class have identical action on every codeword. We will explain how degenerate errors affect the decoding strategy for stabilizer codes. Suppose a codeword $|\psi\rangle$ is transmitted across a channel, resulting in the $n-$qubit state $|\phi\rangle$ to be received. If the observed syndrome is s, then the error that occurred during transmission must be $E = T_s \cdot L_i \cdot S$ for some $L_i \in \mathcal{L}$ and a $S \in \mathcal{S}$, which achieves $E|\psi\rangle = |\phi\rangle$. However, it will be incorrect to claim that the probability of receiving $|\phi\rangle$ is precisely $\mathsf{Prob}(E)$, simply because E is not the only error that satisfies $E|\psi\rangle = |\phi\rangle$ – any error in the logical class $T_s \cdot L_i \cdot \mathcal{S}$ can result in the received state $|\phi\rangle$ whenever $|\psi\rangle$ is transmitted, since $S'|\psi\rangle = |\psi\rangle\,,\ \forall\, S' \in \mathcal{S}$, cf. (Eq. 4.22). Consequently, the probability of receiving $|\phi\rangle$ is *the sum of probabilities of all errors in the logical class* L_i, cf. also [88, 35, 34, 33, 2, 36], which is given by the expression

$$\mathsf{Prob}(L_i\,,\, T_s) = \sum_{S' \in \mathcal{S}} \mathsf{Prob}(T_s\,,\, L_i\,,\, S')\ . \tag{4.43}$$

A decoding strategy must therefore select the error that lies in the *most probable equivalence class, instead of the error which is globally the most probable.* Given a particular syndrome s, the probability of all errors in a logical class can expressed by

$$\mathsf{Prob}(L \mid T_s) = \sum_{S \in \mathcal{S}} \mathsf{Prob}(L\,,\, S \mid T_s)\;, \tag{4.44}$$

where again, the conditional probability is defined as before (cf. Eq. 4.39). So, a decoding strategy which accounts for degenerate errors must therefore select the logical error $L \in \mathcal{L}$ such that $\mathsf{Prob}(L \mid T_s)$, as defined above, is maximum, over all $L \in \mathcal{L}$. This is also referred to as an *optimal* decoding strategy. On the other hand, we say that QMLD strategy is suboptimal – it does not result in obtaining the most probable transmitted codeword for a given syndrome. Equivalently, the QMLD strategy does not yield the most probable logical error that must be performed on the received state, to get back the original transmitted codeword.

In order to circumvent the technical difficulty arising due to a close tie in the maximum probabilities, we will assume a relative *gap* to exist between the probabilities of the most likely logical class and any other logical class, for the given syndrome. When this gap in the problem is below a given value, called a *promise gap*, we will allow the decoder to output any logical error. The decoding strategy we have discussed so far, is called *Degenerate Maximum Likelihood Decoding* (DQMLD) and is formulated as follows, cf. [2].

Definition 4.6.1 *Degenerate Maximum Likelihood Decoding (DQMLD)*

Input: *The generators of a $[[n,k]]$ stabilizer code with stabilizer group $\mathcal{S}$, a memoryless Pauli channel to specify $\mathsf{Prob}(E)$ for all $E \in \overline{\mathcal{G}}_n$, an error syndrome $s \in \mathbb{Z}_2^{n-k}$ and a promise gap Δ.*

Promise: *There exists logical error $L^\star \in \mathcal{L}$ such that for all $L \in \mathcal{L}$,*

$$\mathsf{Prob}(L^\star \mid T_s) - \mathsf{Prob}(L \mid T_s) \geq \Delta \mathsf{Prob}(L^\star \mid T_s)\;,\quad L \neq L^\star\;. \tag{4.45}$$

Output: *If the promise is satisfied, output $L^\star$. Else, output any logical error.*

Note also that for a fixed T_s, the probabilities $\mathsf{Prob}(L, S \mid T_s)$ and $\mathsf{Prob}(L, S, T_s)$ differ only by a constant, so we can perform the optimization in DQMLD on the joint probability instead of the conditional probability.

Assuming that the probability of errors are given by the independent $X - Z$ channel (in

cf. Def. 4.3.3), the sum appearing in the joint probability

$$\mathsf{Prob}(T_s, L, S) = \left(1 - \frac{p}{2}\right)^{2n} \sum_{S \in \mathcal{S}} \tilde{p}^{\mathsf{wt}_2(T_s \cdot L \cdot S)} \,. \tag{4.46}$$

(where $\tilde{p} = p/(2-p)$ and wt_2 is the symplectic weight) being over 2^{n-k} terms, forbids a polynomial-time direct computation of its value. However, for the above case (and also when the probabilities of errors are given by the depolarizing channel), by grouping terms of equal weight in the sum, we arrive at a sum involving only $n+1$ terms, given by

$$\mathsf{Prob}(T_s, L, S) = \left(1 - \frac{p}{2}\right)^{2n} \sum_{i=0}^{n} A_i(s, L)\, \tilde{p}^i \,, \tag{4.47}$$

$$\text{where } A_i(s, L) = |\{S \in \mathcal{S} \;:\; \mathsf{wt}_2(E = T_s \cdot L \cdot S) = i\}| \,, \text{ and } \sum_{i=0}^{n} A_i(s, L) = 2^{n-k} \,. \tag{4.48}$$

Compare the above polynomial in (Eq. 4.47) to the *weight enumerator polynomial* in (Def. 3.6.1). We will refer to the coefficients in $\{A_i(s, L)\}_{i=0}^{n}$ as the *weight enumerator coefficients of the coset associated to T_s and L*. The coset weight enumerators play a very important role in estimating the decoder performances of both QMLD as well as DQMLD.

4.6.1 Decoding performance and the promise gap

In this section we will comment on the probability of failure of the DQMLD strategy. In particular, we will emphasize the significance of the promise gap in the decoding problem. Once again, note that DQMLD is a deterministic strategy – it will always output the logical error with the maximum probability for a given syndrome. However, if the actually error that occurred during transmission is not in the logical class that has the highest probability, then DQMLD simply fails. For a given syndrome s, a code stabilized by $\mathcal{S}$, let us denote the probability of failure of DQMLD, as $P_e(s)$ and the output of DQMLD as $L^\star$. We find that

$$P_e(s) = \mathsf{Prob}(T_s) - \mathsf{Prob}(L^\star,\, T_s) = \mathsf{Prob}(T_s)\left[1 - \mathsf{Prob}(L^\star \,|\, T_s)\right] \,. \tag{4.49}$$

Let us turn to the promise gap condition in (Eq. 4.45), which can be restated as

$$(1 - \Delta)\mathsf{Prob}(L^\star \,|\, T_s) \geq \mathsf{Prob}(L \,|\, T_s) \,. \tag{4.50}$$

Let us substitute a large promise gap, given by

$$\Delta = 1 - \frac{\epsilon}{4^k - 1} \,, \tag{4.51}$$

where $0 \leq \epsilon \leq 1$, in (Eq. 4.50) and sum over all logical errors, $L \in \mathcal{L}$, $L \neq L^\star$. We will find

$$\begin{aligned} &\epsilon \mathsf{Prob}(L^\star \mid T_s) \geq 1 - \mathsf{Prob}(L^\star \mid T_s) \ , \\ &\Rightarrow \ \mathsf{Prob}(L^\star \mid T_s) \geq \frac{1}{1+\epsilon} \geq 1 - 2\epsilon \ . \end{aligned} \tag{4.52}$$

Combining the above condition with (Eq. 4.49), summing over all syndromes and denoting the total probability of error $\bar{P}$, we find

$$\bar{P} \leq 2\epsilon \ . \tag{4.53}$$

This shows that the promise gap, (more importantly, a relative gap, as opposed to an additive promise gap like the one suggested in [67]) is closely related to the probability of failure of DQMLD – larger the promise gap, lower is the probability of decoding failure. In the subsequent section, we will mention a particular window for Δ, which subsequently implies a particular window for the decoding failure rates, under which DQMLD can be replaced by QMLD – the two decoding strategies produce the same outputs on all inputs.

4.7 Importance of degeneracy in decoding

The distinction of DQMLD with QMLD and other types of decoders may still seem a little subtle. Before addressing the computational complexity of DQMLD, we will discuss its practical relevance. For a given syndrome and a stabilizer code, *the two decoders QMLD and DQMLD will provide different outputs whenever the most likely logical class does not contain the error with the highest probability.*

Consider a hypothetical scenario where the logical class of an error L_1 contains a single error of weight a and $2^{n-k} - 1$ errors of weight b, where $a \ll b$. Furthermore, the logical class of another error L_2 contains 2^{n-k} errors of some weight c, where $a \ll c \ll b$. The probabilities of these two logical classes are given by

$$\mathsf{Prob}(L_1) \propto \tilde{p}^a + (2^{n-k} - 1)\tilde{p}^b \approx \tilde{p}^a \text{ and } P(L_2) \propto 2^{n-k}\tilde{p}^c \ . \tag{4.54}$$

where $\tilde{p} = p/2(1-p)$.

The QMLD strategy will chose an error from the class L_1, because L_1 contains the minimum weight error. On the other hand, DQMLD will provide a different answer if

$\mathsf{Prob}(L_1) < \mathsf{Prob}(L_2)$, which might occur whenever

$$\tilde{p} > 2^{-\frac{n-k}{c-a}} \,. \tag{4.55}$$

Therefore, in a scenario where the input satisfies the above criterion, the two decoders namely QMLD and DQMLD can potentially disagree on their outputs for some input. For sufficiently low qubit-error rates which violate the above condition, i.e, $p \leq 2^{k-n}$, the presence of degenerate errors does not affect the decoding problem – outputs of QMLD and DQMLD will agree on every input. A similar condition is also formulated in [4].

We will first aim at strengthening the result in the above discussion. Importantly, we will derive a condition which implies that degenerate errors become unimportant for decoding when the *failure rate* of DQMLD is very low. This condition must be preferred to the condition in [4] because it addresses a larger range of qubit-error rates, over which the outputs of the two decoders will match. Remember from (Sec. 4.6.1) that the failure rate 2ϵ of the code is related to the DQMLD promise gap $\Delta = 1 - 4^{-k}\epsilon$. The following lemma, which is an original contribution of the present thesis, shows that for $\epsilon \geq 2^{k-n}$, the outputs of QMLD and DQMLD must match. So, DQMLD is in NP when the promise gap is as large as $1 - 2^{k-n}$.

Lemma 4.7.1 *Let Δ be a number between $1 - 2^{k-n}$ and 1. When the independent $X - Z$ channel is used to calculate the probability of errors and the inputs to DQMLD satisfy a promise gap Δ (cf. 4.40), its output is equal to the output of QMLD.*

Proof: It suffices to demonstrate that, for any syndrome s, $[[n,k]]$ stabilizer code $\mathcal{Q}$ and promise gap Δ, the logical class containing the minimum weight error (output of QMLD), denoted by $L_{\min}$, must satisfy (Eq. 4.45) with $L^\star = L_{\min}$. This will be true whenever $1 - \mathsf{Prob}(L_{\min} \,|\, T_s) \leq \Delta = 1 - 2^{-n-k}$, or in other words,

$$\frac{\mathsf{Prob}(L_{\min}\,,\, T_s)}{\mathsf{Prob}(T_s)} \geq 2^{-n-k}. \tag{4.56}$$

Let $a(s)$ be the minimum weight of an error, when the syndrome s is observed. We will derive lower and upper bounds separately for the numerator and the denominator of (Eq. 4.56). Every element in the logical class of $L_{\min}$ is constructed by taking the product of the minimum weight error with every stabilizer. To find the lowest possible value of the probability of the product, we suppose that each product results in an element whose weight is the sum of $a(s)$ plus the weight of the corresponding stabilizer. Hence we have

$$\mathsf{Prob}(L_{\min}\,,\, T_s) \geq \left[1 - \frac{p}{2}\right]^{2n} \sum_{i=0}^{n} B_i \tilde{p}^{i+a(s)} \,, \tag{4.57}$$

with $\tilde{p} = p/(2-p)$. $B_i = |\{g \in \mathcal{S} : \mathsf{wt}_2(g) = i\}|$ is the weight enumerator coefficient of $\mathcal{S}$.

Similarly, all the errors whose syndrome is s, can be constructed by starting with the error of weight $a(s)$ and taking its product with every element of the normalizer. For an upper bound to the probability of this product, let us suppose that each product has a weight equal to the difference between $a(s)$ and the weight of the corresponding stabilizer. As the weight of the product cannot be lower than $a(s)$ itself, this gives

$$\mathsf{Prob}(T_s) \leq \left[1 - \frac{p}{2}\right]^{2n} \left[\sum_{i=0}^{2a(s)} B_i^{\perp} \tilde{p}^{a(s)} + \sum_{i=2a(s)+1}^{n} B_i^{\perp} \tilde{p}^{i-a(s)}\right] \tag{4.58}$$

where $B_i^{\perp} = |\{g \in \mathcal{N}(\mathcal{S}) : \mathsf{wt}_2(g) = i\}|$ is a weight enumerator coefficient of $\mathcal{N}(\mathcal{S})$.

The ratio of expressions in (Eqs. 4.57, 4.58), upper bounds the term in (Eq. 4.56) by

$$\frac{\mathsf{Prob}(L_{\mathsf{min}}\,,\, T_s)}{\mathsf{Prob}(T_s)} \geq \frac{1}{\sum_{i=0}^{2a(s)} B_i^{\perp} + \tilde{p} \sum_{i=2a(s)+1}^{n} B_i^{\perp}} \geq \frac{1}{\sum_{i=0}^{n} B_i^{\perp}} = 2^{-n-k}\,. \tag{4.59}$$

□

In the light of these observations, one might imagine that in general, DQMLD can at most offer a marginal improvement over QMLD, i.e. for a given code and a noise rate, the decoding failure probability of QMLD is upper bounded by a function of the decoding failure probability of DQMLD. However, there is strong evidence that this is not the case. The following discussion is intended for experts in quantum error correction. The reader might consider skipping all of the content until (Fig. 4.1).

Monte Carlo simulations on the Kitaev's topological code performed in [89, 90, 91] have shown that QMLD and DQMLD achieve different error thresholds. Thus, for noise rates falling between these two thresholds, the failure probability of DQMLD tends to 0 as the number of qubits increases, while the failure probability of QMLD tends to $1 - 4^{-k}$ (the failure probability of a random guess), so the performances of both decoders can be significantly different.

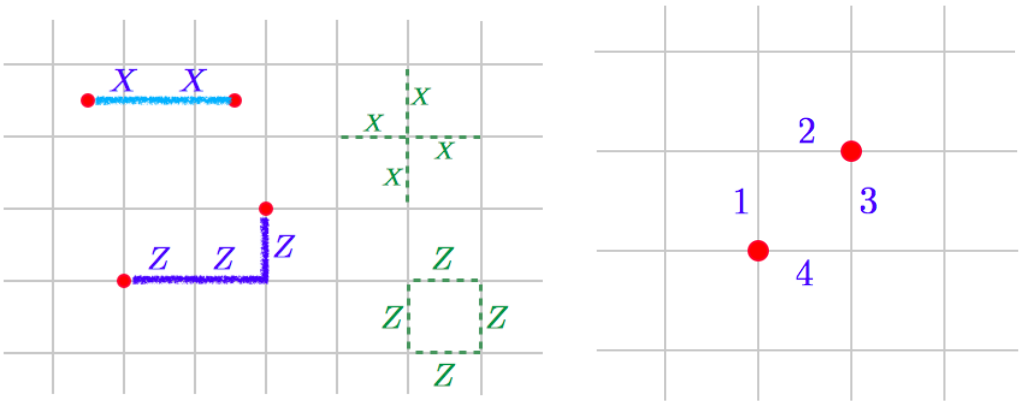
Degeneracy can also severely impact the performances of some decoding algorithms which are neither equivalent to the QMLD strategy nor equivalent to the DQMLD strategy, these are called *marginal decoders* or *qubit-wise decoders* in the sense that they select a $n-$qubit error by independently optimizing the probability of each single qubit Pauli error, in the $n-$fold tensor product. But in the presence of degeneracy, we can have a probability distribution over logical classes sharply peaked over a single class – ensuring the success of

DQMLD – but yet have a very broad marginal distribution over individual qubits – leading to a failure of a marginal decoder. Due to the presence of degenerate errors, the ability to correct errors with a good success probability does not imply that the a posteriori marginal error probability over individual qubits is sharply peaked, in contrast to the classical setting, cf. (Sec. 3.6). This is observed, in particular, for quantum low density parity check (LDPC) codes, whose constructions can be found in [92, 93, 94, 95, 96]. These quantum LDPC codes have the property of admitting a set of stabilizer generators, all of whose weights are bounded by a constant. As a consequence, the weights of various degenerate errors (cf. Sec. 4.4.2) will differ at most by a small constant, irrespective of the number of qubits in the code. Consequently, each logical class will contain many errors of roughly similar probabilities. Thus, we expect the degeneracy in errors to play an important role in the decoding strategy. Based on the discussions in [35], conventional decoding algorithms for LDPC codes (called *belief propagation decoders* in [97]) are *marginal decoders*. So the existence of an optimal decoder for quantum LDPC codes, that works analogous to the classical belief propagation, remains an outstanding open question.

This situation is best illustrated with Kitaev's toric code, cf. [98]. The Toric code on a regular square lattice with periodic boundary conditions, in (Fig. 4.2), has two type of stabilizer generators, shown in green, centred around each face and each vertex of the lattice. In this code, errors correspond to physical chains and the weight of an error is simply equal to the length of the corresponding chain. Chains of Z errors on the lattice generate vertex syndromes at their endpoints, while strings of X errors on the dual lattice generate face syndromes at their endpoints. The different logical classes of errors correspond to the homology classes of chains on the lattice (loops on a torus). The stabilizers are simply all closed chains in the trivial homology class. Chains in the same homology class represent equivalent errors. In (Fig. 4.1b) the qubits are labelled $1-4$ for easy reference and a particular syndrome is shown. There are many errors consistent with this syndrome and amongst them, the lowest weight errors are Z_1Z_2 and Z_3Z_4. By symmetry,

$$\mathsf{Prob}(Z_1Z_2) = \mathsf{Prob}(Z_3Z_4) = \max_{E \in \overline{\mathcal{G}}_n} \mathsf{Prob}(E|s) \ . \tag{4.60}$$

Applying either *one* of the errors, Z_1Z_2 or Z_3Z_4, on the codeword serves as a valid correction since the errors correspond to chains that lie in the same homology class as the error. Errors that correspond to chains from a different homology class have a probability that decreases exponentially with the lattice size, so DQMLD will rarely fail. On the other hand, marginally, qubits 1, 2, 3, and 4 all have identical probabilities, so a marginal decoder will always fail – it will either correct with $Z_1Z_2Z_3Z_4$ or with $\mathbb{1}$, none of which are acceptable corrections.



(a) Kitaev's Toric code on a square lattice **(b)** Syndrome configuration

Figure 4.1

Degenerate errors become important for decoding when there are many low weight stabilizers. Stabilizer codes which contain stabilizers whose weight is less than the distance of the code, are sometimes characterized are *degenerate stabilizer codes*, cf. [32]. It turns out that in order to achieve the true capacity of certain quantum channels (as opposed to the single-shot capacity), it is necessary to encode the message in a degenerate stabilizer code, cf. [88]. It can be shown that for the case of the depolarizing channel, there is a small window of qubit noise rates for which the maximum rate achieved with a *non-degenerate stabilizer code* is strictly lesser than the maximum rate achieved with a degenerate code, cf. [99, 100]. So, one must necessarily use a degenerate stabilizer code to achieve reliable quantum communication over these channels. However, we do not know if the decoding procedure for these codes must necessarily be DQMLD. In particular, the example in [88] uses a generalization of Shor's code (which is a degenerate stabilizer code), yet QMLD and DQMLD always yield the same output. We know of only a few examples of stabilizer codes for which DQMLD can be solved almost efficiently, namely in the case of concatenated codes, cf. [101], convolutional codes, cf. [34], and Bacon-Shor codes, cf. [102]. There exist heuristic methods to take degeneracy into account in topological codes, cf. [33] and for turbo codes, cf. [103].

Finally, we like to remark that though QMLD and DQMLD are stated as decision problems in [2, 4, 5], the problem of practical interest in these cases is clearly the determination a Pauli error that maximizes the probabilities in (Defs. 4.5.2 and 4.6.1) respectively. To enable a decision problem formulation, the authors in [2, 4] have formulated equivalent versions for QMLD and DQMLD to take as input, a constant c. Subsequently, DQMLD would be made to decide the existence of a logical error whose probability (cf. Eq. 4.46) is at least c. This appears to be achieving more than what is really expected from DQMLD (or QMLD) in practice. In

particular, by varying this constant c, along with the input in (Def. 4.6.1), one could use the DQMLD algorithm to not only learn the optimal correction but also the probability of its logical class, in a straightforward manner, whereas, the latter is not necessary to perform error correction. Thus the decision version requires a more powerful algorithm than what is required for DQMLD. This is why we formulated DQMLD as a function problem that does not explicitly reveal the probability of the equivalence class of the optimal correction, and therefore we consider it to be closer to the real world decoding problem.

4.8 Explicit example with Shor's 9−qubit code

Before closing the chapter, we will illustrate all the concepts we have discussed so far in quantum error correction, with the example of the Shor's 9−qubit code. This is the first error correcting code to be invented and named after its inventor, Peter Shor, cf. [104].

Shor's code encodes a 1−qubit message using 9 qubits. It is a $[[9, 1, 3]]$ code which is designed to correct any single qubit error. Due to the simplicity of the stabilizer formalism as well as for the purpose of error correction, we need not specify the codewords explicitly (the codewords are listed in [23]). Nevertheless, it is useful to understand the structure of the codewords of the Shor's code, intuitively, as follows.

A 1−qubit (message) state $\alpha|0\rangle + \beta|1\rangle$ must be protected against bit flip as well as phase flip errors. Let us start with phase flip errors. A phase flip error achieves the transformation $|0\rangle \to |0\rangle$ and $|1\rangle \to -|1\rangle$. Subsequently, it performs a bit flip in the *conjugate basis* or eigenbasis of X, i.e, a phase flip achieves $|+\rangle \leftrightarrow |-\rangle$, where $|\pm\rangle = 1\Big/\sqrt{2}\,(|0\rangle \pm |1\rangle)$. In order to protect a qubit against phase flip errors, we adopt the same technique as in the classical repetition code, the two orthogonal states of the message are encoded into the codewords

$$\begin{aligned} |\bar{0}\rangle &= \frac{1}{2\sqrt{2}}\,(|0\rangle + |1\rangle)\,(|0\rangle + |1\rangle)\,(|0\rangle + |1\rangle) \\ |\bar{1}\rangle &= \frac{1}{2\sqrt{2}}\,(|0\rangle - |1\rangle)\,(|0\rangle - |1\rangle)\,(|0\rangle - |1\rangle) \end{aligned} \tag{4.61}$$

respectively. So, if a phase flip occurs on one of the qubits in the codeword, a simple majority vote can be employed to map the received state to the correct codeword.

Second, let us consider bit flip errors which are caused by X on some qubit. A single bit flip error leaves qubits in each of the three blocks, invariant up to a phase. In order to

prevent this, we will encode each of the qubits within the blocks into a bit flip repetition code, so a code that corrects both bit flip as well as phase flip errors, is given by

$$|\bar{\tilde{0}}\rangle = \frac{1}{2\sqrt{2}} \left(|\tilde{0}\rangle + |\tilde{1}\rangle\right) \left(|\tilde{0}\rangle + |\tilde{1}\rangle\right) \left(|\tilde{0}\rangle + |\tilde{1}\rangle\right)$$
$$|\bar{\tilde{1}}\rangle = \frac{1}{2\sqrt{2}} \left(|\tilde{0}\rangle - |\tilde{1}\rangle\right) \left(|\tilde{0}\rangle - |\tilde{1}\rangle\right) \left(|\tilde{0}\rangle - |\tilde{1}\rangle\right) \tag{4.62}$$

where the encoded qubits within the blocks are

$$|\tilde{0}\rangle = |000\rangle \ , \ |\tilde{1}\rangle = |111\rangle \tag{4.63}$$

Now any bit flip error can be corrected using a majority vote (locally) within each block, i.e, in the above encoding in (Eq. 4.63) and any phase flip error can be corrected by observing the entire codeword, i.e, in the encoding in (Eq. 4.61).

We say that Shor's $[[9,1,3]]$ code is a *concatenation* of $[[3,1,1]]$ bit flip repetition code with a $[[3,1,1]]$ phase flip repetition code. Generalizations of the Shor's code can be obtained by concatenating larger repetition codes, cf. [102, 31]. The stabilizer generators for the Shor's $[[9,1,3]]$ code can be derived as a consequence of the above concatenation. The $Z-$type stabilizer generators can be derived straightforwardly – each of the three blocks in (Eq. 4.62) comprise of codewords of the $[[3,1,1]]$ bit flip repetition code, which are fixed by a product of $Z-$type operators on any two qubits in that block. The $X-$type stabilizer generators can be derived by observing that the codewords in (Eq. 4.62) are fixed by product of $X-$type operators on any pair of blocks. Therefore, the stabilizer generators of the Shor's code are

$$Z_1Z_2 \ , \ Z_2Z_3 \ , \ Z_4Z_5 \ , \ Z_5Z_6 \ , \ Z_7Z_8 \ , \ Z_8Z_9 \ , \ X_1X_2X_3X_4X_5X_6 \ , \ X_1X_2X_3X_7X_8X_9 \ . \tag{4.64}$$

It turns out that the generators of the logical errors ($\mathcal{L}$, cf. Eq. 4.24) are

$$Z_1Z_4Z_7 \ , \ X_1X_2X_3 \tag{4.65}$$

and the generators for the pure errors ($\mathcal{T}$, cf. Eq. 4.27) are

$$X_2X_3 \ , \ X_3 \ , \ X_5X_6 \ , \ X_6 \ , \ X_8X_9 \ , \ X_9 \ , \ Z_4 \ , \ Z_7 \ . \tag{4.66}$$

The above choice of generating sets satisfies the commutation relations prescribed in (Lemma. 4.4.1). The distance of can also be found by noticing that the smallest weight error in $\mathcal{N}(\mathcal{S})\backslash\mathcal{S}$ is $X_1X_2X_3$, which has weight 3, cf. (Eq. 4.34).

The structure of the stabilizer generators in (Eq. 4.64) enables an easy representation

using a lattice, in the following manner.

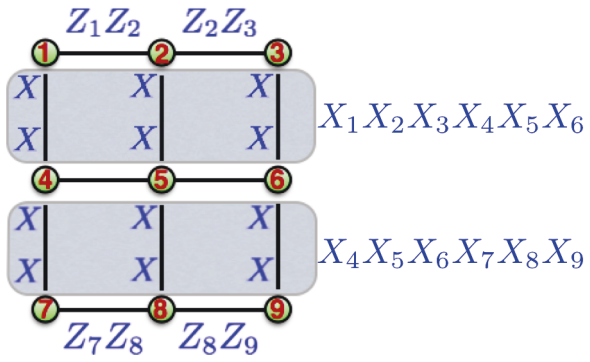


Figure 4.2 Each vertex represents a qubit (numbered on the figure) and each horizontal link between a pair of vertices denotes a $Z-$type stabilizer generator. Additionally, assigning an $X-$type operator on each vertex, a pair of rows denotes a $X-$type stabilizer generator. This figure can therefore be used to recall the stabilizer generators of the Shor's code. Furthermore it facilitates an easy generalization of the Shor's code by choosing lattices of different sizes, cf. (App. A.1) and [102].

From the discussion in (Sec. 4.2.1) on symplectic representation of Pauli errors, the stabilizer generator set can be expressed as the following 8×18 matrix, denoted by M, whose rows contain the symplectic representation of each stabilizer generator.

$$\mathsf{M} = \left[\begin{array}{ccccccccc|ccccccccc} 1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1 \end{array}\right] \tag{4.67}$$

Let us illustrate the QMLD decoding strategy with the Shor's $[[9,1,3]]$. Suppose a codeword is transmitted across a channel and undergoes a single qubit error. The received $n-$qubit state will in general differ from the transmitted codeword. However, we can carry

out a series of measurements, of each stabilizer generator on the received $n-$qubit state, to obtain the error syndrome, cf. (Def. 4.5.1). Suppose the observed error syndrome is

$$s = 10000000\ . \tag{4.68}$$

that corresponds to some error. The above error syndrome will completely specifies the support of the error in $\mathcal{T}$, denoted by T_s, as (cf. Eq. 4.37)

$$T_s = X_2X_3\ . \tag{4.69}$$

Next, task of QMLD (cf. Def. 4.5.2) is to select the error that has the maximum probability, amongst all errors that have the syndrome 10000000, i.e, amongst all errors in the coset $T_s \cdot \mathcal{N}(\mathcal{S})$. There are $2^{n+k} = 2^{10}$ errors with the syndrome 10000000, a few of which are

$$X_2X_3\ ,\ Y_2Y_3\ ,\ Z_1X_2Y_3\ ,\ Z_1Y_2X_3\ ,\ X_2Y_3Z_4Z_7\ ,\ X_2Y_3Z_4Z_9\ ,\ X_1, Y_1Z_3\ ,\ Y_1Z_2\ ,\\ X_1Z_7Z_9\ ,\ Y_1Z_4Z_7\ ,\ Y_1Z_4Z_8. \tag{4.70}$$

The above list is simply formed by multiplying T_s with every element of $\mathcal{N}(\mathcal{S})$ and ignoring the overall phase in the resulting product. Each error can be assigned a probability using the independent $X-Z$ channel (cf. Def. 4.3.3). Now, the QMLD problem is simply to choose the error with the highest probability (or with the lowest symplectic-weight). In the current example, X_1 has the lowest symplectic weight and it is expressed as

$$X_1 = X_2X_3 \cdot X_1X_2X_3 \cdot \mathbb{1}\mathbb{1} \tag{4.71}$$

where $X_2X_3 \in \mathcal{T}\ ,\ X_1X_2X_3 \in \mathcal{L}\ ,\ \mathbb{1}\mathbb{1} \in \mathcal{S}$ (cf. Eqs. 4.69, 4.65, 4.64), in the operator basis (cf. Eq. 4.31). Summarizing, when the input to QMLD (cf. Def. 4.5.2) is the generators of the Shor's code and the syndrome 10000000, its output is X_1.

Let us now turn to the optimal decoding strategy, DQMLD, cf. (Def. 4.6.1). For the syndrome $s = 10000000$, DQMLD requires the classification of errors into logical classes, i.e, the cosets $T_s \cdot L_i \cdot \mathcal{S}$, for each logical error $L_i \in \mathcal{L}$ where $T_s = X_2X_3$. There are 4 logical classes in total, each of which contains 2^8 errors. A few of the errors in these cosets are

$$[L_1]\ :\ X_2X_3\ ,\ Y_2Y_3\ ,\ Z_1X_2Y_3\ ,\ Z_1Y_2X_3\ ,\ \ldots\ldots \tag{4.72}$$

$$[L_2]\ :\ X_2Y_3Z_4Z_7\ ,\ X_2Y_3Z_4Z_9\ ,\ X_2Y_3Z_4Z_8\ ,\ \ldots\ldots \tag{4.73}$$

$$[L_3]\ :\ X_1\ ,\ Y_1Z_3\ ,\ Y_1Z_2\ ,\ X_1Z_7Z_9\ ,\ X_1Z_7Z_8\ \ldots\ldots \tag{4.74}$$

$$[L_4]\ :\ Y_1Z_4Z_7\ ,\ Y_1Z_4Z_9\ ,\ Y_1Z_4Z_8\ \ldots\ldots \tag{4.75}$$

Since there are exponentially many errors in the coset, computing their probabilities is not easy. We will compute their probabilities using the independent $X-Z$ channel, cf. (Def. 4.3.3). Alternatively, one can also consider a depolarizing channel, cf. (Def. 4.3.2). One way to compute the coset probabilities this is to calculate the weight enumerator polynomial of these cosets (cf. Eq. 4.47) which in this case, for the above cosets, is given as follows.

$$\mathsf{Prob}(L_1, s) = (1-p)^9 \left[9q^8 + 84q^6 + 126q^4 + 36q^2 + 1\right] \tag{4.76}$$

$$\mathsf{Prob}(L_2, s) = (1-p)^9 \left[q^9 + 36q^7 + 126q^5 + 84q^3 + 9q\right] \tag{4.77}$$

$$\mathsf{Prob}(L_3, s) = (1-p)^9 \left[q^9 + 18q^7 + 84q^5 + 126q^3 + 27q\right] \tag{4.78}$$

$$\mathsf{Prob}(L_4, s) = (1-p)^9 \left[3q^8 + 46q^6 + 24q^5 + 60q^4 + 80q^3 + 18q^2 + 24q + 1\right] , \tag{4.79}$$

where $q = p/2(1-p)$. The above polynomials have been computed by explicitly counting over the elements in each logical class. Remember that irrespective of the above classification, the output of an algorithm that solves QMLD will always be X_1. Whereas the output of the *optimal* decoder, DQMLD, will be an error in the logical class $[L_3]$, only until it $\mathsf{Prob}(L_3, s)$ in (Eq. 4.78) evaluates to the highest value, compared to the polynomials corresponding to the other logical classes. If any other polynomial evaluates to a higher value than $\mathsf{Prob}(L_3, s)$ the outputs of QMLD and DQMLD will indeed differ.

We have illustrated how DQMLD and QMLD can be performed on a stabilizer code. However, for the special case of the Shor's $9-$qubit code (and also for the generalized Shor code) a close analysis of the polynomials constituting the probabilities of the logical classes in (Eqs. 4.76 to 4.79) will suggest that the outputs of the two decoders QMLD and DQMLD will always coincide. However, some codes have been known for which similar calculations like above will result in the observation that the outputs of QMLD and DQMLD differ. For those codes, the explicit computation of the logical class probabilities is far more complex and tedious (in many cases, a closed form expression for the probabilities is not even known), and therefore we will not mention them here.

4.9 Summary

In this chapter, we have studied basic notions about the stabilizer formalism for quantum error correction and the different error models one can use to study the decoding problem of quantum error correction on the Pauli channel. There were two decoding strategies, QMLD that was a *suboptimal* strategy in the sense that it does not always lead to an inference of the transmitted codeword that has the maximum probability and DQMLD which was the

optimal decoding strategy. The optimal decoding strategy accounts for a sharp non-classical feature in the errors on a quantum channel, i.e, the presence of degenerate errors. The optimal decoding strategy, DQMLD, returns the error that lies in the most probable logical class. So far, we have seen that degenerate errors can change the decoding strategy in a drastic way. If one were to ignore the presence of degenerate errors, then the decoding strategy becomes suboptimal and the decoding problem closely resembles the classical decoding problem. Furthermore, we also saw the suboptimal decoding strategy, QMLD, as well as the classical decoding strategy, MWD, are both NP-complete problems.

We have the necessary background to answer the question that is at the core of this thesis. Is the optimal decoding problem, DQMLD, harder than the suboptimal decoding problem, QMLD, or the classical decoding problem, MWD ? We will address this question in the next chapter.

Chapter 5

Complexity of degenerate decoding

In the previous chapter, we encountered two decoding strategies for stabilizer codes, namely, *quantum maximum likelihood decoding* (QMLD) and degenerate quantum maximum likelihood decoding (DQMLD), the former being a suboptimal strategy. We also studied some reasons and examples of cases where QMLD fails to provide the codeword that has the maximum probability of being transmitted. Furthermore, QMLD was as hard as the decoding problem for classical codes, MWD, cf. (Thm. 4.5.1). On the other hand, there were some extreme cases of inputs to QMLD and DQMLD where their corresponding outputs always agree, cf. (Lemma. 4.7.1). By large, the two decoding strategies were indeed fundamentally different.

The interesting question that remains is the complexity of the optimal decoding problem, DQMLD. In this chapter, we will show that indeed DQMLD is in fact much harder than its suboptimal counterpart, QMLD. This chapter is also the main contribution of this thesis and the following content can be found in [3]. This chapter is structured as follows. In (Sec. 5.1), we will state our main result and discuss a brief overview of the proof. Following this, the detailed proof of the main result is spread over (Secs. 5.1.2 to 5.1.4).

5.1 Complexity of optimal (degenerate) decoding

From our discussion in the previous chapter, we realize that, for a given syndrome s, computing the probability of a coset associated to s and L for every logical error $L \in \mathcal{L}$, given by (Eq. 4.47) and subsequently optimizing over all their values would solve DQMLD. However, a $[[n,k]]$ stabilizer code has $|\mathcal{L}| = 4^k$, implying that even if the weight enumerators can be computed efficiently, a polynomial-time optimization cannot be performed over the different coset probabilities, unless $k = \mathcal{O}(\log_2 n)$. We will address the computation complexity of DQMLD for $[[n,k]]$ stabilizer codes where $k = 1$. Furthermore, we believe DQMLD is at least as hard otherwise, i.e for $k \in \Omega(\log(n))$.

We now state our main result, which establishes that DQMLD is in fact much harder than what previous results anticipated, cf. [2, 4, 5]. Moreover, this being a counting problem of a corresponding NP-Hard decision problem, it is suspected to be much harder than QMLD, c.f (Sec. 2.3). We state our main result as follows.

Theorem 5.1.1 *Computational complexity of DQMLD*

Let $\{S_1, \ldots, S_{n-k}\}$ denote the stabilizer generators of a $[[n,1]]$ quantum code, λ be some number satisfying $\lambda = \Omega(\mathit{polylog}(n))$, s be any $(n-k)-$bit sequence and Δ be a real number satisfying

$$\Delta \leq \frac{2}{2+n^{\lambda}} \,. \tag{5.1}$$

Then, the computational complexity of DQMLD on the independent $X-Z$ channel, defined according to (Def. 4.6.1) whose input is given by the stabilizer generator set $\{S_1, \ldots, S_{n-k}\}$, syndrome s and the promise gap Δ according to (Eq. 5.1) is summarized by the following statement.

$$\mathit{DQMLD} \in \#\mathit{P\text{-}complete} \,. \tag{5.2}$$

In (Def. 3.6.1) we encountered a #P-complete problem pertaining to a linear code which was #P-complete, cf. (Lemma. 3.6.1). In the following sections, we will show that for a linear code $\mathcal{C}$ and $\lambda \in [0,n]$, the problem of computing $\mathsf{WE}_i(\mathcal{C})$ for $i = 0,1,2,\ldots,\lambda$ is polynomial time Turing reducible to DQMLD on an independent $X-Z$ channel, for $\Delta \leq 2[2+n^{\lambda}]^{-1}$.

In the case of a general memoryless Pauli channel, it is not always possible to express the probability of a logical class as a weight enumerator sum (Eq. 4.47) with polynomially many terms in the sum. Hence, in such cases, the containment of DQMLD in #P is not known and we can only claim that it is #P-hard. However, whenever one can express $\mathsf{Prob}(L|s)$ as a sum

with polynomially many terms, each of which is a #P function, then DQMLD can be put into #P. For the independent $X-Z$ channel, the containment in #P is straightforward.

Lemma 5.1.1 *DQMLD in (Def. 4.6.1) on a $[[n,k]]$ stabilizer code with $k=\Omega(\log_2 n)$ on an independent $X-Z$ channel in (Eq. 4.20) or a depolarizing channel, is in #P.*

Proof: From the definition of DQMLD, one can subdivide it into two problems. The first step consists of computing the probabilities of all the logical classes, corresponding to the given syndrome, followed next by an optimization over the probabilities, thereby choosing an error from the logical class with the largest probability. Since $k=\Omega(\log_2 n)$, there are at most polynomially many logical classes and hence a naive search for the maximum amongst the set of probabilities of all the logical classes can be achieved in polynomial-time.

It now remains to show that for each logical class, its probability in (Eq. 4.48) for a general noise rate p, is a #P function. This immediately follows from the polynomial form in p, taken by the logical class probabilities in (Eq. 4.47) and the observation that the coefficients of the polynomial are weight enumerator coefficients (cf. Def. 3.6.1) of a suitable linear code, following the mappings discussed between $\overline{\mathcal{G}}_n$ and $\mathbb{Z}_2^{2n}$ in (Sec. 4.2.1).

The promise gap Δ is irrelevant to the containment of DQMLD in #P unless $\Delta = o(2^{-\mathsf{poly}(n)})$. A gap of $2^{-\mathsf{poly}(n)}$ is essential since the probabilities of various logical classes are represented as bit-strings for performing arithmetic and they must differ in the first polynomially many bits. □

5.1.1 Reduction overview

In this section, we present a polynomial time algorithm that accepts as input a classical linear code $\mathcal{C}$ and outputs $\{\mathsf{WE}_i(\mathcal{C})\}_{i=0}^{n}$ as in (Def. 3.6.1) by querying a DQMLD oracle with an independent $X-Z$ noise model and a promise gap Δ which is $1/\mathsf{quasi\text{-}polynomial}(n)$.

The correspondence between symplectic linear codes and stabilizer codes discussed in (Sec. 4.2.1) implies that with an independent $X-Z$ channel (Def. 4.3.3), the probability of a logical class is related to the weight enumerator polynomial of a classical linear code. In particular, with the trivial syndrome $s=\vec{0}$ and at very low noise rate $p \ll 1/2$, the most likely equivalence class is always the trivial one $L=\mathbb{I}$. Hence, in this setting, the probability of the trivial logical class for a quantum code with stabilizers $\mathcal{S}$ can be expressed, similar to

(Eq. 4.47), in terms of a corresponding classical code $\mathcal{C}_{\mathcal{S}}$ as follows.

$$\mathsf{Prob}(\mathbb{I} \,|\, \vec{0}) = \left(1 - \frac{p}{2}\right)^{2n} \sum_{i=0}^{2n} \mathsf{WE}_i(\mathcal{C}_{\mathcal{S}}) \tilde{p}^i \tag{5.3}$$

Since this probability is a polynomial of degree $2n$, determining its value for $2n+1$ distinct values of the physical noise rate p would enable us to determine the weight enumerator of the corresponding classical code. There are two caveats to this approach. First, this approach only works for classical codes $\mathcal{C}_{\mathcal{S}}$ whose generator matrix corresponds to the symplectic representation of a quantum stabilizer code. Second, this approach requires knowledge of the probability of a certain equivalence class, while DQMLD only outputs the equivalence class with the largest probability; it does not reveal the value of the corresponding probability.

The first caveat can easily be circumvented, by constructing a set stabilizer generators, such that the weight enumerators of the stabilizer group and the underlying classical code, are the same. Consequently, all stabilizers have weights between 0 and n and the probability of the trivial logical class can be expressed as (Eq. 5.3), where the range of sum is up to n. To circumvent the second caveat, we need to use the DQMLD oracle to obtain equality constraints on the weight enumerator of $\mathcal{C}$. This is done by varying the physical noise rate p, always keeping the syndrome trivial. As mentioned above, at very low noise rate the optimal logical class is $\mathbb{I}$. Increasing the noise rate p, we will reach a *crossing point* p_1 where the DQMLD output changes from $\mathbb{I}$ to $L^\star \neq \mathbb{I}$. At this point, the promise gap condition is violated, i.e, $|\mathsf{Prob}(\mathbb{I}|s) - \mathsf{Prob}(L^\star|s)| \leq \Delta\mathsf{Prob}(\mathbb{I}|s)$, which can be expressed in terms of weight enumerators as

$$\sum_{i=0}^{n} \mathsf{WE}_i(\mathcal{C}) \tilde{p}_1^i - \sum_{i=0}^{n} B_i \tilde{p}_1^i \leq \Delta \sum_{i=0}^{n} \mathsf{WE}_i(\mathcal{C}) \tilde{p}_1^i \,, \tag{5.4}$$

where B_i are the weight enumerators of an affine code. In the case where $\Delta = 0$, this crossing point provides an equality condition between two polynomials, which is what we are seeking. But since these are polynomials with integer coefficients, knowing the location of a crossing point within a finite accuracy, which translate into a finite promise gap Δ, is enough to determine the exact crossing point, see (Lemma. 5.1.2). As we will show, there exists a fixed range of p that provably contains a unique crossing point, enabling a polynomial-time accurate determination of the crossing point using a binary search procedure.

This gives us one potential equality, but introduces more unknown coefficients B_i. To get additional linear constraints, we modify the code in a very special way, described in

(Sec. 5.1.2). This modification requires adding one *tunable* qubit and one stabilizer generator. By varying the noise rate on the tunable qubit over a range of values, we can change the location of the crossing point, and thus obtain new linear constraints relating $\{\mathsf{WE}_i(\mathcal{C})\}_{i=0}^n$ and the $\{B_i\}_{i=0}^n$. Repeating this procedure $2n+2$ times and making sure that all the linear constraints are linearly independent (Sec. 5.1.4) enable us to determine the weight enumerator coefficients. While the ability to change the noise rate of the tunable qubit gives us more linear constraints, it breaks the requirement that the noise model be the independent $X-Z$ channel with the same strength on all qubits. We will fix this problem in (App. A.1) by showing that the required channel can be simulated by concatenating the code with a Shor code (a generalized version of the code that is presented in Ex. 4.8). In fact, we will use this technique repeatedly in our proof.

5.1.2 Stabilizer code construction

Let G be the $k \times n$ generator matrix (cf. Def. 3.3.3) of an (n,k) linear code $\mathcal{C} \stackrel{\text{def}}{=} \{x \in \mathbb{Z}_2^n : x = y \cdot \mathsf{G},\ y \in \mathbb{Z}_2^k\}$. Denote $\{g_i\}_{i=1,\ldots k}$ the rows of G and let $\{g_i\}_{i=k+1\ldots n}$ be a generating set of the complement of the row space of G, i.e. in such a way that $\{g_i\}_{i=1\ldots n}$ span $\mathbb{Z}_2^n$. Construct a matrix $\tilde{\mathsf{G}}$ with rows$\{g_i\}_{i=1\ldots n}$. This matrix is full rank, and therefore has an inverse H that can be computed efficiently, and obeys $\tilde{\mathsf{G}} \cdot \mathsf{H}^T = \mathbb{I}$. Denote the rows of H by $\{h_i\}_{i=1}^n$.

We define a $[[2n-k-1,1]]$ quantum code with stabilizer generators (like in Def. 4.4.1) and logical operators given by

$$S_i = Z^{g_i} \qquad \text{for } i = 1,\ldots,k \tag{5.5}$$

$$S_{k+i} = Z^{g_{k+i}} \otimes Z_{n+i} \qquad \text{for } i = 1,\ldots,n-k-1 \tag{5.6}$$

$$S_{n-1+i} = X^{h_{k+i}} \otimes X_{n+i} \qquad \text{for } i = 1,\ldots,n-k-1 \tag{5.7}$$

$$\overline{Z} = Z^{g_n} \tag{5.8}$$

$$\overline{X} = X^{h_n}, \tag{5.9}$$

where it is implicitly assumed that operators are padded to the right by identities to be elements of $\mathcal{G}_{n+k-1}$. The validity of the resulting code can be verified if the above generators satisfy the commutation rules in (Lemma. 4.4.1). We check this explicitly here:

1. There are in total $2n-k-1$ qubits.
2. The $2n-k-2$ stabilizer generators are independent. This follows from the linear independence of the h_i and the linear independence of the g_i, together with the fact that X-type operators are linearly independent of Z type generators.

3. The stabilizer generators mutually commute. This is trivial among the first $n-1$ generators as they contain only Z operators and similarly among the last $n-k-1$ last generators. Between these two sets, the commutation follows from the fact that $h_i \cdot g_j = 0$ except when $i = j$, in which case the presence of additional X and Z on the $n+i$th qubit ensures commutation.

4. The logical operators commute with the stabilizer generators. This follows from the fact that $h_i \cdot g_j = 0$ for $i \neq j$, and the fact that X-type operators commute among themselves and similarly for Z-type operators. This along with the previous property, will satisfy (Lemma. 4.4.1).

As discussed in the material following (Eq. 4.44), the probability of the trivial logical class $\mathbb{I}$ given a trivial syndrome $s = \vec{0}$ is simply the sum of the probabilities of all stabilizer group elements. Suppose now that the last $n-k-1$ qubits are error-free, while the other qubits are subject to an independent $X-Z$ channel (in Def. 4.3.3). Then, the probability of an element of $\mathcal{S}$ is zero if it contains a generator S_i from the above list with $i > k$. Otherwise, this element of $\mathcal{S}$ can be written as $S = Z^x \otimes \mathbb{I}$ for some $x \in \mathbb{Z}_2^n$ and its probability is $(p/2)^{\mathsf{wt}(x)}(1-p/2)^{2n-\mathsf{wt}(x)}$. We conclude that the probability of the trivial logical class is given by (Eq. 5.3) with $\mathcal{C}$ the classical code defined by the generating matrix G (exactly similar to Eq. 3.20).

Constraints on the weight enumerator polynomial will be obtained by finding crossing points where $\mathsf{Prob}(\mathbb{I}|\vec{0}) \approx \mathsf{Prob}(L|\vec{0})$ with $L \neq \mathbb{I}$. For technical reasons, we would like to be able to choose which L will be the one realizing the crossing. This is because we want to force the crossing to happen with the same L every time. To do this, we will modify the stabilizer by adding an extra qubit and making the transformations

$$S_i \to S_i \otimes \mathbb{I} \tag{5.10}$$

$$\overline{Z} \to \overline{Z} \otimes \mathbb{I} \tag{5.11}$$

$$\overline{X} \to \overline{X} \otimes X \tag{5.12}$$

to the stabilizers and logical operators, and adding the following stabilizer generator

$$S_{2n-k} = \overline{Z} \otimes Z. \tag{5.13}$$

This defines an $[[2n-k, 1]]$ stabilizer code, and its validity can easily be verified given the commutation relations worked out above. Moreover, if we assume that the added $(2n-k)$th qubit is also error-free, then the probability of the trivial logical class $\mathbb{I}$ given a trivial

syndrome $s = \vec{0}$ is unchanged, and moreover the only other logical class with non-zero probability is the one associated to $\overline{Z}$, i.e. $\mathsf{Prob}(\overline{X}|\vec{0}) = \mathsf{Prob}(\overline{Y}|\vec{0}) = 0$.

We need to perform one last modification to the code in order to be able to tune the crossing point, and hence obtain linearly independent equalities between $\mathsf{Prob}(\mathbb{I}|\vec{0})$ and $\mathsf{Prob}(\overline{Z}|\vec{0})$. This transformation is quite similar to the previous one, and given by

$$S_i \to S_i \otimes \mathbb{I} \tag{5.14}$$

$$\overline{Z} \to \overline{Z} \otimes Z \tag{5.15}$$

$$\overline{X} \to \overline{X} \otimes \mathbb{I} \tag{5.16}$$

and adding the following stabilizer generator

$$S_{2n-k+1} = \overline{X} \otimes X. \tag{5.17}$$

For this last qubit, we will assume a noise model where $p_X = p_Y = 0$, $p_Z = q$, and $p_{\mathbb{I}} = 1 - q$ with q being a tunable parameter. With this last choice, the only two non-zero probabilities of logical class conditioned on the trivial syndrome are given by

$$\mathsf{Prob}(\mathbb{I}|\vec{0}) = \frac{1}{\mathcal{Z}}(1-q)\sum_{i=0}^{n} \mathsf{WE}_i(\mathcal{C})\tilde{p}_1^i \tag{5.18}$$

$$\mathsf{Prob}(\overline{Z}|\vec{0}) = \frac{1}{\mathcal{Z}}q\sum_{i=0}^{n} B_i\tilde{p}_1^i, \tag{5.19}$$

where $\tilde{p} = p/(2-p)$ as above, $\mathcal{Z}$ is a suitable normalization factor, and B_i are the weight enumerators of the affine code associated to the $\overline{Z}$ logical class

$$B_i = \big|\{x \in \mathcal{C} + g_n : |x| = i\}\big|. \tag{5.20}$$

A crossing point is observed when

$$v\sum_{i=0}^{n} \mathsf{WE}_i(\mathcal{C})\left(\frac{\tilde{p}_1}{2}\right)^i - \sum_{i=0}^{n} B_i\left(\frac{\tilde{p}_1}{2}\right)^i \leq \Delta v\sum_{i=0}^{n} \mathsf{WE}_i(\mathcal{C})\left(\frac{\tilde{p}_1}{2}\right)^i \tag{5.21}$$

where $v = (1-q)/q$ is a tunable parameter over the positive reals. Changing the value of v will change the crossing point between these two logical classes, and provide linear constraints between two degree n polynomials. If we can identify $2n+1$ such crossing points, it would provide enough information to retrieve the two polynomials, and hence solve the weight enumerator problem.

5.1.3 Finding crossing points

At this point, we have a deterministic procedure that, given any classical linear code $\mathcal{C}$, can be used to generate linear constrains on its weight enumerator coefficients. Clearly, the overhead in the runtime of this procedure is the time required to spot a crossing point. A crossing point can potentially be observed at a physical noise rate p anywhere between 0 and 1. One obvious indication of a crossing point is the switch in the output of the DQMLD oracle as we move p across the crossing point. However if we move p across two crossing points, we will not notice any net switch in the outputs of the DQMLD oracle. For this reason, we now want to restrict the values of p to a range where we can prove that there is at most one crossing point. This will be possible by restricting the tunable parameter v to a small interval near 0.

Lemma 5.1.2 *Given the stabilizer code defined above with a tunable parameter $v \leq (1-\Delta)\, n^{-d}$, where $1 \leq d \leq |g_n| \leq n$ is the distance between $\mathcal{C}$ and $\mathcal{C} + g_n$, then there exists exactly one crossing point between the pair of logical classes $\mathbb{I}$ and $\overline{Z}$ in the interval $0 \leq p \leq 1/n$.*

Proof : First, note that for sufficiently small p and for any value of v , DQMLD will output the identity class, simply because its probability is a polynomial in p with a constant term, and the probabilities of all other logical classes contain no constant term. Furthermore, we claim that the probability of the trivial logical class is a strictly decreasing function of the noise rate p, when $0 \leq p \leq 1/n$. To justify this, it suffices to show that the derivative of $\mathsf{Prob}(\mathbb{I}|\vec{0})$ is strictly negative in the prescribed range of noise rates. Recalling the weight enumerator polynomial for $\mathsf{Prob}(\mathbb{I}|\vec{0})$ and computing the derivative w.r.t p, we find:

$$\frac{\partial \mathsf{Prob}(\mathbb{I}|\vec{0})}{\partial p} = \frac{1}{2}\sum_{i=0}^{n} \mathsf{WE}_i(\mathcal{C}) \left(\frac{p}{2}\right)^i \left(1-\frac{p}{2}\right)^{2n-i} \left[\frac{i}{p/2} - \frac{2n-i}{1-p/2}\right] \tag{5.22}$$

$$= \sum_{i=1}^{n} \mathsf{WE}_i(\mathcal{C}) \left(\frac{p}{2}\right)^i \left(1-\frac{p}{2}\right)^{2n-i} \left[\frac{i-np}{p(1-p/2)}\right] - n\left(1-\frac{p}{2}\right)^{2n-1} \tag{5.23}$$

where the last term is added because we have changed the range of summation. It only remains to show that the expression above is strictly negative. Clearly, this cannot be true for all $p \in [0,1]$. However, when $p \leq 1/n$, we know that the first term in (Eq. 5.23) is non-negative. Indeed, all of its terms are positive by definition, except the one in square bracket which is non-negative when $p \leq 1/n$. We claim now that when $p \leq 1/n$, the second term of (Eq. 5.23) is negative and greater in norm than the first one, so the entire

expression is strictly negative. This can be observed by the following inequality:

$$\begin{aligned}\sum_{i=1}^{n} \mathsf{WE}_i(\mathcal{C}) \left(\frac{p}{2}\right)^i \left(1-\frac{p}{2}\right)^{2n-i} \left[\frac{i-np}{p(1-p/2)}\right] &< \sum_{i=1}^{2n} \binom{2n}{i} \left(\frac{p}{2}\right)^i \left(1-\frac{p}{2}\right)^{2n-i} \left[\frac{i-np}{p(1-p/2)}\right] \\ &= \sum_{i=0}^{2n} \binom{2n}{i} \left(\frac{p}{2}\right)^i \left(1-\frac{p}{2}\right)^{2n-i} \left[\frac{i-np}{p(1-p/2)}\right] + n\left(1-\frac{p}{2}\right)^{2n-1} \\ &= n\left(1-\frac{p}{2}\right)^{2n-1}.\end{aligned}$$

Hence, we see that indeed the probability of the trivial logical class is strictly decreasing when $0 \leq p \leq 1/n$. Since there are only two logical classes in our setting, it is clear that the probability associated to the other class is increasing in that interval.

We have identified an interval where the probabilities are monotonic, and what remains to be shown is that there is indeed a crossing point inside this interval when the parameter v is chosen carefully. Intuitively, we can see that decreasing the value of v (and hence of the trivial logical class) will decrease the value of p where the first crossing point occurs. We will now set an upper bound $p_{\max}$ on the value of p where the first crossing point occurs. The first crossing point will occur at the latest when $v\mathsf{Prob}(\mathbb{I}|\vec{0}) - \mathsf{Prob}(\overline{Z}|\vec{0}) = \Delta\mathsf{Prob}(\mathbb{I}|\vec{0})$, so equivalently $v(1-\Delta) = \mathsf{Prob}(\overline{Z}|\vec{0}) \Big/ \mathsf{Prob}(\mathbb{I}|\vec{0})$. On the other hand, the ratio $\mathsf{Prob}(\overline{Z}|\vec{0}) \Big/ \mathsf{Prob}(\mathbb{I}|\vec{0})$ is lower-bounded by $[p/(2-p)]^d$ since each word in $x \in \mathcal{C}$ is mapped onto a word $y = x + g_n$ of weight at most $|x| + |g_n|$ in $\mathcal{C} + g_n$. Hence, the first crossing occurs for a value of p lower or equal to the point where

$$v = (1-\Delta)^{-1}\left[\frac{p}{2-p}\right]^d, \text{ or equivalently } p_{\max} = \frac{2}{1+(1-\Delta)^{\frac{1}{d}}\, v^{-\frac{1}{d}}}. \tag{5.24}$$

By choosing

$$p_{\max} \leq \frac{1}{n}, \text{ or equivalently } v \leq (1-\Delta)^{-1}\, n^{-d}, \tag{5.25}$$

we are sure that the first crossing point occurs in the monotonic region, and hence that the interval $0 \leq p \leq 1/n$ contains a single crossing point. □

The existence of a unique crossing point in the interval $p \in (0, 1/n]$ enables a *binary-search like* procedure to narrow down on the possible location of a crossing point. If DQMLD produces the same output for a pair of p's in that interval, it implies that no such crossing point exists between them, and furthermore, that a crossing point lies outside of this interval. This halves the size of the interval between the next pair of points to be queried with. Hence

the location of the crossing point can be obtained to an accuracy of 2^{-j} with at most j queries to DQMLD.

5.1.4 Independent constraints and the promise gap

We have described a polynomial-time method to estimate the location of crossing points within exponential accuracy, i.e. values of p where (Eq. 5.21) is fulfilled. It remains to show that a polynomial number of such linear constraints are sufficient to determine the weight enumerators of the linear code. Every crossing point provides a linear constraint on $\{\mathsf{WE}_i(\mathcal{C}), B_i\}_{i=0}^{n}$. However these conditions are inequalities. In the following lemma, we establish that for Δ sufficiently small, the system of inequalities provides the same integer solutions as the system of equalities (i.e., obtained with $\Delta = 0$).

Lemma 5.1.3 *Provided that $\Delta \leq 2[2+n^{\lambda}]^{-1}$, the first λ weight enumerator coefficients $\{WE_i(\mathcal{C})\}_{i=0}^{\lambda}$ can be extracted efficiently from the value of $2n+1$ crossing points p_k that produce linearly independent inequalities (Eq. 5.21).*

Proof: Given a crossing point p_k, we can rewrite the inequality (Eq. 5.21) as an equality

$$-v_k \sum_{i=0}^{n} \mathsf{WE}_i(\mathcal{C}) \tilde{p_k}^i + \sum_{i=0}^{n} B_i \tilde{p_k}^i + \delta_k \Delta \left[v_k \sum_{i=0}^{n} \mathsf{WE}_i(\mathcal{C}) \tilde{p_k}^i \right] = 0$$

by introducing an unknown parameter $0 \leq \delta_k \leq 1$. Given $2n+1$ distinct crossing points p_k, we obtain a linear system $\mathsf{M} \cdot \omega - \Delta \ (\mathcal{J} \cdot \mathsf{M}) \cdot \omega = \vec{0}$, where $||\mathcal{J}||_{\infty} \leq 1$ and ω is the vector containing the weight enumerator coefficients. Since the location of a crossing point is invariant under multiplying both weight enumerators $\{\mathsf{WE}(\mathcal{C})_i, B_i\}_{i=0}^{n}$ by a constant, it is clear that $2n+1$ linear independent constraints alone would result in the trivial solution. However, the normalization condition

$$\sum_{i=0}^{n} (\mathsf{WE}(\mathcal{C})_i + B_i) = 2^n \tag{5.26}$$

along with the $2n+1$ linearly independent constraints obtained from the location of crossing points is sufficient to determine the weight enumerators coefficients in $\vec{\omega}$,

$$\vec{\omega} \ = \ [\,[\mathbb{I} + \Delta \ \mathcal{J}] \cdot \mathsf{M}\,]^{-1} \cdot \vec{b} \tag{5.27}$$

where we have defined

$$\mathsf{M} = \begin{bmatrix} 1 & \tilde{p}_1 & \dots & \tilde{p}_1^n & -v_1 & -v_1\tilde{p}_1 & \dots & -v_1\tilde{p}_1^n \\ 1 & \tilde{p}_2 & \dots & \tilde{p}_2^n & -v_2 & -v_2\tilde{p}_2 & \dots & -v_2\tilde{p}_2^n \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\ 1 & \tilde{p}_{2n+1} & \dots & \tilde{p}_{2n+1}^n & -v_{2n+1} & -v_{2n+1}\tilde{p}_{2n+1} & \dots & -v_{2n+1}\tilde{p}_{2n+1}^n \\ 1 & 1 & \dots & 1 & 1 & 1 & \dots & 1 \end{bmatrix} \tag{5.28}$$

$$\vec{b} = \begin{pmatrix} 0 & \cdots 0 & 2^n \end{pmatrix}^{\mathsf{T}}, \ \vec{\omega} = \begin{pmatrix} B_0 & \cdots & B_n & \mathsf{WE}_0(\mathcal{C}) & \cdots & \mathsf{WE}_n(\mathcal{C}) \end{pmatrix}^{\mathsf{T}} \tag{5.29}$$

$$\mathcal{J} = \left[\begin{array}{ccc|ccc} 0 & & & & & \\ & \ddots & & & \mathbb{0} & \\ & & 0 & & & \\ \hline & & & -\delta_1 & & \\ & \mathbb{0} & & & \ddots & \\ & & & & & -\delta_{n+1} \\ \hline 0 & \dots & 0 & 0 & \dots & 1 \end{array}\right] \text{ with } 0 \leq \delta_i \leq 1, \ \forall\, 1 \leq i \leq n+1, \tag{5.30}$$

$$\text{and } v_k = \frac{\sum_{i=0}^{n} B_i \tilde{p}_k^i}{\sum_{i=0}^{n} \mathsf{WE}(\mathcal{C})_i \tilde{p}_k^i}. \tag{5.31}$$

Denote $\vec{\omega}_0$ the solution of this linear system in the case where $\Delta = 0$, i.e., $\vec{\omega}_0 = \mathsf{M}^{-1} \cdot \vec{b}$. This solution is not the weight enumerator, because we omitted the $\Delta\mathcal{J}$ term. However, since the weight enumerator coefficients are integers, we expect that for Δ sufficiently small, rounding up each component of $\vec{\omega}_0$ to the nearest integer will give the correct answer $\vec{\omega}$. For this approach to succeed, it suffices to ensure that $|\vec{\omega}_i - \vec{\omega}_{0,i}| \leq 1/2$. Using (Eq. 5.27) for $\vec{\omega}$ and a similar one for $\vec{\omega}_0$ (with $\Delta = 0$), we can rewrite this condition in terms of the constraint matrix in (Eq. 5.28) as:

$$|\vec{\omega}_i - \vec{\omega}_{0,i}| = \left| \left([\,[\mathbb{I} + \Delta\,\mathcal{J}] \cdot \mathsf{M}\,]^{-1}_{i,n} - [\,\mathsf{M}^{-1}\,]_{i,n} \right) 2^n \right| \leq \frac{1}{2} \tag{5.32}$$

where we exploited the special structure of $\vec{b}$ which ensures that only the elements in the last column of the inverse matrix have a non-trivial contributions to the components of $\vec{\omega}$.

Using properties of matrix multiplication and ∞-norms of matrices in (Eq. 5.32), we obtain

$$\left| [\,[\mathbb{I} + \Delta\,\mathcal{J}] \cdot \mathsf{M}\,]^{-1} - \mathsf{M}^{-1} \right|_{i,n} \leq \left(\left|\left| [\mathbb{I} + \Delta\,\mathcal{J}]^{-1} \right|\right|_\infty - 1 \right) \left| \left[\mathsf{M}^{-1}\right]_{i,n} \right|. \tag{5.33}$$

The explicit form of the matrix in (Eq. 5.30) can be used to verify that $\left|\left| (\,\mathbb{I} + \Delta\,\mathcal{J}\,)^{-1} \right|\right|_\infty \leq (1-\Delta)^{-1}$ and also $[\mathsf{M}^{-1}]_{i,n}\, 2^n = \mathsf{WE}_i(\mathcal{C}) \leq n^i,\ \forall\, 1 \leq i \leq n+1$. Hence we can now state a sufficient condition for (Eq. 5.32) to hold:

$$\left[\frac{1}{1-\Delta} - 1 \right] n^i \ \leq\ \frac{1}{2}, \quad \forall\, 1 \leq i \leq n+1 \tag{5.34}$$

The lemma follows by imposing this condition for all $i \leq \lambda$, yielding

$$\left[\frac{1}{1-\Delta} - 1 \right] n^\lambda \ \leq\ \frac{1}{2} \ \Rightarrow\ \Delta \leq \frac{2}{2+n^\lambda}. \tag{5.35}$$

□

Though it is clear that we must have $2n+1$ linearly independent constraints on $\{\mathsf{WE}_i(\mathcal{C}), B_i\}_{i=0}^n$ along with a promise gap of $\Delta \leq 2[2+n^\lambda]^{-1}$, it is not immediately clear how many distinct crossing points need to be located to obtain these constraints, i.e. some of the crossing points associated to distinct pairs (v_l, p_l) may result in linearly dependent constraints (Eq. 5.21). The following lemma ensures that we can always efficiently find linearly independent constraints.

Lemma 5.1.4 *The identification of at most $8n^2$ distinct crossing points (v_l, p_l) of (Eq. 5.21) will*

produce $2n+1$ *linearly independent constraints.*

Proof: The $(2n+2)\times(2n+2)$ matrix in (Eq 5.28) can be represented as $\mathsf{M} = [\mathsf{V} \,|\, \mathsf{D}\cdot\mathsf{V}]$ where $\mathsf{V}_{i,j} = \tilde{p}_i^j$ is a $(2n+2)\times(n+1)$ *Vandermonde matrix* and $\mathsf{D}_{i,j} = -v_i\delta_{i,j}$ is a diagonal square matrix. We shall prescribe a polynomial-time method to construct a full-rank matrix of this form.

Let T_l represent the sub-matrix of M formed by taking the first l column entries of the first l rows. Hence T_l is a square matrix. Moreover, notice that a sufficient condition for the the matrix in (Eq. 5.28) to describe l independent constraints is that T_l must have a non-vanishing determinant. Hence, it suffices to show that each new constraint can be generated such that the corresponding square matrix T_l satisfy $\det(\mathsf{T}_l) \neq 0$.

Consider the simple case when $l \leq n+1$. In this case, T_l is always given by a *Vandermonde matrix*, which is full rank if and only if all p_i are distinct [57]. Hence, this implies that $n+1$ distinct values of v_i suffice to generate $n+1$ constraints that are necessarily independent, since different v's necessarily produce different p's (see Lemma. 5.1.5).

For $l > n+1$, it is not immediately clear if the procedure prescribed in (Sec. 5.1.2) will provide linearly independent constraints even if we choose all v_i to be distinct. However, a sufficient condition for independency is again the non-vanishing of $\det(\mathsf{T}_l)$. Assume that the choices of $\{v_1,\ldots,v_{l-1}\}$ are such that $\det(\mathsf{T}_{l-1}) \neq 0$. The Laplace-expansion of the determinant suggests that $\det(\mathsf{T}_l) = 0$ when v_l is given by:

$$v_l = \frac{\sum_{i=0}^{n} E_i \tilde{p}_l^i}{\tilde{p}_{n+t}^{t-1}\det(T_{l-1}) + \sum_{j=0}^{t-2}\tilde{p}_{n+j}^{j} C_j} \tag{5.36}$$

where $l > n+1$ and E_i, C_i are constants (independent of p_{n+i}; they are minors of p^i and vp^i respectively).

The above equation classifies the particular values of v_l that fail to generate l linearly independent constraints on $\{\mathsf{WE}_i(\mathcal{C}), B_i\}_{i=0}^{n}$. Note that (v_l, p_l) are related by Eq. (5.31). Hence, equating this with (Eq. 5.36) results in a rational polynomial in p_l, which has at most $2n$ roots, unless the expressions in (Eqs. 5.31, 5.36) are identical. Assume for now that these expressions are not identical. Then we can choose to sample $2n+1$ different values of v_l according to (Eq. 5.31), and for at least one of them, the resulting p_l will violate (Eq. 5.36), so the pair (v_l, p_l) will produces a linearly independent constraint. On the other hand, if the

two functions in (Eqs. 5.31, 5.36) are identical, then we have that $\forall p$:

$$\frac{\sum_{i=0}^{n} \mathsf{WE}_i(\mathcal{C})\tilde{p}^i}{\sum_{i=0}^{n} B_i\tilde{p}^i} = \frac{\sum_{i=0}^{n} E_i\tilde{p}^i}{\tilde{p}^{k-1}\det(T_{n+k-1}) + \sum_{j=0}^{k-2}\tilde{p}^j C_j}. \tag{5.37}$$

The above relation immediately fixes the coefficients $\{\mathsf{WE}_i(\mathcal{C}), B_i\}_{i=0}^{n}$ by just comparing the powers of $\tilde{p}$.

Since we must sample at most $2n+1$ crossing points to obtain a linearly independent constraint on $\{\mathsf{WE}_i(\mathcal{C}), B_i\}_{i=0}^{n}$, we require at most $(2n+1)(2n+1)$ or (for expressional convenience) $8n^2$ distinct crossing points to construct the a full rank matrix of the form in (Eq. 5.28). □

The last ingredient we need is a bound on the distance between crossing points. Remember that we are only able to locate the value of a crossing probability p_k to exponential accuracy. Thus, it is necessary that changing the tunable parameter v has a significant effect on the value of the crossing point in order to generate linearly independent constraints (with significantly different values of p_k). Combining the restriction on the values of the tunable parameter in Eq. (5.25) with Lemma. (5.1.4) immediately tells that the smallest change in the tunable parameter will be at least $(1-\Delta)n^{-n}8^{-1}n^{-2}$. This naturally implies a minimum separation between two crossing points, as the lemma below addresses:

Lemma 5.1.5 *Let $\{v_k, p_k\}_{k=1}^{2n+1}$ denote crossing points of a stabilizer code constructed as above with $|v_k - v_l| \geq [8n^2(1-\Delta)n^n]^{-1}$. Then $|p_i - p_j| \geq 4^{-n\log_2 n}$.*

Proof: We need to relate the difference in v to a difference in p. For this, let us recall that if γ is a small change in v and δ a small change in p, then $\gamma = \delta\,(dv/dp)$, or equivalently: $\gamma(dv/dp)^{-1} = \delta$. Computing the derivative using the expression for v in Eq. (5.31), we have:

$$\begin{aligned}
\delta &= \gamma \cdot \left(\frac{\sum_{i,j=0}^{n} \mathsf{WE}_i(\mathcal{C})\,\mathsf{WE}_j(\mathcal{C}) \left(\frac{p}{1-p}\right)^{i+j}}{\sum_{i,j=0}^{n} B_i\,\mathsf{WE}_j(\mathcal{C}) \left[\left(\frac{p}{1-p}\right)^{j} \frac{d}{dp}\left(\frac{p}{1-p}\right)^{i} - \left(\frac{p}{1-p}\right)^{i} \frac{d}{dp}\left(\frac{p}{1-p}\right)^{j}\right]} \right) \\
&\geq \gamma \cdot \left(\frac{1}{\frac{1}{(1-p)^2}\sum_{i,j=0}^{n}\left[iB_i\,\mathsf{WE}_j(\mathcal{C})\left(\frac{p}{1-p}\right)^{j}\left(\frac{p}{1-p}\right)^{i}\right]} \right) \\
&\geq \gamma \cdot \frac{(1-p)^{2n+2}}{n} \;\Rightarrow\; \delta \geq \gamma \cdot \frac{4^{1-n}}{n}
\end{aligned} \tag{5.38}$$

Since $\gamma \geq (1-\Delta)n^{-n}8^{-1}n^{-2}$, we find: $\delta \geq (1-\Delta)n^{-n}8^{-1}n^{-3}4^{1-n} \geq 4^{-n\log_2 n}$. □

Hence it suffices to estimate the location of each crossing point to within an accuracy of $4^{-n\log_2 n}$, implying that we must run at most $2n\log_2 n$ iterations of the above binary-search like procedure, mentioned at the end of Sec. (5.1.3), to locate the crossing point. Assuming that each query to a DQMLD oracle is answered in constant time, the above procedure takes $\mathcal{O}(n\log_2 n)$ time.

In appendix App. (A.1) we construct a repetition code with polynomially many qubits to encode a single qubit, such that the resulting noise on the encoding qubit obeys $p_Z = q$, $p_I = 1-q$, and $p_Z = p_Y = 0$ with an exponential accuracy (the error-free model is a special case $q = 0$). This implies that the tunable parameters v_k used to find crossing points can be set to exponential accuracy.

With these, we are in a position to prove our main result.

Proof of Thm. (5.1.1) : Given an input classical linear code $\mathcal{C}$, we built a generating set for a stabilizer code, encoding a single qubit, that satisfies (Eq. 5.3). Appending additional qubits to this code and choosing a specific channel (Sec. 5.1.2), on those qubits, enabled the introduction of a tunable parameter in the logical class probabilities (Eqs. 5.18, 5.19). Varying this tunable parameter as per (Lemma. 5.1.2), with a polynomial number of queries to a DQMLD oracle, we can estimate the location of a crossing point to within exponential accuracy, thereby providing a linear constraint on the weight enumerator coefficients. Repeating this procedure $2n+1$ times, we showed in (Lemma. 5.1.3) that, as long as the promise gap is $2[2+n^{\lambda}]^{-1}$, the system of inequalities yield the same solution to the first λ weight enumerator coefficients of $\mathcal{C}$, up to an integer approximation, as the system of equalities considered without any promise gap. Since $\lambda = \mathsf{polylog}(n)$ in (Thm. 5.1.1), it suffices to have a promise gap in DQMLD that is $1/\mathsf{quasi\text{-}polynomial(n)}$. It followed from (Lemma. 5.1.4) that at most $8n^2$ crossing points are sufficient to generate the necessary linearly independent constraints on the weight enumerator coefficients and from (Lemma. 5.1.5) that the accuracy on their location, achievable in polynomial time, is sufficient. Lastly, we showed in (App. A.1) that it suffices to direct all queries to a DQMLD oracle on a independent $X-Z$ channel. Thus we prove (Thm. 5.1.1). □

Chapter 6

Conclusion

In this thesis, we studied the error correction problem for classical as well as quantum codes. The main contribution of the this thesis was presented in (Thm. 5.1.1), by demonstrating that optimal decoding problem for stabilizer codes, called the degenerate quantum maximum likelihood decoding (DQMLD), is a #P-complete problem. Thus, we proved that DQMLD is certainly harder than what was anticipated from the previously known results, cf. [2, 4, 5]. We will close this thesis by mentioning a few open problems in quantum error correction and some problems that we did not address in this thesis.

Firstly, our result for the computational complexity of DQMLD in (Chap. 5) addresses the DQMLD problem on an independent $X - Z$ channel. However, a similar complexity result can be shown with a depolarizing channel as well, by introducing the notion of a *generalized weight*, described in [5]. This will require important modifications to (Sec. 5.1.2) of the proof in (Chap. 5) and the technique in (App. A.1).

The key computational complexity problem in quantum error correction turns out to be study of the *parametrized complexity* (cf. [17]) of the optimal decoding problem with the promise gap as the parameter, cf. (Def. 4.6.1). In this study, we have determined the complexity of the optimal decoding problem for two extreme cases of the parameter, DQMLD is in #P-complete when $\Delta = 1/\text{quasi-polynomial}(n)$, cf. (Thm. 5.1.1) and DQMLD is in NP when $\Delta = 1 - 2^{-n-k}$, cf. (Lemma. 4.7.1). However, for a vast intermediate range of Δ, the computational complexity of DQMLD remains an open problem. As described in (Sec. 4.6.1), the promise gap Δ is related to the decoding failure probability ϵ as $\Delta = 1 - \epsilon$. Thus, the two extreme cases considered above correspond respectively to optimal decoding in a very noisy regime (failure probability approaching unity) and optimal decoding with an exponentially small failure probability. Unfortunately, the case of practical interest falls in between.

The complexity of any problem is only a highlight of the runtime of any algorithm on the worst case instance of the problem, cf. [18, 42]. Hence it is of practical interest to know the runtime of the decoding algorithm for any *typical* inputs. This could refer, for instance, to a *typical* syndrome or a *random* code. It turns out that random stabilizer codes are non-degenerate, cf. [30, 105] – degenerate errors do not affect the decoding strategy in a significant manner, so our result is probably not relevant in this setting. However, for the practically relevant class of *quantum low-density parity-check (LDPC) codes*, the complexity of optimal decoding strategy remains an important open question.

In the course of our reduction (in Thm. 5.1.1), we have assumed a DQMLD oracle that works for *any* stabilizer code. Imposing additional structure on the stabilizer generators would correspond to restricting to a family of stabilizer codes. A famous example is that of quantum LDPC codes, where the weight of each stabilizer generator is upper bounded by a constant (independent of the number of qubits). In the past few years, a significant effort has been made in deriving the stabilizer generators for these codes, cf. [106, 107, 108]. Such a restriction to a type of stabilizer code can also be achieved by imposing additional constraints on the promise gap in (Eq. 4.45) however, details of this approach relate to the difficulties mentioned in the previous paragraph. Moreover, unlike in the classical case, an efficient decoding strategy is not known till today for quantum LDPC codes. For certain specific codes in this subclass, efficient heuristic algorithms are known to exist for performing quantum error correction, see for instance in [101, 35, 34]. Any statement analogous to (Thm. 5.1.1), pertaining to LDPC families of stabilizer codes, would be very interesting.

Lastly, we will highlight two practically far-fetched assumptions which we have made in order to simply our study of the decoding problem. Our analysis is focused on stabilizer codes over Pauli channels. This is particularly convenient due to the discrete nature of the resulting decoding problem. This setting could be generalized in two obvious ways. First, we can consider quantum codes that are not stabilizer codes, defined from a set of commuting projectors. There exists a growing interest for those codes, particularly in the setting of topological quantum order, cf. [98, 109, 110, 111]. In this setting, we could study, for instance, the decoding problem for systems that support non-Abelian anyons, cf. [112]. Second, we can consider errors that are not Pauli operators. This problem is of practical importance because no real-world device undergoes a Pauli channel; for instance the physical process of relaxation is not described by a Pauli channel.

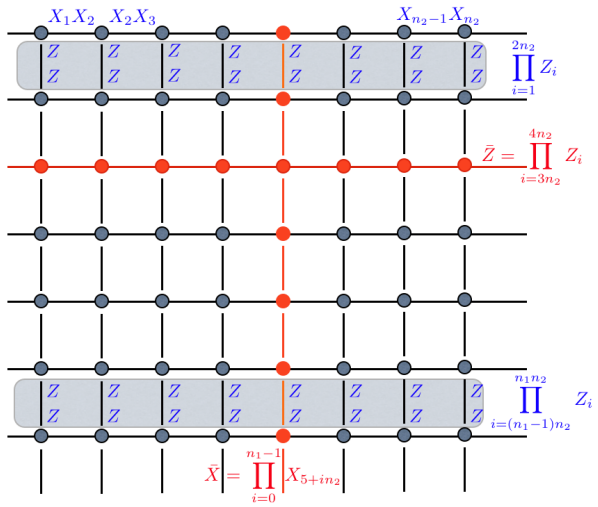
Appendix A

Advanced topics

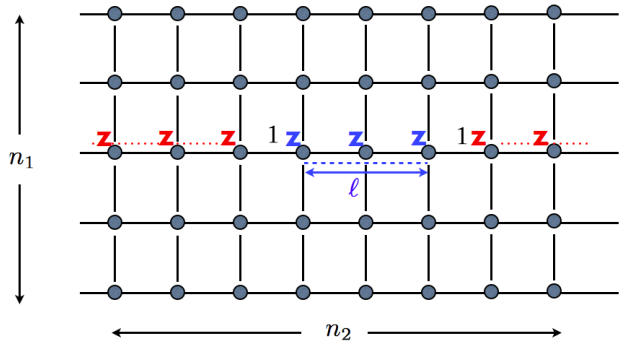
A.1 Simulating Pauli channels with an independent $X - Z$ channel

In our reduction, we have used two features of the a general memoryless Pauli channel, that are not intrinsic to an independent $X - Z$ channel. They involve the freedom of assigning unequal noise rates for $\mathbb{I}, X, Y$ and Z type errors on some qubits of the code. However, when a qubit of the code is encoded into an auxiliary code, the the optimal decoder on this concatenated code will replace the physical noise rates for the qubit with the logical noise rates (logical class probabilities) of the auxiliary code [113]. Therefore, the physical noise rates for that qubit can be controlled by controlling the logical class probabilities of the auxiliary code. The latter can be achieved by varying the syndrome on the auxiliary code.

In our case, the auxiliary code is the Shor code [104, 114] on a $n_1 n_2$ qubits, with $n_1, n_2 \sim \mathsf{poly}(n)$. The stabilizers of this code can be represented graphically on a $n_1 \times n_2$ lattice with a qubits at each vertex, see (Fig. A.1a). Each horizontal link between a pair of vertices represents a $X-$type stabilizer on the qubits corresponding to the vertices. A pair of rows represents a $Z-$type stabilizer on the qubits corresponding to the vertices on those rows.



(a) Stabilizers and logicals of the Shor code on a $n_1 \times n_2$ lattice.



(b) Syndrome corresponding to a $Z-$type error on qubit along a row.

Figure A.1 Shor code on a $n_1 \times n_2$ lattice.

Any $Z-$type error chain will turn on a syndrome, represented by a pair of points on two ends of the chain, c.f. (Fig. A.1b). Let us denote the smallest number of links between the points as ℓ. We will restrict to the case where there is a single continuous error chain, which is confined to a row of the lattice. Hence ℓ completely defines the syndrome. For the syndrome in (Fig. A.1b), one can immediately write the probabilities for the four logical classes $\overline{\mathbb{I}}$, $\overline{X}$, $\overline{Z}$ and $\overline{Y}$ as:

$$\mathsf{Prob}(\overline{\mathbb{I}}|s) = \frac{\left[\frac{p}{2}\left(1-\frac{p}{2}\right)\right]^{\ell}\left[\left(1-\frac{p}{2}\right)\left(1-\frac{p}{2}\right)\right]^{n_2-\ell}}{\tilde{P}(s)} \tag{A.1}$$

$$\mathsf{Prob}(\overline{Z}|s) = \frac{\left[\frac{p}{2}\left(1-\frac{p}{2}\right)\right]^{n_2-\ell}\left[\left(1-\frac{p}{2}\right)\left(1-\frac{p}{2}\right)\right]^{\ell}}{\tilde{P}(s)} \tag{A.2}$$

$$\mathsf{Prob}(\overline{X}|s) = \frac{\left[\frac{p}{2}\left(1-\frac{p}{2}\right)\right]^{\ell}\left[\left(1-\frac{p}{2}\right)\left(1-\frac{p}{2}\right)\right]^{n_2-\ell}\sum_{i=1}^{n_2}\binom{n_2}{i}\left(\frac{p}{2}\right)^{n_1 i}\left(1-\frac{p}{2}\right)^{2n_1n_2-n_1 i}}{\tilde{P}(s)} \tag{A.3}$$

$$\mathsf{Prob}(\overline{Y}|s) = \frac{\left[\frac{p}{2}\left(1-\frac{p}{2}\right)\right]^{n_2-\ell}\left[\left(1-\frac{p}{2}\right)\left(1-\frac{p}{2}\right)\right]^{\ell}\sum_{i=1}^{n_2}\binom{n_2}{i}\left(\frac{p}{2}\right)^{n_1 i}\left(1-\frac{p}{2}\right)^{2n_1n_2-n_1 i}}{\tilde{P}(s)} \tag{A.4}$$

$$\text{where: } \tilde{P}(s) = \left[\left(\frac{p}{2}\right)^{\ell}\left(1-\frac{p}{2}\right)^{2n_2-\ell} + \left(\frac{p}{2}\right)^{n_2-\ell}\left(1-\frac{p}{2}\right)^{2\ell}\right]\sum_{i=0}^{n_2}\binom{n_2}{i}\left(\frac{p}{2}\right)^{n_1 i}\left(1-\frac{p}{2}\right)^{2n_1n_2-n_1 i} \tag{A.5}$$

Given a constant v, we wish to simulate a channel, on the encoded qubit, with $p_I = 1-q$, $p_Z = q$ such that $(1-q)/q = v$. This implies a ratio between $\mathsf{Prob}(\overline{\mathbb{I}}|s)$ and $\mathsf{Prob}(\overline{Z}|s)$ equal to v, and ℓ given by

$$v := \frac{\mathsf{Prob}(\,\overline{\mathbb{I}}\,|\,s\,)}{\mathsf{Prob}(\,\overline{Z}\,|\,s\,)} = \left[\frac{p}{2(1-p)}\right]^{2\ell}\left[\frac{2(1-p)}{p}\right]^{n_2} \;\Rightarrow\; \ell = \frac{1}{2}\left(n_2 + \frac{|\log_2 v|}{1+\log_2\frac{1-p}{p}}\right) \tag{A.6}$$

where in the last simplification, we have assumed $v \leq 1$. As a consequence of the upper bound on v in (Eq. 5.25) it suffices to choose $n_2 = 2n$. We will fix the number of columns in the auxiliary code (Fig. A.1a) to $2n$.

In addition to the ratio between $p_{\mathbb{I}}$ and p_Z, the channel on the encoded qubit also requires that $p_X = p_Y = 0$. To achieve this, we must choose a value of n_1 such that the expressions in (Eqs. A.3, A.4) are vanishingly small. Note that the ratio of the combinatorial sums in the the expressions for $\mathsf{Prob}(\overline{Y}|s)$ and $\tilde{P}(s)$ can be bounded from above as:

$$\frac{\sum_{i=1}^{n_2}\binom{n_2}{i}\left(\frac{p}{2}\right)^{n_1 i}\left(1-\frac{p}{2}\right)^{2n_1n_2-n_1 i}}{\sum_{i=0}^{n_2}\binom{n_2}{i}\left(\frac{p}{2}\right)^{n_1 i}\left(1-\frac{p}{2}\right)^{2n_1n_2-n_1 i}} \leq \left(\frac{p}{2-p}\right)^{n_1} + \frac{n_2^2\left(\frac{p}{2-p}\right)^{2n_1}}{1-n_2\left(\frac{p}{2-p}\right)^{n_1}} \tag{A.7}$$

$$\leq (2n-1)^{-n_1} + \frac{n_2^2(2n-1)^{-2n_1}}{1-n_2(2n-1)^{-n_1}} \tag{A.8}$$

Hence it suffices to take $n_1 = 2n$ for the above ratio to be bounded above by a vanishingly small number. Moreover, as the ratio is an upper bound for $\mathsf{Prob}(\overline{X}|s)$ and $\mathsf{Prob}(\overline{Y}|s)$, we will fix the number of rows in the auxiliary code Fig. (A.1a) to $2n$.

Summarizing, the qubit is encoded into a Shor code on a $2n \times 2n$ lattice (Fig. 4.1) and the syndrome is chosen as indicated in (Fig. A.1b) with ℓ specified by (Eq. A.6). As a result

we have a channel on the qubit with $p_{\mathbb{I}} = 1 - q$, $p_Z = q$, $p_X = p_Y = 0$, to within exponential error bars. Note that an error free channel is a special case of the above channel, with $q = 0$. Hence, we repeat the same encoding but choose the syndrome on the Shor code to be trivial.

Bibliography

[1] E. Berlekamp, R. McEliece, and H. van Tilborg, "On the inherent intractability of certain coding problems (Correspondance)," *IEEE Transactions on Information Theory*, vol. 24, pp. 384 – 386, May 1978.

[2] M. H. Hsieh and F. Le Gall, "NP-Hardness of decoding quantum error-correction codes," *Physical Review A*, vol. 83, p. 052331, May 2011.

[3] P. Iyer and D. Poulin, "Hardness of decoding stabilizer codes," *arXiv:1310.3235*, October 2013.

[4] K. Y. Kuo and C. C. Lu, "On the hardness of decoding quantum stabilizer codes under the depolarizing channel," in *2012 International Symposium on Information Theory and its Applications (ISITA)*, pp. 208 –211, October 2012.

[5] K. Y. Kuo and C. C. Lu, "On the hardnesses of several quantum decoding problems," *arXiv:1306.5173*, June 2013.

[6] R. W. Hamming, *The Art of Doing Science and Engineeering: Learning to Learn*. Gordon and Breach Science Publishers, October 1997.

[7] R. W. Hamming, "Error detecting and error correcting codes," *Bell System Technical Journal*, vol. 29, pp. 147–160, April 1950.

[8] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal (Americal Telephone and Telegraph Co.)*, vol. 27, pp. 379–423, 623–656, July 1948.

[9] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, pp. 21–28, January 1962.

[10] W. Huffman and V. Pless, *Fundamentals of Error-Correcting Codes*. Cambridge University Press, 2003.

[11] R. McEliece, *The Theory of Information and Coding*. Encyclopedia of Mathematics and Its Applications, Cambridge University Press, 2002.

[12] R. Ash, *Information Theory*. Dover Books on Mathematics, Dover Publications, 1990.

[13] F. MacWilliams and N. Sloane, *The Theory of Error Correcting Codes*. North-Holland Mathematical Library, North-Holland, 1978.

[14] A. Barg, *Handbook of Coding Theory*, vol. 1, ch. : Complexity Issues in Coding Theory. Bell laboratories, Lucent Technologies, 600 Mountain Avenue, Room 2C-375, Murray Hill, NJ 07974, USA: Elsevier Science, October 1997.

[15] R. M. Karp, "Reducibility Among Combinatorial Problems," in *Complexity of Computer Computations* (R. E. Miller and J. W. Thatcher, eds.), pp. 85–103, Plenum Press, March 1972.

[16] R. M. Karp, "Combinatorics, complexity and randomness," *Communications of the ACM*, vol. 29, pp. 98–109, February 1986.

[17] S. Arora and B. Barak, *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[18] C. Papadimitriou, *Computational Complexity*. Theoretical computer science, Addison-Wesley, 1994.

[19] M. Sipser, *Introduction to the theory of computation*. Computer Science Series, Thomson Course Technology, 2006.

[20] S. Mertens, "Computational complexity for physicists," *Computing in Science & Engineering*, vol. 4, pp. 31–47, May 2002.

[21] R. P. Feynman, *Feynman Lectures on Computation*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1998.

[22] S. Aaronson, *Quantum Computing Since Democritus*. Quantum Computing Since Democritus, Cambridge University Press, 2013.

[23] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[24] J. Preskill, "Lecture notes for physics 229: Quantum information and computation," 1998.

[25] J. Sakurai and J. Napolitano, *Modern Quantum Mechanics*. Pearson Education, 2014.

[26] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *35th Annual Symposium on Foundations of Computer Science, 1994 Proceedings*, pp. 124–134, November 1994.

[27] S. J. Devitt, W. J. Munro and K. Nemoto, "Quantum error correction for beginners," *Reports on Progress in Physics*, vol. 76, p. 076001, June 2013.

[28] R. Raussendorf, "Key ideas in quantum error correction," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 370, pp. 4541–4565, August 2012.

[29] A. Hagar, "Decoherence," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 370, pp. 4425–4428, August 2012.

[30] D. Gottesman, *Stabilizer Codes and Quantum Error Correction*. PhD thesis, California Institute of Technology, Pasadena, California, May 1997.

[31] D. Lidar and T. Brun, *Quantum Error Correction*. Cambridge University Press, 2013.

[32] F. Gaitan, *Quantum Error Correction and Fault Tolerant Quantum Computing*. Taylor & Francis, 2008.

[33] G. Duclos Cianci and D. Poulin, "Fast decoders for topological quantum codes," *Physical Review Letters*, vol. 104, p. 050504, February 2010.

[34] E. Pelchat and D. Poulin, "Degenerate viterbi decoding," *IEEE Transactions on Information Theory*, vol. 59, pp. 3915–3921, June 2013.

[35] D. Poulin and Y. Chung, "On the iterative decoding of sparse quantum codes," *Quantum Information and Computation*, vol. 8, pp. 987–1000, July 2008.

[36] A. Hutter, J. R. Wootton, and D. Loss, "Efficient markov chain Monte Carlo algorithm for the surface code," *Physical Review A*, vol. 89, p. 022326, February 2014.

[37] B. Copeland, *The Essential Turing*. Clarendon Press, 2004.

[38] D. E. Knuth, "Big omicron and big omega and big theta," *Special Interest Group on Algorithms and Computation Theory (SIGAT) News*, vol. 8, pp. 18–24, Apr. 1976.

[39] V. V. Williams, "Breaking the coppersmith-winograd barrier." Unpublished manuscript, 2011.

[40] D. E. Knuth, *The art of computer programming*. Pearson Education, 2005.

[41] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.

[42] L. Fortnow, *The Golden Ticket: P, NP, and the Search for the Impossible*. Princeton University Press, 2013.

[43] B. A. Cipra, "The ising model is NP-complete," *SIAM News*, vol. 33, pp. 07–00, July 2000.

[44] L. Onsager", "Crystal statistics. i. a two-dimensional model with an order-disorder transition," *Physical Review*, vol. 65, pp. 117–149, February 1944.

[45] S. Istrail, "Statistical Mechanics, Three-dimensionality and NP-completeness: I. Universality of intracatability for the partition function of the Ising model across non-planar surfaces (Extended Abstract)," in *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, STOC 2000, (New York, NY, USA), pp. 87–96, ACM, 2000.

[46] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, (New York, NY, USA), pp. 151–158, ACM, 1971.

[47] B. Trakhtenbrot, "A survey of russian approaches to perebor (brute-force searches) algorithms," *Annals of the History of Computing*, vol. 6, pp. 384–400, October 1984.

[48] F. Barahona, "On the computational complexity of ising spin glass models," *Journal of Physics A: Mathematical and General*, vol. 15, p. 3241, October 1982.

[49] R. Oliveira and B. M. Terhal, "The complexity of quantum spin systems on a two-dimensional square lattice," *Quantum Information and Computation*, vol. 8, pp. 900–924, November 2008.

[50] H. Nishimori, "Gauge glass, spin glass and coding theory – exact results," *Physica A: Statistical Mechanics and its Applications*, vol. 205, pp. 1 – 14, April 1994.

[51] D. Welsh, *Complexity: Knots, Colourings and Countings*. Lecture note series: London Mathematical Society, Cambridge University Press, 1993.

[52] I. Briquel, *Complexity issues incounting, polynomial ecaluation and zero finding*. PhD thesis, Ecole Normale de Lyon and City University of Hong Kong, November 2011.

[53] L. G. Valiant, "The complexity of computing the permanent," *Theoretical Computer Science*, vol. 8, pp. 189–201, October 1979.

[54] S. Toda, "PP is as hard as the polynomial-time hierarchy," *SIAM Journal on Computing*, vol. 20, pp. 865–877, October 1991.

[55] L. Hemaspaandra and A. Selman, *Complexity Theory: Retrospective II*. No. v. 2 in Springer Series in Statistics, Springer New York, 1997.

[56] C. Ikenmeyer, *On the complexity of computing Kronecker coefficients and deciding positivity of Littlewood-Richardson coefficients*. PhD thesis, Fakultät fuï Elektrotechnik and Informatik und Mathematik Institut fuï and Informatik, 33098, Paderborn, October 2008.

[57] K. B. Petersen and M. S. Pedersen, *The Matrix Cookbook*. Technical University of Denmark, Version 20121115 ed., November 2012.

[58] M. Jerrum and A. Sinclair, "Polynomial-time approximation algorithms for the Ising Model," *SIAM Journal on Computing*, vol. 22, no. 5, pp. 1087–1116, 1993.

[59] D. J. A. Welsh, "The computational complexity of some classical problems from statistical physics," *In Disorder In Physical Systems*, pp. 307–321, 1990.

[60] L. A. Goldberg, M. Grohe, M. Jerrum, and M. Thurley, "A complexity dichotomy for partition functions with mixed signs," *SIAM J. Comput.*, vol. 39, pp. 3336–3402, August 2010.

[61] T. Cover and J. Thomas, *Elements of Information Theory*. Wiley, 2012.

[62] R. Blahut, *Theory and practice of error control codes*. Addison-Wesley Pub. Co., 1983.

[63] E. Berlekamp, *Algebraic Coding Theory*. McGraw-Hill series in systems science, Aegean Park Press, 1984.

[64] A. Vardy, "The intractability of computing the minimum distance of a code," *IEEE Transactions on Information Theory*, vol. 43, pp. 1757–1766, November 1997.

[65] K. Genova and V. Guliashki, "Linear integer programming methods and approaches – a survey," *Journal of Cybernetics and Information Technologies*, vol. 11, no. 1, 2011.

[66] E. Moro and E. Martínez-Moro, *Algebraic Geometry Modeling in Information Theory*. Coding Theory and Cryptology, World Scientific, February 2013.

[67] M. N. Vyalyi, "Hardness of approximating the weight enumerator of a binary linear code," *Computing Research Repository*, April 2003.

[68] I. Blake and H. Darabian, "Approximations for the probability in the tails of the binomial distribution (corresp.)," *IEEE Transactions on Information Theory*, vol. 33, pp. 426–428, May 1987.

[69] D. MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, September 2003.

[70] J. Bruck and M. Naor, "The hardness of decoding linear codes with preprocessing," *IEEE Transactions on Information Theory*, vol. 36, pp. 381–385, Mar 1990.

[71] A. Vardy, "Algorithmic complexity in coding theory and the minimum distance problem," in *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, (New York, NY, USA), pp. 92–109, May 1997.

[72] J. T. Coffey, R. M. Goodman, and P. G. Farrell, "New approaches to reduced-complexity decoding," *Discrete Applied Mathematics*, vol. Vol. 33, pp. pp. 43 – 60, November 1991.

[73] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, pp. 260–269, April 1967.

[74] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on Information Theory*, vol. 55, pp. 3051–3073, July 2009.

[75] R. G. Gallager and R. G. Gallager, *Low Density Parity Check Codes*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, July 1963.

[76] S. L. Sweatlock, *Asymptotic weight analysis of low-density parity check (LDPC) code ensembles*. Thesis (Dissertation Ph.D.), Engineering and Applied Science, California Institute of Technology, Pasadena, California, April 2008.

[77] S. Litsyn and V. Shevelev, "On ensembles of low-density parity-check codes: asymptotic distance distributions," *IEEE Transactions on Information Theory*, vol. 48, pp. 887–908, April 2002.

[78] S. Kudekar, T. Richardson, and R. Urbanke, "Spatially coupled ensembles universally achieve capacity under belief propagation," in *IEEE International Symposium on Information Theory Proceedings (ISIT), 2012*, pp. 453–457, July 2012.

[79] S. Kudekar, T. Richardson, and R. Urbanke, "Threshold saturation via spatial coupling: Why convolutional LDPC ensembles perform so well over the BEC," *IEEE Transactions on Information Theory*, vol. 57, pp. 803–834, February 2011.

[80] V. Guruswami and A. Vardy, "Maximum-likelihood decoding of reed-solomon codes is np-hard," in *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, (Philadelphia, PA, USA), pp. 470–478, Society for Industrial and Applied Mathematics, January 2005.

[81] N. Santhi, "Sparse representations for codes and the hardness of decoding ldpc codes," in *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*, pp. 290–294, July 2008.

[82] M. Sudan, "Algorithmic issues in coding theory," in *Foundations of Software Technology and Theoretical Computer Science* (S. Ramesh and G. Sivakumar, eds.), vol. 1346 of *Lecture Notes in Computer Science*, pp. 184–199, Springer Berlin Heidelberg, 1997.

[83] R. Laflamme, "Perfect quantum error correcting code," *Physical Review Letters*, vol. 77, pp. 198–201, July 1996.

[84] E. Knill, R. Laflamme, and W. H. Zurek, "Resilient quantum computation: error models and thresholds," *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 454, pp. 365–384, January 1998.

[85] B. Reichardt, *Mini Crash Course: Quantum Error Correction*. Lecture delivered at the event: Quantum Hamiltonian Complexity Boot Camp, 17 January 2014. Online: http://simons.berkeley.edu/talks/ben-reichardt-2014-01-17.

[86] V. Arvind, P. P. Kurur, and K. Parthasarathy, "Nonstabilizer quantum codes from abelian subgroups of the error group," *arXiv:quant-ph/0210097*, October 2002.

[87] E. Rains, "A nonadditive quantum code," *Physical Review Letters*, vol. 79, pp. 953–954, August 1997.

[88] G. Smith and J. A. Smolin, "Degenerate coding for pauli channels," *Physical Review Letters*, vol. 98, p. 030501, January 2007.

[89] A. Honecker, M. Picco, and P. Pujol, "Nishimori point in the 2D $\pm J$ random-bond Ising model," *Physical Review Letters*, vol. 87, p. 047201, July 2001.

[90] C. Wang, J. Harrington, and J. Preskill, "Confinement-Higgs transition in a disordered gauge theory and the accuracy threshold for quantum memory," *Annals of Physics*, vol. 303, pp. 31–58, January 2003.

[91] H. Bombin, R. S. Andrist, M. Ohzeki, H. G. Katzgraber, and M. A. Martin-Delgado, "Strong resilience of topological codes to depolarization," *Physical Review X*, vol. 2, p. 021004, April 2012.

[92] J.-P. Tillich and G. Zemor, "Quantum LDPC codes with positive rate and minimum distance proportional to $\sqrt{n}$," in *IEEE International Symposium on Information Theory, 2009. ISIT 2009*, pp. 799 –803, July 2009.

[93] D. J. MacKay, G. Mitchison, and P. McFadden, "Sparse-graph codes for quantum error correction," *IEEE Transactions on Information Theory*, vol. 50, pp. 2315–2330, October 2004.

[94] S. Bravyi and M. B. Hastings, "Homological product codes," *arXiv:1311.0885*, November 2013.

[95] T. Camara, H. Ollivier, and J. Tillich, "Constructions and performance of classes of quantum ldpc codes," *arXiv:quant-ph/0502086*, February 2005.

[96] S. Aly, "A class of quantum LDPC codes constructed from finite geometries," in *Global Telecommunications Conference (IEEE GLOBECOM), 2008*, pp. 1–5, November 2008.

[97] S. Aji and R. McEliece, "The generalized distributive law," *IEEE Transactions on Information Theory*, vol. 46, p. 325, March 2000.

[98] A. Y. Kitaev, "Fault-tolerant quantum computation by anyons," *Ann. Phys.*, vol. 303, p. 2, January 2003.

[99] G. Smith and J. Smolin, "Degenerate quantum codes and the quantum channel capacity problem," in *IEEE Information Theory Workshop, 2008. ITW '08*, pp. 358–359, May 2008.

[100] P. Shor and J. Smolin, "Quantum error correcting codes need not completely reveal the error syndrome," *arXiv:quant-ph/9604006*, April 1996.

[101] D. Poulin'', "Optimal and efficient decoding of concatenated quantum block codes," *Physical Review A*, vol. 74, November 2006.

[102] J. Napp and J. Preskill, "Optimal bacon-shor codes," *Quantum Information and Computation*, vol. 13, pp. 490–510, May 2013.

[103] D. Poulin, J.-P. Tillich, and H. Ollivier, "Quantum serial turbo codes," *IEEE Transactions on Information Theory*, vol. 55, pp. 2776–2798, June 2009.

[104] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Physical Review A*, vol. 52, p. 2493, October 1995.

[105] A. Ashikhmin, A. Barg, E. Knill, and S. Litsyn, "Quantum error detection .II. bounds," *IEEE Transactions on Information Theory*, vol. 46, pp. 789–800, May 2000.

[106] D. J. C. MacKay, G. Mitchison, and P. L. McFadden, "Sparse graph codes for quantum error-correction," *IEEE Trans. Info. Theor.*, vol. 50, pp. 2315–2330, October 2004.

[107] T. Camara, H. Ollivier, and J.-P. Tillich, "A class of quantum LDPC codes: construction and performances under iterative decoding," in *IEEE International Symposium on Information Theory, 2007 (ISIT 2007)*, pp. 811–815, June 2007.

[108] J.-P. Tillich and G. Zemor, "Quantum LDPC codes with positive rate and minimum distance proportional to $\sqrt{n}$," in *IEEE International Symposium on Information Theory (ISIT) 2009*, pp. 799 –803, July 2009.

[109] J. Haah and J. Preskill, "Logical operator tradeoff for local quantum codes," *Physical Review A*, vol. 86, p. 032308, September 2012.

[110] O. Landon-Cardinal and D. Poulin, "Local topological order inhibits thermal stability in 2D," *Physical Review Letters*, vol. 110, p. 090502, February 2013.

[111] N. E. Bonesteel and D. P. DiVincenzo, "Quantum circuits for measuring Levin-Wen operators," *Physical Review B*, vol. 86, p. 165113, October 2012.

[112] C. G. Brell, S. Burton, G. Dauphinais, S. T. Flammia, and D. Poulin, "Thermalization, error-correction, and memory lifetime for Ising anyon systems," *arXiv:1311.0019*, October 2013.

[113] D. Poulin", "Optimal and efficient decoding of concatenated quantum block codes," *Physical Review A*, vol. 74, November 2006.

[114] J. Napp and J. Preskill, "Optimal bacon-shor codes," *Quantum Information and Computation*, vol. 13, pp. 490–510, May 2013.