Department of Computer Science
Faculty of Mathematics, Physics and Informatics
Comenius University, Bratislava

Quantum Error Correcting Codes

(Diploma thesis)

Peter Májek

Thesis advisor: doc. RNDr. Daniel Olejár, PhD.

Bratislava, 2005

I hereby declare that I worked on this diploma thesis alone using only the referenced literature.

. . . . . .

ACKNOWLEDGEMENTS

At first place I would like to thank my advisor Daniel Olejár for his guidance and many useful advices. I also would like to thank Michal Sedlák and Ján Bouda for several stimulating conversations and their comments. A special thanks goes to Daniel Gottesman and Andrew M. Steane, who have promptly reacted to my inquiries. I thank also to my thesis opponent Peter Štelmachovič, who has contributed with many relevant remarks. Last but not least, I thank to my whole family, which have always encouraged and supported me during my university study especially during the time I have worked on the thesis.

ABSTRACT

This thesis deals with quantum error correcting codes. In first two chapters necessary introduction to quantum computation and classical error correction is presented. Previous results on construction of quantum error correcting codes are presented in the third and fourth chapter. Mainly Calderbank-Steane-Shor (CSS) codes and stabilizer codes are discussed together with the introduction to coding, decoding and recovery circuits' construction.

Second part of the thesis presents our own results. We have concentrated our effort on the exploration of new CSS codes and examination of their usability. CSS codes are presented as a wide class of quantum codes in literature, but conditions for their practical construction are quite complicated. Well known CSS codes are Steane code correcting errors on a single qubit and [[23,1,7]] code derived from Golay code (correcting three errors). However, no CSS code encoding one logical qubit and correcting errors on up to two qubits is established in the field of quantum error correction. Such code would need to use at least seventeen encoding qubits. We present probabilistic algorithm searching for CSS codes. We used this algorithm and found [[19,1,5]] CSS code. It remains an open question whether 17 or 18 qubits will suffice to construct a CSS code correcting two arbitrary errors.

In last two chapters of the thesis results of numerical and theoretical analysis of found [[19,1,5]] code are shown. The concept of fault tolerant quantum computation is used in the analysis. Found [[19,1,5]] code is compared to Steane code. It follows that [[19,1,5]] CSS code provides better results than Steane code, if fault rate of used quantum gates is below $2,5.10^{-4}$. We have optimized fault tolerant error correction schema for [[19,1,5]] code using the analysis and numerical simulations of several potential architectures. Probability that [[19,1,5]] code would fail to protect encoded qubit is shown (theoretically and also experimentally) to be $O(\xi^3)$, where ξ is probability of single gate failure. In the thesis also coding, decoding and recovery circuits for found [[19,1,5]] code are designed.

ABSTRACT

Táto práca pojednáva o kvantových samoopravných kódoch. V prvých dvoch kapitolách je prezentovaný základný úvod do problematiky kvantového počítania a klasických samoopravných kódov. Výsledky predchádzajúcich prác a publikácii pojednávajúcich o tvorbe kvantových samoopravných kódov sú zhrnuté v tretej a štvrtej kapitole. Pozornost' je upriamená hlavne na Calderbank-Steane-Shor-ové kódy (CSS) a stabilizačné kódy, vrátane konštrukcie im prisluchajúcich obvodov.

Druhá časť práce prezentuje naše vlastné výsledky. Naša práca bola postupne zameraná hlavne na hl'adanie nových CSS kódov a analýzu ich opravovacích schopností. CSS kódy sú síce v literatúre prezentované ako široká trieda kvantových kódov, no podmienky na ich konštrukciu sú značne netriviálne. Pomerne známe CSS kódy sú Steanov kód opravujúci l'ubovolne chyby na jednom qubite a [[23,1,7]] kód odvodený z Golayovho kódu, opravujúci tri chyby. Avšak žiaden CSS kód opravujúci chyby na dvoch qubitoch nie je udomácnený v oblasti kvantového kódovania. Dolný odhad na počet kódujúcich qubitov potrebných na opravu dvoch chýb je sedemnást' qubitov. V tretej kapitole prezentujeme pravdepodobnostný algoritmus, ktorým sme sa pokúsili nájst' CSS kód opravujúci dve chyby a používajúci čo najmenej kodovacích qubitov. Pomocou tohto algoritmu sa nám podarilo nájst' [[19,1,5]] kód. Existencia kódu, ktorý by používal sedemnást' alebo osemnást' kódujúcich qubitov ostáva otvoreným problémom.

V posledných dvoch kapitolách prezentujeme výsledky teoretických analýz a numerických simulácii nájdeného [[19,1,5]] kódu. Všetky analýzy sú vykonané za použitia *fault-tolerant* schém pre kvantové počítanie. Nájdený [[19,1,5]] kód je porovnaný so známym Steanovym kódom. Z prezentovaných analýz vyplýva, že nájdený kód ma pravdepodobnost' zlyhania $O(\xi^3)$, kde ξ je pravdepodobnost' zlyhania jednotlivého hradla v kvantovom obvode. Za použitia teoretických odhadov a numerických simulácii sme zoptimalizovali *fault-tolerant* schému pre nájdený [[19,1,5]] kód. Ak je chybovost' použitých hradiel menšia ako $2,5.10^{-4}$, potom [[19,1,5]] kód preukazuje lepšie správanie ako Steanov kód. V práci sú tiež prezentované kódovacie, dekódovacie a opravné kvantové obvody navrhnuté pre nájdený [[19,1,5]] kód.

LIST OF TABLES

LIST OF FIGURES

CONTENTS

1. INTRODUCTION TO QUANTUM COMPUTING

In this chapter we shortly present preliminaries of quantum computing (QC) necessary to understand the issue of quantum error correcting codes. The effort is paid to depict crucial differences between classical computing and QC. To fully understand details of QC, readers are encouraged to read one of books [1, 2], which provide detailed overview of wide range of QC topics. Readers familiar with QC may skip this preliminary chapter.

1.1 Bits versus Qubits

Since the early ideas of Charles Babbage (1791-1871) [3] and eventual creation of the first computer by German engineer Konrad Zuse in 1941, the basic concept of computers was not changed. Surprisingly, the high-speed modern computers are fundamentally no different from their 20-ton ancestors. Although computers have become much smaller and considerably faster in performing their tasks, the tasks are still the same: to manipulate and interpret an encoding of binary bits into useful computational results.

A fundamental unit of information in classical computers is a bit. Bit can exist in two distinct states either logical 0 or logical 1. The choice of binary bits naturally comes from classical logic and the values 1 and 0 correspond to logical *true* and *false* respectively. We know also more-valued logical systems as fuzzy logic, but the architecture of digital systems is usually based on classical two-valued logic. Each classical bit in computer is physically represented by a macroscopic physical system, such as the magnetization on a hard disk or the charge on a capacitor in the memory. A document, for instance, comprised of n-characters stored on the hard drive of a typical computer is accordingly described by a sequence of $8n$ zeros and ones. In classical algorithm we can read any part of the record without disturbing it. The data may be copied several times, without disrupting the initial data. We can manipulate stored bits via arbitrary *Boolean logic* gates. In other words, the classical bits are fully accessible to the computation.

Herein lie the key differences between classical computer and a quantum computer. Where a classical computer obeys the well-understood laws of classical physics, a quantum computer behaves according to phenomena of quantum mechanics. In a quantum

computer, the fundamental unit of information is called a quantum bit or shortened qubit. In parallel to classical bit, qubit can acquire two different states $|0\rangle$ and $|1\rangle$[1]. These two values constitute a standard base for the qubit. Moreover, qubit can acquire any complex superposition of these two basic states, namely $\alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers. This qubit property arises as a direct consequence of the laws of quantum mechanics which radically differ from the laws of classical physics. Though mathematical theory allows arbitrary coefficients α, β, usually only normalized combinations are considered, what means that

$$|\alpha|^2 + |\beta|^2 = 1. \tag{1.1}$$

The superposition phenomenon may seem counterintuitive because our every day experiences are governed by classical physics, not quantum mechanics – which rules the world at the atomic level. This strange concept can be explained through an experiment. Consider an experiment on figure 1.1:

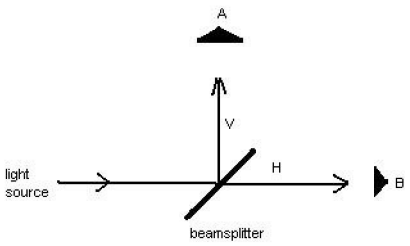


Fig. 1.1: Experiment a.

Here a light source emits a photon along a path towards a half-silvered mirror. This mirror splits the light, reflecting half of it vertically toward a detector A and transmitting other half toward a detector B. A photon, however, is a single quantum of energy and cannot be split, so it is detected with equal probability at either detector A or B. It can be shown that the photon does not split by verifying that if one detector registers a signal, then second detector does not. It suggests that any given photon travels either vertically or horizontally, randomly choosing between the two paths at the beam splitter. However, quantum mechanics predicts that the photon actually travels both paths simultaneously, collapsing into one path only after measurement in detectors A and B! This is more clearly demonstrated by more elaborate experiment shown on figure 1.2.

In this experiment, the photon first encounters a half-silvered mirror, then a fully silvered mirror, and finally another half-silvered mirror before reaching a detector A or

[1] Notation like '$|\ \rangle$' is called the *Dirac notation*.

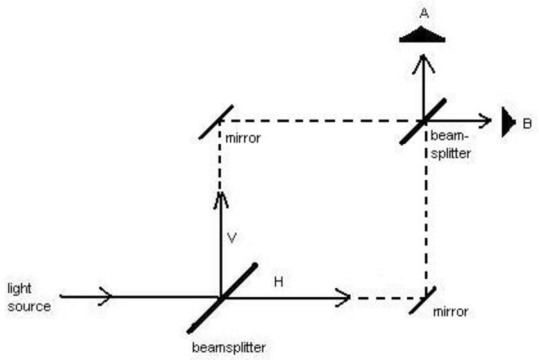


Fig. 1.2: Experiment b.

B. If we hypothesize a theory, that a photon randomly chooses a path after encountering each beam splitter, then both detectors A and B should register signals 50% of time. However, experiment shows that in reality the detector A always registers the photon, and never the detector B! The only conceivable conclusion is that every particular photon is somehow traveling both paths simultaneously, creating interference at the point of intersection that destroyed the possibility of the signal reaching B. This is known as quantum interference and results from the superposition of the possible photon states (potential paths). The difference from classical wave inference is that the photon is interfering with its own possible states and not with other photon. If, for example, we would put an obstacle to one path of the photons, both detectors would register signal in 50% of times. Consider that we would put some kind of device to the one path witch would try to detect if the photon is traveling that way but let it pass through. Even this slight dissimilarity cause that our measurement destroys photon superposition from both ways and place it just to one of them which will result again that detector B would register a photon 50% of times. This unique characteristic makes the current research in *QC* not merely a continuation of previous ideas about a computer, but rather an entirely new branch of thought.

1.2 Feasibility of Quantum Computers

The field of *QC* has made numerous promising advancements since its conception, including the building of five-qubit quantum computer [4] capable of some simple arithmetic. However, a few large obstacles still remain that prevent us from building a real quantum computer. Among these difficulties, error correction, decoherence, and hardware architecture are probably the most crucial issues. Decoherence is a tendency of a quantum system (qubits) to decay from a given initial state into an incoherent state as the system interacts with its surrounding environment. These interactions between

the environment and qubits seem to be unavoidable, and induce the breakdown of information stored in the quantum computer, and thus introduce errors in computation. Before any significant computation could be performed the principles of error correction have to be involved, unless the new decoherence-free quantum technology would be invented. The theoretical backgrounds of error correcting codes in quantum systems are currently quite sufficient. In 1998 [5], researches at Los Alamos National Laboratory and MIT led by Raymond Laflamme managed to spread a single bit of quantum information (qubit) across three nuclear spins in each molecule of a liquid solution of alanine or trichloroethylene molecules. They accomplished this using the techniques of nuclear magnetic resonance (*NMR*). This milestone has provided argument against skeptics, and hope for believers. More information about error correction techniques is provided in chapter 3. Currently, research of quantum error correction continues mainly in groups at Caltech, MIT and Los Alamos.

Current state of the theory of *QC* is much further than experimental realization of quantum computation. *QC* hardware is, still in its infancy and a lot of theoretical results are waiting for real experiments. *NMR* has become the most popular component in quantum hardware architecture, since it provides several significant experimental results (i.e. [4]). Only within the past years, a group from Los Alamos National Laboratory and MIT constructed the first experimental demonstrations of a quantum computer using *NMR* technology. Physicists came also with some other possible representations of qubit. For example:

- the ground and exited states of ions stored in ion trap [6, 7]
- polarizations of photons [8]
- nuclear spin states (as of hydrogen) [9]

Though these devices have mild success in performing experiments, the technologies each have serious limitations. Ion trap computers are limited in speed by the vibration frequency of the modes in the trap. *NMR* devices have an exponential attenuation of signal to noise as the number of qubits in a system increases. Cavity Quantum Electrodynamics (*QED*) [10] is slightly more promising; however, it still has only been demonstrated with a few qubits. The future of quantum computer architecture will probably be different from what we know today; however, the current research has helped to provide insight to obstacles that must be broken in future devices.

1.3 The Potential and the Usage of Quantum Computers

The potential of *QC* is in possible high parallelism. Quantum parallelism is a fundamental feature of majority of quantum algorithms. Quantum parallelism would allow

quantum computers to evaluate a function $f(x)$ for many different values of x simultaneously. The actual computational power of quantum computers is still open problem. There are three classes of quantum algorithms which provide speedup over best known classical algorithms. The first class of algorithms is based on Quantum Fourier transform (QFT). Shor's polynomial[2] algorithms for prime factoring and discrete logarithm [11] are based on QFT. The second class of algorithms are quantum search algorithms, which need $O(\sqrt{N})$ operations to find a specific element in the unsorted set of N elements. The classical algorithms clearly need $O(N)$ operations. The last known class of algorithms, where significant speedup is provided is quantum simulation, whereby a quantum computer is used to simulate quantum systems.

1.4 Basic Operations

Significant difference between classical and quantum computation is in performable operations over single bit or qubit. On the single bit particular *Boolean function* $f : \{0,1\} \to \{0,1\}$ can be performed. The case with qubit is more complicated. First of all, the internal state of the qubit is not accessible to the algorithm absolutely as state of bits are in the classical case. To get an information about the data stored in the qubit, measurement of that qubit must be hold. The formal definition of a measurement will be given later. There is no way to extract exact values of coefficients α and β from equation (1.1) about single qubit. One of possible measurements that could be held on single qubit is measurement in standard basis set, which outcome is either $|0\rangle$ or $|1\rangle$. When the qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ is measured in standard basis set, the measurement outcome $|0\rangle$ occurs with the probability $p(|0\rangle) = |\alpha|^2$ and outcome $|1\rangle$ occurs with the probability $p(|1\rangle) = |\beta|^2$. From equation (1.1) follows that $p(|0\rangle) + p(|1\rangle)$ sums to one. Moreover, after the measurement the initial state $|\psi\rangle$ collapses to the measured state and initial superposition $\alpha|0\rangle + \beta|1\rangle$ is definitely lost.

One could think that it is possible to store arbitrary amount of information in single qubit by encoding that information to the binary representation of α or β. However it is true, this kind of information cannot be retrieved from the single qubit. Only possible way how to get estimations of α and β in superposition formula for the qubit is to design a source which produces sufficiently enough qubits in the same state. By measuring sufficient amount of produced qubits we can get estimations of $|\alpha|^2$ and $|\beta|^2$.

The measurement or even whole representation of computation can be carried also in different basis sets than in already mentioned standard basis set. Very useful basis

[2] The best known classical algorithms are exponential from the number of input's bits.

set is the *dual base* $\{|0'\rangle, |1'\rangle\}$.

$$|0'\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \qquad |1'\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{1.2}$$

Example 1.1. Suppose we prepare a qubit in the state $|\psi\rangle = |1\rangle$. Then we measure this qubit in dual base. What are the probabilities of both possible outcomes $|0'\rangle$ and $|1'\rangle$?

To find out probabilities of particular outcomes we need to express $|\psi\rangle$ in components of dual basis set and find out the complex coefficients in the expression:

$$|\psi\rangle = |1\rangle = \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) - \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{1}{\sqrt{2}}|0'\rangle - \frac{1}{\sqrt{2}}|1'\rangle$$

That means, that both results $|0'\rangle$, $|1'\rangle$ occur with the same probability $|\pm\frac{1}{\sqrt{2}}|^2 = 0.5$.

Remark 1.2. Notice that successive measurements of single qubit in standard base produce the same results[3]: Without lost of generality, suppose that first measurement outcome was $|0\rangle$. So the qubit has collapsed to state $|0\rangle = 1|0\rangle + 0|1\rangle$. Additional measurement will therefore result in state $|0\rangle$ with probability $P(|0\rangle) = |1|^2 = 1$.

The other actions we can perform (beside the measurements) on quantum systems (qubits) are application of operators. Application of operator U on qubit $|\psi\rangle$ changes the state of the qubit to the state $U(|\psi\rangle)$. The operators we are able to perform are not arbitrary. For further discussion is useful *column notation* of a qubit. The qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ can be represented as a column vector $\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$. It follows from the principles of quantum mechanics that only linear operators describe evolutions of closed quantum systems[4]. Consider an operator U which action on $|0\rangle$ and $|1\rangle$ is

$$U(|0\rangle) = |\psi_0\rangle \ , \ U(|1\rangle) = |\psi_1\rangle\,, \tag{1.3}$$

then from linearity of U we obtain the formula

$$U(\alpha|0\rangle + \beta|1\rangle) = \alpha|\psi_0\rangle + \beta|\psi_1\rangle\,. \tag{1.4}$$

Therefore U can be represented by a 2×2 matrix with columns corresponding to column notations of $|\psi_0\rangle$ and $|\psi_1\rangle$. The impact of operator U on qubit $|\psi\rangle$ can be expressed by product of matrix representation of U and vector representation of the qubit $|\psi\rangle$.

[3] We suppose that the second measurement is held immediately after the first one and thus the qubit does not have a time to evolve spontaneously between the measurements.

[4] Closed quantum system is a system, that does not interact with the surrounding environment. It is not easy task to isolate a quantum system, there is lot of unwanted interactions which cause the decoherence to the computation

But not every 2×2 matrix corresponds to an operator from the real world. Necessary and sufficient condition for each operator U, to be applicable on a qubit is: If qubit in normalized state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ enters the operator U, than the outcome $U(|\psi\rangle)$ has to be normalized too. Otherwise we would get to the states which probability distribution does not sum to one. The notation $U|\psi\rangle$ is usually used in quantum mechanics as a simplification of $U(|\psi\rangle)$, and we will use this notation in the reminder of the thesis.

Theorem 1.3. *Linear operator acting on single qubit fulfill normalization preserving condition if and only if $U^\dagger U = I$, where $U^\dagger$ is adjoint of U (obtained by transposing and then complex conjugating U), and I is the 2×2 identity matrix.*

Proof of theorem 1.3 is not so complicated, but rather technical. Interested readers can found it in the appendix.

Matrices and corresponding operators which satisfy condition $U^\dagger U = I$ are called *unitary*. The only possible actions performable on a single qubit are measurements and transformations performed by *unitary* operators. Some of the most important single qubit operators are Pauli operators:

$$X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}; \quad Y \equiv \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}; \quad Z \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{1.5}$$

It can be easily shown that every single qubit operator U can be expressed as a superposition of Pauli operators and identity operator:

$$U = u_I I + u_x X + u_y Y + u_Z Z, \qquad u_I, u_X, u_Y, u_Z \in \mathcal{C}. \tag{1.6}$$

As will be shown later, this property is crucial for correcting arbitrary errors on quantum systems. Other single quantum gate, that will play important part in quantum computations is Hadamard gate:

$$H \equiv \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{1.7}$$

Its effect may be seen as transformation from dual basis set to standard basis set and wise versa.

1.5 Quantum Registers

Let us return back to the classical computation. For more convenient computations over multiple bits computer scientists have introduced the concept of registers. An $n-$bit

register is a sequence of n bits, which can be manipulated with more sophisticated operations. Particular n-bit register can be represented by a binary vector of length n or as n-bit number. Consequently, the register can be in 2^n different states. In other words, $n-$bit register can be represented by elements from n-dimensional vector space over field Z_2.

We shall use registers in QC, too. The basic concept is the same as in the classical case. An $n-$qubit register is just a name for the sequence of n qubits. Such *q-register* has 2^n different base states. But again as in the case of single qubit, any normalized linear superposition[5] of these 2^n base states can be in q-register.

Example 1.4. In general, $2-$qubit register can be in state:

$$|\psi\rangle = \alpha_{00}\,|00\rangle + \alpha_{01}\,|01\rangle + \alpha_{10}\,|10\rangle + \alpha_{11}\,|11\rangle\,, \alpha_{ij} \in \mathcal{C},$$

where normalization condition holds $|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1$. The *column notation* of two-qubit register is a column vector $(\alpha_{00}, \alpha_{01}, \alpha_{10,\alpha_{01}})^T$.

The state of $n-$qubit register can be represented by 2^n dimensional complex vector space. These vector spaces with inner product are called Hilbert spaces. The significant difference between classical registers and *q-registers* is in entanglement. In a classical register are all its bits independent, we can read or manipulate each bit without any inference to other bits. In q-registers the situation may be more complicated. Consider a 3-qubit register in state

$$|\psi_3\rangle = \frac{1}{\sqrt{2}}(|001\rangle + |111\rangle) = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \otimes |1\rangle\,. \tag{1.8}$$

The operator $\otimes$ in (1.8) is called tensor product. The tensor product is a way of putting vector spaces together to form larger vector spaces. The notation $|111\rangle$ is shorter form of exact notation: $|1\rangle \otimes |1\rangle \otimes |1\rangle$. The formal definition of tensor product can be found in appendix in (B.1)-(B.3).

We can see that the third qubit of $|\psi_3\rangle$ is in state $|1\rangle$. Since $|\psi_3\rangle$ cannot be rewritten in the form $\frac{1}{\sqrt{2}}\,|a_1\rangle \otimes (|a_2a_3\rangle + |b_2b_3\rangle)$, we cannot say in what state is the first qubit. The first and second qubits are *entangled* and we cannot get the full description of one of them without mentioning also the other. Partial description of such qubits could be given by formalism of density operators (see section 1.7). More about entangled qubits and entanglement can be found in [12].

[5] With coefficients from complex numbers.

1.6 Hilbert Spaces

Hilbert spaces provide mathematical apparatus suitable for description and formalisation of QC operations.

Definition 1.5. The Hilbert space $\mathcal{H}$ is the vector space with complex inner product such that each Cauchy sequence of vectors in $\mathcal{H}$ converges.

A function $(\cdot,\cdot)$ from $\mathcal{H}\times\mathcal{H}$ to $\mathcal{C}$ is an inner product if it satisfies the requirements that:

1. $(\cdot,\cdot)$ is linear in the second argument,
$$\left(|v\rangle, \sum_i \lambda_i |w_i\rangle\right) = \sum_i \lambda_i \left(|v\rangle, |w_i\rangle\right). \tag{1.9}$$
2. $\left(|v\rangle, |w\rangle\right) = \left(|w\rangle, |v\rangle\right)^*$
3. $\left(|v\rangle, |v\rangle\right) \geq 0$ with equality if and only if $|v\rangle = 0$.

Let us return to the example 1.4 from previous section. To the 2−qubit register corresponds Hilbert space H_4 with vector space basis:

$$|u_1\rangle = |00\rangle, \ |u_2\rangle = |01\rangle, \ |u_3\rangle = |10\rangle, \ |u_4\rangle = |11\rangle. \tag{1.10}$$

So every single vector from H_4 can be written as $|v\rangle = \sum_{i=1}^{4} \alpha_i |u_i\rangle$, $\alpha_i \in \mathcal{C}$. The possible way how to define inner product over H_4 is

$$\left(\sum_{i=1}^{4} \alpha_i |u_i\rangle, \sum_{j=1}^{4} \beta_j |u_j\rangle\right) = \sum_{i=1}^{4} \alpha_i^* \beta_i. \tag{1.11}$$

Of course, this is not the only way how to define inner product. The inner product express how are two vectors (states) in Hilbert space similar. Actually, vectors $|v\rangle$ and $|w\rangle$ are not perfectly recognizable by any measurement, unless $\left(|v\rangle, |w\rangle\right) = 0$. The Gram-Schmidt procedure tells us that there always exists orthonormal basis set for each vector space. The elements of orthonormal basis set are therefore discernible. Usual assumption is that 2^n-dimensional Hilbert space $\mathcal{H}$ was chosen in such a way, that vectors $|0\rangle, |1\rangle, \ldots, |2^n-1\rangle$ are mutually orthogonal. The standard quantum mechanics notation for inner product $\left(|v\rangle, |w\rangle\right)$ is $\langle v|w\rangle$. We will use this notation in the remainder of the thesis.

Another concept of Hilbert space formalism used in quantum computation is the *outer product*. It is useful to represent linear operators using concept of *inner product*.

Suppose $|v\rangle$ and $|w\rangle$ are vectors in Hilbert space $\mathcal{H}$. We define $|v\rangle\langle w|$ to be the linear operator on $\mathcal{H}$. The action of operator $|v\rangle\langle w|$ on particular vector $|u\rangle \in \mathcal{H}$ is defined as

$$(|w\rangle\langle v|)(|u\rangle) \equiv |w\rangle\langle v|u\rangle = \langle v|u\rangle |w\rangle . \tag{1.12}$$

1.6.1 Quantum Operators

The basic ideas about operations on single qubit were given in part 1.4. Now we propose more formal and complex definitions for the case of more dimensional quantum systems based on formalism of Hilbert spaces. For every closed quantum system there exists a corresponding Hilbert space, which dimension is $2^{\textbf{number of particles}}$. States of the system correspond to the normalized vectors from that Hilbert space.

Definition 1.6. Let $\mathcal{H}$ be a Hilbert space and A be an arbitrary linear operator on $\mathcal{H}$. If for operator B and all vectors $|v\rangle$, $|w\rangle \in \mathcal{H}$ holds

$$(|v\rangle, A|w\rangle) = (B|v\rangle, |w\rangle), \tag{1.13}$$

then operator B is called *adjoint* or *Hermitian conjugate* of the operator A.

Lemma 1.7. *Let $\mathcal{H}$ be finite dimensional Hilbert space. There exists unique Hermitian conjugate operator $A^\dagger$ for every linear operator A on $\mathcal{H}$.*

Proof of lemma 1.7 is given in appendix.

Definition 1.8. Let U be an arbitrary operator on Hilbert space $\mathcal{H}$. Then U is called *unitary*, if $U^\dagger U = I$, where I denotes identity[6] operator.

It follows from the same principle as in single qubit case, that only unitary operators may be performed on closed quantum systems. Measurements have slightly different character since they are not operations on closed system, but involve interactions with system used for the measurements.

Definition 1.9. The operator A satisfying the identity $A^\dagger = A$ is called *Hermitian*.

Definition 1.10. Let $\mathcal{H}$ be a Hilbert space, then the operator $P : \mathcal{H} \to \mathcal{H}$ is called *projection operator* if it satisfies

$$P = P^\dagger \tag{1.14}$$

$$P = P^2. \tag{1.15}$$

[6] $I|v\rangle = |v\rangle$ for all $|v\rangle \in \mathcal{H}$

Definition 1.11. Let $\mathcal{H}$ be a Hilbert space, then the operator $M : \mathcal{H} \to \mathcal{H}$ is *positive* if for all $|\psi\rangle \in \mathcal{H}$, $\langle\psi|M|\psi\rangle \geq 0$. The notation is $M \geq 0$.

There is plenty of unitary and hermitian operators. One of the most useful unitary operators in QC are *controlled operators*. The *controlled operators* have two different inputs: control qubits and target qubits. The effect is following: If all control qubits are $|1\rangle$, than desired operator is applied to target qubits. The prototypical controlled operation is controlled-NOT (CNOT). If CNOT's control qubit is $|1\rangle$ then X operator is applied to its target qubit. Thus the action of CNOT in computational basis set is given by $|c\rangle\,|t\rangle \overset{CNOT}{\to} |c\rangle\,|t \oplus c\rangle$[7] and its matrix representation is

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{1.16}$$

Example 1.12. Application of CNOT to so known EPR pair $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ will result in disentangled state $|0'\rangle\,|0\rangle$.

In order to have a quantum computer it is important to be able to perform all unitary operators with arbitrary precision using some small basic set of physically implementable unitary operators. In classical computers NAND itself comprise an gate from which all other Boolean functions may be constructed. In quantum case the problem is more complicated. There are several known universal sets of quantum gates [13], but figuring out how to get a given operator with the minimum number of basic gates is still a hard problem of quantum algorithm design. For instance, CNOT gate together with all single qubit gates comprise an universal set.

The algorithms of QC are often represented by quantum circuits. The semantics of quantum circuit is similar to the semantics of classical circuits. Variables (or qubits) correspond to particular wires. Wires may enter a gate (usually from the left side) and the output of the gate is pushed to the output wires of the gate. Several gates may be composed together and connected with wires to represent more complicated computations. Gates often used in quantum circuits are shown on figure 1.3.

[7] The operator $\oplus$ stays for addition modulo 2. Shortcuts c and t stand for control and target qubits respectively.

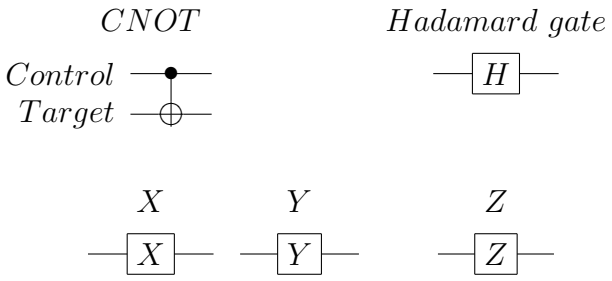


Fig. 1.3: Circuit representation for CNOT, X, Z,Y and Hadamard gate.

1.6.2 General Quantum Measurements

Single qubit measurements were informally introduced in section 1.4. Formally, quantum measurement is described by a collection $\{M_i\}$ of measurement operators on system being measured. The index i refers to the measurement outcome that may occur in the measurement. If the state of the quantum system before the measurement is $|\psi\rangle$ then the probability that the result i occurs is:

$$p(i) = \langle\psi| M_i^\dagger M_i |\psi\rangle \tag{1.17}$$

and the state of the system after the measurement is

$$\frac{M_i |\psi\rangle}{\sqrt{\langle\psi| M_i^\dagger M_i |\psi\rangle}}, \tag{1.18}$$

where following conditions $p_i \in \mathcal{R}, p_i \geq 0$ and $\sum_i p(i) = 1$ must hold. The first two conditions are fulfilled if we require all M_i to be *positive* operators. The last condition is equivalent to the completeness equation $\sum_i M_i^\dagger M_i = I$. As we have mentioned before we require normalized quantum states ($\langle\psi|\psi\rangle = 1$). The factor $\sqrt{\langle\psi| M_i^\dagger M_i |\psi\rangle}$ occurs in the formula (1.18) to maintain the state normalized after the measurement.

Example 1.13. Let V be a 2-qubit system and let V be in state $|\psi\rangle = |00\rangle$. We perform measurement in dual base. What are the possible measurement outcomes and their probabilities?

Let's analyze the example in the standard base. Formally the measurement in the dual base has four measurement components, expressed in the form of outer product:

$$M_1 = |0'0'\rangle \langle 0'0'| \qquad M_2 = |0'1'\rangle \langle 0'1'|$$
$$M_3 = |1'0'\rangle \langle 1'0'| \qquad M_4 = |1'1'\rangle \langle 1'1'|$$

The matrix representation of these operators in the standard base turns to be

$$M_1 = \begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix} \qquad M_2 = \begin{pmatrix} \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} \end{pmatrix}$$

$$M_3 = \begin{pmatrix} \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix} \qquad M_4 = \begin{pmatrix} \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} \\ -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\ \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} \end{pmatrix}$$

To calculate probability of outcome i we need to calculate expression $\left\langle 00|M_i^\dagger M_i|00 \right\rangle$, which is equivalent to $\langle 00|M_i^2|00\rangle$. It can be shown that $M_i^2 = M_i$ for each i, so we need to calculate $p(i) = \langle 00|M_i|00\rangle$. To calculate $M_i\,|00\rangle$ one has to multiply $M_i \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}^T$. Namely:

$$M_1 \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \frac{1}{4}\,|0'0'\rangle \qquad M_2 \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} = \frac{1}{4}\,|0'1'\rangle$$

$$M_3 \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix} = \frac{1}{4}\,|1'0'\rangle \qquad M_4 \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix} = \frac{1}{4}\,|1'1'\rangle$$

It follows that possible outcomes are $|0'0'\rangle$,$|0'1'\rangle$,$|1'0'\rangle$and $|1'1'\rangle$, each with probability of one forth.

Remark 1.14. The previous example would be calculated easier in dual base as matrices M_1 through M_4 represented in dual base would have just one non-zero value and the outcomes would be determined easily even without writing down the matrices. The purpose of the example was not to solve the example, but to show how to work with the measurements formalism.

1.7 Density Operators

The formalism of state vectors from Hilbert space, we have discussed, is one possible representation of quantum systems. This formalism is quite understandable and usable for isolated quantum systems. But when we start to work with systems where some stochastic actions can occur or which are ensembles of bigger systems, which full description is not known than the formalism of density operators or density matrices is

suitable. It is mathematically equivalent to state vectors approach, but in some scenarios frequently occurring in quantum computation it offers more convenient language. More about usage of density operators in quantum information processing can be found in [14].

The density operator is a mean for a description of a quantum state, which is not completely known. Let us to consider a quantum system which is in one of the states $|\psi_i\rangle$, $i \in \{1..k\}$. Suppose that the system is in state $|\psi_i\rangle$ with probability p_i. Then the density operator of this system is defined as

$$\rho = \sum_{i=1}^{k} p_i \left|\psi_i\right\rangle \left\langle\psi_i\right| . \tag{1.19}$$

There is an easy way how to distinguish whether a given operator is a density operator or not.

Theorem 1.15. *An operator ρ is the density operator associated to some ensemble $\{p_i, |\psi_i\rangle\}$ if and only if it satisfies the conditions: (1) (Trace condition) ρ has trace equal to one (2) (Positivity condition) ρ is a positive operator.*

Systems which are with probability 100% in some state $|\psi\rangle$ are called *pure* systems. The system is called to be in a *mixed* state if the system is not fully known and it is described by ensemble $\{p_i, |\psi_i\rangle\}$ or corespoding density operator. In the formulation of density operator we can easily distinguish among pure and mixed states by counting $\mathrm{tr}(\rho^2)$. It can be shown that $\mathrm{tr}(\rho^2) = 1$ for pure states and that $\mathrm{tr}(\rho^2) < 1$ for mixed states.

Unitary transformations: We know that evolutions of closed quantum system are described by unitary operators. When the unitary operator U act upon the system described by density operator ρ, then resulting density operator after operator U took action is

$$\rho' = U\rho U^\dagger. \tag{1.20}$$

Previous definition is consistent with already defined action of U on pure states.

Measurements: Quantum measurements are described by a collection $\{M_k\}$ of *measurement operators*. These are *positive* operators acting on the state space of the system being measured. The index m refers to the measurement outcomes that may occur in the experiment. If the state of the quantum system is ρ immediately before the measurement then the probability $p(i)$ that result i occurs is given by

$$p(i) = tr(M_i^\dagger M_i \rho), \tag{1.21}$$

and the state of the system after the measurement is described by the density operator ρ':

$$\rho' = \frac{M_i \rho M_i^\dagger}{tr(M_i^\dagger M_i \rho)} \tag{1.22}$$

The measurement operators must satisfy the completeness equation

$$\sum_m M_m^\dagger M_m = I. \tag{1.23}$$

2. ERROR CORRECTING CODES

Error correcting codes (ECC) are tools for reliable information processing. Since we do not have perfect hardware which is error free, we need to introduce a concepts of error detection and if possible also their correction. In this chapter we summarize the achievements of ECCs in classical computational theory. We present just results concerning binary linear codes. For more information on classical ECC see textbooks [15] and [16].

2.1 Preliminaries

Consider that we maintain our data in the blocks of n bits, so a single data block can take on 2^n different words from $\{0,1\}^n$. The principle of ECCs is that the set of words from $\{0,1\}^n$ that represent some meaningful information (=*code words*) is subset of $\{0,1\}^n$. If all words from $\{0,1\}^n$ would represent meaningful data, then just a single error in data storage would alter the data and we would not be able to detect the error.

On the other hand if we use some smaller subset of $\{0,1\}^n$ as code words, it is probable that occurrence of an error on the code word will put the result block out of coding subset and we would be able to detect that an error occurred. Even we would be able to guess with high probability the initial code word.

Definition 2.1. An error correcting code is any subset of $\{0,1\}^n$

Definition 2.2. Hamming weight $wt(\mathbf{u})$ of vector $\mathbf{u} \in \{0,1\}^n$ is:

$$wt(\mathbf{u}) = \sum_{i=1}^{n} (u_i = 1) \tag{2.1}$$

Definition 2.3. Minimal distance of code C is:

$$d_C^* = \min_{\mathbf{u},\mathbf{v} \in C, \mathbf{u} \neq \mathbf{v}} (wt(\mathbf{u} \oplus \mathbf{v}))^{1} \tag{2.2}$$

If the code has minimal distance d_C^*, than all its words differ at least on d_C^* positions.

[1] $\oplus$ stands for bitwise addition modulo 2 of vectors $\mathbf{u}$ and $\mathbf{v}$.

What kind of errors do we consider as possible to occur in our systems? We suppose following types of errors:

• The environment cause modification of single transmitted symbol from $\{0, 1\}$ to its opposite.

• The probability of error occurrence on the particular bit does not depend on a position in the data block and on the error occurrence on other bits. The errors are independent of each other.

Usage of code C with minimal distance d^* for information protection allows to detect $d^* - 1$ and correct $\frac{d^*-1}{2}$ errors in proposed error model.

2.2 Linear Codes

Linear codes are the most popular *ECC*s. Their advantage is in existence of convenient mathematical formalism for codes' manipulation. Recall that code words of binary codes are from the set $\{0, 1\}^n$. The set $\{0, 1\}^n$ may be seen as a vector space V over field Z_2 with operations for $\forall \mathbf{u}, \mathbf{v} \in V, a \in Z_2, \mathbf{u} = (u_1, \ldots, u_n), \mathbf{v} = (v_1, \ldots, v_n)$ defined as:

addition: $\mathbf{u} \oplus \mathbf{v} = (u_1 \oplus v_1, \ldots, u_n \oplus v_n)$
multiplication with scalar: $a.\mathbf{u} = (a \cdot u_1, \ldots, a \cdot u_n)$
inner product: $\langle \mathbf{u}, \mathbf{v} \rangle = \sum_{i=1}^{n} u_i v_i \mod 2$

The linear code is any sub-space of V. It follows from the Lagrange theorem that any linear code C has 2^k different elements and dimension k for some $k \leq n$. The notation $[n, k, d]$ stands for linear code which encode vectors from k dimensional space to n dimensional space[2], with minimal distance d and thus corrects $\frac{d-1}{2}$ errors.

Let C be a linear code over V. It is easy to see that set $C^\perp$, defined as

$$C^\perp = \{\mathbf{u} \in V \mid \forall \mathbf{v} \in C \quad \langle \mathbf{u}, \mathbf{v} \rangle = 0\} \tag{2.3}$$

is also subspace of V and therefore forms so known *dual code* to code C. If dimension of C is k then dimension of $C^\perp$ is $n - k$.

Definition 2.4. $[n, k, d]$ linear code C with $d = 2t + 1$ is said to be *perfect* if for every possible word $w_0 \in V$, there is a unique code word $w \in C$,such that $wt(w_0 + w) \leq t$.

It is straightforward to show that C is perfect if

$$\sum_{i=0}^{t} \binom{n}{i} = 2^{n-k}. \tag{2.4}$$

[2] $[n, k, d]$ code has n-elements code words and each codeword encodes k information symbols.

Examples of *perfect* codes are Hamming codes (section 2.3) and the Golay code [17].

2.2.1 Coding and Decoding

Consider k-dimensional linear code C with basis set $\{\mathbf{u}_1, \ldots, \mathbf{u}_k\}$. Our purpose is to find reasonably transformation of arbitrary k information bits to n bits of the code C. To manage this, the basis set of C can be used. The generation matrix G of the code C is a matrix which columns consists of vectors $\mathbf{u}_1$,...,$\mathbf{u}_k$. Now each arbitrary information vector $\mathbf{i}$ consisting of k bits can be transformed to the n bits vector $\mathbf{u}$ from C by formula

$$\mathbf{u} = G\mathbf{i}. \tag{2.5}$$

Both vectors $\mathbf{u}$ and vectors $\mathbf{i}$ are column vectors. Obviously all vectors $\mathbf{u}$ acquired by formula (2.5) are from the code C since $\mathbf{u}$ is linear combination of the C basis vectors. Moreover, if two different information vectors $\mathbf{i}_1$,$\mathbf{i}_2$ are transformed to C, the corresponding code vectors $\mathbf{u}_1$, $\mathbf{u}_2$ are different too.

The coding procedure of b-bits message consists of two steps: the sequence of information bits is divided into $\lceil \frac{b}{k} \rceil$ k-bit groups (blocks), and every k-bit block is multiplied by generation matrix and a corresponding code word is computed. The remaining question is: How can we decode original encoded information and how can we correct potential errors? To answer this question most easily, we introduce an alternative (but equivalent) definition of linear codes in terms of *parity check* matrices. In this alternative definition an $[n, k]$ code is defined to be a set of all $n-$element column vectors $\mathbf{u}$ over Z_2 such that

$$H\mathbf{u} = \mathbf{0}, \tag{2.6}$$

where H is an $n - k \times n$ binary matrix called *parity check* matrix of code C. Parity check matrix H has linearly independent rows to maintain the dimension of C equal to k.

To determine parity check matrix H from generation matrix G we pick $n-k$ linearly independent column vectors $x_1, \ldots, x_{n-k}$ orthogonal[3] to all columns of G and set the rows of H as $x_1^T, \ldots, x_{n-k}^T$.

Example 2.5. Consider linear code $C_1 = [4, 3, 2]$. Every code word of code C_1 contains three information bits and the value of its fourth bit is the check sum of all information bits. Code C_1 has 8 code words

$$\{0000, 0011, 0101, 0110, 1001, 1010, 1100, 1111\}. \tag{2.7}$$

[3] By orthogonal is meant that scalar product is equal to 0.

Basis set of this code may be chosen as $\{0011, 0101, 1100\}$. Then generation matrix G_1 is

$$G_1 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}. \tag{2.8}$$

Now we need to find $n - k = 4 - 3 = 1$ vector which is orthogonal to all vectors of the basis set. Such vector is $(1, 1, 1, 1)^T$ and therefore the parity check matrix for C_1 is $H_1 = [1111]$.

Exactly the parity check matrix is used for the error detection procedure. Consider that k-bit message $\mathbf{v}$ is encoded as $\mathbf{u} = G\mathbf{v}$. During a transmission or just after some time, error $\mathbf{e}$[4] occurs on $\mathbf{u}$ and corrupts it to state $\mathbf{u}' = \mathbf{u} \oplus \mathbf{e}$. We get the error syndrome by applying H on $\mathbf{u}'$

$$H\mathbf{u}' = H(\mathbf{u} \oplus \mathbf{e}) = H\mathbf{u} \oplus H\mathbf{e} = H\mathbf{e} \tag{2.9}$$

If $\mathbf{e}$ is $\mathbf{0}$, which means that no error occur, the error syndrome is also $\mathbf{0}$. In case that $\mathbf{e}$ is not zero and $\mathbf{u}'$ does not belong to the code, nontrivial error syndrome is counted. Important is that the error syndrome is independent from the code word $\mathbf{u}$, it depends only on occurred error. Last part of the error correction scheme we need is a pre-computed values of $H\mathbf{e}'$ for all possible errors $\mathbf{e}'$. If we posses such table and error syndrome $H\mathbf{e}$ occurred we just need to look in the syndrome table to recognize the error $\mathbf{e}$. Once the error is recognized, the addition $\mathbf{u}' \oplus \mathbf{e}$ will cancel the error. Of course, different errors can have the same error syndrome. The error correction is based on the fact that the all errors with weight $wt(e') \leq \frac{d^*_C - 1}{2}$ have different error syndromes. If an error with $wt(e') > \frac{d^*_C - 1}{2}$ occur, than incorrect error may be detected and the encoded information may get lost. Since we consider stochastic source of errors this happens with probability $O(p^{\frac{d^*_C + 1}{2}})$, where p is probability of single error.

After error detection and recovery we would like to decode previous encoded message $\mathbf{v}$. This task can be the most easiest accomplished when basis set of G is orthonormal[5], then the i-th element of $\mathbf{v}$ is counted as

$$v_i = \langle \mathbf{u}_i, \mathbf{u} \rangle, \tag{2.10}$$

where $\mathbf{u}_i$ is i-th vector of the C basis set (or equivalently i-th column of generation matrix). The orthonormal basis set can be constructed from arbitrary basis set using Gram-Schmidt procedure. Decoding of codes in systematic form (section 2.2.3) is also simple.

[4] A vector which has 1s on positions where error occurred and 0s elsewhere.

[5] $\langle \mathbf{u}_i, \mathbf{u}_j \rangle = \delta_{ij}$, where δ_{ij} is Kronecker's delta

2.2.2 Properties of Linear Codes

Following properties are useful in hunting for new linear codes with specific characteristics.

Lemma 2.6. *Minimal distance of linear code C defined in (2.2) is equal to minimal Hamming weight of all code words different from 0.*

$$d_C^* = \min_{\boldsymbol{u}\in C, \boldsymbol{u}\neq 0} wt(\boldsymbol{u}) \tag{2.11}$$

• There is a linkage between minimal distance of particular code and its parity check matrix:

Theorem 2.7. *Linear code contains non-zero code word with weight less or equal d if and only if its parity check matrix H contains d linearly dependent columns.*

Corollary 2.8. (**Singleton bound**) Linear code C with parity check matrix H has minimal distance d^* if and only if in matrix H arbitrary $d^* - 1$ columns are linearly independent and there exists d^* linearly dependent columns of matrix H.

Since parity check matrix has $n - k$ rows we get easily following inequality:

Theorem 2.9. *Linear $[n, k, d]$ code fulfills following inequality:*

$$d \leq 1 + n - k \tag{2.12}$$

2.2.3 Codes in Systematic Form

Suppose a linear code C with basis set $\{\mathbf{u}_1, \ldots, \mathbf{u}_k\}$. From the linear algebra [18] we know that if we change arbitrary $\mathbf{u}_i$ by adding some linear combination of other basis vectors:

$$\mathbf{u'}_i = \mathbf{u}_i + \sum_{i\neq j} a_j \mathbf{u}_j, \tag{2.13}$$

then basis set $\{\mathbf{u}_1, \ldots, \mathbf{u}_i - 1, \mathbf{u'}_i, \mathbf{u}_{i+1}, \ldots, \mathbf{u}_k\}$ span the same vector space C. The modification (2.13) corresponds to adding columns of generation matrix together. Also exchange of generation matrix columns does not change code space C and its correcting abilities. Exchanging rows of generation matrix neither change the correcting abilities of the code. Using all these operations we can transform generation matrix to so known code in *systematic form* [19, 20]. Code in systematic form has generation matrix in the form

$$G_S = \left[\frac{I_k}{P}\right], \tag{2.14}$$

where I_k stands for identity matrix $[k \times k]$ and matrix P is arbitrary with dimensions $[n-k \times k]$. The parity check matrix to matrix G_S is computed very easily, namely:

$$H_S = [P\,|I_{n-k}]\,. \tag{2.15}$$

One can easily verified that $H_S G_S = P + P = 0$. If the code is in *systematic form* its control bits follow after information bits. Decoding of these code is really simple[6], therefore are often used. The idea of *systematic form* may be used to lower the search space, when one tries to find a *ECC*, since significant parts of generation and parity check matrices are already set and only search through smaller search-space is needed.

Each code can be also transformed to the code with the same correcting abilities and in the form $G'_S = [P'\,|I_k]^T$, $H'_S = [I_{n-k}\,|P]$. In this alternative form control bits precede information bits.

2.3 Hamming Codes

Hamming codes are class of linear error correcting codes with parameters $[2^m-1, 2^{m-1}, 3]$. Columns of their parity check matrices are simply all non zero binary vectors of length m. Hamming codes are perfect codes, since they satisfies equation 2.4.

Very useful in quantum error correcting codes is $C = [7, 4, 3]$ Hamming code, which gives base to quantum Steane Code. The parity check matrix of [7,4,3] code is

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}. \tag{2.16}$$

The valuable property of [7,4,3] Hamming code is that its dual code is *weakly self-dual* code, what means that $C^\perp \subset C$. A code for which $C^\perp = C$ is called *self-dual*. Weakly self dual codes are the best material for constructing quantum codes as will be discussed in section 3.2.2. Another popular code which dual code is *weakly self-dual* is [23,12,7] Golay code [17].

[6] One just extract first k bits throwing the rest away.

3. BASICS OF QUANTUM ERROR CORRECTION (QEC)

The error correction, both in classical computation (CC) and quantum computation (QC), is essential for keeping the computation consistent. The need for error correction in QC is even higher than in CC since much more complex errors can occur in quantum systems.

Suppose we would like to protect arbitrary single qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ against error. Note the principal problems of error correction in QC:

1. As we already mentioned in section 1.4 we are not able to reliably recognize the superposition coefficients α,β using just single qubit.

2. According to Non-Cloning theorem [21] we are not able to copy the state of a qubit $|\psi\rangle$ to other qubits to obtain a redundancy required by classical error correction procedures.

3. The problem with error detection: We could not just simple measure qubits to detect an error as in classical error correction codes. The measurement of the qubit destroys hidden superposition and system will collapse to a single measured state. Therefore we need to manage error detection in such a way that we would not gain any information about encoded data. We must ensure that the only information we obtain from the error detection concerns the occurred error.

4. Much more complicated errors can occur in quantum systems. In classical digital systems it is often sufficient to consider just bit-flip errors. Bit-flip error means that a transformation $0 \rightarrow 1$ or $1 \rightarrow 0$ can possibly occur on some transmitted bits with small probability. In classical coding theory we sometimes consider also errors of other kinds, i.e. errors of synchronization, duplication or deletion some parts of transmitted data. In the quantum world principally continuous range of errors changing qubit in state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ to erroneous state $|\psi_e\rangle = \alpha'|0\rangle + \beta'|1\rangle$ can occur. Moreover, the quantum system can get entangled with the environment. Entanglement may lead to the errors later in the computation when the environment is measured and the entanglement with the data qubit will cause a change of the qubit's state. How can we protect against such diverse kind of errors?

The very first insights were really pessimistic and rose the question whether the error correction in quantum systems is even possible. Fortunately, as was proven later,

the opposite is true. The entanglement first looked as a obstacle in quantum processing, now it is principal tool of reliable quantum information processing. Several quantum codes were developed and some of them practically verified [22].We shall present the main ideas on *QEC* in this chapter. In the first section we formally describe possible kinds of errors which can occur in quantum systems (*QS*) and different formalisms for description of such errors. In section 3.2 we introduce several *QEC* codes usable in *QS*. New code searching technique is presented in this part. The last topic discussed in this chapter is construction of coding and decoding circuits for some *QEC* codes.

3.1 Quantum Noise and Quantum Operations

We begin with the description of noise that can occur during transmission and processing of data and then we introduce the necessary formalism. The quantum noise can be described as any other quantum operations. In quantum systems can not occur anything else than quantum operations. We already described two types of quantum operations. The first are unitary operators and the second types of operations are measurements. The evolution of closed system can be described by unitary operators. The measurement introduces an outer interference (usually desired) with the principal system. Unfortunately, there are also undesirable interactions between environment and the principal system. Since the perfectly isolated system just does not exist, the systems suffer from interactions with outer environment.

The mathematical formalism describing evolutions in open systems has been studied for last 30 years. We will introduce the crucial notions and main results of *QECC* theory, which are necessary for our further work. More information and summaries of quantum operations formalism from the point of view of *QEC* can be found in [23] or [24].

3.1.1 Operator Sum Representation

A noisy quantum channel can be a regular communication channel suffering from some inference with environment, or it can be the time period during which qubits does not change the spatial position but can interact with environment and change their state. It can also represent an imperfectly implemented quantum gate which introduce some noise to qubits. All these cases can be modeled by so known *Operator sum representation*. The effect of quantum channel can be described as

$$\rho' = \mathcal{E}(\rho), \tag{3.1}$$

where $\mathcal{E}$ is a quantum operation transforming initial density operator ρ to density operator ρ'. For unitary operator U and measurement which outcome is m are $\mathcal{E}_U(\rho) =$

$U\rho U^\dagger$ and $\mathcal{E}_m(\rho) = M_m \rho M_m^\dagger$, respectively. It turns out that any quantum operation and consequently any noisy quantum channel can be described as

$$\mathcal{E}(\rho) = \sum_i E_i \rho E_i^\dagger, \tag{3.2}$$

for some set of operators $\{E_i\}$ which map the input Hilbert space to the output Hilbert space, and $\sum_i E_i^\dagger E_i \leq I$. Note that input and output Hilbert spaces need not be the same. This formalism therefore allows us to describe all processes as coding, decoding or error recovery in uniform way. Nevertheless, we will usually use different means for description of these processes, to make the presentation more comprehensible.

There is a following physical interpretation of the operator sum representation. The action of operation $\mathcal{E}$ with operation elements $\{E_i\}$ is equivalent to random replacing of the initial state ρ with the state $E_k \rho E_k^\dagger / \mathrm{tr}(E_k \rho E_k^\dagger)$ with probability $\mathrm{tr}(E_k \rho E_k^\dagger)$. We require that the probability of possible outcomes must sum to one:

$$1 = \sum_i tr(E_i \rho E_i^\dagger) = tr\left(\sum_i E_i \rho E_i^\dagger\right) = tr\left(\sum_i E_i E_i^\dagger \rho\right). \tag{3.3}$$

Equation (3.3) must hold for all density operators ρ which have trace equal to 1 and therefore

$$\sum_i E_i E_i^\dagger = I. \tag{3.4}$$

Quantum operations satisfying the equation (3.4) are called *trace-preserving*. We shall use only trace-preserving operations[1].

The output of quantum operations may differ significantly from its input state. Complex mixed state may be a result of quantum operation applied on pure state. How can we correct mixed state to the initial pure state?

The idea of error correction of mixed states is following: Mixed state can be considered as an ensemble of pure states with some probability distribution. If we are able to correct each pure state of that ensemble to the initial pure state, we are able to return the mixed state to its initial state and thus brake the entanglement of the system with its outer environment.

3.2 Quantum Codes

We begin with design of a *QEC* code for simpler quantum channel. Let us suppose that the quantum channel is transmitting encoded qubits according to following rules.

[1] However, there are also not-trace preserving quantum operations. For further details see [25].

With probability $1-p$ it remains the qubit untouched and with probability p the qubit $|\psi\rangle$ is subjected to operator X which has following effect on single qubit:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \overset{X}{\rightarrow} \beta\,|0\rangle + \alpha\,|1\rangle \tag{3.5}$$

This error is called bit-flip as it flips between $|0\rangle$ and $|1\rangle$. We suppose that no error can occur during the qubits encoding, error detection or decoding.
Simple idea how to protect data against bit flip errors consists in encoding logical qubit $\alpha\,|0\rangle + \beta\,|1\rangle$ as three entangled qubits

$$\alpha\,|0\rangle + \beta\,|1\rangle \overset{X\text{coding}}{\rightarrow} \alpha\,|000\rangle + \beta\,|111\rangle\,. \tag{3.6}$$

Since Non-Cloning theorem holds, we cannot perform the following encoding of unknown qubit:

$$\alpha\,|0\rangle + \beta\,|1\rangle \overset{\text{coding}}{\rightarrow} (\alpha\,|0\rangle + \beta\,|1\rangle) \otimes (\alpha\,|0\rangle + \beta\,|1\rangle) \otimes (\alpha\,|0\rangle + \beta\,|1\rangle) \tag{3.7}$$

The coding (3.6) can be performed using the circuit on figure 3.1.

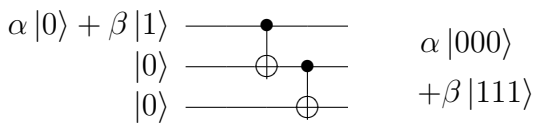


Fig. 3.1: Coding circuit for encoding (3.6).

During a transmission only bit-flip errors possibly occur on some qubits in our model of noisy channel. For example if the bit flip would occur on second qubit of encoded data, the state of three qubits would be $|\psi_2\rangle = \alpha\,|010\rangle + \beta\,|101\rangle$ The error correction is based on two procedures, the first procedure (*error detection*) detects the error and then the second procedure (*recovery from error*), using the information gained by error detection recovers the initial state. Error detection procedure can be performed by projective measurement, with four projection operators:

$$P_0 = |000\rangle\langle 000| + |111\rangle\langle 111| \quad (\texttt{no error}) \tag{3.8}$$

$$P_1 = |100\rangle\langle 100| + |011\rangle\langle 011| \quad (\texttt{bit flip on first qubit}) \tag{3.9}$$

$$P_2 = |010\rangle\langle 010| + |101\rangle\langle 101| \quad (\texttt{bit flip on second qubit}) \tag{3.10}$$

$$P_3 = |001\rangle\langle 001| + |110\rangle\langle 110| \quad (\texttt{bit flip on third qubit}) \tag{3.11}$$

If an error on i-th qubit occurs on the state (3.6) and transforms the three qubits to the state $|\psi_i\rangle$ then $\langle\psi_i|\,P_j\,|\psi_i\rangle = \delta_{ij}$ what imply that the outcome of error detection

on the state $|\psi_i\rangle$ is certainly i. Notice that this measurement never changes the state of measured system, since $P_i |\psi_i\rangle = |\psi_i\rangle$.

Recovery procedure is rather simple. If error on i-th qubit occurred, the measurement gives us the output i. If no error occurred the outcome of the error measurement is 0. The recovery action is following: if output of the error detection is 0 no action is needed, otherwise if output i is obtained then we will flip i-th qubit back to its initial encoded state.

Other important class of errors on qubits is so known phase flip (application of Z operator) which can be formally described in following way:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle \xrightarrow{Z} \alpha |0\rangle - \beta |1\rangle \tag{3.12}$$

The previous code does not protect against this kind of errors, but we can transform problem of protecting against phase-flip errors to previous, already solved problem. The main idea is that the phase-flip has the same effect on states $|0'\rangle$ and $|1'\rangle$ as bit-flip on states $|0\rangle$, $|1\rangle$. Therefore we can use encoding

$$\alpha |0\rangle + \beta |1\rangle \overset{X\text{coding}}{\rightarrow} \alpha |0'0'0'\rangle + \beta |1'1'1'\rangle . \tag{3.13}$$

We also change 1 to $1'$ and 0 to $0'$ in defining error-detection measurement in formulas (3.8)-(3.11) and instead of bit-flips in error-recovery we perform phase-flips to correct encoded state. Both previous codes can protect against single bit flip or phase flip, respectively. However none of them can correct both types of errors. This problem will be solved in following section.

3.2.1 The Shor Code

The problem of *QEC* is more complicated if we want to protect data against arbitrary error on single qubit. It turns out that a code which can correct both bit flip and phase flip errors is able to correct an arbitrary error on single qubit [26, 27]. The first solution of this problem provided Shor by introducing so known 9-qubits *Shor code* which protects against arbitrary error on a single qubit [28]. The Shor code encodes one logical qubit to nine qubits as:

$$|0\rangle \rightarrow |0_L\rangle \equiv \frac{(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)}{2\sqrt{2}} \tag{3.14}$$

$$|1\rangle \rightarrow |1_L\rangle \equiv \frac{(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)}{2\sqrt{2}} \tag{3.15}$$

$$\alpha |0\rangle + \beta |1\rangle \rightarrow \alpha |0_L\rangle + \beta |1_L\rangle \tag{3.16}$$

Let us shortly describe the idea underlying the correction of arbitrary single qubit error. Suppose the noise has impact on a single qubit. We mentioned in section 3.1

that noise can be described by quantum operation $\mathcal{E}$ with operation elements E_i. The state of encoded qubit is $|\psi\rangle = \alpha|0_L\rangle + \beta|1_L\rangle$ to which correspond density operator $|\psi\rangle\langle\psi|$. The effect of noisy channel can be expressed as:

$$\mathcal{E}(|\psi\rangle\langle\psi|) = \sum_i E_i|\psi\rangle\langle\psi|E_i^\dagger \tag{3.17}$$

That means, that the initially pure state $|\psi\rangle$ transforms to the ensemble

$$\left\{tr(E_i|\psi\rangle\langle\psi|E_i^\dagger), E_i|\psi\rangle/\sqrt{tr(E_i|\psi\rangle\langle\psi|E_i^\dagger)}\right\}. \tag{3.18}$$

Now it is sufficient to show that our error correction procedure properly corrects each ensemble state $E_i|\psi\rangle$ to the initial state $|\psi\rangle$. That would ensure that also the ensemble (3.17) will be corrected properly. Consider that the state $E_j|\psi\rangle$ is result of the noise. If E_j influences only the k-th qubit then it can be expanded as

$$E_j = e_{j0}I + e_{j1}X_k + e_{j2}Z_k - ie_{j3}Y_k, \tag{3.19}$$

for some complex numbers e_{jl} (see equation (1.6)). Using the identity $Y_k = iX_kZ_k$ the equation (3.19) can be rewritten as

$$E_j = e_{j0}I + e_{j1}X_k + e_{j2}Z_k + e_{j3}X_kZ_k \tag{3.20}$$

If only the k-th qubit is changed, the system of nine qubits resides in the state:

$$|\psi_e\rangle = \frac{1}{c}(e_{j0}|\psi\rangle + e_{j1}X_k|\psi\rangle + e_{j2}Z_k|\psi\rangle + e_{j3}X_kZ_k|\psi\rangle), \tag{3.21}$$

where $c = \sqrt{\sum_{l=0}^{4}|e_{jl}|^2}$ is a normalization factor. As we can see, the state $|\psi_e\rangle$ is superposition of states corresponding to following situations: no error, phase flip, bit flip or both phase and bit flip on k-th qubit occurred. The error detection will project this superposition to one of the subspaces where just one of these four cases occur. The error-detection measurement for Shor code consists of four measurements. First three measurements measure triples of qubits to detect bit flip errors in each triple. For the next analysis, without loss of generality we suppose that $k \in \{1,2,3\}$. The first measurement involves the first three qubits and has four measurement components defined in (3.8)-(3.11). The outcome of this measurement is

0 with probability $\frac{|e_{j0}|^2+|e_{j2}|^2}{c^2}$ and state collapses to $|\psi_{e0}\rangle = \frac{e_{j0}|\psi\rangle+e_{j2}Z_k|\psi\rangle}{\sqrt{|e_{j0}|^2+|e_{j2}|^2}}$,
k with probability $\frac{|e_{j1}|^2+|e_{j3}|^2}{c^2}$ and state collapse to $|\psi_{ek}\rangle = \frac{e_{j1}X_k|\psi\rangle+e_{j3}X_kZ_k|\psi\rangle}{\sqrt{|e_{j1}|^2+|e_{j3}|^2}}$.

Results different from 0 and k are not possible in first measurement. If the outcome k is obtained, then we perform operator X_k on $|\psi_{ek}\rangle$ obtaining

$$|\psi_{ek}\rangle \rightarrow \frac{e_{j1}|\psi\rangle + e_{j3}Z_k|\psi\rangle}{\sqrt{|e_{j1}|^2 + |e_{j3}|^2}}, \tag{3.22}$$

since $X_k^2 = I$. If the outcome of first measurement is 0 we do not need to do anything for now. The measurements performed on qubits (4,5,6) or (7,8,9) respectively will certainly end up with results 0 and they do not change the state of system being measured. So after first three error-correction measurements the system is either in state $|\psi_{e0}\rangle$ or $|\psi_{ek}\rangle$ transformed according to (3.22), which are principally in the same form. Without loss of generality we can suppose that result k occurred in first measurement and state of nine qubits is $|\psi_e\rangle = \frac{e_{j1}|\psi\rangle + e_{j2}Z_k|\psi\rangle}{\sqrt{|e_{j1}|^2+|e_{j2}|^2}}$. That means that bit flip on $|\psi\rangle$ was corrected properly, what remains uncorrected is possible phase flip on k-th qubit. To detect phase flip, when we are sure that bit-flips were already corrected we perform final measurement with the measurement components described in equations (3.23) - (3.26). To make measurement description more understandable we label following states of three successive qubits $|000\rangle + |111\rangle$ as $|0_3\rangle$ and $|000\rangle - |111\rangle$ as $|1_3\rangle$.

$$P_0 = |0_3 0_3 0_3\rangle\langle 0_3 0_3 0_3| + |1_3 1_3 1_3\rangle\langle 1_3 1_3 1_3| \tag{3.23}$$

$$P_1 = |1_3 0_3 0_3\rangle\langle 1_3 0_3 0_3| + |0_3 1_3 1_3\rangle\langle 0_3 1_3 1_3| \tag{3.24}$$

$$P_2 = |0_3 1_3 0_3\rangle\langle 0_3 1_3 0_3| + |1_3 0_3 1_3\rangle\langle 1_3 0_3 1_3| \tag{3.25}$$

$$P_3 = |0_3 0_3 1_3\rangle\langle 0_3 0_3 1_3| + |1_3 1_3 0_3\rangle\langle 1_3 1_3 0_3| \tag{3.26}$$

The outcome from this measurement in our case can be only 0 or 1 (we consider $k \in \{1,2,3\}$). If $k \in \{4,5,6\}$ ($k \in \{7,8,9\}$) the possible outcome would be $\{0,2\}$ ($\{0,4\}$). The probabilities of measurements outcomes and final states of the system after the measurements in our case are:

$$\begin{aligned} p(0) &= \frac{|e_{j1}|^2}{|e_{j1}|^2 + |e_{j3}|^2} \text{ and the system collapse to the state } |\psi\rangle \\ p(1) &= \frac{|e_{j3}|^2}{|e_{j1}|^2 + |e_{j3}|^2} \text{ and the system collapse to the state } Z_k|\psi\rangle \end{aligned} \tag{3.27}$$

When the outcome of final measurement is 0 no other recovery is needed, if the outcome is 1 we perform one of the operations Z_1, Z_2 or Z_3 on the state $Z_k|\psi\rangle$ as they all take the system to the state $|\psi\rangle$. So we accomplished the task of recovering from arbitrary operation element E_i (acting on single qubit). In the same way we recover from whole operation $\mathcal{E}$. Shor code would work perfectly in environment where at most the single qubit error can occur. The strong precondition of our analysis was that the operation element of the noise $\mathcal{E}$ are acting on single qubit[2].

[2] All elements E_i must act on same qubit.

3.2.2 Calderbank-Shor-Steane Codes

The class of codes which construction is based on properties of classical linear codes are so known *Calderbank-Shor-Steane codes* (*CSS codes*) [26, 27]. For the construction of particular *CSS* code we need two classical linear codes C_1, C_2 with following properties. Suppose C_1 and C_2 are [n,k_1] and [n,k_2] codes respectively. Moreover, let $C_2 \subset C_1$ and both C_1 and $C_2^{\perp}$ correct t errors. Then we can construct [n,$k_1 - k_2$] quantum code $C_q = CSS(C_1, C_2)$ capable of correcting arbitrary error acting on up to t qubits. The codewords of C_q correspond to cosets of C_2 in C_1. To particular codeword x from C_1 corresponds following quantum codeword from C_q:

$$|x + C_2\rangle \equiv \frac{1}{\sqrt{|C_2|}} \sum_{y \in C_2} |x \oplus y\rangle \,, \tag{3.28}$$

where $\oplus$ (in $x \oplus y$) means bitwise addition of binary vectors x and y. Suppose $x, x' \in C_1$, such that $x \oplus x' \in C_2$, what means that x, x' belong to same coset of C_2 in C_1. Then $|x + C_2\rangle$ and $|x' + C_2\rangle$ are same states since

$$\sum_{y \in C_2} |x \oplus y\rangle = \sum_{y \in C_2} |x \oplus x' \oplus x' \oplus y\rangle = \sum_{y \in C_2} |x' \oplus (x \oplus x' \oplus y)\rangle = \sum_{z \in C_2} |x' \oplus z\rangle \,. \tag{3.29}$$

In addition if $x \oplus x' \notin C_2$ then inner product of $|x + C_2\rangle$ and $|x' + C_2\rangle$ is zero[3]. Therefore C_q has exactly $\frac{|C_1|}{|C_2|}$ different codewords and is [[n,$k_1 - k_2$]] quantum code[4]. We take arbitrary $2^{k_1 - k_2}$ different vectors $\{x_i\}$ $x_i \in C_1$, such that for all $i \neq j$ x_i and x_j belong to different cosets of C_1/C_2. We define coding function $\mathcal{C} : H \mapsto C_q$, where H is $2^{k_1 - k_2}$ dimensional Hilbert space corresponding to $k_1 - k_2$ information qubits, as

$$\mathcal{C}(|i\rangle) = |x_i + C_2\rangle \,. \tag{3.30}$$

More general, for any $|\psi\rangle \in H$, $|\psi\rangle = \sum_{i=0}^{2^{k_1 - k_2}} c_i \, |i\rangle$ the coding is

$$\mathcal{C}(|\psi\rangle) \equiv |\psi_q\rangle = \sum_{i=0}^{2^{k_1 - k_2}} c_i \, |x_i + C_2\rangle \,. \tag{3.31}$$

The error correction procedure using code C_q is based on properties of linear codes C_1, C_2. We show how C_q protects against error which can be described by at most t bit flips and t phase flips. Suppose that binary vector b describes positions[5] where bit flips occurred and binary vector p describes positions where phase flips occurred during

[3] And thus the states $|x + C_2\rangle$ and $|x' + C_2\rangle$ are recognizable by quantum measurement.

[4] Notation [[n,k]] stands for a quantum code which uses n-physical qubits to encode k-logical qubits.

[5] The vector b has 1s on the positions where bit flip occurred and 0s elsewhere.

transmission. Suppose that initially we had encoded state $|\psi\rangle \xrightarrow{C} |\psi_q\rangle$ defined by (3.31). After the error occurred the encoded state changed to

$$\sum_{i=0}^{2^{k_1-k_2}} \frac{c_i}{\sqrt{|C_2|}} \sum_{y\in C_2} (-1)^{(x_i\oplus y)p} |x_i \oplus y \oplus b\rangle . \tag{3.32}$$

At first we correct the bit flips. To perform this correction we need a circuit performing operation $|x_i\rangle |0\rangle \to |x_i\rangle |H_1(x_i)\rangle$, where H_1 is a parity check matrix of code C_1. Such circuit can be constructed straightforwardly from the given matrix H_1. It is shown in section 3.4. Note that for representing $|H_1(x_i)\rangle$ one needs $n-k_1$ ancillary qubits[6]. Using that H_1 is parity check matrix for code C_1 and $x_i \oplus y \in C_1$ we get $H_1(x_i \oplus y \oplus b) = H_1(x_i \oplus y) + H_1(b) = H_1(b)$. Therefore the application of given procedure leads to

$$\sum_{i=0}^{2^{k_1-k_2}} \frac{c_i}{\sqrt{|C_2|}} \sum_{y\in C_2} (-1)^{(x_i\oplus y)p} |x_i \oplus y \oplus b\rangle |H_1(b)\rangle = \left(\sum_{i=0}^{2^{k_1-k_2}} \frac{c_i}{\sqrt{|C_2|}} \sum_{y\in C_2} (-1)^{(x_i\oplus y)p} |x_i \oplus y \oplus b\rangle \right) \otimes |H_1(b)\rangle . \tag{3.33}$$

By measuring $n-k_1$ ancilla qubits we get the result $|H_1(b)\rangle$. If the vector b has Hamming weight t or less we correctly determine b from measured syndrome $H_1(b)$. Then we simply flip (by means of NOT gates) the corrupted bits back to the state

$$\sum_{i=0}^{2^{k_1-k_2}} \frac{c_i}{\sqrt{|C_2|}} \sum_{y\in C_2} (-1)^{(x_i\oplus y)p} |x_i \oplus y\rangle . \tag{3.34}$$

After correcting bit-flips errors we apply Hadamard gate to each qubit, which allows us to correct phase flips in similar way as the bit flips were corrected. The state (3.34) after application of Hadamard gates changes to

$$\sum_{i=0}^{2^{k_1-k_2}} \frac{c_i}{\sqrt{|C_2|}} \sum_{y\in C_2} \frac{1}{\sqrt{2^n}} \sum_{z\in\{0,1\}^n} (-1)^{(x_i\oplus y)p+(x_i\oplus y)z} |z\rangle . \tag{3.35}$$

This state may be rewritten by substituting $z' \equiv z \oplus p$ and then by changing the order of summation to:

$$\sum_{i=0}^{2^{k_1-k_2}} \frac{c_i}{\sqrt{|C_2|2^n}} \sum_{z'\in\{0,1\}^n} \sum_{y\in C_2} (-1)^{(x_i\oplus y)z'} |z' \oplus p\rangle . \tag{3.36}$$

[6] Ancillary qubit in the concept of quantum computing is often called *ancilla*.

It can be shown that for $z' \in C_2^{\perp}$ the $\sum_{y \in C_2} (-1)^{yz'} = |C_2|$, while if $z' \notin C_2^{\perp}$ then $\sum_{y \in C_2} (-1)^{yz'} = 0$. For interested readers the proofs of these statements are presented in the appendix as lemma B.1. Thus the state (3.36) may be rewritten as:

$$\sum_{i=0}^{2^{k_1-k_2}} \frac{c_i}{\sqrt{2^n/|C_2|}} \sum_{z' \in C_2^{\perp}} (-1)^{x_i z'} |z' \oplus p\rangle . \tag{3.37}$$

This notation has the same structure as the state (3.32)[7]. Therefore we act analogically: We couple k_2 ancilla to this state:

$$\left(\sum_{i=0}^{2^{k_1-k_2}} \frac{c_i}{\sqrt{2^n/|C_2|}} \sum_{z' \in C_2^{\perp}} (-1)^{x_i z'} |z' \oplus p\rangle \right) \otimes |H_2(p)\rangle . \tag{3.38}$$

Again, measuring k_2 ancilla qubits gives us information on which qubits phase flip occurred (if we assume that at most t qubits were phase flipped). Applying bit flip on detected positions we change the state to

$$\sum_{i=0}^{2^{k_1-k_2}} \frac{c_i}{\sqrt{2^n/|C_2|}} \sum_{z' \in C_2^{\perp}} (-1)^{x_i z'} |z'\rangle = \sum_{i=0}^{2^{k_1-k_2}} \frac{c_i}{\sqrt{|C_2|2^n}} \sum_{z \in \{0,1\}^n} \sum_{y \in C_2} (-1)^{(x_i \oplus y) z} |z\rangle . \tag{3.39}$$

The process of error-correction is ended by re-applying Hadamard gates to each qubit. Since the Hadamard gate is self inverse the operation results in the initial state (3.34) without phase flip errors:

$$\sum_{i=0}^{2^{k_1-k_2}} \frac{c_i}{|C_2|} \sum_{y \in C_2} (-1)^{(x_i \oplus y)} |x_i \oplus y\rangle = \sum_{i=0}^{2^{k_1-k_2}} c_i |x_i + C_2\rangle = |\psi_q\rangle . \tag{3.40}$$

We have shown that particular error described by at most t bit flip and t phase flip errors can be corrected. In reality more complicated errors described by error operator $\mathcal{E}$ can occur. We can correct errors which operator sum representation $\{E_i\}$ contains just operation elements E_i which can be written as combination of operators that acts on up to t qubits. Formally, by usage of code C_q we can protect the system against errors described by quantum operation $\mathcal{E}$ with operation elements

$$E_i = \sum_i \alpha_i X_{a_{i1}} \dots X_{a_{im_i}} Z_{b_{i1}} \dots Z_{b_{in_i}}, \tag{3.41}$$

where there exist sets B, P: $B, P \subset \{1, \dots, n\}$, $|B| \leq t, |P| \leq t$ and all $a_{ij} \in B$, $b_{ij} \in P$. These types of errors are processed precisely in the same way as in the above

[7] With the difference of used codes. In formula (3.32) we have used correcting properties of C_1 and now we use the code $C_2^{\perp}$

mentioned case. The only difference in analysis occurs in computing of syndromes corresponding to matrices H_1 and H_2 in (3.33) and (3.38). In this more general case the output of measurement of ancilla qubits is not deterministic and the output of syndrome measurements will cause a collapse of occurred error to the one consistent with it. After measuring of H_1 only components from (3.41) which X operators leads to syndrome measured by H_1 will remain. Then after applying H_2 in (3.38) the error will collapse to the error described by terms from superposition (3.41) which are consistent with measured syndromes (as shown in detail for Shor code). The error is corrected exactly in the same way as shown in (3.39). It is really fascinating property of quantum codes that despite of the fact that there is uncountable amount of possible errors, it is sufficient to find a correction process just for bit flip and phase flip and all other errors acting on limited number of qubits can be corrected.

3.2.3 CSS Code Correcting One Error

The most known example of CSS codes is *Steane code*. It is a [[7,1,3]] quantum code. Its construction is based on C_H=[7,4,3] classical Hamming code. Classical codes C_1 and C_2, in definition of CSS codes are chosen as $C_1 = C_H$ and $C_2 = C_H^{\perp}$. We know that both C_1 and $C_2^{\perp}$ correct one error. We need to check whether $C_2 \subset C_1$. The generator matrix of C_H is

$$G(C_H) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \tag{3.42}$$

One of possible generator matrix of C_2 is

$$G(C_H^{\perp}) = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.43}$$

as the columns of $G(C_H^{\perp})$ are linearly independent and orthogonal to the columns of $G(C_H)$. One can check that

$$\begin{aligned} G(C_H^{\perp})_1 &= G(C_H)_2 + G(C_H)_3 + G(C_H)_4 \\ G(C_H^{\perp})_2 &= G(C_H)_1 + G(C_H)_3 + G(C_H)_4, \\ G(C_H^{\perp})_3 &= G(C_H)_1 + G(C_H)_2 + G(C_H)_4 \end{aligned} \tag{3.44}$$

where $G(C_H^{\perp})_i$ and $G(C_H)_i$ stand for i-th column of matrix $G(C_H^{\perp})$ and $G(C_H)$ respectively. Therefore every vector from $C_H^{\perp}$ can be written as linear combination of vectors from C, thus $C_H^{\perp} \subset C_H$.

Steane code has two different codewords. First of them is $|0_L\rangle = \left|0000000 + C_H^{\perp}\right\rangle$. Second codeword $|1_L\rangle$ can be determined by finding an x such that $x \in C_H$ and $x \notin C_H^{\perp}$. The vector (1,1,1,1,1,1,1) satisfies these conditions since it belongs to C_H, because $G[C_H](1,1,1,1)^T = (1,1,1,1,1,1,1)$ and it is not orthogonal to itself so does not belong to $C_H^{\perp}$. Therefore $|1_L\rangle = \left|1111111 + C_H^{\perp}\right\rangle$. Writing $|0_L\rangle$, $|1_L\rangle$ explicitly yields:

$$\begin{aligned} |0_L\rangle = \frac{1}{\sqrt{8}} \quad [\quad & |0000000\rangle + |1010101\rangle + |0110011\rangle + |1100110\rangle \\ + \quad & |0001111\rangle + |1011010\rangle + |0111100\rangle + |1101001\rangle] \end{aligned} \tag{3.45}$$

$$\begin{aligned} |1_L\rangle = \frac{1}{\sqrt{8}} \quad [\quad & |1111111\rangle + |0101010\rangle + |1001100\rangle + |0011001\rangle \\ + \quad & |1110000\rangle + |0100101\rangle + |1000011\rangle + |0010110\rangle] \end{aligned} \tag{3.46}$$

3.3 New CSS Codes

Steane code is correcting arbitrary error which influences a single qubits. If we want to protect our quantum information against two qubit errors, more encoding qubits are needed. The construction using a CSS codes is not the most efficient, since there exists codes which can protect against the same errors using fewer coding qubits. The five qubit code which protects against single-qubit errors is presented in section 4.4.2. However, CSS codes are much easier used in fault tolerant quantum computation. In [29] is proved that the necessary and sufficient condition to implement CNOT operator fault tolerantly is the usage of CSS codes. These are the reasons, why we have searched for new CSS codes.

Steane code uses 7 qubits and protects against single-qubit errors. [[23,1,7]] CSS code may be obtained from weakly self-dual Golay code. These are the examples of small CSS codes, but none CSS code correcting two-qubits errors has been established among scientists working in the area of *QC* yet. Therefore, we have tried to find new CSS code correcting two errors. The detailed analysis of fault tolerant implementation of found code is presented in the chapter 6.

3.3.1 Searching for the Code

We aim our effort to find a code which encodes just one logical qubit since such codes are more suitable [30] for fault tolerant computation, described in chapter 5. To construct [[n,1,5]] CSS code correcting two errors we need to find [n,k] linear code C_1 correcting two errors and [n,$k-1$] linear code C_2 which is subset of C_1. Moreover, $C_2^{\perp}$ which is $[n, n-k+1]$ code must also correct two errors.

An comprehensive overview on known upper and lower bounds on CSS Codes and some searching methods were presented by Steane in [31]. We propose significantly different approach to codes search. The code is most probable to be found when $k_{max} \equiv \max(k, n-k+1)$ is minimal. The minimal k_{max} for fixed odd n is reached at $k = \frac{n+1}{2}$. From lower and upper bounds presented in [31] follows that the CSS code correcting two errors needs to have at least 17 data qubits. In the same paper was mentioned that [[19,1,5]] qubit should exists, but the code was not given. We have developed probabilistic search algorithm, by which we have tried to find either 17, 18 or 19 data qubits code. The main idea of our algorithm is based on corollary 2.8.

The idea is following: To construct [[n,1,5]] code we need to find classical linear codes C_1, C_2 with following parameters:

$$C_1 = [n, k, 5] \tag{3.47}$$

$$C_2^{\perp} = [n, n-k+1, 5] \tag{3.48}$$

such that $C_2 \subset C_1$. To find $[n, k, 5]$ using the corollary 2.8 we need to find parity check matrix H_1 with dimensions $n-k \times n$ where all four elements subsets of matrix columns are linearly independent. To satisfy condition (3.48) we need to construct $k-1 \times n$ parity check matrix $H_2^{\perp}$, where again no four columns of $H_2^{\perp}$ are linearly dependent. The more rows of particular matrix, the easier such matrix can be found. Therefore we want to maximize $\min(n-k, k-1)$. For odd n the maximum is reached at $k = \frac{n+1}{2}$, for even n the maximum is reached at either $k_1 = n/2$ or $k_2 = (n+2)/2$. We have used following algorithm for these optimal values of k.

Basic Parity Check Matrix Search Algorithm

The algorithm starts with random matrix H_1. Then all four element subsets of H_1 columns are repeatedly checked for independence. If the linearly dependent subset S_d is found, the algorithm tries to locate a column r among columns of S_d with the following property. There exists a vector v', which differ from vector of r just in few positions and if v' would substitute the column r, all subsets of four or less columns, which contain column r are linearly independent. If such a column r and vector v' are

```
Start with random H in systematic form
{
 MatrixFound=true
 For all column subsets S_i (|S_i|<=4)
 {
     if (S_i is linearly dependent)
     {
         MatrixFound=false
         r,v = Find a column r of S_i, which alternation v won't make
               any other subset dependent and will broke dependence
               of S_i; r must not be from identity part of H
         if (r==null)
              Change random column of S_i,
              leaving H in systematic form
          else
               substitute v to r
      }
  }
}while(MatrixFound==false)
```

Tab. 3.1: Probabilistic algorithm searching for [n,k,5] classical code.

found the v' is substituted to the column r. If such a column does not exists we simply alternate one of S_d's columns to break the linear dependence of S_d. Of course some other dependence will occur in the second case, but we will deal with it in the next repetition of dependency check. The algorithm is shown in table 3.1.

Systematic Form Improvement

The algorithm may be improved in such a way that we search for the code in the systematic form. We know (see section 2.2.3) that each code is equivalent with some systematic code and thus we do not lose any generality. But we significantly reduce the search space of the algorithm, since first $n-k$ columns of the matrix are fixed to form an identity matrix. The previous algorithm was able to find several [19,10,5] codes, but no single [17,9,5] code was found without this improvement. The improved algorithm can find [17,9,5] in few hundreds of checking repetitions.

Algorithms for finding C_1 and C_2

Algorithm 3.1 may be used to find separately matrices H_1 and $H_2^{\perp}$, but we must comply to the condition: $C_2 \subset C_1$. Two different approaches were tried:

Self Duality Method

At first we search just for H_1 using algorithm 3.1 and then verify whether the obtained code C_1 complies the condition

$$C_1^{\perp} \subset C_1. \tag{3.49}$$

If this would be the true, we have done. We can pick $C_2 \equiv C_1^{\perp}$. If the condition (3.49) does not hold we start with other random check matrix H_1. This approach has not provided any result, since the condition (3.49) holds very rare.

Random Subsets Method

We have realized that there are plenty of [19,10,5] codes and in reasonably running time such a code may be found simply by picking up a random parity check matrix. The idea of constructing CSS code is thus following. At first, using algorithm 3.1 we find code C_1. Then we construct several (≈ 1000) random subsets of C_1, with dimension one less (9) than dimension of C_1. To each random subset C_i^s we construct its dual code $C_i^{s\perp}$ and verify either $C_i^{s\perp}$ does fulfill columns independence condition. If none of $C_i^{s\perp}$ fulfill the condition we start with a new random matrix H_1.

We found [[19,1,5]] code using this approach to the code search. The algorithm ran approximately 40 minutes on 550MHz CPU until the code has been found. There is probably many more [[19,1,5]] codes, which may be found using this approach. However, none of our algorithms was able to find smaller quantum code correcting two errors. For $n = 17$, we can find enough [17,9,5] codes, but none of them comply with condition (3.49), neither the method of random subsets have not been successful for them. The reason why random subset method have not been successful for $n = 17$ is, that there is much less [17,9,5] codes than [19,10,5] codes. Therefore it is quite improbable finding a [17,9,5] code randomly as is required in second part of the method. Even none [17,9,5] code have been found using the algorithm 3.1 without systematic form improvement. For $n = 18$ the problem is opposite than for the case of $n = 17$. There is plenty of [18,9,5] codes and it is possible to find such codes just by random guessing in reasonable time. The problem for $n = 18$ is in the first part of the algorithm. Algorithm 3.1 even with several minor improvements was not able to find any [18,10,5] code.

$$H_1 = \begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&1&1&1&0&0&0&0&1&0\\
0&1&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&1\\
0&0&1&0&0&0&0&0&0&1&0&1&0&1&0&1&1&1&0\\
0&0&0&1&0&0&0&0&0&0&1&0&1&1&1&0&1&0&1\\
0&0&0&0&1&0&0&0&0&1&0&1&1&1&1&0&1&0&0\\
0&0&0&0&0&1&0&0&0&1&0&0&0&0&0&0&1&1&1\\
0&0&0&0&0&0&1&0&0&0&0&0&1&1&0&1&1&1&0\\
0&0&0&0&0&0&0&1&0&1&1&1&0&1&1&1&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&1&1&0&1&1&1&1&1
\end{bmatrix}$$

Tab. 3.2: Parity check matrix H_1 of found [[19,1,5]] code.

$$H_2^{\perp} = \begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&0&1&1\\
0&1&0&0&0&0&0&0&0&1&0&0&1&1&0&0&1&1&1\\
0&0&1&0&0&0&0&0&0&1&1&1&1&0&0&1&1&1&1\\
0&0&0&1&0&0&0&0&0&1&0&0&1&1&1&1&0&1&0\\
0&0&0&0&1&0&0&0&0&0&1&0&0&1&1&1&1&1&0\\
0&0&0&0&0&1&0&0&0&1&0&1&0&1&0&1&1&1&0\\
0&0&0&0&0&0&1&0&0&0&0&1&0&0&1&0&0&1&1\\
0&0&0&0&0&0&0&1&0&0&1&1&1&1&0&0&1&1&0\\
0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&1&0&1&1
\end{bmatrix}$$

Tab. 3.3: Parity check matrix $H_2^{\perp}$ of found [[19,1,5]] code.

Summary

The proposed algorithm found [[19,1,5]] code and thus proved Steane's claim in [31]. Parity check matrices of found code are shown in tables 3.2 and 3.3. The existence of either 17 or 18 qubits CSS codes correcting two errors remains an open question.

3.4 Syndrome Measurement Circuits

In this section we will show how error-correcting quantum circuits may be constructed directly from the parity check matrices H_1 and $H_2^{\perp}$. The encoding and decoding circuits are presented in section 4.5.

According to section 3.2.2 we need to compute syndromes $H_1 |c + e\rangle$ or respectively syndrome $H_2^{\perp} |c' + e\rangle$. To compute the desired syndrome we prepare special ancillary qubits for each row of particular parity check matrix. To measure X error syndrome

we apply CNOT gate on j-th ancilla qubit and i-th data qubit, for each $H_1[j,i] = 1$. Ancilla qubits act as targets of CNOT gates. After all CNOT gates are applied, each ancilla qubit is measured in standard basis set to obtain syndrome bits. To obtain syndrome bits of Z errors, at first Hadamard gate to each data qubit must be applied. Then similarly, CNOTs on j-th ancilla bit and i-th data qubit are performed, for each $H_2^{\perp}[j,i] = 1$ and ancilla bits are measured. At the end of Z errors correction procedure Hadamard gates are re-applied to the data qubits. Circuits for found [[19,1,5]] code are shown on figures 3.2 and 3.3.

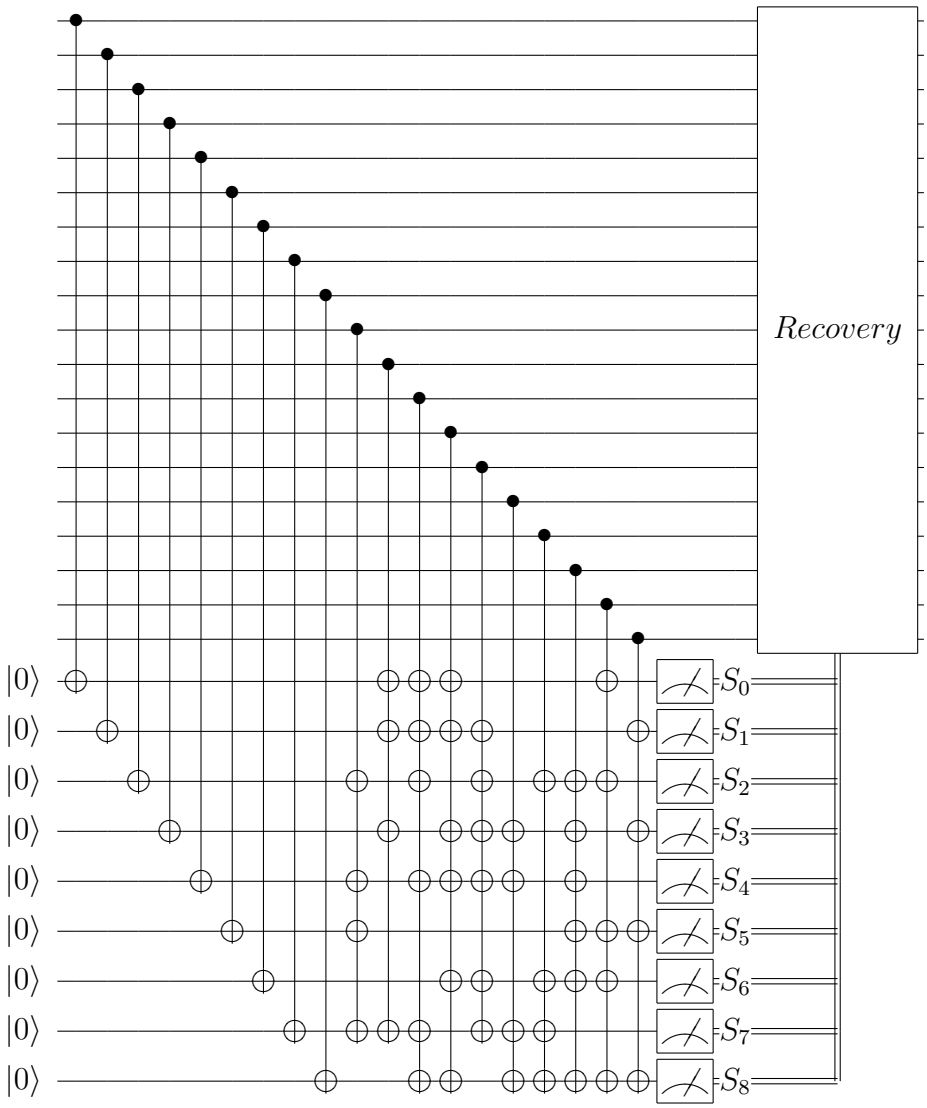


Fig. 3.2: Circuit correcting X errors of qubit encoded in [[19,1,5]] code.

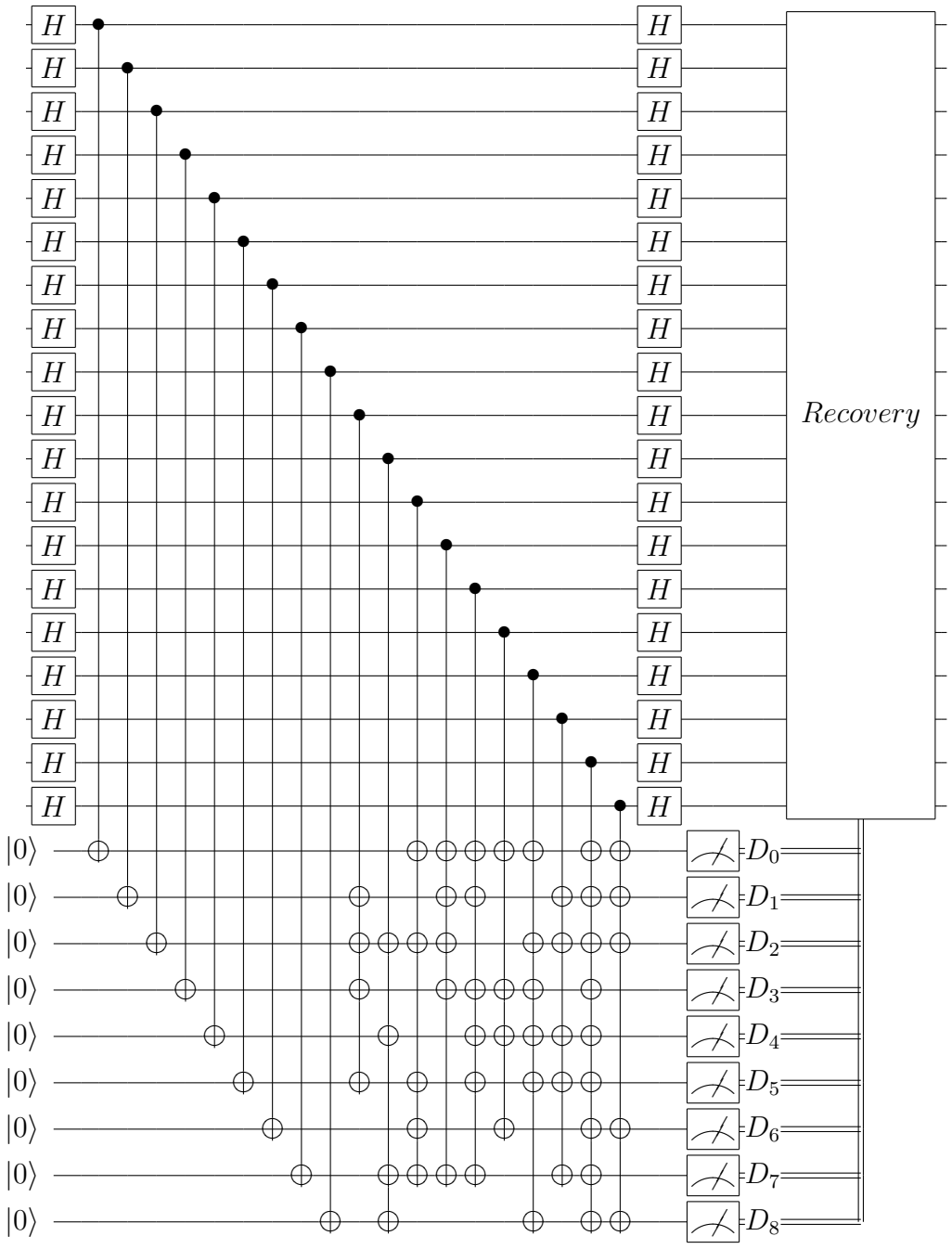


Fig. 3.3: Circuit correcting Z errors of qubit encoded in [[19,1,5]] code.

4. STABILIZER CODES

The important class of quantum codes are *stabilizer codes*. The stabilizer codes were invented by Gottesman [32]. These codes are useful for building quantum fault tolerant circuits. Many quantum codes, as Shor code or CSS codes can be described in Stabilizer formalism. Moreover, the stabilizer formalism provides more compact description of the codes than already presented enumeration of code words, especially for larger codes. To properly understand stabilizer codes we first introduce stabilizer formalism. It provides powerful method for understanding a wide class of operations in quantum computation. It enables us to build compact coding and decoding circuits for [[19,1,5]] code. Methods for implementation of quantum operations on the data encoded in *ECC*s are presented.

4.1 The Formalism of Stabilizer Codes

The fundamental element of stabilizer formalism is the Pauli group G_n on n qubits. The Pauli group for one qubit is defined to consists of all Pauli matrices, together with multiplicative factors $\pm 1, \pm i$:

$$G_1 = \{\pm I, \pm iI, \pm X, \pm iX, \pm Y, \pm iY, \pm Z, \pm iZ\} \tag{4.1}$$

The set of matrices G_1 forms a group under the operation of matrix multiplication. In general, group G_n consists of all tensor products of Pauli matrices on n qubits again with multiplicative factors $\pm 1, \pm i$.

The subgroups of G_n can define a stabilizer code. As an example consider the Shor code. Both vectors $|0_L\rangle$, $|1_L\rangle$ and also their arbitrary superposition $\alpha |0_L\rangle + \beta |0_L\rangle$ are stabilized by all operators M_1 through M_8 from table 4.1. The fact that an operator M_i stabilizes a state $|\psi\rangle$ can be formally expressed as

$$M_i |\psi\rangle = |\psi\rangle . \tag{4.2}$$

Operators M_1 through M_8 are not all operators that stabilize Shor code. For example the operator $M = Z \otimes I \otimes Z \otimes I \otimes I \otimes I \otimes I \otimes I \otimes I = Z_1 Z_3$ also stabilizes all codewords from Shor code. We can get the operator M from operators $M_1, \ldots, M_8$ as $M = M_1 M_3$. All operators that stabilize the Shor code constitute a subgroup S of G_n and the operators

M_1 through M_8 constitute a basis set of S. Moreover, it can be shown that codewords of Shor code are the only vectors stabilized by subgroup $\langle M_1, M_2, \ldots, M_8 \rangle$. We now define stabilizer codes more precisely:

Definition 4.1. Let S be a subgroup of G_n. Define V_S to be the set of all n qubits states stabilized by every element of S. The V_S is the vector space stabilized by S and the S is said to be the stabilizer of V_S.

The V_S corresponding to stabilizer S is a subspace of n-qubits state space[1]. To show this, we need to verify that for all $|\psi_1\rangle, |\psi_2\rangle \in V_S$ and $a_1, a_2 \in \mathcal{C}$ holds $a_1 |\psi_1\rangle + a_2 |\psi_2\rangle \in \mathcal{C}$. This is true because for all $s \in S$

$$s(a_1 |\psi_1\rangle + a_2 |\psi_2\rangle) = a_1 s(|\psi_1\rangle) + a_2 s(|\psi_2\rangle) = a_1 |\psi_1\rangle + a_2 |\psi_2\rangle . \tag{4.3}$$

Moreover, V_S is an intersection of subspaces fixed by particular operators from S. Not every stabilizer S stabilizes non-trivial vector subspace of n-qubits vector space. Two necessary conditions that must hold are

a) $-I \notin S$

b) all operators of S commute

Consider that a converse of a) or b) would hold. If there would be $-I \in S$ then for each $|\psi\rangle \in V_S$ must hold $-I |\psi\rangle = |\psi\rangle$ what is true only if $V_S = \emptyset$. Next suppose that there would be anticommuting[2] operators M, N in S, what means that $MN = -NM$. Then for all $|\psi\rangle \in S$ we get

$$MN |\psi\rangle = M |\psi\rangle = |\psi\rangle \tag{4.4}$$

$$NM |\psi\rangle = N |\psi\rangle = |\psi\rangle \tag{4.5}$$

Since $MN = -NM$ and on V_S also $MN = NM$, the V_S needs to be empty.

[1] The alternative notation for stabilizer code corresponding to stabilizer S is $C(S)$, $C(S) \equiv V_S$

[2] Arbitrary operators of G_n either commute or anticommute.

M_1	Z	Z	I	I	I	I	I	I	I
M_2	I	Z	Z	I	I	I	I	I	I
M_3	I	I	I	Z	Z	I	I	I	I
M_4	I	I	I	I	Z	Z	I	I	I
M_5	I	I	I	I	I	I	Z	Z	I
M_6	I	I	I	I	I	I	I	Z	Z
M_7	X	X	X	X	X	X	I	I	I
M_8	I	I	I	X	X	X	X	X	X

Tab. 4.1: The generator of Shor code stabilizer.

4.2 Generators of Stabilizer Codes

To specify a stabilizer S of a code V_S we do not need to enumerate all elements of S. It is sufficient to name the independent set of generators for S, or its basis set. However, determining either given set of generators is independent or not is time consuming without more sophisticated approach.

The concept of generators *check matrices* gives us clever tool for stabilizers analysis. Consider the stabilizer $S = \langle g_1, \ldots, g_l \rangle$. The check matrix corresponding to S is a $l \times 2n$ matrix whose rows correspond to the generators g_1 through g_l. The i-th row of the check matrix is constructed as follows: If g_i contains I on the j-th qubit then the matrix contains 0 in j-th and $n+j$-th columns. If g_i contains an X on the j-th qubit then the element in j-th column is 1 and in $n+j$-th column is 0. If it contains Z on j-th qubit then j-th column contains 0 and $n+j$-th element contains 1. And in the last, if g_i contains operator Y on j-th qubit then both j-th and $n+j$-th columns contain 1.

Example 4.2. The check matrix to the stabilizer generator from table 4.1 is:

$$\left[\begin{array}{cccccccccc|cccccccccc} 0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1\\ 1&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0 \end{array}\right] \tag{4.6}$$

Note that check matrix does not contain any information about overall multiplicative factor of g_i. We denote $r(g)$ to represent a row representation of operator g from check matrix, $r(g)$ is $2n$ elements binary row vector. Denote following $2n \times 2n$ matrix as Λ:

$$\Lambda = \begin{bmatrix} 0_{n\times n} & I_{n\times n} \\ I_{n\times n} & 0_{n\times n} \end{bmatrix}. \tag{4.7}$$

Following two lemmas help us to determine either given set of generators is independent and commuting.

Lemma 4.3. *Let g, h be operators from G_n. The g and h commute if and only if $r(g)\Lambda r(h)^T = 0$. Therefore the generators of stabilizer $S =< g_1, \ldots, g_l >$ with corresponding check matrix M commute if and only if $M\Lambda M^T = 0$.*

Lemma 4.4. *Let $S =< g_1, \ldots, g_l >$ be such that $-I$ is not an element of S. The generators $g_i, i \in \{1, \ldots, l\}$ are independent if and only if the rows of the corresponding check matrix are linearly independent.*

Previous lemmas give us useful mechanism for determining suitable generators of a stabilizer. Following proposition looks simple, but is surprisingly useful in error-correction procedure of stabilizer codes.

Proposition 4.5. *Let S be a stabilizer generated by l independent generators $\{g_1, \ldots, g_l\}$ and satisfies $-I \notin S$. Fix i in the range $1, \ldots, l$. Then there exists $g \in G_n$ such that $gg_ig^\dagger = -g_i$ and for all $j \neq i$ $gg_jg^\dagger = g_j$.*

From lemma 4.4 following proposition can be inferred:

Proposition 4.6. *Let $S =< g_1, \ldots, g_{n-k} >$ be a subgroup of G_n, such that $-I \notin S$ and g_1 through g_{n-k} are independent commuting generators. Then V_S is 2^k dimensional subspace of n-qubit state space.*

This can be intuitively seen as each operator from G_n has two eigen values ± 1. The state vectors from the code are eigen vectors with eigen value $+1$. Moreover, a subspace with dimension 2^{n-1} corresponds to each operator's eigen value. Since the codewords are $+1$ eigen vectors of all generators and generators are independent the each additional generator shrinks the dimension of code by factor $1/2$. Therefore the dimension is 2^{n-l}. The formal proofs of propositions 4.6 and 4.5 can be found in appendix.

4.2.1 Quantum Dynamics Using Stabilizer Formalism

Suppose that we have encoded data in the stabilizer code $C(S)$ with stabilizer S. The important question is how does the stabilizer evolve, if the codewords are subjected to unitary operator U_L. Suppose the logical quantum state $|\psi_u\rangle$ is encoded in the codeword $|\psi_c\rangle$ from code $C(S)$ and we would like to perform unitary operator U_L on encoded state $|\psi_u\rangle$. The one way how we can handle this task is to decode the codeword $|\psi_c\rangle$ back to state $|\psi_u\rangle$, perform the operator $|\psi_u\rangle \rightarrow U_L |\psi_u\rangle$ and encode the state $U_L |\psi_u\rangle$ back to the code $C(S)$:

$$U_L |\psi_u\rangle \overset{code}{\rightarrow} |\psi'_c\rangle \tag{4.8}$$

The alternative solution of this problem would be applying the operator U_L on encoded data. Suppose that there would be an operator U acting on codewords of $C(S)$ such that

$$U |\psi_c\rangle = |\psi'_c\rangle , \tag{4.9}$$

for all $|\psi_c\rangle \in C(S)$. Obviously the alternative method is theoretically[3] simpler and also it protects the data in encoded form for the whole time of computation. In the first

[3] It does not need to be straightforward to obtain U from U_L.

approach the data became unprotected for a while and a possible error during that time would be uncorrectable.

What is the stabilizer of the code after the unitary operator U acted on the code word? Let $|\psi\rangle$ be any element of V_S. Then for any element $g \in S$,

$$U|\psi\rangle = Ug|\psi\rangle = UgU^\dagger U|\psi\rangle, \tag{4.10}$$

thus the state $U|\psi\rangle$ is stabilized by operator $UgU^\dagger$. The following proposition follows from this analysis:

Proposition 4.7. *Let S be the stabilizer corresponding to state space V_S. Then the vector space $UV_S \equiv \{U|\psi\rangle \,|\, |\psi\rangle \in V_S\}$ is stabilized by the stabilizer $USU^\dagger \equiv \{UgU^\dagger | g \in S\}$. Moreover, if the S has the generators $\{g_1, \ldots, g_l\}$, then the generators of $USU^\dagger$ are $\{Ug_1U^\dagger, \ldots, Ug_lU^\dagger\}$.*

Example 4.8. Suppose we have a stabilizer for Steane code and that the Hadamard gate is applied to first qubit of codeword. Since

$$H_1 I_1 H_1^\dagger = I_1; \quad H_1 X_1 H_1^\dagger = Z_1; \quad H_1 Y_1 H_1^\dagger = -Y_1; \quad H_1 Z_1 H_1^\dagger = X_1, \tag{4.11}$$

the resulting state will be stabilized by the stabilizer which is obtained from the initial stabilizer by altering operators X_1, Y_1, Z_1 to operators $Z_1, -Y_1, X_1$ respectively.

Unfortunately, not all unitary operators U map elements of G_n to itself under conjugation. Important example of such gate is $\pi/8$ gate. The $\pi/8$ gate T is defined as:

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix} = e^{\frac{i\pi}{8}} \begin{bmatrix} e^{-\frac{i\pi}{8}} & 0 \\ 0 & e^{\frac{i\pi}{8}} \end{bmatrix} \tag{4.12}$$

We can easily calculate the effect of $\pi/8$ gate on Pauli matrices:

$$TZT^\dagger = Z; \qquad TXT^\dagger = \frac{X+Y}{\sqrt{2}}, \tag{4.13}$$

and therefore $\pi/8$ gates cannot be easily used with stabilizer codes. Denote by $N(G_n)$ the set of unitary matrices U, such that $UG_nU = G_n$. It was shown in [29, 1] that all matrices from $N(G_n)$ can be generated from Hadamard gate, phase[4] gate and CNOT gate.

[4] Phase gate is defined by matrix $\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$.

4.3 Correcting Errors in Stabilizer Codes

Suppose $C(S)$ is a stabilizer code with stabilizer S. We denote $N(S)$ a subset of G_n, which is defined to consists of all elements $E \in G_n$ such that $EgE^\dagger \in S$ for all $g \in S$. Following theorem specifies the correction power of $C(S)$.

Theorem 4.9. *Let S be the stabilizer for a stabilizer code $C(S)$. Suppose $\{E_j\}$ is a set of operators in G_n such that $E_j^\dagger E_k \notin N(S) - S$ for all j and k. Then $\{E_j\}$ is a correctable set of errors for the code $C(S)$.*

Proof of theorem 4.9 can be found in [1] (Theorem 10.8). The error correction procedure of $[[n, k]]$ stabilizer code with stabilizer generators $g_1, \ldots, g_{n-k}$ is following: Each generator g_1 through g_{n-k} is measured to obtain error syndromes $\beta_1, \ldots, \beta_{n-k}$, $\beta_i \in \{+1, -1\}$. If no error have occurred, then all $\beta_i = +1$. On the other hand, if some β_j is equal to -1 we need to change encoded state in such a way that it remains in the same eigen space of all β_i, $i \neq j$ and the eigen space of g_j is changed to $+1$ eigen space. This can be easily done using the result of proposition 4.5. In chapter 3 we have shown different error-correction procedure for CSS codes and thus this schema is new correction alternative for CSS codes.

4.4 Construction of Stabilizer Codes

In this section we show how the stabilizer formalism can be used for construction of quantum codes. The stabilizers for CSS codes will be presented. Also stabilizer for 5-qubit code will be shown. It is the smallest possible code correcting arbitrary errors on single qubit.

4.4.1 Standard Form of Stabilizer Codes

There are more possible ways how to choose generators of stabilizer S. So called *standard form* of a stabilizer code is an useful tool for further analysis. It can be shown that check matrix of stabilizer code $C(S)$ can be chosen in the form

$$\begin{array}{r} \\ r\{ \\ n-k-r\{ \end{array} \left[\begin{array}{ccc|ccc} \overbrace{I}^{r} & \overbrace{A_1}^{n-k-r} & \overbrace{A_2}^{k} & \overbrace{B}^{r} & \overbrace{0}^{n-k-r} & \overbrace{C}^{k} \\ 0 & 0 & 0 & D & I & E \end{array}\right], \tag{4.14}$$

where r is a rank of the X part of the check matrix. More details about standard stabilizer form may be found in [29, 1].

4.4.2 Logical Operators for Stabilizer Codes

Suppose we have a stabilizer S such that $-I \notin S$ and S has $n-k$ independent and commuting generators $g_1, \ldots, g_{n-k}$. The vector space V_S consists of 2^k different vectors. We would like to pick up the k elements basis sets for logical encoded qubits. The one possible way how to do it is following: we choose operators $\bar{Z}_1, \ldots, \bar{Z}_k \in G_n$ such that $g_1, \ldots, g_{n-k}, \bar{Z}_1, \ldots, \bar{Z}_k$ forms independent and commuting set. The $\bar{Z}_j$ operator plays the role of a logical Pauli Z operator on j-th logical qubit. Recall that the state $|0\rangle$ is eigenvector of Z with eigenvalue $+1$ and the state $|1\rangle$ is eigenvector of $-Z$ also with eigenvalue $+1$. Therefore Z stabilizes state $|0\rangle$ and $-Z$ stabilizes state $|1\rangle$. So the logical computational basis state $|x_1, \ldots, x_k\rangle_L$ is therefore defined as the state with stabilizer

$$\left\langle g_1, \ldots, g_{n-k}, (-1)^{x_1}\bar{Z}_1, \ldots, (-1)^{x_k}\bar{Z}_k \right\rangle. \tag{4.15}$$

If the operators $g_1, \ldots, g_{n-k}$ are given in standard form (4.14),then the check matrix corresponding to the k encoded $\bar{Z}$ operators can be defined as

$$G_z = [\overbrace{0}^{r}\ \overbrace{0}^{n-k-r}\ \overbrace{0}^{k}\ |\ \overbrace{A_2^T}^{r}\ \overbrace{0}^{n-k-r}\ \overbrace{I}^{k}]. \tag{4.16}$$

It is clear that these $\bar{Z}$ operators commute with each other since they consist only from Z operators. The commutativity of $\bar{Z}$ operators with generators (4.14) can be checked using the slight modification of lemma 4.3. The commutativity follows from the equation:

$$I.(A_2^T)^T + A_2.I = A_2 + A_2 = 0 \tag{4.17}$$

Since no X operators appear in $\bar{Z}$, the independence of $\bar{Z}$ from first r generators of stabilizer (4.14) is obvious. The independence from the set of remaining $n-k-r$ generators follows from the appearing of identity matrix in the check matrix for those $n-k-r$ operators and the lack of any corresponding elements in the check matrix G_z.

How are $\bar{X}_j$ operators defined? Suppose we have a logical state

$$|x_1, \ldots, x_k\rangle_L \tag{4.18}$$

and we wish to apply $\bar{X}_j$ on it and transform it to

$$|x_1, \ldots, x_{j-1}, x_j \oplus 1, x_{j+1}, \ldots, x_k\rangle_L. \tag{4.19}$$

The state (4.18) is defined to be stabilized by the stabilizer

$$S_{j1} = \left\langle g_1, \ldots, g_{n-k}, (-1)^{x_1}\bar{Z}_1, \ldots, (-1)^{x_k}\bar{Z}_k \right\rangle. \tag{4.20}$$

g_1	X	Z	Z	X	I
g_2	I	X	Z	Z	X
g_3	X	I	X	Z	Z
g_4	Z	X	I	X	Z
$\bar{Z}$	Z	Z	Z	Z	Z
$\bar{X}$	X	X	X	X	X

Tab. 4.2: Stabilizer generators and operators on logical qubit for five qubit code.

We want to change this stabilizer to stabilize the state (4.19) and thus transform it to the form

$$S_{j2} = \left\langle g_1, \ldots, g_{n-k}, (-1)^{x_1}\bar{Z}_1, \ldots, (-1)^{x_j+1}\bar{Z}_j, \ldots (-1)^{x_k}\bar{Z}_k \right\rangle . \tag{4.21}$$

In section 4.2.1 we showed that after action of unitary operator U on the state stabilized by operator S, stabilizer S changes to $USU^\dagger$. Therefore the operator $\bar{X}_j$ should be defined as operator on n qubits that comply the equation

$$\bar{X}_j S_{j1} \bar{X}_j^\dagger = S_{j2}. \tag{4.22}$$

Moreover, $\bar{X}_j$ should commute with all g_i, $i \in \{1, \ldots, n-k\}$ and $\bar{Z}_i, (i \neq j)$ and anti-commute with $\bar{Z}_j$.

Again if we suppose that generators of stabilizer are in the form (4.14),then the check matrix corresponding to the k encoded $\bar{X}$ operators is

$$G_x = [\overbrace{0}^{r}\ \overbrace{E^T}^{n-k-r}\ \overbrace{I}^{k}\ |\ \overbrace{C^T}^{r}\ \overbrace{0}^{n-k-r}\ \overbrace{0}^{k}]. \tag{4.23}$$

It can be shown (similarly as for $\bar{Z}$ operators) that operators $\bar{X}$ defined in (4.23) fulfill all independence, commuting and (4.22) conditions.

Note that the form of generators and $\bar{Z}$, $\bar{X}$ operators is not necessary the most convenient. Consider the five qubit code which generators and $\bar{Z}$, $\bar{X}$ operators are given in table 4.2. The generators and logical operators obtained by standard constructions are in table 4.3. We can see that first table is more synoptic, since $\bar{Z} = Z_1Z_2Z_3Z_4Z_5$, $\bar{X} = X_1X_2X_3X_4X_5$ and g_2 through g_4 can be obtain by cyclic shift of g_1. Of course the terms given in table 4.3 are equivalent to the former.

4.4.3 Stabilizers for CSS Codes

Now we will show that CSS codes are also stabilizer codes. The check matrix corresponding to the generator of stabilizer of CSS code defined by classical codes C_1, C_2 is

g_1	Y	Z	I	Z	Y
g_2	I	X	Z	Z	X
g_3	Z	Z	X	I	X
g_4	Z	I	Z	Y	Y
$\bar{Z}$	Z	Z	Z	Z	Z
$\bar{X}$	Z	I	I	Z	X

Tab. 4.3: Stabilizer generators and operators on logical qubit for five qubit code in **standard form**.

g_1	X	I	I	I	I	I	I	I	I	I	I	X	X	X	X	X	I	X	X
g_2	I	X	I	I	I	I	I	I	I	X	I	I	X	X	I	I	X	X	X
g_3	I	I	X	I	I	I	I	I	I	X	X	X	X	I	I	X	X	X	X
g_4	I	I	I	X	I	I	I	I	I	X	I	I	X	X	X	X	I	X	I
g_5	I	I	I	I	X	I	I	I	I	I	X	I	I	X	X	X	X	X	I
g_6	I	I	I	I	I	X	I	I	I	X	I	X	I	X	I	X	X	X	I
g_7	I	I	I	I	I	I	X	I	I	I	I	X	I	I	X	I	I	X	X
g_8	I	I	I	I	I	I	I	X	I	I	X	X	X	X	I	I	X	X	I
g_9	I	I	I	I	I	I	I	I	X	I	X	I	I	I	I	X	I	X	X
g_{10}	Z	I	I	I	I	I	I	I	I	I	Z	Z	Z	I	I	I	I	Z	I
g_{11}	I	Z	I	I	I	I	I	I	I	I	Z	Z	Z	Z	I	I	I	I	Z
g_{12}	I	I	Z	I	I	I	I	I	I	Z	I	Z	I	Z	I	Z	Z	Z	I
g_{13}	I	I	I	Z	I	I	I	I	I	I	Z	I	Z	Z	Z	I	Z	I	Z
g_{14}	I	I	I	I	Z	I	I	I	I	Z	I	Z	Z	Z	Z	I	Z	I	I
g_{15}	I	I	I	I	I	Z	I	I	I	Z	I	I	I	I	I	I	Z	Z	Z
g_{16}	I	I	I	I	I	I	Z	I	I	I	I	I	Z	Z	I	Z	Z	Z	I
g_{17}	I	I	I	I	I	I	I	Z	I	Z	Z	Z	I	Z	Z	Z	I	I	I
g_{18}	I	I	I	I	I	I	I	I	Z	I	I	Z	Z	I	Z	Z	Z	Z	Z
$\bar{Z}$	Z	Z	Z	I	I	I	Z	I	Z	I	I	I	I	I	I	I	I	I	Z
$\bar{X}$	I	I	I	I	I	I	I	I	I	X	I	I	X	X	I	I	X	X	X

Tab. 4.4: Stabilizer generators and operators on logical qubit for [[19,1,5]] code.

following:

$$M = \left[\begin{array}{c|c} H(C_2^{\perp}) & 0 \\ 0 & H(C_1) \end{array}\right]. \tag{4.24}$$

To show that M defines a stabilizer we must first verify whether its generators commute with each other. Using the lemma 4.3 we must show that $M\Lambda M^T = 0$. But we know that

$$M\Lambda M^T = H(C_2^{\perp})H(C_1)^T = G(C_2)^T H(C_1)^T = (H(C_1)G(C_2))T = 0, \tag{4.25}$$

where the last equality holds since $C_2 \subset C_1$. Now we will show that arbitrary generator of this stabilizer stabilizes states of $CSS(C_1, C_2)$. We show this for each codeword $|\psi_q\rangle \in CSS(C_1, C_2)$

$$|\psi_q\rangle = \sum_{i=0}^{2^{k_1-k_2}} c_i \,|x_i + C_2\rangle\,, \qquad \left(\sum_{i=0}^{2^{k_1-k_2}} |c_i|^2 = 1\right). \tag{4.26}$$

At first suppose that a generator g_i corresponding to the row of $H(C_2^{\perp})$ from M acts on $|\psi_q\rangle$.

$$g_i\,|\psi_q\rangle = \sum_{i=0}^{2^{k_1-k_2}} \frac{c_i}{\sqrt{|C_2|}} \sum_{y\in C_2} |x_i \oplus y \oplus r_x(g_i)\rangle\,, \tag{4.27}$$

where $r_x(g_i)$ denotes n-elements binary vector consisting of first n components of $r(g_i)$[5]. It follows that $r_x(g_i) \in C_2$, because $r_x(g_i)$ is a row from matrix $H(C_2^{\perp})$. Therefore $\{y + r_x(g_i) | y \in C_2\} \equiv C_2$ and it follows that

$$g_i\,|x + C_2\rangle = |x + C_2\rangle\,. \tag{4.28}$$

Now consider a stabilizer generator g_i corresponding to the row of $H(C_1)$ from M. We denote $r_z(g_i)$ an n-elements binary vector consisting of last n components of $r(g_i)$. Using the ideas from section 3.2.2, we can rewrite codeword $|\psi_q\rangle$ as

$$|\psi_q\rangle = H\left(\sum_{i=0}^{2^{k_1-k_2}} \frac{c_i}{\sqrt{2^n/|C_2|}} \sum_{z'\in C_2^{\perp}} (-1)^{x_i z'}\,|z'\rangle\right). \tag{4.29}$$

Application of g_i on the state $|\psi_q\rangle$ changes it to the state

$$H\left(\sum_{i=0}^{2^{k_1-k_2}} \frac{c_i}{\sqrt{2^n/|C_2|}} \sum_{z'\in C_2^{\perp}} (-1)^{x_i z'}\,|z' + r_z(g_i)\rangle\right), \tag{4.30}$$

[5] $r(g_i)$ is defined in section 4.2.

because Hadamard gate changes Z operators from g_i to act as X operators. It follows that $r_z(g_i) \in C_1^\perp$, because $r_z(g_i)$ is a row from matrix $H(C_1)$. Since $C_1^\perp \subset C_2^\perp$, we have $r_z(g_i) \in C_2^\perp$. Again we use the argument that

$$\{z' + r_z(g_i) | z' \in C_2^\perp\} \equiv C_2^\perp \tag{4.31}$$

to show that formulas in (4.29) and (4.30) are identical and thus $g_i |\psi_q\rangle = |\psi_q\rangle$.

We have shown that stabilizer defined by check matrix M (4.24) stabilizes all codewords from $CSS(C_1, C_2)$. From proposition 4.6 we know that V_S is a 2^{n-l} subspace of n-qubit state space. In our case the following holds:

$$l = \#rows(H(C_1)) + \#rows(H(C_2^\perp)) = n - k_1 + k_2, \tag{4.32}$$

and thus V_S should be a $2^{k_1-k_2}$ dimensional subspace of n-qubit state space. The $CSS(C_1, C_2)$ has exactly $2^{k_1-k_2}$ dimensions and therefore the given stabilizer stabilizes just $CSS(C_1, C_2)$ and nothing else. Stabilizer of [[19,1,5]] code with its logical operators are shown in table 4.4. It can be also shown that the correction procedure for stabilizer codes (from section 4.3) corrects exactly the same errors as proposed by correction procedure in section 3.2.2.

4.5 Encoding and Decoding Stabilizer Codes

If the stabilizer of the code is known, then construction of encoding circuit is straightforward. The process of encoding may be written as

$$|c_1 \dots c_k\rangle \rightarrow \left(\sum_{M \in S} M\right) \bar{X}_1^{c_1} \cdots \bar{X}_k^{c_k} |0 \dots 0\rangle \tag{4.33}$$

$$= (I + M_1) \cdots (I + M_{n-k}) \bar{X}_1^{c_1} \cdots \bar{X}_k^{c_k} |0 \dots 0\rangle , \tag{4.34}$$

where M_1 through M_{n-k} are generators of stabilizer S and $\bar{X}_1$ through $\bar{X}_k$ are encoded X operators for k logical qubits. To encode k qubit state to stabilizer code we only need to care about encoding of basis states $|c_1 \dots c_k\rangle$. At first we need to encode operators $\bar{X}_i$. Since the standard form of $\bar{X}$'s ensures that each $\bar{X}_i$ operates on a single one of last k qubits, we can substitute $|c_i\rangle$ for the $(n-k+i)$th input qubit and apply $\bar{X}_i$ conditioned on it[6]. This produce the state $\bar{X}_1^{c_1} \cdots \bar{X}_k^{c_k} |0 \dots 0\rangle$. We can see that $\bar{X}$ operators act as Z operators on first r qubits and as X operators on the next $n-k-r$ qubits. Since Z operator acts trivially on $|0\rangle$, we can ignore Z parts of each $\bar{X}_i$, leaving just the CNOTs in this part of the encoder. The first r qubits remain automatically in the state $|0\rangle$ after

[6] Application of $\bar{X}_i$ is controlled by $(n-k+i)$th qubit.

this step of encoding. Now we must to encode operators $(I + M_i)$. Note that we must just encode the operator $(I + M_i)$, only if M_i contains some X or Y operators. Suppose that M_i contains just Z operators. Since M_i commutes with all other generators and every $\bar{X}_j$ we can commute application of operator $(I + M_i)$ to act directly on the state $|0 \dots 0\rangle$. Since Z operators of M_i act trivially on $|0\rangle$, we can ignore any M_i in encoding, which is just tensor product of Z operators. We need to care about first r stabilizers which contain X operators. The standard form of the first r generators of stabilizer applies single bit flip at first r qubits. Therefore after the application of $(I + M_i)$; the first r qubits will be in state $|0\rangle + |1\rangle$. Therefore we apply the Hadamard transform to the first r qubits. Then we simply apply M_i, omitting X_i, (for $i = 0 \dots r$) conditioned on i-th qubit. We must specially handle with the case when M_i contains operator Z_i. Z_i introduces minus sign if i-th qubit is $|1\rangle$ and acts trivially on $|0\rangle$. Therefore we can simply apply Z gate after Hadamard gate for all M_i that contain operator Z_i. The encoding circuit for [[19,1,5]] code is shown on figure 4.1. It is based on stabilizer from the table 4.4. The generators of stabilizer from the table 4.4 which contain just X operators are in the standard form. Also $\bar{X}$ and $\bar{Z}$ operators are in standard form and therefore the encoding circuit can be obtained using the previous rules, since operators which contain just Z operators are not involved in the encoding procedure. We can decode an encoded qubit by performing the encoding circuit in reverse. More details on stabilizer codes encoding/decoding can be found in [29] in fourth chapter.

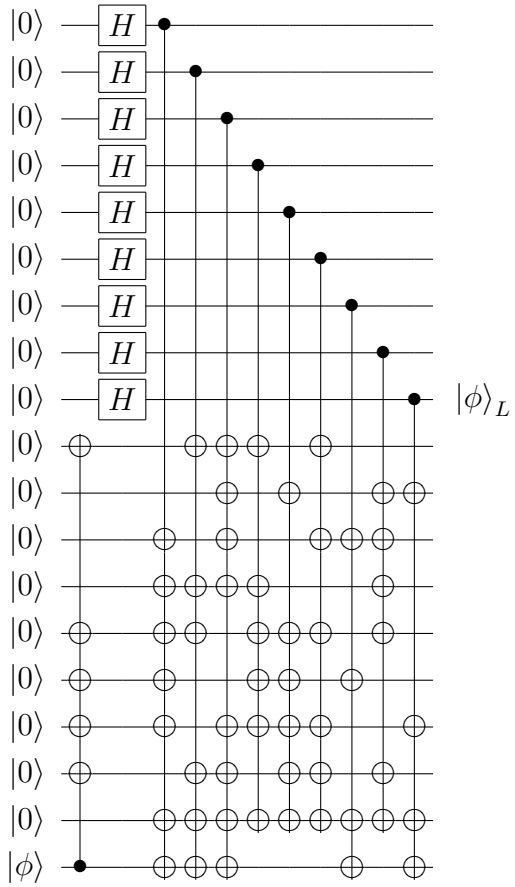


Fig. 4.1: Encoding circuit for [[19,1,5]] code.

5. FAULT-TOLERANT QUANTUM COMPUTATION

We have shown how to encode qubits in data blocks to protect them from errors occurring in transition media. A concept for performing quantum operations on the data was presented in section 4.2.1. We will be interested in the operators U, which do not modify the stabilizer of the code ($USU^\dagger = S$). For instance, logical operators $\bar{X}, \bar{Z}$ of stabilizer codes satisfy this condition. Also many other operators do not change stabilizer of the code. Moreover, we require that operator U is implemented *fault-tolerantly*. We define the fault-tolerance of a quantum circuit to be the property that if only k components in the procedure (circuit) fails then the failure causes error at most on k qubits in each encoded block of output from the procedure. By *component* we mean any of the elementary operations used in the circuit: noisy gates, noisy measurements, noisy quantum wires and noisy quantum state preparation.

Example 5.1. Application of CNOT gate on two qubits from the same data block is not a fault tolerant operation. Suppose that X error occurred on the control qubit just before the application of CNOT. If the unitary operator for CNOT gate is denoted by U_C, then the effective action of the circuit is $U_C X_1 = U_C X_1 U_C U_C^\dagger = X_1 X_2 U_C$. It looks like the U_C was applied correctly on error-free input and then X error occurred on both target and control bits. Initially single error in input results in two correlated errors in output from the gate.

If the CNOT would be applied on the qubits from two different data blocks the implementation would be fault-tolerant since the output would have just single error in each data block.

Useful observation is that the quantum operation is automatically fault-tolerant if it can be implemented in bitwise manner. The property, that an encoded gate can be implemented in bitwise manner is called *transversality*. For example recall Steane code and its logical operators $\bar{X}, \bar{Z}$,

$$\begin{aligned} \bar{X} &= X_1 \otimes X_2 \otimes X_3 \otimes X_4 \otimes X_5 \otimes X_6 \otimes X_7 \\ \bar{Z} &= Z_1 \otimes Z_2 \otimes Z_3 \otimes Z_4 \otimes Z_5 \otimes Z_6 \otimes Z_7. \end{aligned} \tag{5.1}$$

Therefore both $\bar{X}$ and $\bar{Z}$ can be implemented transversally and so fault-tolerantly. Moreover, also transversal application of Hadamard gate to each qubit implements

Hadamard gate on encoded qubit. This observation comes from the fact, that Hadamard gate can be defined as a gate which exchanges X and Z operators under conjugation:

$$HXH^\dagger = Z \quad HZH^\dagger = X \tag{5.2}$$

Since $\bar{H}$ defined as $\bar{H} = H_1H_2H_3H_4H_5H_6H_7$ satisfies conditions (5.2) for $\bar{X}$,$\bar{Z}$,$\bar{H}$ it acts as Hadamard gate on encoded qubit.

5.1 The Rules of Fault-Tolerant Computation

To perform fault-tolerant computation (FTC) we need to ensure several rules to avoid error accumulation in the data. These laws are described in more details in [33, 34, 35].

The first rule: **Do not use the same qubit twice.** A peculiarly quantum mechanical feature is that, while even a classical CNOT gate propagate bit flip errors from source to target, for quantum gates we must also worry about phase errors, which propagate in opposite direction, from the target to the source. Therefore in quantum computation we need to be especially careful about propagation of errors. Example of this rule may be shown on circuit correcting X errors of [[19,1,5]] code (figure 3.2). This circuit does not comply to the first rule of FTC, since each ancilla qubit is used several times with different encoded qubits and a single phase error in ancilla qubit would propagate to encoded qubits and cause multiple correlated errors in encoded data. The bit flip detection of [[19,1,5]] code should be made fault tolerantly as described in section 5.2.

The second rule: **During the error detection copy the error, not the data.** This rule is crucial and we cannot perform consistent computation without fulfillment of the second law. Suppose a circuit design for measuring first syndrome bit for Steane code on figure 5.1 a). In order to fulfill the first rule of FTC we use four ancilla qubits, each initially in the state $|0\rangle$ and CNOT gates on particular data qubits and ancilla qubits are applied as shown on the figure. Then each ancilla qubit would be measured separately in computational basis and according to the parity of these four measurements' outcomes would be decided whether an error occurred. The problem is that from such measurements we would have learnt not just the parity but also the values of all four data qubits, which acted in CNOT gates. Suppose that the encoded qubits were in the state $\alpha|0_L\rangle + \beta|1_L\rangle$ and the outcome of measurement of ancilla qubits would be $|1010\rangle$. The initial state after the measurements therefore collapse to the state

$$\alpha|1011010\rangle + \beta|0101010\rangle\,, \tag{5.3}$$

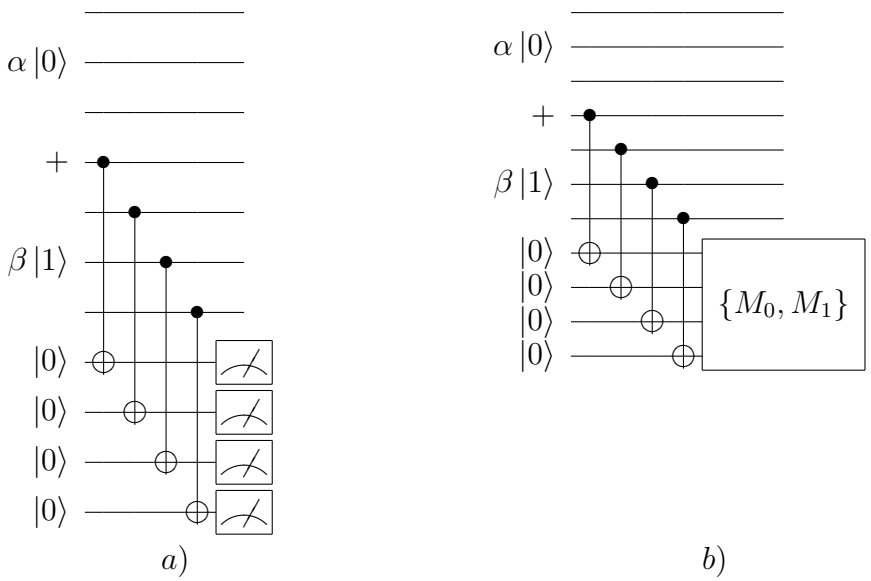


Fig. 5.1: Detection of the first syndrome bit of Steane code; a) Incorrect syndrome detection destroying encoded state; b) correct fault tolerant syndrome bit detection.

what definitely destroys initial encoding in Steane's code. The better approach would be to make a quantum measurement of all four ancilla qubits with measurement components:

$$M_0 = \sum_{v_i \in \{0,1\}^4, wt(v_i) \equiv 0 \ mod\ 2} |v_i\rangle \langle v_i| \tag{5.4}$$

$$M_1 = \sum_{v_i \in \{0,1\}^4, wt(v_i) \equiv 1 \ mod\ 2} |v_i\rangle \langle v_i| \,, \tag{5.5}$$

as shown on figure 5.1 b). This measurement really detects just the error (the parity) and does not corrupt encoded data. The measurement described with components (5.4)-(5.5) is quite complicated as it involves measuring four qubits simultaneously. The single qubits measurements are easier performed experimentally. The complexity of final measurement using more ancilla qubits can be transformed to the complexity of preparation ancilla qubits in special initial state. This concept is described in section 5.2.1.

The third rule: **Verify when you encode a known quantum state.** The error during preparation of known state is critical to subsequent computation. Therefore every time the preparation of known state is done verification measurement should follow, to ensure the correctness of the encoding.

The fourth rule: **Repeat the operations.** This rule should be fulfilled with all operations, but most important application of this rule is to the measurement of the error syndrome. An error during syndrome detection can both damage data and result in

erroneous state after the recovery. If we mistakenly accept the measured syndrome and act accordingly, we will cause further damage instead of correcting the error. Therefore, the high confidence about detected syndrome is very important. To achieve sufficient confidence, we can repeat the syndrome detection several times, as fourth rule says.

The fifth rule: **Use the right code.** The choice of particular code has to be considered carefully. Different codes are more or less effective against particular type of errors. We should ensure that quantum gates could be applied on the data encoded in a code efficiently adhering to previous rules of *FTC*.

It turns out that arbitrary long reliable quantum computation can be achieved even with faulty quantum gates, provided that the error probability per gate is below a certain constant threshold ($\approx 10^{-4}$). The methods which enable arbitrary long quantum computation use several levels of encoding of so known *concatenate codes* [36]. These methods give us promising results but their scale up is much higher than of some simpler codes. To practically use concatenate codes we would need to have quantum computers consisting of many (exponential from # of encoding levels) physical qubits.

5.2 Fault Tolerant Error Detection of CSS Codes

The syndrome detection for CSS codes consists of application of CNOT gates with control qubits corresponding to 1's in parity check matrices of C_1 and $C_2^{\perp}$. But the basic concept as presented in section 3.4 is not fault tolerant. It violates the third rule of *FTC*. The fault of a single gate may cause an error on all data qubits which access the same ancilla qubits.

5.2.1 Fault Tolerant Syndrome Bit Detection

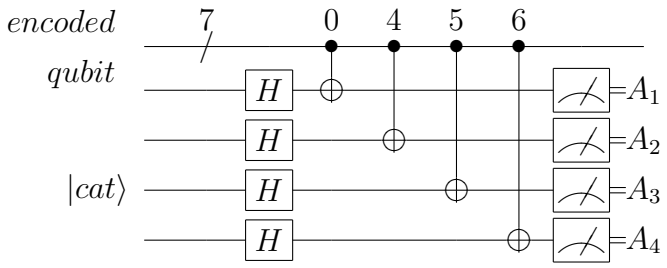
Shor [33] proposed fault tolerant detection of each syndrome bit. For each syndrome bit we need so known *cat* state

$$|cat\rangle = \frac{1}{\sqrt{2}}(|0...0\rangle + |1...1\rangle), \tag{5.6}$$

which after the application of Hadamard gate transforms to the state, which is an equal superposition of all even weighted states:

$$H^{\otimes k}\,|cat\rangle = 2^{\frac{1-k}{2}} \sum_{even|x\rangle} |x\rangle \tag{5.7}$$

Instead of multiple CNOTs acting on one ancilla qubit, in this scope each CNOT is targeted to different ancilla qubit. After application of CNOT gates, each ancilla qubit is measured in standard basis. Fault tolerant circuit is shown on figure 5.2



(5.8)

Fig. 5.2: Fault tolerant syndrome bit detection for Steane code.

If all CNOT gates acted faultlessly the ancilla qubits rest in the state which is equal superposition of all even weighted or odd weighted states, depends on the parity of qubits being the control qubits for CNOT gates. Therefore after the measurement of each ancilla qubit we do not get any information about qubits being measured. Only information we get is a parity of particular data qubits acted in CNOT gates. If k of the gates in syndrome detection would fail, then the error just to k data qubits would be introduced. The xor of measured values A_i forms desired syndrome bit.

$$\text{syndrome bit } = A_1 \oplus A_2 \oplus A_3 \oplus A_4 \tag{5.9}$$

5.2.2 Preparation of cat State

The *cat* state must be also prepared precisely, since errors in *cat* state may propagate to the data and destroy it as mentioned in the third rule of *FTC*. To fulfill fault tolerance of *cat* state we introduce several verification qubits, which will detect X errors in prepared *cat* state. The amount of verification qubits depends on the measure of credibility we require from the *cat* state. In Steane code we need to ensure that two or more errors in the *cat* state occur with the probability $O(\xi^2)$, where ξ is a probability of gate failure. To ensure this, one verification qubit is sufficient. We perform two CNOT operations taking verification bit as a target and first and last qubit of *cat* state as control qubits. The last step of verification is a measurement of verification qubit in standard base. If the measurement results as $|1\rangle$ we repeat the preparation procedure. In the case of [[19,1,5]] code, we need to ensure that three or more X errors occur with probability $O(\xi^3)$. The rows of H_1 and $H_2^{\perp}$ do not have same amount of one's as Steane code parity check matrices therefore *cat* state will have different amount of qubits for each syndrome bit. The preparation of *cat* state will depend on the syndrome bit we want to detect. For simplicity we can always prepare the biggest *cat* state needed and then use just as much qubits as are needed for particular syndrome detection. To ensure that k qubits *cat* state has three or more X errors with the probability $O(\xi^3)$ we introduce k verification qubits. To each verification qubit we apply CNOT of two neighbor qubits

from *cat* state. After the application of all CNOT gates, all verification qubits are measured in standard base. If any of verification measurements would results with $|1\rangle$, whole preparation will be repeated. By slightly complex analysis one can verify that any pair of failures in the preparation or verification part of the circuit, which result in X errors of *cat* state will be detected. Five qubit *cat* state preparation is shown on figure 5.3.

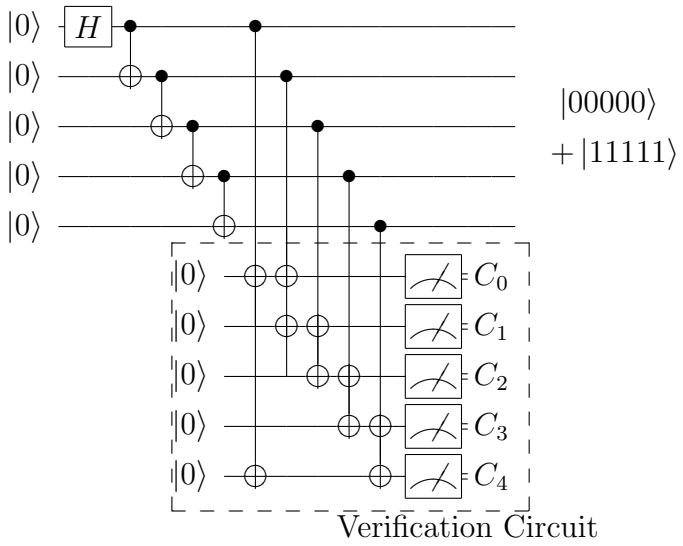


Fig. 5.3: Preparation circuit with verification part for the ancilla consisting of five qubits: The verification is more complex and ensures that three or more X errors occur with probability $O(\xi^3)$.

However, several Z errors can occur in the *cat* state prepared in this way. But the Z errors do not pass through CNOT gates from target qubits to control qubits and thus these errors could not propagate to the qubits of encoded data. Of course these Z errors may cause that incorrect syndrome bit will be obtained. The probability that the Z error from *cat* preparation will modify particular syndrome bit is approximately $m\xi$ where m is sum of state preparations,Hadamard gates and CNOT gates in preparation circuit, for the case shown on previous figure $m = 25$.

5.2.3 Ensuring Correct Syndrome Detection

The potential danger of introducing more errors to the data qubits resides in the possibility of incorrect syndrome detection due to failure of several measuring gates. After incorrect syndrome detection we would try to correct wrong data qubits and thus would introduce additional errors. To diminish the probability of getting wrong syndrome bit to $O(\xi^2)$ we may detect each syndrome bit independently several times and as a result we take a majority of these detections. Approach of multiple detections inquire prob-

ability of $O(\xi^2)$, but the disadvantage is that it provides more places where errors to the data qubits may be introduced. One must take care also about possible errors introduced to the data in the middle of syndrome detection procedure. This error would not be detected in syndrome bits obtained before the error occurred and thus incorrect syndrome would be determined.

The solution to this problem is with additional usage of classical code [29, 37]. We can encode $n-k$ syndrome bits using a classical $[m, n-k, d']$ linear code C_S with generator matrix G_S and parity check matrix H_S. To determine which data qubits we need to measure to obtain encoded syndrome bits we use a matrices $G_S H_1$ and $G_S H_2^{\perp}$ instead of matrices H_1 and $H_2^{\perp}$. The code C_S is used just for the error detection, the automatic error correction of obtained syndrome bits would be dangerous. If error is detected in obtained syndrome bits, all syndrome bits detections are repeated. If no error is detected in obtained syndrome bits, initial syndrome bits corresponding to matrices H_1 and $H_2^{\perp}$ are decoded.

For Steane code the sufficient code C_S is such linear code, which can detect single error, therefore simple parity check code with $G_S = \begin{bmatrix} 100 \\ 010 \\ 001 \\ 111 \end{bmatrix}$ is sufficient. Obtained syndrome bits are checked against parity check matrix H_S. If an error is detected in these four bits, whole syndrome detection is repeated. The probability that a wrong syndrome would be determined using this schema is of order $O(\xi^2)$[1]

For [[19,1,5]] we require credibility of determined syndrome of $O(\xi^3)$, therefore we need a code G_S which detects any two errors in syndrome bits. Classical error correction code with minimal distance $d = 3$ is sufficient to detect any two errors. We use $[n_c, 9, 3]$ code. The smallest possible n_c according to the corollary 2.8 is $n_{min} = 13$. We have used following [13,9,3] *ECC* C_S in standard form, with generation matrix

$$G_S = \begin{bmatrix} 1000001011011 \\ 0100010101101 \\ 0010100110110 \\ 0001111000111 \end{bmatrix}. \tag{5.10}$$

The parity check matrix H_S can be obtained easily according to the rule (2.15). The mechanism is similar to the previous. We detect 13 syndrome bits and we check them

[1] The only possible way, how a wrong syndrome would be determined with probability $O(\xi)$ is that detections of first two syndrome bits return 0 and during the third syndrome bit detection a fault would occur introducing an error to the data qubit. Then other two syndromes may return 1 and wrong syndrome would be decoded. This case must be handled separately.

against parity check matrix H_S. If error is detected all 13 syndrome bits detections are repeated. If no error is detected, initial 9-bit syndrome is decoded from obtained 13 bits. Since C_S is in standard form we simply take first nine bits as original error syndrome.

6. QUANTUM CODES ANALYSIS

In previous chapters we have presented several quantum error correcting codes. In this chapter we will compare and analyze found [[19,1,5]] code and Steane Code. Present work is based on previous analysis of Zalka [37], Steane [38] and partially on [35]. We have not considered concatenated codes [36] in our analysis, since concatenate coding is known to be inefficient in both space (number of physical qubits) and time (number of elementary operations), compared with non-concatenated codes. We have developed theoretical estimates of the codes ability to protect encoded states using fault tolerant circuits. Theoretical estimates have been verified by numerical Monte Carlo simulations.

6.1 Noise Model

At first we present our noise model. 'Noise' in the context of QEC means any process which causes the state of the physical qubits to be different from what it should ideally be [39, 30]. It is difficult to discover quantum error-correcting codes (*ECC*) for general types of interactions. In classical theory of error-correction, it is often assumed that errors occur independently at each bit. This assumption seems physically reasonable in many situations. In cases where it is not strictly true it can still lead to a systematic approach for finding good *ECC*. In some systems, there may be errors that move the system outside of the computational space. For instance, if the data is stored in a polarization of photon, the photon might escape. This sort of error is called *leakage error*. In practice, simple mechanism may be introduce to detect leakage error and convert it to the located error. In the case of photon, if the photon escapes, we just introduce new photon with the random polarization. In our analysis we therefore consider just errors where particular qubits may change its states in computational basis. An error on particular qubit occurs randomly with some probability during each time step (memory errors). The second source of errors are failures of gates, the qubits are subjected to. In our model we suppose that memory errors rate may be ignored compare to the rates of gates' failures. We simply assume that the majority of errors on physical qubits is caused by the interactions in quantum gates. Moreover, we assume that particular gates fail stochastically, independent on each other with some fixed probability ξ. The

failure of a gate introduce an error to the qubits it acts on. In particular physical systems various types of errors can be more or less probable. In this chapter we will consider that error on single qubit is equally likely (with probability $\xi/3$) to be a X, Z or XZ error. In real systems, the assumption that errors are equally likely to be bit-flip, phase flip or both of them is a poor one. In practice, some linear combinations of X, Z and XZ are going to be more likely than others. For instance, when the qubits are ground or excited states of an ion, a likely source of errors is spontaneous emission. Therefore an error altering the excited state to the ground state is more probable then the opposite. Already known codes are able to correct arbitrary single qubit errors but some codes may be more efficient in particular physical representations than others. Understanding the physical likely sources of errors will certainly be an important part of engineering quantum computers.

6.2 Comparison of Quantum Codes - Error Free Correction Procedure

We have introduced the probability of error on a single qubit. Suppose now that we are using an error correcting code which uses n qubits and is capable of correcting up to t errors on n qubits block. Suppose that procedures of coding and decoding are error free. Using such code, the probability of the successful survival of a single encoded qubit subjected to logical gate is[1]

$$\mathrm{P}_{(n,t)} = \sum_{i=0}^{t} p_i = \sum_{i=0}^{t} \binom{n}{i} \xi^i (1-\xi)^{n-i}, \tag{6.1}$$

where p_i stands for the probability that i errors occur in the code word.

The natural question is, which codes are better for given fault rate of a single gate ξ. The graph on figure 6.1 shows dependence $\mathrm{P}_{(n,t)}$ on ξ for Steane code, [[19,1,5]] code and qubit not protected by error correction. We can read from the graph that if $\xi < 0.027$ then the best choice from these possibilities is to use [[19,1,5]] code. If the estimated error rate is in interval $< 0.027,\ 0.057 >$ we should use Steane code and if the error rate $\xi > 0.057$ we should not use neither [[19,1,5]] nor Steane code as both have less probability $\mathrm{P}_{(n,t)}$ than $1-\xi$. Exact threshold values (0.027, 0.057) were computed from the equations

$$\mathrm{P}_{(7,1)} = \mathrm{P}_{(19,2)} \tag{6.2}$$

$$1-\xi = \mathrm{P}_{(7,1)} \tag{6.3}$$

[1] We suppose that logical gates act in bitwise manner on encoded qubits.

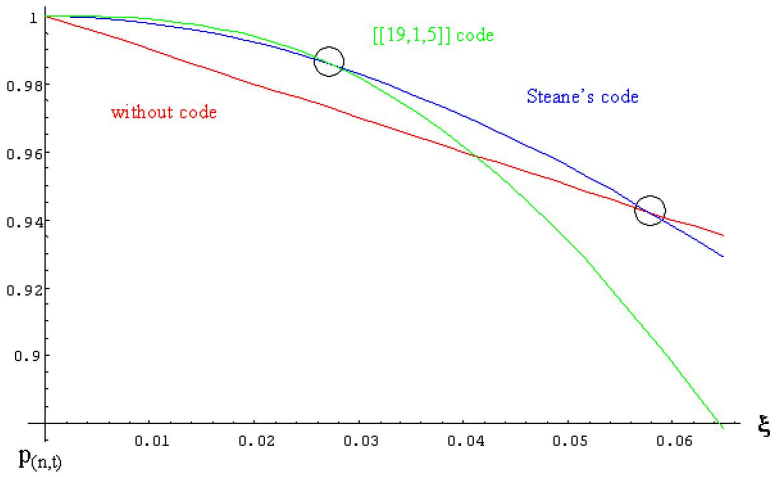


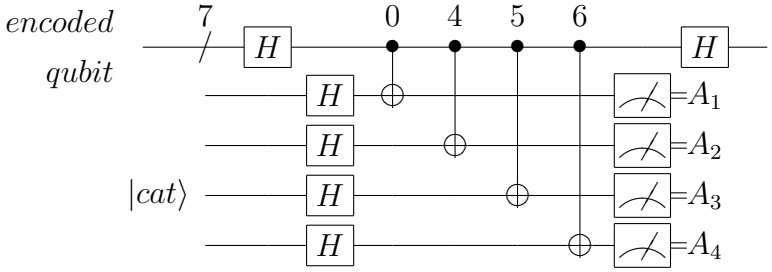
Fig. 6.1: Comparison of $P_{(7,1)}$ and $P_{(19,2)}$.

6.3 Imperfect Error Detection, Coding and Decoding

Results of previous paragraph have used very strong assumption, namely that procedure of syndrome detection and error correction are error free. This assumption is realistic in classical computers where the error rate in correcting hardware is insignificant compared with the error rate in the transmission media. However the environment of quantum systems has still significant error rates and its effect is crucial to the successfulness of error correction protocol. More complex analysis is needed. The error correction mechanism has to be build in fault tolerant way. The concept of fault tolerance was explained in details in the chapter 5.

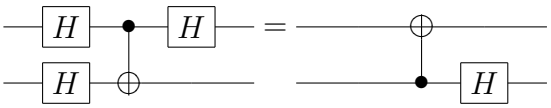
Reducing the number of Hadamard transforms

The effort is always paid to decrease the number of operations executed on the data qubits as each gate may introduce error to the data. Till now we have been using following schema for detecting particular Z-errors syndrome bit:



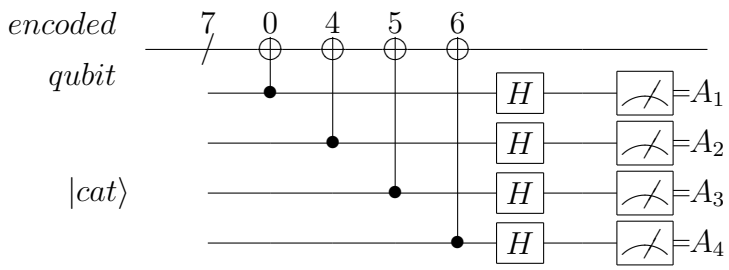
(6.4)

Using the identity (6.5), the detection of Z-errors syndrome bit



(6.5)

can be simplified to the schema (6.6), where Hadamard gates are not applied to the data qubit. This simplification saves $2n$ application of Hadamard gates to data qubits and thus the probability that encoded state would get unrecoverable damaged is lowered. The simplification was shown for detecting first syndrome bit of Steane code, but in the same manner the Z-errors syndrome detection may be simplified for any CSS code.



(6.6)

6.3.1 Theoretical Analysis

The probability that the single encoded quantum qubit remains error free is much less than the estimate given in the equation (6.1). For the simplification, suppose now that the probability of gate crash ξ is small enough and therefore we often approximate term $(1 - n_l\xi) \approx 1$. We want to estimate the probability that the encoded state will be unrecoverable damaged by the noise using the fault tolerant implementations of Steane or [[19,1,5]] code. We consider following model of qubit protection. The qubit is subjected to n_l logical single qubit gates[2] and then error correction is applied.

The Estimate of Failure Probability for Steane Code

The encoded qubit may become unrecoverable damaged in three different ways:

- Two errors occur during computing n_l logical gates: This is with probability

$$p_1 \approx \binom{7}{2}(n_l\xi)^2 \approx \frac{1}{2}(7n_l\xi)^2.$$

- Two errors occur during syndrome detection: with probability

$$p_2 \approx \binom{k}{2}\xi^2 \approx \frac{1}{2}(k\xi)^2,$$

where k is number of places where an error to the data qubits may be introduced.

- One error occurs during n_l logical operations and one error occurs during error detection, this happens with probability $p_3 \approx (7n_l\xi)(k\xi)$.

[2] We consider just single qubit gates in our analysis, since it makes the analysis feasible.

Adding these probabilities together give us that the probability, that an encoded qubit will crash (counted relatively to the number of performed operations) is:

$$\mathrm{p}_S(\xi, n_l) = \frac{7}{9}\frac{p_1 + p_2 + p_3}{n_l} = \frac{7}{18}\frac{(7n_l + k)^2}{n_l}\xi^2 \tag{6.7}$$

This theoretical estimate is not precise as we have ignored factors of $(1 - n_l\xi)$ and other simplification is that we were not considering X and Z errors separately. We just introduce additional factor of 7/9 because 2/9 of errors pairs are correctable with the Steane code (X_iZ_j error pairs are correctable). Methods for estimating k from equation (6.7) are described below.

Optimal number of logical operations between two correction steps is computed from equation:

$$\frac{\partial \mathrm{p}_S(\xi, n_l)}{\partial n_l} = 0 \tag{6.8}$$

$$14(7n_l^* + k)n_l^* - (7n_l^* + k)^2 = 0 \tag{6.9}$$

$$n_l^* = \frac{k}{7} \tag{6.10}$$

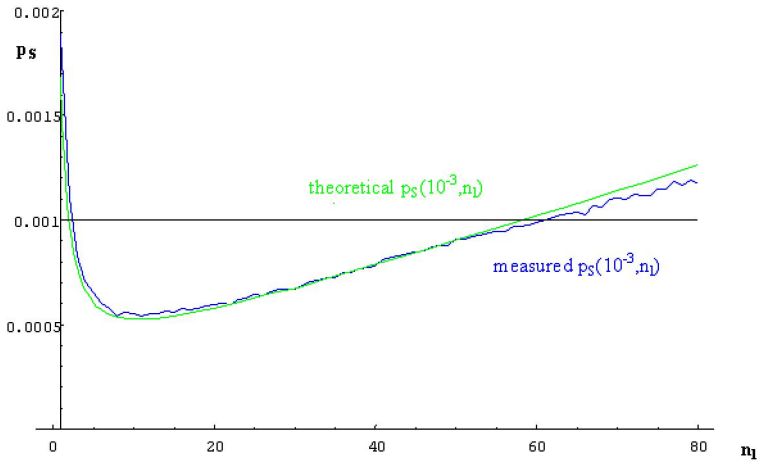


Fig. 6.2: Steane Code: Function $\frac{9}{14}\mathrm{p}_S(10^{-3}, n_l)$ is compared with data obtained by computer simulation for $n_l \in \{1, \ldots, 80\}$.

The Estimate of Failure Probability for [[19,1,5]] Code

Theoretical estimate of failure probability for [[19,1,5]] code is constructed in similar way as for Steane code. The qubit protected by [[19,1,5]] code will become unrecoverable damaged if three or more errors occur on its 19 encoding qubits. This may happen as:

- Three errors occur during computing n_l logical gates: This happens with probability

$$p_1' \approx \binom{19}{3}(n_l\xi)^3 \approx \frac{1}{6}(19n_l\xi)^3$$

- Three errors occur during syndrome detection: with probability

$$p'_2 \approx \binom{k}{3}\xi^3 \approx \frac{1}{6}(k\xi)^3,$$

where k is number of places where error to data qubits may be introduced

- One error occurs during n_l logical operations and two errors occur during error detection, this happens with probability

$$p'_3 \approx (19n_l\xi)\binom{k}{2}\xi^2 \approx \frac{1}{2}19n_lk^2\xi^3.$$

- Two error occur during n_l logical operations and one error occurs during error detection, this happens with probability

$$p'_4 \approx \binom{19}{2}(n_l\xi)^2(k\xi) \approx \frac{1}{2}(19n_l)^2k\xi^3.$$

Again the estimation is not precise, just 5/9 of error triples are not correctable. We also neglected factors of $(1-(n_l\xi))$ and all occurrences of four and more failures. The estimated crash probability therefore is:

$$\mathrm{p}_{19}(\xi, n_l) = \frac{5}{9}\frac{p'_1 + p'_2 + p'_3 + p'_4}{n_l} = \frac{5}{54}\frac{(19n_l + k)^3}{n_l}\xi^3 \tag{6.11}$$

Optimal number of logical operations between two correction steps is computed from equation:

$$\frac{\partial \mathrm{p}_{19}(\xi, n_l)}{\partial n_l} = 0 \tag{6.12}$$

$$3.19n_l^*(19n_l^* + k)^2 - (19n_l^* + k)^3 = 0 \tag{6.13}$$

$$n_l^* = \frac{k}{38} \tag{6.14}$$

Estimation of k in Equations (6.7) and (6.11)

To make equations (6.7) and (6.11) useful, we must provide values of k in these formulas. Let's analyze whole syndrome detection circuit. Note that if the syndrome detection works properly, no additional error to the qubit would be introduced in the error correction part of the protocol, because the recovery mechanism acts only on erroneous qubits and thus cannot cause more errors than already were there. Let us mark the number of ones in matrix $G_S H_1$[3] as n_x and number of ones in matrix $G_S H_2^{\perp}$ as n_z. In the syndrome detection there are three parts where possible failure may introduce an error to the data. Most obvious part are CNOTs between the data qubits and ancilla qubits.

[3] Matrix G_S was introduced in section 5.2.3

Number of these are $n_x + n_z$. Second place, where the failure may cause the error is failure of Hadamard gates applied to *cat* state before CNOTs take action. According to the idea of section 6.3 Hadamard gates are applied just in X error detection part and thus n_x times. The last place where the single failure may introduce an error to the data qubits are X failures of last gates applied to *cat* state in the *cat* state preparation. Since the failure occur in the last gate it cannot be detected, unless it introduce an X error also to the verification qubit. In our noise model the X error is not introduced to particular qubit in 1/3 of times. Therefore the expected value of k in equations (6.7) and (6.11), without considering syndrome detection repetition is

$$k_{nr} = 2\frac{1}{3}n_x + 1\frac{1}{3}n_z. \tag{6.15}$$

Value of k is actually even higher because the whole syndrome detection may be repeated several (r) times due to wrong syndromes are detected[4]. At $\xi = 10^{-3}$ the expected amount of syndrome detection repetitions for Steane code obtained from simulations is $r = 1.31$ so therefore more accurate k is

$$k = r.k_{nr} = 1,3\left(2\frac{1}{3}n_x + 1\frac{1}{3}n_z\right) = 1,3.57 \approx 75. \tag{6.16}$$

Therefore in the case of ξ around 10^{-3}, the expected failure probability of Steane code is

$$\mathrm{p}_S(n_l, \xi) \approx \frac{7}{18}\frac{(7n_l + 75)^2}{n_l}\xi^2, \tag{6.17}$$

and optimal value of n_l is

$$n_l^* = \frac{k}{7} \approx 11. \tag{6.18}$$

Improvement of [[19,1,5]] Code

We have considered additional improvement of syndrome detection for [[19,1,5]] code, according to fourth rule of *FTC*. The Z errors from *cat* state preparation are not detected and propagate to the syndrome detection procedure and may cause that wrong syndrome is determined. The probability of wrong syndrome detection may be lowered by multiple detections of each syndrome bit. Of course, each detection must use its own *cat* and verification qubits and thus this introduces the need of more auxiliary qubits. The disadvantage is that more operations with data qubits are performed and thus more errors may be introduced during extraction of one syndrome qubit. The advantage is that the probability that the Z errors from *cat* state preparation cause wrong syndrome to be extracted is reduced from $O(\xi)$ to $O(\xi^2)$. Therefore much less repetitions of all

[4] Matrix H_S is used to detect errors in obtained syndrome bits.

syndrome bits extractions is needed. We tried to detect each syndrome bit three times and then we take the majority of outcomes. The minor improvement of this is: If first two detections of particular bit end with the same results the third detection is not performed. It is not clear if this method introduces an improvement, since it has both positive and negative influence. Numerical simulations of error detection with single syndrome bit extraction and multiple syndrome bit extractions were done at failure rate $\xi = 10^{-4}$.The actual values of n_x and n_z for matrix G_S used with [[19,1,5]] code are $n_x = 93$ and $n_z = 105$, so $k_{nr} = 350$. The detection of all syndrome bits may be repeated several times because of error detected by code C_S. Therefore actual $k = k_{nr}.r$, where r is expected amount of syndrome bits detection cycles. From the numerical simulations we obtained repetition constants for single syndrome bit detection ($r_1 = 1.65$) and multiple detections ($r_2 = 1.01$). Therefore total number of k for both methods are approximately

$$k_1 = k_{nr}.r_1 \approx 580, \tag{6.19}$$

$$k_2 = 2k_{nr}.r_2 \approx 700, \tag{6.20}$$

which conclude that the single syndrome bit detection provides better results at $\xi = 10^{-4}$ fault rate.

6.3.2 Model of Quantum Computer and Numerical Analysis

The effectiveness of quantum error correction codes can be evaluated using a numerical simulations on a classical computer. We have argued several times that I, X, Z, XZ error basis is sufficient for noise analysis. Therefore it is sufficient to maintain two bits of classical information denoting the occurrence of X and Z errors for each physical qubit in simulated quantum computer. Of course, more complicated noise may occur, but after the error syndrome is measured each particular noise collapse to the one of base errors. We do not need to maintain complete information about the noise when it occurs, we just maintain the noise subspace it will collapse later, during the ancillas measurements.

Simulated Architecture

The architecture of simulated quantum computer (QC) is following. QC consists of n logical qubits, which are engaged directly in the computation. Each of these n qubits is encoded in CSS code. Two codes were tested: Steane Code and [[19,1,5]] code. Since qubits are encoded in CSS code we suppose that required quantum gates may be applied fault tolerantly. We know that one qubit gates and CNOT gate form an

universal set of gates [13]. Therefore we decided to simulate this set of gates. We suppose that each quantum algorithm is first rewritten to the CNOTs and single qubit gates. We know that for the CSS codes CNOT, X,Y,Z gates may be applied in bitwise manner. We have used little stronger assumption in the simulation, namely that also all single qubit gates may be applied in bitwise manner. This assumption makes the numerical simulation much easier and the outcome it brings should be not far from the reality. The crucial question is how often should be the error correction applied. This parameter was obtained both theoretically (n_l^*) and from numerical simulations. So the architecture is following: The computation evolve n_l steps on all logical qubits and then on each qubit error syndrome is measured and corrected.

Modeling gates

Only two-qubit gate in the computer simulation is CNOT. If CNOT is applied, one must remember that X error evolve from control qubit to target qubit and Z error evolve in the different orientation. Each gate may fail with probability ξ. The failure of a gate introduce one of X,Z or XZ errors to the qubit(s) it is applied to.

Error Correction Simulation

Each of n logical qubits is corrected after n_l computational steps. To perform error correction, the QC has extra qubits for each logical qubit. The error correction procedure includes *cat* state preparation, *cat* state verification, application of CNOTs among *cat* state and encoded qubits, ancilla qubits measurement and final recovery from the error. Each of the components may fail. If a failure is detected the particular part of the correction protocol must be repeated. Qubit preparation and also qubit measurements are simulated in the same manner as quantum gates, and thus may introduce errors. Other special gate used is Hadamard gate. The Hadamard gate exchanges X and Z errors on the qubit it is applied on.

Data from simulations were obtained in following way: Computation evolve in following cycle: n_l logical operations are performed on QC qubits followed by error correction of each qubit. The simulator program checks whether the errors on QC qubits are correctable with given code, after the error correction part of each cycle. If the simulator finds that the state of QC qubits is not correctable, then all qubits are initialized to the state without error and simulator increases its crash-counter. This is repeated until the crash-counter reaches some given limit. Probability of system crash is counted from the reached limit and total number of logical operations successfully performed on qubits during the simulation. Particular value of limit differs for different simulations

but always chosen from interval $\langle 50, 500\rangle$, it depends on the failure rate ξ and amount of qubits n in simulated QC. The lower ξ and higher amount of qubits in QC, the more CPU time is needed for the simulation. Results for QC consisting of $n = 100$ qubits shown in table 6.1 were obtained after 30 hours of CPU time on 2500MHz processor. Typically simulations of single qubit lasted less than few hours, but the required time increases significantly for lower failure rates ξ. Therefore we do not report simulation results for $\xi < 1.10^{-5}$.

Results of Numerical Simulations Compared with the Theory

The simulated architecture was verified against the theory in the most simple case. We have considered only single qubit in the computer with n_l single qubit gates applied between the error corrections. The data obtained from numerical simulations confirm that the theoretical analysis provides enough accuracy. The functions $p(n_l, \xi)$ were just scaled by overall multiplicative factor to best fit the data obtained numerically. The reason for this factor comes from the neglecting factors $(1 - n_l\xi)$ in theoretical estimations. The multiplicative factor for Steane Code is 9/14 and 8/15 for [[19,1,5]] code. Results are shown on figures 6.2,6.3 and 6.4.

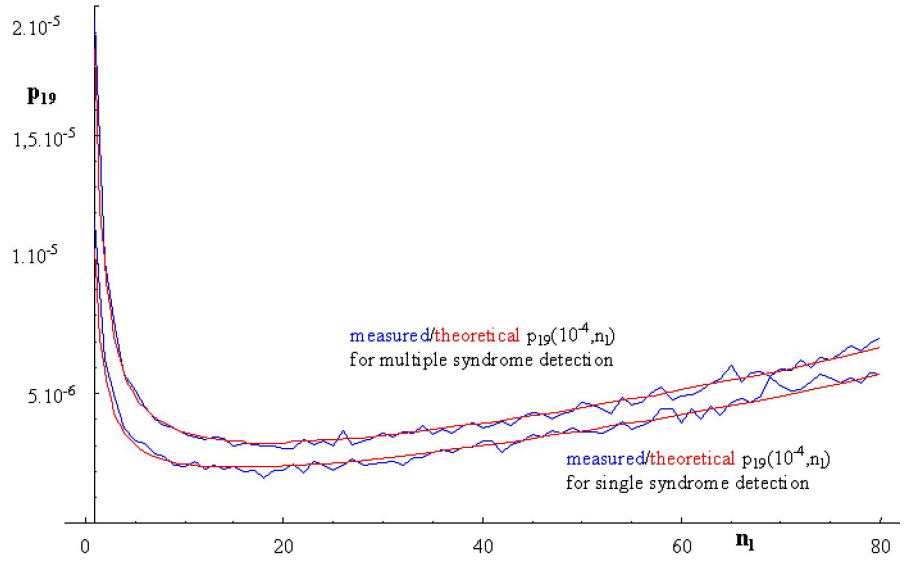


Fig. 6.3: [[19,1,5]] code: Comparison of $\frac{8}{15}p_{19}(10^{-4}, n_l)$ and data obtained from numerical simulations for single syndrome detection and multiple syndrome detection.

6.3.3 Comparison of Steane Code and [[19,1,5]] Code

We have seen in section 6.2 that there is a threshold, of gate failure probability for which [[19,1,5]] code provides better protection of single qubit than Steane code. The threshold

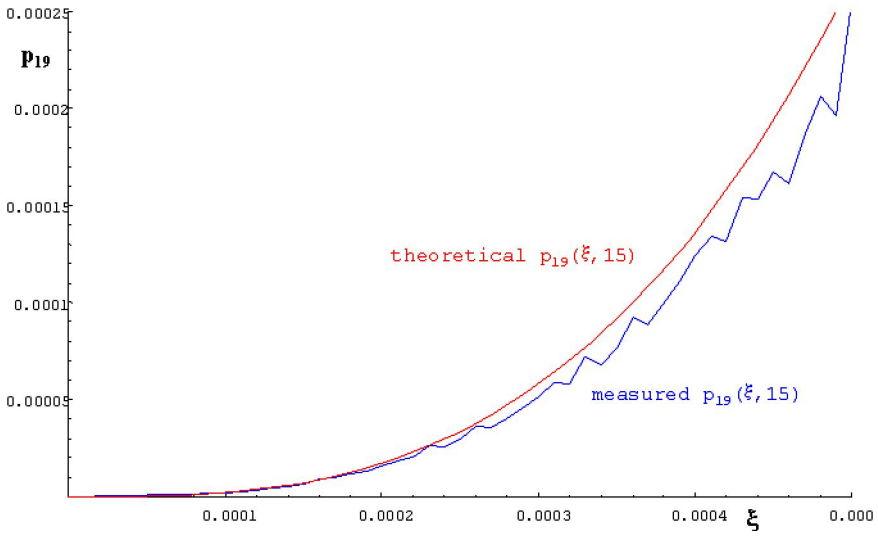


Fig. 6.4: [[19,1,5]] code: Probability of code failure as a function of ξ. Comparison of $\frac{8}{15}p_{19}(\xi, 15)$ and data obtained from numerical simulations for single syndrome detection mode.

may be obtained from simulations and inferred theoretically also in this realistic model of error correction. The theoretical thresholds may be obtained from identities:

$$\mathrm{p}_S(\xi_1, 11) = \xi_1 \tag{6.21}$$

$$\mathrm{p}_{19}(\xi_2, 11) = \xi_2 \tag{6.22}$$

$$\mathrm{p}_S(\xi_3, 11) = \mathrm{p}_{19}(\xi_3, 15), \tag{6.23}$$

where ξ_1 and ξ_2 are gates fault rates at which usage of Steane code and [[19,1,5]]code respectively starts to be useful. At fault rate lower than ξ_3 the [[19,1,5]] code becomes more effective than Steane code. Numerical roots of equations (6.21) - (6.23) are

$$\xi_1 \doteq 2.10^{-3} \tag{6.24}$$

$$\xi_2 \doteq 7.10^{-4} \tag{6.25}$$

$$\xi_3 \doteq 2,5.10^{-4} \tag{6.26}$$

Theoretical threshold ξ_1 comply with the results reported by Zalka in [37]. The dependence of p_S and p_{19} on ξ is shown on figure 6.5, where may be seen that the thresholds ξ_1, ξ_2 and ξ_3 obtained from simulations are approximately the same with the theoretical thresholds. For the failure rate $\xi = 10^{-5}$ we obtained from numerical simulations $p_S(10^{-5}, 11) = 5, 7.10^{-8}$, $p_{19}(10^{-5}, 15) = 5, 6.10^{-9}$ what confirms the observation: The lower failure rate ξ, the better to use [[19,1,5]] code for the data protection.

We have also tried to compare the usage of these codes in more complex computations. Current research in universal quantum gates reports that if random unitary

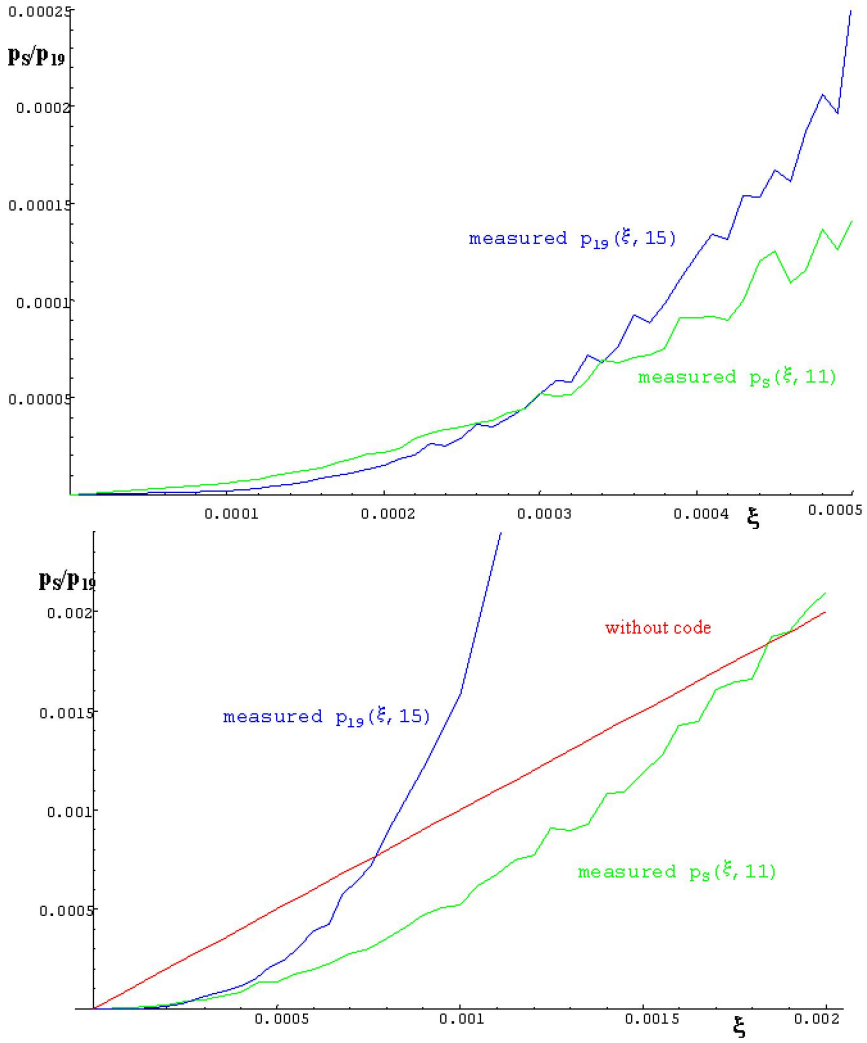


Fig. 6.5: Probability of error on qubit protected with Steane code and [[19,1,5]] code obtained with Monte Carlo simulations.

operator is rewritten to CNOTs and single qubit gates, then approximately 50% of gates are CNOTs. Therefore we simulate the computation of QC in following way: In each computational step a gate is applied on each logical qubit. In 50% of cases the gate is a single qubit gate and in 50% of cases CNOT gate with other logical qubit is applied. After n_l computational steps all logical qubits are corrected from errors. The application of CNOT gates among logical qubits propagate errors among them and thus make the computation more fragile. The results of Monte Carlo simulations of QC consisting of $n = 100$ qubits and different failure rates are shown in the table 6.1. From the table we can see that for smaller ξ the usage of [[19,1,5]] code provides significant improvement over the Steane code. The optimal number of logical operations n_l between two successive error corrections was obtained by binary search through the interval of reasonable values. The value of n_l is 2 or 3 for Steane code and considered

ξ	$3.2\ 10^{-4}$	$1.0\ 10^{-4}$	$3.2\ 10^{-5}$	$1.0\ 10^{-5}$
Steane code	$1.2\ 10^{-2}$	$1.1\ 10^{-3}$	$1.2\ 10^{-4}$	$8.3\ 10^{-6}$
[[19,1,5]] code	$3.4\ 10^{-2}$	$1.3\ 10^{-3}$	$3.6\ 10^{-5}$	$2.4\ 10^{-6}$

Tab. 6.1: Results of numerical simulations of QC consisting of $n = 100$ qubits.

model of QC. For [[19,1,5]] code the optimal number of logical operations between two successive error corrections is in interval $\langle 4, 6\rangle$, it depends on particular value of ξ.

6.4 Summary

The concept of quantum error correction using CSS codes have been introduced in section 3.2.2. Our effort through the remainder of the thesis was paid to the exploration of CSS codes correcting errors on up to two qubits. We have learnt from the work of Steane that such a code would need from 17 to 19 encoding qubits. In section 3.3.1 we proposed a probabilistic algorithm searching for new CSS codes. Using this algorithm we have found CSS code using 19 encoding qubits with minimal distance 5. We have derived theoretical estimates of successfulness of given code using fault tolerant error detection. The theoretical estimates predict that the encoded state would crash with probability of $O(\xi^3)$, where ξ is a probability of single gate failure. We have verified this result by numerical simulations of quantum computations. Moreover, in chapters 3,4,5 we have developed encoding, decoding and recovery circuits for the new code. We have also examined several improvements of new code: either by using multiple syndrome detections, usage of classical code for encoding syndrome bits or reducing number of gates applied to the data qubits. Theoretical estimations and numerical simulations used in this chapter can be used also to verify the successfulness of other quantum codes.

The existence of smaller CSS code correcting two errors remains an open question. The proposed algorithm seems to be week to find any smaller code. We have not tried to prove higher lower bound than the bound given by Steane (17). The future work can be invested in improvement of our model of quantum computer. More complex simulations of quantum computer may be achieved by considering also memory errors, which we have neglected in our simulations. We have also considered just uniform failure rates for quantum states preparations, quantum gates and measurements; this may be extended by introducing separate fault rate for each of previous actions. Considering memory errors, one should also consider that the measurements will probably take more time than quantum gates. However, the theoretical estimations of more complex model would be more complicated, if feasible at all.

APPENDIX

A. GLOSSARY

$\boldsymbol{\alpha^*}$ Complex conjugate of α

$\boldsymbol{A^T}$ transpose of the matrix A

$\boldsymbol{A^\dagger}$ Adjoint matrix to matrix A. Matrix $A^\dagger$ is obtained by transposing matrix A and then complex conjugating elements of A^T

$\boldsymbol{I}$ Identity matrix or identity operator

$\langle \boldsymbol{\phi} | \boldsymbol{\psi} \rangle$ - Inner product of vectors $|\phi\rangle$,$|\psi\rangle$

ancilla qubit An qubit prepared in special known state to act as an auxiliary qubit in quantum computation. Ancilla qubit is often measured at the and of its usage to extract an information about the data in the computation process.

***cat* state** The n-qubit state $1/\sqrt{2}(|0\ldots0\rangle + |1\ldots1\rangle)$. *Cat* states are often used in fault tolerant quantum operations as ancillary qubits.

CSS code Shortcut for Calderbank-Shor-Steane code.A CSS code is formed from two classical error-correcting codes. CSS codes can easily take advantage of results from the theory of classical error-correcting codes and are also well-suited for fault-tolerant quantum computation.

decoherence The process whereby a quantum system interacts with its environment, which acts to change the system. Decoherence is a major cause of errors in quantum computers.

error syndrome A number (or a binary vector), which identifies the error that has

occurred.

fault-tolerance The property of quantum circuits that errors on up to k physical qubits or gates can only result in up to k errors in any given block of an error-correcting code.

leakage error An error in which a qubit leaves the allowed computational space. A leakage error is typically not considered in error correction protocols, but they may be easily detected and converted to located errors.

linear code A classical error correction code whose codewords form a vector space under bitwise addition.

located error A located error is an error which acts on a known (located) qubit in an unknown way.

NMR Nuclear magnetic resonance, method which helps to realize qubits experimentally

quantum error correction code Abbreviated as *QECC*.A *QECC* is a set of quantum states that can be restored to their original state after some number of errors occur.

qubit A single two-state quantum system that serves as the fundamental unit of a quantum computer. The word qubit is shorter name for quantum bit.

stabilizer The set of tensor products of Pauli matrices that fix every state in the coding space. The stabilizer is an subgroup of the group G_n defined in section 4.1. The stabilizer contains all of the vital information about a code.

stabilizer code A quantum code that can be described by giving its stabilizer.

transversal operation An operation applied in parallel to the various qubits in a block of a quantum error correcting code. Qubits from one block can only interact with corresponding qubits from another block or with an corresponding ancilla qubits. Any transversal operation is automatically fault-tolerant.

B. DETAILS FROM QUANTUM COMPUTATION

B.1 Details from Linear Algebra

Tensor Product Formal Definition

Suppose V and W are Hilbert spaces of dimension m and n respectively. Then $V \otimes W$ is an mn dimensional vector space. The elements of $V \otimes W$ are linear combinations of tensor products $|v\rangle \otimes |w\rangle$ of elements $|v\rangle \in V$ and $|w\rangle \in W$. Moreover, if $\{|i\rangle\}$ and $\{|j\rangle\}$ are orthonormal bases for the spaces V and W then $\{|i\rangle \otimes |j\rangle\}$ forms a orthonormal basis for $V \otimes W$. Instead of tensor product $|v\rangle \otimes |w\rangle$ we often use shorter notations $|v\rangle |w\rangle$, $|v, w\rangle$ or $|vw\rangle$.

By definition the tensor product satisfies the following properties:
(1) For an arbitrary scalar a and elements $|v\rangle \in V$ and $|w\rangle \in W$:

$$a(|v\rangle \otimes |w\rangle) = (a\,|v\rangle) \otimes |w\rangle = |v\rangle \otimes (a\,|w\rangle). \tag{B.1}$$

(2) For arbitrary $|v_1\rangle, |v_2\rangle \in V$ and $|w\rangle \in W$:

$$(|v_1\rangle + |v_2\rangle) \otimes |w\rangle = |v_1\rangle \otimes |w\rangle + |v_2\rangle \otimes |w\rangle\,. \tag{B.2}$$

(3) For arbitrary $|v\rangle \in V$ and $|w_1\rangle, |w_2\rangle \in W$:

$$|v\rangle \otimes (|w_1\rangle + |w_2\rangle) = |v\rangle \otimes |w_1\rangle + |v\rangle \otimes |w_2\rangle\,. \tag{B.3}$$

B.2 Proofs

Proof of theorem 1.3 :
Operator acting on single qubit fulfill normalization preserving condition if and only if $U^\dagger U = I$, *where* $U^\dagger$ *is adjoint of* U *(obtained by transposing and then complex conjugating* U*), and* I *is the* 2×2 *identity matrix.*

Proof. The definition of performable operator U is that for the particular normalized $|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ also $U \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ has to be normalized. Let the matrix representation of U be $\begin{pmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{pmatrix}$. After applying U on $|\psi\rangle$ we obtain

$$U|\psi\rangle = \begin{pmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha u_{11} + \beta u_{22} \\ \alpha u_{21} + \beta u_{22} \end{pmatrix}. \tag{B.4}$$

Normalization condition for $U|\psi\rangle$ holds:

$$\begin{aligned} 1 \quad &= (\alpha u_{11} + \beta u_{12})\overline{(\alpha u_{11} + \beta u_{12})} \; + \; (\alpha u_{21} + \beta u_{22})\overline{(\alpha u_{21} + \beta u_{22})} = \qquad \text{(B.5)} \\ = \quad &|\alpha|^2(|u_{11}|^2 + |u_{21}|^2) + |\beta|^2(|u_{12}|^2 + |u_{22}|^2) + \overline{\alpha}\beta(\overline{u_{11}}u_{12} + \overline{u_{21}}u_{22}) + \alpha\overline{\beta}(u_{11}\overline{u_{12}} + u_{21}\overline{u_{22}}) \end{aligned}$$

The (B.5) has to hold for all α, β that satisfy $|\alpha|^2 + |\beta|^2 = 1$. By setting $\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ to (B.5) we obtain:

$$|u_{11}|^2 + |u_{21}|^2 = 1 \tag{B.6}$$

Similarly by setting $\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ in (B.5) we obtain:

$$|u_{12}|^2 + |u_{22}|^2 = 1 \tag{B.7}$$

Adding (B.5), (B.6) and (B.7) all together we obtain the condition

$$\overline{\alpha}\beta(\overline{u_{11}}u_{12} + \overline{u_{21}}u_{22}) + \alpha\overline{\beta}(u_{11}\overline{u_{12}} + u_{21}\overline{u_{22}}) = 0$$

Previous equation can hold for all complex numbers α, β only if

$$\overline{u_{11}}u_{12} + \overline{u_{21}}u_{22} = 0 \tag{B.8}$$

One can easily see that equations (B.6), (B.7) and (B.8) are equivalent with matrix equation $U^{\perp}U = I$. The converse implication is straightforward. □

Proof of lemma 1.7:

Let $\mathcal{H}$ be final dimensional Hilbert space. There exists unique Hermitian conjugate operator $A^{\dagger}$ for every linear operator A on $\mathcal{H}$.

Proof. Let $|u_i\rangle$ be orthonormal basis set for $\mathcal{H}$. Let A_{ij} be element from i-th row and j-th column of matrix representation of operator A. We will show that operator of $A^{\dagger}$

defined by matrix representation $A^{\dagger}_{ij} = A^{*}_{ji}$ fulfills the equality from the definition. Let $|v\rangle = \sum_i \alpha_i |u_i\rangle$ and $|w\rangle = \sum_i \beta_i |u_i\rangle$ be arbitrary vectors from $\mathcal{H}$. It follows that:

$$\begin{aligned}
(|v\rangle, A|w\rangle) &= \left(\sum_i \alpha_i |u_i\rangle, \sum_i \beta_i A|u_i\rangle\right) = \left(\sum_i \alpha_i |u_i\rangle, \sum_i \beta_i \sum_j A_{ji}|u_j\rangle\right) = \\
&= \left(\sum_i \alpha_i |u_i\rangle, \sum_j \sum_i \beta_i A_{ji}|u_j\rangle\right) = \sum_i \alpha_i^* \sum_k \beta_k A_{ik} = \\
&= \left(\sum_i \alpha_i \sum_k A^*_{ik}|u_k\rangle, \sum_k \beta_k |u_k\rangle\right) = \left(\sum_i \alpha_i A^{\dagger}|u_i\rangle, |w\rangle\right) = \left(A^{\dagger}|v\rangle, |w\rangle\right).
\end{aligned} \tag{B.9}$$

We have shown that the operator $A^{\dagger}$ is *Hermitian conjugate* to the operator A. To show uniqueness, suppose that two different operators B,C are *Hermitian conjugate* to the operator A. Then from the definition: $(|u_i\rangle, A|u_k\rangle) = (B|u_i\rangle, |u_k\rangle) = (C|u_i\rangle, |u_k\rangle)$. It can be true only if $B_{ik} = C_{ik} = A^*_{ki}$, what can be easily shown as in existence part of the proof. By altering this equality over all possible couples of i, k we get that $B \equiv C$. □

Proofs of propositions 4.5 and 4.6 are taken from [1] chapter 10.

Proof of proposition 4.5:
Let S be a stabilizer generated by l independent generators $\{g_1, \ldots, g_l\}$ and satisfies $-I \notin S$. Fix i in the range $1, \ldots, l$. Then there exists $g \in G_n$ such that $gg_ig^{\dagger} = -g_i$ and for all $j \neq i$ $gg_jg^{\dagger} = g_j$.

Proof. Let G be the check matrix associated to $g_1, \ldots, g_l$. The rows of G are linearly independent by lemma 4.4, so there exists a $2n$-dimensional vector x such that $G\Lambda x = e_i$, where e_i is l-dimensional vector with a 1 in the ith position and 0s elsewhere. Let g be such that $r(g) = x^T$. Then by definition of x we have $r(g_j)\Lambda r(g)^T = 0$ for $j \neq i$ and $r(g_i)\Lambda r(g)^T = 1$, and thus $gg_ig^{\dagger} = -g_i$ and $gg_jg^{\dagger} = g_j$ for all $j \neq i$. □

Proof of proposition 4.6 :
Let $S =< g_1, \ldots, g_{n-k} >$ be a subgroup of G_n, such that $-I \notin S$ and g_1 through g_{n-k} are independent commuting generators. Then V_S is 2^k dimensional subspace of n-qubit state space.

Proof. Let $x = (x_1, \ldots, x_{n-k})$ be a vector of $n-k$ elements of Z_2. Define

$$P^x_S \equiv \frac{\prod_{j=1}^{n-k} (I + (-1)^{x_j} g_j)}{2^{n-k}} \tag{B.10}$$

Because $(I + g_j)/2$ is the projector onto $+1$ eigenspace of g_j, it is easy to see that $P_S^{(0,\ldots,0)}$ must be the projector onto V_S. By Proposition 4.5 for each x there exists g_x in G_n such that $g_x P_S^{(0,\ldots,0)}(g_x)^\dagger = P_S^x$, and therefore the dimension of P_S^x is the same as the dimension of V_S. Furthermore, for distinct x the P_S^x are easily seen to be orthogonal. The proof is concluded with the algebraic observation that

$$I = \sum_x P_S^x \tag{B.11}$$

The left hand side is a projector onto 2^n-dimensional space, while the right hand side is a sum over 2^{n-k} orthogonal projectors of the same dimension as V_S, and thus the dimension of V_S must be 2^k. □

Lemma B.1. *Let C be a linear code. Then if $x \in C^\perp$, then*

$$\sum_{y \in C} (-1)^{x.y} = |C|, \tag{B.12}$$

while if $x \notin C^\perp$, then

$$\sum_{y \in C} (-1)^{x.y} = 0. \tag{B.13}$$

Proof. First, suppose that $x \in C^\perp$, then from the definition we get $\forall y \in C, x.y = 0$, and thus (B.12) is true. Otherwise, if $x \notin C^\perp$ then from the definition of $C^\perp$ there exists $z \in C$, such that $x.z = 1$. Moreover we use the identity $\{z + y | y \in C\} \equiv C$. Then we count right side of equation (B.12) as

$$\begin{aligned}
\sum_{y \in C} (-1)^{x.y} &= \frac{1}{2}\left(\sum_{y \in C} (-1)^{x.y} + \sum_{y \in C} (-1)^{x.y}\right) && \text{(B.14)}\\
&= \frac{1}{2}\left(\sum_{y \in C} (-1)^{x.y} + \sum_{y \in C} (-1)^{x.(y+z)}\right) && \text{(B.15)}\\
&= \frac{1}{2}\sum_{y \in C} \left((-1)^{x.y} + (-1)^{x.y+1)}\right) && \text{(B.16)}\\
&= \frac{1}{2}\sum_{y \in C} 0 = 0. && \text{(B.17)}
\end{aligned}$$

□

BIBLIOGRAPHY

[1] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, The Edingburg Building, Cambridge CB2 2RU, UK, 2000.

[2] Jozef Gruska. *Quantum Computing*. McGraw-Hill, London, UK, 1999.

[3] A. Hyman. *Charles Babbage : pioneer of the computer*. Oxford, 1982.

[4] R. Marx, A. F. Fahmy, J. M. Myers, and S. J. Glaser W. Bermel. Realization of a 5-bit nmr quantum computer using a new molecular architecture. 1999. `quant-ph/9905087`.

[5] D. G. Cory, W. Mass, M. Price, E. Knill, R. Laflamme, W. H. Zurek, T. F. Havel, and S. S. Somaroo. Experimental quantum error correction. 1998. `quant-ph/9802018`.

[6] J.I. Cirac and P. Zoller. Quantum computations with cold trapped ions. *Phys. Rev. Lett.*, 74:4091–4094, 1995.

[7] C. Monroe, D. M. Meekhof, B. E. King, W. M. Itano, and D. J. Wineland. Demonstaration of a fundamental quantum logic gate. *Phys. Rev. Lett.*, 75:4714–4717, 1995. `http://www.boulder.nist.gov/timefreq/general/pdf/140.pdf`.

[8] Q. A. Turchette, C. J. Hood, W. Lange, H. Mabuchi, and H. J. Kimble. Measurement of conditional phase shifts for quantum logic. *Phys Rev. Lett.*, 75:4710–4713, 1995. `quant-ph/9511008`.

[9] N. Gershenfeld and I Chuang. Bulk spin resonance quantum computation. *Science*, 275:350–356, 1997. `http://www.media.mit.edu/physics/publications/papers/97.01.science.pdf`.

[10] Christopher Gerry and Peter Knight. *Introductory Quantum Optics*. Cambridge University Press, 1982.

[11] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J.Sci.Statist.Comput.*, 26:1484, 1997. quant-ph/9508027.

[12] Martin B. Plenio and Vlatko Vedral. Entanglement in quantum information theory. 1998. quant-ph/9804075.

[13] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Phys. Rev. A.*, 52:3457, 1995. quant-ph/9503016.

[14] Jan Bouda. *Mixed states in quantum information processing.* PhD thesis, FMFI UK, Bratislava, Slovakia, 2003.

[15] J. H. Van Lint. *Introduction to Coding Theory.* Springer verlag, New York, 3rd edition edition, 1998.

[16] W. W. Peterson and E. J. Weldon. *Error Correction Codes.* MIT Press, Cambridge, 2nd edition edition, 1972.

[17] J. H. van Lint. *An Introduction to Coding Theory, 2nd ed.* Springer-Verlag, New York, 1992.

[18] T. Katrinak, J. Smital, M. Gavalec, and E. Gedeonova. *Algebra and Theoretical Arithmetic I.* Alfa, Bratislava, 1985.

[19] J. Adamek. *Foundation of Coding.* John Wiley, Chichester, 1991.

[20] R. E. Blahut. *Theory and practice of error control codes.* Addison Wesley, Moscow, 1986.

[21] W. K. Wootters and W. H. Zurek. A single quantum cannot be cloned. *Nature*, 299:802–803, October 1982.

[22] E. Knill, R. Laflamme, R. Martinez, and C. Negrevergne. Implementation of the five qubit error correction benchmark. 2001. quant-ph/0101034.

[23] Benjamin Schumacher. Sending entanglement through noisy quantum channels. *Phys. Rev. A.*, 54:2614–2628, 1996. http://citeseer.ist.psu.edu/schumacher96sending.html.

[24] Carlton M. Caves. Quantum error correction and reversible operations. *Journal of Superconductivity*, 12:707–718, 1999. quant-ph/9811082.

[25] Pablo Arrighi and Christophe Patricot. The conal representation of quantum states and non trace-preserving quantum operations. *Phys. Rev. A.*, 68, 2003. quant-ph/0212062.

[26] A. R. Calderbank and Peter W. Shor. Good quantum error-correcting codes exist. *Phys. Rev. A*, 54:1098–1105, 1996. quant-ph/9512032.

[27] Andrew M. Steane. Multiple particle inference and quantum error correction. *Proc. Roy. Soc. A.*, 452:2551, May 1996. quant-ph/9601029.

[28] Peter W. Shor. Scheme for reducing decoherence in quantum memory. *Phys Rev. A.*, 52:2493, 1995. http://www.theory.caltech.edu/people/preskill/ph229/shor_error.ps.

[29] Daniel Gottesman. *Stabilizer Codes and Quantum Error Correction*. PhD thesis, California Institute of Technology, Pasadena,CA, May 1997. quant-ph/9705052.

[30] Andrew M. Steane. Efficient fault-tolerant quantum computing. *Nature*, pages 124–126, May 1999. quant-ph/9809054.

[31] Andrew M. Steane. Simple error correcting codes. *Phys. Rev. A*, 54:47414751, December 1996. quant-ph/9605021.

[32] Daniel Gottesman. Class of quantum error-correcting codes saturating the quantum hamming bound. *Phys. Rev. A*, 54:1862, 1996. quant-ph/9604038.

[33] Fault-tolerant quantum computation. In *Synopsium on the Foundations of Computer Science*, Los Alamitos, CA, 1996. IEEE Press. quant-ph/9605011.

[34] John Preskill. Reliable quantum computers. *Proc. Roy. Soc. Lond. A*, 454:385–410, 1998. quant-ph/9705031.

[35] Adriano Bareco, Todd A. Brun, R*ü*diger Schack, and Timothy P. Spiller. Effects of noise on quantum error correction algorithms. 1996. quant-ph/9612047.

[36] Emanuel Knill and Raymond Laflamme. Concatenated quantum codes. 1996. quant-ph/9608012.

[37] Christof Zalka. Threshold estimate for fault tolerant quantum computing. 1997. quant-ph/9612028.

[38] A. M. Steane. Space, time, parallelism and noise requirements for reliable quantum computing. *Fortsch. Phys.*, 46:443–458, 1998. quant-ph/9708021.

[39] Andrew M. Steane. Quantum computing and error correction. *Decoherence and its implications in quantum computation and information transfer*, pages 284–298, 2001. `quant-ph/0304016`.