The Selmer Center
Department of Informatics
University of Bergen
Norway

Master of Science Thesis

On Self-Dual Quantum Codes, Graphs, and Boolean Functions

Lars Eirik Danielsen

March 2005

Abstract

A short introduction to *quantum error correction* is given, and it is shown that *zero-dimensional quantum codes* can be represented as *self-dual additive codes over* GF(4) and also as *graphs*. We show that graphs representing several such codes with high minimum distance can be described as *nested regular graphs* having *minimum regular vertex degree* and containing long cycles. Two graphs correspond to equivalent quantum codes if they are related by a sequence of *local complementations*. We use this operation to generate orbits of graphs, and thus classify all inequivalent self-dual additive codes over GF(4) of length up to 12, where previously only all codes of length up to 9 were known. We show that these codes can be interpreted as *quadratic Boolean functions*, and we define *non-quadratic quantum codes*, corresponding to Boolean functions of higher degree. We look at various cryptographic properties of Boolean functions, in particular the *propagation criteria*. The new *aperiodic propagation criterion* (APC) and the *APC distance* are then defined. We show that the distance of a zero-dimensional quantum code is equal to the APC distance of the corresponding Boolean function. Orbits of Boolean functions with respect to the $\{I,H,N\}^n$ transform set are generated. We also study the *peak-to-average power ratio* with respect to the $\{I,H,N\}^n$ transform set (PAR_{IHN}), and prove that PAR_{IHN} of a quadratic Boolean function is related to the size of the maximum independent set over the corresponding orbit of graphs. A construction technique for non-quadratic Boolean functions with low PAR_{IHN} is proposed. It is finally shown that both PAR_{IHN} and APC distance can be interpreted as partial entanglement measures.

Acknowledgements

I would like to thank my supervisor, Matthew G. Parker, for all his helpful advice and good ideas. I also thank Tor Helleseth and the Selmer Center for financial support enabling me to attend the conference “Sequences and Their Applications”, SETA’04, in Seoul, South Korea, where some of the results in this thesis were presented. Most of the contributions in this thesis are also found in the two papers referenced below.

Danielsen, L. E. and Parker, M. G.: “Spectral orbits and peak-to-average power ratio of Boolean functions with respect to the $\{I,H,N\}^n$ transform”, January 2005. To appear in the proceedings of Sequences and Their Applications, SETA’04, Lecture Notes in Computer Science, Springer-Verlag. `http://www.ii.uib.no/~larsed/papers/seta04-parihn.pdf`.

Danielsen, L. E., Gulliver, T. A., and Parker, M. G.: “Aperiodic propagation criteria for Boolean functions”, October 2004. Submitted to Information and Computation. `http://www.ii.uib.no/~larsed/papers/apc.pdf`.

Lars Eirik Danielsen
Bergen, March 2005

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

In this thesis we will look at a set of objects that can, under suitable interpretations, be represented as

- *zero-dimensional quantum codes*,
- *quantum states*,
- *self-dual additive codes over* GF(4),
- *isotropic systems*,
- *simple undirected graphs*,
- and *quadratic Boolean functions*.

Each interpretation reveals different properties about the underlying objects and suggests different generalisations.

There has been a lot of interest in *quantum computing* since the discovery of Shor's algorithm, which can factor an integer in polynomial time. Practical quantum computers have not yet been built, but it is clear that *quantum error correction* must be a crucial part of any implementation. *Zero-dimensional quantum codes* only represent single *quantum states*, but are still of interest to physicists since codes of high distance represent highly *entangled* states which could be used for testing the decoherence properties of a quantum computer. It has also been shown that a special type of quantum computer can be implemented by performing measurements on a particular class of entangled states.

Zero-dimensional *quantum stabilizer codes* can be represented as *self-dual additive codes over* GF(4). These codes are of interest to coding theorists, and several construction techniques and classifications have been published. A code of this type can be represented by an *isotropic system*, a combinatorical object that has been the subject of much research. A self-dual additive code over GF(4) can also be represented by a *simple undirected graph.* This allows us to use concepts and algorithms from graph theory to characterise the graphs corresponding to strong codes. A generalisation to *hypergraphs* is also suggested by this interpretation.

The same objects are also equivalent to *quadratic Boolean functions*, and the generalisation to Boolean functions of higher degree is natural. Boolean functions are of great interest to cryptographers, since they can be used, for instance, to analyse and construct S-boxes in block ciphers and nonlinear combiners in stream ciphers. Many criteria for the cryptographic strength of Boolean functions exist, and it turns out that zero-dimensional quantum codes with high minimum distance correspond to Boolean functions that satisfy such a criterion. This suggests that highly entangled quantum states may correspond to cryptographically strong Boolean functions. Conversely, various properties derived from transformations of Boolean functions can be interpreted as partial entanglement measures of the corresponding quantum states.

1.2 Overview

The second chapter of this thesis gives a very short introduction to the theory of *quantum computing* and *quantum error correction*. The properties of *superposition* and *entanglement* are explained, and we we show how computer algorithms can be implemented by using transformations and measurements of quantum states. Some of the important discoveries in the theory of quantum computing are also mentioned. A few basic concepts from classical coding theory are presented before we see that error correction is also possible in quantum computers. Although an infinite number of different errors can affect a quantum state, we show that quantum codes only need to consider a small set of basis errors. Quantum codes can be expressed in the *stabilizer formalism*, but have an equivalent representation as *additive codes over* GF(4). We finally introduce the type of codes that will be studied in this thesis, namely quantum codes of dimension zero, called *self-dual quantum codes*, which represent single quantum states. Some bounds on the distance of such codes are presented.

Chapter 3 starts with an introduction to graph theory and defines the notation that we will use. It is then shown that the computer program `nauty` can detect *graph isomorphisms*. By using a simple mapping from *hypergraphs* to ordinary graphs, `nauty` can also detect isomorphism of hypergraphs. A special type of self-dual quantum codes are the *graph codes* which can be represented by undirected graphs. It can be shown that any self-dual quantum code is equivalent to a graph code, and therefore that there is a one-to-one correspondence between the set of simple undirected graphs and the set of self-dual quantum codes. A method for converting any self-dual quantum code into a graph code is described. By exploiting the special form of the generator matrix of a graph code, the distance and *partial weight distribution* of the code can be found by efficient algorithms. The well-known *Quadratic residue construction* can be used to find self-dual quantum codes of high distance. These codes can be represented by a class of *strongly regular* graphs, called *Paley graphs*. A small modification of such a code produces a *bordered quadratic residue code*. We construct quadratic residue codes, and their bordered versions, for all possible lengths up to 30. For length 18, the quadratic residue construction does not give an optimal code, but there is a modified technique that does.

In chapter 4, we look at the graphs corresponding to two well-known self-dual quantum codes, the Hexacode and the Dodecacode. Both codes can be represented by graphs with a special nested structure, which we define as *nested clique graphs*. We show that there is a lower bound on the vertex degree in graphs representing self-dual quantum codes, and that graphs with *minimum regular vertex degree* satisfy this bound with equality. We perform an exhaustive search of all graph codes with circulant generator matrices, for lengths up to 30. Many codes with optimal distance and minimum regular vertex degree are identified, and their nested structures are described. The more general *nested regular graphs* are also defined. We finally discuss the observation that nested regular graphs corresponding to codes of high distance also contain long cycles.

Chapter 5 deals with the equivalence of self-dual quantum codes. We first see that the quantum states represented by equivalent self-dual quantum codes are related by a simple transformation. This transformation corresponds to a simple operation, known as *local complementation*, on the graph representations of the codes. In addition to local complementations, graph isomorphism must be considered, since isomorphic graphs also correspond to equivalent quantum codes. We give three different algorithms for generating *LC orbits*, the equivalence classes of self-dual quantum codes with respect to local complementation and graph isomorphism. By implementing these algorithms, using various optimisation techniques and a cluster computer, we are able to generate all LC orbits of codes of length up to 12. This gives a complete classification of all self-dual additive codes over GF(4) of length up to 12, where previously only all codes of length up to 9 had been classified. A database containing a representative of each LC orbit is also available. We next look at the LC orbits of some strong codes, and search for regular graph structures. The non-existence of any regular graph representation is established for some codes. Finally, we prove that a single LC operation on the graph corresponding to a bordered quadratic residue code produces a regular graph.

Boolean functions are introduced in chapter 6. The *algebraic normal form transformation* and the *Walsh-Hadamard transformation* are defined, and an efficient algorithm for these and other transformations is described. After defining the *periodic autocorrelation*, we see how Boolean functions are used in cryptography. The properties of *correlation immunity*, *resilience*, and *perfect nonlinearity* are of particular interest in this context. We also study the more general *propagation criteria*, and define the new *aperiodic propagation criterion* (APC), which is related to the *aperiodic autocorrelation*. We also define the *APC distance* of a Boolean function. It is explained that Boolean functions can be interpreted as quantum states, and that quadratic Boolean functions correspond to the self-dual quantum codes studied in the previous chapters. Boolean functions of higher degree can be represented by hypergraphs and correspond to a new type of zero-dimensional quantum codes, the *non-quadratic quantum codes*. We see how errors on a quantum state can be expressed as operations on the corresponding Boolean function, and show that the distance of a zero-dimensional quantum code is equal to the APC distance of the corresponding Boolean function. The transform set $\{I, H, N\}^n$ is introduced, and it is shown that the LC orbits of equivalent self-dual quantum codes can be generated by this transform set. Finally, we define two types of orbits of Boolean functions and enumerate all inequivalent functions of up to 5 variables, and all functions of 6 variables with degree up to 3. We also give examples of non-quadratic quantum codes with high APC distance.

In chapter 7, we study another property of Boolean functions and their corresponding graphs and quantum states, namely the *peak-to-average power ratio* with respect to the $\{I, H, N\}^n$ transform (PAR_{IHN}). We calculate the PAR_{IHN} of all quadratic Boolean functions with up to 12 variables, and we prove that the PAR_{IHN} of a quadratic Boolean function equals 2^λ, where λ is the size of the largest independent set in the corresponding LC orbit of graphs. We also define Λ_n, the minimum value of λ over all LC orbits of graphs on n vertices. The values of Λ_n for n up to 14 are given, and bounds on Λ_n are provided for n up to 21. A construction technique for non-quadratic Boolean functions with low PAR_{IHN} is proposed, using good quadratic functions as building blocks.

We also look at PAR with respect to other transform sets, in particular PAR_{IH} and $\mathrm{PAR}_{\mathcal{U}}$, and show that $\mathrm{PAR}_{\mathcal{U}} = \mathrm{PAR}_{IH}$ for quadratic Boolean functions corresponding to bipartite graphs. We show that APC distance and PAR_{IHN} tell something about the degree of entanglement in a quantum state and briefly mention other measures derived from the $\{I, H, N\}^n$ spectrum. We also show that PAR_{IHN} is related to an entanglement measure known as the *Schmidt measure*.

We give some final conclusions and present some open problems and ideas for future research in chapter 8.

While chapter 2 and chapter 3 of this thesis mostly contain previously known results, the later chapters contain many new contributions, and most of these are listed here.

- An exhaustive search of all circulant graph codes of length up to 30 is performed.
- It is shown that many self-dual quantum codes of high distance can be represented by *nested clique graphs* or *nested regular graphs*, and that these graphs also contain long cycles.
- *Minimum regular vertex degree* is defined, and many graphs with this property are identified, corresponding to self-dual quantum codes of high distance.
- All self-dual additive codes over GF(4) of length up to 12 are classified and made available in a database [14]. Previously only all codes of length up to 9 were known. The new numbers of inequivalent codes have been added to *The On-Line Encyclopedia of Integer Sequences* [56].
- It is shown that there are no regular graphs corresponding to [[11,0,5]] or [[18,0,8]] codes, but that *bordered quadratic residue codes* can be transformed into regular graphs by a simple graph operation.
- The *aperiodic propagation criterion* and the *APC distance* of a Boolean function are defined. It is shown that Boolean functions with APC distance d can be interpreted as zero-dimensional quantum codes with distance d.
- We define *non-quadratic quantum codes*, corresponding to hypergraphs and Boolean functions of degree higher than two. Several non-quadratic quantum codes with high distance are found.
- We define two types of orbits of Boolean functions and enumerate all inequivalent functions of up to 5 variables, and all functions of 6 variables with degree up to 3.
- The *peak-to-average power ratio* with respect to the $\{I, H, N\}^n$ transform (PAR_{IHN}) is studied, and it is shown that the PAR_{IHN} of a quadratic Boolean function equals 2^λ, where λ is the size of the largest independent set in the corresponding LC orbit of graphs.
- We define Λ_n, the minimum value of λ over all LC orbits of graphs on n vertices, and give the values of Λ_n for n up to 14. Bounds on Λ_n are provided for n up to 21.
- A construction technique for non-quadratic Boolean functions with low PAR_{IHN} is proposed.

Chapter 2

Quantum Computing and Quantum Codes

2.1 Quantum Computing

2.1.1 Introduction

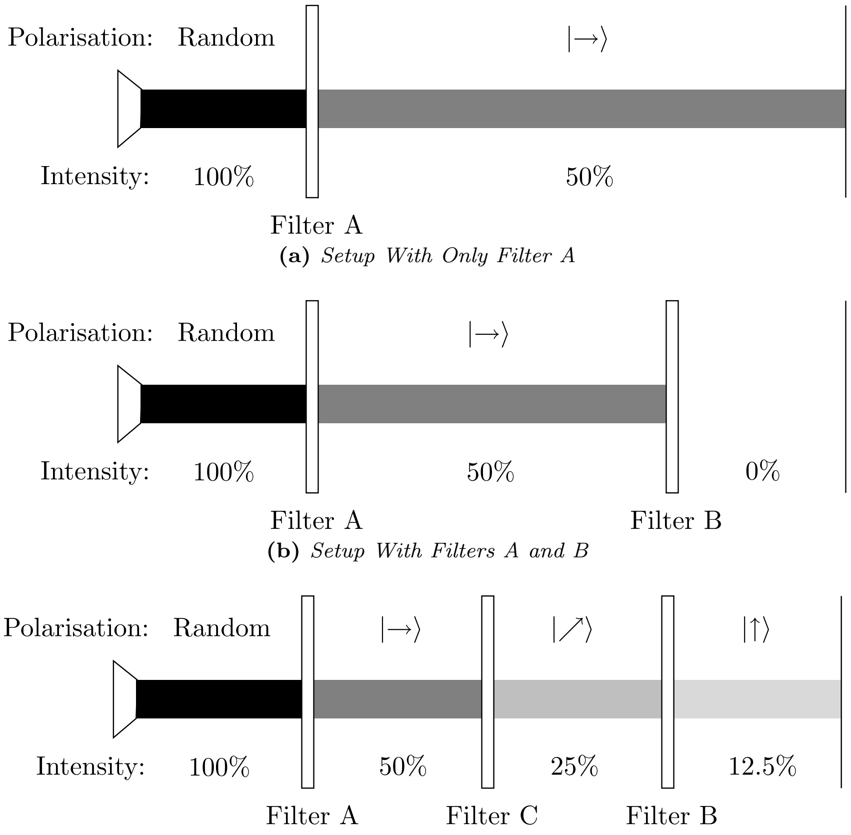
We will only give a brief presentation of quantum computing. For more details, we refer to some of the many good introductions to the topic [33, 36, 50]. Quantum mechanics is a physical theory that describes the behaviour of elementary particles, such as atoms or photons. The laws of quantum mechanics predicts effects which are very different from the physical reality that we ordinarily observe. Of particular interest are the properties of *superposition* and *entanglement*.

2.1.2 Quantum Superposition

A simple experiment demonstrating quantum effects uses the polarisation of light. The light from an ordinary light source consists of photons with a random polarisation. If we put filter A, which has horizontal (0°) polarisation, between the light source and a screen, as shown in Figure 2.1.2, the intensity of the light reflected from the screen will be half of the original, and all photons that pass the filter will now have horizontal polarisation. This can be verified by adding filter B, which has vertical (90°) polarisation, between filter A and the screen. This time, no light reaches the screen at all, as seen in Figure 2.1.2. A most confusing fact is that after adding another filter between A and B, some of the photons do reach the screen. The filter C, with 45° polarisation, is added between filters A and B, and as shown in Figure 2.1.2, we will observe light with $\frac{1}{8}$ of the original intensity reflected from the screen.

What happens is that the randomly polarised photons are "measured" when they hit filter A. Half of them get a horizontal polarisation and pass through, and the other half get a vertical polarisation and are stopped. If the horizontally polarised photons then hit filter B, they will all be stopped. But if they hit filter C they will again be "measured", but now with respect to another basis. Half of them will receive a 45° polarisation and pass through. The other half get an orthogonal (135°) polarisation and are stopped. One fourth of the original photons will pass through both filter A and C. If these photons, which now have a 45° polarisation, then hit filter B, they will again be "measured", and half of them, $\frac{1}{8}$ of the initial amount, will pass through.

The results observed in this experiment are due the property of *quantum superposition.* An object which is in a superposition can be viewed as having



(a) *Setup With Only Filter A*

(b) *Setup With Filters A and B*

(c) *Setup With Filters A, C, and B*

Figure 2.1: *Demonstrating Quantum Effects With Polarisation Filters*

two or more values for an observable quantity at the same time. Once the quantity is measured, the superposition will randomly collapse into one of the values, according to probabilities associated with each possible outcome. A photon could, for instance, have horizontal polarisation with probability a and at the same time vertical polarisation with probability b. When this photon is "measured" by a horizontal polarisation filter, it will with probability a receive horizontal polarisation and pass through the filter, and with probability b receive vertical polarisation and be stopped by the filter.

2.1.3 Bra/Ket Notation

The *bra/ket*-notation invented by Dirac is much used in quantum mechanics. $\langle\phi|$ is a *bra* (the left side of a bracket), and $|\phi\rangle$ is a *ket* (the right side of a bracket). Kets are used to describe states. The state of horizontal polarisation could be described by $|{\rightarrow}\rangle$, and vertical polarisation by $|{\uparrow}\rangle$. A photon which is in a superposition of these states could be described by $\alpha\,|{\rightarrow}\rangle + \beta\,|{\uparrow}\rangle$, where α and β are complex numbers. $|\alpha|^2$ is then the probability of the state collapsing to $|{\rightarrow}\rangle$ upon measurement, and $|\beta|^2$ is the probability of measuring $|{\uparrow}\rangle$. We must have that $|\alpha|^2 + |\beta|^2 = 1$.

Measurement of a quantum state must be done with respect to a specific basis. In the experiment with photons we used the bases $\{|{\rightarrow}\rangle, |{\uparrow}\rangle\}$ and $\{|{\nearrow}\rangle, |{\nwarrow}\rangle\}$. Since $|{\uparrow}\rangle = \frac{1}{\sqrt{2}}\,|{\nearrow}\rangle + \frac{1}{\sqrt{2}}\,|{\nwarrow}\rangle$ and $|{\rightarrow}\rangle = \frac{1}{\sqrt{2}}\,|{\nearrow}\rangle - \frac{1}{\sqrt{2}}\,|{\nwarrow}\rangle$, vertically and horizontally polarised photons will have probability $\frac{1}{2}$ of getting through a filter with 45° polarisation, which is what we observed in the experiment. The negative sign in the expression $|{\rightarrow}\rangle = \frac{1}{\sqrt{2}}|{\nearrow}\rangle - \frac{1}{\sqrt{2}}|{\nwarrow}\rangle$ denotes *phase.* Information about the phase is lost once the superposition collapses, and in this example it can be ignored. With the basis states of a measurement basis we associate orthonormal basis vectors. For example, $|{\rightarrow}\rangle = \binom{1}{0}$ and $|{\uparrow}\rangle = \binom{0}{1}$. The state $\alpha|{\rightarrow}\rangle + \beta|{\uparrow}\rangle$ can then be described by the vector $\binom{\alpha}{\beta}$. Each ket has a corresponding bra, $\langle\phi| = |\phi\rangle^{\dagger}$, where the operator † first conjugates and then transposes a vector, e.g., $\binom{\alpha}{\beta}^{\dagger} = (\overline{\alpha}, \overline{\beta})$. The *inner product,* $\langle\phi|\,|\psi\rangle$ (also written $\langle\phi|\psi\rangle$), is a scalar, and is equal to zero if the vectors associated with $|\phi\rangle$ and $|\psi\rangle$ are orthogonal. The *outer product,* $|\phi\rangle\langle\psi|$, is a matrix which can be used to express transformations on quantum states.

2.1.4 Quantum Bits

A quantum bit, or *qubit*, has two possible states, labelled $|0\rangle$ and $|1\rangle$. All measurements will be done with respect to the basis $\{|0\rangle, |1\rangle\}$. A qubit can be represented by any two-level quantum system. Using vertical and horizontal polarisation of a photon, we could assign $|0\rangle = |{\rightarrow}\rangle$ and $|1\rangle = |{\uparrow}\rangle$, or $|0\rangle = |{\nearrow}\rangle$ and $|1\rangle = |{\nwarrow}\rangle$. Other possible implementations are the up/down spin of an electron or two energy levels of an atom.

Unlike a classical bit, a qubit can be in a superposition of $|0\rangle$ and $|1\rangle$. The state of a general qubit can be denoted $\alpha|0\rangle + \beta|1\rangle$, with $|\alpha|^2$ being the probability of getting the result $|0\rangle$ when measuring the qubit, and $|\beta|^2$ the probability of getting a $|1\rangle$. Several qubits can be combined to form a *quantum register.* The state of a two-qubit register can be denoted $\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$, or equivalently by the vector $(\alpha, \beta, \gamma, \delta)^T$, where $|\alpha|^2$ is the probability of measuring both qubits as zero, and so on. It is also possible to measure only one of the two qubits. If we measure the first qubit, the probability of getting $|0\rangle$ is $|\alpha|^2 + |\beta|^2$, and the probability of getting $|1\rangle$ is $|\gamma|^2 + |\delta|^2$. Upon measurement, the state will collapse, so later measurements of the same qubit will always yield the same value as the first time. If the first qubit is measured as $|0\rangle$, the remaining state is

$$\frac{\alpha}{\sqrt{|\alpha|^2 + |\beta|^2}}|00\rangle + \frac{\beta}{\sqrt{|\alpha|^2 + |\beta|^2}}|01\rangle .$$

If the first qubit is measured as $|1\rangle$, the remaining state is

$$\frac{\gamma}{\sqrt{|\gamma|^2 + |\delta|^2}}|10\rangle + \frac{\delta}{\sqrt{|\gamma|^2 + |\delta|^2}}|11\rangle .$$

If α, β, γ and δ are all equal to $\frac{1}{2}$, the first qubit we measure will be $|0\rangle$ or $|1\rangle$ with probability $\frac{1}{2}$, and the second qubit we measure will also be $|0\rangle$ or $|1\rangle$ with probability $\frac{1}{2}$. But this is not the general case. Consider the state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$. If we measure the first qubit to be $|0\rangle$, the state will collapse to $|00\rangle$, and if we measure the first qubit to be $|1\rangle$, the state will collapse to $|11\rangle$. We see that the value of the second qubit is determined when we measure the first, and that the two qubits will always have the same value. We have observed another fundamental property of quantum mechanics, namely *quantum entanglement.*

2.1.5 The Tensor Product

Definition 2.1. The *tensor product* (also known as the Kronecker product) of the $n \times m$ matrix A, and the $k \times l$ matrix B, gives the $nk \times ml$ matrix

$$A \otimes B = \begin{pmatrix} AB_{0,0} & AB_{0,1} & \cdots & AB_{0,l-1} \\ AB_{1,0} & AB_{1,1} & \cdots & AB_{0,l-1} \\ \vdots & \vdots & \ddots & \vdots \\ AB_{k-1,0} & AB_{k-1,1} & \cdots & AB_{k-1,l-1} \end{pmatrix}, \tag{2.1}$$

where $B_{i,j}$ is the value in row i and column j of B.

The tensor product of $\mathbf{u}$, a column vector of length n, and $\mathbf{v}$, a column vector of length k, is a column vector of length nk,

$$\mathbf{u} \otimes \mathbf{v} = \begin{pmatrix} \mathbf{u}v_1 \\ \mathbf{u}v_2 \\ \vdots \\ \mathbf{u}v_k \end{pmatrix}. \tag{2.2}$$

When we write $|01\rangle$, it is in fact shorthand notation for $|0\rangle \otimes |1\rangle$, where we take the tensor product of the basis vectors associated with the quantum states,

$$|01\rangle = |0\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} 1 \\ \begin{pmatrix} 0 \\ 1 \end{pmatrix} 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}.$$

2.1.6 Quantum Entanglement

An entangled quantum state is a multi-qubit state where the values of the qubits are not independent. There is no classical counterpart to this situation. Qubits that are not entangled can be separated and described independently, using the tensor product. For example, $\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Measuring one of these qubits will not affect the outcome of the other. The state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$, however, can not be factorised in this way, and we have already seen that this is an entangled state.

Classical computers only use the tensor-factorisable space, and a register of n classical bits will at any time be in one of 2^n possible states. A register of n qubits in a quantum computer, however, has a state space defined by 2^n basis vectors, which is an exponentially larger space than in the classical case. The state space also grows exponentially with the number of qubits. It is these properties that give quantum computers their advantage.

A fascinating fact is that one can generate two maximally entangled qubits, $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, (called an EPR pair), and then separate the two particles by an arbitrary distance. When we then measure one of the qubits, the combined state changes instantaneously, and a later measurement of the second qubit will always give the same value as the that of the first qubit. This effect of quantum mechanics was thought to be a paradox, but it has been proved that it is not possible to use entangled particles to communicate faster than the speed of light, so there is no violation of the fundamental laws of physics.

In addition to being entangled with each other, it is possible that qubits in a quantum register could be entangled with the environment, i.e., any particles outside the register. Quantum states that are entangled with the environment are called *mixed states*, and can be described by *density operators*. We will, however, only consider *pure states*, i.e., quantum states that are not entangled with the environment.

2.1.7 Quantum Transformations

Definition 2.2. A matrix U is a *unitary matrix* if $UU^\dagger = I$, where † means conjugate transpose and I is the identity matrix.

Definition 2.3. A matrix that can be written as a tensor product of 2×2 unitary matrices is a *local unitary matrix*.

In addition to measurements, we can perform transformations on quantum states. A quantum transformation must be reversible, and it can be shown that it must therefore be defined by a unitary transformation matrix. Transformations given by local unitary matrices operate independently on each qubit, and therefore do not change the overall entanglement properties of the quantum state. We can think of local unitary transformations as “rotations” which enables us to look at the same quantum state “from another angle”, without changing its properties. In the bra/ket notation, transformations can be described by outer products. For instance,

$$|0\rangle\langle 1| + |1\rangle\langle 0| = \begin{pmatrix}0\\1\end{pmatrix}(1,0) + \begin{pmatrix}1\\0\end{pmatrix}(0,1) = \begin{pmatrix}0 & 0\\1 & 0\end{pmatrix} + \begin{pmatrix}0 & 1\\0 & 0\end{pmatrix} = \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix} \quad (2.3)$$

is the transformation that maps $|0\rangle$ to $|1\rangle$ and $|1\rangle$ to $|0\rangle$,

$$(|0\rangle\langle 1| + |1\rangle\langle 0|)(\alpha|0\rangle + \beta|1\rangle) = \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}\begin{pmatrix}\alpha\\\beta\end{pmatrix} = \begin{pmatrix}\beta\\\alpha\end{pmatrix} = \alpha|1\rangle + \beta|0\rangle . \quad (2.4)$$

This is called the *bit-flip* or “not” transformation. It is easy to verify that it is unitary and self-inverse.

A local unitary transformation on an n-qubit quantum state is given by a tensor product of n 2×2 unitary matrices,

$$U = U_0 \otimes U_1 \otimes \cdots \otimes U_{n-1}. \quad (2.5)$$

To express that the transform U_0 should be applied to qubit number i, we can write $U_0^{(i)}$. The transformation that applies U_0 to the ith qubit and U_1 to all other qubits is then

$$U = U_0^{(i)} \bigotimes_{j \neq i} U_1^{(j)}. \quad (2.6)$$

Note the factors must be placed in the correct order before this tensor multiplication can be carried out.

Definition 2.4. We define the *Pauli matrices*,

$$\sigma_x = \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}, \quad \sigma_z = \begin{pmatrix}1 & 0\\0 & -1\end{pmatrix}, \quad \sigma_y = \begin{pmatrix}0 & -i\\i & 0\end{pmatrix}, \quad I = \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}.$$

The Pauli matrices is a useful set of quantum transformations. σ_x represents a *bit-flip*, σ_z is a *phase-flip*, and σ_y is a combination of both, since $\sigma_y = i\sigma_x\sigma_z$. The factor i in the definition of σ_y makes some manipulations easier, but in most cases the overall phase factor of a quantum state can be ignored. We also include the identity matrix, I, which makes no change to the qubit it is applied to. We will later see that Pauli matrices can be used to represent errors on quantum states.

Definition 2.5. The *Hadamard* transformation is defined by the matrix

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} .$$

Observe that by applying H, we transform the state $|0\rangle$ into the superposition $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, while the state $|1\rangle$ is transformed into $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. The *Walsh-Hadamard* transformation applies H to every qubit of a state. If we apply the Walsh-Hadamard transformation to an n-qubit all-zero state, $|00\cdots 0\rangle$, we get a superposition of all the 2^n basis states, each with the same probability.

We have already seen that measurements of quantum states destroy much of the information the states contain. It is also easy to show that it is impossible to make perfect copies of a quantum state, since there is no unitary, and thus non-destructive, transformation which performs this copying.

Theorem 2.6 (Dieks, and Wootters and Zurek)**.** *A quantum state can not be cloned, i.e., there is no operation that takes $|\phi\rangle$ to $|\phi\phi\rangle$, where $|\phi\rangle$ is any quantum state.*

Proof. Assume that such an operation exists, and let $|\phi\rangle$ and $|\psi\rangle$ be two distinct quantum states. Then the cloning operation gives

$$|\phi\rangle \to |\phi\phi\rangle \, , \tag{2.7}$$

$$|\psi\rangle \to |\psi\psi\rangle \, , \tag{2.8}$$

$$|\phi\rangle + |\psi\rangle \to (|\phi\rangle + |\psi\rangle) \otimes (|\phi\rangle + |\psi\rangle) = |\phi\phi\rangle + |\psi\psi\rangle + |\phi\psi\rangle + |\psi\phi\rangle \, . \tag{2.9}$$

But, since all quantum operations must be linear, it follows from (2.7) and (2.8) that

$$|\phi\rangle + |\psi\rangle \to |\phi\phi\rangle + |\psi\psi\rangle \, , \tag{2.10}$$

which is a contradiction of (2.9). □

2.1.8 Quantum Computers

The idea of using quantum mechanical effects to perform computations was first introduced by Feynman in the 1980s, when he discovered that classical computers could not simulate all aspects of quantum physics efficiently. In 1985, Deutsch showed that it is possible to implement any function which is computable by a classical computer using registers of entangled qubits and arrays of *quantum gates*, each performing a unitary quantum transformation.

The advantage of quantum computers, compared to classical computers, is the property of *quantum parallelism.* We have seen that an n-qubit quantum register can be in a superposition of all its 2^n basis states. A function of n variables, implemented by an array of quantum gates, can therefore be applied to all the basis states simultaneously, and the result will be a superposition of the function's 2^n possible outputs. If we try to measure the result directly, the superposition will collapse, and we will only observe one random value of the function, which is not very useful. The advantage of quantum computers comes from the discovery that appropriate transformations on a superposition of states enables us to observe a common property of all the states. This makes it possible, for instance, to find the period of a function by applying the function once to a superposition of all possible input values. Another way to make use of quantum parallelism is to use transformations that amplify the probability of desired results.

Shor's algorithm, discovered in 1994, can factor an integer in polynomial time. For classical computers, all known algorithms require a running time that grows exponentially with the number of bits in the integer to be factored. Interest in quantum computing increased with the discovery of Shor's algorithm, since the security of many popular public-key cryptography schemes is based on the assumed infeasibility of factoring large integers. The factoring problem can be reduced to the problem of finding the period of a function. In Shor's algorithm, this is accomplished by applying the *quantum Fourier transform* to a superposition of all values of the function.

Grover's algorithm, discovered in 1996, can be used to search for an element in an unsorted list in running time of order $O(\sqrt{n})$. Classical computers can not do better than $O(n)$. This is another example of the advantages of quantum computing, although not as impressive as the exponential gain of Shor's algorithm. Grover's algorithm finds a value for which a given statement is true. This is done by evaluating the statement for a superposition of all possible values, and then repeatedly using a transformation that increases the probability of the state that satisfies the statement. When we finally read the value of the quantum register, we will with very high probability observe the desired state.

Many different techniques for the construction of quantum computers are being researched, but the best implementations so far only operate on 2 or 3 qubits. Although there are many interesting theoretical results about quantum computers, a practical and scalable implementation is not possible with the technology available today.

Quantum computing should not be confused with the concept of *quantum key distribution*, although both exploit the property of quantum superposition. In quantum key distribution, a sequence of qubits, typically represented by the polarisation of photons, is sent over an insecure quantum channel. Only the sender knows which basis each qubit is encoded with. It is impossible for an eavesdropper to clone the qubits, and if he tries to measure one using the wrong basis, its state will change. Eavesdropping can later be detected, when the choices of encoding bases are made public. Quantum key distribution requires much simpler technology than quantum computing. Working systems for quantum key distribution, using up to 150 kilometres of optical cables, have been successfully implemented.

2.2 Classical Error Correction

We here give a short introduction to some basic concepts of error correcting codes that will be useful when we later discuss quantum error correction.

Definition 2.7. Let $\mathcal{A}$ be an alphabet, and let $\mathcal{A}^n$ be the set of all n-tuples of elements from $\mathcal{A}$. A *code*, $\mathcal{C}$, over $\mathcal{A}$ of length n is a subset of $\mathcal{A}^n$, $\mathcal{C} \subset \mathcal{A}^n$.

A code maps a vector $\boldsymbol{v}$ of k symbols to a vector $\boldsymbol{u}$ of n symbols, called a codeword, where $n > k$. Any alphabet of symbols may be chosen, but the binary alphabet $\{0, 1\}$, where the symbols are called bits, is often used. The $n-k$ extra symbols added by the encoding process provides redundancy. If a codeword is changed by a transmission error, this redundancy may enable us to determine the original codeword, or at least to detect that an error occurred. An error in this context is an operation that change one or more symbols of a codeword into other symbols. If the binary alphabet is used, an error flips the value of one or more bits, $0 \mapsto 1$ or $1 \mapsto 0$.

Definition 2.8. The code $\mathcal{C}$ can be defined by a $k \times n$ matrix G, called a *generator matrix*, where $\mathcal{C} = \{\boldsymbol{v}G \mid \boldsymbol{v} \in \mathcal{A}^k\}$.

Encoding is a simple process once we know G, the generator matrix of a code $\mathcal{C}$, since the codeword corresponding to $\boldsymbol{v}$ is $\boldsymbol{u} = \boldsymbol{v}G$. The rows of G are called the basis codewords of $\mathcal{C}$, since any codeword is a linear combination of these rows.

Definition 2.9. Let $\mathcal{C}$ be a code over the alphabet $\mathcal{A}$, where $\mathcal{A} = \mathrm{GF}(q)$ is a finite field, and let G be the generator matrix of $\mathcal{C}$. If all linear combinations of the rows of G are codewords in $\mathcal{C}$, then $\mathcal{C}$ is a vector space and a k-dimensional subspace of $\mathrm{GF}(q)^n$. A code that fulfils these criteria is called a *linear code.*

Definition 2.10. Let $\mathcal{C}$ be a code over a finite field with generator matrix G. If any sum of the rows of G, i.e., any GF(2)-linear combination, is a codeword in $\mathcal{C}$, and all codewords in $\mathcal{C}$ are GF(2)-linear combinations of the rows of G, then $\mathcal{C}$ is an *additive code.* If the binary alphabet is used, all additive codes are linear codes, but this is not true for the general case.

Definition 2.11. The code $\mathcal{C}$ can also be defined by an $(n-k) \times n$ matrix H, called the *parity check matrix* of $\mathcal{C}$. $\mathcal{C} = \{\boldsymbol{u} \in \mathcal{A}^n \mid \boldsymbol{u}H^T = \boldsymbol{0}\}$, where $\boldsymbol{0}$ is the all-zero vector.

Given the parity check matrix H of a code $\mathcal{C}$, it is easy to check whether a vector $\boldsymbol{u}$ is codeword by checking if $\boldsymbol{u}H^T = \boldsymbol{0}$. If we receive a vector that does not satisfy this criteria, we know that an error has occurred. If the codeword $\boldsymbol{u}$ is transmitted and $\boldsymbol{u'}$ is the received vector, then we can write $\boldsymbol{u'} = \boldsymbol{u} + \boldsymbol{e}$, where $\boldsymbol{e}$ is the transmission error. It is easy to verify that $\boldsymbol{u'}H^T = \boldsymbol{u}H^T + \boldsymbol{e}H^T = \boldsymbol{0} + \boldsymbol{e}H^T = \boldsymbol{e}H^T$. We see that the value of $\boldsymbol{u'}H^T$ only depends on the error $\boldsymbol{e}$, and we therefore call this value the *syndrome* of $\boldsymbol{e}$. Given a set of errors with distinct syndromes, we can determine which of the errors has occurred by using the syndrome calculated from the received vector.

Definition 2.12. The *Hamming weight* of a vector $\boldsymbol{a}$ of length n, denoted $w_H(\boldsymbol{a})$, is the number of non-zero coordinates of $\boldsymbol{a}$, i.e., $w_H(\boldsymbol{a}) = |\{a_i \neq 0 \mid i \in \mathbb{Z}_n\}|$, where a_i is the ith coordinate of $\boldsymbol{a}$.

Definition 2.13. The *Hamming distance* between two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, both of length n, denoted $d(\boldsymbol{a}, \boldsymbol{b})$, is the number of coordinates where the two vectors have different values, i.e., $d(\boldsymbol{a}, \boldsymbol{b}) = |\{a_i \neq b_i \mid i \in \mathbb{Z}_n\}|$.

Definition 2.14. The *minimum distance* of a code $\mathcal{C}$, denoted $d(\mathcal{C})$, is the smallest number of symbol errors needed to change one codeword into another, i.e., $d(\mathcal{C}) = \min\{d(\boldsymbol{a}, \boldsymbol{b}) \mid \boldsymbol{a}, \boldsymbol{b} \in \mathcal{C}, \boldsymbol{a} \neq \boldsymbol{b}\}$.

Proposition 2.15. *A code can detect s errors if $d(\mathcal{C}) \geq s + 1$. A code can correct t errors if $d(\mathcal{C}) \geq 2t + 1$.*

Definition 2.16. A code of length n containing M codewords and having minimum distance d is called an (n, M, d) code. For linear codes, the notation $[n, k, d]$ may also be used, where k is the dimension of the code. The number of codewords in a linear code over $\mathrm{GF}(q)$ is then q^k.

Proposition 2.17. *The distance of a linear code $\mathcal{C}$ can easily be found as $d(\mathcal{C}) = \min\{w_H(\boldsymbol{u}) \mid \boldsymbol{u} \in \mathcal{C} \backslash \{\boldsymbol{0}\}\}$, i.e., the weight of the minimum weight non-zero codeword in $\mathcal{C}$.*

Definition 2.18. Every code $\mathcal{C}$ over $\mathrm{GF}(q)$ has a *dual code*, $\mathcal{C}^\perp = \{\boldsymbol{u} \in \mathrm{GF}(q) \mid \boldsymbol{u} \cdot \boldsymbol{c} = 0, \forall \boldsymbol{c} \in \mathcal{C}\}$. If $\mathcal{C} \subseteq \mathcal{C}^\perp$, then $\mathcal{C}$ is a *self-orthogonal* code. If $\mathcal{C} = \mathcal{C}^\perp$, then $\mathcal{C}$ is a *self-dual* code.

If $\mathcal{C}$ has generator matrix G and parity check matrix H, then the dual code, $\mathcal{C}^\perp$, has generator matrix H and parity check matrix G.

2.3 Quantum Error Correction

2.3.1 Introduction

For more detailed information about quantum error correction, we refer to some of the many introductions to the subject [25, 32, 36].

A major problem for the implementation of quantum computers is that it is impossible to totally isolate a few qubits from the rest of the world. The qubits will rapidly interact with the environment, and entanglement will be destroyed in a process known as *decoherence*. Because of decoherence, the state of a quantum register will not remain stable for long enough time to do any useful computations. We have seen that we can not observe a quantum state without destroying entanglement, and that we can not make copies of it. Surprisingly, it was shown by Steane and Shor that quantum error correction is still possible. It can even be shown that it is possible to process quantum information arbitrarily accurately, given that the effects of decoherence can be kept under a certain threshold for each step of the computation.

A classical bit can only have one of two values, 0 or 1, and the only possible error is a bit-flip. A qubit has a continuous state space, since α and β in the expression $\alpha|0\rangle + \beta|1\rangle$ can take any complex values. Since any 2×2 unitary matrix describes a possible transformation, an infinite number of different errors may affect a single qubit.

Proposition 2.19. *The set of Pauli matrices, introduced in Definition 2.4, span the space of* 2×2 *unitary matrices. Any error on a single qubit,* $|\phi\rangle \to E|\phi\rangle$*, may therefore be expressed as a linear combination of the Pauli matrices,*

$$|\phi\rangle \to (aI + b\sigma_x + c\sigma_y + d\sigma_z)|\phi\rangle = a|\phi\rangle + b\sigma_x|\phi\rangle + c\sigma_y|\phi\rangle + d\sigma_z|\phi\rangle . \quad (2.11)$$

We will see that the error correction process causes the superposition in (2.11) to collapse into one of four states, so that we observe no error with probability $|a|^2$, a bit-flip error with probability $|b|^2$, a phase-flip error with probability $|c|^2$, and a combined bit-flip and phase-flip error with probability $|d|^2$. The process will also determine which Pauli error has occurred. We can then recover the state $|\phi\rangle$ by applying the same Pauli transformation, since all the Pauli matrices are self-inverse. This procedure is performed without observing the state $|\phi\rangle$ directly, but by comparing the values of several qubits. A comparison of two qubits can be done without learning the value of either qubit, and therefore without collapsing their superpositions.

As in the classical case, quantum error correction is done by adding redundant qubits which are used to detect or correct errors. A quantum code encodes k qubits using n qubits. It has 2^k basis codewords, but any linear combination of those is also a valid codeword, since the code must be able to encode all superpositions of the basis states. We assume that errors affect each qubit independently, which may in reality not be the case. We describe the errors by error operators, which are tensor products of Pauli matrices. The *weight* of an error operator is the number of positions in which it is different from identity. For instance, the error operator $I \otimes \sigma_x \otimes \sigma_z \otimes I \otimes I$ has weight 2. Note in particular that a combined bit-flip and phase-flip error only count as one error. If the errors described by all Pauli error operators of weight up to t can be corrected by a code, then the code can correct an arbitrary error affecting up to t qubits. If a code should be able to correct the two errors E_a and E_b, then the code must be able to tell the difference between $E_a |\phi_i\rangle$ and $E_b |\phi_j\rangle$, the two errors operating on two different basis codewords. To guarantee that this is possible, the vectors corresponding to the states $E_a |\phi_i\rangle$ and $E_b |\phi_j\rangle$ must be orthogonal.

Example 2.20. The repetition code is a simple classical code that encodes a bit by making a number of copies of it. Decoding is achieved by a majority rule. Quantum coding is not that easy, since qubits can not be copied, but it is possible to encode, for instance, one qubit using three qubits by mapping the basis states $|0\rangle$ to $|000\rangle$ and $|1\rangle$ to $|111\rangle$. The state $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$ would in that case be encoded into $|\psi\rangle = \alpha|000\rangle + \beta|111\rangle$. Note that these three qubits are highly entangled, and not three independent copies of $|\phi\rangle$. This code can correct any single bit-flip, but does not correct phase-flips. Consider the error $(\sigma_x \otimes I \otimes I)|\psi\rangle = \alpha|100\rangle + \beta|011\rangle$. We can not observe the value of any qubit, but it is possible to compare two qubits and learn if they have the same value. Comparing the first and second qubit tells us that they are different, so one of them must be wrong. When we find that the second and third qubits are equal, we know that the error is in the first qubit, assuming only one error has occurred. We bit-flip the first qubit to correct the error. This code is also able to correct any linear combination of single bit-flip error operators. Consider, for instance, the error

$$(\frac{3}{5}\sigma_x \otimes I \otimes I + \frac{4}{5}I \otimes \sigma_x \otimes I)|\psi\rangle = \frac{3}{5}(\alpha|100\rangle + \beta|011\rangle) + \frac{4}{5}(\alpha|010\rangle + \beta|101\rangle).$$

When we compare the values of the qubits, this superposition will collapse into $\alpha|100\rangle + \beta|011\rangle$ with probability $\frac{9}{25}$ and $\alpha|010\rangle + \beta|101\rangle$ with probability $\frac{16}{25}$. The results of the comparisons will be according to the chosen state, and the error correction proceeds as in the previous case.

In order to learn what error we must correct, a number of extra qubits, known as an *ancilla*, are added temporarily. After appropriate transformations, we may read a *syndrome*, which tells us what error has occurred, from these qubits. It is in fact the act of measuring the syndrome that collapses the superposition of errors into a single error.

Example 2.21. A code that can correct both bit-flips and phase-flips is Shor's "nine-qubit repetition code". This code maps $|0\rangle$ to $(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)$ and $|1\rangle$ to $(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)$. Bit-flips are corrected by the inner layer of this code, by exactly the same procedure as in Example 2.20. By comparing the signs of the three outer blocks, we may correct any single phase-flip. The two steps are actually independent, so both one phase-flip and one bit-flip can always be corrected. Note that a phase-flip on the first qubit followed by a phase-flip on the second qubit, i.e., the error operator $\sigma_z \otimes \sigma_z \otimes I \otimes \cdots \otimes I$, leaves a codeword unchanged. The code need not be able to correct this error, nor to tell which qubit in each block of three has been affected in case of a phase-flip. Codes where any error operator has this property are called *degenerate codes.*

Definition 2.22. The minimum distance, d, of a quantum code, is the minimum weight error operator that gives an errored state not orthogonal to the original state, and therefore not guaranteed to be detectable.

It follows from Proposition 2.15 that a quantum code with $d \geq s+1$ can detect s errors, and that a quantum code with $d \geq 2t+1$ can correct t errors.

Definition 2.23. A quantum code that encodes k qubits using n qubits and have distance d is called an $[[n, k, d]]$ code.

The double brackets helps us distinguish a quantum code from a classical code. The nine-qubit code described in Example 2.21 is a $[[9, 1, 3]]$ code.

2.3.2 Stabilizer Codes

An $[[n, k, d]]$ quantum code can be described by a *stabilizer* given by a set of $n-k$ error operators. Such codes are called *stabilizer codes* [24, 25].

Consider the nine-qubit code from Example 2.21. When we compare the two first qubits to detect a possible bit-flip in one of them, what we really do is to measure the eigenvalue of the operator $M_1 = \sigma_z \otimes \sigma_z \otimes I \otimes \cdots \otimes I$, i.e., we find the value m in $M_1 |\phi\rangle = m |\phi\rangle$. If the qubits have the same value, then the result is $+1$, and otherwise it is -1. To compare the first two three-qubit blocks to detect a phase-flip in one of them, we measure the eigenvalue of $\sigma_x \otimes \sigma_x \otimes \sigma_x \otimes \sigma_x \otimes \sigma_x \otimes \sigma_x \otimes I \otimes I \otimes I$. The complete stabilizer for Shor's nine-qubit code is given by the 8 operators,

$$
\begin{aligned}
M_1 &= \sigma_z \otimes \sigma_z \otimes I \otimes I \otimes I \otimes I \otimes I \otimes I \otimes I,\\
M_2 &= I \otimes \sigma_z \otimes \sigma_z \otimes I \otimes I \otimes I \otimes I \otimes I \otimes I,\\
M_3 &= I \otimes I \otimes I \otimes \sigma_z \otimes \sigma_z \otimes I \otimes I \otimes I \otimes I,\\
M_4 &= I \otimes I \otimes I \otimes I \otimes \sigma_z \otimes \sigma_z \otimes I \otimes I \otimes I,\\
M_5 &= I \otimes I \otimes I \otimes I \otimes I \otimes I \otimes \sigma_z \otimes \sigma_z \otimes I,\\
M_6 &= I \otimes I \otimes I \otimes I \otimes I \otimes I \otimes I \otimes \sigma_z \otimes \sigma_z,\\
M_7 &= \sigma_x \otimes \sigma_x \otimes \sigma_x \otimes \sigma_x \otimes \sigma_x \otimes \sigma_x \otimes I \otimes I \otimes I,\\
M_8 &= I \otimes I \otimes I \otimes \sigma_x \otimes \sigma_x \otimes \sigma_x \otimes \sigma_x \otimes \sigma_x \otimes \sigma_x.
\end{aligned}
$$

The error we want to detect *anticommutes* with the operator we actually measure, since the Pauli operators anticommute, i.e., $AB = -BA$, where $A, B \in \{\sigma_x, \sigma_y, \sigma_z\}$ and $A \neq B$. For a valid codeword, it must be true for all i that $M_i |\phi\rangle = |\phi\rangle$, i.e., that the eigenvalue of all operators is $+1$. If a correctable error has occurred, the set of operators that give eigenvalues -1 will identify the error. Consider, for instance, the error $E = \sigma_x \otimes I \otimes \cdots \otimes I$, a bit-flip error on the first qubit, which takes $|\phi\rangle$ to $E|\phi\rangle$. E will anticommute with M_1, so $M_1 E |\phi\rangle = -E M_1 |\phi\rangle = -E|\phi\rangle$, and the resulting eigenvalue is -1. Likewise, E will anticommute with M_2, but will commute with the other six operators. This gives us a set of eigenvalues uniquely identifying the error E.

The *stabilizer*, S, is an Abelian group generated by the set of $n-k$ operators. (An Abelian group is a group where all elements commute.) S consists of all operators M for which $M|\phi\rangle = |\phi\rangle$ for all codewords $|\phi\rangle$. Two given errors can be corrected if there exists an operator in S that can distinguish them, i.e., measuring the eigenvalue of the operator gives different values for the two errors. Let the *centraliser* of S, $C(S)$, be the set of errors that commute with all the $n-k$ generators of S. $C(S) \backslash S$ is then the set of errors that are not detectable. Hence, the distance, d, of a stabilizer code is the minimum weight of any operator in $C(S) \backslash S$.

In Example 2.21, we studied a $[[9, 1, 3]]$ stabilizer code. This code does not represent an optimal way of encoding one qubit with the possibility of correcting any single error. In fact, there exists a $[[5,1,3]]$ code. The 4 operators generating its stabilizer are given by the following matrix, each row corresponding to one operator.

$$S = \begin{pmatrix} \sigma_x & \sigma_z & \sigma_z & \sigma_x & I \\ I & \sigma_x & \sigma_z & \sigma_z & \sigma_x \\ \sigma_x & I & \sigma_x & \sigma_z & \sigma_z \\ \sigma_z & \sigma_x & I & \sigma_x & \sigma_z \end{pmatrix}$$

An alternate representation of the stabilizer S uses two binary matrices, the bit-flip matrix X and the phase-flip matrix Z. Let $X_{i,j} = 1$ when $S_{i,j} = \sigma_x$ or $S_{i,j} = \sigma_y$, and $X_{i,j} = 0$ otherwise. Let $Z_{i,j} = 1$ when $S_{i,j} = \sigma_z$ or $S_{i,j} = \sigma_y$, and $Z_{i,j} = 0$ otherwise. We combine the two matrices to make the $n \times 2n$ binary stabilizer matrix $S_b = (Z \mid X)$. For the $[[5,1,3]]$ code, we get

$$S_b = \left(\begin{array}{ccccc|ccccc} 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{array}\right).$$

2.3.3 Quantum Codes over GF(4)

Proposition 2.24 (Calderbank et al. [10]). *We can consider a quantum error correcting code as an additive code over the finite field* GF(4)*, by identifying the four Pauli matrices with the elements of* GF(4)*. We denote* $\mathrm{GF}(4) = \{0, 1, \omega, \omega^2\}$*, where* $\omega^2 = \omega + 1$*. The mappings used are* $I \mapsto 0$*,* $\sigma_z \mapsto 1$*,* $\sigma_x \mapsto \omega$*, and* $\sigma_y \mapsto \omega^2$*.*

As an example, the $[[5,1,3]]$ code previously described can be represented by the additive code over GF(4) generated by the matrix

$$C = \begin{pmatrix} \omega & 1 & 1 & \omega & 0 \\ 0 & \omega & 1 & 1 & \omega \\ \omega & 0 & \omega & 1 & 1 \\ 1 & \omega & 0 & \omega & 1 \end{pmatrix}.$$

Definition 2.25. *Conjugation* in GF(4) is defined by $\overline{x} = x^2$. The *trace map*, $\mathrm{tr} : \mathrm{GF}(4) \mapsto \mathrm{GF}(2)$, is defined by $\mathrm{tr}(x) = x + \overline{x}$. The *trace inner product* of two vectors of length n over GF(4), $\boldsymbol{u}$ and $\boldsymbol{v}$, is given by $\boldsymbol{u} * \boldsymbol{v} = \sum_{i=1}^{n} tr(u_i \overline{v_i})$.

In addition to replacing the symbols we use, we must make sure that the properties of a stabilizer code are preserved in a code over GF(4). A stabilizer is a group generated by $n - k$ operators. This corresponds to an additive subset of $GF(4)^n$, generated by $n - k$ vectors. The stabilizer is an Abelian group, which means that any two operators in the stabilizer commute. The corresponding property of an additive code over GF(4), $\mathcal{C}$, is that any two codewords, $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{C}$, must have trace inner product $\boldsymbol{u} * \boldsymbol{v} = 0$. This is equivalent to saying that the code must be *self-orthogonal* with respect to the trace inner product, or that $\mathcal{C} \subseteq \mathcal{C}^\perp$, where $\mathcal{C}^\perp = \{\boldsymbol{u} \in \mathrm{GF}(4)^n \mid \boldsymbol{u} * \boldsymbol{c} = 0, \forall \boldsymbol{c} \in \mathcal{C}\}$. If the stabilizer S corresponds to $\mathcal{C}$, then the centraliser $C(S)$, the set of errors that commute with all generators of S, corresponds to $\mathcal{C}^\perp$. The set of undetectable errors, $C(S) \backslash S$, corresponds to $\mathcal{C}^\perp \backslash \mathcal{C}$. Hence, the weight of the minimum weight non-zero vector in $\mathcal{C}^\perp \backslash \mathcal{C}$ is the distance of a quantum code over GF(4).

2.3.4 Self-Dual Quantum Codes

The codes studied in this thesis will be of the special case where the dimension $k = 0$. A zero-dimensional stabilizer code with high distance represents a single quantum state which is robust to error, sometimes called a *stabilizer state.* Codes of higher dimension can be constructed from zero-dimensional quantum codes, but identifying stabilizer states is also an interesting application in itself, since the states corresponding to codes of high distance will be highly entangled. Highly entangled quantum states could be used for testing the decoherence properties of a quantum computer, and it has also been shown that a *one-way quantum computer* can be implemented by performing measurements on a particular class of entangled states, known as *cluster states* [8, 48, 49]. An $[[n,0,d]]$ code is nondegenerate by definition, and is generated by an $n \times n$ generator matrix, corresponding to an $(n, 2^n, d)$ classical code. The GF(4)-representation of such codes will be *self-dual*, i.e., $\mathcal{C} = \mathcal{C}^{\perp}$, and we therefore call zero-dimensional quantum codes of this type *self-dual quantum codes.* The distance of a self-dual quantum code is simply the minimum distance of $\mathcal{C}$, i.e., the weight of the minimum weight codeword in $\mathcal{C}$.

Example 2.26. As an example, consider the self-dual quantum code with generator matrix

$$C = \begin{pmatrix} \omega & 0 & 0 & 1 & 1 & 1 \\ 0 & \omega & 0 & \omega^2 & 1 & \omega \\ 0 & 0 & \omega & \omega^2 & \omega & 1 \\ 0 & 1 & 0 & \omega & \omega^2 & 1 \\ 0 & 0 & 1 & \omega & 1 & \omega^2 \\ 1 & \omega^2 & 0 & \omega & 0 & 0 \end{pmatrix}.$$

There are 64 GF(2)-linear combinations of the 6 rows of C. In addition to the all-zero codeword, we have 45 codewords of weight 4 and 18 of weight 6. This is therefore a $[[6,0,4]]$ code.

Definition 2.27. We distinguish between two types of self-dual quantum codes. A code is of *type II* if all codewords have even weight, otherwise it is of *type I.* It can be shown that a type II code must have even length.

Theorem 2.28 (Rains and Sloane [47])**.** *Let d_I be the minimum distance of a type I code of length n. Then d_I is upper-bounded by*

$$d_I \le \begin{cases} 2\left\lfloor \frac{n}{6} \right\rfloor + 1, & \textit{if } n \equiv 0 \ (\textit{mod } 6) \\ 2\left\lfloor \frac{n}{6} \right\rfloor + 3, & \textit{if } n \equiv 5 \ (\textit{mod } 6) \\ 2\left\lfloor \frac{n}{6} \right\rfloor + 2, & \textit{otherwise.} \end{cases} \tag{2.12}$$

There is a similar bound on d_{II}, the distance of a type II code of length n,

$$d_{II} \le 2\left\lfloor \frac{n}{6} \right\rfloor + 2. \tag{2.13}$$

A code that meets the appropriate bound is called *extremal.* Calderbank et al. also use a linear programming bound [10] on the distance of self-dual quantum codes and give a table of the best bounds. This table has later been extended by Grassl [27]. For some lengths, no code meeting the best upper bound on distance has been discovered, so it remains uncertain whether such a code exists. In particular, for $n = 24$, the best known self-dual quantum code has distance 8, while the upper bound is 10. Let d_m be the highest attainable distance for self-dual additive codes over GF(4). (Non-additive codes [46] with higher distance may exist.) Table 2.1 shows, for lengths up to 30, the values of d_I, d_{II} and d_m. Note that type II codes where the length is a multiple of 6, i.e., 6, 12, 18, 24 and 30, are particularly strong codes.

Table 2.1: *Bounds on the Distance of Self-Dual Quantum Codes*

n	d_I	d_{II}	d_m
2	2	2	2
3	2		2
4	2	2	2
5	3		3
6	3	4	4
7	4		3
8	4	4	4
9	4		4
10	4	4	4
11	5		5
12	5	6	6
13	6		5
14	6	6	6
15	6		6
16	6	6	6
17	7		7
18	7	8	8
19	8		7
20	8	8	8
21	8		8
22	8	8	8
23	9		8–9
24	9	10	8–10
25	10		8–9
26	10	10	8–10
27	10		9–10
28	10	10	10
29	11		11
30	11	12	12

Chapter 3

Quantum Codes and Graphs

3.1 Introduction to Graph Theory

A **graph** is a pair $G = (V, E)$, where $V = \{v_0, v_1, \ldots, v_{n-1}\}$ is a set of n **vertices** (or **nodes**), and E is a set of distinct pairs of elements from V, i.e., $E \subseteq V \times V$. A pair $\{v_i, v_j\} \in E$ is called an **edge**. We will only consider **undirected graphs**, which are graphs where E is a set of distinct *unordered* pairs of elements from V. Furthermore, the graphs we will look at will all be **simple graphs**, which are graphs with no *self-loops*, $\{v_i, v_i\} \notin E$. A graph $G' = (V', E')$ that satisfies $V' \subseteq V$ and $E' \subseteq E$ is a **subgraph** of G, denoted $G' \subseteq G$. Given a subset of vertices $A \subseteq V$, the **induced subgraph** $G(A) \subseteq G$ has vertices A and edges $\{\{v_i, v_j\} \in E \mid v_i, v_j \in A\}$, i.e., all edges from E whose endpoints are both in A. The **complement graph** $\overline{G}$ has vertices $\overline{V} = V$ and edges $\overline{E} = V \times V - E$, i.e., the edges in E are changed to non-edges, and the non-edges to edges.

Two **isomorphic** graphs are structurally equal, but the labelling of the vertices may differ. More formally, two graphs $G = (V, E)$ and $G' = (V, E')$ are isomorphic iff there exists a permutation π of V such that $\{v_i, v_j\} \in E \iff \{\pi(v_i), \pi(v_j)\} \in E'$.

A graph may be represented by an **adjacency matrix** Γ. This is a $|V| \times |V|$ matrix where $\Gamma_{i,j} = 1$ if $\{v_i, v_j\} \in E$, and $\Gamma_{i,j} = 0$ otherwise. For simple graphs, the adjacency matrix must have 0s on the diagonal, i.e., $\Gamma_{i,i} = 0$. The adjacency matrix of an undirected graph will be *symmetric*, i.e., $\Gamma_{i,j} = \Gamma_{j,i}$.

Two vertices are called **adjacent** (or **neighbours**) if they are joined by an edge. The **neighbourhood** of a vertex v, denoted N_v, is the set of vertices that are adjacent to v. The **vertex degree** (or **valency**) of a vertex is the number of neighbours it has. A **regular graph** is a graph where all vertices have the same degree. A regular graph where all vertices have degree k is called a ***k*-regular graph**. We will also denote any k-regular graph on n vertices R_n^k. Note that R_n^k does not uniquely define a graph; there may be several non-isomorphic graphs R_n^k for the same value of k and n. A **strongly regular graph** [11] with parameters (n, k, λ, μ) is a k-regular graph on n vertices, with the additional property that any two adjacent vertices have λ common neighbours, and any two non-adjacent vertices have μ common neighbours. An example of a strongly regular graph with parameters $(10, 3, 0, 1)$ is the well-known *Petersen graph*, shown in Figure 3.1.

A **complete graph** is a graph where all pairs of vertices are connected by an edge. The complete graph on n vertices has $\binom{n}{2}$ undirected edges. A **clique** is

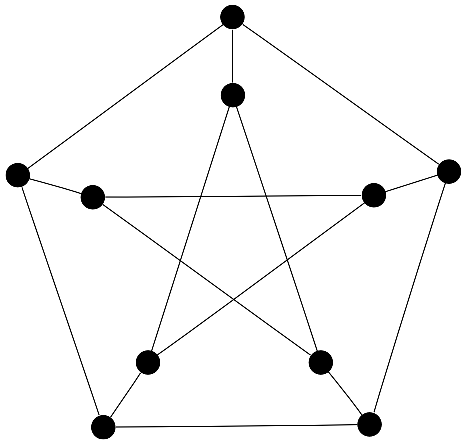


Figure 3.1: *The Strongly Regular Petersen Graph*

a complete subgraph. A k-clique is a clique consisting of k vertices. We will use the notation K_n for both complete graphs on n vertices and n-cliques. Note that $K_n = R_n^{n-1}$ and that K_n does define a unique graph, up to isomorphism. An **independent set** is the complement of a clique, i.e., a subgraph with no edges. The **independence number**, $\alpha(G)$, is the size of the largest independent set in G. A **bipartite graph** is a graph where the vertices can be partitioned into two independent sets, i.e., $V = A \cup B$ where the induced subgraphs $G(A)$ and $G(B)$ both contain no edges.

A **path** is a sequence of vertices $(u_0, u_1, \ldots, u_{k-1})$ where $\{u_0, u_1\}, \{u_1, u_2\}, \ldots, \{u_{k-2}, u_{k-1}\} \in E$. A **connected graph** is a graph where there is path from every vertex to all other vertices. A **simple path** never visits the same vertex more than once. A **cycle** (or **circuit**) is a path $(u_0, u_1, \ldots, u_{k-1}, u_0)$, i.e., a path that starts and ends at the same vertex. A **simple cycle** never visits the same vertex more than once, except the first vertex, which is also visited last. Let C_n denote the graph consisting of only a simple cycle on n vertices. Note that $C_n = R_n^2$ and that C_n does define a unique graph, up to isomorphism. A **Hamiltonian path** is a simple path that visits every vertex of the graph once. If there is also an edge between the first and the last vertex of a Hamiltonian path, we have a **Hamiltonian cycle**.

A **hypergraph**, $G = (V, E)$, is a generalised graph where an edge may connect more than two vertices. An edge, $e \in E$, of a hypergraph is given by a set of at least two vertices, $e = \{u_0, u_1, \ldots, u_{k-1}\}$. Edges on more than two vertices are called **hyperedges**.

3.2 Graph Isomorphism with nauty

Determining whether two graphs are isomorphic is considered to be a hard problem, but an efficient algorithm has been developed by McKay and implemented in the program `nauty` [35]. `nauty` can also produce a *canonical representative* of a graph. The canonical representative is isomorphic to the original graph, but may have a different vertex labelling. This labelling is arbitrary with no special properties, but it is chosen in a consistent way such that all isomorphic graphs will have the same canonical representative. `nauty` also includes a utility called `geng` which can generate all non-isomorphic graphs on a given number of vertices.

Checking for hypergraph isomorphism is not directly supported by `nauty`, but `nauty` can detect isomorphism of graphs where the vertices have been divided into a set of disjoint partitions, $V = P_1 \cup P_2 \cup \cdots \cup P_k$. Two such partitioned graphs are isomorphic if their partitions are of the same sizes, and if a relabelling of the vertices of one of the graphs produces the other, with the restriction that labels can only be exchanged within partitions. There is one-to-one mapping between a hypergraph, $G = (V, E)$, on n vertices with m hyperedges and an ordinary graph, $G' = (V', E')$, on $n + m$ vertices with partitions of size n and m. We first add all vertices in V to V' and all simple graph edges in E to E'. For each hyperedge, $e_i \in E$, $0 \leq i < m$, we add an extra vertex, v_{n+i}, to V'. If $e_i = \{u_0, u_1, \ldots, u_{k-1}\}$, we add the k edges $\{u_0, v_{n+i}\}, \ldots, \{u_{k-1}, v_{n+i}\}$ to E'. The canonical representative of G' is found by `nauty`, and the result is mapped back to a hypergraph, which is the canonical representative of G.

Example 3.1. We have the hypergraph $G = (V, E)$ with $V = \{v_0, v_1, v_2, v_3\}$ and $E = \{\{v_0, v_1, v_2\}, \{v_1, v_2, v_3\}, \{v_1, v_2\}, \{v_1, v_3\}\}$. We map this hypergraph to the graph G' on 6 vertices with edges $E' = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_0, v_4\}, \{v_1, v_4\}, \{v_2, v_4\}, \{v_1, v_5\}, \{v_2, v_5\}, \{v_3, v_5\}\}$, where the vertices are partitioned into the sets $\{v_0, v_1, v_2, v_3\}$ and $\{v_4, v_5\}$. We use `nauty` to find the canonical labelling of this partitioned graph, and we then map the resulting graph to the hypergraph $G'' = (V, E'')$, where $E'' = \{\{v_0, v_2, v_3\}, \{v_1, v_2, v_3\}, \{v_1, v_3\}, \{v_2, v_3\}\}$. Any hypergraph isomorphic to G will also have canonical representative G''.

3.3 Graph Codes

Definition 3.2. A *graph code* is a self-dual additive code over GF(4) with generator matrix $C = \Gamma + \omega I$, where I is the identity matrix and Γ is the adjacency matrix of a simple undirected graph, which must be symmetric with 0s along the diagonal.

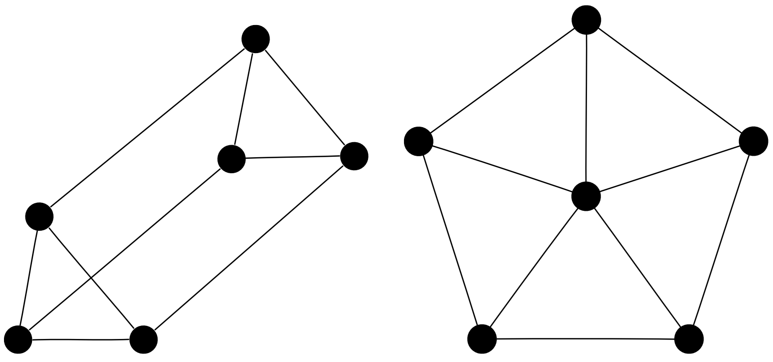
Example 3.3. Consider the graph shown in Figure 3.2a. This graph has adjacency matrix

$$\Gamma = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

The corresponding self-dual additive code over GF(4) is generated by the matrix

$$C = \Gamma + \omega I = \begin{pmatrix} \omega & 1 & 1 & 1 & 0 & 0 \\ 1 & \omega & 1 & 0 & 1 & 0 \\ 1 & 1 & \omega & 0 & 0 & 1 \\ 1 & 0 & 0 & \omega & 1 & 1 \\ 0 & 1 & 0 & 1 & \omega & 1 \\ 0 & 0 & 1 & 1 & 1 & \omega \end{pmatrix}.$$

The same code can also be described using stabilizer formalism. The stabilizer



(a) *The "2-clique of 3-cliques"* **(b)** *The "Wheel Graph"*

Figure 3.2: *Two Graph Representations of the [[6,0,4]] Hexacode*

code is generated by operators given by the rows of the matrix

$$S = \begin{pmatrix} \sigma_x & \sigma_z & \sigma_z & \sigma_z & I & I \\ \sigma_z & \sigma_x & \sigma_z & I & \sigma_z & I \\ \sigma_z & \sigma_z & \sigma_x & I & I & \sigma_z \\ \sigma_z & I & I & \sigma_x & \sigma_z & \sigma_z \\ I & \sigma_z & I & \sigma_z & \sigma_x & \sigma_z \\ I & I & \sigma_z & \sigma_z & \sigma_z & \sigma_x \end{pmatrix}.$$

Stabilizer codes of this type are known as graph codes, and the single quantum states they encode are called *graph states.*

Schlingemann and Werner [54] studied quantum codes associated with graphs, and first proved the following theorem. Briegel and Raussendorf [8] had previously studied arrays of entangled particles, which can be modelled by graphs.

Theorem 3.4 (Schlingemann and Werner [54], Grassl et al. [28], Glynn [22], and Van den Nest et al. [59])**.** *For any self-dual quantum code, there is an equivalent graph code. This means that there is a one-to-one correspondence between the set of simple undirected graphs and the set of self-dual additive codes over* GF(4)*.*

It follows from Theorem 3.4 that, without loss of generality, we can restrict our study of self-dual additive codes over GF(4) to those with generator matrices of the form $\Gamma + \omega I$.

Van den Nest et al. [59] describe the following algorithm for transforming any stabilizer code into a graph code. We will operate on the transpose of the binary stabilizer matrix, $T = S_b^T = \binom{Z^T}{X^T} = \binom{A}{B}$. It is easy to see that a graph code given by the adjacency matrix Γ corresponds to the binary stabilizer $S_b = (\Gamma \mid I)$, and to the transpose binary stabilizer $T = \binom{\Gamma}{I}$. Our goal is to convert $T = \binom{A}{B}$, the transpose binary stabilizer of a given code, into $T' = \binom{A'}{I}$, the transpose binary stabilizer of an equivalent graph code. A' will then be the adjacency matrix of the corresponding graph. Right-multiplying T with an invertible $n \times n$ matrix will perform a basis change, an operation that gives us an equivalent stabilizer code. If B is an invertible matrix, we can simply multiply T by the inverse of B and get $TB^{-1} = \binom{AB^{-1}}{I}$. AB^{-1} will then be the resulting adjacency matrix. If this matrix has elements on the diagonal that are not 0, those elements may simply be changed to 0. In some cases B may not be invertible. It has been proved by Van den Nest et al. [59] that T can then always be transformed into an equivalent code $T' = \binom{A'}{B'}$, where B' is invertible. They also show how the appropriate transformation is found.

Example 3.5. We are given the following generator matrix of a $[[6,0,4]]$ stabilizer code.

$$S = \begin{pmatrix} \sigma_x & I & I & \sigma_z & \sigma_z & \sigma_z \\ I & \sigma_x & I & \sigma_y & \sigma_z & \sigma_x \\ I & I & \sigma_x & \sigma_y & \sigma_x & \sigma_z \\ I & \sigma_z & I & \sigma_x & \sigma_y & \sigma_z \\ I & \sigma_z & \sigma_z & I & \sigma_x & \sigma_x \\ \sigma_z & \sigma_z & I & \sigma_z & I & \sigma_x \end{pmatrix}$$

The corresponding binary stabilizer is

$$S_b = (Z \mid X) = \left(\begin{array}{cccccc|cccccc} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right).$$

Let $A = Z^T$ and $B = X^T$. The transpose binary stabilizer is then $T = \binom{A}{B}$. Since B is invertible,

$$TB^{-1} = \begin{pmatrix} AB^{-1} \\ I \end{pmatrix} = \left(\begin{array}{cccccc} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right).$$

We set the nonzero diagonal element in AB^{-1} to 0 and get the adjacency matrix of the simple undirected graph shown in Figure 3.2b.

3.4 Efficient Algorithms for Graph Codes

We have seen that a graph code, $\mathcal{C}$, is a self-dual additive code over GF(4) whose generator matrix is of the form $\Gamma + \omega I$. It can be shown that the additive code over $\mathbb{Z}_4$ given by $2\Gamma + I$ has the same weight distribution as $\mathcal{C}$. For graph codes, but not in the general case, we may therefore replace the elements from GF(4) with elements from $\mathbb{Z}_4$ by the mappings $0 \mapsto 0$, $1 \mapsto 2$, $\omega \mapsto 1$, $\omega^2 \mapsto 3$.

Example 3.6. A self-dual additive code over GF(4) generated by

$$\begin{pmatrix} \omega & 0 & 1 & 0 & 1 \\ 0 & \omega & 0 & 1 & 0 \\ 1 & 0 & \omega & 0 & 1 \\ 0 & 1 & 0 & \omega & 0 \\ 1 & 0 & 1 & 0 & \omega \end{pmatrix},$$

has the same weight distribution as the additive code over $\mathbb{Z}_4$ generated by

$$\begin{pmatrix} 1 & 0 & 2 & 0 & 2 \\ 0 & 1 & 0 & 2 & 0 \\ 2 & 0 & 1 & 0 & 2 \\ 0 & 2 & 0 & 1 & 0 \\ 2 & 0 & 2 & 0 & 1 \end{pmatrix}.$$

The interpretation as a code over $\mathbb{Z}_4$ is an advantage when we write computer programs to operate on such codes, since $\mathbb{Z}_4$ arithmetic may be faster and simpler to implement.

Proposition 3.7. *Let $\mathcal{C}$ be a self-dual additive code over* GF(4) *with generator matrix $C = \Gamma + \omega I$. Let $\boldsymbol{s} \in \mathcal{C}$ be a codeword formed by adding k different rows of C. Then it must be true that $w_H(\boldsymbol{s}) \geq k$.*

Proof. Each row of C has an element ω, and this element is in a different position in each row. All other elements in C are 0 or 1. Let row number i of C be one of the rows we added to get $\boldsymbol{s}$. Element s_i will then be $a_0 + a_1 + \cdots + a_k$, where $a_i = \omega$ and $a_j \in \{0, 1\}$, $\forall j \neq i$. It follows that $s_i \in \{\omega, \omega + 1\}$. $\boldsymbol{s}$ will have k elements of the same form and therefore $w_H(\boldsymbol{s}) \geq k$. □

The special form of the generator matrix of a graph code makes it easier to find the distance of the code. An $[[n, 0, d]]$ code has 2^n codewords, but if the generator matrix is given in graph form, it is not necessary to check all the codewords to find the distance of the code. If we have found a codeword $\boldsymbol{s}$, where $w_H(\boldsymbol{s}) \leq e$, we know that no codeword formed by adding e or more rows of the generator matrix can have lower weight. This fact is used in Algorithm 3.1. A similar technique can also be used to find the weight distribution of a code. To find w_p, the number of codewords of weight p, only codewords formed by adding p or fewer rows of the generator matrix needs to be considered. This approach is used by Algorithm 3.2. To find the complete weight distribution, $\boldsymbol{w} = \{w_0, w_1, \ldots, w_n\}$, we must generate all codewords, but a *partial weight distribution*, $\boldsymbol{w}_p = \{w_0, w_1, \ldots, w_p\}$, where $p < n$, can be found more efficiently. We will later use the partial weight distribution to distinguish inequivalent codes.

3.5 Quadratic Residue Codes

Definition 3.8. A *Paley graph*, $G = (V, E)$, is constructed as follows. Given a prime power m, such that $m \equiv 1 \pmod 4$, let the elements of the finite field GF(m) be the set of vertices, V. Let two vertices, i and j, be joined by an edge, $\{i, j\} \in E$, iff their difference is a quadratic residue (square) in GF$(m)\backslash\{0\}$, i.e., there exists an $x \in$ GF$(m)\backslash\{0\}$ such that $x^2 \equiv i - j$.

Proposition 3.9. *A Paley graph is a strongly regular graph [11] with parameters $(4t + 1, 2t, t - 1, t)$, i.e., it has $4t + 1$ vertices, each with degree $2t$, and the properties that any two adjacent vertices have $t - 1$ common neighbours and any two non-adjacent vertices have t common neighbours.*

We will study graph codes based on Paley graphs. Some bounds on the distance of self-dual quantum codes constructed from strongly regular graphs in general have been given by Tonchev [57].

Algorithm 3.1 Finding the Distance of a Graph Code

```
Input    $C$: a generator matrix in graph form
Output   $d$: the distance of the code generated by $C$

procedure FindDistance($C$)
    $d \leftarrow \infty$
    $i \leftarrow 1$
    while i $< d$ do
        for all codewords $\boldsymbol{s}$, such that $\boldsymbol{s}$ is a sum of $i$ rows do
            if $w_H(\boldsymbol{s}) < d$ then
                $d \leftarrow w_H(\boldsymbol{s})$
                if $d = i$ then
                    return $d$
                end if
            end if
        end for
        $i \leftarrow i + 1$
    end while
    return $d$
end procedure
```

Algorithm 3.2 Finding the Number of Codewords of a Given Weight

```
Input    $C$: a generator matrix in graph form
         $p$: weight of the codewords we want to count
Output   $w_p$: the number of codewords of weight $p$

procedure CountWeight($C$, $p$)
    $w_p \leftarrow 0$
    for i $\leftarrow$ 0 to p do
        for all codewords $\boldsymbol{s}$, such that $\boldsymbol{s}$ is a sum of $i$ rows do
            if $w_H(\boldsymbol{s}) = p$ then
                $w_p \leftarrow w_p + 1$
            end if
        end for
    end for
    return $w_p$
end procedure
```

Definition 3.10. A self-dual additive code over GF(4) with generator matrix $\Gamma + \omega I$, where Γ is the adjacency matrix of a Paley graph, is a type of *quadratic residue code* [23, 43].

When p is a prime, the adjacency matrix of the Paley graph on GF(p) will be circulant, with each row being the cyclic shift of a *Legendre sequence*. Let QR be the set of all quadratic residues modulo p. a is a quadratic residue modulo p iff $a \not\equiv 0 \pmod{p}$ and the congruence $y^2 \equiv a \pmod{p}$ has a solution $y \in \mathbb{Z}_p$. The Legendre sequence of length p, $\boldsymbol{l}_p = (l_0, l_1, \ldots, l_{p-1})$, is a binary sequence with $l_i = 1$ if $i \in QR$, and $l_i = 0$ otherwise. Let $\boldsymbol{l}_p \gg s$ be a Legendre sequence cyclically shifted s times to the right. Form the $p \times p$ matrix Γ by letting row i be $\boldsymbol{l}_p \gg i$, for $0 \leq i < p$. It can be shown that p must be a prime of the form $4k+1$ for Γ to be symmetric, which is a requirement for the adjacency matrix of an undirected graph.

Definition 3.11. To get a *bordered quadratic residue code* [23, 43] of length $m+1$, first construct the quadratic residue code of length m. Then add a top row of m 1s, $(1, 1, \ldots, 1)$, to the generator matrix. Finally, add a leftmost column with an ω followed by m 1s, $(\omega, 1, 1, 1, \ldots, 1)^T$, to the generator matrix.

Example 3.12. We will construct the quadratic residue code of length 5 and bordered quadratic residue code of length 6. The quadratic residues modulo 5 are $QR = \{1, 4\}$, and the Legendre sequence of length 5 is $\boldsymbol{l} = (0, 1, 0, 0, 1)$. From this sequence we construct the matrix

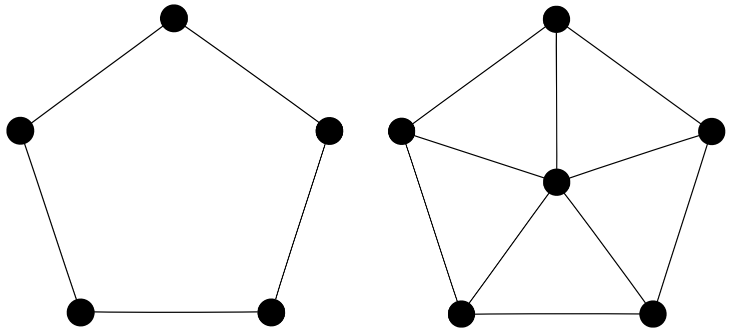
$$\Gamma = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

This is the adjacency matrix of the Paley graph on 5 vertices, which is the graph C_5, shown in Figure 3.3a. $\Gamma + \omega I$ is the generator matrix of a $[[5, 0, 3]]$ quantum code. We border the matrix and get

$$\Gamma' = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix},$$

the adjacency matrix of the "wheel graph", shown in Figure 3.3b, which represents the extremal $[[6, 0, 4]]$ code, also known as the *Hexacode.*

The integers $m \leq 30$ where m is a prime of the form $4k+1$ are 5, 13, 17, and 29. The quadratic residue codes for $m = 5$, 13 and 29, and their bordered extensions, achieve the highest possible distance, as given by Table 2.1 on page 18. For $m = 17$ the construction gives a $[[17, 0, 5]]$ code, but there exists a code with distance 7. The bordered extension is a $[[18, 0, 6]]$ code, when distance 8 is achievable. It can be shown (under some constraints) that there exists a unique $[[18, 0, 8]]$ code [4]. This code was constructed by MacWilliams et al. [34] using a construction similar to bordered quadratic residue codes. Glynn et al. [23] present the following technique for constructing the $[[18, 0, 8]]$ quantum code. For a prime $p = 4k+1$, generate a set, K, of all powers of



(a) *The C_5 Graph* **(b)** *The "Wheel Graph"*

Figure 3.3: *Graphs of the QR and BQR Codes for $m = 5$*

4 modulo p. For instance, for $p = 5$, $K = \{4^0 = 1, 4^1 = 4\}$, which is also the set of quadratic residues modulo 5. In general, if 2 is a primitive root modulo p, we will construct a quadratic residue code. This is the case for $p = 5$, 13 and 29, but for $p = 17$ we get $K = \{4^0 = 1, 4^1 = 4, 4^2 = 16, 4^3 = 13\}$. We then find a partition of $\mathbb{Z}_{17}\backslash\{0\}$ into sets that are multiples of K, $\mathbb{Z}_{17}\backslash\{0\} = K \cup 2K \cup 3K \cup 6K$, where $2K = \{2, 8, 15, 9\}$, $3K = \{3, 12, 14, 5\}$, and $6K = \{6, 7, 11, 10\}$. We must combine two of these sets to make the set H, which must satisfy $2H + H = \mathbb{Z}_{17}\backslash\{0\}$. The valid combinations are $H = K \cup 3K$ and $H = K \cup 6K$. Note that the quadratic residue set, $K \cup 2K$ is not valid. The two valid combinations generate the sequences $(0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1)$ and $(0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1)$. We make two circulant adjacency matrices by cyclically shifting the sequences, in exactly the same way as we did with Legendre sequences. The two matrices correspond to two $[[17, 0, 7]]$ codes, which can be shown to be equivalent. Bordering either of the matrices will generate the $[[18, 0, 8]]$ code.

Paley graphs and quadratic residue codes can be constructed for any prime power $m = p^n$ where $m \equiv 1 \pmod 4$. The only possible non-prime lengths below 30 are 9 and 25. The generator matrices of these codes are not circulant, but they are composed of circulant submatrices. When $n = 2$, there will be p^2 $p \times p$ circulant matrices. For $m = 9$, we generate the adjacency matrix

$$\left(\begin{array}{ccc|ccc|ccc}
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\
1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
\hline
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
\hline
0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0
\end{array}\right).$$

This corresponds to a $[[9, 0, 3]]$ code. The optimal distance for length 9 is 4. The bordered $[[10, 0, 4]]$ code is, however, extremal. We have also generated the quadratic residue code for $m = 25$. It is a $[[25, 0, 5]]$ code, and the bordered extension is a $[[26, 0, 6]]$ code. In both cases the best achievable distance is at least 8. This suggests that quadratic residue codes of non-prime lengths are not as strong as for prime lengths. Table 3.1 summarises the distance of quadratic residue and bordered quadratic residue codes for lengths up to 30.

Table 3.1: *Distance (d) of Quadratic Residue Codes of Length m and Bordered Quadratic Residue Codes of Length $m+1$*

QR codes		BQR codes	
m	d	$m+1$	d
5	3	6	4
9	3	10	4
13	5	14	6
17	5	18	6
25	5	26	6
29	11	30	12

Chapter 4

Nested Regular Graph Codes

4.1 The Hexacode and the Dodecacode

The unique extremal $[[6,0,4]]$ quantum code is also known as the *Hexacode.* As all self-dual quantum codes, it can be represented by a graph with adjacency matrix Γ, such that $\Gamma + \omega I$ is the generator matrix of a self-dual additive $[[6,0,4]]$ code over GF(4). One such representation of the Hexacode is given by the generator matrix

$$C = \begin{pmatrix} \omega & 0 & 1 & 1 & 1 & 0 \\ 0 & \omega & 0 & 1 & 1 & 1 \\ 1 & 0 & \omega & 0 & 1 & 1 \\ 1 & 1 & 0 & \omega & 0 & 1 \\ 1 & 1 & 1 & 0 & \omega & 0 \\ 0 & 1 & 1 & 1 & 0 & \omega \end{pmatrix}.$$

The corresponding graph is shown in Figure 4.1a. The figure emphasises the fact that the graph consists of two instances of the complete graph K_3, or equivalently, that there are two 3-cliques in the graph that partition all six vertices into two disjoint sets. We also see that each vertex in each 3-clique is connected to exactly one vertex in the other 3-clique. We will view these connections as an "outer" 2-clique; the rationale behind this will become clear. We call the whole graph a "2-clique of 3-cliques", or $K_2[K_3]$ for short. We also note that C is a circulant matrix, and that all vertices have vertex degree 3.

Another well-known unique extremal quantum code is the $[[12,0,6]]$ *Dodecacode.* It can be represented as a graph code with a circulant generator matrix whose first row is $(\omega 00101110100)$. The corresponding graph is shown in Figure 4.1c. We see how this graph can be called a "3-clique of 4-cliques", or $K_3[K_4]$.

Definition 4.1. A general *nested clique graph*, $G = (V,E)$, can be described as an "n_1-clique of n_2-cliques of $\cdots$ of n_l-cliques", denoted $K_{n_1}[K_{n_2}[\cdots[K_{n_l}]]]$. The number of vertices in the graph is $n = n_1 n_2 \cdots n_l$. It must be possible to partition the vertices into $\frac{n}{n_l}$ disjoint subsets of size n_l, $V = V_1 \cup V_2 \cup \cdots \cup V_{\frac{n}{n_l}}$, such that the induced subgraph on each subset is the complete graph K_{n_l}. Let E_i be the edges of the induced subgraph on V_i. Let E_{ij} be the set of edges $\{u,v\}$ where $u \in V_i$ and $v \in V_j$. $|E_{ij}|$ must be either 0 or n_l. If $|E_{ij}|$ is n_l, every vertex in V_i must be connected to one vertex in V_j, and every vertex in V_j must be connected to one vertex in V_i. If this is the case, we say that the sets V_i

and V_j are connected. All edges in E must be part of a clique or a connection between cliques, i.e, $\bigcup_i E_i \bigcup_{i,j} E_{ij} = E$. We then form the graph $G' = (V', E')$ on $\frac{n}{n_l}$ vertices, each vertex $v'_i \in V'$ corresponding to a subset $V_i \subset V$. Let there be an edge between v'_i and v'_j iff V_i and V_j are connected. G' must be the nested clique graph $K_{n_1}[K_{n_2}[\cdots[K_{n_{l-1}}]]]$, or, if $l = 2$, G' must be the complete graph K_{n_1}.

There may be many non-isomorphic $K_{n_1}[K_{n_2}[\cdots[K_{n_l}]]]$ graphs, so the nested clique characterisation does not uniquely identify a graph, but only partially describes its structure. In particular, the connections between the inner cliques of a nested clique graph are not defined. For instance, in the $K_3[K_4]$ graph shown in Figure 4.1c, the 12 edges that are not part of an inner 4-clique form a Hamiltonian cycle. Another $K_3[K_4]$ graph, where these 12 edges form four disjoint cycles of length three (or four 3-cliques), corresponds to a $[[12, 0, 4]]$ code. This means that a highly regular nested structure is not enough to guarantee optimal distance.

4.2 Graph Codes with Minimum Regular Vertex Degree

Proposition 4.2. *All vertices of a graph corresponding to an* $[[n, 0, d]]$ *quantum code have a vertex degree of at least* $d - 1$.

Proof. If a graph has a vertex v_i of degree δ, then the ith row of the adjacency matrix, Γ, will have weight δ. The ith row of the generator matrix, $\Gamma + \omega I$, of the corresponding self-dual additive code over GF(4), will then define a codeword of weight $\delta + 1$. A vertex with degree less than $d - 1$ would therefore correspond to a codeword of weight less than d, which is impossible. □

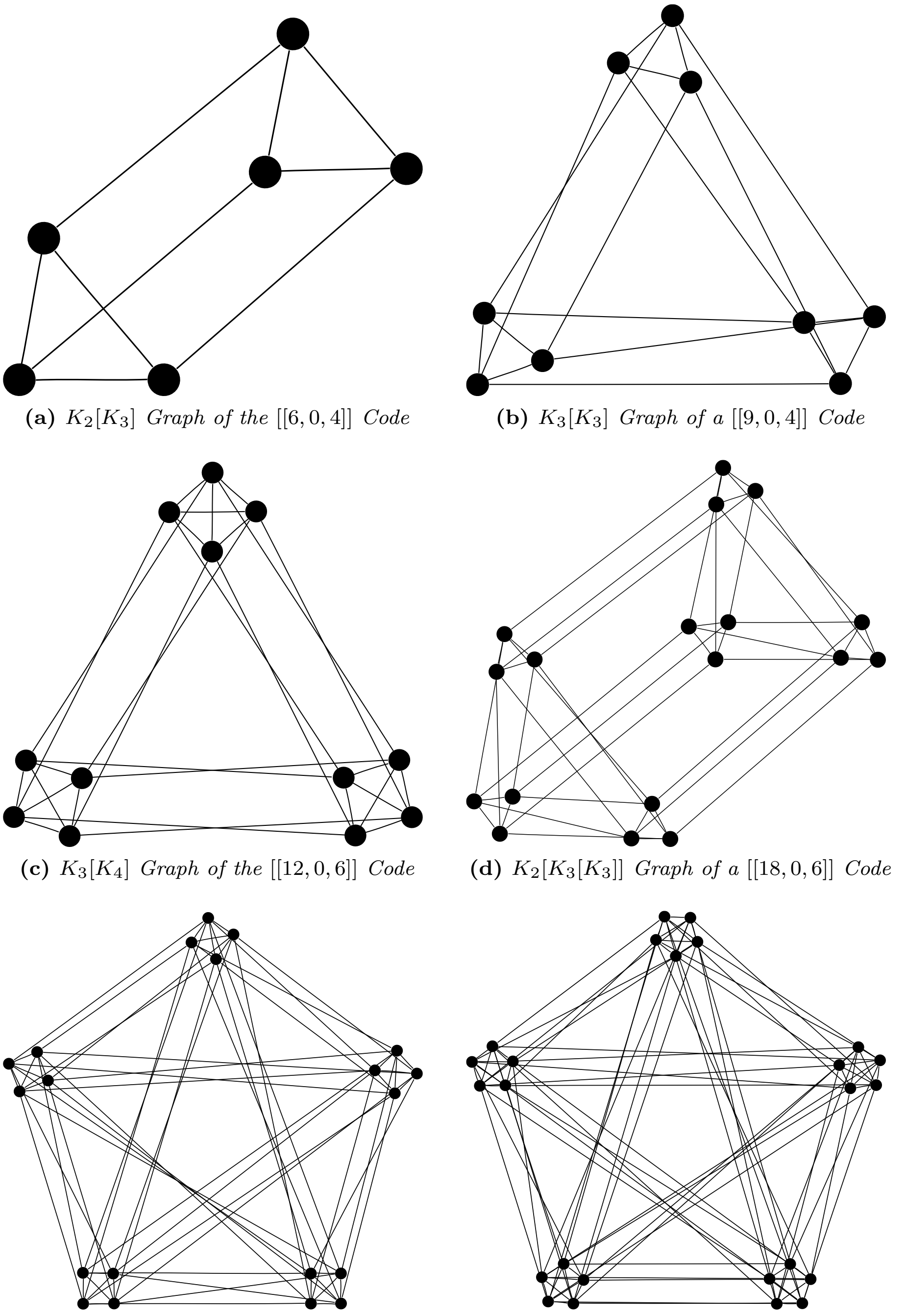
Proposition 4.3. *In a k-regular graph with an odd number of vertices, k must be even.*

Proof. Let Δ be the sum of the vertex degrees of all vertices in a graph. Since each edge of a graph is incident on two vertices, $\Delta = 2|E|$, i.e., twice the number of edges. It follows that Δ must always be an even number. In a k-regular graph on n vertices, $\Delta = kn$, and thus either k or n must be an even number. □

Definition 4.4. If a graph corresponding to an $[[n, 0, d]]$ quantum code has a regular vertex degree of $d - 1$, then it has *minimum regular vertex degree.* By Proposition 4.3, if n is odd and d is even, a regular vertex degree of $d - 1$ is impossible. In this case, a regular vertex degree of d is the minimum regular vertex degree.

The graph representation of the Hexacode shown in Figure 4.1a is 3-regular, and the graph representation of the Dodecacode shown in Figure 4.1c is 5-regular. With the distances of the two codes being 4 and 6, respectively, we conclude that the graph representations of both codes have minimum regular vertex degree. This also implies that no other graph representation of these two codes can have a smaller number of edges. When n is odd and d is even, a graph representation with minimum regular vertex degree is not necessarily the graph representation with the fewest edges.

Proposition 4.5. *The nested clique graph* $K_{n_1}[K_{n_2}[\cdots[K_{n_l}]]]$ *is a k-regular graph, where* $k = (n_1 - 1) + (n_2 - 1) + \cdots + (n_l - 1)$.



(a) *$K_2[K_3]$ Graph of the $[[6,0,4]]$ Code*

(b) *$K_3[K_3]$ Graph of a $[[9,0,4]]$ Code*

(c) *$K_3[K_4]$ Graph of the $[[12,0,6]]$ Code*

(d) *$K_2[K_3[K_3]]$ Graph of a $[[18,0,6]]$ Code*

(e) *$K_5[K_4]$ Graph of a $[[20,0,8]]$ Code*

(f) *$K_5[K_5]]$ Graph of a $[[25,0,8]]$ Code*

Figure 4.1: *Nested Clique Graphs*

Proof. Every vertex of the graph has $n_l - 1$ neighbours as part of an n_l-clique. Each vertex must also be connected to one vertex in $n_{l-1} - 1$ other n_l-cliques, which contributes $n_{l-1} - 1$ to its degree. The same must be true for the other layers of nesting. □

4.3 Other Nested Regular Graph Codes

We have observed that the extremal $[[6, 0, 4]]$ code corresponds to a $K_2[K_3]$ graph, and that the extremal $[[12, 0, 6]]$ code corresponds to a $K_3[K_4]$ graph. The intuitive next step is to search for an extremal $[[20, 0, 8]]$ code among $K_4[K_5]$ graphs. An exhaustive computer search of all $K_4[K_5]$ graphs did, however, not find such a code. The best result was a $[[20, 0, 6]]$ code, but with only 3 codewords of weight 6 and none of weight 7.

Searching through all $2^{\binom{n}{2}}$ undirected graphs on n vertices is infeasible for graphs with more than a few vertices. We have seen that the Hexacode and the Dodecacode have graph representations with circulant adjacency matrices, and this is also true for all quadratic residue codes of prime length. It therefore seems reasonable to restrict our search to the $2^{\lceil \frac{n-1}{2} \rceil}$ circulant symmetric adjacency matrices of graphs on n vertices. We have performed an exhaustive search of these graphs for $n \leq 30$. The most important parameter to optimise is the distance of the quantum code. For each length n, we identify the circulant adjacency matrices corresponding to $[[n, 0, d]]$ codes with optimal distance, as listed in Table 2.1 on page 18, or highest possible distance if no codes with optimal distance are found. We next want to minimise the regular vertex degree. Among the graphs corresponding to codes with highest possible distance and with lowest possible regular vertex degree, we try to identify structures similar to the nested clique representations of the Hexacode and the Dodecacode. Algorithms for finding all cliques in a graph have a running time that increases exponentially with the number of vertices. But we will only consider graphs of up to 30 vertices, and finding all cliques in such graphs can be done quickly with a suitable algorithm [9]. Table 4.1 summarises the results of the search by showing data about one code of each length. Distances and degrees marked $*$ in the table are not optimal. Degrees marked † are equal to d, but still optimal according to Proposition 4.3. Some of the nested regular structures identified can be seen in Figure 4.1 and Figure 4.2.

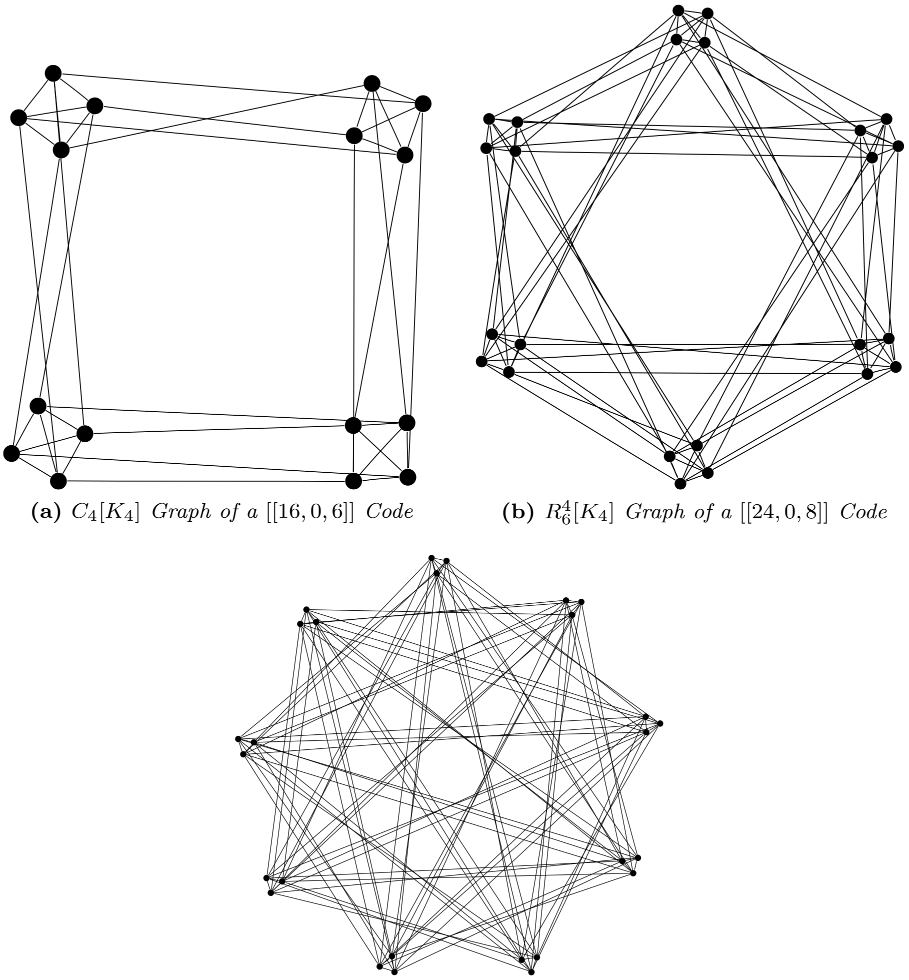
Proposition 4.6. *A graph, G, with no isolated vertex, i.e., no vertex with degree 0, corresponds to a self-dual $[[n, 0, d]]$ quantum code, $\mathcal{C}$, with minimum distance $d \geq 2$.*

Proof. It follows from Proposition 3.7 that any codeword with non-zero weight formed by adding 2 or more rows of the generator matrix of $\mathcal{C}$ will have weight higher than 2. A codeword of weight less than 2 must therefore be a row of the generator matrix. A vertex in G with degree $\delta \geq 1$ corresponds to a row in the generator matrix of $\mathcal{C}$, and hence a codeword, of weight $\delta + 1 \geq 2$. □

$n = 2$, 3, and 4 are not particularly interesting cases, since any graph with no isolated vertex will correspond to a code with $d = 2$. Graphs with minimum regular vertex degree representing extremal self-dual quantum codes of length 2, 3, and 4 can be described as K_2, K_3, and $K_2 + K_2$ (two unconnected 2-cliques), respectively. For lengths 5 and 7, the extremal distance is 3. The minimum regular vertex degree is 2, and the only 2-regular graph structure is the cycle graph. C_5 and C_7 correspond to $[[5, 0, 3]]$ and $[[7, 0, 3]]$ codes. For $n = 8$, a

Table 4.1: *Nested Regular Graphs with Degree δ Corresponding to Circulant Graph Codes of Length n and Distance d*

n	d	δ	Graph	First row of generator matrix
2	2	1	K_2	ω1
3	2	$2^\dagger$	K_3	ω11
4	2	1	$K_2 + K_2$	ω010
5	3	2	C_5	ω0110
6	4	3	$K_2[K_3]$	ω01110
7	3	2	C_7	ω001100
8	4	3	$K_2[C_4]$	ω0011100
9	4	$4^\dagger$	$K_3[K_3]$	ω00111100
10	4	3	$K_2[C_5]$	ω000111000
11	4^*	$4^\dagger$		ω0001111000
12	6	5	$K_3[K_4]$	ω00101110100
13	5	4		ω000101101000
14	6	5		ω0001011101000
15	6	8^*		ω01110011001110
16	6	5	$C_4[K_4]$	ω000100111001000
17	7	8^*		ω0100011111100010
18	6^*	5	$K_2[K_3[K_3]]$	ω00000101110100000
19	7	6		ω000101001100101000
20	8	7	$K_5[K_4]$	ω0000100111110010000
21	7^*	6		ω00001000111100010000
22	8	7		ω000001001111100100000
23	8^*	10^*		ω0000011101111011100000
24	8^*	7	$R_6^4[K_4]$	ω00000100011111000100000
25	8^*	$8^\dagger$	$K_5[K_5]$	ω000010001101101100010000
26	8^*	7		ω0000000100111110010000000
27	8^*	$8^\dagger$	$R_9^6[K_3]$	ω00000001100111100110000000
28	10	11^*		ω000001110100111001011100000
29	11	14^*		ω0110000101110110111010000110
30	12	17^*		ω01100001101111111110110000110



(a) $C_4[K_4]$ *Graph of a* $[[16, 0, 6]]$ *Code*

(b) $R_6^4[K_4]$ *Graph of a* $[[24, 0, 8]]$ *Code*

(c) $R_9^6[K_3]$ *Graph of a* $[[27, 0, 8]]$ *Code*

Figure 4.2: *Nested Regular Graphs*

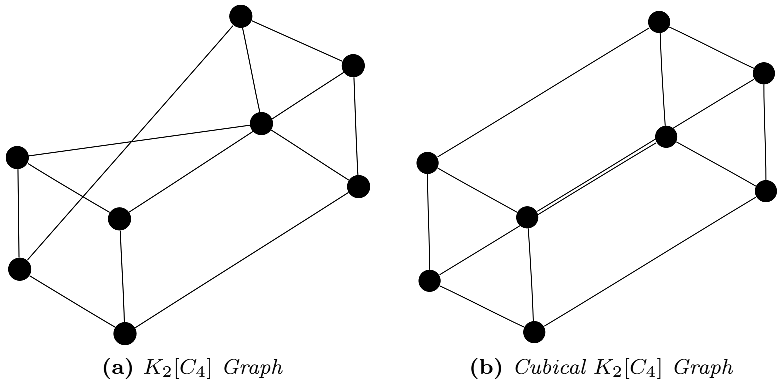

(a) *$K_2[C_4]$ Graph* **(b)** *Cubical $K_2[C_4]$ Graph*

Figure 4.3: *Two $K_2[C_4]$ Graphs Corresponding to* $[[8,0,4]]$ *Codes*

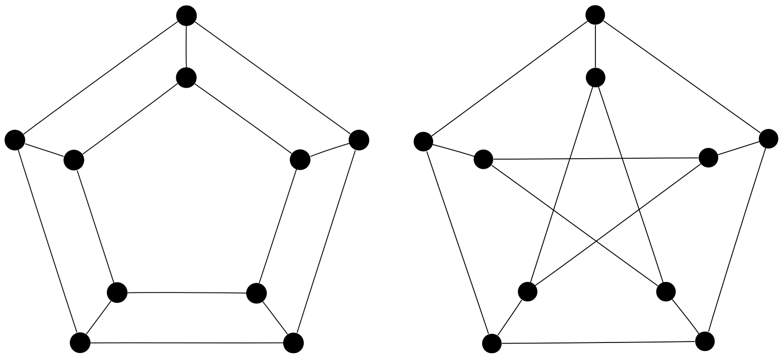
graph with minimum regular vertex degree and extremal distance consists of two 4-cycles which are connected in an "outer" 2-clique. This graph is depicted in Figure 4.3a. We see that we need to extend our definition of nested clique graphs to *nested regular graphs.* Recall that R_n^k denotes a k-regular graph on n vertices, $R_n^{n-1} = K_n$, and $R_n^2 = C_n$.

Definition 4.7. For a general *nested regular graph*, $G = (V, E)$, we use the notation $R_{n_1}^{k_1}[R_{n_2}^{k_2}[\cdots[R_{n_l}^{k_l}]]]$. The number of vertices in the graph is $n = n_1 n_2 \cdots n_l$. It must be possible to partition the vertices into $\frac{n}{n_l}$ disjoint subsets of size n_l, $V = V_1 \cup V_2 \cup \cdots \cup V_{\frac{n}{n_l}}$, such that the induced subgraph on each subset is a k_l-regular graph, $R_{n_l}^{k_l}$. Let E_i be the edges of the induced subgraph on V_i. Let E_{ij} be the set of edges $\{u, v\}$ where $u \in V_i$ and $v \in V_j$. $|E_{ij}|$ must be either 0 or n_l. If $|E_{ij}|$ is n_l, every vertex in V_i must be connected to one vertex in V_j, and every vertex in V_j must be connected to one vertex in V_i. If this is the case, we say that the sets V_i and V_j are connected. All edges in E must be part of a subset or a connection between subsets, i.e, $\bigcup_i E_i \bigcup_{i,j} E_{ij} = E$. We form the graph $G' = (V', E')$ on $\frac{n}{n_l}$ vertices, where each vertex $v_i' \in V'$ corresponds to a subset $V_i \subset V$. Let there be an edge between v_i' and v_j' iff V_i and V_j are connected. G' must be the nested regular graph $R_{n_1}^{k_1}[R_{n_2}^{k_2}[\cdots[R_{n_{l-1}}^{k_{l-1}}]]]$ or, if $l = 2$, G' must be the regular graph $R_{n_1}^{k_1}$.

Proposition 4.8. *$R_{n_1}^{k_1}[R_{n_2}^{k_2}[\cdots[R_{n_l}^{k_l}]]]$ is a regular graph with vertex degree $k_1 + k_2 + \cdots + k_l$.*

Proof. Every vertex of the graph has k_l neighbours as part of a k_l-regular subgraph. Each vertex must also be connected to one vertex in k_{l-1} of the $n_{l-1} - 1$ other k_l-regular graphs, which contributes k_{l-1} to its degree. The same must be true for the other layers of nesting. □

The cubical graph shown in Figure 4.3b is another $K_2[C_4]$ graph which also corresponds to an $[[8,0,4]]$ code. This graph is not isomorphic to the one shown in Figure 4.3a, and it does not have a circulant generator matrix. For $n = 9$, our search reveals a $K_3[K_3]$ graph, as shown in Figure 4.1b. For $n = 10$, we find the $K_2[C_5]$ graph shown in Figure 4.4a. The famous strongly regular *Petersen graph*, seen in Figure 4.4b, can also be described as $K_2[C_5]$. It also corresponds to a $[[10,0,4]]$ code, but does not have a circulant adjacency matrix, and is not isomorphic to the $K_2[C_5]$ graph in Figure 4.4a. A $K_2[K_5]$ graph corresponding to a $[[10,0,4]]$ code also exists, with a suboptimal regular degree of 4. For $n = 11$,



(a) $K_2[C_5]$ *Graph* **(b)** $K_2[C_5]$ *Petersen Graph*

Figure 4.4: *Two* $K_2[C_5]$ *Graphs Corresponding to* $[[10, 0, 4]]$ *Codes*

we did not find any circulant code with extremal distance $d = 5$. Since 11 and 13 are primes, there can be no nested regular graphs corresponding to codes of these lengths. Were we not able to find a nested clique description of any graph on 14 or 15 vertices corresponding to an extremal code, and no graph with minimum regular vertex degree corresponding to a $[[15, 0, 6]]$ code was found. For $n = 16$, we found the $C_4[K_4]$ graph shown in Figure 4.2a. No nested regular graphs exist for the prime lengths 17 and 19. For $n = 18$, the optimal distance is $d = 8$, but the best code corresponding to a circulant adjacency matrix is an $[[18, 0, 6]]$ code. This code can be described as a $K_2[K_3[K_3]]$ graph with minimum regular vertex degree. In addition, there also exists a $K_3[K_6]$ graph corresponding to an $[[18, 0, 6]]$ code. For $n = 20$, we discovered a $[[20, 0, 8]]$ code corresponding to a $K_5[K_4]$ graph, shown in Figure 4.1e. There is no circulant extremal code of length 21, and we did not find any nested regular graph corresponding to a $[[21, 0, 7]]$ code. Neither did we find such a graph description for any circulant $[[22, 0, 8]]$ code. For lengths from 23 to 27, the best codes we found all have distance 8. According to Table 2.1, this distance is not extremal, but no codes of higher distance are known, except for length 27, where a code of distance 9 exists. We did not find nested regular graphs for $n = 23$ or 26, but for $n = 24$ we found a $R_6^4[K_4]$ graph, as seen in Figure 4.2b. For $n = 25$ we found the $K_5[K_5]$ graph seen in Figure 4.1f, and for $n = 27$ we found the $R_9^6[K_3]$ graph seen Figure 4.2c. We have also found circulant graph codes of extremal distance for lengths 28, 29 and 30, but these graphs do not have minimal regular vertex degree and can not be described as nested regular graphs.

Gulliver and Kim [29] have studied the more general case of self-dual additive codes over GF(4) with circulant generator matrices. As for all self-dual quantum codes, there will be a graph code equivalent to any such code, but the graph code will not necessarily be circulant. Gulliver and Kim [29] also classified some other types of circulant based codes for lengths up to 30, but did not find any codes with higher distance than we found in our search of circulant graph codes.

4.4 Long Cycles in Nested Regular Graph Codes

The nested regular description of a graph does not specify the manner in which the regular subgraphs are connected. It turns out that these connections are highly structured in nested regular graphs corresponding to strong self-dual quantum codes. We have already mentioned that the $K_3[K_4]$ graph representation of the Dodecacode has a Hamiltonian cycle. The $K_5[K_5]$ graph, shown in Figure 4.1f, corresponding to a $[[25, 0, 8]]$ code, contains two edge-disjoint Hamiltonian cycles. We can then account for all the 100 edges of this graph, 50 being part of inner 5-cliques and 25 of the remaining edges in each Hamiltonian cycle. But this is still not a description of a unique graph, as the following example shows.

Example 4.9. Let $G = (V, E)$ be a $K_5[K_5]$ graph. Let the five inner 5-cliques be on the vertex sets $V_1 = \{v_0, v_1, v_2, v_3, v_4\}$, $V_2 = \{v_5, v_6, v_7, v_8, v_9\}$, $V_3 = \{v_{10}, v_{11}, v_{12}, v_{13}, v_{14}\}$, $V_4 = \{v_{15}, v_{16}, v_{17}, v_{18}, v_{19}\}$, and $V_5 = \{v_{20}, v_{21}, v_{22}, v_{23}, v_{24}\}$. Let the remaining 50 edges form two Hamiltonian cycles given by the sequences of vertices, $H_1 = (v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12}, v_{13}, v_{14}, v_{15}, v_{16}, v_{17}, v_{18}, v_{19}, v_{20}, v_{21}, v_{22}, v_{23}, v_{24}, v_0)$ and $H_2 = (v_0, v_7, v_{14}, v_{21}, v_3, v_{10}, v_{17}, v_{24}, v_6, v_{13}, v_{20}, v_2, v_9, v_{16}, v_{23}, v_5, v_{12}, v_{19}, v_1, v_8, v_{15}, v_{22}, v_4, v_{11}, v_{18}, v_0)$. This graph corresponds to a $[[25, 0, 8]]$ self-dual quantum code. But if we replace the Hamiltonian cycle H_2 with $H_2' = (v_0, v_8, v_{16}, v_{24}, v_7, v_{15}, v_{23}, v_6, v_{14}, v_{22}, v_5, v_{13}, v_{21}, v_4, v_{12}, v_{20}, v_3, v_{11}, v_{19}, v_2, v_{10}, v_{18}, v_1, v_9, v_{17}, v_0)$, we get a $[[25, 0, 6]]$ code.

The $K_5[K_4]$ graph corresponding to a $[[20, 0, 8]]$ code, shown in Figure 4.1e, contains one Hamiltonian cycle, in addition to two vertex-disjoint cycles, each visiting half the vertices of the graph. We generated the codes corresponding to all $K_5[K_4]$ graphs with two Hamiltonian cycles, and the highest distance found was 6. Other nested regular graphs listed in Table 4.1 also contain Hamiltonian cycles or long cycles. Both the nested regular structure and the presence of long cycles seem to be important characteristics of the graphs corresponding to self-dual quantum codes of high distance.

Although we have not found a construction technique for nested graph codes giving a predictable distance, initial results suggest that the set of nested regular graphs give a small search space in which strong codes are likely to be found. Randomly generated nested regular graphs with long disjoint cycles typically give codes of higher distance than totally random graphs. It seems like the long cycles should be arranged in such a way that no smaller cycles are induced in the graph. The results shown in Table 4.1 suggest that for codes of length above 25 and distance higher than 8, graph structures get more complicated. To describe these structures, a further generalisation of nested regular graphs may be necessary.

Chapter 5

Orbits of Self-Dual Quantum Codes

5.1 Local Transformations and Local Complementations

We have seen that an $[[n, 0, d]]$ quantum stabilizer code represents a single quantum state, and that all such codes can be transformed into equivalent graph codes. The quantum state corresponding to a graph code is known as a graph state, and in section 6.3 we will see how a state $|G\rangle$ corresponding to a graph G can be found. In section 2.1 we saw that local unitary transformations are reversible transformations that act independently on each qubit in a quantum state. If there exists a local unitary transformation U, such that $U|G\rangle = |G'\rangle$, the states $|G\rangle$ and $|G'\rangle$ will have the same entanglement properties. If $|G\rangle$ and $|G'\rangle$ are graph states, we say that their corresponding graphs, G and G', are *LU-equivalent*. G and G' will then represent equivalent quantum codes, with the same distance, weight distribution, and other properties.

Determining whether two graphs are LU-equivalent seems like a difficult task, but a sufficient condition for equivalence was given by Hein et al. [30]. Let the graphs $G = (V, E)$ and $G' = (V, E')$ on n vertices correspond to the n-qubit graph states $|G\rangle$ and $|G'\rangle$.

Definition 5.1. We define the two 2×2 unitary matrices,

$$\tau_x = \sqrt{-i\sigma_x} = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 & i \\ i & -1 \end{pmatrix}, \quad \tau_z = \sqrt{i\sigma_z} = \begin{pmatrix} w & 0 \\ 0 & w^3 \end{pmatrix},$$

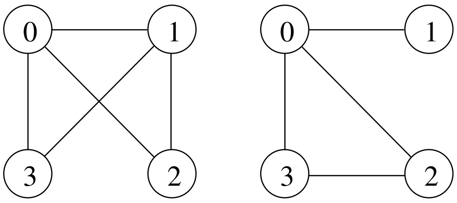
where $w^4 = i^2 = -1$, and σ_x and σ_z are Pauli matrices.

Definition 5.2. Given a graph $G = (V = \{0, 1, \ldots, n-1\}, E)$, corresponding to the graph state $|G\rangle$, we define a local unitary transformation,

$$U_a = \bigotimes_{i \in N_a} \tau_x^{(i)} \bigotimes_{i \notin N_a} \tau_z^{(i)}, \tag{5.1}$$

where $a \in V$ is any vertex, $N_a \subset V$ is the neighbourhood of a, and $\tau_x^{(i)}$ means that the transform τ_x should be applied to the qubit corresponding to vertex i.

Given a graph G, if there exists a finite sequence of vertices $(u_0, u_1, \ldots, u_{k-1})$, such that $U_{u_{k-1}} \cdots U_{u_1} U_{u_0} |G\rangle = |G'\rangle$, then G and G' are LU-equivalent. It was discovered by Hein et al. [30], and by Van den Nest et al. [59], that the sequence of transformations taking $|G\rangle$ to $|G'\rangle$ can equivalently be expressed as a sequence of simple graph operations taking G to G'. Exactly the same graph



(a) *The Graph G* **(b)** *The LC Image G^0*

Figure 5.1: *Example of Local Complementation*

operation, called *vertex neighbourhood complementation* (VNC), was described by Glynn et al. [22, 23] as an operation that maps equivalent self-dual additive codes over GF(4) to each other. VNC is another name for *local complementation* (LC), referred to in the context of *isotropic systems* by Bouchet [5, 7].

Definition 5.3. Given a graph $G = (V, E)$ and a vertex $v \in V$, let $N_v \subset V$ be the neighbourhood of v. The subgraph induced by N_v is complemented to obtain the LC image G^v, i.e., $G^v(N_v) = \overline{G(N_v)}$. It is easy to verify that $(G^v)^v = G$.

Example 5.4. We will perform local complementation on vertex 0 of the graph G, shown in Figure 5.1a. We see that the neighbourhood of 0 is $N_0 = \{1, 2, 3\}$, and that the induced subgraph on the neighbourhood, $G(N_0)$, has edges $\{1, 2\}$ and $\{1, 3\}$. The complement of this subgraph, $\overline{G(N_0)}$, contains the single edge $\{2, 3\}$. The resulting LC image, G^0, is seen in Figure 5.1b.

Theorem 5.5 (Glynn et al. [22, 23], Hein et al. [30], and Van den Nest et al. [59])**.** *The graphs G and G' are* LC-equivalent*, i.e., they correspond to equivalent self-dual quantum codes, if there is a finite sequence of vertices $u_0, u_1, \ldots, u_{k-1}$, such that $(((G^{u_0})^{u_1})^{\cdots})^{u_{k-1}} = G'$.*

LC operations on the graph $G = (V, E)$, represented by the adjacency matrix Γ, can also be described in terms of operations on $C = \Gamma + \omega I$, the generator matrix of a self-dual additive code over $GF(4)$. We can then verify that none of the matrix operations will change the properties of the code. LC on a vertex $a \in V$ corresponds to the following sequence of operations on C.

- For all vertices $i \in N_a$, add row a to row i in C. This operation, which does not change the properties of an additive code, implements the neighbourhood complementation of the corresponding graph, but also leaves us with a code that is not a graph code. The two following steps are needed to restore the code to graph form.

- Scale column a of C by ω, i.e., multiply coordinate a in all rows of C by ω. Scaling the same coordinate in all codewords by some nonzero value gives an equivalent code.

- For all vertices $i \in N_a \cup \{a\}$, conjugate column i of C. Conjugating coordinates does not change the properties of the code. We now have a generator matrix of the form $C' = \Gamma' + \omega I$.

5.2 Enumerating LC Orbits

Van den Nest et al. [58] report on an efficient algorithm, first described by Bouchet [6], which determines whether two graphs are LC-equivalent by solving a set of equations. This algorithm has complexity $O(n^2)$, where n is the number of vertices in the input graphs. Note that we can also consider codes corresponding to isomorphic graphs to be equivalent, since permuting coordinates of a self-dual additive code over GF(4) gives an equivalent code. We therefore want to detect LC-equivalence of graphs *up to isomorphism*, which means that a permutation of vertex labels is allowed before each LC operation. Permuting the vertex labels of a graph causes the qubits in the corresponding graph state to be reordered. Reordering the qubits in a quantum state does not change the overall entanglement properties, but can not be performed by any local unitary transformation. The above-mentioned algorithm only considers equivalence via local unitary transformations, and can therefore not be used to detect LC-equivalence up to isomorphism.

Definition 5.6. The *LC orbit* $\boldsymbol{L} = [G]$, of a graph G, is the set of all non-isomorphic graphs, including G itself, that can be transformed into G by any sequence of local complementations and vertex permutations.

Example 5.7. Consider the "2-cliques of 3-cliques" representation of the $[[6, 0, 4]]$ Hexacode shown in Figure 3.2a on page 22. An LC operation on any vertex of this graph will produce a graph isomorphic to the "wheel graph" shown in Figure 3.2b. An LC operation on the "centre" of the "wheel" will again produce a graph isomorphic to the "wheel graph", while an LC operation on any of the 5 other vertices gives a graph isomorphic to the "2-clique of 3-cliques". These two graphs therefore make up the complete LC orbit of the Hexacode.

Let $\mathcal{G}_n$ be the set of all non-isomorphic simple undirected connected graphs on n vertices. (We will later consider a set where unconnected graphs are included.) Let $\mathcal{L}_n = \{\boldsymbol{L}_1, \boldsymbol{L}_2, \ldots, \boldsymbol{L}_k\}$ be the set of all distinct LC orbits of graphs in $\mathcal{G}_n$. All $\boldsymbol{L} \in \mathcal{L}_n$ are disjoint, and $\mathcal{L}_n$ is a partitioning of $\mathcal{G}_n$, i.e., $\bigcup_i \boldsymbol{L}_i = \mathcal{G}_n$. Two graphs, G and K, are equivalent with respect to local complementations and vertex permutations if one of the graphs is in the LC orbit of the other, for instance, $K \in [G]$. We will need Algorithm 5.1, a recursive algorithm that generates the LC orbit of a given graph. The package `nauty`, described in section 3.2, is used to implement the procedure NautyCanonise(G), which returns a canonical representative of the graph G. Every isomorphic graph has the same canonical representative. In our algorithms, we will also require data structures for storage of graphs. Let $\boldsymbol{T}$ be such a data structure. The exact implementation of $\boldsymbol{T}$ may vary, but we assume that there is a procedure Add($\boldsymbol{T}$, G) that causes the graph G to be added to $\boldsymbol{T}$.

Example 5.8. As an example, we will generate $\mathcal{L}_4$, the set of all LC orbits on 4 vertices. There are $2^{\binom{4}{2}} = 64$ undirected simple graphs on 4 vertices, but the number of non-isomorphic connected graphs is only $|\mathcal{G}_4| = 6$. We use Algorithm 5.1 on these graphs and find that there are $|\mathcal{L}_4| = 2$ distinct LC orbits on 4 vertices. The orbits, $\mathcal{L}_4 = \{\boldsymbol{L}_1, \boldsymbol{L}_2\}$, are shown in Figure 5.2.

We would like to partition $\mathcal{G}_n$ into a set of LC orbits, $\mathcal{L}_n$, for n as high as possible. In particular, we want to count the number of LC orbits, $|\mathcal{L}_n|$, which is also the number of inequivalent self-dual additive codes over GF(4) of length n. If we can also find one representative of each LC orbit, we can characterise the properties of all such codes. Self-dual additive codes over GF(4) of length n have

Algorithm 5.1 Generating the LC Orbit of a Graph

```
Input    $G$: a graph, $G = (V, E)$
Output   $\boldsymbol{L}$: data structure containing all graphs in $[G]$

procedure GenerateOrbit($G$)
    initialise $\boldsymbol{L}$
    RecursiveGenerateOrbit($G$, $\boldsymbol{L}$)
    return $\boldsymbol{L}$
end procedure

procedure RecursiveGenerateOrbit($G$, $\boldsymbol{L}$)
    if $G \notin \boldsymbol{L}$ then
        for all $v \in V$ do
            $K \leftarrow$ NautyCanonise($G^v$)
            Add($\boldsymbol{L}$, $K$)
            RecursiveGenerateOrbit($K$, $\boldsymbol{L}$)
        end for
    end if
end procedure
```

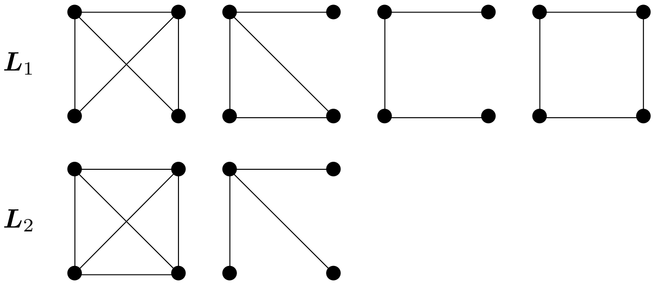


Figure 5.2: *The Two LC Orbits for* $n = 4$

previously been enumerated by Calderbank et al. [10] for $n \leq 5$, by Höhn [31] for $n \leq 7$, by Hein et al. [30] for $n \leq 7$, and by Glynn et al. [23] for $n \leq 9$. Glynn has also posted his results as sequence A090899 in *The On-Line Encyclopedia of Integer Sequences* [56]. For higher n, only partial classifications of extremal codes have been performed [4, 20, 21].

Algorithm 5.2, our first attempt at an algorithm for generating all LC orbits, is inspired by the concept of canonical representatives, as used by `nauty`. We define a procedure LCcanonise, which returns the same canonical representative for every member of the same LC orbit. The procedure FindOrbits1 with $\mathcal{G}_n$ as input will canonise every graph in $\mathcal{G}_n$ and remove all duplicates in the resulting set. We would then have one representative of every LC orbit in $\mathcal{L}_n$. Generating the set $\mathcal{G}_n$ can be done by the utility `geng` from the `nauty` package. It is not important how the canonical representative of an LC orbit is chosen, as long as it is done consistently. Algorithm 5.2 gives an implementation of LCcanonise(G) that first generates the complete orbit $\boldsymbol{L} = [G]$. The procedure First($\boldsymbol{L}$) then picks the "first" graph in the set $\boldsymbol{L}$ by some lexicographical ordering. The exact implementation of this ordering is not important.

Finding all LC orbits in $\mathcal{L}_n$ by Algorithm 5.2 is clearly not efficient. We are doing much redundant work by going through the whole LC orbit of every

Table 5.1: *Sizes of Different Sets of Graphs*

n	$\lvert\mathcal{G}_n\rvert$	$\lvert\mathcal{E}'_n\rvert$	$\lvert\mathcal{E}_n\rvert$
1	1	-	-
2	1	1	1
3	2	3	2
4	6	7	5
5	21	30	14
6	112	124	48
7	853	693	228
8	11,117	3,302	1,338
9	261,080	25,755	11,309
10	11,716,571	224,840	123,899
11	1,006,700,565	3,204,036	2,138,482
12	164,059,830,476	82,815,479	66,150,188
13	50,335,907,869,219	5,217,308,460	?

graph in $\mathcal{G}_n$, since many orbits will then be generated several times. A great improvement in running time is achieved by storing all LC orbits in memory at the same time. Algorithm 5.3 guarantees that all distinct LC orbits will only be generated once, at the cost of extra memory requirement. The procedure will store all members of every LC orbit it generates in the temporary set $\boldsymbol{T}$. If a graph is already in $\boldsymbol{T}$, its orbit is not generated again. If we call the procedure FindOrbits2 with $\mathcal{G}_n$ as input, $\boldsymbol{T}$ will contain all graphs in $\mathcal{G}_n$ at the time the procedure terminates.

A straightforward implementation of Algorithm 5.3 will work when $n \leq 8$. The total number of undirected graphs on 8 vertices is $2^{\binom{8}{2}} = 268{,}435{,}456$. Since we have more than 300 MB of memory available, we can simply make $\mathbf{T}$ a binary array with one bit representing each graph. All bits are initialised to zero, and will be set to one once the corresponding graph has been discovered. This is clearly a waste of memory, since all the graphs we need to store are the $|\mathcal{G}_8| = 11{,}117$ non-isomorphic connected graphs on 8 vertices. When $n = 9$, such an array would require about 10 GB of memory, so a more clever approach is needed. We therefore use a *binary search tree* as $\mathbf{T}$. Every time we discover a graph not isomorphic to any graph in $\mathbf{T}$, we add a new node to the tree. Since our graphs are undirected, only the lower or upper triangle of the adjacency matrix needs to be stored. This means that $\binom{n}{2}$ bits of memory are needed to store each graph. These bits are easily interpreted as a numerical value for comparisons in the binary search tree. With this method, only non-isomorphic graphs will be stored in $\mathbf{T}$, so the memory requirement is proportional to $|\mathcal{G}_n|$. Values of $|\mathcal{G}_n|$ are listed in Table 5.1, and is also sequence A001349 in *The On-Line Encyclopedia of Integer Sequences* [56].

While the orbits for $n = 9$ are easily computed with the binary search tree implementation of Algorithm 5.3, $n = 10$ takes about one hour of running time and uses more than 100 MB of memory. The memory requirement is the biggest obstacle, and for $n = 11$ it becomes infeasible. To solve this problem, we tried to find an *invariant*, i.e., some property that has the same value for all graphs in the same LC orbit, and that can be calculated quickly. One such property is the weight distribution of the codes corresponding to the graphs, since we can with certainty say that two codes with different weight distributions are not in the same LC orbit. (The converse is however not true, since many LC orbits will

Algorithm 5.2 Finding LC Orbits By Canonisation

```
Input     $\boldsymbol{F}$: a set of graphs
Output    $\boldsymbol{O}$: a set with one representative of each LC orbit present in $\boldsymbol{F}$

procedure FindOrbits1($\boldsymbol{F}$)
    initialise $\boldsymbol{O}$
    for all $G \in \boldsymbol{F}$ do
        $K \leftarrow$ LCcanonise($G$)
        if $K \notin \boldsymbol{O}$ then
            add($\boldsymbol{O}$, $K$)
        end if
    end for
    return $\boldsymbol{O}$
end procedure

procedure LCcanonise($G$)
    $\boldsymbol{L} \leftarrow$ GenerateOrbit($G$)
    $K \leftarrow$ First($\boldsymbol{L}$)
    return $K$
end procedure
```

Algorithm 5.3 Finding LC Orbits Quickly

```
Input     $\boldsymbol{F}$: a set of graphs
Output    $\boldsymbol{O}$: a set with one representative of each LC orbit present in $\boldsymbol{F}$

procedure FindOrbits2($\boldsymbol{F}$)
    initialise $\boldsymbol{O}$ and $\boldsymbol{T}$
    for all $G \in \boldsymbol{F}$ do
        $K \leftarrow$ NautyCanonise($G$)
        if $K \notin \boldsymbol{T}$ then
            add($\boldsymbol{O}$, $K$)
            $\boldsymbol{L} \leftarrow$ GenerateOrbit($K$)
            for all $I \in \boldsymbol{L}$ do
                add($\boldsymbol{T}$, $I$)
            end for
        end if
    end for
    return $\boldsymbol{O}$
end procedure
```

have exactly the same weight distribution.) To reduce the memory requirement of our algorithm, we calculate the weight distribution of the codes corresponding to graphs in $\mathcal{G}_n$ and store the graphs in k different sets, $\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_k$, with the only restriction that codes with the same weight distribution must be in the same set. We do not necessarily have to compute the complete weight distributions, since the partial weight distribution $\boldsymbol{w}_p$, i.e., the numbers of codewords of weights up to p, is also an invariant over the LC orbit. $\boldsymbol{w}_p$ can be calculated efficiently by the method described in Algorithm 3.2 on page 25. We choose a p that is high enough to give a good separation of the codes, while still being computable for all graphs in $\mathcal{G}_n$ in reasonable time. The distribution of codes will not be uniform, so the resulting sets will be of various sizes. When the sets $\mathbf{F}_1, \mathbf{F}_2, \ldots, \mathbf{F}_k$ are generated, we call the procedure FindOrbits2 k times, once with each set as input. The sets are processed independently, and the data structure $\mathbf{T}$ in procedure FindOrbits2 can be reset for each set, reducing the amount of memory needed. This method also allows us to process the k sets in parallel. The outputs returned by the k procedure calls are concatenated to form a complete set of representatives of $\mathcal{L}_n$. This approach was used to classify all inequivalent $[[11, 0, d]]$ codes. Generating all non-isomorphic graphs and splitting them into 1,000 files according to partial weight distribution took about 3 days on an ordinary desktop computer. Many of the resulting files were empty or nearly empty, whereas the largest were several 100 MB. The total size of all the files after compression was about 6 GB. The processing of the files was done in parallel on a cluster computer in a matter of hours, and 40,457 inequivalent codes were found. Processing the largest file required more than 1 GB of memory, so using this method for $n = 12$ was not feasible with the available resources.

We have seen that the procedure FindOrbits1 is too slow, and that the procedure FindOrbits2 requires too much memory. In Algorithm 5.4 we define FindOrbits3 which, like FindOrbits1, only stores a single LC orbit in memory at any time. It does not, however, generate the LC orbit of every graph in the input set, and is therefore faster than FindOrbits1. FindOrbits3 also benefits from splitting up the input set using partial weight distribution. Algorithm 5.4 uses the procedure Remove($\boldsymbol{T}$, G) which removes the graph G from $\boldsymbol{T}$, and the procedure RemoveNext($\boldsymbol{T}$), which removes some graph from $\boldsymbol{T}$ and returns it. The order in which graphs are removed from $\boldsymbol{T}$ is not important here.

Definition 5.9. The $2^n - 1$ *extensions* of a graph on n vertices is formed by adding a new vertex and joining it to all possible combinations of at least one of the old vertices. The set $\mathcal{E}'_n$, containing $|\mathcal{L}_{n-1}| \times \left(2^{n-1} - 1\right)$ graphs, is formed by making all possible extensions of one representative from each LC orbit in $\mathcal{L}_{n-1}$. Let $\mathcal{E}_n$ contain the same graphs as $\mathcal{E}'_n$, except that all isomorphisms are removed.

Proposition 5.10 (Glynn et al. [23])**.** *The set $\mathcal{E}'_n$ will contain at least one representative from each LC orbit in $\mathcal{L}_n$.*

Proof. Let $G = (V, E)$ be any graph on n vertices. Choose any subset $W \subset V$ of $n - 1$ vertices. By doing LC operations on vertices in W, we can transform the subgraph $G(W)$ into any member of the LC orbit $[G(W)]$. One of these members was extended when the set $\mathcal{E}'_n$ was constructed. It follows that for all $G \in \mathcal{G}_n$, some $G' \in [G]$ must be part of $\mathcal{E}'_n$. □

Table 5.1 gives the values of $|\mathcal{E}'_n|$ and $|\mathcal{E}_n|$, which are much smaller than the values of $|\mathcal{G}_n|$. It will therefore be more efficient to use $\mathcal{E}_n$ instead of $\mathcal{G}_n$ as input

Algorithm 5.4 Finding LC Orbits Quickly Using Less Memory

```
Input   $\boldsymbol{F}$: a set of graphs
Output  $\boldsymbol{O}$: a set with one representative of each LC orbit present in $\boldsymbol{F}$

procedure FindOrbits3($\boldsymbol{F}$)
    initialise $\boldsymbol{O}$
    while $\boldsymbol{F}$ is not empty do
        $G \leftarrow$ RemoveNext($\boldsymbol{F}$)
        $K \leftarrow$ NautyCanonise($G$)
        add($\boldsymbol{O}$, $K$)
        $\boldsymbol{L} \leftarrow$ GenerateOrbit($K$)
        for all $I \in \boldsymbol{F}$ do
            if $I \in \boldsymbol{L}$ then
                remove($\boldsymbol{F}$, $I$)
            end if
        end for
    end while
    return $\boldsymbol{O}$
end procedure
```

to our algorithms, and, by Proposition 5.10, we will still find one representative of every LC orbit in $\mathcal{L}_n$. We managed to find all inequivalent self-dual quantum codes of length $n = 12$ by using the set $\mathcal{E}_{12}$ as input to FindOrbits3. The set of 66,150,188 graphs in $\mathcal{E}_{12}$ was first divided into 1,000 files by the partial weight distribution $\boldsymbol{w}_7$, a process that took a few hours. The files were then processed in parallel on a cluster computer. A *hash table* was used as the data structure **L** in the algorithm, to allow for fast look-up. The processing took a little more than a week to finish, using a total of more than 4,000 CPU-hours, and 1,274,068 LC orbits were found.

Table 5.2 shows the value of $|\mathcal{L}_n|$, which is also the number of inequivalent self-dual additive codes over GF(4), for n up to 12. A database of orbit representatives with information about orbit size, distance, and weight distribution is also available [14]. The numbers of inequivalent codes have been added to sequence A090899 in *The On-Line Encyclopedia of Integer Sequences* [56], which previously only had values for n up to 9. The numbers of codes of various distances are shown in Table 5.3. Table 5.4 gives the number of self-dual quantum codes of type II, i.e., codes where all codewords have even weight.

Definition 5.11. Let "a", where $a \in \mathbb{N}$, denote a connected graph on a vertices. Let "$a_1^{b_1} a_2^{b_2} \cdots a_k^{b_k}$", where $a_i, b_i \in \mathbb{N}$, denote an unconnected graph composed of b_0 connected components with a_0 vertices, b_1 connected components with a_1 vertices, and so on.

Recall that $\mathcal{L}_n$ only contains LC orbits of connected graphs. Codes corresponding to connected graphs are called *indecomposable.* Let the set $\mathcal{L}'_n \subset \mathcal{L}_n$ also include all LC orbits of unconnected graphs. An unconnected graph on n vertices is composed of a set of connected components with less than n vertices. Likewise, codes corresponding to unconnected graphs, called *decomposable* codes, can always be expressed as a combination of indecomposable codes of shorter length. Table 5.5, which is an extended version of a table given by Glynn et al. [23], counts the number of non-isomorphic graphs for all possible such combinations. The values of $|\mathcal{L}'_n|$ for n up to 12 are also shown in Table 5.2, and is a new sequence, A094927, in *The On-Line Encyclopedia of Integer Sequences* [56].

Table 5.2: *Number of Self-Dual Quantum Codes of Length n*

n	$\lvert\mathcal{L}_n\rvert$	$\lvert\mathcal{L}'_n\rvert$
1	1	1
2	1	2
3	1	3
4	2	6
5	4	11
6	11	26
7	26	59
8	101	182
9	440	675
10	3,132	3,990
11	40,457	45,144
12	1,274,068	1,323,363

Table 5.3: *Number of Indecomposable Self-Dual Quantum Codes of Length n and Distance d*

	n										
d	2	3	4	5	6	7	8	9	10	11	12
2	1	1	2	3	9	22	85	363	2,436	26,750	611,036
3				1	1	4	11	69	576	11,200	467,513
4					1		5	8	120	2,506	195,455
5										1	63
6											1
All	1	1	2	4	11	26	101	440	3,132	40,457	1,274,068

Table 5.4: *Number of Indecomposable Type II Self-Dual Quantum Codes of Length n and Distance d*

	n					
d	2	4	6	8	10	12
2	1	1	3	11	84	2,133
4			1	3	19	792
6						1
All	1	1	4	14	103	2,926

Note that the number of non-isomorphic graphs of type "4^2" is only 3, and not 4. Likewise, the number of "5^2" graphs is 10, the number of "4^3" graphs is 4, and the number of "6^2" graphs is 66. For all other combinations used in Table 5.5, the number of non-isomorphic unconnected graphs is simply found by multiplying the numbers of non-isomorphic connected graphs for each connected component, e.g., the number of "641^2" graphs is $11 \times 2 \times 1 \times 1 = 22$.

Example 5.12. We will count the number of LC orbits in $\mathcal{L}'_4$. The unconnected graphs of type "2^2", i.e., graphs formed by combining 2 connected graphs on 2 vertices, must be included. There is only one orbit in $\mathcal{L}_2$, and therefore only one inequivalent decomposable code of type "2^2". Another way to construct unconnected graphs on 4 vertices is to combine a connected graph on 3 vertices with an isolated vertex. This combination is denoted "31". Since $\mathcal{L}_3 = \mathcal{L}_1 = 1$, there is only one LC orbit of type "31". The other combinations that make unconnected graphs on 4 vertices, "21^2" and "1^4", also give one orbit each. $\mathcal{L}'_4$ also contains the 2 orbits of connected graphs in $\mathcal{L}_4$. (This combination is simply denoted "4".) Thus, when we add all the combinations, $|\mathcal{L}'_4| = 6$.

5.3 The LC Orbits of Some Strong Codes

In chapter 4, we studied graph codes with circulant generator matrices. For some lengths we found that the best circulant graph codes had lower distance than the best known self-dual quantum codes. In particular, we did not find any $[[11, 0, 5]]$, $[[18, 0, 8]]$, $[[21, 0, 8]]$ or $[[27, 0, 9]]$ codes. Generator matrices for the best known quantum codes are listed by Grassl [26]. We have transformed some of these codes into graph codes and generated their LC orbits.

Remark. There are no regular graphs corresponding to $[[11, 0, 5]]$ or $[[18, 0, 8]]$ quantum codes.

The unique $[[11, 0, 5]]$ code is already known from our complete classification of all self-dual quantum codes of length up to 12. This code has 4,742 non-isomorphic graphs in its LC orbit, and none of these graphs are regular. The $[[18, 0, 8]]$ code has been shown (under some constraints) to be unique [4], and it can be generated as shown in section 3.5. There is no regular graph among the 3,828 non-isomorphic graphs in the LC orbit of this code, but the graph representation with the fewest edges contains only one vertex of degree 9, with the rest of the vertices having degree 7. This is the closest to minimum regular vertex degree we can get without achieving it, since the code is of type II, and all its graph representations must therefore have only odd vertex degrees. We also generate the LC orbit of the $[[21, 0, 8]]$ code listed by Grassl [26]. It has 77,394 members in its LC orbit, and again no regular graph is found. Note that neither did Gulliver and Kim [29] find any $[[21, 0, 8]]$ code with a circulant based generator matrix. We also tried to generate the LC orbit of a $[[27, 0, 9]]$ code, but after finding about 10 million non-isomorphic graphs, the memory resources of the computer were exhausted. None of the graphs found were regular. The $[[30, 0, 12]]$ code we discovered in our search of circulant graph codes had a regular vertex degree of 17. We could only generate about 10 million members of its LC orbit, but this sufficed to find a graph with regular vertex degree 15. This graph can not be described as a nested regular graph, however, and the degree is still far from the minimal 11. The graph representation with the fewest edges found had 171 edges, an "average vertex degree" of 11.4.

In section 3.5 we showed how the $[[30, 0, 12]]$ code can be constructed as a bordered quadratic residue code. The graph corresponding to the $[[29, 0, 11]]$

Table 5.5: *Numbers of Decomposable Self-Dual Quantum Codes*

1		2		3		4		5		6		7		8		9		10		11		12	
1	1	1^{2}	1	1^{3}	1	1^{4}	1	1^{5}	1	1^{6}	1	1^{7}	1	1^{8}	1	1^{9}	1	1^{10}	1	1^{11}	1	1^{12}	1
		2	1	21	1	21^{2}	1	21^{3}	1	21^{4}	1	21^{5}	1	21^{6}	1	21^{7}	1	21^{8}	1	21^{9}	1	21^{10}	1
				3	1	31	1	31^{2}	1	31^{3}	1	31^{4}	1	31^{5}	1	31^{6}	1	31^{7}	1	31^{8}	1	31^{9}	1
						2^{2}	1	$2^{2}1$	1	$2^{2}1^{2}$	1	$2^{2}1^{3}$	1	$2^{2}1^{4}$	1	$2^{2}1^{5}$	1	$2^{2}1^{6}$	1	$2^{2}1^{7}$	1	$2^{2}1^{8}$	1
						4	2	41	2	41^{2}	2	41^{3}	2	41^{4}	2	41^{5}	2	41^{6}	2	41^{7}	2	41^{8}	2
								32	1	321	1	321^{2}	1	321^{3}	1	321^{4}	1	321^{5}	1	321^{6}	1	321^{7}	1
								5	4	51	4	51^{2}	4	51^{3}	4	51^{4}	4	51^{5}	4	51^{6}	4	51^{7}	4
										2^{3}	1	$2^{3}1$	1	$2^{3}1^{2}$	1	$2^{3}1^{3}$	1	$2^{3}1^{4}$	1	$2^{3}1^{5}$	1	$2^{3}1^{6}$	1
										42	2	421	2	421^{2}	2	421^{3}	2	421^{4}	2	421^{5}	2	421^{6}	2
										3^{2}	1	$3^{2}1$	1	$3^{2}1^{2}$	1	$3^{2}1^{3}$	1	$3^{2}1^{4}$	1	$3^{2}1^{5}$	1	$3^{2}1^{6}$	1
										6	11	61	11	61^{2}	11	61^{3}	11	61^{4}	11	61^{5}	11	61^{6}	11
												32^{2}	1	$32^{2}1$	1	$32^{2}1^{2}$	1	$32^{2}1^{3}$	1	$32^{2}1^{4}$	1	$32^{2}1^{5}$	1
												52	4	521	4	521^{2}	4	521^{3}	4	521^{4}	4	521^{5}	4
												43	2	431	2	431^{2}	2	431^{3}	2	431^{4}	2	431^{5}	2
												7	26	71	26	71^{2}	26	71^{3}	26	71^{4}	26	71^{5}	26
														2^{4}	1	$2^{4}1$	1	$2^{4}1^{2}$	1	$2^{4}1^{3}$	1	$2^{4}1^{4}$	1
														42^{2}	2	$42^{2}1$	2	$42^{2}1^{2}$	2	$42^{2}1^{3}$	2	$42^{2}1^{4}$	2
														$3^{2}2$	1	$3^{2}21$	1	$3^{2}21^{2}$	1	$3^{2}21^{3}$	1	$3^{2}21^{4}$	1
														62	11	621	11	621^{2}	11	621^{3}	11	621^{4}	11
														53	4	531	4	531^{2}	4	531^{3}	4	531^{4}	4
														4^{2}	3	$4^{2}1$	3	$4^{2}1^{2}$	3	$4^{2}1^{3}$	3	$4^{2}1^{4}$	3
														8	101	81	101	81^{2}	101	81^{3}	101	81^{4}	101
																32^{3}	1	$32^{3}1$	1	$32^{3}1^{2}$	1	$32^{3}1^{3}$	1
																52^{2}	4	$52^{2}1$	4	$52^{2}1^{2}$	4	$52^{2}1^{3}$	4
																432	2	4321	2	4321^{2}	2	4321^{3}	2
																72	26	721	26	721^{2}	26	721^{3}	26
																3^{3}	1	$3^{3}1$	1	$3^{3}1^{2}$	1	$3^{3}1^{3}$	1
																63	11	631	11	631^{2}	11	631^{3}	11
																54	8	541	8	541^{2}	8	541^{3}	8
																9	440	91	440	91^{2}	440	91^{3}	440
																		2^{5}	1	$2^{5}1$	1	$2^{5}1^{2}$	1
																		42^{3}	2	$42^{3}1$	2	$42^{3}1^{2}$	2
																		$3^{2}2^{2}$	1	$3^{2}2^{2}1$	1	$3^{2}2^{2}1^{2}$	1
																		62^{2}	11	$62^{2}1$	11	$62^{2}1^{2}$	11
																		532	4	5321	4	5321^{2}	4
																		$4^{2}2$	3	$4^{2}21$	3	$4^{2}21^{2}$	3
																		82	101	821	101	821^{2}	101
																		43^{2}	2	$43^{2}1$	2	$43^{2}1^{2}$	2
																		73	26	731	26	731^{2}	26
																		64	22	641	22	641^{2}	22
																		5^{2}	10	$5^{2}1$	10	$5^{2}1^{2}$	10
																		(10)	3,132	(10)1	3,132	$(10)1^{2}$	3,132
																				32^{4}	1	$32^{4}1$	1
																				52^{3}	4	$52^{3}1$	4
																				432^{2}	2	$432^{2}1$	2
																				72^{2}	26	$72^{2}1$	26
																				$3^{3}2$	1	$3^{3}21$	1
																				632	11	6321	11
																				542	8	5421	8
																				92	440	921	440
																				53^{2}	4	$53^{2}1$	4
																				$4^{2}3$	3	$4^{2}31$	3
																				83	101	831	101
																				74	52	741	52
																				65	44	651	44
																				(11)	40,457	(11)1	40,457
																						2^{6}	1
																						42^{4}	2
																						$3^{2}2^{3}$	1
																						62^{3}	11
																						532^{2}	4
																						$4^{2}2^{2}$	3
																						82^{2}	101
																						$43^{2}2$	2
																						732	26
																						642	22
																						$5^{2}2$	10
																						(10)2	3,132
																						3^{4}	1
																						63^{2}	11
																						543	8
																						93	440
																						4^{3}	4
																						84	202
																						75	104
																						6^{2}	66
																						(12)	1,274,068
1		2		3		6		11		26		59		182		675		3,990		45,144		1,323,363	

quadratic residue code will have a regular vertex degree of 14. Bordering this code adds a vertex of degree 29 and increases the degree of all other vertices to 15. It turns out that, by using only a single LC operation, this graph can be transformed into one with regular vertex degree 15. Furthermore, this LC operation may be performed on any vertex, except the vertex of degree 29. The "wheel graph" representation of the $[[6, 0, 4]]$ Hexacode, as shown in Figure 3.2b on page 22, is also a bordered quadratic residue code, and is also turned into a regular graph by a single LC operation on any vertex except the "centre" of the "wheel".

Theorem 5.13. *Let $G = (V = \{v_0, v_1, \ldots, v_{m-1}\}, E)$, be a Paley graph on $m = 4t + 1$ vertices. Form the graph $K = (V \cup \{v_m\}, E \cup \{\{u, v_m\} \mid u \in V\})$, i.e., add a vertex v_m and connect it to all existing vertices. K is a graph where the vertices $u \in V$ have degree $2t+1$ and vertex v_m has degree $4t+1$. K^u, where local complementation is performed on any vertex $u \in V$, will be a $2t+1$-regular graph.*

Proof. The neighbourhood of u is $N_u = v_m \cup N'_u$, where N'_u consists of $2t$ vertices. Since the vertex v_m has degree $4t+1$ in K and degree $2t$ in $K(N_u)$, its degree in K^u will be $2t + 1$. Since G is a Paley graph, and therefore a strongly regular graph with parameters $(4t + 1, 2t, t - 1, t)$, any vertex $w \in N'_u$ will have $t - 1$ neighbours in N'_u. Since w is also connected to v_m in K, it is connected to t of the $2t$ vertices in $K(N_u)$. The degree of w in $\overline{K(N_u)}$ will therefore remain t, and thus its degree in K^u will remain $2t + 1$. □

Corollary 5.14. *The graph corresponding to a bordered quadratic residue code of length n can be transformed into an $\frac{n}{2}$-regular graph by a single LC operation on any vertex, except the one added by the bordering.*

Note that Theorem 5.13 holds for Paley graphs over both prime and non-prime fields, and could also be extended to other strongly regular graphs. The result does, however, not hold for the $[[18, 0, 8]]$ code constructed by a technique similar to bordered quadratic residue, since it does not contain a strongly regular graph on 17 vertices. Even though quadratic residue and bordered quadratic residue codes achieve high distance and have regular graph representations, their vertex degree of $n/2$ is far from optimal for high n.

Chapter 6

Quantum Codes and Boolean Functions

6.1 Introduction to Boolean Functions

A *Boolean function* of n variables is a function $f : \mathbb{Z}_2^n \to \mathbb{Z}_2$. There are 2^n vectors $\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1}) \in \mathbb{Z}_2^n$, and each $\boldsymbol{x}$ can be interpreted as an integer $2^{n-1}x_{n-1}+\cdots+2x_1+x_0 \in \mathbb{Z}_{2^n}$. If we evaluate $f(\boldsymbol{x})$ for each $\boldsymbol{x} \in \mathbb{Z}_2^n$ in increasing order of the corresponding integers, we get a column vector $\boldsymbol{t}$, known as the *truth table* of f. A Boolean function can also be represented by the *algebraic normal form* (ANF) which is a sum of monomials of n variables. Let the 2^n monomials of n variables be ordered $1, x_0, x_1, x_0x_1, x_2, x_0x_2, \ldots, x_0x_1 \cdots x_{n-1}$, i.e., monomial number $k = 2^{n-1}k_{n-1} + \cdots + 2k_1 + k_0$ is $x^{(k)} = \prod_{k_i=1} x_i$. Note that we will sometimes use an abbreviated ANF notation for some many-term Boolean functions, e.g., $012, 12, 0$ is short for $x_0x_1x_2 + x_1x_2 + x_0$. The ANF of a Boolean function may be represented by the column vector $\boldsymbol{a}$ with 2^n binary coefficients, such that $f(\boldsymbol{x}) = \sum_{i \in \mathbb{Z}_{2^n}} a_i x^{(i)}$. To transform an ANF representation of a Boolean function into a truth table, we could perform 2^n evaluations of the function, but there is a more efficient method. This method also enables us to do the reverse transformation from truth table to ANF. The *algebraic normal form transformation* (ANFT) can be expressed as a multiplication of the vector $\boldsymbol{t}$ or $\boldsymbol{a}$ by a $2^n \times 2^n$ matrix A_n, such that $A_n\boldsymbol{t} = \boldsymbol{a}$ and $A_n\boldsymbol{a} = \boldsymbol{t}$.

Definition 6.1. The algebraic normal form transformation (ANFT) can be performed using the matrix A_n, which can be decomposed as an n-fold tensor product,

$$A_n = \bigotimes_{i=0}^{n-1} \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}. \tag{6.1}$$

The straight-forward way to perform ANFT is to generate the $2^n \times 2^n$ matrix A_n and then calculate the product $A_n\boldsymbol{t}$ or $A_n\boldsymbol{a}$. This operation has complexity $O(N^2)$, where $N = 2^n$. A much more efficient algorithm, with complexity $O(N \log N)$, can be used for any transformation T that can be decomposed into 2×2 matrices, $T = T_0 \otimes T_1 \otimes \cdots \otimes T_{n-1}$, where

$$T_i = \begin{pmatrix} t_i^{(0,0)} & t_i^{(0,1)} \\ t_i^{(1,0)} & t_i^{(1,1)} \end{pmatrix}. \tag{6.2}$$

Algorithm 6.1 is a simple version of this efficient algorithm. Different optimisation techniques, as described by Fuller et al. [19], can give faster implementations,

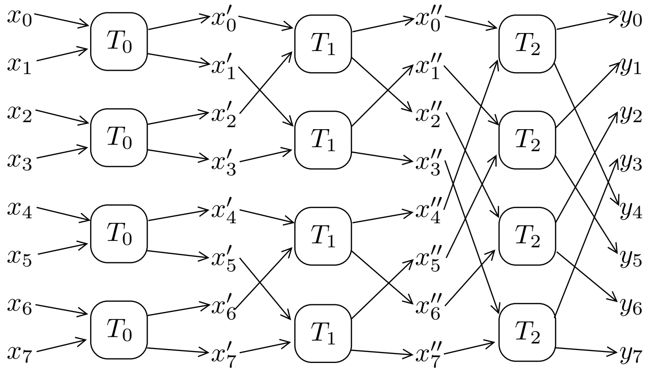


Figure 6.1: *Iterations of Algorithm for Tensor-Decomposable Transformations*

but with the same order of complexity. Figure 6.1 illustrates the iterations of the algorithm.

Algorithm 6.1 Algorithm for Tensor-Decomposable Transformations

```
Input     $n$: an integer
          $\boldsymbol{x}$: a vector of length $2^n$
          $\{T_0, T_1, \ldots, T_{n-1}\}$: $n$ $2 \times 2$ matrices, decomposition of $T$
Output    $\boldsymbol{y}$: the vector $T\boldsymbol{x}$

procedure Transform($n$, $\boldsymbol{x}$, $\{T_0, T_1, \ldots, T_{n-1}\}$)
    for $i \leftarrow 0$ to $n-1$ do
        for $j \leftarrow 0$ to $2^n - 1$ do
            if $((j \gg i) \mod 2) = 0$ then          ▷ "$\gg i$" means $i$ right bit shifts
                $y_j \leftarrow t_i^{(0,0)} x_j + t_i^{(0,1)} x_{j+2^i}$
            else
                $y_j \leftarrow t_i^{(1,0)} x_{j-2^i} + t_i^{(1,1)} x_j$
            end if
        end for
        for $j \leftarrow 0$ to $2^n - 1$ do
            $x_j \leftarrow y_j$
        end for
    end for
    return $\boldsymbol{y}$
end procedure
```

The *degree* of a Boolean function is the degree of the highest order term in its algebraic normal form. A function of degree 2 is called a *quadratic function*, and corresponds to an undirected graph. Functions of higher degree correspond to undirected *hypergraphs*. We will only consider simple graphs, i.e., graphs with no self-loops, as we will later see that linear and constant terms in Boolean functions can be ignored for our applications. The Boolean function $f(\boldsymbol{x})$ of n variables, $x_0, x_1, \ldots, x_{n-1}$, corresponds to a hypergraph on n vertices, $v_0, v_1, \ldots, v_{n-1}$, where the edge $\{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\} \in E$ iff the monomial $x_{i_1} x_{i_2} \cdots x_{i_k}$ occurs in the algebraic normal form of f. In particular, the quadratic function f can be represented by the adjacency matrix Γ, where $\Gamma_{i,j} = \Gamma_{j,i} = 1$ if $x_i x_j$ occurs in f, and $\Gamma_{i,j} = \Gamma_{j,i} = 0$ otherwise. Local complementation on a graph can then be described in terms of the corresponding quadratic Boolean function.

Definition 6.2. Let f be a quadratic Boolean function corresponding to the graph $G = (V, E)$. Let x_a be a variable of f corresponding to the vertex $v_a \in V$. An LC operation on the variable x_a produces the function

$$f'(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{\substack{v_j, v_k \in N_{v_a} \\ j<k}} x_j x_k \pmod 2, \tag{6.3}$$

where N_{v_a} comprises the neighbours of v_a in G.

Definition 6.3. The *Walsh-Hadamard transform* (WHT) of a function $f : \mathbb{Z}_2^n \to \mathbb{R}$ is given by the function $\widehat{f} : \mathbb{Z}_2^n \to \mathbb{R}$ defined as

$$\widehat{f}(\boldsymbol{b}) = 2^{-\frac{n}{2}} \sum_{\boldsymbol{x} \in \mathbb{Z}_2^n} f(\boldsymbol{x}) \cdot (-1)^{\boldsymbol{b} \cdot \boldsymbol{x}}, \tag{6.4}$$

where $\boldsymbol{b} \cdot \boldsymbol{x} = b_0 x_0 + b_1 x_1 + \cdots + b_{n-1} x_{n-1} \pmod 2$. WHT can be performed using the transformation matrix H_n, defined by the n-fold tensor product

$$H_n = \bigotimes_{i=0}^{n-1} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{6.5}$$

The WHT of a function can be calculated by Algorithm 6.1. Reverse WHT is performed by the same transformation, since $H_n = H_n^{-1}$.

Definition 6.4. Sometimes we prefer to work with the *bipolar representation*, $\chi_f : \mathbb{Z}_2^n \to \{-1, 1\}$, of a Boolean function f. We therefore define

$$\chi_f(\boldsymbol{x}) = (-1)^{f(\boldsymbol{x})}. \tag{6.6}$$

Definition 6.5. The WHT of χ_f is known as the *Walsh spectrum* of f, and is given by the function

$$\widehat{\chi_f}(\boldsymbol{b}) = 2^{-\frac{n}{2}} \sum_{\boldsymbol{x} \in \mathbb{Z}_2^n} \chi_f(\boldsymbol{x}) \cdot (-1)^{\boldsymbol{b} \cdot \boldsymbol{x}} = 2^{-\frac{n}{2}} \sum_{\boldsymbol{x} \in \mathbb{Z}_2^n} (-1)^{f(\boldsymbol{x}) + \boldsymbol{b} \cdot \boldsymbol{x}}. \tag{6.7}$$

Note that the addition in the exponent of -1 in (6.7) is modulo 2, and that all additions of binary values will be modulo 2, unless otherwise stated.

Theorem 6.6 (Parseval's Theorem)**.** *For any function, $f : \mathbb{Z}_2^n \to \mathbb{R}$, the following relationship holds*

$$\sum_{\boldsymbol{b} \in \mathbb{Z}_2^n} \widehat{f}^2(\boldsymbol{b}) = \sum_{\boldsymbol{x} \in \mathbb{Z}_2^n} f^2(\boldsymbol{x}). \tag{6.8}$$

Definition 6.7. The *correlation* of two Boolean functions, f and g, is

$$\kappa(f, g) = \sum_{\boldsymbol{x} \in \mathbb{Z}_2^n} \chi_f(\boldsymbol{x}) \cdot \chi_g(\boldsymbol{x}) = \sum_{\boldsymbol{x} \in \mathbb{Z}_2^n} (-1)^{f(\boldsymbol{x}) + g(\boldsymbol{x})}. \tag{6.9}$$

The Hamming distance between f and g, $d(f, g)$, which is the number of positions where their truth tables have different values, can be derived from their correlation, since

$$d(f, g) = 2^{n-1} - \frac{\kappa(f, g)}{2}. \tag{6.10}$$

High positive correlation of two functions implies that the distance between them is low, and therefore that they closely resemble each other, i.e., that they will often give the same output for the same input. It is easy to verify that $\kappa(f, g) = -\kappa(f, g + 1)$, so if f has high negative correlation with g, it will have equally high positive correlation with $g + 1$, the complement of g.

Definition 6.8. The *periodic autocorrelation* of the Boolean function f is given by the function $r : \mathbb{Z}_2^n \to \mathbb{R}$ defined as

$$r(\boldsymbol{a}) = \sum_{\boldsymbol{x} \in \mathbb{Z}_2^n} \chi_f(\boldsymbol{x}) \cdot \chi_f(\boldsymbol{x} + \boldsymbol{a}) = \sum_{\boldsymbol{x} \in \mathbb{Z}_2^n} (-1)^{f(\boldsymbol{x})+f(\boldsymbol{x}+\boldsymbol{a})}, \tag{6.11}$$

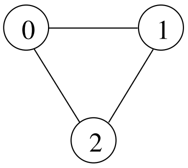
where $\boldsymbol{x} + \boldsymbol{a} = (x_0 + a_0, x_1 + a_1, \ldots, x_{n-1} + a_{n-1})$.

The autocorrelation coefficient $r(\boldsymbol{a})$, for some $\boldsymbol{a} \in \mathbb{Z}_2^n$, is the correlation of the functions $f(\boldsymbol{x})$ and $f(\boldsymbol{x} + \boldsymbol{a})$. The periodic autocorrelation function therefore gives the correlation of a function and all its *periodic shifts*, i.e., all possible combinations of variable inversions.

Theorem 6.9 (The Wiener-Khintchine Theorem)**.** *The periodic autocorrelation and the Walsh spectrum of the Boolean function f are related, since*

$$r(\boldsymbol{a}) = 2^{\frac{n}{2}} \widehat{\chi_f}^2(\boldsymbol{a}). \tag{6.12}$$

Example 6.10. Consider the graph shown below.



The graph has edges $E = \{\{0,1\}, \{0,2\}, \{1,2\}\}$ and corresponds to the Boolean function $f(\boldsymbol{x}) = x_0x_1 + x_0x_2 + x_1x_2$. The truth table and ANF of f is given by the following table, and so are the truth tables of $\widehat{f}$, χ_f and $\widehat{\chi_f}$. The table also gives the periodic autocorrelation, r, of f.

k_2	k_1	k_0	ANF	$f(\boldsymbol{k})$	$\widehat{f}(\boldsymbol{k})$	$\chi_f(\boldsymbol{k})$	$\widehat{\chi_f}(\boldsymbol{k})$	$r(\boldsymbol{k})$
0	0	0	0	0	$2^{\frac{1}{2}}$	1	0	8
0	0	1	0	0	$-2^{-\frac{1}{2}}$	1	$2^{\frac{1}{2}}$	0
0	1	0	0	0	$-2^{-\frac{1}{2}}$	1	$2^{\frac{1}{2}}$	0
0	1	1	1	1	0	-1	0	0
1	0	0	0	0	$-2^{-\frac{1}{2}}$	1	$2^{\frac{1}{2}}$	0
1	0	1	1	1	0	-1	0	0
1	1	0	1	1	0	-1	0	0
1	1	1	0	1	$2^{-\frac{1}{2}}$	-1	$-2^{\frac{1}{2}}$	-8

6.2 Propagation Criteria for Boolean Functions

Definition 6.11. A Boolean function, f, is *balanced* if the Hamming weight of its truth table is 2^{n-1}, or, equivalently, if

$$\sum_{\boldsymbol{x} \in \mathbb{Z}_2^n} (-1)^{f(\boldsymbol{x})} = 0. \tag{6.13}$$

Definition 6.12. The bipolar truth tables, χ_f and χ_g, of the Boolean functions f and g are orthogonal vectors if

$$\sum_{\boldsymbol{x} \in \mathbb{Z}_2^n} \chi_f(\boldsymbol{x}) \cdot \chi_g(\boldsymbol{x}) = \sum_{\boldsymbol{x} \in \mathbb{Z}_2^n} (-1)^{f(\boldsymbol{x})+g(\boldsymbol{x})} = 0, \tag{6.14}$$

which implies that the correlation $\kappa(f,g) = 0$ and that $f(\boldsymbol{x}) + g(\boldsymbol{x})$ is a balanced function.

Definition 6.13. A *linear Boolean function* is a function of degree less than two that can be written $\boldsymbol{x} \cdot \boldsymbol{b}$ for some $\boldsymbol{b} \in \mathbb{Z}_2^n$. The set of *affine functions* consists of the linear functions and their complements, i.e., functions of the form $\boldsymbol{x} \cdot \boldsymbol{b} + c$, where $\boldsymbol{b} \in \mathbb{Z}_2^n$ and $c \in \mathbb{Z}_2$.

Boolean functions have important applications in cryptography. Mappings from n bits to k bits, known as *S-boxes*, are important components in *block ciphers.* Such mappings can be viewed as k mappings from n bits to 1 bit, which can each be expressed as a Boolean function. These functions, and all their linear combinations, must satisfy certain criteria for the cipher to be secure. Common requirements are high algebraic degree and that the functions are balanced. We also want functions with a high degree of *nonlinearity*, which means that no affine functions closely resemble them, since the cipher could otherwise have been approximated by affine functions in an attack known as *linear cryptanalysis.* The nonlinearity of the Boolean function f can be derived from the Walsh spectrum $\widehat{\chi_f}$, since

$$2^{\frac{n}{2}} \widehat{\chi_f}(\boldsymbol{b}) = \kappa(f(\boldsymbol{x}), \boldsymbol{b} \cdot \boldsymbol{x}), \tag{6.15}$$

i.e., the Walsh spectrum gives the correlation of f with all possible linear functions, $\boldsymbol{x} \cdot \boldsymbol{b}$ for all $\boldsymbol{b} \in \mathbb{Z}_2^n$. The coefficient of the Walsh spectrum with highest magnitude corresponds to the affine function closest to f.

Definition 6.14. A Boolean function is *correlation immune* of order m if $\widehat{\chi_f}(\boldsymbol{b}) = 0$ for all $\boldsymbol{b}$ where $1 \leq w_H(\boldsymbol{b}) \leq m$. If the function is also balanced, i.e., $\widehat{\chi_f}(\boldsymbol{b}) = 0$ for all $\boldsymbol{b}$ where $0 \leq w_H(\boldsymbol{b}) \leq m$, then it is called *$m$-resilient.* The function obtained by fixing the value of m or fewer variables of an m-resilient function will still be a balanced function.

Boolean functions are also critical components in *stream ciphers*, where they can be used to represent *nonlinear combiners.* In this context, the correlation immunity or resilience of a Boolean function is important, since they give the number of *linear feedback shift registers* required to realise a correlation attack on the stream cipher.

Definition 6.15. A vector $\boldsymbol{s}$ of length 2^n is *flat* if $|s_i| = |s_j|$ for all $i, j \in \mathbb{Z}_{2^n}$, where s_k denotes the kth coordinate of $\boldsymbol{s}$.

It follows from Theorem 6.6 that

$$\sum_{\boldsymbol{b} \in \mathbb{Z}_2^n} \widehat{\chi_f}^2(\boldsymbol{b}) = \sum_{\boldsymbol{x} \in \mathbb{Z}_2^n} \chi_f^2(\boldsymbol{x}) = 2^n. \tag{6.16}$$

If the Walsh spectrum of a Boolean function is flat, all coefficients of the Walsh spectrum must therefore be 1 or -1. Such functions have the highest possible minimum distance to any affine function.

Definition 6.16. A Boolean function with a flat Walsh spectrum is called *perfect nonlinear* or *bent.*

A vector of length 2^n with entries 1 and -1 is the bipolar truth table of a Boolean function. The Walsh spectrum of a bent function therefore gives its *dual function*, which will also be bent. It follows from Theorem 6.9 that the condition for a function to be bent can also be expressed in terms of the periodic autocorrelation. The function f is bent if $r(\boldsymbol{a}) = 0$ for all $\boldsymbol{a}$ where $1 \leq w_H(\boldsymbol{a}) \leq n$, which means that the bipolar truth tables of $f(\boldsymbol{x})$ and $f(\boldsymbol{x} + \boldsymbol{a})$

must be orthogonal vectors. It can be shown that only Boolean functions of an even number of variables can be bent, and that bent functions can never be balanced. In general, it is not possible to optimise all desired properties of a Boolean function, so some compromise must be made. As a generalisation of the bent criterion, the *propagation criterion* of Boolean functions was defined by Preneel et al. [44] and later studied by Carlet [12].

Definition 6.17. A Boolean function satisfies PC(l), the *propagation criterion* of degree l, if $r(\boldsymbol{a}) = 0$ for all $\boldsymbol{a}$ where $1 \leq w_H(\boldsymbol{a}) \leq l$.

Bent functions satisfy PC(n), but functions satisfying PC(l) for $l < n$ can in addition be balanced and are therefore better suited for cryptographic purposes. Boolean functions used in cryptography should also have good properties when subsets of their variables are set to fixed values.

Definition 6.18. The Boolean function f satisfies the *propagation criterion* of degree l and order m if any function obtained by setting m or fewer variables of f to any fixed values satisfies PC(l). It is required that the set of fixed bits and the set of modified bits are disjoint, and therefore that $l + m \leq n$.

A function satisfying PC(l) of order m is resistant against attacks where the attacker knows the value of up to m input bits for a large number of plaintext/ciphertext pairs. The attacker is further able to modify up to l of the other input bits and compare the modified output with the original. This technique is known as *differential cryptanalysis.* In a more general scenario, the sets of m known bits and l modified bits need not be disjoint. Preneel et al. [44] define the *extended propagation criterion*, where this restriction is removed. Carlet [12] reformulated this criterion as follows.

Proposition 6.19 (Carlet [12])**.** *The Boolean function f satisfies* EPC(l) *of order m, the extended propagation criterion of degree l and order m, if $f(\boldsymbol{x}) + f(\boldsymbol{x}+\boldsymbol{a})$ is m-resilient for any $\boldsymbol{a}$ where $1 \leq w_H(\boldsymbol{a}) \leq l$. An equivalent requirement is that*

$$\sum_{\boldsymbol{x}\in\mathbb{Z}_2^n} (-1)^{f(\boldsymbol{x})+f(\boldsymbol{x}+\boldsymbol{a})+\boldsymbol{b}\cdot\boldsymbol{x}} = 0, \tag{6.17}$$

or that the bipolar truth tables of $f(\boldsymbol{x})$ and $f(\boldsymbol{x}+\boldsymbol{a}) + \boldsymbol{b}\cdot\boldsymbol{x}$ must be orthogonal, for any $\boldsymbol{a}$ and $\boldsymbol{b}$ where $w_H(\boldsymbol{a}) \leq l$, $w_H(\boldsymbol{b}) \leq m$, and $w_H(\boldsymbol{a})$ and $w_H(\boldsymbol{b})$ are not both zero. The same criterion can be used for PC, if we add the restriction that $\boldsymbol{a}$ and $\boldsymbol{b}$ have disjoint supports, i.e., that they never both have the value 1 in the same coordinate.

Definition 6.20. Given two vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{Z}_2^n$, we say that $\boldsymbol{u}$ is *covered by* $\boldsymbol{v}$, denoted $\boldsymbol{u} \preceq \boldsymbol{v}$, if $u_i \leq v_i$ for all $i \in \mathbb{Z}_n$. We define the negation of $\boldsymbol{u} \in \mathbb{Z}_2^n$ as $\overline{\boldsymbol{u}} = (u_0 + 1, u_1 + 1, \ldots, u_{n-1} + 1)$. Given a vector $\boldsymbol{u} \in \mathbb{Z}_2^n$, let $V_{\boldsymbol{u}} \subseteq \mathbb{Z}_2^n$ be the set $V_{\boldsymbol{u}} = \{\boldsymbol{v} \in \mathbb{Z}_2^n \mid \boldsymbol{v} \preceq \boldsymbol{u}\}$. Given a vector $\boldsymbol{u} \in \mathbb{Z}_2^n$ and a subset $V \subseteq \mathbb{Z}_2^n$, $\boldsymbol{u} + V = \{\boldsymbol{u} + \boldsymbol{v} \mid \boldsymbol{v} \in V\}$ is a *coset* of V.

Definition 6.21. Let the *fixed-periodic autocorrelation* of a Boolean function f be defined as

$$r(\boldsymbol{a}, \boldsymbol{\mu}, \boldsymbol{k}) = \sum_{\boldsymbol{x}\in\boldsymbol{k}+V_{\overline{\boldsymbol{\mu}}}} (-1)^{f(\boldsymbol{x})+f(\boldsymbol{x}+\boldsymbol{a})}, \tag{6.18}$$

where $\boldsymbol{k} \preceq \boldsymbol{\mu}$ and $\boldsymbol{a} \preceq \overline{\boldsymbol{\mu}}$. The vector $\boldsymbol{\mu}$ indicates a subset of the variables of f which are fixed to the values given by $\boldsymbol{k}$, while the vector $\boldsymbol{a}$ indicates a subset of the remaining variables which are periodically shifted (flipped).

Proposition 6.22. *It can be shown [15] that a Boolean function satisfies* PC(l) *of order m if $r(\boldsymbol{a},\boldsymbol{\mu},\boldsymbol{k}) = 0$ for all $\boldsymbol{a},\boldsymbol{\mu},\boldsymbol{k} \in \mathbb{Z}_2^n$ where $\boldsymbol{k} \preceq \boldsymbol{\mu}$, $\boldsymbol{a} \preceq \overline{\boldsymbol{\mu}}$, $1 \leq w_H(\boldsymbol{a}) \leq l$, and $0 \leq w_H(\boldsymbol{\mu}) \leq m$.*

Definition 6.23. Let the *aperiodic autocorrelation* of a Boolean function f be defined as

$$s(\boldsymbol{a},\boldsymbol{k}) = \sum_{\boldsymbol{x} \in \boldsymbol{k}+V_{\overline{\boldsymbol{a}}}} (-1)^{f(\boldsymbol{x})+f(\boldsymbol{x}+\boldsymbol{a})}, \tag{6.19}$$

where $\boldsymbol{k} \preceq \boldsymbol{a}$. Variables indicated by $\boldsymbol{a}$ are shifted aperiodically, and are therefore assigned the fixed values given by $\boldsymbol{k}$.

Definition 6.24. Let the *fixed-aperiodic autocorrelation* of a Boolean function f be defined as

$$s(\boldsymbol{a},\boldsymbol{\mu},\boldsymbol{k}) = \sum_{\boldsymbol{x} \in \boldsymbol{k}+V_{\overline{\boldsymbol{\mu}}}} (-1)^{f(\boldsymbol{x})+f(\boldsymbol{x}+\boldsymbol{a})}, \tag{6.20}$$

where $\boldsymbol{a},\boldsymbol{k} \preceq \boldsymbol{\mu}$. The vector $\boldsymbol{\mu}$ indicates a subset of the variables of f which are fixed to the values given by $\boldsymbol{k}$, including those that are aperiodically shifted as indicated by $\boldsymbol{a}$. Let $\boldsymbol{\theta} = \boldsymbol{\mu} + \boldsymbol{a}$ indicate the variables that are fixed but not shifted. The supports of $\boldsymbol{\theta}$ and $\boldsymbol{a}$ are disjoint by definition.

Just as PC can be expressed in terms of fixed-periodic autocorrelation, as described in Proposition 6.22, the fixed-aperiodic autocorrelation also corresponds to a propagation criterion which we call the *aperiodic propagation criterion* (APC) [15].

Definition 6.25. A Boolean function satisfies APC(l) of order m, the aperiodic propagation criterion of degree l and order m, if $s(\boldsymbol{a},\boldsymbol{\mu},\boldsymbol{k}) = 0$ for all $\boldsymbol{a},\boldsymbol{\mu},\boldsymbol{k} \in \mathbb{Z}_2^n$ where $\boldsymbol{a},\boldsymbol{k} \preceq \boldsymbol{\mu}$, $\boldsymbol{\theta} = \boldsymbol{\mu} + \boldsymbol{a}$, $1 \leq w_H(\boldsymbol{a}) \leq l$, $0 \leq w_H(\boldsymbol{\theta}) \leq m$, and $\boldsymbol{a}$ and $\boldsymbol{\theta}$ have disjoint supports.

In a cryptographic scenario, a Boolean function satisfying APC(l) of order m is resistant against attacks where the attacker knows up to $m + l$ of the input bits, and is further allowed to change up to l of these known bits.

Definition 6.26. A Boolean function f has *APC distance d* if it satisfies APC(l) of order m for all positive integers l and m such that $l + m < d$.

Definition 6.27. We define the weight operator $w_H(\boldsymbol{a},\boldsymbol{b}) = w_H(\boldsymbol{a}) + w_H(\boldsymbol{b}) - w_H(\boldsymbol{a} \wedge \boldsymbol{b})$, where $\boldsymbol{a} \wedge \boldsymbol{b} = (a_0b_0, a_1b_1, \ldots, a_{n-1}b_{n-1})$.

Proposition 6.28. *It can be shown [15] that the APC distance of the Boolean function f is equal to the smallest nonzero $w_H(\boldsymbol{a},\boldsymbol{b})$ where*

$$\sum_{\boldsymbol{x} \in \mathbb{Z}_2^n} (-1)^{f(\boldsymbol{x})+f(\boldsymbol{x}+\boldsymbol{a})+\boldsymbol{b}\cdot\boldsymbol{x}} \neq 0. \tag{6.21}$$

Proposition 6.29. *If the Boolean function f has APC distance d, then $f(\boldsymbol{x}) + f(\boldsymbol{x}+\boldsymbol{a})$ must be $(d - w_H(\boldsymbol{a}) - 1)$-resilient, for all $\boldsymbol{a}$ where $w_H(\boldsymbol{a}) < d$.*

6.3 Quantum Codes as Boolean Functions

Definition 6.30. The Boolean function $f(\boldsymbol{x})$ corresponds to the vector $\boldsymbol{s} = 2^{-\frac{n}{2}}(-1)^{f(\boldsymbol{x})}$, which can be interpreted as the probability distribution vector of the quantum state

$$|\psi_f\rangle = 2^{-\frac{n}{2}} \sum_{\boldsymbol{x} \in \mathbb{Z}_2^n} (-1)^{f(\boldsymbol{x})} |\boldsymbol{x}\rangle \,. \tag{6.22}$$

We can find the single quantum state represented by a self-dual quantum code once we know the Boolean function corresponding to an equivalent graph code. We interpret the Boolean function as a quantum state as described in Definition 6.30. (See section 2.1 for the definition of a quantum state.) The normalisation factor $2^{-\frac{n}{2}}$ ensures that the sum of all probabilities is 1, but in many cases it can be ignored, and the bipolar truth table of f can then be interpreted directly as a quantum state.

Example 6.31. The function $f(\boldsymbol{x}) = x_0x_1 + x_0x_2 + x_1x_2$ corresponds to the bipolar vector $\boldsymbol{s} = 2^{-\frac{3}{2}}(1,1,1,-1,1,-1,-1,-1)^T$. This is the probability distribution vector of the quantum state $|\psi_f\rangle = 2^{-\frac{3}{2}}(|000\rangle + |001\rangle + |010\rangle - |011\rangle + |100\rangle - |101\rangle - |110\rangle - |111\rangle)$.

We will only consider *bipolar quantum states*, i.e., states where all coefficients of the probability distribution vectors are either 1 or −1. Parker and Rijmen [40] define the more general *algebraic polar form*, $\boldsymbol{s} = m(\boldsymbol{x})(-1)^{p(\boldsymbol{x})}$, where the two Boolean functions $m(\boldsymbol{x})$ and $p(\boldsymbol{x})$ describes magnitude and phase, respectively. All vectors with coefficients from the set $\{-1,0,1\}$ are covered by this definition. We will only study the case where $m(\boldsymbol{x}) = 1$.

The self-dual quantum codes studied in the previous chapters have all corresponded to *quadratic* Boolean functions. A bipolar quantum state may correspond to a Boolean function of any degree, but unlike quadratic functions, functions of degree higher than two do not correspond to stabilizer codes or additive codes over GF(4). By going back to the original definition of quantum error correcting codes, as described in section 2.3, it is possible to extend the definition of zero-dimensional quantum codes to what might be called "hypergraph codes", corresponding to non-quadratic Boolean functions. To guarantee that an error on a single quantum state can be detected, the errored state must be orthogonal to the original state. The distance of a zero-dimensional quantum code is therefore the weight of the minimum weight quantum error operator that gives an errored state not orthogonal to the original state. In quantum error correction we only need to consider the three errors described by the Pauli matrices σ_x, σ_z and σ_y, corresponding, respectively, to bit-flip, phase-flip and combined bit-flip and phase-flip. A Pauli error operating on the hypergraph state $|\psi_f\rangle$ corresponds to an operation on the corresponding Boolean function $f(\boldsymbol{x})$. Let the binary vector $\boldsymbol{a}$ indicate which qubits have been bit-flipped, i.e., qubit number k has been bit-flipped iff $a_k = 1$. These bit-flips correspond to a transformation on $|\psi_f\rangle$,

$$|\psi_f\rangle \to \left(\bigotimes_{a_k=1} \sigma_x^{(k)} \bigotimes_{a_k=0} I^{(k)}\right) |\psi_f\rangle \,. \tag{6.23}$$

The corresponding bit-flip operation on the Boolean function $f(\boldsymbol{x})$ is

$$f(\boldsymbol{x}) \to f(\boldsymbol{x} + \boldsymbol{a}). \tag{6.24}$$

Similarly, let the binary vector $\boldsymbol{b}$ indicate the qubits which have been phase-flipped. We get the error operation

$$|\psi_f\rangle \to \left(\bigotimes_{b_k=1} \sigma_z^{(k)} \bigotimes_{b_k=0} I^{(k)}\right) |\psi_f\rangle \,, \tag{6.25}$$

and we can express the same phase-flips in terms of a Boolean function,

$$f(\boldsymbol{x}) \to f(\boldsymbol{x}) + \boldsymbol{b} \cdot \boldsymbol{x}. \tag{6.26}$$

The third error we must consider is the combined bit-flip and phase-flip. We have used the definition $\sigma_y = i\sigma_x\sigma_z$, but the overall phase factor i has no significance and can be ignored. Neither is the order of the operations important, since $\sigma_z\sigma_x = -\sigma_x\sigma_z$, and this phase factor can also be ignored.

$$|\psi_f\rangle \rightarrow \left(\bigotimes_{\substack{a_k=1\\ b_k=1}} \sigma_y^{(k)} \bigotimes_{a_k=0} I^{(k)} \bigotimes_{b_k=0} I^{(k)} \right) |\psi_f\rangle \qquad (6.27)$$

corresponds, up to a global phase factor, to an operation on a Boolean function,

$$f(\boldsymbol{x}) \rightarrow f(\boldsymbol{x}+\boldsymbol{a}) + \boldsymbol{b}\cdot\boldsymbol{x} + \boldsymbol{b}\cdot\boldsymbol{a}. \qquad (6.28)$$

The linear term $\boldsymbol{b}\cdot\boldsymbol{a}$ can safely be ignored, because, as $\boldsymbol{a}$ and $\boldsymbol{b}$ are fixed, it reduces to a constant and therefore contributes another global phase factor. We can finally consider a general Pauli error,

$$|\psi_f\rangle \rightarrow \left(\bigotimes_{\substack{a_k=1\\ b_k=0}} \sigma_x^{(k)} \bigotimes_{\substack{a_k=0\\ b_k=1}} \sigma_z^{(k)} \bigotimes_{\substack{a_k=1\\ b_k=1}} \sigma_y^{(k)} \bigotimes_{\substack{a_k=0\\ b_k=0}} I^{(k)} \right) |\psi_f\rangle\,, \qquad (6.29)$$

which can also be expressed in terms of a Boolean function,

$$f(\boldsymbol{x}) \rightarrow f(\boldsymbol{x}+\boldsymbol{a}) + \boldsymbol{b}\cdot\boldsymbol{x}. \qquad (6.30)$$

Note that the error operator σ_y, although composed of two errors, only counts as one error. The weight of a quantum error operator in terms of the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is therefore given by the weight function $w_H(\boldsymbol{a},\boldsymbol{b})$, as defined in Definition 6.27. We can then define the distance of the zero-dimensional quantum code corresponding to the state $|\psi_f\rangle$ as the smallest nonzero $w_H(\boldsymbol{a},\boldsymbol{b})$ such that the bipolar truth tables of $f(\boldsymbol{x})$ and $f(\boldsymbol{x}+\boldsymbol{a})+\boldsymbol{b}\cdot\boldsymbol{x}$ are orthogonal vectors. By comparing this criterion to the definition of APC distance in Proposition 6.28, we see that they are equal.

Theorem 6.32. *Let $|\psi_f\rangle$ be a bipolar quantum state with probability distribution vector $2^{-\frac{n}{2}}(-1)^{f(\boldsymbol{x})}$, where f is a Boolean function. Then $|\psi_f\rangle$ corresponds to an $[[n,0,d]]$ quantum code where d is the APC distance of f. Conversely, any Boolean function with APC distance d corresponds to an $[[n,0,d]]$ quantum code.*

A Boolean function of degree higher than two corresponds to a *non-quadratic quantum code* which is not equivalent to any stabilizer code or additive code over GF(4). While we can quickly find the APC distance of a quadratic Boolean function as the distance of the corresponding additive code over GF(4) using Algorithm 3.1, no such shortcut is known for higher degree functions. We must therefore check the condition (6.21) for all errors of increasing weight, until an error not satisfying the condition is found, as described in Algorithm 6.2.

Example 6.33. The hypergraph shown in Figure 6.2 corresponds to the cubic Boolean function $f(\boldsymbol{x}) = 012, 03, 04, 13, 15, 24, 25$. It can be verified that no error with $w_H(\boldsymbol{a},\boldsymbol{b}) \leq 2$ satisfies (6.21). There are, however, vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ with $w_H(\boldsymbol{a},\boldsymbol{b}) = 3$ such that (6.21) is satisfied. One such error is given by $\boldsymbol{a} = (0,0,0,0,0,1)$ and $\boldsymbol{b} = (0,1,1,0,0,0)$, and it produces the errored state $f(\boldsymbol{x}+\boldsymbol{a})+\boldsymbol{b}\cdot\boldsymbol{x} = 012, 03, 04, 12, 13, 15, 24, 25$. We verify that $f(\boldsymbol{x})+f(\boldsymbol{x}+\boldsymbol{a})+\boldsymbol{b}\cdot\boldsymbol{x} = x_1x_2$ is not a balanced function. This means that the APC distance of f is 3, and that f corresponds to a $[[6,0,3]]$ non-quadratic quantum code.

Note that Proposition 4.6 does not hold for hypergraphs. Connected hypergraphs where all vertices have high degree may still correspond to Boolean functions with APC distance 1.

Algorithm 6.2 Finding the APC Distance of a Boolean Function

```
Input    f: a Boolean function
         n: the number of variables of f
Output   d: the APC distance of f

procedure FindAPCdistance($f$, $n$)
    $\boldsymbol{s} \leftarrow \chi_f$, the bipolar truth table of $f(\boldsymbol{x})$
    for all $\boldsymbol{a} \in \mathbb{Z}_2^n$ and $\boldsymbol{b} \in \mathbb{Z}_2^n$ in increasing order of $w_H(\boldsymbol{a},\boldsymbol{b})$ do
        $\boldsymbol{s}' \leftarrow$ the bipolar truth table of $f(\boldsymbol{x}+\boldsymbol{a}) + \boldsymbol{b}\cdot\boldsymbol{x}$
        if $\boldsymbol{s}\cdot\boldsymbol{s}' \neq 0$ then
            return $w_H(\boldsymbol{a},\boldsymbol{b})$
        end if
    end for
end procedure
```

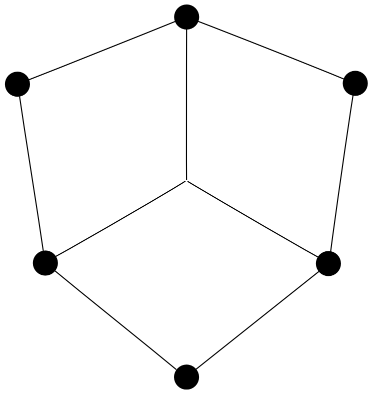


Figure 6.2: *A Hypergraph Corresponding to a* $[[6,0,3]]$ *Quantum Code*

6.4 The $\{I, H, N\}^n$ Transform Set

As described in section 5.1, Hein et al. [30] showed that local complementation on a graph corresponds to a transformation defined by the tensor product of the matrices $\tau_x = \sqrt{-i\sigma_x}$ and $\tau_z = \sqrt{i\sigma_z}$. The LC orbit can then be generated by repeated transformations of this form.

Definition 6.34. Let $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ be a set of k 2×2 unitary matrices. The transform set $\mathcal{T}^n$ is then the set of k^n $2^n \times 2^n$ transformation matrices of the form $U = U_0 \otimes U_1 \otimes \cdots \otimes U_{n-1}$, where $U_i \in \mathcal{T}$. Given a vector $\boldsymbol{s}$ of length 2^n, the k^n transforms $\boldsymbol{S} = U\boldsymbol{s}$, for all possible choices of $U \in \mathcal{T}^n$, is a multispectra with $(2k)^n$ spectral points. We refer to this multispectra as the spectrum with respect to the $\mathcal{T}^n$ transform.

Definition 6.35. Let $\mathcal{D}$ be the infinite set of all 2×2 diagonal and anti-diagonal unitary matrices, i.e, matrices of the form

$$\begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} \text{ and } \begin{pmatrix} 0 & a \\ b & 0 \end{pmatrix},$$

where $a, b \in \mathbb{R}$. A transform $F \in \mathcal{D}^n$ is then a tensor product of any n matrices from $\mathcal{D}$.

Proposition 6.36 (Riera and Parker [51])**.** *Applying a transformation from* $\mathcal{D}^n$ *to a vector* $\boldsymbol{s}$ *will not change the magnitudes of the coefficients of* $\boldsymbol{s}$*. If two*

2×2 unitary matrices, A and B, satisfy $FA = B$ where $F \in \mathcal{D}$, they can be considered equivalent, $A \simeq B$.

Theorem 6.37 (Riera and Parker [51])**.** *To within a subsequent transformation from $\mathcal{D}^n$, all members of the LC orbit of the Boolean function f can be found as a subset of the finite set of 3^n $\{I, \tau_x, \tau_x\tau_z\}^n$ transforms of f.*

Example 6.38. For a Boolean function $f(\boldsymbol{x})$ of $n = 2$ variables, we find the corresponding vector $\boldsymbol{s} = 2^{-\frac{n}{2}}(-1)^{f(\boldsymbol{x})}$. We then find the 9 vectors $\boldsymbol{S} = U\boldsymbol{s}$, for all $U \in \{I\otimes I,\ I\otimes\tau_x,\ I\otimes\tau_x\tau_z,\ \tau_x\otimes I,\ \tau_x\otimes\tau_x,\ \tau_x\otimes\tau_x\tau_z,\ \tau_x\tau_z\otimes I,\ \tau_x\tau_z\otimes\tau_x,\ \tau_x\tau_z\otimes\tau_x\tau_z\}$. By using appropriate transformations from $\mathcal{D}^n$, some of the vectors $\boldsymbol{S}$ may be transformed into bipolar vectors which correspond to truth tables of Boolean functions. We use ANFT to recover all such functions, and this set will comprise all members of the LC orbit of f.

Definition 6.39. Let

$$I = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}, \quad H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}, \quad N = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & i\\ 1 & -i\end{pmatrix},$$

where $i^2 = -1$, be the Identity, *Hadamard*, and *Negahadamard* transformations, respectively.

Let f be a Boolean function on n variables and $\boldsymbol{s} = 2^{-\frac{n}{2}}(-1)^{f(\boldsymbol{x})}$ be a vector of length 2^n. The spectrum with respect to the $\{I,H,N\}^n$ transform is the set of 3^n transforms $\boldsymbol{S} = U\boldsymbol{s}$, for all $U \in \{I,H,N\}^n$.

Theorem 6.40 (Riera and Parker [51])**.** *It can be shown that $N \simeq \tau_x$ and $H \simeq \tau_x\tau_z$. It follows that the LC orbit of a Boolean function can be generated by the transform set $\{I,H,N\}^n$.*

There is a good reason for choosing the transform set $\{I,H,N\}^n$ instead if $\{I,\tau_x,\tau_x\tau_z\}^n$. We have already seen that the Hadamard transformation is used in both the theory of quantum computing and in the analysis of Boolean functions. The $\{H\}^n$ transform, which is simply the Walsh-Hadamard transform as defined in (6.5), is an n-dimensional 2-point *discrete Fourier transform.* We know from Theorem 6.9 that the $\{H\}^n$ spectrum is the “dual” of the periodic autocorrelation, defined in Definition 6.8. The $\{I,H\}^n$ spectrum can be shown to be the “dual” of the fixed-periodic autocorrelation, defined in Definition 6.21. The Hadamard and Negahadamard transformations together define a two times oversampled discrete Fourier transform, and the $\{H,N\}^n$ spectrum is therefore the “dual” of the aperiodic autocorrelation, defined in Definition 6.23. Finally, the $\{I,H,N\}^n$ spectrum is the “dual” of the fixed-aperiodic autocorrelation, defined in Definition 6.24, which is related to the aperiodic propagation criterion, as shown in Definition 6.25.

Definition 6.41. Let f be a function, $f : \mathbb{Z}_2^n \to \mathbb{Z}_m$, where $m \in 2\mathbb{N}$, i.e., m is some even positive integer. Let $u = e^{\frac{2\pi i}{m}}$, i.e., u is a complex mth root of 1. The vector $\boldsymbol{s} = u^{f(\boldsymbol{x})}$ is *Boolean flat* if it is flat and there exists some $F \in \mathcal{D}^n$ such that $F\boldsymbol{s} = u^{\frac{m}{2}f'(\boldsymbol{x})} = (-1)^{f'(\boldsymbol{x})}$, where f' is a Boolean function. It follows that f and f' may be interpreted as equivalent functions.

Definition 6.42. For $k \in \mathbb{N}$, we define the “$\mathbb{Z}_k$-phase-flip” matrix

$$\rho_k = \begin{pmatrix}1 & 0\\ 0 & e^{\frac{2\pi i}{k}}\end{pmatrix},$$

Proposition 6.43. *Let $\boldsymbol{s} = u^{f(\boldsymbol{x})}$ where $u = e^{\frac{2\pi i}{m}}$, $f : \mathbb{Z}_2^n \to \mathbb{Z}_m$, and $m \in 2\mathbb{N}$. The vector $\boldsymbol{s}$ is Boolean flat if we can write $\boldsymbol{s} = u^{\frac{m}{2}f'(\boldsymbol{x})+h(\boldsymbol{x})}$, where f' is a Boolean function and h is an affine function from $\mathbb{Z}_2^n$ to $\mathbb{Z}_m$.*

Proof. Let $\boldsymbol{s}'$ be the transform

$$\boldsymbol{s}' = \left((\rho_m^b)^{(k)} \bigotimes_{l \neq k} I^{(l)} \right) \boldsymbol{s},$$

where $b \in \mathbb{Z}_m$. It can then be verified that $\boldsymbol{s}' = u^{f(\boldsymbol{x})+bx_k}$. Let $\boldsymbol{s}'' = u^c\boldsymbol{s}$, where $u = e^{\frac{2\pi i}{m}}$ and $c \in \mathbb{Z}_m$. It can be verified that $\boldsymbol{s}'' = u^{f(\boldsymbol{x})+c}$. Thus all linear and constant terms of $h(\boldsymbol{x})$ can be removed by a transformation $F \in \mathcal{D}^n$ of the form

$$F = u^c \bigotimes_{k=0}^{n-1} \rho_m^{b_k}, \tag{6.31}$$

where $u = e^{\frac{2\pi i}{m}}$ and $b_k, c \in \mathbb{Z}_m$. □

Proposition 6.44 (Riera and Parker [51])**.** *We can perform local complementation on the variable x_k of the Boolean function f with the transformation*

$$U_k = N^{(k)} \bigotimes_{l \neq k} I^{(l)}. \tag{6.32}$$

If $\boldsymbol{s} = 2^{-\frac{n}{2}}(-1)^{f(\boldsymbol{x})}$, then $\boldsymbol{S} = U_k\boldsymbol{s}$ will be Boolean flat with coefficients from the set $\{w^a \mid w^4 = -1, a \in \mathbb{Z}_8\}$. Furthermore, $\boldsymbol{S}$ can always be expressed as $\boldsymbol{S} = w^{4f'(\boldsymbol{x})+2h(\boldsymbol{x})+c}$, where f' is a Boolean function, h is an affine function from $\mathbb{Z}_2^n$ to $\mathbb{Z}_4$, and $c \in \mathbb{Z}_8$.

It follows from Proposition 6.43 that the vector $\boldsymbol{S} = U_m\boldsymbol{s}$, where U_m is of the form given by (6.32), can be turned into a bipolar vector by some transformation of the form

$$F = w^a \bigotimes_{k=0}^{n-1} \rho_4^{b_k}, \tag{6.33}$$

where $w^4 = -1$, $a \in \mathbb{Z}_8$, and $b_k \in \mathbb{Z}_4$. We can then implement a sequence of LC operations by transformations U_m of the form given by (6.32), with appropriate transformations F of the form given by (6.33) performed after each U_m.

Lemma 6.45. *Define the product $U = U_kU_{k-1}\cdots U_1$, where $k \in \mathbb{N}$ and $U_i \in \{I, H, N, \sigma_x, \rho_4\}$. Then $U = U_3'U_2'U_1'$, where $U_1' \in \{I, H, N\}$, $U_2' \in \{I, \sigma_x\}$, and $U_3' = w^a\rho_4^b$, where $w^4 = -1$, $a \in \mathbb{Z}_8$, and $b \in \mathbb{Z}_4$.*

Proof. The result follows from the identities,

$$\begin{aligned}
HH &= I, \\
HN &= \rho_4, \\
H\rho_4 &= N, \\
H\sigma_x &= \sigma_z H = \rho_4^2 H, \\
NH &= w\rho_4^3\sigma_x N, \\
NN &= w\rho_4^3 H, \\
N\rho_4 &= \sigma_x H, \\
N\sigma_x &= -\sigma_y N = i\rho_4^2\sigma_x N, \\
\sigma_x\rho_4 &= i\rho_4^3\sigma_x, \\
\sigma_x\sigma_x &= I.
\end{aligned} \tag{6.34}$$

Theorem 6.46. *Given any sequence of transformations $U = U_k U_{k-1} \cdots U_1$, where $k \in \mathbb{N}$ and $U_i \in \{I, H, N, \sigma_x, \rho_4\}^n$. Then $U = U_3' U_2' U_1'$, where $U_1' \in \{I, H, N\}^n$, $U_2' \in \{I, \sigma_x\}^n$, and $U_3' = w^a \bigotimes_{k=0}^{n-1} \rho_4^{b_k}$, where $w^4 = -1$, $a \in \mathbb{Z}_8$, and $b_k \in \mathbb{Z}_4$.*

Proof. This result is a simple generalisation of Lemma 6.45 to a transformation on n qubits. □

Corollary 6.47. *Any sequence of LC operations is equivalent to a single transformation from $\{I, H, N\}^n$ followed by some transformation from $\mathcal{D}^n$ of the form*

$$F = w^a \left(\bigotimes_{k=0}^{n-1} \rho_4^{b_k} \right) \left(\bigotimes_{l=0}^{n-1} \sigma_x^{c_l} \right), \tag{6.35}$$

where $w^4 = -1$, $a \in \mathbb{Z}_8$, $b_k \in \mathbb{Z}_4$, and $c_l \in \mathbb{Z}_2$.

Definition 6.48. A function $f : \mathbb{Z}_2^n \to \mathbb{Z}_m$, where $m \in \mathbb{N}$, can be described by the truth table $\boldsymbol{t}$ or the algebraic normal form $\boldsymbol{a}$, both being vectors of length 2^n with coefficients from $\mathbb{Z}_m$. The *generalised algebraic normal form transformation*, ANFT$_m$, can be used to transform $\boldsymbol{t}$ into $\boldsymbol{a}$ and vice versa. To transform a truth table into ANF, we use

$$\boldsymbol{a} = \left(\bigotimes_{k=0}^{n-1} \begin{pmatrix} 1 & 0 \\ m-1 & 1 \end{pmatrix} \right) \boldsymbol{t} \pmod{m}. \tag{6.36}$$

Transformation from ANF to truth table is performed by

$$\boldsymbol{t} = \left(\bigotimes_{k=0}^{n-1} \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \right) \boldsymbol{a} \pmod{m}. \tag{6.37}$$

If $\boldsymbol{s} = 2^{-\frac{n}{2}} (-1)^{f(\boldsymbol{x})}$ and $\boldsymbol{S} = U\boldsymbol{s}$ is flat for some choice of $U \in \{I, H, N\}^n$, then we can recover the function $\boldsymbol{S} = w^{g(\boldsymbol{x})}$ by using ANFT$_8$. By separating the monomials of $g(\boldsymbol{x})$ that are divisible by 4 and those that are not, we can write $\boldsymbol{S} = w^{4f'(\boldsymbol{x}) + h(\boldsymbol{x})}$, where f' is a Boolean function, h is any function from $\mathbb{Z}_2^n$ to $\mathbb{Z}_8$, and $w^4 = -1$. If $h(\boldsymbol{x})$ is an affine function, it can be shown that the coefficients of all its linear terms must be divisible by two. We can then eliminate $h(\boldsymbol{x})$ by post-multiplication with a transformation F of the form given by (6.33), and get $F\boldsymbol{S} = (-1)^{f'(\boldsymbol{x})}$.

Definition 6.49. Given a Boolean function f, find the set of all Boolean flat $\{I, H, N\}^n$ transforms of f, and recover the corresponding Boolean functions. The set of all distinct Boolean functions recovered from the set of all $\{I, H, N\}^n$ transforms of f, including f itself, is called the $\{I, H, N\}^n$ *orbit* of f.

Corollary 6.50. *By Theorem 6.40, if f is quadratic then the $\{I, H, N\}^n$ orbit is the LC orbit.*

Theorem 6.51. *It can be shown [15] that if two Boolean functions are in the same $\{I, H, N\}^n$ orbit, then they will have the same APC distance.*

Example 6.52. Consider the Boolean function $f(\boldsymbol{x}) = x_0 x_1 + x_0 x_2$. The bipolar vector describing the corresponding quantum state, if we ignore the normalisation factor $2^{-\frac{3}{2}}$, is

$$\boldsymbol{s} = (-1)^{f(\boldsymbol{x})} = (1, 1, 1, -1, 1, -1, 1, 1)^T.$$

We apply the transformation $U = N \otimes I \otimes I$ and get the result

$$\boldsymbol{S} = U\boldsymbol{s} = (w, w^7, w^7, w, w^7, w, w, w^7)^T,$$

where $\mathrm{w}^4 = -1$. We observe that $|S_k| = 1$ for all k, which means that $\boldsymbol{S}$ is flat and can be expressed as $w^{g(x)}$, where the function $g : \mathbb{Z}_2^3 \to \mathbb{Z}_8$ has truth table $(1, 7, 7, 1, 7, 1, 1, 7)^T$. Using ANFT$_8$, as described by (6.36), we find that the ANF of g is $(1, 6, 6, 4, 6, 4, 4, 0)^T$, and we can therefore write

$$\boldsymbol{S} = w^{4(x_0x_1+x_0x_2+x_1x_2)+(6x_0+6x_1+6x_2+1)}.$$

We observe that $\boldsymbol{S}$ is Boolean flat, since the terms that are not divisible by 4 are all linear or constant. We also see that all linear terms are divisible by 2, which, by Proposition 6.44, we should expect. We can eliminate all linear and constant terms by applying a transformation as described by (6.33), in this case we must use the transformation

$$F = w^7 \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}.$$

We get the result

$$F\boldsymbol{S} = (-1)^{x_0x_1+x_0x_2+x_1x_2}.$$

We can conclude that $f(\boldsymbol{x}) = x_0x_1 + x_0x_2$ and $f'(\boldsymbol{x}) = x_0x_1 + x_0x_2 + x_1x_2$ are in the same $\{I, H, N\}^n$ orbit, and since they are quadratic functions, the same LC orbit. This can be verified by applying the LC operation to the vertex corresponding to the variable x_0 in the graph representation of either function.

6.5 Orbits of Boolean Functions

For quadratic Boolean functions, LC operations on the associated graphs generate the orbits of equivalent functions. It is unknown whether there exists a similar operation on hypergraphs that generates the orbits of equivalent non-quadratic functions, but the $\{I, H, N\}^n$ orbit can be generated for Boolean functions of any degree. For non-quadratic functions, there are also other symmetries that must be considered.

We have already seen that permuting the variables of a Boolean function gives an equivalent function,

$$f(x_0, x_1, \ldots, x_{n-1}) \simeq f(x_{\pi(0)}, x_{\pi(1)}, \ldots, x_{\pi(n-1)}). \tag{6.38}$$

We can determine whether two functions are equivalent via the *permutation symmetry* by checking if their corresponding graphs are isomorphic. For non-quadratic functions we must check for hypergraph isomorphism. The package `nauty` can be used in both cases, as described in section 3.2.

Adding an affine offset to a Boolean function, i.e., adding any linear or constant terms, will not change the APC distance of the associated quantum state. We call this the *affine symmetry*,

$$f(\boldsymbol{x}) \simeq f(\boldsymbol{x}) + \boldsymbol{b} \cdot \boldsymbol{x} + c, \quad \boldsymbol{b} \in \mathbb{Z}_2^n, c \in \mathbb{Z}_2. \tag{6.39}$$

Note that the addition of linear terms corresponds to the phase-flip operation defined in (6.26). Similarly, all bit-flips on a Boolean function will produce an equivalent function. This is the *bit-flip symmetry*,

$$f(\boldsymbol{x}) \simeq f(\boldsymbol{x} + \boldsymbol{a}), \quad \boldsymbol{a} \in \mathbb{Z}_2^n. \tag{6.40}$$

Table 6.1: *Number of Orbits of Boolean Functions of* n *Variables*

n	Degrees	$\|\mathcal{O}_{1,n}\|$	$\|\mathcal{O}_{2,n}\|$
3	All	3	2
4	All	33	29
5	All	22,400	22,014
6	≤ 3	850,705	746,326

The affine symmetry may be taken into consideration by simply deleting any linear or constant terms from a Boolean function. We have implicitly done this for quadratic functions by only considering simple graphs.

Remark. A bit-flip on a Boolean function of degree m can change the coefficients of terms with degree at most $m - 1$.

Bit-flips on quadratic functions can only produce linear terms, which is also what phase-flips do. Therefore the affine symmetry and bit-flip symmetry coincide for quadratic functions. For non-quadratic functions, bit-flips can produce nonlinear terms, and the bit-flip symmetry may therefore generate non-trivial orbits of functions.

Definition 6.53. Let $\mathcal{B}_n$ be the set of all non-isomorphic connected Boolean functions of n variables with no linear or constant terms. (Two Boolean functions are isomorphic or connected if their corresponding hypergraphs are isomorphic or connected.) Given a Boolean function $f \in \mathcal{B}_n$, then the set of all distinct non-isomorphic Boolean functions recovered from the set of $\{I, \sigma_x\}^n$ transforms of f, with linear and constant terms removed, is called the $\{I, \sigma_x\}^n$ orbit of f. $\mathcal{B}_n$ can be partitioned into a set of disjoint $\{I, \sigma_x\}^n$ orbits, called $\mathcal{O}_{1,n}$. $|\mathcal{O}_{1,n}|$ is then the number of inequivalent connected Boolean functions of n variables with respect to the permutation, affine, and bit-flip symmetries.

We have generated all orbits in $\mathcal{O}_{1,n}$ for $n \leq 5$. For 6 variables, we could only generate orbits for all functions of degree up to 3. For these values of n, Table 6.1 gives the number of orbits, while Table 6.2 and Table 6.4 show the number of orbits by APC distance and degree for $n = 5$ and $n = 6$. A database containing one representative of each orbit is also available [13].

Orbits of non-quadratic functions with respect to $\{I, H, N\}^n$ can not be generated without also considering bit-flips, since it follows from the identities in (6.34) that a sequence of transformations from the set $\{I, H, N\}$ may induce bit-flips. For instance, applying first H and then N to one variable is equivalent to performing a bit-flip on the variable. The set M of Boolean functions found within the set of $\{I, H, N\}^n$ transforms of the non-quadratic Boolean function f will therefore contain some members of the $\{I, \sigma_x\}^n$ orbit of f. If f' is found in M, then $\{I, H, N\}^n$ transforms of f' may induce bit-flips that produce functions not found in M.

Definition 6.54. Given a non-quadratic Boolean function f, it follows from Theorem 6.46 that a closed orbit of functions may be generated by first finding the set M of all distinct Boolean functions recovered from $\{I, H, N\}^n$ transforms of f. We then generate M', containing all functions in the $\{I, \sigma_x\}^n$ orbits of every function in M. The set of non-isomorphic functions in M', with linear and constant terms removed, is the $\{I, \sigma_x\}^n\{I, H, N\}^n$ orbit of f. $\mathcal{B}_n$ can be partitioned into a set of disjoint $\{I, \sigma_x\}^n\{I, H, N\}^n$ orbits, called $\mathcal{O}_{2,n}$. $|\mathcal{O}_{2,n}|$ is then the number of inequivalent connected Boolean functions of n variables with respect to the permutation, affine, bit-flip, and $\{I, H, N\}$ symmetries.

Algorithm 6.3 can be used to generate all members of the $\{I, \sigma_x\}^n\{I, H, N\}^n$ orbit of the Boolean function f. It first finds all inequivalent Boolean functions corresponding to $\{I, H, N\}^n$ transforms of f, and then generates the $\{I, \sigma_x\}^n$ orbits of all these functions. When we only want to generate the $\{I, \sigma_x\}^n$ orbit of a function, we use the same algorithm but skip the first loop. When we want to count the orbits in $\mathcal{O}_{1,n}$ or $\mathcal{O}_{2,n}$, or find a member of each orbit, we use an approach similar to the one used in Algorithm 5.2. By picking a canonical representative of each orbit, we canonise all Boolean functions in $\mathcal{B}_n$ and remove all duplicates. This can be done much faster by using a generalisation of Definition 5.9, i.e., by generating a set of functions of n variables by extending one member from each orbit of functions of $n-1$ variables, and then using this set of functions as input to the above-mentioned canonisation algorithm. A function is extended by adding a variable and all combinations of monomials including this variable.

We use Algorithm 6.1 to find $\{I, H, N\}^n$ transforms or $\{I, \sigma_x\}^n$ transforms of a Boolean function. Note that we can skip iterations of Algorithm 6.1 where the identity transformation is applied. It is possible to order any set of transformations, $\mathcal{T}^n$, such that only one factor of each n-fold tensor product changes from one transformation to the next. By using some additional temporary transformation matrices, it is then possible to find all $\mathcal{T}^n$ transforms of a function by only using $|\mathcal{T}|^n$ single iterations of Algorithm 6.1. A *Gray code sequence* gives an ordering of transformations that makes this possible.

Definition 6.55. Given a sequence $\boldsymbol{s}$ of length n, let $\boldsymbol{s}' = \operatorname{rev}(\boldsymbol{s})$ be the sequence $\boldsymbol{s}$ in reversed order, i.e., $s'_i = s_{n-1-i}$. We also define $\boldsymbol{s}' = \operatorname{pre}(c, \boldsymbol{s})$, where $c \in \mathbb{Z}_n$. If c is even, then $\boldsymbol{s}'$ consists of all elements from $\boldsymbol{s}$ with the prefix c added, e.g., $\operatorname{pre}(0, (0, 1, 2)) = (00, 01, 02)$. If c is odd, then $\boldsymbol{s}'$ consists of all elements from $\operatorname{rev}(\boldsymbol{s})$ with the prefix c added, e.g., $\operatorname{pre}(1, (0, 1, 2)) = (12, 11, 10)$. Let $\operatorname{conc}(\{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_m\})$ be the concatenation of all sequences $\boldsymbol{s}_i$ to a single sequence.

Definition 6.56. The n-ary *Gray code sequence* of order r, denoted $\boldsymbol{q}_{n,r}$, contains all n^r elements from $\mathbb{Z}_n^r$. The sequence has the property that any two consecutive elements have the same symbol in $r-1$ positions, so that only one symbol needs to be changed to jump from one element of $\boldsymbol{q}_{n,r}$ to the next. A Gray code sequence can be constructed recursively,

$$\boldsymbol{q}_{n,1} = (0, 1, \ldots, n-1) \tag{6.41}$$

$$\boldsymbol{q}_{n,r} = \operatorname{conc}\left(\{\operatorname{pre}(c, \boldsymbol{q}_{n,r-1}) \mid c \in \mathbb{Z}_n\}\right). \tag{6.42}$$

Example 6.57. We want to find the 9 $\{I, H, N\}^2$ transforms of some Boolean function f of 2 variables. We generate the Gray code sequence $\boldsymbol{q}_{3,2}$ = (00, 01, 02, 12, 11, 10, 20, 21, 22). By the mapping $0 \mapsto I$, $1 \mapsto H$, and $2 \mapsto N$, we get the following sequence of transformations, ($I \otimes I$, $I \otimes H$, $I \otimes N$, $H \otimes N$, $H \otimes H$, $H \otimes I$, $N \otimes I$, $N \otimes H$, $N \otimes N$). To jump from H to N, N to H, and N to I, we need the temporary transform matrices NH^{-1}, HN^{-1}, and N^{-1}. (Note that $H^{-1} = H$.) The sequence of transformations we will use is then ($I \otimes I$, $I \otimes H$, $I \otimes NH^{-1}$, $H \otimes I$, $I \otimes H$, $I \otimes H$, $N \otimes I$, $I \otimes H$, $I \otimes NH^{-1}$), where only the first transformation is applied to the function f, and the subsequent transformations are applied to the transform from the previous step, so that the cumulative effect gives the desired result.

Using Algorithm 6.3, we have generated all orbits in $\mathcal{O}_{2,n}$ for $n \leq 5$, and also the orbits of functions of 6 variables with degree up to 3. For these values

Algorithm 6.3 Generating an $\{I, \sigma_x\}^n\{I, H, N\}^n$ Orbit

```
Input    $f$: a Boolean function
         $n$: the number of variables of $f$
Output   $\boldsymbol{L}$: the set of all Boolean functions in the orbit of $f$

procedure GenerateOrbit($f$, $n$)
    $\boldsymbol{s} \leftarrow$ bipolar truth table of $f$
    for all transformations $U \in \{I, H, N\}^n$ do        ▷ use Gray code ordering
        $\boldsymbol{s}' \leftarrow U\boldsymbol{s}$                        ▷ use single iteration of Algorithm 6.1
        $\boldsymbol{s}'' \leftarrow \mathbb{Z}_8$-ANF corresponding to $\boldsymbol{s}'$                   ▷ use ANFT$_8$
        if $\boldsymbol{s}''$ is Boolean flat then
            $f' \leftarrow$ Boolean function corresponding to $\boldsymbol{s}''$
            Remove all linear and constant terms from $f'$
            $f'' \leftarrow$ NautyCanonise($f'$)          ▷ use hypergraph canonisation
            if $f'' \notin \boldsymbol{L}$ then
                Add($\boldsymbol{L}$, $f''$)
            end if
        end if
    end for
    for all functions $g \in \boldsymbol{L}$ do
        $\boldsymbol{s} \leftarrow$ bipolar truth table of $g$
        for all transformations $U \in \{I, \sigma_x\}^n$ do     ▷ use Gray code ordering
            $\boldsymbol{s}' \leftarrow U\boldsymbol{s}$                    ▷ use single iteration of Algorithm 6.1
            $g' \leftarrow$ ANF corresponding to $\boldsymbol{s}'$
            Remove all linear and constant terms from $g'$
            $g'' \leftarrow$ NautyCanonise($g'$)          ▷ use hypergraph canonisation
            if $g'' \notin \boldsymbol{L}$ then
                Add($\boldsymbol{L}$, $g''$)
            end if
        end for
    end for
    return $\boldsymbol{L}$
end procedure
```

of n, Table 6.1 gives the number of orbits, while Table 6.3 and Table 6.5 show the number of orbits by APC distance and degree. A database containing one representative of each orbit is also available [13]. The numbers of inequivalent functions are not reduced considerably when the $\{I, H, N\}$ symmetry is considered in addition to bit-flips, but the difference in orbit size is larger for functions with low degree and high APC distance. Recall that for quadratic functions, bit-flips only generate affine offsets. For functions of high degree, bit-flips account for almost all symmetries.

Remark. There are no non-quadratic functions of 5 or less variables with APC distance higher than 2, but there are 11 functions of 6 variables, belonging to different orbits in $\mathcal{O}_{2,6}$, with degree 3 and APC distance 3.

A representative of each of the 11 inequivalent non-quadratic $[[6, 0, 3]]$ quantum codes is listed in Table 6.6. The first function in this table corresponds to the hypergraph shown in Figure 6.2. The table also gives examples of codes of higher length found by non-exhaustive searches of functions with a limited number of non-quadratic terms, or by adding several higher degree terms to strong quadratic codes. The functions in Table 6.6 all belong to different $\{I, \sigma_x\}^n\{I, H, N\}^n$ orbits, and the representative from each orbit with fewest monomials is listed. The table also gives the value of PAR_{IHN} of each function. PAR_{IHN} will be introduced in the next chapter.

Table 6.2: *Number of Orbits in $\mathcal{O}_{1,5}$ with APC Distance d and Degree δ*

	δ				
d	2	3	4	5	All
1		625	10,756	10,688	22,069
2	18	109	201		328
3	3				3
All	21	734	10,957	10,688	22,400

Table 6.3: *Number of Orbits in $\mathcal{O}_{2,5}$ with APC Distance d and Degree δ*

	δ				
d	2	3	4	5	All
1		505	10,570	10,688	21,763
2	3	61	186		250
3	1				1
All	4	566	10,756	10,688	22,014

Table 6.4: *Number of Orbits in $\mathcal{O}_{1,6}$ with APC Distance d and Degree δ*

	δ					
d	2	3	4	5	6	2 and 3
1		804,326	?	?	?	804,326
2	94	46,243	?	?	?	46,337
3	16	24	?	?	?	40
4	2		?	?	?	2
All	112	850,593	?	?	?	850,705

Table 6.5: *Number of Orbits in $\mathcal{O}_{2,6}$ with APC Distance d and Degree δ*

	δ					
d	2	3	4	5	6	2 and 3
1		717,741	?	?	?	717,741
2	9	28,563	?	?	?	28,572
3	1	11	?	?	?	12
4	1		?	?	?	1
All	11	746,315	?	?	?	746,326

Table 6.6: *Boolean Functions of n Variables with Degree δ, APC Distance d, and PAR_{IHN} p*

n	δ	d	p	Function
6	3	3	8	$012, 03, 04, 13, 15, 24, 25$
6	3	3	4.5	$012, 03, 05, 14, 15, 23, 24, 25, 34$
6	3	3	4.5	$023, 012, 04, 05, 13, 15, 23, 24, 25, 34$
6	3	3	8	$123, 124, 125, 01, 02, 14, 25, 34, 35, 45$
6	3	3	4.5	$012, 013, 03, 04, 13, 15, 24, 25, 34, 35, 45$
6	3	3	4.5	$012, 013, 014, 03, 05, 14, 15, 23, 24, 25, 34$
6	3	3	4.5	$012, 014, 024, 123, 134, 234, 03, 13, 15, 24, 25, 34, 45$
6	3	3	8	$015, 012, 013, 014, 03, 05, 14, 15, 23, 24, 25, 34, 35, 45$
6	3	3	8	$025, 245, 012, 124, 023, 234, 04, 05, 13, 15, 23, 24, 35, 45$
6	3	3	4.5	$245, 235, 145, 135, 024, 023, 014, 013, 02, 05, 14, 15, 23, 34, 35, 45$
6	3	3	8	$125, 145, 135, 245, 235, 012, 014, 013, 024, 023, 05, 13, 15, 24, 25, 34$
7	3	3	8	$012, 03, 05, 14, 16, 25, 26, 34$
7	3	3	8	$014, 02, 05, 13, 15, 26, 36, 45, 46$
7	3	3	8	$014, 02, 05, 13, 16, 26, 35, 45, 46$
7	3	3	8	$015, 02, 06, 13, 16, 25, 35, 45, 46$
7	3	3	8	$012, 03, 06, 14, 16, 25, 26, 34, 35, 45$
7	3	3	9	$012, 04, 05, 14, 16, 25, 26, 34, 35, 36$
7	3	3	8	$013, 04, 06, 15, 16, 23, 26, 34, 35, 45$
7	3	3	9	$013, 04, 06, 15, 16, 24, 25, 34, 35, 36$
7	3	3	16	$456, 04, 05, 14, 16, 25, 26, 34, 35, 36$
7	3	3	8	$024, 045, 01, 03, 12, 26, 35, 45, 46, 56$
7	3	3	8	$012, 03, 06, 14, 15, 24, 25, 26, 35, 36, 46$
7	3	3	9	$012, 05, 06, 13, 15, 23, 26, 34, 45, 46, 56$
7	3	3	8	$012, 05, 06, 13, 15, 24, 26, 34, 35, 46, 56$
7	3	3	8	$016, 02, 03, 14, 15, 24, 26, 35, 36, 46, 56$
7	3	3	8	$012, 03, 05, 14, 15, 23, 24, 26, 36, 46, 56$
7	3	3	8	$035, 025, 023, 06, 13, 15, 16, 24, 25, 26, 34, 36, 45$
7	3	3	8	$013, 023, 123, 012, 04, 05, 14, 16, 25, 26, 34, 35, 36$
7	4	3	12.25	$0123, 04, 05, 14, 16, 25, 26, 34, 35, 36$
7	4	3	8	$0126, 03, 05, 14, 15, 23, 24, 26, 36, 46, 56$
8	3	4	16	$012, 04, 05, 07, 14, 16, 17, 25, 26, 27, 34, 35, 36$
8	3	4	16	$016, 02, 03, 04, 12, 13, 15, 27, 36, 46, 47, 56, 57, 67$
8	3	4	8	$067, 01, 02, 03, 14, 16, 25, 27, 36, 37, 46, 47, 56, 57$
8	3	4	9	$234, 02, 03, 05, 12, 13, 14, 26, 37, 46, 47, 56, 57, 67$
8	3	4	8	$024, 046, 01, 03, 05, 12, 14, 25, 27, 34, 36, 46, 47, 56, 57, 67$
8	3	4	8	$127, 126, 067, 01, 02, 03, 14, 16, 25, 27, 36, 37, 46, 47, 56, 57$
8	3	4	8	$456, 056, 246, 026, 245, 025, 04, 07, 15, 16, 17, 23, 26, 27, 35, 36, 45, 46, 47, 57$
8	4	4	16	$0123, 04, 05, 06, 14, 15, 17, 24, 26, 27, 35, 36, 37$
8	4	4	16	$0167, 02, 03, 04, 12, 13, 15, 26, 37, 46, 47, 56, 57, 67$
9	3	4	8	$016, 02, 04, 07, 13, 15, 18, 23, 26, 36, 45, 47, 58, 67, 68, 78$
9	3	4	9	$235, 234, 02, 04, 06, 13, 15, 16, 26, 28, 36, 37, 47, 48, 57, 58, 78$
9	3	4	8	$037, 137, 034, 134, 01, 07, 08, 14, 18, 23, 25, 28, 36, 37, 45, 46, 57, 58, 67, 68$
9	3	4	16	$024, 124, 047, 147, 02, 05, 08, 15, 17, 18, 23, 26, 34, 37, 38, 46, 47, 48, 56, 58, 67$
9	3	4	16	$234, 023, 124, 012, 03, 06, 08, 14, 17, 18, 23, 24, 25, 37, 38, 46, 48, 56, 57, 58, 67$
10	3	4	16	$016, 02, 03, 08, 12, 13, 17, 26, 39, 45, 46, 48, 57, 59, 67, 68, 79, 89$

Chapter 7

Peak-to-Average Power Ratio

7.1 Peaks and Independent Sets

Definition 7.1. The *peak-to-average power ratio* of the vector $\boldsymbol{s}$ with respect to the $\mathcal{T}^n$ transform, for some transform set $\mathcal{T} = \{T_1, T_2, \ldots, T_m\}$, is defined as

$$\mathrm{PAR}_{\mathcal{T}}(\boldsymbol{s}) = 2^n \max_{\substack{U \in \mathcal{T}^n \\ k \in \mathbb{Z}_{2^n}}} |S_k|^2, \tag{7.1}$$

where $\boldsymbol{S} = U\boldsymbol{s}$. In other words, the $\mathrm{PAR}_{\mathcal{T}}$ of $\boldsymbol{s}$ is the highest squared magnitude of the $(2m)^n$ coefficients in the $\mathcal{T}^n$ spectrum of $\boldsymbol{s}$.

The $\mathrm{PAR}_{\mathcal{T}}$ of $\boldsymbol{s}$ can alternatively be expressed in terms of the *generalised nonlinearity* [37],

$$\gamma(f) = 2^{\frac{n}{2}-1}\left(2^{\frac{n}{2}} - \sqrt{\mathrm{PAR}_{\mathcal{T}}\left(\boldsymbol{s}\right)}\right), \tag{7.2}$$

but we will use the $\mathrm{PAR}_{\mathcal{T}}$ measure. We will in particular study PAR_{IHN}, the peak-to-average power ratio with respect to the $\{I, H, N\}^n$ transform [16]. If a vector, $\boldsymbol{s}$, has a completely flat $\{I, H, N\}^n$ spectrum, which is impossible [51], then $\mathrm{PAR}_{IHN}(\boldsymbol{s}) = 1$. If $\boldsymbol{s} = 2^{-\frac{n}{2}}(1, 1, \ldots, 1, 1)^T$ then $\mathrm{PAR}_{IHN}(\boldsymbol{s}) = 2^n$. A typical vector, $\boldsymbol{s}$, will have a $\mathrm{PAR}_{IHN}(\boldsymbol{s})$ somewhere between these extremes. For quadratic functions, PAR_{IHN} will always be a power of 2 [52].

Let $\boldsymbol{s} = 2^{-\frac{n}{2}}(-1)^{f(\boldsymbol{x})}$, as before. When we talk about the PAR_{IHN} of f, or its associated graph G, we mean $\mathrm{PAR}_{IHN}(\boldsymbol{s})$. For cryptographic purposes, it is desirable to find Boolean functions with high generalised nonlinearity and therefore low PAR_{IHN}. PAR_{IHN} is an invariant of the $\{I, \sigma_x\}^n\{I, H, N\}^n$ orbit and, for quadratic functions, the LC orbit. We observe that Boolean functions from LC orbits associated with self-dual additive codes over GF(4) with high distance typically have low PAR_{IHN}. This is not surprising, since the distance of a self-dual quantum code is equal to the APC distance of the associated quadratic Boolean function, and APC is derived from the fixed-aperiodic autocorrelation which is, in turn, the autocorrelation "dual" of the spectra with respect to $\{I, H, N\}^n$. Table 7.1 shows the value of PAR_{IHN} for every LC orbit of codes with length $n \leq 12$.

Definition 7.2. Let $\alpha(G)$ be the independence number of a graph G, i.e., the size of the maximum independent set in G. Let $[G]$ be the set of all graphs in the LC orbit of G. We then define

$$\lambda(G) = \max_{K \in [G]} \alpha(K), \tag{7.3}$$

Table 7.1: *Number of LC Orbits with Length n and PAR_{IHN} p*

	n											
p	1	2	3	4	5	6	7	8	9	10	11	12
2	1	1										
4			1	1	1	1						
8				1	2	5	6	9	2	1		
16					1	4	14	52	156	624	3,184	12,323
32						1	5	32	212	1,753	25,018	834,256
64							1	7	60	639	10,500	380,722
128								1	9	103	1,578	43,013
256									1	11	163	3,488
512										1	13	249
1024											1	16
2048												1

Table 7.2: *Range of λ for Codes of Length n and Distance d*

	n										
d	2	3	4	5	6	7	8	9	10	11	12
2	**1**	**2**	**2**,3	3,4	3–5	3–6	3–7	4–8	4–9	4–10	4–11
3				**2**	3	**3**,4	3,4	3–5	4–6	4–7	4–8
4					**2**		**3**,4	**3**,4	**3**–5	4–6	4–7
5										**4**	4
6											**4**

i.e., λ is the size of the maximum independent set over all graphs in the LC orbit of G.

Consider as an example the Hexacode which has two non-isomorphic graphs in its orbit, as seen in Figure 3.2 on page 22. It is evident that the independence number of each graph is 2, so $\lambda = 2$. The values of λ for all LC orbits for $n \leq 12$ clearly show that λ and d, the distance of the associated self-dual additive code over GF(4), are related. LC orbits associated with codes of high distance typically have small values for λ. Table 7.2 summarises this observation by listing the ranges of λ observed for all LC orbits associated with codes of given lengths and distances. For instance, $[[12, 0, 2]]$ codes exist with any value of λ between 4 and 11, while $[[12, 0, 5]]$ and $[[12, 0, 6]]$ codes only exist with $\lambda = 4$.

Definition 7.3. Let Λ_n be the minimum value of λ over all LC orbits of graphs on n vertices, i.e,

$$\Lambda_n = \min_{[G]\in\mathcal{L}_n} \lambda(G). \tag{7.4}$$

From Table 7.2 we observe that $\Lambda_n = 2$ for n from 3 to 6, $\Lambda_n = 3$ for n from 7 to 10, and $\Lambda_n = 4$ when n is 11 or 12.

Proposition 7.4. *$\Lambda_{n+1} \geq \Lambda_n$, i.e., Λ_n is monotonically nondecreasing when n is increasing.*

Proof. Consider a graph $G = (V, E)$ with $n + 1$ vertices. Select a vertex v and let G' be the induced subgraph on the n vertices $V\backslash\{v\}$. We generate the LC orbit of G'. The LC operations may add or remove edges between G' and v,

Table 7.3: *Values of Λ_n for $n \leq 14$ and Bounds on Λ_n for $n \leq 21$*

n	Λ_n
2	1
3	2
4	2
5	2
6	2
7	3
8	3
9	3
10	3
11	4
12	4
13	4
14	4
15	4–5
16	4–5
17	4–5
18	4–6
19	4–6
20	4–6
21	4–9

but the presence of v does not affect the LC orbit of G'. The size of the largest independent set in the LC orbit of G' is at least Λ_n. This is also an independent set in the LC orbit of G, so $\Lambda_{n+1} \geq \Lambda_n$. □

Definition 7.5. There is a number $r = R(m, n)$, called a *Ramsey number* [45], such that it is guaranteed that all simple undirected graphs on at least r vertices will have either an independent set of size m or a clique of size n.

Proposition 7.6. *If r is the Ramsey number $R(k, k+1)$, then $\Lambda_n \geq k$ for $n \geq r$.*

Proof. Consider a graph containing a clique of size m. An LC operation on any vertex in the clique will produce an independent set of size $m-1$. Thus the maximum clique in an LC orbit, where the largest independent set has size λ, can not be larger than $\lambda + 1$. Since $r = R(k, k+1)$, it follows that all graphs on at least r vertices must have $\lambda \geq k$, and therefore that $\Lambda_n \geq k$ for $n \geq r$. □

For instance, the Ramsey number $R(3, 4)$ is 9, so by Proposition 7.6, $\Lambda_n \geq 3$ for $n \geq 9$, which means that no LC orbit with at least 9 vertices can have λ smaller than 3. Similarly, $R(4, 5) = 25$, so $\Lambda_n \geq 4$ for $n \geq 25$. For n from 13 to 21, we have computed the values of λ for some graphs corresponding to self-dual additive codes over GF(4) with high distance. This gives upper bounds on the value of Λ_n, as shown in Table 7.3. The bounds on Λ_{13} and Λ_{14} are tight, since $\Lambda_{12} = 4$ and $\Lambda_{n+1} \geq \Lambda_n$. The bounds on Λ_n given by Proposition 7.6 are very loose, since we can see from Table 7.3 that $\Lambda_n \geq 3$ for $n \geq 7$ and that $\Lambda_n \geq 4$ for $n \geq 11$. The connection to Ramsey theory is still interesting, and it may be possible to improve the bound.

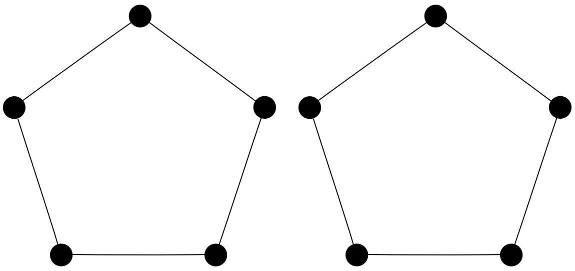


Figure 7.1: *The "Double 5-Cycle" Graph*

Remark. For $n = 10$, there is a unique LC orbit that satisfies, optimally, $\lambda = 3$, $\mathrm{PAR}_{IHN} = 8$ and $d = 4$. One of the graphs in this orbit is the *graph complement* of the "double 5-cycle" graph, shown in Figure 7.1.

Proposition 7.7 (Parker and Rijmen [40]). *Given a graph $G = (V, E)$ with a maximum independent set $A \subset V$, $|A| = \alpha(G)$. Let $\boldsymbol{s} = (-1)^{f(\boldsymbol{x})}$, where f is the Boolean function representation of G. Let $U = \bigotimes_{k\in A} H^{(k)} \bigotimes_{k\notin A} I^{(k)}$, i.e., the transformation applying H to variables corresponding to vertices $k \in A$ and I to all other variables. Let $\boldsymbol{S} = U\boldsymbol{s}$. Then*

$$\max_{m\in\mathbb{Z}_{2^n}} |S_m|^2 = 2^{\alpha(G)}. \tag{7.5}$$

Arratia et al. [2, 3] introduced the *interlace polynomial* $q(G, z)$ of a graph G. Aigner and van der Holst [1] later introduced the interlace polynomial $Q(G, z)$. Riera and Parker [52] showed that $q(G, z)$ is related to the $\{I, H\}^n$ spectra of the quadratic Boolean function corresponding to G, and that $Q(G, z)$ is related to the $\{I, H, N\}^n$ spectra.

Proposition 7.8 (Aigner and van der Holst [1]). *The interlace polynomial $Q(G, z)$ can be defined recursively as*

$$Q(G, z) = Q(G\backslash u, z) + P(G^u\backslash u, z) + P(((G^u)^v)^u\backslash u, z), \tag{7.6}$$

where G^u denotes the LC operation on vertex u of G, and $G\backslash u$ is the graph obtained by removing vertex u and all edges incident on u from G.

Proposition 7.9 (Riera and Parker [52]). *Let f be a quadratic Boolean function and G its associated graph. Then PAR_{IHN} of f is equal to $2^{\deg Q(G,z)}$, where $\deg Q(G, z)$ is the degree of the interlace polynomial $Q(G, z)$.*

Aigner and van der Holst [1] proved that the degree of $q(G, z)$ is equal to the size of the maximum independent set in the *switch-class* of G, which is the same as the $\{I, H\}^n$ orbit of G. This proof can be extended to show that the degree of $Q(G, z)$ equals λ, the size of the maximum independent set in the $\{I, H, N\}^n$ orbit of G.

Theorem 7.10. *If the maximum independent set over all graphs in the LC orbit $[G]$ has size $\lambda(G)$, then all functions corresponding to graphs in the orbit will have $PAR_{IHN} = 2^{\lambda(G)}$.*

Proof. Let us for brevity define $P(G) = \mathrm{PAR}_{IHN}(\boldsymbol{s})$, where $\boldsymbol{s} = 2^{-\frac{n}{2}}(-1)^{f(\boldsymbol{x})}$, and $f(\boldsymbol{x})$ is the Boolean function representation of G. From Proposition 7.7 it follows that $P(G) \geq 2^{\lambda(G)}$. Choose $K = (V, E) \in [G]$ with $\alpha(K) = \lambda(G)$.

If $|V| = 1$ or 2, the theorem is true. We will prove the theorem for $n > 2$ by induction on $|V|$. We will show that $P(K) \leq 2^{\alpha(K)}$, which is equivalent to saying that $P(G) \leq 2^{\lambda(G)}$. It follows from Proposition 7.8 and Proposition 7.9 that $P(K) = \max\{P(K\backslash u), P(K^u\backslash u), P(((K^u)^v)^u\backslash u)\}$. Assume, by induction hypothesis, that $P(K\backslash u) = 2^{\lambda(K\backslash u)}$. Therefore, $P(K\backslash u) = 2^{\alpha(K\backslash u)}$ for some $K\backslash u \in [K\backslash u]$. Note that $K\backslash u \in [K\backslash u]$ implies $K \in [K]$. It must then be true that $\alpha(K\backslash u) \leq \alpha(K) \leq \alpha(K)$, and it follows that $P(K\backslash u) \leq 2^{\alpha(K)}$. Similar arguments hold for $P(K^u\backslash u)$ and $P(((K^u)^v)^u\backslash u)$, so $P(K) \leq 2^{\alpha(K)}$.

□

As an example, the Hexacode has $\lambda = 2$ and therefore $\text{PAR}_{IHN} = 2^2 = 4$. The vector containing the highest peak can be found by taking either of the two graphs in the LC orbit, since both have independence number λ, and then by applying the H transformation to all variables corresponding to vertices in an independent set of size λ and the I transformation to all other variables.

Corollary 7.11. *Any quadratic Boolean function on n or more variables must have $PAR_{IHN} \geq 2^{\Lambda_n}$.*

Definition 7.12. $\text{PAR}_{\mathcal{U}}$ is the peak-to-average power ratio with respect to the infinite transform set $\mathcal{U}^n$, where $\mathcal{U}$ consists of matrices of the form

$$U = \begin{pmatrix} \cos\theta & \sin\theta e^{i\phi} \\ \sin\theta & -\cos\theta e^{i\phi} \end{pmatrix},$$

where $i^2 = -1$, and θ and ϕ can take any real values. $\mathcal{U}$ comprises all 2×2 unitary transforms to within a post-multiplication by a matrix from $\mathcal{D}$, the set of 2×2 diagonal and anti-diagonal unitary matrices. (Note that Parker and Rijmen [40] refer to $\text{PAR}_{\mathcal{U}}$ as PAR_l.)

Theorem 7.13 (Parker and Rijmen [40])**.** *If $\boldsymbol{s}$ corresponds to a bipartite graph, then $PAR_{\mathcal{U}}(\boldsymbol{s}) = PAR_{IH}(\boldsymbol{s})$, where PAR_{IH} is the peak-to-average power ratio with respect to the transform set $\{I, H\}^n$.*

It is obvious that $\{I, H\}^n \subset \{I, H, N\}^n \subset \mathcal{U}^n$, and it follows that $\text{PAR}_{IH} \leq \text{PAR}_{IHN} \leq \text{PAR}_{\mathcal{U}}$. We then get the following corollary of Theorem 7.10 and Theorem 7.13.

Corollary 7.14. *If an LC orbit, $[G]$, contains a bipartite graph, then all functions corresponding to graphs in the orbit will have $PAR_{\mathcal{U}} = 2^{\lambda(G)}$.*

All LC orbits with a bipartite member have $\text{PAR}_{IHN} = \text{PAR}_{\mathcal{U}}$, but note that these orbits will always have $\text{PAR}_{\mathcal{U}} \geq 2^{\lceil \frac{n}{2} \rceil}$ [40], and that the fraction of LC orbits which have a bipartite member appears to decrease exponentially as the number of vertices increases. In the general case, PAR_{IHN} is only a lower bound on $\text{PAR}_{\mathcal{U}}$. For example, the Hexacode has $\text{PAR}_{IHN} = 4$, but a tighter lower bound on $\text{PAR}_{\mathcal{U}}$ is 4.486 [40]. (This bound has later been improved to 5.103 [39].)

7.2 Constructions for Low PAR

So far we have only considered the PAR_{IHN} of quadratic Boolean functions. For cryptographic purposes, we are interested in Boolean functions of degree higher than 2. As shown in section 6.3, such functions correspond to hypergraphs and non-quadratic quantum codes. Table 6.6 on page 70 gives the value of

Table 7.4: *Sampled Range of PAR$_{IHN}$ for Length (n) from 6 to 10*

n	Samples	Range of PAR$_{IHN}$
6	50,000	6.5–25.0
7	20,000	9.0–28.125
8	5,000	12.25–28.125
9	2,000	14.0625–30.25
10	1,000	18.0–34.03

PAR$_{IHN}$ for some non-quadratic Boolean functions with high APC distance. Many of these functions have the same PAR$_{IHN}$ as the best quadratic functions, but no non-quadratic function with lower PAR$_{IHN}$ than the best quadratic functions has yet been found. Exhaustive searching for non-quadratic Boolean functions with low PAR$_{IHN}$ becomes infeasible for more than a few variables. We therefore propose a construction technique using the best quadratic functions as building blocks [16]. Before we describe our construction we must first state what we mean by "low PAR$_{IHN}$". For $n = 6$ to $n = 10$ we computed PAR$_{IHN}$ for samples from the space $\mathbb{Z}_2^{2^n}$, to determine the range of PAR$_{IHN}$ we can expect just by guessing. Table 7.4 summarises these results. If we can construct Boolean functions with PAR$_{IHN}$ lower than the sampled minimum, we can consider our construction to be somewhat successful.

Parker and Tellambura [41, 42] proposed a generalisation of the Maiorana-McFarland construction for Boolean functions that satisfies a tight upper bound on PAR with respect to the $\{H, N\}^n$ transform (and other transform sets), this being a form of Golay Complementary Set construction and a generalisation of the construction of Rudin and Shapiro and of Davis and Jedwab [17].

Construction 7.15. Let $p(\boldsymbol{x})$ be a Boolean function on $n = \sum_{j=0}^{L-1} t_j$ variables, where $T = \{t_0, t_1, \ldots, t_{L-1}\}$ is a set of positive integers and $\boldsymbol{x} \in \mathbb{Z}_2^n$. Let $\boldsymbol{y_j} \in \mathbb{Z}_2^{t_j}$, $0 \le j < L$, such that $\boldsymbol{x} = \boldsymbol{y_0} \times \boldsymbol{y_1} \times \cdots \times \boldsymbol{y_{L-1}}$. Construct $p(\boldsymbol{x})$ as follows.

$$p(\boldsymbol{x}) = \sum_{j=0}^{L-2} \theta_j(\boldsymbol{y_j})\gamma_j(\boldsymbol{y_{j+1}}) + \sum_{j=0}^{L-1} g_j(\boldsymbol{y_j}), \tag{7.7}$$

where θ_j is a permutation: $\mathbb{Z}_2^{t_j} \to \mathbb{Z}_2^{t_{j+1}}$, γ_j is a permutation: $\mathbb{Z}_2^{t_{j+1}} \to \mathbb{Z}_2^{t_j}$, and g_j is any Boolean function of t_j variables. It has been shown [42] that the function $p(\boldsymbol{x})$ will have PAR$_{HN} \le 2^{t_{\max}}$, where $t_{\max}$ is the largest integer in T. It is helpful to visualise this construction graphically, as in Figure 7.2. In this example, the size of the largest partition is 3, so PAR$_{HN} \le 8$, regardless of what choices we make for θ_j, γ_j, and g_j.

Observe that if we set $L = 2$, $t = t_0 = t_1$, let θ_0 be the identity permutation, and $g_0 = 0$, Construction 7.15 reduces to the Maiorana-McFarland construction over $2t$ variables. Construction 7.15 can also be viewed as a generalisation of the path graph, $f(\boldsymbol{x}) = x_0x_1 + x_1x_2 + \cdots + x_{n-2}x_{n-1}$, which has optimal PAR with respect to $\{H, N\}^n$. Unfortunately, the path graph is not a particularly good construction for low PAR$_{IHN}$. But as we have seen, graphs corresponding to self-dual additive codes over GF(4) with high distance do give us Boolean functions with low PAR$_{IHN}$.

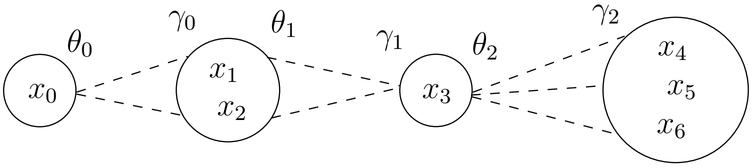


Figure 7.2: *Example of Construction for PAR$_{HN} \leq 8$*

Construction 7.16. We propose the generalised construction,

$$p(\boldsymbol{x}) = \sum_{i=0}^{L-1} \sum_{j=i+1}^{L-1} \Gamma_{i,j}(\boldsymbol{y_i})\Gamma_{j,i}(\boldsymbol{y_j}) + \sum_{j=0}^{L-1} g_j(\boldsymbol{y_j}), \tag{7.8}$$

where $\Gamma_{i,j}$ is either a permutation: $\mathbb{Z}_2^{t_i} \rightarrow \mathbb{Z}_2^{t_j}$, or $\Gamma_{i,j} = 0$, and g_j is any Boolean function on t_j variables.

It is evident that Γ can be thought of as a "generalised adjacency matrix", where the entries, $\Gamma_{i,j}$, are no longer 0 or 1 but, instead, 0 or permutations from $\mathbb{Z}_2^{t_i}$ to $\mathbb{Z}_2^{t_j}$. Construction 7.15 then becomes a special case where $\Gamma_{i,j} = 0$ except for when $j = i + 1$, i.e., a "generalised adjacency matrix" of the path graph. In order to minimise PAR$_{IHN}$ we choose the form of the matrix Γ according to the adjacency matrix of a self-dual additive code over GF(4) with high distance. We also choose the "offset" functions, g_j, to be Boolean functions corresponding to self-dual additive codes over GF(4) with high distance. Finally for the non-zero $\Gamma_{i,j}$ entries, we choose selected permutations, preferably nonlinear to increase the overall degree. Here are some initial results which demonstrate that, using Construction 7.16, we can construct Boolean functions of algebraic degree greater than 2 with low PAR$_{IHN}$.

Example 7.17 ($n = 8$)**.** Use the Hexacode graph f = 01, 02, 03, 04, 05, 12, 23, 34, 45, 51 as a template. Let $t_0 = 3$, $t_1 = t_2 = t_3 = t_4 = t_5 = 1$. (See Figure 7.3.) We use the following matrix Γ.

$$\Gamma = \begin{pmatrix} 0 & 02,1 & 02,1 & 02,1 & 02,1 & 02,1 \\ 3 & 0 & 3 & 0 & 0 & 3 \\ 4 & 4 & 0 & 4 & 0 & 0 \\ 5 & 0 & 5 & 0 & 5 & 0 \\ 6 & 0 & 0 & 6 & 0 & 6 \\ 7 & 7 & 0 & 0 & 7 & 0 \end{pmatrix}$$

Let $g_0(\boldsymbol{y_0})$ = 01, 02, 12 and all other g_j any arbitrary affine functions. Then, using Construction 7.16 to construct $p(\boldsymbol{x})$ we get $p(\boldsymbol{x})$ = 023, 024, 025, 026, 027, 01, 02, 12, 13, 14, 15, 16, 17, 34, 37, 45, 56, 67. It can be verified that $p(\boldsymbol{x})$ has PAR$_{IHN} = 9.0$.

Example 7.18 ($n = 8$)**.** Use the Hexacode graph f = 01, 02, 03, 04, 05, 12, 23, 34, 45, 51 as a template. Let $t_0 = 3$, $t_1 = t_2 = t_3 = t_4 = t_5 = 1$. (See Figure 7.3.) We use the following matrix Γ.

$$\Gamma = \begin{pmatrix} 0 & 02,1 & 12,0,1,2 & 01,02,12,1,2 & 01,02,12 & 02,12,1,2 \\ 3 & 0 & 3 & 0 & 0 & 3 \\ 4 & 4 & 0 & 4 & 0 & 0 \\ 5 & 0 & 5 & 0 & 5 & 0 \\ 6 & 0 & 0 & 6 & 0 & 6 \\ 7 & 7 & 0 & 0 & 7 & 0 \end{pmatrix}$$

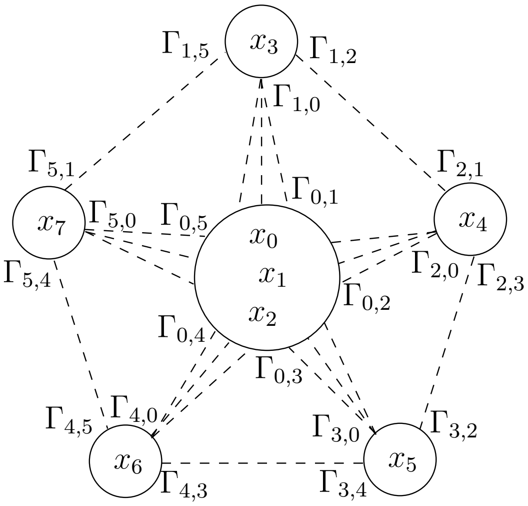


Figure 7.3: *Example of Construction for Low PAR$_{IHN}$*

Let $g_0(\boldsymbol{y_0}) = 01, 12$ and all other g_j any arbitrary affine functions. Then, using Construction 7.16 to construct $p(\boldsymbol{x})$ we get $p(\boldsymbol{x}) =$ 015, 016, 023, 025, 026, 027, 124, 125, 126, 127, 01, 04, 12, 13, 14, 15, 17, 24, 25, 27, 34, 37, 45, 56, 67, where $p(\boldsymbol{x})$ has PAR$_{IHN} = 9.0$.

Example 7.19 ($n = 9$)**.** Use the triangle graph $f = 01, 02, 12$ as a template. Let $t_0 = t_1 = t_2 = 3$. (See Figure 7.4.) Assign the permutations

$$\begin{aligned}
\Gamma_{0,1} = \Gamma_{0,2} &= (12, 0, 1, 2)(01, 2)(02, 1, 2),\\
\Gamma_{1,0} &= (34, 5)(35, 4, 5)(45, 3, 4, 5),\\
\Gamma_{1,2} &= (45, 3, 4, 5)(34, 5)(35, 4, 5),\\
\Gamma_{2,0} &= (68, 7, 8)(78, 6, 7, 8)(67, 8),\\
\Gamma_{2,1} &= (78, 6, 7, 8)(67, 8)(68, 7, 8).
\end{aligned}$$

Let $g_0(\boldsymbol{y_0}) = 01, 02, 12$, $g_1(\boldsymbol{y_1}) = 34, 35, 45$, and $g_2(\boldsymbol{y_2}) = 67, 68, 78$. Then, using Construction 7.16 to construct $p(\boldsymbol{x})$ we get, $p(\boldsymbol{x}) =$ 0135, 0178, 0245, 0267, 1234, 1268, 3467, 3568, 4578, 014, 015, 016, 017, 018, 023, 024, 025, 028, 034, 068, 125, 127, 128, 134, 145, 167, 168, 234, 235, 245, 267, 268, 278, 348, 357, 358, 378, 456, 457, 458, 468, 478, 567, 568, 578, 05, 07, 08, 13, 14, 17, 23, 25, 26, 28, 36, 37, 38, 46, 56, 58, 01, 02, 12, 34, 35, 45, 67, 68, 78, where $p(\boldsymbol{x})$ has PAR$_{IHN} = 10.25$.

The examples of our construction satisfy a low PAR$_{IHN}$. Further work should ascertain the proper choice of permutations. Finally, there is an even more obvious variation of Construction 7.16, suggested by the graphs of Figure 4.1, where the functions g_j are chosen either to be quadratic cliques or to be further “nested” versions of Construction 7.16.

7.3 Quantum Interpretations of Spectral Measures

A quantum code with good error correcting capability must produce encoded states that are highly entangled. The single quantum state corresponding to a zero-dimensional quantum code of high distance must also be a highly entangled

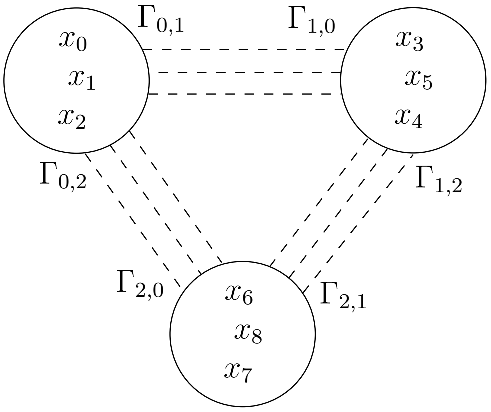


Figure 7.4: *Example of Construction for Low PAR$_{IHN}$*

state [30]. The distance of self-dual quantum codes, or more generally, the APC distance of Boolean functions, may therefore be interpreted as entanglement measures of the corresponding quantum states. No entanglement measure is known that completely quantifies the degree of entanglement in a pure multipartite quantum state of more than 3 qubits, and APC distance is also only a partial measure.

The PAR$_{\mathcal{U}}$ of a vector $\boldsymbol{s}$ also gives information about the quantum state with probability distribution vector $\boldsymbol{s}$. We recall from section 2.1 that the values $|s_i|^2$, where $i \in \mathbb{Z}_{2^n}$, are the probabilities of observing each of 2^n basis states in a particular measurement basis associated with $\boldsymbol{s}$. The maximum value of $|s_i|^2$ over all $i \in \mathbb{Z}_{2^n}$ gives the probability of the most likely outcome of a measurement in this particular measurement basis, and can therefore be interpreted as a partial measure of the uncertainty of the quantum state. If this value is high, the state has low uncertainty in this measurement basis. A local unitary transformation corresponds to a change of measurement basis. Local unitary transformations of the vector $\boldsymbol{s}$ are reversible and do not change the overall entanglement properties of the corresponding quantum state, but the measurement basis is changed, and the magnitudes of the coefficients of $\boldsymbol{s}$, and therefore the uncertainty, may also change. If we can apply any local unitary transformation to $\boldsymbol{s}$, i.e., use any measurement basis, what is the lowest uncertainty, i.e., the highest probability of any basis state, we can achieve? The answer to this question is simply the value of PAR$_{\mathcal{U}}(\boldsymbol{s})$, as defined in Definition 7.12. Since PAR$_{IHN}$ is a lower bound on PAR$_{\mathcal{U}}$, it is also a lower bound on the uncertainty of a quantum state, where uncertainty means the highest probability of observing any basis state in any measurement basis.

Definition 7.20. Given an n-qubit quantum state $|\psi\rangle$, we can write

$$|\psi\rangle = \sum_{i=1}^{R} c_i \, |\phi_i\rangle \,, \tag{7.9}$$

where $c_i \in \mathbb{C}$ and $|\phi_i\rangle$, for $1 \leq i \leq R$, are R of the 2^n basis states in a particular measurement basis. Let r be the smallest possible value of R for any measurement basis. The *Schmidt measure* of $|\psi\rangle$ is then given by

$$E_S(|\psi\rangle) = \log_2(r). \tag{7.10}$$

The Schmidt measure was defined by Eisert and Briegel [18] and later studied in the context of graph states by Hein, Eisert, and Briegel [30]. We can also define the Schmidt measure in terms of the probability distribution vector associated with a quantum state.

Definition 7.21. Let $\boldsymbol{s}$ be a complex-valued vector of length 2^n corresponding to the quantum state $|\psi\rangle$. Given a local unitary transform $\boldsymbol{S} = U\boldsymbol{s}$, where $U \in \mathcal{U}^n$, we define $R = |\{S_i \neq 0 \mid 0 \leq i < n\}|$, i.e., the number of non-zero coefficients of $\boldsymbol{S}$. Let r be the smallest possible value of R for any transformation $U \in \mathcal{U}^n$. The Schmidt measure of $|\psi\rangle$ is then $E_S(|\psi\rangle) = \log_2(r)$.

Proposition 7.22. *Let $|\psi_f\rangle$ be a bipolar quantum state described by the vector $\boldsymbol{s} = 2^{-\frac{n}{2}}(-1)^{f(\boldsymbol{x})}$, where f is a quadratic Boolean function. We can then give an upper bound on the Schmidt measure of $|\psi_f\rangle$,*

$$E_S(|\psi_f\rangle) \leq n - log_2(PAR_{IHN}(\boldsymbol{s})). \tag{7.11}$$

Proof. It can be shown that the coefficients of any $\{I, H, N\}^n$ transform of a quadratic Boolean function have only two different magnitudes [52]. It follows that in the $\{I, H, N\}^n$ transform containing the highest spectral peak, all coefficients must either be zero or have the same magnitude as the peak, and therefore that there must be $\frac{2^n}{\mathrm{PAR}_{IHN}(\boldsymbol{s})}$ non-zero coefficients. □

By Theorem 7.10, (7.11) can also be expressed in terms of λ,

$$E_S(|\psi_f\rangle) \leq n - \lambda(G), \tag{7.12}$$

where G is the graph corresponding to $\boldsymbol{s}$ and $\lambda(G)$ is the size of the maximum independent set over the LC orbit of G. The size of the *minimum vertex cover* of a graph G, denoted $\nu(G)$, is given by $\nu(G) = n - \alpha(G)$, where $\alpha(G)$ is the size of the maximum independent set of G. It was shown by Hein et al. [30] that

$$E_S(|\psi_f\rangle) \leq \nu(G). \tag{7.13}$$

Proposition 7.23. *Let $|\psi_f\rangle$ be a bipolar quantum state described by the vector $\boldsymbol{s} = 2^{-\frac{n}{2}}(-1)^{f(\boldsymbol{x})}$, where f is any Boolean function. The Schmidt measure of $|\psi_f\rangle$ is then lower bounded by*

$$E_S(|\psi_f\rangle) \geq n - log_2(PAR_{\mathcal{U}}(\boldsymbol{s})). \tag{7.14}$$

Proof. In this case, the transforms may have more than two different magnitudes. We do therefore not know how many non-zero coefficients the transform with the highest peak has, but we know that no transform can have any coefficient with higher magnitude than $\mathrm{PAR}_{\mathcal{U}}$, and therefore that all transforms must have at least $\left\lceil \frac{2^n}{\mathrm{PAR}_{\mathcal{U}}(\boldsymbol{s})} \right\rceil$ non-zero coefficients. □

Corollary 7.24. *It follows from Theorem 7.13 that if $\boldsymbol{s} = 2^{-\frac{n}{2}}(-1)^{f(\boldsymbol{x})}$ corresponds to a bipartite graph G, then*

$$E_S(|\psi_f\rangle) = n - log_2(PAR_{\mathcal{U}}(\boldsymbol{s})) = n - log_2(PAR_{IH}(\boldsymbol{s})). \tag{7.15}$$

When we look at non-quadratic Boolean functions, or when we use other transform sets than $\{I, H, N\}$, the transforms may have more than two different magnitudes. The connection between PAR and Schmidt measure in these cases is not clear.

We recall that a Boolean function is called bent if its $\{H\}^n$ transform is flat. Riera, Petrides, and Parker [51, 53] have proposed that this criteria can be generalised by counting how many of the 6^n $\{I,H,N\}^n$ transforms of a Boolean function are flat. A high number of flat spectra indicates maximal distance to a large subset of generalised affine functions, and a high number of maximum uncertainty measurement bases within the complete set of $\{I,H,N\}^n$ measurement bases. Other entanglement measures can also be derived from the $\{I,H,N\}^n$ spectrum.

Definition 7.25. Given a vector $\boldsymbol{s}$, let a be the sum of the fourth powers of all 6^n spectral magnitudes in the $\{I,H,N\}^n$ spectrum, i.e.,

$$a = \sum_{\substack{U\in\{I,H,N\}^n \\ k\in\mathbb{Z}_{2^n}}} |S_k|^4, \tag{7.16}$$

where $\boldsymbol{S} = U\boldsymbol{s}$. The *Clifford merit factor* (CMF) [38] of $\boldsymbol{s}$ is then given by

$$\mathrm{CMF}(\boldsymbol{s}) = \frac{6^n}{a - 6^n}. \tag{7.17}$$

High APC distance, low PAR_{IHN}, and high CMF are clearly correlated properties, and the best self-dual quantum codes optimise all of them. Non-quadratic Boolean functions with equally good properties as the best quadratic functions have also been found. The CMF is an entanglement measure, and it can be shown that CMF remains invariant under any local unitary transformation, i.e., $\mathrm{CMF}(\boldsymbol{s}) = \mathrm{CMF}(U\boldsymbol{s})$, where $U \in \mathcal{U}^n$. Since the overall entanglement of a quantum state does not change under local unitary transformations, all entanglement measures should have this property. The Schmidt measure and $\mathrm{PAR}_{\mathcal{U}}$ are also invariant under local unitary transformations. It may be possible to generalise CMF by extending the transform set while increasing the power of the spectral magnitudes. This could give an infinite sequence of entanglement measures, all invariant under local unitary transformations, which converges towards $\mathrm{PAR}_{\mathcal{U}}$.

Chapter 8

Conclusions and Open Problems

In this thesis we have studied zero-dimensional quantum codes and their interpretations as quantum states, self-dual additive codes over GF(4), graphs, and Boolean functions. We have looked at different properties and generalisations under each interpretation. For reasons of computational complexity, we have restricted our study to self-dual quantum codes of length up to 30. But even for codes of such short length, there are many interesting problems.

Problem 8.1. As seen in Table 2.1, the best achievable distances for codes of length from 23 to 27 are not known. What are the optimal distances for these lengths? In particular, does there exist a $[[24, 0, 10]]$ quantum code?

Many examples have been shown of self-dual quantum codes of high minimum distance with highly structured and regular graph representations. In particular, we have searched all circulant graph codes with length up to 30 for *nested regular graphs*. The nested regular description does not completely characterise the structure of a graph, but we have also shown that nested regular graph representations of strong codes contain long cycles. Initial results further suggest that the long cycles should be arranged in such a way that no smaller cycles are induced. We have also identified graph representations with *minimum regular vertex degree* for many self-dual quantum codes. Graphs with minimum regular vertex degree have the lowest possible number of edges, and the corresponding generator matrices are therefore as sparse as possible. In many applications of classical coding theory, sparsity of the generator matrix is a desired property.

Problem 8.2. Identify other regular graph structures than those listed in Table 4.1, in particular for codes of length 28 and above.

Problem 8.3. Is it possible to further generalise and extend the description of highly regular graphs corresponding to strong self-dual quantum codes?

Problem 8.4. Devise a construction technique for nested regular graphs, giving self-dual quantum codes with a predictable minimum distance.

Problem 8.5. Find graphs with minimum regular vertex degree corresponding to self-dual quantum codes of length above 27 and distance higher than 8. Of particular interest is the existence of a graph with regular vertex degree 11 corresponding to a $[[30, 0, 12]]$ code.

We investigated all self-dual quantum codes of length up to 30 corresponding to graphs with circulant adjacency matrices, and found codes of equally high distance to those of Gulliver and Kim [29] in their more general search of all self-dual additive codes over GF(4) with circulant generator matrices.

Problem 8.6. How many GF(4)-circulant codes also have circulant graph representations, and are codes with circulant graph representations stronger than general codes?

We have seen that the quadratic residue construction produces codes corresponding to strongly regular Paley graphs. The largest clique in a Paley graph is known to be very small, compared to the number of vertices in the graph [55]. Since it can also be shown that all Paley graphs are isomorphic to their complements, their independence numbers must be equally low. We have seen that graphs corresponding to strong self-dual quantum codes have small independent sets over their whole *LC orbit*, and this implies that the largest independent set of a Paley graph remains small when any sequence of local complementations is applied.

Problem 8.7. Give bounds on the size of the largest independent set over the LC orbit of a Paley graph.

Problem 8.8. Can other families of strongly regular graphs, or families of graphs known to have small independence numbers, be used to construct self-dual quantum codes of high distance?

We showed that some self-dual quantum codes, in particular the $[[11, 0, 5]]$ and $[[18, 0, 8]]$ codes, have no regular graph representation. Glynn et al. [23] have used finite geometry to construct and characterise the $[[18, 0, 8]]$ code.

Problem 8.9. Do strong self-dual quantum codes with no regular graph representation correspond to graphs that are highly structured in some other way, and can this structure be generalised?

We have classified all self-dual additive codes over GF(4) of length up to 12. Enumerating all codes of higher length, with a reasonable amount of computational resources, is not possible using the algorithms in section 5.2.

Problem 8.10. Devise an algorithm for canonising a graph, like Algorithm 5.2, but without requiring that the whole LC orbit of the graph is generated. This could give a much faster method for enumerating LC orbits.

Problem 8.11. Devise an efficient algorithm for determining whether two graphs are LC-equivalent, like the one described by Van den Nest et al. [58] and Bouchet [6], but that also considers equivalence via graph isomorphism.

We have seen that the *APC distance* of a Boolean function is equal to the distance of the corresponding zero-dimensional quantum code. For quadratic Boolean functions, which correspond to self-dual additive codes over GF(4), we can use the efficient Algorithm 3.1 to find the APC distance. Boolean functions of higher degree correspond to non-quadratic quantum codes, and we have used the much more complex Algorithm 6.2 to find their distance. We have found several non-quadratic Boolean functions, listed in Table 6.6, with exactly the same APC distance and PAR_{IHN} as the best quadratic functions.

Problem 8.12. Are there better algorithms for finding the APC distance of a non-quadratic Boolean function? For instance, do cubic Boolean functions correspond to some generalisation of additive codes over GF(4) where the distance can be found efficiently?

Problem 8.13. Improve Algorithm 6.3 and classify all orbits in $\mathcal{O}_{1,n}$ and $\mathcal{O}_{2,n}$ for $n > 5$.

Problem 8.14. Do there exist non-quadratic Boolean functions with higher APC distance or lower PAR_{IHN} than the best quadratic functions?

We have defined λ, the size of the largest independent set over an LC orbit, and Λ_n, the minimum value of λ over all LC orbits of graphs on n vertices. It has also been shown that $\mathrm{PAR}_{IHN} = 2^\lambda$ for quadratic Boolean functions, and we have given bounds on Λ_n. The bounds on Λ_n are also bounds on PAR_{IHN} and $\mathrm{PAR}_{\mathcal{U}}$.

Problem 8.15. Improve the bounds on Λ_n, or find exact values of Λ_n for $n \geq 15$.

Problem 8.16. Can other symmetries of non-quadratic Boolean functions be found by adding other matrices to the set $\{I, H, N\}$?

Problem 8.17. Is there an operation, similar to local complementation, that generates orbits of equivalent hypergraphs?

Problem 8.18. Is there a relationship between the maximum independent set in the orbit of a hypergraph and the PAR_{IHN} of the corresponding Boolean function? Maybe we need to consider PAR with respect to a larger transform set than $\{I, H, N\}$ in order to find such a relationship.

We have seen that Construction 7.15 gives Boolean functions of high degree with predictable PAR_{HN}, and we have proposed the more general Construction 7.16 for Boolean functions of high degree with low PAR_{IHN}.

Problem 8.19. What is the best choice of permutations in Construction 7.16?

Problem 8.20. Find bounds on the PAR_{IHN} of functions generated by Construction 7.16 when specific permutations are used.

Construction 7.15 is a generalisation of the path graph, which have optimal PAR_{HN} but high PAR_{IHN}. It can be shown that complete graphs (cliques) have optimal PAR_{IH}, but that they also have high PAR_{IHN}. Functions that optimise PAR_{IHN} should do well for both PAR_{HN} and PAR_{IH}, and should perhaps be some compromise between clique graphs and path graphs. We have seen that some of the quadratic Boolean functions with lowest PAR_{IHN} correspond to nested clique graphs, which contain both cliques and long disjoint paths.

Zero-dimensional quantum codes of high distance correspond to highly entangled quantum states, and also to Boolean functions that satisfy the *aperiodic propagation criterion* of high order or degree. APC is related to several other criteria which measure the cryptographic strength of a Boolean function. This suggests that a highly entangled quantum state may correspond to a cryptographically strong Boolean function. We have shown that spectral properties of Boolean functions, such as APC distance, $\mathrm{PAR}_{\mathcal{U}}$, and PAR_{IHN} can be used to measure the degree of entanglement in a quantum state, as can the Schmidt measure and the Clifford Merit Factor [38]. In this thesis, we have studied quantum states with coefficients from the set $\{-1, 1\}$, represented by the bipolar truth table of a Boolean function. This is, of course, only a subset of all possible quantum states.

Problem 8.21. Classify spectral measures of quantum states using other transform sets than $\{I, H, N\}^n$.

Problem 8.22. Find efficient techniques for approximating $\mathrm{PAR}_{\mathcal{U}}$.

Problem 8.23. Study the properties of zero-dimensional quantum codes corresponding to non-bipolar quantum states, for instance quantum states with coefficients from the sets $\{0, 1\}$ or $\{-1, 0, 1\}$.

Bibliography

[1] Aigner, M. and van der Holst, H.: "Interlace polynomials". *Linear Algebra and its Applications*, **377**, pp. 11–30, January 2004.

[2] Arratia, R., Bollobás, B., and Sorkin, G. B.: "The interlace polynomial: a new graph polynomial". In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 237–245, Society for Industrial and Applied Mathematics, 2000.

[3] Arratia, R., Bollobás, B., and Sorkin, G. B.: "The interlace polynomial of a graph". *Journal of Combinatorial Theory, Series B*, **92**(2), pp. 199–233, November 2004. `http://arxiv.org/pdf/math/0209045`.

[4] Bachoc, C. and Gaborit, P.: "On extremal additive $\mathbb{F}_4$ codes of length 10 to 18". *Journal de Théorie des Nombres de Bordeaux*, **12**(2), pp. 255–271, 2000. `http://almira.math.u-bordeaux.fr/jtnb/2000-2/Bachoc.ps`.

[5] Bouchet, A.: "Isotropic systems". *European Journal of Combinatorics*, **8**(3), pp. 231–244, 1987.

[6] Bouchet, A.: "An efficient algorithm to recognize locally equivalent graphs". *Combinatorica*, **11**(4), pp. 315–329, 1991.

[7] Bouchet, A.: "Recognizing locally equivalent graphs". *Discrete Mathematics*, **114**(1–3), pp. 75–86, April 1993.

[8] Briegel, H. J. and Raussendorf, R.: "Persistent entanglement in arrays of interacting particles". *Physical Review Letters*, **86**(5), pp. 910–913, January 2001. `http://arxiv.org/pdf/quant-ph/0004051`.

[9] Bron, C. and Kerbosch, J.: "Algorithm 457: Finding all cliques of an undirected graph". *Communications of the ACM*, **16**(9), pp. 575–577, September 1973.

[10] Calderbank, A. R., Rains, E. M., Shor, P. M., and Sloane, N. J. A.: "Quantum error correction via codes over GF(4)". *IEEE Transactions on Information Theory*, **44**(4), pp. 1369–1387, July 1998. `http://www.research.att.com/~njas/doc/qc2.pdf`.

[11] Cameron, P. J.: "Strongly regular graphs". In *Topics in Algebraic Graph Theory*, edited by L. W. Beineke and R. J. Wilson, volume 102 of *Encyclopedia of Mathematics and its Applications*, chapter 8, pp. 203–221, Cambridge University Press, December 2004. `http://www.maths.qmw.ac.uk/~pjc/preprints/bw_srg.ps`.

[12] Carlet, C.: "On cryptographic propagation criteria for Boolean functions". *Information and Computation*, **151**(1–2), pp. 32–56, May 1999.

[13] Danielsen, L. E.: "Database of nonquadratic Boolean functions". Web page, February 2005. `http://www.ii.uib.no/~larsed/nonquad/`.

[14] Danielsen, L. E.: "Database of self-dual quantum codes". Web page, February 2005. `http://www.ii.uib.no/~larsed/vncorbits/`.

[15] Danielsen, L. E., Gulliver, T. A., and Parker, M. G.: "Aperiodic propagation criteria for Boolean functions", October 2004. Submitted to Information and Computation. `http://www.ii.uib.no/~larsed/papers/apc.pdf`.

[16] Danielsen, L. E. and Parker, M. G.: "Spectral orbits and peak-to-average power ratio of Boolean functions with respect to the $\{I, H, N\}^n$ transform", January 2005. To appear in the proceedings of Sequences and Their Applications, SETA'04, Lecture Notes in Computer Science, Springer-Verlag. `http://www.ii.uib.no/~larsed/papers/seta04-parihn.pdf`.

[17] Davis, J. A. and Jedwab, J.: "Peak-to-mean power control in OFDM, Golay complementary sequences and Reed-Muller codes". *IEEE Transactions on Information Theory*, **45**(7), pp. 2397–2417, November 1999.

[18] Eisert, J. and Briegel, H. J.: "Schmidt measure as a tool for quantifying multiparticle entanglement". *Physical Review A*, **64**(022306), August 2001. `http://arxiv.org/pdf/quant-ph/0007081`.

[19] Fuller, J., Millan, W., and Dawson, E. P.: "Efficient algorithms for analysis of cryptographic Boolean functions". In *Proceedings of the Thirteenth Australasian Workshop on Combinatorial Algorithms (AWOCA 2002)*, edited by E. Billington, D. Donovan, and A. Khodkar, pp. 133–150, May 2002. `http://www.isrc.qut.edu.au/people/fuller/efficient.ps`.

[20] Gaborit, P., Huffman, W. C., Kim, J.-L., and Pless, V.: "On the classification of extremal additive codes over GF(4)". In *Proceedings of the 37th Allerton Conference on Communication, Control, and Computing*, pp. 535–544, September 1999. `http://www2.math.uic.edu/~jlkim/all_final.ps`.

[21] Gaborit, P., Huffman, W. C., Kim, J.-L., and Pless, V.: "On additive GF(4) codes". In *Codes and Association Schemes*, edited by A. Barg and S. Litsyn, volume 56 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pp. 135–149, American Mathematical Society, 2001. `http://www.math.unl.edu/~jlkim/dimacs4.ps`.

[22] Glynn, D. G.: "On self-dual quantum codes and graphs", April 2002. Submitted to the Electronic Journal of Combinatorics. `http://homepage.mac.com/dglynn/.cv/dglynn/Public/SD-G3.pdf-link.pdf`.

[23] Glynn, D. G., Gulliver, T. A., Maks, J. G., and Gupta, M. K.: "The geometry of additive quantum codes", April 2004. Submitted to Springer-Verlag.

[24] Gottesman, D.: *Stabilizer Codes and Quantum Error Correction*. Ph.D. thesis, California Institute of Technology, May 1997. `http://arxiv.org/pdf/quant-ph/9705052`.

[25] Gottesman, D.: "An introduction to quantum error correction". In *Quantum Computation: A Grand Mathematical Challenge for the Twenty-First Century and the Millennium*, edited by S. J. Lomonaco, Jr., volume 58 of *Proceedings of Symposia in Applied Mathematics*, American Mathematical Society, April 2002. `http://arxiv.org/pdf/quant-ph/0004072`.

[26] Grassl, M.: "Encoding circuits for quantum error-correcting codes". Web page, February 2002. `http://iaks-www.ira.uka.de/home/grassl/QECC/circuits/`.

[27] Grassl, M.: "Bounds on d_{min} for additive $[[n, k, d]]$ QECC". Web page, February 2003. `http://iaks-www.ira.uka.de/home/grassl/QECC/TableIII.html`.

[28] Grassl, M., Klappenecker, A., and Rötteler, M.: "Graphs, quadratic forms, and quantum codes". In *Proceedings of the 2002 IEEE International Symposium on Information Theory*, p. 45, 2002. `http://faculty.cs.tamu.edu/klappi/papers/ISIT2002.pdf`.

[29] Gulliver, T. A. and Kim, J.-L.: "Circulant based extremal additive self-dual codes over GF(4)". *IEEE Transactions on Information Theory*, **50**(2), pp. 359–366, February 2004. `http://www.math.unl.edu/~jlkim/quandcc_final.ps`.

[30] Hein, M., Eisert, J., and Briegel, H. J.: "Multi-party entanglement in graph states". *Physical Review A*, **69**(062311), June 2004. `http://arxiv.org/pdf/quant-ph/0307130`.

[31] Höhn, G.: "Self-dual codes over the Kleinian four group". *Mathematische Annalen*, **327**, pp. 227–255, October 2003. `http://arxiv.org/pdf/math/0005266`.

[32] Knill, E., Laflamme, R., Ashikhmin, A., Barnum, H. N., Viola, L., and Zurek, W. H.: "Introduction to quantum error correction". *Los Alamos Science*, **27**, pp. 188–225, 2002. `http://lib-www.lanl.gov/cgi-bin/getfile?00783364.pdf`.

[33] Knill, E., Laflamme, R., Barnum, H. N., Dalvit, D. A., Dziarmaga, J. J., Gubernatis, J. E., Gurvits, L., Ortiz, G., Viola, L., and Zurek, W. H.: "Quantum information processing: A hands-on primer". *Los Alamos Science*, **27**, pp. 2–37, 2002. `http://lib-www.lanl.gov/cgi-bin/getfile?00783350.pdf`.

[34] MacWilliams, F. J., Odlyzko, A. M., Sloane, N. J. A., and Ward, H. N.: "Self-dual codes over GF(4)". *Journal of Combinatorial Theory, Series A*, **25**, pp. 288–318, 1978.

[35] McKay, B. D.: *nauty User's Guide*. 2003. `http://cs.anu.edu.au/~bdm/nauty/nug.pdf`.

[36] Nielsen, M. A. and Chuang, I. L.: *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[37] Parker, M. G.: "Generalised S-box nonlinearity". Public Document NES/DOC/UIB/WP5/020/A, NESSIE, February 2003. `https://www.cosic.esat.kuleuven.ac.be/nessie/reports/phase2/SBoxLin.pdf`.

[38] Parker, M. G.: "Univariate and multivariate merit factors", January 2005. To appear in the proceedings of Sequences and Their Applications, SETA'04, Lecture Notes in Computer Science, Springer-Verlag. `http://www.ii.uib.no/~matthew/seta04-mf.pdf`.

[39] Parker, M. G. and Gulliver, T. A.: "On graph symmetries and equivalence of the six variable double-clique and wheel", July 2003. Unpublished.

[40] Parker, M. G. and Rijmen, V.: "The quantum entanglement of binary and bipolar sequences". In *Sequences and Their Applications, Proceedings of SETA'01*, edited by T. Helleseth, P. V. Kumar, and K. Yang, Discrete Mathematics and Theoretical Computer Science Series, Springer-Verlag, London, 2002. `http://arxiv.org/pdf/quant-ph/0107106`.

[41] Parker, M. G. and Tellambura, C.: "A construction for binary sequence sets with low peak-to-average power ratio". In *Proceedings of the 2002 IEEE International Symposium on Information Theory*, p. 239, 2002. `http://www.ii.uib.no/~matthew/634isit02.pdf`.

[42] Parker, M. G. and Tellambura, C.: "A construction for binary sequence sets with low peak-to-average power ratio". Technical Report 242, Department of Informatics, University of Bergen, Bergen, Norway, February 2003. `http://www.ii.uib.no/publikasjoner/texrap/pdf/2003-242.pdf`.

[43] Pless, V. S. and Huffman, W. C. (editors): *Handbook of Coding Theory*. Elsevier, Amsterdam, 1998.

[44] Preneel, B., Van Leekwijck, W., Van Linden, L., Govaerts, R., and Vandewalle, J.: "Propagation characteristics of Boolean functions". In *Advances in Cryptology - EUROCRYPT '90*, edited by I. B. Damgård, volume 473 of *Lecture Notes in Computer Science*, pp. 161–173, Springer-Verlag, 1991. `http://www.cosic.esat.kuleuven.ac.be/publications/article-42.pdf`.

[45] Radziszowski, S. P.: "Small Ramsey numbers". *The Electronic Journal of Combinatorics*, July 2004. Dynamical Survey DS1. `http://www.combinatorics.org/Surveys/ds1.pdf`.

[46] Rains, E. M., Hardin, R. H., Shor, P. W., and Sloane, N. J. A.: "A nonadditive quantum code". *Physical Review Letters*, **79**(5), pp. 953–954, August 1997. `http://arxiv.org/pdf/quant-ph/9703002`.

[47] Rains, E. M. and Sloane, N. J. A.: "Self-dual codes". In *Handbook of Coding Theory*, edited by V. S. Pless and W. C. Huffman, volume 1, pp. 177–294, Elsevier, Amsterdam, 1998. `http://arxiv.org/pdf/math.CO/0208001`.

[48] Raussendorf, R. and Briegel, H. J.: "A one-way quantum computer". *Physical Review Letters*, **86**(22), pp. 5188–5191, May 2001. `http://arxiv.org/pdf/quant-ph/0010033`.

[49] Raussendorf, R., Browne, D. E., and Briegel, H. J.: "Measurement-based quantum computation on cluster states". *Physical Review A*, **68**(022312), August 2003. `http://arxiv.org/pdf/quant-ph/0301052`.

[50] Rieffel, E. G. and Polak, W.: "An introduction to quantum computing for non-physicists". *ACM Computing Surveys*, **32**(3), pp. 300–335, September 2000. `http://arxiv.org/pdf/quant-ph/9809016`.

[51] Riera, C. and Parker, M. G.: "Generalised bent criteria for Boolean functions (I)", December 2004. Submitted to IEEE Transactions on Information Theory. `http://arxiv.org/pdf/cs.IT/0502049`.

[52] Riera, C. and Parker, M. G.: "Spectral interpretations of the interlace polynomial", March 2005. Accepted for WCC 2005. `http://www.ii.uib.no/~matthew/WCC7.pdf`.

[53] Riera, C., Petrides, G., and Parker, M. G.: "Generalised bent criteria for Boolean functions (II)", December 2004. Submitted to IEEE Transactions on Information Theory. `http://arxiv.org/pdf/cs.IT/0502050`.

[54] Schlingemann, D. and Werner, R. F.: "Quantum error-correcting codes associated with graphs". *Physical Review A*, **65**(012308), January 2002. `http://arxiv.org/pdf/quant-ph/0012111`.

[55] Shearer, J. B.: "Lower bounds for small diagonal Ramsey numbers". *Journal of Combinatorial Theory, Series A*, **42**(2), pp. 302–304, July 1986.

[56] Sloane, N. J. A.: "The On-Line Encyclopedia of Integer Sequences". Web page, 2005. `http://www.research.att.com/~njas/sequences/`.

[57] Tonchev, V. D.: "Error-correcting codes from graphs". *Discrete Mathematics*, **257**(2–3), pp. 549–557, November 2002.

[58] Van den Nest, M., Dehaene, J., and De Moor, B.: "An efficient algorithm to recognize local Clifford equivalence of graph states". *Physical Review A*, **70**(034302), September 2004. `http://arxiv.org/pdf/quant-ph/0405023`.

[59] Van den Nest, M., Dehaene, J., and De Moor, B.: "Graphical description of the action of local Clifford transformations on graph states". *Physical Review A*, **69**(022316), February 2004. `http://arxiv.org/pdf/quant-ph/0308151`.