Studies on Boolean Functions Related to Quantum Computing

by

Mika Hirvensalo

To be presented, with the permission of the Faculty of Mathematics and Natural Sciences of the University of Turku, for public criticism in Auditorium XXI of the University on February 8th, 2003, at 12 noon

University of Turku
Department of Mathematics
FIN-20014 Turku, Finland

2003

Supervisor

Professor Juhani Karhumäki
Department of Mathematics
University of Turku
FIN-20014 Turku
Finland

Reviewers

Professor Farid Ablayev
Department of Theoretical Cybernetics
Kazan State University
420008 Kazan
Russia

Professor Juraj Hromkovič
Lehrstuhl für Informatik I
RWTH Aachen
Ahornstraße 55
D-52074 Aachen
Germany

Opponent

Professor Rūsiņš Freivalds
Institute of Mathematics and Computer Science
University of Latvia
Raiņa bulvaris 29
LV-1459 Riga
Latvia

ISBN 951-29-2203-7
ISSN 1239-1883
Painosalama Oy
2003

Horas non numero nisi serenas

To my dear wife Virpi

Acknowledgements

I would like to thank my supervisor Professor Juhani Karhumäki, first for suggesting me to study quantum computing, a model of computing that had not been studied in the University of Turku before. Studying quantum computing has been most instructive: I believe that many aspects of mathematics and theoretical computer science would still be hidden to me if I never had entered into this fascinating field. I also thank Juhani Karhumäki for fruitful discussions, not forgetting the fact that he has provided me excellent working conditions; this work has been supported by the Academy of Finland under grants 14047 and 44087. For outstanding working conditions, I also thank Turku Centre for Computer Science (TUCS).

The support by Professor Grzegorz Rozenberg and Academician Arto Salomaa is gratefully acknowledged. Their encouragement ensured that I was able to write a monograph on quantum computing, and the writing period of that book was indeed a very interesting interval, in its own special way also supporting this work.

For co-operation, I thank Dr. Vesa Halava, Dr. Tero Harju, Prof. Juhani Karhumäki, Dr. Sebastian Seibert and Dr. Ronald de Wolf. Here it does not matter that the articles we have written together do not have any direct connections to *this* monograph, it has been nice to work with you, anyway! I hope to have some co-operation in future, too.

Special thanks go to Ph. Lic. Esa Latikka for clarifying me the importance of mathematics a little bit more than ten years ago. I also thank Dr. Hannu Tarnanen, with whom I formerly shared the office room, for discussions about mathematics, and especially for his seemingly non-conventional views about the life itself. Dr. Tero Harju also deserves special thanks for always being open for mathematical discussions. Because of the same reason, I thank all my colleagues and friends, but here I cannot mention all of them by name, because this page is short anyway.

Last but not least, I thank the reviewers, Professors Farid Ablayev and Juraj Hromkovič for their careful revision work.

Turku, Finland, January 2003 *Mika Hirvensalo*

Contents

Chapter 1

Introduction

The motivation to this work comes from the theory of quantum computing, but the main object studied here is the *algebraic representation degree* of Boolean functions. Studying together these two apparently different-looking objects, quantum computing and Boolean functions, is not a contradictory idea at all. In fact, there is a very close connection between an algebraic property of a Boolean function and a quantum query algorithm computing that function. Anyway it should be mentioned that quantum computing plays only a minor role in this thesis, the main emphasis is on Boolean functions.

Boolean functions have, indeed, many aspects to be studied: they have combinatorial properties which can be used to characterize their complexity issues, but also their algebraic properties are of great interest. The relationships between algebraic and combinatorial properties are not, in general, known well enough. The main purpose of this work is to study the representation degree of Boolean functions and to introduce some new tools which may be used to learn more about Boolean functions. Especially, analogues to the classical theory of continuous functions of many variables may be of interest. Anyway, I want to welcome the reader into the endless maze of algebraic properties of Boolean functions.

A very short Chapter 2 is devoted to basic facts of Boolean functions. Chapter 3 is to explain quantum computing, query algorithms, and the connections between quantum query algorithms and the representation degrees of Boolean functions. Chapter 4 includes the basic facts of Boolean functions which will be used throughout this thesis. In that chapter, there are also some examples which demonstrate the usefulness of the formalism used in this work. In Chapter 5, a new basis for representing Boolean functions is introduced. Some of the most important aspects of the basis introduced in Chapter 5 are also studied in Chapter 6. The interesting topics, like "obvious analogues" to classical mathematics, are however shattered throughout the chapters.

The following words are also worth mentioning: The purpose of this thesis is to introduce tools for learning about algebraic properties of the Boolean functions. The degree properties can be used to provide lower bounds for quantum query complexity, but the other purpose is also important: I hope that this thesis could be able to serve as a good source for studying the degree properties of Boolean functions for anyone who is interested in this area.

Chapter 2

Boolean Functions

The notion of a *Boolean function* is one of the most fundamental things in mathematics. By a *Boolean variable* we mean a variable which assumes only two different values. Unless explicitly stated otherwise, we will assume that the domain of the Boolean variables is the *binary field* $\mathbb{F}_2$, the field of two elements (see the definitions in Chapter 4). Primarily, by a Boolean function we will understand a function whose domain is the Cartesian product of the binary field, $\mathbb{F}_2^N$, and who has a two-element set as the range, but secondarily, we will also study more general functions $\mathbb{F}_2^N \to \mathbb{C}$.

Many things in mathematics can be conceptually regarded as Boolean functions. For instance, to define a subset of $\mathbb{F}_2^N$ is equivalent to defining a Boolean function $f : \mathbb{F}_2^N \to \{0, 1\}$. Therefore, the theory of binary error-correcting codes, for instance, can be regarded as a subarea of the study of Boolean functions – not so literally, only because of this very broad definition of Boolean functions.

In fact, any function $f : A \to B$, where A and B are finite sets, has its interpretation by Boolean functions. To be more precise, it is always possible to encode the elements of A and B into bit strings, and to shatter function f itself into many separate two-valued functions f_1, …, f_k, each of which are computing a bit of the outcome.

To study the computational complexity, Boolean functions are, because of their conceptual simplicity, of a great interest. It is traditional to define a simple set of Boolean functions like $\{\vee, \wedge, \neg\}$ (logical or, and, not) and to question how many applications of these primitive functions (which will be also called *gates*) are needed in order to describe a given Boolean function. It is a well-known fact that these three gates (in fact, one of $\vee$, $\wedge$ could be even omitted) are sufficient to define any Boolean function, see [12], for instance. A simple counting argument [22] shows that *almost all* Boolean functions on N variable need roughly $2^N/N$ of gates to be implemented, but despite this, we do not know any family of Boolean functions which provably would require more than a linear number of gates to be implemented!

The point of view chosen for this thesis is to study several representations of Boolean functions. The major points of interest are the *representation* and *approximation* degrees of particular Boolean functions.

Chapter 3

Quantum Computing

3.1 On the Development of Physical Theories

To describe, to explain, to predict, and to understand the phenomena in the nature are fundamental tasks of natural sciences. Mathematical models have turned out to be successful when traveling from a description to understanding. To illustrate this journey, we will discuss about the development of our picture of the solar system.

In the *geocentric* picture, earth remains stagnant and the other heavenly bodies such as the sun, the moon, the planets and also the sphere of the fixed stars orbit around the earth. Such a model can relatively well *explain* some of the visible effects on the earth: sun rises and sets, moon has its phases, eclipses occur, etc. However, the model was not very successful in explaining the tracks of the planets: sometimes it seemed that the planets were able to reverse their traveling direction, move backwards, and then to begin travel forward again. This effect was not understandable nor predictable in the geocentric model.

Some modifications were proposed, but, as the time passed by, it turned out that the *heliocentric* view of the universe was more successful and simpler than the geocentric model. In fact, after replacing the idea of circular planet orbits by elliptic ones, the *predictions* of the heliocentric model became rather precise. We can thus say that the heliocentric model with elliptic planet orbits *described*, *predicted*, and *explained* very well the motions of heavenly objects, i.e., it gave a good picture of what is going on in the solar system, but it did not offer deeper *understanding*. Why are the planets orbiting around the sun? Why are the orbits elliptic?

In his famous work [20], Isaac Newton published his ideas of the *gravitation* and explained how the gravitational effects force the planets orbit elliptically around the sun. It should be emphasized here that he did not explain the *nature* of gravitation, but rather the *way* how gravitation works. However, the model based on the gravitational force gave a deeper *under-*

standing of the solar system behaviour: we can use the principles of the gravitation to *mathematically derive* the form of planetary orbits, as well as to explain the behaviour of many other phenomena such as comets, asteroids, etc.

As time went on, it turned out that several observations, such as the drifting of the perihelion of planet Mercury, was not fully explained by the Newtonian theory of gravitation. Physics, in its very core, is an experimental science, implying that a theory that does not agree with the observations must be modified or rejected. Centuries after Newton, Albert Einstein offered, in his *general theory of relativity*, an explanation for the *mechanism* of the gravity. Einstein's theory had two advantages: First, the Newtonian theory of gravity can be obtained from Einstein's theory as an approximation, but it seems that Einstein's theory of gravity is much more accurate. Secondly, Einstein offered an *explanation* for the gravitational effects as a curvature of space-time structure, thus giving, besides the better *predictions*, a deeper *understanding* on gravitational effects. On the other hand, Einstein's theory of the gravity has a disadvantage of being mathematically much more complicated than Newton's theory.

3.2 On the Development of Quantum Physics

Another revolutionary theory was born in the beginning of 20th century. Several features in the physical world, such as the *black body radiation*, the *photoelectric effect*, and the *stability of hydrogen atom* were not explained – or were only partially explained by the physical theories developed so far. For a reader interested in the treatment of these phenomena, we give [27] as a general source referring to the early work on *quantum physics*.

To illustrate the reasons leading to the development of quantum physics, rather a good example is *light*, or *electromagnetic radiation* in general. For over centuries, the nature of light has caused discussion in the scientific community: should we regard light as a flow of small particles or as an undulatory phenomenon?

In the beginning of 19th century, Young demonstrated by his famous two-slit experiment that light indeed has inherently wave-like characteristics. Several decades after that, Maxwell and Hertz carried out research revealing that light should be encountered as electromagnetic radiation, which has good mathematical models describing it as waves of the electromagnetic field.

On the other hand, the theory of electromagnetic radiation was not sufficient to explain the observed *frequency spectrum* of a radiating *black body*. There were two theories, Wien's law of radiation and Rayleigh-Jean's law, which both attempted to explain the spectrum, but these two contradictory laws both failed.

In 1900, Max Planck published his radiation law, which eventually succeeded in *describing* the observed spectrum. An outstanding feature in Planck's radiation law was that it was derived under the assumption that radiation can be extracted only in discrete packets whose energy E is proportional to the frequency ν:

$$E = h\nu, \tag{3.2.1}$$

where h is the famous *Planck's constant*, approximately given as

$$h = 6.62608 \cdot 10^{-34} Js. \tag{3.2.2}$$

It could be said that the assumption of discrete radiation packets, *quanta*, reopened the old discussion about the nature of the electromagnetic radiation: apparently, the radiation had also some corpuscular features.

In 1905 Albert Einstein explained the photoelectric effect basing on Planck's quantum hypothesis. The photoelectric effect means that negatively charged metal loses its charge when exposed to radiation of a certain frequency. Previously it was not understood why the effect itself depended on the frequency of the radiation, not on its intensity. Einstein pointed out that under Planck's quantum hypothesis, the dependency on frequency is natural: Assuming that the radiation consists on discrete packets, whose energy is given by Equation (3.2.1), it is plain that the higher the frequency, the greater is the energy of those radiation packets to extract the electrons. Einstein also suggested name *photon* for the light packets.

A former model of the hydrogen atom consisted of an electron spinning around a proton. The classical theory of Maxwell however suggested that an electron orbiting around the nucleus should consistently emit electromagnetic radiation, thus loosing its energy and finally to collapse to the nucleus. In 1912 Niels Bohr suggested a model where the electron of the hydrogen atom can have *stationary orbits*, where it is not loosing its energy, and a gain or a loss of energy causes it to transfer to lower or higher orbit, respectively. Bohr's model explained the observed *energy spectrum* of hydrogen up to the measurement precision of that time. It turned out that replacing the notion of a *corpuscular electron* by that one of an *wave-like electron* leads into a model easier to comprehend.

Consequently, Bohr's explanation suggested that electrons, which have traditionally been regarded as particles, have, in some situations, a more natural explanation as a wave-like phenomena.

Encouraged by the previous results, Luis de Broglie introduced in 1924 a general hypothesis of the *wave-particle duality*: each particle can also be described as waves, whose wavelength λ is

$$\lambda = \frac{h}{p},$$

where p is the momentum of the particle and h the Planck's constant (3.2.2).

Many famous physicists, such as W. Heisenberg, M. Born, P. Jordan, W. Pauli, and P. A. M. Dirac have given their substantial impacts on the development of early quantum physics.

We conclude this section by noticing that quantum physics was developed, loosely speaking, in order to unify two apparently different views: objects should be treated *both* as particles and as waves. Consequently, the mathematical formalism of quantum physics is different from that one of classical physics, and the next section is devoted for that formalism. It must be emphasized here that we intend to describe only the dynamics of so-called *closed quantum systems*. The mathematical description of those systems is somewhat simpler than the general formalism, and for the purposes of this thesis, the description based on the closed systems is sufficient.

3.3 Mathematical Formalism of Quantum Physics

In this section, we represent, on the level needed to follow this thesis, the mathematical formalism of so-called *finite-level* quantum systems. For a more accurate representation, see [12]. By a finite-level quantum system we understand a quantum physical system which has only finitely many *pairwisely distinguishable* states. We say that system states $\{s_1, \ldots, s_k\}$ are pairwisely distinguishable, if we can always tell, with a probability of 1, which is the state of the system, provided that the system actually is one of the states s_1, ..., s_k.

The mathematical formulation of a finite-level quantum system becomes easier to comprehend after the following example concerning a classical probabilistic system.

Example 3.1. Let $A = \{a_1, \ldots, a_n\}$ be a finite set and consider a physical system S capable of being in n distinguishable states. Labeling those n states by $[a_1]$, $[a_2]$, ..., $[a_n]$ we can say that S is capable of representing set A or that the system is a *realization* of an element of A.

Whenever system S is in one of states $[a_1]$, ..., $[a_n]$, it is possible, because the states were assumed to be distinguishable, to tell with certainty which is the element of A that S is currently representing. We call the states $[a_1]$, ..., $[a_n]$ of S *pure*.

It is also possible to introduce *mixed states* of S as *convex combinations* of the pure states: a general mixed state of S will be expressed as

$$p_1[a_1] + p_2[a_2] + \ldots + p_n[a_n], \tag{3.3.1}$$

where $p_1 \geq 0$ and $p_1 + p_2 + \ldots + p_n = 1$. A mixed state (3.3.1) is simply interpreted as a probability distribution over pure states $[a_i]$: when the system in state (3.3.1) is observed, we find that the system represents element a_i with a probability of p_i.

It is sometimes useful to understand the pure states $[a_1]$, ..., $[a_n]$ as basis vectors of an n-dimensional vector space over real numbers. In this interpretation, (3.3.1) is simply a vector having nonnegative coordinates that sum up to 1. Moreover, if some *operation* performed of the system causes transformation

$$[a_i] \mapsto p_{i1}[a_1] + p_{i2}[a_2] + \ldots + p_{in}[a_n]$$

for each $[a_i]$, we can express the operation on (3.3.1) as

$$\begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{pmatrix} \mapsto \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \cdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{pmatrix}.$$

Matrix P above has nonnegative entries and the sum of each row is 1. Such matrices are called *Markov* matrices.

3.4 Mathematical Background

The basic element for treating n-level quantum systems is an n-dimensional *Hilbert space.*

A Hilbert space H_n is an n-dimensional vector space over $\mathbb{C}$, the field of complex numbers, equipped with an inner product $H_n \times H_n \mapsto \mathbb{C}$ which satisfies the following axioms for each $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z} \in H_n$, and $\alpha, \beta \in \mathbb{C}$:

1. $\langle \boldsymbol{x} \,|\, \boldsymbol{y} \rangle = \langle \boldsymbol{y} \,|\, \boldsymbol{x} \rangle^*$.
2. $\langle \boldsymbol{x} \,|\, \boldsymbol{x} \rangle \geq 0$ and $\langle \boldsymbol{x} \,|\, \boldsymbol{x} \rangle = 0$ if and only if $\boldsymbol{x} = 0$.
3. $\langle \boldsymbol{x} \,|\, \alpha\boldsymbol{y} + \beta\boldsymbol{z} \rangle = \alpha\langle \boldsymbol{x} \,|\, \boldsymbol{y} \rangle + \beta\langle \boldsymbol{x} \,|\, \boldsymbol{z} \rangle$.

In the Axiom 1, w^* stands for the complex conjugate of a complex number w.

The inner product induces a *norm* in H_n by $\|\boldsymbol{x}\| = \sqrt{\langle \boldsymbol{x} \,|\, \boldsymbol{x} \rangle}$. As the norm is given, we can define the *distance* of two vectors as $d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|$.

A mapping $T : H_n \to H_n$ is said to be *linear*, if $T(\alpha\boldsymbol{x}+\beta\boldsymbol{y}) = \alpha T\boldsymbol{x}+\beta T\boldsymbol{y}$ for each $\boldsymbol{x}$, $\boldsymbol{y} \in H_n$ and α, $\beta \in \mathbb{C}$. For each linear mapping $T : H_n \to H_n$ there exists an *adjoint mapping* T^* satisfying $\langle \boldsymbol{x} \,|\, T\boldsymbol{y} \rangle = \langle T^*\boldsymbol{x} \,|\, \boldsymbol{y} \rangle$ for each $\boldsymbol{x}$, $\boldsymbol{y} \in H_n$. If $T = T^*$, we say that T is a *self-adjoint mapping* and if $T^* = T^{-1}$, then T is called *unitary.*

If $U : H_n \to H_n$ is unitary, then $\langle U\boldsymbol{x} \,|\, U\boldsymbol{y} \rangle = \langle U^*U\boldsymbol{x} \,|\, \boldsymbol{y} \rangle = \langle \boldsymbol{x} \,|\, \boldsymbol{y} \rangle$, which is to say that an application of a unitary mapping on two vectors cannot change their inner product. It follows that a unitary mapping preserves the norms, and also that all unitary mappings $H_n \to H_n$ are bijections.

3.4.1 Describing the Quantum States

The description we here use for an n-level quantum system is analogous to that one of a probabilistic system in Example 3.1. Assume that a quantum system has n pairwisely distinguishable states, denoted by $|\,a_1\rangle$, ..., $|\,a_n\rangle$ and called the *basis states.* [1]

We also establish an n dimensional vector space H_n over complex numbers having $\{|\,a_1\rangle, \dots, |\,a_n\rangle\}$ as an orthonormal basis. A fixed orthonormal basis is usually referred as to a a *computational basis.* Space H_n is called the *state space* of the systems.

A general state of an n-level quantum system is given as

$$c_1\,|\,a_1\rangle + \ldots + c_n\,|\,a_n\rangle, \tag{3.4.1}$$

where $|c_1|^2 + \ldots + |c_n|^2 = 1$. In other words, a state of an n-level quantum system is described as a unit-length vector in H_n. We say that (3.4.1) is a *superposition* of basis state $|\,a_1\rangle$, ..., $|\,a_n\rangle$ with *amplitudes* c_1, ..., c_n.[2]

The interpretation of (3.4.1) is as follows: when a quantum system in state (3.4.1) is observed, then, with a probability of $|c_i|^2$ we learn that the system is state a_n.

The following two definitions are important in quantum computing.

Definition 3.1. A *quantum bit* (qubit) is a two-level quantum system. It is traditional to denote the computational basis of a qubit by $|\,0\rangle$ a and $|\,1\rangle$. Notice that $|\,0\rangle$ does not refer to the zero vector, but to a unit-length vector associated to the logical zero.

A general state of a qubit is given as

$$c_0\,|\,0\rangle + c_1\,|\,1\rangle, \tag{3.4.2}$$

where $|c_0|^2 + |c_1|^2 = 1$. An observation of a qubit in state (3.4.2) will give logical 0 as an outcome with a probability of $|c_0|^2$, and logical 1 as an outcome with a probability of $|c_1|^2$.

Definition 3.2. A called a *quantum register* of length n is a system of n quantum bits. Such a system can be described by using 2^n-dimensional Hilbert space H_{2^n} as the state space. It is useful to denote the computational basis as

$$\{|\,\boldsymbol{x}\rangle \mid \boldsymbol{x} \in \{0,1\}^n\},$$

[1] Notation $|\,a_i\rangle$ is due to P. Dirac, and it is quite useful in some situations (see [12]), but here we use that notation only because of the tradition.

[2] In quantum mechanics, state (3.4.1) is not mixed as its probabilistic analogue in Example 3.1, but pure. Mixed quantum states are not handled in this thesis, their description can be found in [12].

i.e., the "labels" of the basis vectors are the bit strings of length n. A general state of an n-qubit system is depicted as

$$\sum_{\boldsymbol{x}\in\{0,1\}^n} c_{\boldsymbol{x}} \left|\boldsymbol{x}\right\rangle, \tag{3.4.3}$$

where

$$\sum_{\boldsymbol{x}\in\{0,1\}^n} |c_{\boldsymbol{x}}|^2 = 1.$$

An observation of the quantum register in state (3.4.3) will yield value $\boldsymbol{y} \in \{0,1\}^n$ with a probability $|c_{\boldsymbol{y}}|^2$.

Example 3.2. Consider a system of two qubits having H_4 as the state space. As in the above example, we choose $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ as an orthonormal basis of H_4. On the other hand, instead of H_4 we could consider the *tensor product* $H_2 \otimes H_2$, which is isomorphic to H_4. From our point of view, the tensor products will not be important, and therefore we will not pay much attention to them, but merely refer to [12] for a more precise treatment.

Here we will only say that states $|\,00\rangle$, $|\,01\rangle$, $|\,10\rangle$, and $|\,11\rangle$ can be written as *product states* $|0\rangle\,|0\rangle$, $|0\rangle\,|1\rangle$, $|1\rangle\,|0\rangle$, and $|1\rangle\,|1\rangle$, respectively. Moreover, if we have one qubit in state $a_0\,|0\rangle + a_1\,|1\rangle$, and another in state $b_0\,|0\rangle + b_1\,|1\rangle$, we can express their *compound state* as

$$\begin{aligned}
& (a_0\,|0\rangle + a_1\,|1\rangle)(b_0\,|0\rangle + b_1\,|1\rangle) \\
= \;& a_0b_0\,|0\rangle\,|0\rangle + a_0b_1\,|0\rangle\,|1\rangle + a_1b_0\,|1\rangle\,|0\rangle + a_1b_1\,|1\rangle\,|1\rangle \\
= \;& a_0b_0\,|00\rangle + a_0b_1\,|01\rangle + a_1b_0\,|10\rangle + a_1b_1\,|11\rangle.
\end{aligned}$$

A two-qubit state $c_0\,|00\rangle + c_1\,|01\rangle + c_2\,|10\rangle + c_3\,|11\rangle$ which can be written as a product of two one-qubit states, is called a *decomposable* state, otherwise we say that the state is *entangled*. For instance, state

$$\frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

can be written as

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle),$$

and hence it is decomposable, whereas state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ is entangled, as easily seen [12].

3.4.2 The Dynamics of the Quantum Systems

In order to handle quantum systems it is, of course, important to know how to describe those systems mathematically, but so far we have been dealing only with *instantaneous* descriptions. That is, we have not yet discussed about the *time evolution* of quantum systems. For a more detailed treatment, we refer to [12], but here we will only consider so-called *closed time evolutions*, which are treated analogously to Example 3.1.

Consider a quantum system having basis states $\mid a_1\rangle$, ..., $\mid a_n\rangle$ and an operation which transforms each basis state as

$$\mid a_i\rangle \mapsto c_{i1} \mid a_1\rangle + c_{i2} \mid a_2\rangle \dots c_{in} \mid a_n\rangle.$$

Then, the action of that operation on state (3.4.1) can be described as

$$\begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} \mapsto \begin{pmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2n} \\ \vdots & \vdots & \ddots & \dots \\ c_{n1} & c_{n2} & \dots & c_{nn} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix},$$

where the matrix C above preserves the norm in H_2. It can be shown that a matrix C preserves norm in H_2 if and only if C is unitary (see [12] for example).

Example 3.3. Let

$$W_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

W_2 is called *Hadamard-Walsh -transform.* Its operation on states $\mid 0\rangle$ and $\mid 1\rangle$ is easy to depict:

$$\begin{aligned} W_2 \mid 0\rangle &= \frac{1}{\sqrt{2}} \mid 0\rangle + \frac{1}{\sqrt{2}} \mid 1\rangle \\ W_2 \mid 1\rangle &= \frac{1}{\sqrt{2}} \mid 0\rangle - \frac{1}{\sqrt{2}} \mid 1\rangle. \end{aligned}$$

Example 3.4. Let

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

The operation of C on the basis vectors is then given by $C \mid 00\rangle = \mid 00\rangle$, $C \mid 01\rangle = \mid 01\rangle$, $C \mid 10\rangle = \mid 11\rangle$, and $C \mid 11\rangle = \mid 10\rangle$. That is, the second bit is flipped if and only if the first bit is set to one. Mapping C is called the *controlled not*-gate.

3.5 From Classical to Quantum Computing

In 1982, a famous physicist Richard Feynman suggested in his article [7] that it may be impossible to simulate a quantum mechanical system by an ordinary computer without an exponential slowdown in the simulation. He also proposed that this slowdown could be avoided if we were able to use a computer running according to the laws of quantum mechanics. As an implicit statement in that suggestion one can read that a *quantum computer* may be exponentially faster than any classical one.

The article of Feynman mentioned above can be seen as a starting point of the theory of quantum computing. In 1985, David Deutsch re-examined Church-Turing thesis, stating a physical version of it in his pioneering article [4]. In his article, Deutsch introduced the notion of a *quantum Turing machine* and even more importantly, the notion of a *universal quantum Turing machine*.

Despite of the two pioneering articles mentioned above, quantum computing remained rather a marginal issue in the theory of computing until 1994, when Peter Shor introduced his celebrated quantum algorithms for factoring integers and extracting discrete logarithms in polynomial time [24]. After Shor's work, quantum computing has been an intensively growing research area, and nowadays the quantum counterpart of theoretical computer science can be roughly divided at least into two subareas: *quantum computing* and *quantum information processing*.

It is of course somewhat artificial to try to draw boundaries restricting these two subareas, but at least some characteristic features can be mentioned. The research area of quantum information processing is mainly concentrating on quantum cryptography, quantum communication protocols, quantum error-correcting, and quantum teleportation, for instance. On the other hand, the area of quantum computing attempts to concentrate on quantum counterparts of the traditional computing devices, quantum algorithms, and on the complexity theory based on the quantum computational machines.

Formally, the way how a quantum counterpart of a classical (finite-state) computing machine can be obtained, is quite straightforward. First we have to think about all the potential configurations that the classical machine is able to possess, and then to establish a quantum physical representation for those configurations. Another thing to do is to find a suitable *dynamics* for the system: the classical computational operations must be replaced by their quantum counterparts, which, in the closed systems, means that a unitary time-evolution must be introduced.

Example 3.5. A *deterministic finite automaton* (DFA for short) over an alphabet Σ is a five-tuple $(Q, \Sigma, \delta, q_0, F)$, where Q is a (finite) set of *internal states*, Σ is the *alphabet*, $\delta : Q \times \Sigma \to Q$ is the *transition function*, $q_0 \in Q$ is the *initial state*, and $F \subseteq Q$ is the *set of accepting states*.

The *intended interpretation* of a deterministic finite automaton is that there is an *input word* $w \in \Sigma^*$, which is treated by the automaton in the way described below.

The transition function $\delta : Q \times \Sigma \to Q$ is first extended to a function $\delta : Q \times \Sigma^* \to Q$ as follows: for any word $w \in \Sigma^*$ and any state $q \in Q$ we define

$$\delta(q,w) = \begin{cases} q, & \text{if } w \text{ is the empty word} \\ \delta(\delta(q,a),w'), & \text{if } w = aw', \text{ where } a \in \Sigma. \end{cases}$$

By using this extension of δ, it is easy to define the *intended action* of the automaton: we say that the automaton *accepts* the word $w \in \Sigma^*$ if and only if $\delta(q_0, w) \in F$.

In order to make a quantum version of a DFA, we have first to consider which would be the underlying physical system representing the configurations of the automaton in question. In this case this is an easy task – if $Q = \{q_0, \ldots, q_{n-1}\}$, then we have to establish a physical system capable of representing n different states. Therefore, we will consider a quantum system having n basis states (vectors)

$$\{|\, q_0\rangle, \ldots, |\, q_{n-1}\rangle\}.$$

As usual, the vector space spanned by the above vectors will be denoted by H_n.

The next problem is to find the quantum counterpart for the dynamics. This is naturally resolved in by defining a unitary mapping $U_a : H_n \to H_n$ for each letter $a \in \Sigma$. Finally, the state reached by the automaton when the input is $w = a_1 \ldots a_k \in \Sigma^k$ is defined as

$$U_{a_k} \ldots U_{a_1} \,|\, q_0\rangle,$$

which is, in general, a superposition of all states $|\, q_0\rangle$, …, $|\, q_n\rangle$.

To transform a Turing machine into its quantum counterpart is not essentially more difficult than that one concerning finite automata – the only difference is that for Turing machines, there are infinitely many (but countably many) potential configurations. A somewhat more severe problem may be seen in the question associated to closed quantum systems in general: the time evolution is unitary, and therefore also invertible. On the other hand, computation can be irreversible, as well. The problem concerning irreversible computations has been accessed by Lecerf, who showed that any irreversible Turing machine can be simulated by a reversible one [18]. Later, Lecerf's result has been re-established by Bennett, who showed that a reversible simulation of an irreversible Turing machine can be done with a *constant slowdown* [2].

The simulation of an irreversible computation by a reversible one establishes the following fact: whatever is computable by a traditional computer, is also computable by a quantum computer.

3.6 Query Algorithms

The query algorithms are among the simplest algorithmic notions for computing functions defined on a Cartesian power of a finite set. It must be emphasized that the notion of a query algorithm studied here is equivalent to that of *decision tree*. However, in this thesis we choose to use the query algorithms because their extensions to probabilistic and quantum computing are relatively straightforward.

In this section we will represent the notions of deterministic, probabilistic, and quantum *query complexity*, and how these query complexities can be bounded below by using the *degree of* the function computed by query algorithms.

3.6.1 Deterministic Query Algorithms

Let A be a finite set of *variable values*, M a finite set called *memory* and V a finite set of *target values*.

Definition 3.3. A *deterministic query algorithm with k queries* is a $k+4$-tuple

$$(S, s_0, Q, C_0, C_1, \ldots, C_k),$$

where $S = \{1, \ldots, N\} \times A \times M \times V$ is the set of *internal states*, $s_0 = (i_0, a_0, m_0, v_0) \in S$ is the *initial state*, $Q = \{Q_{\boldsymbol{a}} \mid \boldsymbol{a} \in A^N\}$ is the set of *query operators*, and each C_i is a *computation operator* $C_i : S \to S$.

For each $(i, a, m, v) \in S$ and each $\boldsymbol{a} = (a_1, \ldots, a_N) \in A^N$ the query operator $Q_{\boldsymbol{a}}$ must satisfy the following:

$$Q_{\boldsymbol{a}}(i, a, m, v) = (i, a_i, m, v),$$

that is, when the query operator $Q_{\boldsymbol{a}}$ is applied to triple (i, a, m, v), it returns the ith coordinate a_i in the second component and leaves all other components untouched.

Definition 3.4. The *value* computed by the query algorithm with input $\boldsymbol{a}$ is v_k, where

$$(i_k, a_k, m_k, v_k) = C_k Q_{\boldsymbol{a}} \ldots Q_{\boldsymbol{a}} C_1 Q_{\boldsymbol{a}} C_0 s_0. \tag{3.6.1}$$

It is clear that each function $f : A^N \to V$ can be computed by a query algorithm. In fact, we can define $M = A^N$, $s_0 = (1, a_0, a_0^N, v_0)$ (a_0 and v_0 are arbitrary), C_0 as an identity operator, and, for $i \in \{1, \ldots, N\}$ C_i as

$$C_i(i, a, \boldsymbol{a}', v) = (i+1, a, \boldsymbol{a}'', v),$$

where $\boldsymbol{a}''$ is $\boldsymbol{a}'$ having the ith coordinate replaced by a. That is, C_i writes the value a called upon by the previous query into the ith memory coordinate and increases the coordinate number by one. Finally, we define C_N as

$$C_N(i, a, \boldsymbol{a}, v) = (i, a, \boldsymbol{a}, f(\boldsymbol{a})),$$

which simply means that C_N writes the correct function value into the last coordinate.

Definition 3.5. The (deterministic) *query complexity* of a function $f : A^N \to V$ is defined to be the minimum number of queries of a query algorithm computing f and denoted by $D(f)$.

As we saw above, $D(f) \leq N$ for each function $f : A^N \to V$. Let us now assume that $|A| = 2$. Then we usually fix $A = \mathbb{F}_2$ to be the *binary field*. If we also assume that $V = \mathbb{C}$, the field of complex numbers, we can establish a *polynomial representation* for any function $f : \mathbb{F}_2^N \to \mathbb{C}$. In this section, we will refer to this concept only informally, a more precise study is included in the next chapter. The only necessary thing we need to know here is that monomial $X_i : \mathbb{F}_2^N \to \mathbb{C}$ is defined as a function which has value $1 \in \mathbb{C}$, if $x_i = 1 \in \mathbb{F}_2$ and 0 otherwise. Monomials consisting of several variables and polynomials are built in a natural way. We also emphasize that it turns out that each function $f : \mathbb{F}_2^N \to \mathbb{C}$ can be uniquely written as a polynomial on N variables $X_1, \ldots, X_N$ having degree at most N. We define $\deg_P(f)$ as the degree of the polynomial which represents function f.

The following proposition is well-known, but, for the sake of completeness, we represent its proof.

Proposition 3.1. *Let $f : \mathbb{F}_2^N \to \mathbb{C}$ be a function. Then $\deg_P(f) \leq D(f)$.*

Proof. Assume that T is a query algorithm which computes function f on some input $\boldsymbol{x} = (x_1, \ldots, x_N)$ and has complexity $k = D(f)$. Let us denote $s_1 = C_0 s_0 = (i_1, a_1, m_1, v_1)$. Applying the query operator $Q_{\boldsymbol{x}}$ on s_1 results in state $(i_1, 1, m_1, v_1)$ if $\boldsymbol{x}_{i_1} = 1$, and in state $(i_1, 0, m_1, v_1)$ if $\boldsymbol{x}_{i_1} = 0$. We can now symbolically write the resulting state as

$$X_{i_1}(i_1, 1, m_1, v_1) + (1 - X_{i_1})(i_1, 0, m_1, v_1). \tag{3.6.2}$$

Applying C_1 results in state

$$\begin{aligned} & X_{i_1} C_1(i_1, 1, m_1, v_1) + (1 - X_{i_1}) C_1(i_1, 0, m_1, v_1) \\ = \;& X_{i_1}(i_2, a_2, m_2, v_2) + (1 - X_{i_1})(i_2', a_2', m_2', v_2'). \end{aligned}$$

It is now clear that polynomial

$$X_{i_1} v_2 + (1 - X_{i_1}) v_2'$$

of degree at most one represents the function computed by the query algorithm having only two computation operators C_0 and C_1 (and making only one query). Assume then that if the query algorithm makes $l < k$ queries, the corresponding function can be represented by a polynomial having degree at most l. Especially, algorithm with initial state $(i_1, 0, m_1, v_1)$ (resp.

$(i_1, 1, m_1, v_1))$ with operators C_1, C_2, …, C_k makes $k-1$ queries and hence the corresponding function is represented by some polynomial $P_0(X)$ (resp. $P_1(X)$) on variables X_1, …, X_N of degree at most $k-1$. It follows that the polynomial

$$X_{i_1}P_1(X) + (1 - X_{i_1})P_0(X)$$

has degree at most k and represents the function computed by the query algorithm. □

In this thesis, we are basically interested in decision trees which compute functions $\mathbb{F}_2^N \to \mathbb{C}$ having only *two* potential values. Such functions are called *Boolean functions*.

Remark 3.1. It has been show that $D(f) \leq 2\deg(f)^4$ holds for any Boolean function f [28]. Hence $D(f)$ and $\deg(f)$ can be only polynomially apart from each other, if f is Boolean.

Example 3.6. (Nisan & Szegedy [21]) Let $E_1 : \{0,1\}^3 \to \{0,1\}$ be defined as

$$E_1(x_1, x_2, x_3) = \begin{cases} 0, & \text{if } x_1 = x_2 = x_3 = 0 \text{ or } x_1 = x_2 = x_3 = 1, \\ 1, & \text{otherwise} \end{cases}$$

Then E_1 can be represented as

$$E_1(x_1, x_2, x_3) = x_1 + x_2 + x_3 - x_1x_2 - x_1x_3 - x_2x_3.$$

Clearly $\deg(E_1) = 2$, but it is also easy to see that $D(f) = 3$. If $k > 1$, we define function E_k on 3^k variables recursively as

$$E_k(X_1, X_2, X_3) = E_1(E_{k-1}(X_1), E_{k-1}(X_2), E_{k-1}(X_3)),$$

where X_1, X_2, and X_3 are vectors of 3^{k-1} disjoint variables. It is easy to see that $\deg(E_k) = 2^k$, whereas $D(f) = 3^k$. Denoting $N = 3^k$ we have $\deg(E_k) = N^{\log_3 2} = N^{0.63\ldots}$.

We say that a function $f : \mathbb{F}_2^N \to V$ *depends* on variable X_i, if there exists a vector $\boldsymbol{x} \in \mathbb{F}_2^N$ such that $f(\boldsymbol{x}) \neq f(\boldsymbol{x}^{(i)})$, where $\boldsymbol{x}^{(i)}$ is obtained from $\boldsymbol{x}$ by flipping its ith coordinate. If a function $f : \mathbb{F}_2^N \to V$ depends only on $k < N$ variables, we say that f is *degenerate*. If f depends on all its variables, then f is *nondegenerate*. If f is degenerate, it is clear that $D(f) < N$.

Proposition 3.2. *Let $f : \mathbb{F}_2^N \to V$ be a nondegenerate function. Then $D(f) \geq \log_2(N+1)$.*

Proof. Let T be a query algorithm computing f with $k = D(f)$ queries. At each query, the computation of T splits into at most two branches, and it follows that there are at most $2^0 + 2^1 + 2^2 + \ldots + 2^{k-1} = 2^k - 1$ variables that are queried in the computations. Since f depends on all its variables X_1, …, X_N, each variable must be queried at least once in some computation. Hence $2^k - 1 \geq N$, and the claim follows. □

Example 3.7. (Simon [26]) Let $k \geq 1$ and $f : \{0,1\}^k \times \{0,1\}^{2^k} \to \{0,1\}$ be defined as follows: For each $\boldsymbol{y} \in \{0,1\}^{2^k}$, we label the coordinates of $\boldsymbol{y} = (y_0, y_1, \ldots, y_{2^k-1})$ and define $n(\boldsymbol{x}) \in \{0,1,\ldots,2^k-1\}$ be the number represented by the binary string $\boldsymbol{x} \in \{0,1\}^k$. We define

$$f(\boldsymbol{x}, \boldsymbol{y}) = y_{n(\boldsymbol{x})},$$

It is easy to see that f depends on all its $N = k + 2^k$ input variables.

On the other hand, it is easy to see that f can be computed by a query algorithm making only $k+1$ queries: It suffices to query all the coordinates of $\boldsymbol{x}$, and then the coordinate of $\boldsymbol{y}$ referred to by $\boldsymbol{x}$. Hence $D(f) \leq k+1$ and

$$\log_2 N = k + \log_2(1 + \frac{k}{2^k}) \geq D(f) - 1,$$

so $D(f) \leq \log_2 N + 1$, quite sharply matching the lower bound $D(f) \geq \log_2(N+1)$.

3.6.2 Probabilistic Query Algorithms

In this section, we concentrate only on decision trees computing functions $f : \mathbb{F}_2^N \to S$, where $S \subseteq \mathbb{C}$ is a set of two elements. Typically we choose either $S = \{0,1\}$ or $S = \{-1,+1\}$.

Let A, M, and V be as in the definition of the deterministic query algorithm.

Definition 3.6. A *probabilistic query algorithm on N variables with k queries* is a $k+4$-tuple

$$(S, s_0, Q, P_0, P_1, \ldots, P_k),$$

where S, s_0, and Q are defined exactly in the same way as in the deterministic case, and each P_i is a *probabilistic computation operator* associating to each state $\boldsymbol{s} \in S$ a probability distribution over the states.

Let $d = |S|$. In order to handle the computations of probabilistic query algorithms, we will define a d-dimensional vector space $\mathcal{P}$ over real numbers with basis S. Hence any vector $\boldsymbol{v} \in \mathcal{P}$ can be represented as

$$\boldsymbol{v} = \sum_{s \in S} p_s \boldsymbol{s}, \tag{3.6.3}$$

where $p_s \in \mathbb{R}$. If $\sum_{s \in S} p_s = 1$ and $p_s \geq 0$ for each $s \in S$, we say that (3.6.3) is a *convex combination* of basis vectors $\boldsymbol{s}$.

A more precise definition for probabilistic computation operators P_i can now be obtained by requiring that each $P_i : \mathcal{P} \to \mathcal{P}$ is a linear mapping which maps a convex combination of basis vectors into another convex combination.

The matrices of such mappings are called *Markov matrices*. Moreover, we require that each query operator $Q_{\boldsymbol{a}} : \mathcal{P} \to \mathcal{P}$ is a linear operator defined as

$$Q_{\boldsymbol{a}}(i, a, m, v) = (i, a_i, m, v).$$

Our first intention is to view a probabilistic query algorithm as a device for computing a probability distribution on set V, and that will be achieved as follows: By the assumption on mappings P_i we see that

$$\boldsymbol{v}_k = P_k Q_{\boldsymbol{a}} P_{k-1} Q_{\boldsymbol{a}} \ldots Q_{\boldsymbol{a}} P_0 s_0$$

is a convex combination on basis vectors S. It follows that $\boldsymbol{v}_k$ can be represented as

$$\boldsymbol{v}_k = \sum_{s \in S} p_s \boldsymbol{s}, \tag{3.6.4}$$

where $\sum_{s \in S} p_s = 1$ and $p_s \geq 0$ for each $s \in S$. By using (3.6.4), we define the distribution computed by the probabilistic query algorithm as

$$P(v_i) = \sum_{\substack{(i,b,m,v) \in S \\ v = v_i}} p_{(i,b,m,v)}.$$

Analogously to Proposition 3.1 we can get the following proposition:

Proposition 3.3. *Let P be a query algorithm which makes k queries to input variable $\boldsymbol{x} \in \mathbb{F}_2^N$. Then the probability $P(v_i)$ of having v_i as outcome can be represented as a polynomial $P_{v_i}(X)$ of degree at most k. Moreover,*

$$\sum_{v \in V} P_v(X) = 1$$

identically.

Assume now that $V = \{0, 1\}$ is a set of two elements, and that the query algorithm for computing function $f : \mathbb{F}_2^N \to \{0, 1\}$ produces a correct output with a probability of at least $\frac{2}{3}$. Then $P_1(\boldsymbol{x}) \geq \frac{2}{3}$ whenever $f(\boldsymbol{x}) = 1$, and $P_1(\boldsymbol{x}) \leq \frac{1}{3}$ if $f(\boldsymbol{x}) = 0$. This means that

$$|P_1(\boldsymbol{x}) - f(\boldsymbol{x})| \leq \frac{1}{3}$$

holds for any $\boldsymbol{x} \in \mathbb{F}_2^N$, which is to say that polynomial P_1 *approximates* function f within threshold $\frac{1}{3}$. On the other hand, if the query algorithm makes k queries, we know, by the previous proposition, that $\deg(P_1) \leq k$.

Definition 3.7. The probabilistic query complexity $R_\epsilon(f)$ is the minimal number of queries that a probabilistic query algorithm makes to compute a Boolean function f in such a way that the probability of an erratic answer is at most ϵ.

Definition 3.8. The ϵ-*approximation degree* $\widetilde{\deg}_\epsilon(f)$ of a function $f : \mathbb{F}_2^N \to \mathbb{C}$ is defined as the minimum degree of a polynomial $P : \mathbb{F}_2^N \to \mathbb{C}$ which satisfies

$$|f(\boldsymbol{x}) - P(\boldsymbol{x})| \leq \epsilon$$

for each $\boldsymbol{x} \in \mathbb{F}_2^N$.

By the previous considerations, the below proposition is evident.

Proposition 3.4. $\widetilde{\deg}_\epsilon(f) \leq R_\epsilon(f)$.

3.6.3 Quantum Query Algorithms

A *quantum query algorithm* is a quantum counterpart of the concept of a deterministic, as well as a probabilistic query algorithm. For the sake of simplicity, we will concentrate only on functions $f : \mathbb{F}_2^N \to \{0,1\}$ in this section.

Definition 3.9. Let $A = \{0,1\}$, M a finite *memory set*, and $V = \{0,1\}$ the set of *function values*. A *quantum query algorithm on N variables with k queries* is a $k+4$-tuple

$$(S, s_0, Q, U_0, U_1, \ldots, U_k),$$

where $S = \{1,\ldots,N\} \times A \times M \times V$ and s_0 are the same as in the deterministic and probabilistic case, and each U_i is a *unitary computation operator*.

Let $d = |S|$. In order to formulate quantum query algorithms precisely, we will use a d-dimensional Hilbert space H_d, whose basis vectors will be denoted by $|\, i, a, m, v\rangle$. We also write $|\, s_0\rangle$ for the initial state of the query algorithm.

We will also make the query operators reversible, which is guaranteed if we define $Q_{\boldsymbol{a}}$ for each $\boldsymbol{a} \in A^N$ to be a linear operator $Q_{\boldsymbol{a}} : H_d \to H_d$ defined by

$$Q_{\boldsymbol{a}} \,|\, i, a, m, v\rangle = |\, i, a \oplus a_i, m, v\rangle,$$

where $a \oplus a_i$ means the addition modulo 2. Clearly $Q_{\boldsymbol{a}}$ is invertible, and it is also easy to see that $Q_{\boldsymbol{a}} : H_d \to H_d$ is unitary.

The fact that each $U_i : H_d \to H_d$ is a unitary computation operator means that each mapping U_i preserves the norm in H_d. It follows that vector

$$|\, v_k\rangle = U_k Q_{\boldsymbol{a}} U_{k-1} \ldots U_1 Q_{\boldsymbol{a}} U_0 \,|\, s_0\rangle$$

has unit length, which means that $|\, v_k\rangle$ can be written as

$$|\, v_k\rangle = \sum_{s \in S} c_s \,|\, s\rangle,$$

where $\sum_{s\in S} |c_s|^2 = 1$.

Analogously to Proposition 3.1 we can get the following proposition, whose proof has been introduced in [1].

Proposition 3.5. *Let Q be a quantum query algorithm making k queries on input $\boldsymbol{x} = (x_1, \ldots, x_N)$. Then the final state $|v_k\rangle$ can be written as*

$$|v_k\rangle = \sum_{s \in S} P_s(X)\, |s\rangle,$$

where

$$\sum_{s \in S} |P_s(X)|^2 = 1$$

identically, and each $P_s(X)$ is a polynomial having degree at most k.

As the probabilistic query algorithms, quantum algorithms also compute probability distributions on potential outputs $\{0, 1\}$: The probability that 1 is seen as outcome is given by

$$P_1(X) = \sum_{\substack{(i,a,m,v) \\ v=1}} \left|P_{(i,a,m,v)}(X)\right|^2,$$

and the probability that 0 is the outcome is

$$P_0(X) = \sum_{\substack{(i,a,m,v) \\ v=0}} \left|P_{(i,a,m,v)}(X)\right|^2.$$

By the previous proposition it follows that $P_1(X)$ and $P_0(X)$ are polynomials with real coefficients having degree at most $2k$.

Definition 3.10. We define $Q_\epsilon(f)$ to be the minimum number of queries that a quantum query algorithm computing f within threshold ϵ makes.

The following proposition is an easy consequence of the above considerations.

Proposition 3.6. $Q_\epsilon(f) \geq \frac{1}{2}\widetilde{\deg}_\epsilon(f)$.

By the above proposition, we can obtain lower bounds for quantum query complexity by finding lower bounds for the approximation degrees of functions $f : \mathbb{F}_2^N \to \{0, 1\}$.

The examples below have been discovered in [1] and [6]. Later we will introduce new machinery to obtain these results.

Example 3.8. Function OR on N variables defined as $OR(x_1, \ldots, x_N) = 0$ if and only if $x_i = 0$ for each i has representation degree N. On the other hand, we will see later that it has approximation degree $\Omega(\sqrt{N})$ (the constant included in the Ω-notation depends on the threshold ϵ), which shows, by the above proposition, that a quantum query algorithm computing OR must make $\Omega(\sqrt{N})$ queries. The upper bound is known: Lov Grover has devised an algorithm which can compute OR-function by using $O(\sqrt{N})$ queries [8].

Example 3.9. The parity function $f : \mathbb{F}_2^N \to \mathbb{F}_2$ defined as $f(x_1, \ldots, x_N) = x_1 + \ldots + x_N$ has representation degree N. Later we will see that for each positive threshold ϵ, its approximation degree is also N. Therefore, a quantum query algorithm computing parity must make at least $N/2$ queries even if some error probability is allowed. This is also known to be a matching upper bound [6].

Chapter 4

Function Space V_N – Basic Properties

In this chapter, we study some basic properties of Boolean functions.

4.1 Notations and Terminology

Notation $\mathbb{F}_2 = \{0,1\}$ will stand for the *binary field* having two elements with addition and multiplication defined obviously but $1+1=0$. An N-dimensional vector space over $\mathbb{F}_2$ is naturally denoted by $\mathbb{F}_2^N$. The cardinality of a set A is denoted by $|A|$.

As a *Boolean function* on N variables we understand a function from $\mathbb{F}_2^N$ into a set of two elements. A very natural (and also useful, when thinking about the compositions) choice for the target set would be $\mathbb{F}_2$ itself, but in order to smoothen the notations in mathematical treatments, we sometimes choose the target set to be $\{-1,1\} \subseteq \mathbb{C}$ or $\{0,1\} \subseteq \mathbb{C}$. Furthermore, to introduce the machinery of Fourier analysis, we study functions $f : \mathbb{F}_2^N \to \mathbb{C}$ in general.

Definition 4.1. Notation V_N stands for the set of all functions $f : \mathbb{F}_2^N \to \mathbb{C}$, as well as for the vector space consisting of these functions with addition and scalar product defined pointwise.

For any vector $\boldsymbol{x} = (x_1, \ldots, x_N) \in \mathbb{F}_2^N$ we define the *support* of $\boldsymbol{x}$ to be the set of indices i such that $x_i = 1$. Formally speaking,

$$\operatorname{supp}(\boldsymbol{x}) = \{i \in \{1, \ldots, N\} \mid x_i = 1\} \subseteq \{1, 2, \ldots, N\}.$$

The *Hamming weight* of $\boldsymbol{x}$ is the number of coordinates which equal to one;

$$\operatorname{wt}(\boldsymbol{x}) = |\operatorname{supp}(\boldsymbol{x})|\,.$$

For $i \in \{1, \ldots, N\}$, we also define vectors $\boldsymbol{e}_i \in \mathbb{F}_2^N$ such that $\operatorname{supp}(\boldsymbol{e}_i) = \{i\}$. Vectors $\boldsymbol{e}_i$ form the basis of $\mathbb{F}_2^N$, so called *natural basis*. The terminology "natural basis" is easily justified: for any $\boldsymbol{x} = (x_1, \ldots, x_N) \in \mathbb{F}_2^N$, it is easy to see that $\boldsymbol{x} = x_1\boldsymbol{e}_1 + \ldots + x_N\boldsymbol{e}_N$.

The rth *Hamming sphere* $S_r^{(N)}$ is defined as

$$S_r^{(N)} = \{\boldsymbol{x} \in \mathbb{F}_2^N \mid \mathrm{wt}(\boldsymbol{x}) = r\}.$$

If there is no danger of confusion, we omit the superscript N, and use notation S_r instead of $S_r^{(N)}$. Defining the addition and scalar multiplication pointwise, we can give a vector space structure for the set V_N of functions $f : \mathbb{F}_2^N \to \mathbb{C}$.

Functions $T_{\boldsymbol{y}} : \mathbb{F}_2^N \to \mathbb{C}$ defined by

$$T_{\boldsymbol{y}}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{x} = \boldsymbol{y} \\ 0, & \text{otherwise.} \end{cases} \tag{4.1.1}$$

form a basis, the natural basis of V_N. Notice that

$$f = \sum_{\boldsymbol{y} \in \mathbb{F}_2^N} f(\boldsymbol{y}) T_{\boldsymbol{y}},$$

which is to say that the coordinates of the function f with respect to the natural basis are in fact the values of f. Hence V_N is a 2^N-dimensional complex vector space and therefore isomorphic to $\mathbb{C}^{2^N}$. Space V_N can be equipped with the standard inner product defined by

$$\langle f \,|\, g \rangle = \sum_{\boldsymbol{x} \in \mathbb{F}_2^N} f(\boldsymbol{x})^* g(\boldsymbol{x}).$$

Clearly, the natural basis is orthonormal with respect to the above inner product.

Ignoring the scalar multiplication, V_N can also be viewed as an Abelian group, thus we can also find out the *characters* of V_N [1] It turns out that for each $\boldsymbol{y} \in \mathbb{F}_2^N$ there is a character $\chi_{\boldsymbol{y}}$ defined by

$$\chi_{\boldsymbol{y}}(\boldsymbol{x}) = (-1)^{\boldsymbol{x} \cdot \boldsymbol{y}}, \tag{4.1.2}$$

where $\boldsymbol{x} \cdot \boldsymbol{y} = x_1 y_1 + \ldots + x_N y_N$ is computed in $\mathbb{F}_2$ and $(-1)^b$ for $b \in \mathbb{F}_2$ is interpreted in the most obvious way. Moreover, it turns out that all of the characters of V_N are of form (4.1.2) [12]. It is also well known that the characters are orthogonal:

$$\langle \chi_{\boldsymbol{x}} \,|\, \chi_{\boldsymbol{y}} \rangle = \begin{cases} 2^N, & \text{if } \boldsymbol{x} = \boldsymbol{y} \\ 0, & \text{otherwise.} \end{cases}$$

[1] The characters χ are mappings $\mathbb{F}_2^N \to \mathbb{C}$ satisfying $\chi(\boldsymbol{x} + \boldsymbol{z}) = \chi(\boldsymbol{x})\chi(\boldsymbol{z})$.

By renormalizing the characters we get so-called *Walsh functions*. For each $\boldsymbol{y} \in \mathbb{F}_2^N$ we define

$$W_{\boldsymbol{y}} = \frac{1}{\sqrt{2^N}} \chi_{\boldsymbol{y}},$$

hence getting another orthonormal basis for V_N consisting of the Walsh functions. Evidently, the Walsh functions are symmetric with respect to the argument and the index: $W_{\boldsymbol{y}}(\boldsymbol{x}) = W_{\boldsymbol{x}}(\boldsymbol{y})$.

4.1.1 Fourier Representation

Each function $f \in V_N$ has a representation in natural basis:

$$f = \sum_{\boldsymbol{y} \in \mathbb{F}_2^N} f(\boldsymbol{y}) T_{\boldsymbol{y}},$$

as well as in the basis consisting of the Walsh functions:

$$f = \sum_{\boldsymbol{y} \in \mathbb{F}_2^N} \widehat{f}(\boldsymbol{y}) W_{\boldsymbol{y}}. \tag{4.1.3}$$

Representation (4.1.3) will be called the *Fourier representation* of function $f \in V_N$. The coefficients in the Fourier representation associate for each $\boldsymbol{y} \in \mathbb{F}_2^N$ a complex number $\widehat{f}(\boldsymbol{y})$, which is to say that the coefficients in representation (4.1.3) also determine a function $\widehat{f} : \mathbb{F}_2^N \to \mathbb{C}$. This function is called *the Fourier transform of* f. To compute the inner product $\langle W_{\boldsymbol{x}} \,|\, f \rangle$ by using representation (4.1.3) is to find out, by recalling that the Walsh functions are orthonormal, that

$$\langle W_{\boldsymbol{x}} \,|\, f \rangle = \widehat{f}(\boldsymbol{x}),$$

which can also be written as (the values of Walsh functions are always real)

$$\widehat{f}(\boldsymbol{y}) = \langle W_{\boldsymbol{y}} \,|\, f \rangle = \sum_{\boldsymbol{x} \in \mathbb{F}_2^N} f(\boldsymbol{x}) W_{\boldsymbol{y}}(\boldsymbol{x}) = \sum_{\boldsymbol{x} \in \mathbb{F}_2^N} f(\boldsymbol{x}) W_{\boldsymbol{x}}(\boldsymbol{y}). \tag{4.1.4}$$

Equations (4.1.3) and (4.1.4) reveal an interesting fact: representations of f and $\widehat{f}$ with respect to the Walsh function basis are perfectly symmetric. It follows also that $\widehat{\widehat{f}} = f$.

More interesting and important facts are also easily recoverable: by the very definition of the inner product we have that

$$\begin{aligned} &\sum_{\boldsymbol{x} \in \mathbb{F}_2^N} |f(\boldsymbol{x})|^2 = \langle f \,|\, f \rangle \\ = \;& \langle \sum_{\boldsymbol{y} \in \mathbb{F}_2^N} \widehat{f}(\boldsymbol{y}) W_{\boldsymbol{y}} \,|\, \sum_{\boldsymbol{z} \in \mathbb{F}_2^N} \widehat{f}(\boldsymbol{z}) W_{\boldsymbol{z}} \rangle = \sum_{\boldsymbol{y} \in \mathbb{F}_2^N} \left| \widehat{f}(\boldsymbol{y}) \right|^2 . \end{aligned}$$

Equation

$$\sum_{\boldsymbol{x}\in\mathbb{F}_2^N} |f(\boldsymbol{x})|^2 = \sum_{\boldsymbol{y}\in\mathbb{F}_2^N} \left|\widehat{f}(\boldsymbol{y})\right|^2. \tag{4.1.5}$$

is known as the *Parseval's identity.*

If $f \in V_N$ has a representation

$$f = \sum_{\boldsymbol{y}\in\mathbb{F}_2^N} \widehat{f}(\boldsymbol{y}) W_{\boldsymbol{y}}, \tag{4.1.6}$$

we define the *Fourier degree* of f to be the maximal Hamming weight of such $\boldsymbol{y}$ for which $\widehat{f}(\boldsymbol{y}) \neq 0$. In symbols,

$$\deg_F(f) = \max\{\mathrm{wt}(\boldsymbol{y}) \mid \widehat{f}(\boldsymbol{y}) \neq 0\}.$$

4.1.2 Polynomial Representation

Let $S \subseteq \{1, 2, \ldots, N\}$, $\boldsymbol{x} = (x_1, \ldots, x_N)$, and $X_S : \mathbb{F}_2^N \to \mathbb{C}$ be defined by

$$X_S(\boldsymbol{x}) = \begin{cases} 1, & \text{if } x_i = 1 \text{ for each } i \in S, \\ 0, & \text{otherwise.} \end{cases} \tag{4.1.7}$$

It can be shown that the function X_S also form a basis of V_N [12]. Thus all functions $f : \mathbb{F}_2^N \to \mathbb{C}$ can also be uniquely represented as

$$f = \sum_{S\subseteq\{1,\ldots,N\}} c_S X_S. \tag{4.1.8}$$

Functions X_S are called *monomials*, and representation (4.1.8) is referred as to the *polynomial representation*. Especially, when $S = \{i\}$ is a singleton, we denote $X_S = X_i$, and have that $X_i(\boldsymbol{x}) = x_i$, when $x_i \in \mathbb{F}_2$ is interpreted as a real number in the most obvious way. Thus we can represent each X_S as

$$X_S = \prod_{i\in S} X_i.$$

The *polynomial degree* of function X_S is defined to be $\deg_P(X_S) = |S|$, and the polynomial degree of function f, $\deg_P(f)$ is defined to be the maximal degree of a monomial occurring in representation (4.1.8).

Proposition 4.1. *For any function $f \in V_N$, $\deg_P(f) = \deg_F(f)$.*

Proof. Let $S = \{i_1, \ldots, i_d\}$ be the set of nonzero coordinates of $\boldsymbol{y}$. Then clearly

$$\chi_{\boldsymbol{y}} = \prod_{i\in S}(1 - 2X_i),$$

which shows that that $\deg_P(\chi_{\boldsymbol{y}}) = d = \deg_F(\chi_{\boldsymbol{y}})$. Thus, for each $f \in V_N$, $\deg_P(f) \leq \deg_F(f)$.

On the other hand, if $S = \{i_1, \ldots, i_d\}$, we can express the monomial X_S of degree d as

$$X_S = \frac{1}{2}(1 - \chi_{\boldsymbol{e}_{i_1}}) \cdot \ldots \cdot \frac{1}{2}(1 - \chi_{\boldsymbol{e}_{i_d}}), \tag{4.1.9}$$

where $\boldsymbol{e}_i \in \mathbb{F}_2^N$ stands for the vector which has 1 in the ith coordinate and 0 everywhere else. Since $\chi_{\boldsymbol{e}_i} \cdot \chi_{\boldsymbol{e}_j} = \chi_{\boldsymbol{e}_i + \boldsymbol{e}_j}$, expanding (4.1.9) we see that $\deg_F(X_S) \leq d = \deg_P(X_S)$, which implies that $\deg_F(f) \leq \deg_P(f)$ for any function $f \in V_N$. □

By the above proposition, we can give the following definition.

Definition 4.2. The *degree* of a function $f : \mathbb{F}_2^N \to \mathbb{C}$ which is not identically zero, is defined by $\deg(f) = \deg_F(f) = \deg_P(f)$. The degree of $f \equiv 0$ is symbolically defined to be $-\infty$.

Notice that for any $f : \mathbb{F}_2^N \to \mathbb{C}$ we have that $\deg(f) \leq N$.

4.2 Zeros of Functions

It is a well-known fact that a non-constant polynomial p with complex coefficients having a degree d can have at most d distinct zeros. Here we will represent the counterpart of this fact for functions $\mathbb{F}_2^N \to \mathbb{C}$, known as *Schwartz' lemma* [25].

Lemma 4.1 (Schwartz' lemma). *Let $f : \mathbb{F}_2^N \to \mathbb{C}$ be a nonzero function having degree d (see Definition 4.2). Then*

$$\left|\{\boldsymbol{x} \in \mathbb{F}_2^N \mid f(\boldsymbol{x}) = 0\}\right| \leq 2^N - 2^{N-d}.$$

Proof. If $N = 1$, then necessarily $\deg(f) = 1$ (because f is not a constant function) and f has at most $1 = 2^1 - 2^{1-1}$ zeros.

Assume then that $N > 1$ and that the claim holds for spaces $\mathbb{F}_2^{N'}$, where $N' < N$. Obviously we can write the Fourier representation of f as $f = g + (-1)^{x_1} h$, where g and h do not depend on x_1, and $\deg(g) \leq d$, $\deg(h) \leq d - 1$. Since g and h do not depend on x_1, we can regard them as functions $\mathbb{F}_2^{N-1} \to \mathbb{C}$, and apply the induction hypothesis. We have now three cases:

1. Substitution $x_1 = 0$ makes f identically zero. In this case, substitution $x_1 = 1$ cannot make f identically zero (because f is not identically zero). Now $g = -h$ which means that $f = ((-1)^{x_1} - 1)h$ and therefore f can have at most $2^{N-1} + 2^{N-1} - 2^{N-1-(d-1)} = 2^N - 2^{N-d}$ zeros.

2. The case that substitution $x_1 = 1$ makes f identically zero is analogous to the above case.

3. Both $f_0 = g + h$ and $f_1 = g - h$ are not identically zero. Then f can have at most $2^{N-1} - 2^{N-1-d} + 2^{N-1} - 2^{N-1-d} = 2^N - 2^{N-d}$ zeros.

$\square$

Unfortunately, the above lemma is generally too weak to provide us any good estimations for the degrees of the functions in V_N. However, some specific results can be obtained.

Corollary 4.1. *The natural basis functions (see (4.1.1)) $T_{\boldsymbol{y}} : \mathbb{F}_2^N \to \mathbb{C}$ have degree N.*

Proof. Each function $T_{\boldsymbol{y}}$ has $2^N - 1$ zeros, but the equation $2^N - 1 \leq 2^N - 2^{N-d}$ can be satisfied only if $d = N$. $\square$

Example 4.1. let $S = \{1, 2, \ldots, d\}$ and consider a function X_S (see (4.1.7)). Function X_S has degree d and is nonzero if and only if $x_1 = \ldots = x_d = 1$. Therefore, X_S has 2^{N-d} nonzeros and $2^N - 2^{N-d}$ zeros. However, if $d < N$, X_S is a *degenerate* function in the sense that it actually does not depend on *all* variables $x_1, \ldots x_N$.

4.3 Discrete Derivatives

For any $f \in V_N$ and $\boldsymbol{h} \in \mathbb{F}_2^N$, we define the *discrete derivative* (afterwards called merely *derivative*) of f onto direction $\boldsymbol{h}$ by

$$\Delta_{\boldsymbol{h}} f(\boldsymbol{x}) = f(\boldsymbol{x} + \boldsymbol{h}) - f(\boldsymbol{x}). \tag{4.3.1}$$

4.3.1 Basic Properties

By the definition (4.3.1) it is easy to see that the operator $\Delta_{\boldsymbol{h}}$ transforms a function $f \in V_N$ into another function $\Delta_{\boldsymbol{h}} f \in V_N$ *linearly*, that is,

$$\Delta_{\boldsymbol{h}}(\alpha f + \beta g) = \alpha \Delta_{\boldsymbol{h}} f + \beta \Delta_{\boldsymbol{h}} g$$

for any $\alpha, \beta \in \mathbb{C}$ and $f, g \in V_N$.

An easily verifiable property of the discrete derivative is given in the following lemma.

Lemma 4.2. *If $\boldsymbol{h}$, $\boldsymbol{g} \in \mathbb{F}_2^N$, then*

$$\Delta_{\boldsymbol{h}+\boldsymbol{g}} f(\boldsymbol{x}) = \Delta_{\boldsymbol{h}} \Delta_{\boldsymbol{g}} f(\boldsymbol{x}) + \Delta_{\boldsymbol{h}} f(\boldsymbol{x}) + \Delta_{\boldsymbol{g}} f(\boldsymbol{x}).$$

Proof. A straightforward computation gives that

$$\begin{aligned}
& \Delta_{\boldsymbol{h}} \Delta_{\boldsymbol{g}} f(\boldsymbol{x}) + \Delta_{\boldsymbol{h}} f(\boldsymbol{x}) + \Delta_{\boldsymbol{g}} f(\boldsymbol{x}) \\
= \; & \Delta_{\boldsymbol{g}} f(\boldsymbol{x} + \boldsymbol{h}) - \Delta_{\boldsymbol{g}} f(\boldsymbol{x}) + f(\boldsymbol{x} + \boldsymbol{h}) - f(\boldsymbol{x}) + \Delta_{\boldsymbol{g}} f(\boldsymbol{x}) \\
= \; & f(\boldsymbol{x} + \boldsymbol{h} + \boldsymbol{g}) - f(\boldsymbol{x} + \boldsymbol{h}) + f(\boldsymbol{x} + \boldsymbol{h}) - f(\boldsymbol{x}) \\
= \; & \Delta_{\boldsymbol{h}+\boldsymbol{g}} f(\boldsymbol{x}).
\end{aligned}$$

$\square$

As a direct consequence of the above lemma we have:

Corollary 4.2. $\Delta_{\boldsymbol{h}}\Delta_{\boldsymbol{g}} f = \Delta_{\boldsymbol{g}}\Delta_{\boldsymbol{h}} f$ *for each* $\boldsymbol{g}, \boldsymbol{h} \in \mathbb{F}_2^N$.

Corollary 4.3. *For any* $f \in V_N$ *and* $\boldsymbol{h} \in \mathbb{F}_2^N$,

$$\Delta_{\boldsymbol{h}}\Delta_{\boldsymbol{h}} f(\boldsymbol{x}) = -2\Delta_{\boldsymbol{h}} f(\boldsymbol{x})$$

Proof. Obviously $\Delta_{\boldsymbol{0}} f = 0$ for any $f \in V_N$, so the claim follows from the previous lemma by letting $\boldsymbol{g} = \boldsymbol{h}$. □

Also the following lemma can be proven by straightforward computation:

Lemma 4.3. $\Delta_{\boldsymbol{h}}(fg)(\boldsymbol{x}) = \Delta_{\boldsymbol{h}} f(\boldsymbol{x}) g(\boldsymbol{x}+\boldsymbol{h}) + f(\boldsymbol{x})\Delta_{\boldsymbol{h}} g(\boldsymbol{x})$.

4.3.2 Derivatives of the Polynomials

Example 4.2. Let $\boldsymbol{e}_i = (0, \dots, 1 \dots, 0) \in \mathbb{F}_2^N$. If $i \notin S$, then clearly

$$\Delta_{\boldsymbol{e}_i} X_S(\boldsymbol{x}) = X_S(\boldsymbol{x}+\boldsymbol{e}_i) - X_S(\boldsymbol{x}) = X_S(\boldsymbol{x}) - X_S(\boldsymbol{x}) = 0. \tag{4.3.2}$$

On the other hand, it is easy to see that $X_i(\boldsymbol{x}+\boldsymbol{e}_i) = 1 - X_i(\boldsymbol{x})$, so

$$\Delta_{\boldsymbol{e}_i} X_i(\boldsymbol{x}) = X_i(\boldsymbol{x}+\boldsymbol{e}_i) - X_i(\boldsymbol{x}) = 1 - 2X_i(\boldsymbol{x}).$$

Thus if $i \in S$, we can write $X_S = X_{S\setminus\{i\}} X_i$ and use Lemma (4.3) to get

$$\Delta_{\boldsymbol{e}_i} X_S = X_{S\setminus\{i\}}(1-2X_i) = X_{S\setminus\{i\}} - 2X_S. \tag{4.3.3}$$

As an application, we show how the *Moebius inversion formula* can be obtained by repetitive use of discrete derivatives.

Corollary 4.4 (The Moebius inversion formula). *If* $f \in V_N$ *has a representation*

$$f = \sum_{S \subseteq \{1,\dots,N\}} c_S X_S, \tag{4.3.4}$$

then

$$c_S = \sum_{T \subseteq S} (-1)^{|S|-|T|} f\Big(\sum_{i \in T} \boldsymbol{e}_i\Big). \tag{4.3.5}$$

Proof. By Equations (4.3.2) and (4.3.3) it is clear that

$$\Delta_{\boldsymbol{e}_{i_r}} \dots \Delta_{\boldsymbol{e}_{i_1}} f(\boldsymbol{0}) = c_{\{i_1,\dots,i_r\}}. \tag{4.3.6}$$

On the other hand, if we denote $S = \{i_1, \dots, i_r\}$, then it is easy to see, by using induction, that the left hand side of (4.3.6) can be written as

$$\begin{aligned}
& \Delta_{\boldsymbol{e}_{i_r}} \dots \Delta_{\boldsymbol{e}_{i_1}} f(\boldsymbol{0}) \\
= \quad & \Delta_{\boldsymbol{e}_{i_r}} \dots \Delta_{\boldsymbol{e}_{i_2}} \big(f(\boldsymbol{e}_{i_1}) - f(\boldsymbol{0})\big) \\
= \quad & \Delta_{\boldsymbol{e}_{i_r}} \dots \Delta_{\boldsymbol{e}_{i_3}} \big(f(\boldsymbol{e}_{i_2} + \boldsymbol{e}_{i_1}) - f(\boldsymbol{e}_{i_1}) - f(\boldsymbol{e}_{i_2}) + f(\boldsymbol{0})\big) \\
\dots & \\
= \quad & \sum_{T \subseteq S} (-1)^{|S|-|T|} f\big(\sum_{i \in T} \boldsymbol{e}_i\big).
\end{aligned}$$

□

4.3.3 Derivatives of Walsh Functions

Example 4.3. For a Walsh function $W_{\boldsymbol{y}}$ the derivative becomes

$$\begin{aligned}
\Delta_{\boldsymbol{h}} W_{\boldsymbol{y}}(\boldsymbol{x}) &= W_{\boldsymbol{y}}(\boldsymbol{x} + \boldsymbol{h}) - W_{\boldsymbol{y}}(\boldsymbol{x}) \\
&= \frac{1}{\sqrt{2^N}} (\chi_{\boldsymbol{y}}(\boldsymbol{x} + \boldsymbol{h}) - \chi_{\boldsymbol{y}}(\boldsymbol{x})) \\
&= W_{\boldsymbol{y}}(\boldsymbol{x})(\chi_{\boldsymbol{y}}(\boldsymbol{h}) - 1),
\end{aligned}$$

which is to say that a linear operator $\Delta_{\boldsymbol{h}} : V_N \to V_N$ has $W_{\boldsymbol{y}} \in V_N$ as an eigenvector belonging to the eigenvalue $\chi_{\boldsymbol{y}}(\boldsymbol{h}) - 1$.

Thus, if

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in \mathbb{F}_2^n} \widehat{f}(\boldsymbol{y}) W_{\boldsymbol{y}}(\boldsymbol{x}),$$

we have that

$$\Delta_{\boldsymbol{h}} f(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in \mathbb{F}_2^N} (\chi_{\boldsymbol{y}}(\boldsymbol{h}) - 1) \widehat{f}(\boldsymbol{y}) W_{\boldsymbol{y}}(\boldsymbol{x}),$$

i.e.,

$$\widehat{\Delta_{\boldsymbol{h}} f}(\boldsymbol{y}) = (\chi_{\boldsymbol{y}}(\boldsymbol{h}) - 1) \widehat{f}(\boldsymbol{y}) = \begin{cases} -2\widehat{f}(\boldsymbol{y}), & \text{if } \boldsymbol{y} \cdot \boldsymbol{h} = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{4.3.7}$$

Notice that the above example introduces restrictions to functions that can be represented as a directed derivative.

Lemma 4.4. *A function $g \in V_N$ can be represented as $g = \Delta_{\boldsymbol{h}} f$ for some $f \in V_N$ if and only if $\widehat{g}(\boldsymbol{y}) = 0$ whenever $\boldsymbol{y} \cdot \boldsymbol{h} = 0$.*

Proof. We use the notations above and assume first that $g = \Delta_{\boldsymbol{h}} f$. Then

$$\widehat{g}(\boldsymbol{y}) = \widehat{\Delta_{\boldsymbol{h}} f}(\boldsymbol{y}) = \begin{cases} -2\widehat{f}(\boldsymbol{y}), & \text{if } \boldsymbol{y} \cdot \boldsymbol{h} = 1, \\ 0, & \text{otherwise.} \end{cases}$$

Assume then that $\boldsymbol{y} \cdot \boldsymbol{h} = 0$ implies $\widehat{g}(\boldsymbol{y}) = 0$. Letting $f = -\frac{1}{2}g$ we see that

$$\begin{aligned} \Delta_{\boldsymbol{h}} f &= \Delta_{\boldsymbol{h}}\big(-\frac{1}{2}\sum_{\boldsymbol{y}\in\mathbb{F}_2^N} \widehat{g}(\boldsymbol{y})W_{\boldsymbol{y}}\big) \\ &= \Delta_{\boldsymbol{h}}\big(-\frac{1}{2}\sum_{\boldsymbol{y}\cdot\boldsymbol{h}=1} \widehat{g}(\boldsymbol{y})W_{\boldsymbol{y}}\big) \\ &= \sum_{\boldsymbol{y}\cdot\boldsymbol{h}=1} \widehat{g}(\boldsymbol{y})W_{\boldsymbol{y}} = g. \end{aligned}$$

The second-last equality is due to the example above. □

Definition 4.3. If $\boldsymbol{h} \in \mathbb{F}_2^N$ we define the *influence* of direction $\boldsymbol{h}$ to function f as the probability that for a uniformly drawn $\boldsymbol{x} \in \mathbb{F}_2^N$ the value $f(\boldsymbol{x}+\boldsymbol{h})$ differs from $f(\boldsymbol{x})$. In symbols:

$$\mathrm{Inf}_{\boldsymbol{h}}(f) = P(f(\boldsymbol{x}+\boldsymbol{h}) \neq f(\boldsymbol{x})) = P(\Delta_{\boldsymbol{h}} f(\boldsymbol{x}) \neq 0).$$

Lemma 4.5. *For any $f \in V_N$ and $\boldsymbol{h} \in \mathbb{F}_2^N$, $\mathrm{Inf}_{\boldsymbol{h}}(f) = \mathrm{Inf}_{\boldsymbol{h}}(\Delta_{\boldsymbol{h}} f)$.*

Proof. By the definition of influence, $\mathrm{Inf}_{\boldsymbol{h}}(\Delta_{\boldsymbol{h}} f) = P(\Delta_{\boldsymbol{h}}\Delta_{\boldsymbol{h}} f(\boldsymbol{x}) \neq 0)$. But, by Example 4.3, we can write

$$\mathrm{Inf}_{\boldsymbol{h}}(\Delta_{\boldsymbol{h}} f) = P(-2\Delta_{\boldsymbol{h}} f(\boldsymbol{x}) \neq 0) = P(\Delta_{\boldsymbol{h}} f(\boldsymbol{x}) \neq 0) = \mathrm{Inf}_{\boldsymbol{h}}(f).$$

□

Lemma 4.6. *For any $f \in V_N$ and $\boldsymbol{h} \in \mathbb{F}_2^N$ there is a representation $f = f_1 + f_2$ such that $\mathrm{Inf}_{\boldsymbol{h}}(f_1) = 0$.*

Proof. If f has Fourier expansion

$$f = \sum_{\boldsymbol{y}\in\mathbb{F}_2^N} \widehat{f}(\boldsymbol{y})W_{\boldsymbol{y}},$$

define

$$f_1 = \sum_{\boldsymbol{y}\cdot\boldsymbol{h}=0} \widehat{f}(\boldsymbol{y})W_{\boldsymbol{y}}$$

and

$$f_2 = \sum_{\boldsymbol{y}\cdot\boldsymbol{h}=1} \widehat{f}(\boldsymbol{y})W_{\boldsymbol{y}}.$$

Then clearly $f = f_1 + f_2$ and $\Delta_{\boldsymbol{h}} f_1 = 0$ by Example 4.3. Hence $\mathrm{Inf}_{\boldsymbol{h}}(f_1) = P(\Delta_{\boldsymbol{h}} f_1(\boldsymbol{x}) \neq 0) = 0$. □

Lemma 4.7. *If $\mathrm{Inf}_{\boldsymbol{h}}(f) = 0$, then $\widehat{f}(\boldsymbol{y}) = 0$ whenever $\boldsymbol{y} \cdot \boldsymbol{h} = 1$.*

Proof. $\mathrm{Inf}_{\boldsymbol{h}}(f) = 0$ means that $\Delta_{\boldsymbol{h}} f$ is identically zero. But then also $\widehat{\Delta_{\boldsymbol{h}} f}$ is identically zero, and by (4.3.7), $\widehat{\Delta_{\boldsymbol{h}} f(\boldsymbol{y})} = -2\widehat{f}(\boldsymbol{y})$, whenever $\boldsymbol{y} \cdot \boldsymbol{h} = 1$, so the claim follows immediately. □

Example 4.4. $\mathrm{Inf}_{\boldsymbol{e}_i}(X_S) = P(\Delta_{\boldsymbol{e}_i}(X_S) \neq 0)$. Hence if $i \notin S$, then, by Example 4.2, we have that $\mathrm{Inf}_{\boldsymbol{e}_i}(X_S) = 0$. On the other hand, if $i \in S$, then $\Delta_{\boldsymbol{e}_i} X_S = X_{S\setminus\{i\}}(1 - 2X_i)$. Thus $\Delta_{\boldsymbol{e}_i} X_S$ is nonzero if and only if $x_j = 1$ for each $j \in S \setminus \{i\}$. But there are $2^{N-|S\setminus\{i\}|} = 2^N/2^{|S|-1}$ such vectors $\boldsymbol{x}$. Therefore, in the case $i \in S$, we have that

$$\mathrm{Inf}_{\boldsymbol{e}_i}(X_S) = \frac{2^N}{2^N \cdot 2^{|S|-1}} = \frac{2}{2^{|S|}}. = \frac{2}{2^{\deg(X_S)}}.$$

Definition 4.4. Function $f \in V_N$ *depends* on variable X_i, if $\Delta_{\boldsymbol{e}_i} f$ is not identically zero.

Proposition 4.2. *If $f \in V_N$ does not depend on variables $X_{i_1}, \ldots, X_{i_k}$, then* $\deg(f) \leq N - k$.

Proof. By Lemma 4.7, $\widehat{f}(\boldsymbol{y}) = 0$ whenever at least one of the coordinates $\boldsymbol{y}_{i_1}, \ldots, \boldsymbol{y}_{i_k}$ equals to one, but that happens always if $\mathrm{wt}(\boldsymbol{y}) > N - k$. □

Let now $f \in V_N$ be a two-valued function $f : \mathbb{F}_2^N \to \{-1, 1\}$. Then the expression $|f(\boldsymbol{x} + \boldsymbol{h}) - f(\boldsymbol{x})|^2$ is either 0 or 4, depending on whether $f(\boldsymbol{x} + \boldsymbol{h})$ equals to $f(\boldsymbol{x})$ or not. Therefore, for such a function we can express the influence as

$$\mathrm{Inf}_{\boldsymbol{h}}(f) = \frac{1}{4 \cdot 2^N} \sum_{\boldsymbol{x} \in \mathbb{F}_2^N} |f(\boldsymbol{x} + \boldsymbol{h}) - f(\boldsymbol{x})|^2 = \frac{1}{4 \cdot 2^N} \sum_{\boldsymbol{x} \in \mathbb{F}_2^N} |\Delta_{\boldsymbol{h}} f(\boldsymbol{x})|^2 .$$

By using Parseval's identity we can write

$$\mathrm{Inf}_{\boldsymbol{h}}(f) = \frac{1}{4 \cdot 2^N} \sum_{\boldsymbol{y} \in \mathbb{F}_2^N} \left| \widehat{\Delta_{\boldsymbol{h}} f}(\boldsymbol{y}) \right|^2 ,$$

which, by using (4.3.7), can also be written as

$$\mathrm{Inf}_{\boldsymbol{h}}(f) = \frac{1}{2^N} \sum_{\boldsymbol{y} \in \mathbb{F}_2^N} \left| \widehat{f}(\boldsymbol{y}) \right|^2 \cdot \frac{1}{2}(1 - (-1)^{\boldsymbol{y} \cdot \boldsymbol{h}}).$$

Consider now a subset $T \subseteq \mathbb{F}_2^N$. Then

$$\sum_{\boldsymbol{h} \in T} \mathrm{Inf}_{\boldsymbol{h}}(f) = \frac{1}{2^N} \sum_{\boldsymbol{y} \in \mathbb{F}_2^N} \left| \widehat{f}(\boldsymbol{y}) \right|^2 \sum_{\boldsymbol{h} \in T} \frac{1}{2}(1 - (-1)^{\boldsymbol{y} \cdot \boldsymbol{h}}). \tag{4.3.8}$$

Next we show how the concepts defined above can be used to provide a lower bound on the degree of a non-degenerate Boolean function, a result by N. Nisan and M. Szegedy.

Theorem 4.1 (Nisan and Szegedy [21]). *If $f : \mathbb{F}_2^N \to \{-1, 1\}$ is a Boolean function which depends on all $N \geq 3$ variables and has degree at least* 2*, then* $\deg(f) \geq \log_2 N - \log_2 \log_2 N$.

Proof. Let $d = \deg(f)$. Choosing $T = S_1$ (the Hamming sphere of weight one) in equation (4.3.8) gives

$$\begin{aligned}\sum_{\boldsymbol{h}\in S_1} \mathrm{Inf}_{\boldsymbol{h}}(f) &= \frac{1}{2^N} \sum_{\boldsymbol{y}\in\mathbb{F}_2^N} \left|\widehat{f}(\boldsymbol{y})\right|^2 \mathrm{wt}(\boldsymbol{y}) \\ &\leq \frac{1}{2^N} \sum_{\boldsymbol{y}\in\mathbb{F}_2^N} \left|\widehat{f}(\boldsymbol{y})\right|^2 d \\ &= \frac{d}{2^N} \sum_{\boldsymbol{x}\in\mathbb{F}_2^N} |f(\boldsymbol{x})|^2 = d.\end{aligned}$$

The inequality above follows from the assumption that $\deg(f) = d$ and the second-last equality is due to Parseval's identity (4.1.5). On the other hand, since f was assumed to depend on all of its variables, all the summands in

$$\sum_{\boldsymbol{h}\in S_1} \mathrm{Inf}_{\boldsymbol{h}}(f) = \sum_{i=1}^{N} \mathrm{Inf}_{\boldsymbol{e}_i}(f) = \sum_{i=1}^{N} P(\Delta_{\boldsymbol{e}_i} f \neq 0)$$

are nonzero. In fact, by Schwartz' lemma (4.1) we have that

$$P(\Delta_{\boldsymbol{e}_i} f \neq 0) \geq \frac{1}{2^d}$$

for each $i \in \{1, \ldots, N\}$. Combining these inequalities gives us

$$\frac{N}{2^d} \leq d,$$

or equivalently,

$$d2^d \geq N. \tag{4.3.9}$$

Now that function $f(x) = \ln x - \ln\ln x$ is increasing for each $x \geq \mathrm{e}$, inequality (4.3.9) implies that

$$\ln(d2^d) - \ln\ln(d2^d) \geq \ln N - \ln\ln N,$$

which is equivalent to

$$d \ln 2 - \ln\ln 2 - \ln\left(1 + \frac{\log_2 d}{d}\right) \geq \ln N - \ln\ln N. \tag{4.3.10}$$

For $d \geq 2$, term $\ln(1+\log_2 d/d)$ is nonnegative, and therefore (4.3.10) implies

$$d \ln 2 \geq \ln N - \ln\ln N + \ln\ln 2 = \ln N - \ln \log_2 N,$$

which, by dividing by $\ln 2$, gives the desired result. $\square$

Chapter 5

More Properties of V_N

5.1 Polynomials on one Variable

In the continuation, we will make use of polynomials depending of the Hamming weight of elements of $\mathbb{F}_2^N$. In this section, we will represent some useful notations.

5.1.1 Discrete Derivatives of Univariate Polynomials

Let f be any complex-valued function defined on integers. We define the *discrete derivative*[1] of f by

$$\Delta_x f(x) = f(x+1) - f(x).$$

If there is no danger of confusion, we omit the subscript x and just write $\Delta_x f(x) = \Delta f(x)$. By the very definition, it is clear that Δ is a linear operator, i.e.,

$$\Delta(\alpha f(x) + \beta g(x)) = \alpha \Delta f(x) + \beta \Delta g(x).$$

Discrete derivatives of higher order are defined inductively: $\Delta^0 f(x) = f(x)$, and $\Delta^{k+1} f(x) = \Delta(\Delta^k f(x))$ for $k \geq 0$. By using induction, it is easy to conclude that

$$\Delta^l f(x) = \sum_{k=0}^{l} (-1)^k \binom{l}{k} f(x+l-k) = (-1)^l \sum_{k=0}^{l} (-1)^k \binom{l}{k} f(x+k), \quad (5.1.1)$$

whenever $l \geq 0$. If M and N are integers, we can easily verify the following *discrete analogue of the fundamental theorem of calculus:*

$$\sum_{k=M}^{N} \Delta f(k) = f(N+1) - f(M). \quad (5.1.2)$$

[1] To be precise, we should say "discrete derivative on unit interval"

Moreover, differentiation of products has also easily verifiable analogue in the discrete case:

$$\Delta\big(f(x)g(x)\big) = \Delta f(x) \cdot g(x) + f(x+1) \cdot \Delta g(x). \tag{5.1.3}$$

Using induction, the above formula generalizes into discrete version of *Leibniz' rule:*

$$\Delta^l\big(f(x)g(x)\big) = \sum_{k=0}^{l} \binom{l}{k} \Delta^{l-k} f(x+k) \Delta^k g(x). \tag{5.1.4}$$

The formula (5.1.3) together with (5.1.2) easily yields the discrete analogue for integrating by parts:

$$\begin{aligned} & \sum_{k=M}^{N} \Delta f(k) \cdot g(k) \\ = \quad & f(N+1)g(N+1) - f(M)g(M) - \sum_{k=M}^{N} f(k+1)\Delta g(k), \end{aligned} \tag{5.1.5}$$

which is sometimes more useful in an asymmetric form:

$$\sum_{k=M}^{N} \Delta f(k) \cdot g(k) = f(N+1)g(N) - f(M)g(M) - \sum_{k=M}^{N-1} f(k+1)\Delta g(k).$$

5.1.2 Shifted Power Representation

Here we will consider polynomials having complex coefficients and degree at most N. However, the standard *power representation*

$$f(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_N x^N \tag{5.1.6}$$

is not very well compatible with the discrete derivative. For a better representation we will use *shifted power*[2] defined by $x^{(0)} = 1$, $x^{(1)} = x$ and

$$x^{(k)} = x(x-1) \cdot \ldots \cdot (x-k+1),$$

whenever $k \geq 2$. It should be noticed that $x^{(k)}$ is a polynomial of degree k, having 1 as the leading coefficient and $\{0, 1, \ldots, k-1\}$ as zeros. Moreover, it is trivial to see that the polynomials $x^{(0)}$, $x^{(1)}$, ..., $x^{(N)}$ form a basis of the vector space consisting of polynomials having degree at most N: That is, each polynomial f having degree at most N can be uniquely represented as

$$f(x) = d_0 + d_1 x^{(1)} + d_2 x^{(2)} + \ldots + d_N x^{(N)}. \tag{5.1.7}$$

[2] Also called *factorial polynomial* in [19].

We call representation (5.1.7) *shifted power representation.* Using the definition of $x^{(k)}$, it is easy to see that the following equations hold:

$$(x+1)^{(k)} = (x+1)x^{(k-1)} \tag{5.1.8}$$
$$x^{(k)} = x^{(k-1)}(x-k+1). \tag{5.1.9}$$

Subtracting (5.1.9) from (5.1.8) we see that

$$\Delta x^{(k)} = kx^{(k-1)}, \tag{5.1.10}$$

whenever $k \geq 1$. Formula (5.1.10) gives a beautiful analogy to the differentiation of powers. Together with (5.1.2) it is quite useful for evaluating sums.

Using the shifted power representation (5.1.7) and formula $\Delta \frac{1}{k+1} x^{(k+1)} = x^{(k)}$ we can establish

Proposition 5.1. *If f is a polynomial of degree N with leading coefficient c, then $\Delta^k f$ is a polynomial of degree $N-k$ with leading coefficient $N(N-1)\dots(N-k+1)c$. Moreover, for each polynomial f there is a polynomial g such that $\Delta g = f$.*

5.1.3 Binomial Representation

We will find even more convenient basis for polynomials having degree at most N: For $k \geq 0$, we define the *generalized binomial coefficients* by

$$\binom{x}{k} = \frac{x^{(k)}}{k!}. \tag{5.1.11}$$

It is easy to see that

$$\binom{-x}{k} = (-1)^k \binom{x+k-1}{k} \tag{5.1.12}$$

and that each $\binom{x}{k}$ is a polynomial of degree k with leading coefficient $1/k!$. Moreover, any polynomial of degree at most N has a unique representation as

$$f(x) = f_0 - f_1\binom{x}{1} + f_2\binom{x}{2} + \ldots + f_N(-1)^N\binom{x}{N}. \tag{5.1.13}$$

We call (5.1.13) a *binomial representation* of f. The reason for choosing alternating sign for generalized binomial coefficients will become apparent very soon. By (5.1.11) it is clear that

$$\Delta\binom{x}{k} = \binom{x}{k-1}$$

for $k \geq 1$, so binomial representation is very well compatible with discrete derivative. It is also clear that

$$\Delta^l f(x) = (-1)^l f_l + (-1)^{l+1} f_{l+1} \binom{x}{1} + \ldots + (-1)^N f_N \binom{x}{N-l}, \quad (5.1.14)$$

whenever $0 \leq l \leq N$. Equation (5.1.14) gives us

$$(-1)^l f_l = \Delta^l f(0), \quad (5.1.15)$$

which, using (5.1.1), can be written as

$$f_l = (-1)^l \Delta^l f(0) = \sum_{k=0}^{l} (-1)^k \binom{l}{k} f(k). \quad (5.1.16)$$

The reason for choosing alternating signs for the generalized binomial coefficients becomes now reasonable:

Since $x^{(k)}$ has zeros at $\{0, 1, \ldots, k-1\}$, also $\binom{x}{k}$ does. It follows that the value $f(l)$ at integer point $l \in \{0, 1, \ldots, N\}$ depends only on coefficients f_0, f_1, …, f_l. To be precise,

$$f(l) = \sum_{k=0}^{l} (-1)^l \binom{l}{k} f_k,$$

which is gives a beautiful symmetry with (5.1.16). To summarize:

Proposition 5.2. *Each polynomial having degree at most N has unique representation*

$$f(x) = \sum_{k=0}^{N} (-1)^k \binom{x}{k} f_k. \quad (5.1.17)$$

For each $l \in \{0, 1, \ldots, N\}$ the value $f(l)$ and coefficient f_l can be found using the following symmetric formulae:

$$f(l) = \sum_{k=0}^{l} (-1)^k \binom{l}{k} f_k \quad (5.1.18)$$

$$f_l = \sum_{k=0}^{l} (-1)^k \binom{l}{k} f(k) \quad (5.1.19)$$

Representation (5.1.17) and the associated formulas (5.1.18) and (5.1.19) are pretty handy: They can be used to straightforwardly establish the following well-known facts:

Theorem 5.1 (The Interpolation Theorem). *Let $\{v_0, v_1, \ldots v_N\}$ be a set of complex numbers. There exists a polynomial P having degree at most N such that $P(i) = v_i$ for each $i \in \{0, 1, \ldots, N\}$.*

Proof. Polynomial P can be found by using formula (5.1.19). □

Theorem 5.2. *Let f be a polynomial of degree N with leading coefficient $c \neq 0$ (in the power representation), and*

$$\|f\|_\infty = \max\{|f(0)|, |f(1)|, \ldots, |f(N)|\}.$$

Then $\|f\|_\infty \geq |c| \frac{N!}{2^N}$.

Proof. If f is represented as

$$f(x) = \sum_{k=0}^{N} (-1)^k \binom{x}{k} f_k,$$

then the leading coefficient of power representation (5.1.6) of f is

$$c = (-1)^N f_N / N!.$$

By triangle inequality and formula (5.1.19),

$$\begin{aligned} |f_N| &= \left| \sum_{k=0}^{N} (-1)^k \binom{N}{k} f(k) \right| \leq \sum_{k=0}^{N} \binom{N}{k} |f(k)| \\ &\leq \|f\|_\infty \sum_{k=0}^{N} \binom{N}{k} = \|f\|_\infty 2^N. \end{aligned}$$

Thus

$$\|f\|_\infty \geq \frac{|f_N|}{2^N} = |c| \frac{N!}{2^N},$$

as claimed. □

Remark 5.1. The above bound for $\|f\|_\infty$ is quite tight: By the interpolation theorem (Theorem 5.1) one can find f such that $f(k) = (-1)^k c \frac{N!}{2^N}$, for each $k \in \{0, 1, \ldots, N\}$. For such a polynomial f, $\|f\|_\infty = |c| \frac{N!}{2^N}$, and $f_N = cN!$, so the leading coefficient in the power representation is c.

5.2 Character Basis

5.2.1 Krawtchouk Polynomials

Let us define

$$P^{(r)}(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in S_r} \chi_{\boldsymbol{y}}(\boldsymbol{x}). \tag{5.2.1}$$

It is rather clear that the value $P^{(r)}(\boldsymbol{x})$ is invariant under any permutation of the coordinates of $\boldsymbol{x}$, which is to say that the value $P^{(r)}(\boldsymbol{x})$ actually depends only on the Hamming weight of $\boldsymbol{x}$. In fact, to count the value $P^{(r)}(\boldsymbol{x})$, let us denote $x = \mathrm{wt}(\boldsymbol{x})$ and let X be the set of indices of nonzero coordinates of $\boldsymbol{x}$. To choose a vector $\boldsymbol{y}$ in the rth Hamming sphere, one must choose r indices for nonzero coordinates. We can choose l coordinates in X and $r-l$ outside of X. The total number of ways to do so is

$$\binom{x}{l}\binom{N-x}{r-l}, \tag{5.2.2}$$

and then $(-1)^{\boldsymbol{x}\cdot\boldsymbol{y}} = (-1)^l$. On the other hand, l can be any number between 0 and r, so

$$P^{(r)}(\boldsymbol{x}) = \sum_{l=0}^{r}(-1)^l\binom{N-x}{r-l}\binom{x}{l},$$

when $x = \mathrm{wt}(\boldsymbol{x})$. Notice that we can regard expression (5.2.2) as a product of generalized binomial coefficients, defined for all real (and also complex) values of x. Doing so, we notice that (5.2.2) is a polynomial of x having degree $l + r - l = r$. Hence the polynomial

$$K_r(x) = \sum_{l=0}^{r}(-1)^l\binom{N-x}{r-l}\binom{x}{l}$$

induced by $P^{(r)} \in V_N$, called the *rth Krawtchouk polynomial*, has degree at most r. In fact, we can quite easily find the Binomial representation of $K_r(x)$ to notice that

$$K_r(x) = \sum_{l=0}^{r}(-2)^l\binom{N-l}{r-l}\binom{x}{l}$$

actually has degree r. Since

$$\begin{aligned}
&\sum_{\boldsymbol{x}\in\mathbb{F}_2^N} P^{(r)}(\boldsymbol{x})P^{(s)}(\boldsymbol{x}) = \langle P^{(r)} \,|\, P^{(s)}\rangle \\
&= \langle \sum_{\boldsymbol{y}\in S_r}\chi_{\boldsymbol{y}} \,|\, \sum_{\boldsymbol{z}\in S_s}\chi_{\boldsymbol{z}}\rangle \\
&= \begin{cases} 2^N\binom{N}{r}, & \text{if } r = s, \\ 0 & \text{otherwise.}\end{cases}
\end{aligned}$$

But on the other hand,

$$\begin{aligned}
&\sum_{\boldsymbol{x}\in\mathbb{F}_2^N} P^{(r)}(\boldsymbol{x})P^{(s)}(\boldsymbol{x}) = \sum_{k=0}^{N}\sum_{\boldsymbol{x}\in S_k} K_r(k)K_r(k) \\
&= \sum_{k=0}^{N}\binom{N}{k}K_r(k)K_s(k).
\end{aligned}$$

Thus the Krawtchouk polynomials satisfy

$$\sum_{k=0}^{N}\binom{N}{k}K_r(k)K_s(k)=\begin{cases}2^N\binom{N}{r}, & \text{if } r=s,\\ 0, & \text{otherwise.}\end{cases}$$

It is not difficult to conclude that polynomials having degree at most N form a vector space P_N, when sum and scalar multiplication are defined pointwise. Also, one can straightforwardly check that

$$\langle P_1 \,|\, P_2\rangle_K=\sum_{k=0}^{N}\binom{N}{k}P_1(k)^*P_2(k) \tag{5.2.3}$$

defines an inner product on that vector space. The above formulae can therefore interpreted as follows: with respect to inner product (5.2.3) the Krawtchouk polynomials form an orthogonal basis of P_N, even such that $\deg(K_r)=r$ for each $r\in\{0,1,\ldots,N\}$.

5.2.2 Symmetric Functions

Recall that any function $f\in V_N$ has a Fourier representation

$$f=\sum_{\boldsymbol{y}\in\mathbb{F}_2^N}\widehat{f}(\boldsymbol{y})W_{\boldsymbol{y}}.$$

We say that f is *symmetric*, if $f(\boldsymbol{x})=f(\pi(\boldsymbol{x}))$, whenever $\boldsymbol{x}\in\mathbb{F}_2^N$ and π is a permutation on coordinates of $\boldsymbol{x}$.

Proposition 5.3. *If f is a symmetric function, then $\widehat{f}(\boldsymbol{y}_1)=\widehat{f}(\boldsymbol{y}_2)$ whenever* $\mathrm{wt}(\boldsymbol{y}_1)=\mathrm{wt}(\boldsymbol{y_2})$

Proof. By (4.1.4),

$$\widehat{f}(\boldsymbol{y})=\sum_{\boldsymbol{x}\in\mathbb{F}_2^N}f(\boldsymbol{x})W_{\boldsymbol{x}}(\boldsymbol{y}). \tag{5.2.4}$$

Since f is symmetric, the value $f(\boldsymbol{x})$ depends only on the Hamming weight of $\boldsymbol{x}$. We denote $f_k=f(\boldsymbol{x})$ if $\mathrm{wt}(\boldsymbol{x})=k$, and write (5.2.4) as

$$\begin{aligned}\widehat{f}(\boldsymbol{y}) &= \sum_{i=0}^{N}\sum_{\boldsymbol{x}\in S_i}f(\boldsymbol{x})W_{\boldsymbol{x}}(\boldsymbol{y})\\ &= \frac{1}{\sqrt{2^N}}\sum_{i=0}^{N}f_i\sum_{\boldsymbol{x}\in S_i}\chi_{\boldsymbol{x}}(\boldsymbol{y})\\ &= \frac{1}{\sqrt{2^N}}\sum_{i=0}^{N}f_iK_i(\mathrm{wt}(\boldsymbol{y})),\end{aligned}$$

which shows that $\widehat{f}(\boldsymbol{y})$ depends only on the Hamming weight of $\boldsymbol{y}$. □

Corollary 5.1. *Each symmetric function* $f : \mathbb{F}_2^N \to \mathbb{C}$ *of degree* d *can be expressed as polynomial of* $\mathrm{wt}(\boldsymbol{x})$ *of degree* d.

Proof. By the above proposition, $\widehat{f}(\boldsymbol{y})$ depends only on $i = \mathrm{wt}(\boldsymbol{y})$. We can write $\widehat{f}(\boldsymbol{y}) = \widehat{f}_i$, and see that

$$\begin{aligned} f(\boldsymbol{x}) &= \sum_{\boldsymbol{y}\in\mathbb{F}_2^N} \widehat{f}(\boldsymbol{y}) W_{\boldsymbol{y}}(\boldsymbol{x}) \\ &= \frac{1}{\sqrt{2^N}} \sum_{i=0}^{N} \sum_{\boldsymbol{y}\in S_i} \widehat{f}_i \chi_{\boldsymbol{y}}(\boldsymbol{x}) \\ &= \frac{1}{\sqrt{2^N}} \sum_{i=0}^{N} \widehat{f}_i K_i(\mathrm{wt}(\boldsymbol{x})), \end{aligned}$$

which is the representation required. □

5.3 The Hybrid Basis

We have already seen that the character basis, as well as its scaled version, the Fourier basis consisting of Walsh functions, has some significant properties, e.g., the symmetric inversion formulae (4.1.3) and (4.1.4) and Parseval's identity (4.1.5). It is also a well-known fact that the best approximations of a Boolean function f with respect to so-called L_2-norm (which we will define later) can be obtained by truncating the Fourier representation (representation in the basis of Walsh functions). On the other hand, we will later see that the approximations obtained by using the Walsh function basis can be quite bad with respect to so-called L_∞-norm (which we will also define later). For better approximations and an analogous formula to (5.2.1) we will study another basis.

5.3.1 Functions $B^{(N)}$

For each $\boldsymbol{y} \in \mathbb{F}_2^N$, let us define a function

$$B_{\boldsymbol{y}}^{(N)} : \mathbb{F}_2^N \to \mathbb{C}$$

by

$$B_{\boldsymbol{y}}^{(N)}(\boldsymbol{x}) = \binom{\mathrm{wt}(\boldsymbol{y})}{|\mathrm{supp}(\boldsymbol{x}) \cap \mathrm{supp}(\boldsymbol{y})|} \chi_{\boldsymbol{y}}(\boldsymbol{x}).$$

A very first and evident observation on functions $B_{\boldsymbol{y}}^{(N)}$ to be made is that the value $B_{\boldsymbol{y}}^{(N)}(\boldsymbol{x})$ depends on only those bits of $\boldsymbol{x}$, which are in $\mathrm{supp}(\boldsymbol{y})$. Thus we have immediately, by the Proposition 4.2

Proposition 5.4.

$$\deg(B_{\boldsymbol{y}}^{(N)}) \leq |\mathrm{supp}(\boldsymbol{y})| = \mathrm{wt}(\boldsymbol{y}).$$

Another straightforward but important observation is supplied in the following proposition.

Proposition 5.5. *Assume that* $d = \mathrm{wt}(\boldsymbol{y}) < N$ *and that* $\mathrm{supp}(\boldsymbol{y}) = \{i_1, \ldots, i_d\}$. *Then*

$$B_{\boldsymbol{y}}^{(N)} = \binom{d}{|\mathrm{supp}(\boldsymbol{x}) \cap \mathrm{supp}(\boldsymbol{y})|} \chi_{\boldsymbol{y}}(\boldsymbol{x}) = B_{\mathbf{1}}^{(d)}(x_{i_1}, \ldots, x_{i_d}),$$

where $\mathbf{1} \in \mathbb{F}_2^d$ *stands for the vector* $\mathbf{1} = (1, 1, \ldots, 1)$.

Example 5.1. For $\mathbb{F}_2^1 = \mathbb{F}_2$ we have two functions $B_{\mathbf{0}}^{(1)}$ and $B_{\mathbf{1}}^{(1)}$, which can be expressed as

$$\begin{cases} B_{\mathbf{0}}^{(1)}(x) & = & \binom{0}{|\mathrm{supp}(x) \cap \emptyset|} \chi_{\mathbf{0}}(x) = 1, \\ B_{\mathbf{1}}^{(1)}(x) & = & \binom{1}{|\mathrm{supp}(x) \cap \mathrm{supp}(\mathbf{1})|} \chi_{\mathbf{1}}(x) = (-1)^x. \end{cases}$$

By the above proposition, to find representations for functions $B_{\boldsymbol{y}}^{(N)}$, it is sufficient to find the representations in the case $\boldsymbol{y} = \mathbf{1}$; all the remaining cases can be treated recursively in a space with a smaller dimension.

In case $\boldsymbol{y} = \mathbf{1}$, the functions B become

$$B_{\mathbf{1}}^{(N)}(\boldsymbol{x}) = \binom{N}{\mathrm{wt}(\boldsymbol{x})} \chi_{\mathbf{1}}(\boldsymbol{x}).$$

For a small notational simplicity, we first consider functions

$$C_N : \mathbb{F}_2^N \to \mathbb{C}$$

defined by

$$C_N(\boldsymbol{x}) = \binom{N}{\mathrm{wt}(\boldsymbol{x})}.$$

Functions C_N are clearly symmetric, and therefore, in their fourier representation

$$C_N = \frac{1}{\sqrt{2^N}} \sum_{\boldsymbol{y} \in \mathbb{F}_2^N} \widehat{C_N}(\boldsymbol{y}) \chi_{\boldsymbol{y}},$$

the coefficient $\widehat{C_N}(\boldsymbol{y})$ depends only on the Hamming weight of $\boldsymbol{y}$. Let us therefore consider the value

$$\widehat{C_N}(11 \ldots 10 \ldots 0)$$

with d 1's and $N-d$ 0's. We have that

$$\begin{aligned}
& \widehat{C_N}(11\dots10\dots0) \\
= & \frac{1}{\sqrt{2^N}} \sum_{\boldsymbol{x}\in\mathbb{F}_2^N} \binom{N}{\mathrm{wt}(\boldsymbol{x})}(-1)^{x_1+\dots+x_d} \\
= & \frac{1}{\sqrt{2^N}} \sum_{\boldsymbol{x}'\in\mathbb{F}_2^d} \sum_{\boldsymbol{x}''\in\mathbb{F}_2^{N-d}} \binom{N}{\mathrm{wt}(\boldsymbol{x}')+\mathrm{wt}(\boldsymbol{x}'')}(-1)^{\mathrm{wt}(\boldsymbol{x}')} \\
= & \frac{1}{\sqrt{2^N}} \sum_{l=0}^{d} \sum_{\boldsymbol{x}'\in S_l^{(d)}} \sum_{\boldsymbol{x}\in\mathbb{F}_2^{N-d}} \binom{N}{\mathrm{wt}(\boldsymbol{x})+l}(-1)^{l} \\
= & \frac{1}{\sqrt{2^N}} \sum_{l=0}^{d} \binom{d}{l} \sum_{i=0}^{N-d} \sum_{\boldsymbol{x}\in S_i^{N-d}} \binom{N}{i+l}(-1)^{l} \\
= & \frac{1}{\sqrt{2^N}} \sum_{l=0}^{d} \binom{d}{l}(-1)^{l} \sum_{i=0}^{N-d} \binom{N-d}{i}\binom{N}{i+l} \\
= & \frac{1}{\sqrt{2^N}} \sum_{l=0}^{d} (-1)^{l}\binom{d}{l}\binom{2N-d}{N-l}.
\end{aligned} \tag{5.3.1}$$

The final equality is obtained by using Vandermonde's convolution (see Appendix).

Lemma 5.1. *If* $\mathrm{wt}(\boldsymbol{y}) = d = 2r+1$ *is an odd number, then* $\widehat{C_N}(\boldsymbol{y}) = 0$

Proof. By using the expression above, we see that if $d = 2r+1$, then

$$\begin{aligned}
& \sum_{l=0}^{d}(-1)^l\binom{d}{l}\binom{2N-d}{N-l} \\
= & \sum_{l=0}^{r}(-1)^l\binom{2r+1}{l}\binom{2N-2r-1}{N-l} \\
+ & \sum_{l=r+1}^{2r+1}(-1)^l\binom{2r+1}{l}\binom{2N-2r-1}{N-l} \\
= & \sum_{l=0}^{r}(-1)^l\binom{2r+1}{l}\binom{2N-2r-1}{N-l} \\
+ & \sum_{k=0}^{r}(-1)^{2r+1-k}\binom{2r+1}{2r+1-k}\binom{2N-2r-1}{N-2r-1+k}
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{l=0}^{r}(-1)^l\binom{2r+1}{l}\binom{2N-2r-1}{N-l} \\
&- \sum_{l=0}^{r}(-1)^l\binom{2r+1}{l}\binom{2N-2r-1}{N-l} = 0,
\end{aligned}
$$

which proves the claim. □

Lemma 5.2. *If* $\mathrm{wt}(\boldsymbol{y}) = d = 2r$ *is an even number, then*

$$\widehat{C_N}(\boldsymbol{y}) = \frac{(-1)^r}{\sqrt{2^N}}\frac{\binom{N}{r}\binom{2N}{N}}{\binom{2N}{2r}}.$$

Proof. By (5.3.1), for $d = 2r = \mathrm{wt}(\boldsymbol{y})$ we can write the coefficient $\widehat{C_N}(\boldsymbol{y})$ as

$$
\begin{aligned}
& \frac{1}{\sqrt{2^N}}\Big(\sum_{l=0}^{r-1}(-1)^l\binom{2r}{l}\binom{2N-2r}{N-l} + \sum_{l=r+1}^{2r}(-1)^l\binom{2r}{l}\binom{2N-2r}{N-l} \\
+ & (-1)^r\binom{2r}{r}\binom{2N-2r}{N-r}\Big) \\
= & \frac{1}{\sqrt{2^N}}\Big(\sum_{l=0}^{r-1}(-1)^l\binom{2r}{l}\binom{2N-2r}{N-l} \\
+ & \sum_{k=0}^{r-1}(-1)^{-k+2r}\binom{2r}{2r-k}\binom{2N-2r}{N+k-2r} + (-1)^r\binom{2r}{r}\binom{2N-2r}{N-r}\Big) \\
= & \frac{1}{\sqrt{2^N}}\Big(\sum_{l=0}^{r-1}(-1)^l\binom{2r}{l}\binom{2N-2r}{N-l} + \sum_{l=0}^{r-1}(-1)^l\binom{2r}{l}\binom{2N-2r}{N-l} \\
+ & (-1)^r\binom{2r}{r}\binom{2N-2r}{N-r}\Big) \\
= & \frac{1}{\sqrt{2^N}}\Big(2\sum_{l=0}^{r-1}(-1)^l\binom{2r}{l}\binom{2N-2r}{N-l} + (-1)^r\binom{2r}{r}\binom{2N-2r}{N-r}\Big)
\end{aligned}
$$

The latest expression can be also rewritten as

$$
\begin{aligned}
& \frac{1}{\sqrt{2^N}}\Big(2\sum_{l=0}^{r-1}(-1)^l\binom{2r}{l}\binom{2N-2r}{N-l} + (-1)^r\binom{2r}{r}\binom{2N-2r}{N-r}\Big) \\
= & \frac{1}{\sqrt{2^N}}\Big(2\sum_{l=0}^{r-1}(-1)^l\binom{2r}{l}\frac{(2N-2r)!}{(N-l)!(N-2r+l)!} \\
+ & (-1)^r\binom{2r}{r}\frac{(2N-2r)!}{(N-r)!(N-r)!}\Big) \\
= & \frac{(2N-2r)!}{\sqrt{2^N}}\Big(2\sum_{l=0}^{r-1}(-1)^l\binom{2r}{l}\frac{N^{(l)}N^{(2r-l)}}{N!N!} + (-1)^r\binom{2r}{r}\frac{N^{(r)}N^{(r)}}{N!N!}\Big)
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{N^{(r)}(2N-2r)!}{N!N!\sqrt{2^N}} \\
&\times \left(2\sum_{l=0}^{r-1}(-1)^l\binom{2r}{l}N^{(l)}(N-r)^{(r-l)}+(-1)^r\binom{2r}{r}N^{(r)}\right) \quad (5.3.2)
\end{aligned}
$$

To finish the proof, we will use an auxiliary result.

Lemma 5.3.

$$
2\sum_{l=0}^{r-1}(-1)^l\binom{2r}{l}N^{(l)}(N-r)^{(r-l)}+(-1)^r\binom{2r}{r}N^{(r)}=(-1)^r(2r)^{(r)}.
$$

Proof. We first write

$$
f(x)=2\sum_{l=0}^{r-1}(-1)^l\binom{2r}{l}x^{(l)}(x-r)^{(r-l)}+(-1)^r\binom{2r}{r}x^{(r)} \qquad (5.3.3)
$$

and notice that $f(x)$ is a polynomial of x having degree at most r. As each $(x-r)^{(r-l)}$ becomes zero by substituting $x=r$ in (5.3.3), it is easy to see that

$$
f(r)=(-1)^r\binom{2r}{r}r^{(r)}=(-1)^r(2r)^{(r)}.
$$

We will still show that $f(0)=f(1)=\ldots f(r-1)=(-1)^r(2r)^{(r)}$, and the proof is complete, since a polynomial of degree at most r having value $(-1)^r(2r)^{(r)}$ at $r+1$ points must be a constant polynomial.

By (5.1.12), $f(x)$ can be written as

$$
\begin{aligned}
f(x) &= 2\sum_{l=0}^{r-1}(-1)^l\binom{2r}{l}x^{(l)}(r-x+r-l-1)^{(r-l)}(-1)^{r-l} \\
&+ (-1)^r\binom{2r}{r}x^{(r)} \\
&= 2(-1)^r\sum_{l=0}^{r-1}\binom{2r}{l}x^{(l)}(2r-x-l-1)^{(r-l)}+(-1)^r\binom{2r}{r}x^{(r)},
\end{aligned}
$$

which shows that for any $0\le k<r$ we have that

$$
f(k)=2(-1)^r\sum_{l=0}^{r-1}\binom{2r}{l}k^{(l)}(2r-k-l-1)^{(r-l)}. \qquad (5.3.4)
$$

By writing $M=2r$ and by noticing that the summands become zero as l

exceeds k, we can re-express (5.3.4) as

$$\begin{aligned} f(k) &= 2(-1)^r \sum_{l=0}^{k} \binom{M}{l} k^{(l)} (M-k-l-1)^{(r-l)} \\ &= 2(-1)^r \sum_{l=0}^{k} \binom{M}{l} k^{(l)} (M-k-l-1)^{(k-l)} (M-2k-1)^{(r-k)} \\ &= 2(-1)^r (M-2k-1)^{r-k} \sum_{l=0}^{k} \binom{M}{l} k^{(l)} (M-k-l-1)^{(k-l)}. \end{aligned}$$

To complete the proof of Lemma 5.3, we will investigate the expression

$$\sum_{l=0}^{k} \binom{M}{l} k^{(l)} (M-k-l-1)^{(k-l)}.$$

It is useful to notice that

$$\binom{M}{l} k^{(l)} = \binom{k}{l} M^{(l)}.$$

Lemma 5.4.

$$\sum_{l=0}^{k} \binom{k}{l} M^{(l)} (M-k-l-1)^{(k-l)} = 2^k \prod_{l=1}^{k} (M-(2l-1)).$$

Proof. Let

$$g(x) = \sum_{l=0}^{k} \binom{k}{l} x^{(l)} (x-k-l-1)^{(k-l)}.$$

Clearly g has degree k, and the coefficient of the highest term of g is

$$\sum_{l=0}^{k} \binom{k}{l} = 2^k,$$

so we have

$$g(x) = 2^k \prod_{l=1}^{k} (x-\alpha_i)$$

for some numbers α_i. It remains to show that g has zeros $2r-1$ for $r \in \{1, 2, \ldots, k\}$. Now

$$g(2r-1) = \sum_{l=0}^{k} \binom{k}{l} (2r-1)^{(l)} (2r-k-l-2)^{(k-l)},$$

which becomes, (using (5.1.12)) under the condition that $k \geq 2r-1$,

$$\begin{aligned} g(2r-1) &= \sum_{l=0}^{2r-1} \binom{k}{l} (-1)^{k-l} (2r-1)^{(l)} (2k-2r+1)^{(k-l)} \\ &= \sum_{l=0}^{r-1} \binom{2r-1}{l} (-1)^{k-l} k^{(l)} (2k-2r+1)^{(k-l)} \\ &+ \sum_{l=r}^{2r-1} \binom{2r-1}{l} (-1)^{k-l} k^{(l)} (2k-2r+1)^{(k-l)} \end{aligned}$$

By letting $l = -a + 2r - 1$ in the latest sum, we can rewrite the above expression as

$$\begin{aligned} &= \sum_{l=0}^{r-1} \binom{2r-1}{l} (-1)^{k-l} k^{(l)} (2k-2r+1)^{(k-l)} \\ &+ \sum_{a=0}^{r-1} \binom{2r-1}{-a+2r-1} (-1)^{k+a-2r+1} k^{(2r-1-a)} (2k-2r+1)^{(k-2r+1+a)}, \end{aligned}$$

which gives, by again replacing a by l, that

$$\begin{aligned} & g(2r+1) \\ &= \sum_{l=0}^{r-1} \binom{2r-1}{l} (-1)^{k-l} k^{(l)} (2k-2r+1)^{(k-l)} \\ &- \sum_{l=0}^{r-1} \binom{2r-1}{l} (-1)^{k-l} k^{(2r-1-l)} (2k-2r+1)^{(k-2r+1+l)} \\ &= \sum_{l=0}^{r-1} \binom{2r-1}{l} (-1)^{k-l} k^{(l)} (2k-2r+1)^{(k-2r+1+l)} (k-l)^{(2r-2l-1)} \\ &- \sum_{l=0}^{r-1} \binom{2r-1}{l} (-1)^{k-l} k^{(l)} (k-l)^{(2r-1-2l)} (2k-2r+1)^{(k-2r+1+l)} \\ &= 0. \end{aligned}$$

Recall that the above result holds true under the condition $k \geq 2r-1$, which is equivalent to $r \leq \frac{k+1}{2}$. The remaining cases can be handled by noticing that

$$g(2k-x) = \sum_{l=0}^{k} \binom{k}{l} (2k-x)^{(l)} (2k-x-k-l-1)^{(k-l)},$$

which can be rewritten, by replacing l by $k-l$ and using (5.1.12), as

$$\begin{aligned} g(2k-x) &= \sum_{l=0}^{k} \binom{k}{k-l} (2k-x)^{(k-l)} (k-x-(k-l)-1)^{(l)} \\ &= \sum_{l=0}^{k} \binom{k}{l} (x-k-l-1)^{(k-l)} x^{(l)} (-1)^k \\ &= (-1)^k g(x), \end{aligned}$$

which is to say that $g(x) = (-1)^k g(2k-x)$.

Thus, if $r > \frac{k+1}{2}$, then

$$\begin{aligned} g(2r-1) &= g(2k-(2k-2r+1)) \\ &= (-1)^k g(2k-2r+1) \\ &= (-1)^k g(2(k-r+1)-1) = 0, \end{aligned}$$

since $k-r+1 < \frac{k+1}{2}$. □

The proof of Lemma 5.3 completed. Now that the identity

$$\sum_{l=0}^{k} \binom{k}{l} M^{(l)} (M-k-l-1)^{(k-l)} = 2^k \prod_{l=1}^{k} (M-(2l-1))$$

is known to be true, we can write (5.3.4) as

$$f(k) = 2(-1)^r (M-2k-1)^{(r-k)} 2^k \prod_{l=1}^{k} (M-(2l-1)) \tag{5.3.5}$$

for any k such that $0 \le k < r$ (recall that $M = 2r$). We will now demonstrate that for the appropriate values of k, expression (5.3.5) is equal to $(-1)^r (2r)^{(r)}$.

For $k = 0$, (5.3.5) just becomes

$$f(0) = 2(-1)^r (2r-1)^{(r)} = 2(-1)^r (2r-1)^{(r-1)r} = (-1)^r (2r)^{(r)}.$$

Assume then that the claim is true for some value $k < r-1$. For $k+1$ we write

$$\begin{aligned} f(k+1) &= 2(-1)^r (2r-2k-3)^{(r-k-1)} 2^{k+1} \prod_{l=1}^{k+1} (2r-(2l-1)) \\ &= 2(-1)^r (2r-2k-3)^{(r-k-2)} (2r-2k-2)(2r-2k-1) \\ &\times \; 2^k \prod_{l=1}^{k} (2r-(2l-1)) \\ &= 2(-1)^r (2r-2k-1)^{(r-k)} 2^k \prod_{l=1}^{k} (2r-(2l-1)) \\ &= f(k) = (-1)^r (2r)^{(r)}, \end{aligned}$$

which shows that $f(k) = (-1)^r(2r)^{(r)}$ for each $k \in \{0, 1, \ldots, r-1\}$ and completes the proof of Lemma 5.3. $\square$

The proof of Lemma 5.2 completed. If $\mathrm{wt}(\boldsymbol{y}) = 2r$, we have, by (5.3.2) and by the above results (Lemmata 5.3 and 5.2) that

$$\begin{aligned}
\widehat{C_N}(\boldsymbol{y}) &= \frac{N^{(r)}(2N-2r)!}{N!N!\sqrt{2^N}}(-1)^r(2r)^{(r)} \\
&= \frac{(-1)^r}{\sqrt{2^N}}\frac{N^{(r)}}{r!}\frac{(2N)!}{N!N!}\frac{(2N-2r)!(2r)!}{(2N)!} \\
&= \frac{(-1)^r}{\sqrt{2^N}}\frac{\binom{N}{r}\binom{2N}{N}}{\binom{2N}{2r}}. \qquad (5.3.6)
\end{aligned}$$

$\square$

Corollary 5.2. *Function $B_{\boldsymbol{y}}^{(N)}$ has degree* $\mathrm{wt}(\boldsymbol{y})$.

Proof. By Proposition 5.4 we only have to show that $\deg(B_{\boldsymbol{y}}^{(N)})$ cannot be less than $d = \mathrm{wt}(\boldsymbol{y})$. Writing $\mathrm{supp}(\boldsymbol{y}) = \{i_1, \ldots, i_d\}$ we have

$$B_{\boldsymbol{y}}^{(N)}(\boldsymbol{x}) = B_{\mathbf{1}}^{(d)}(x_{i_1}, \ldots, x_{i_d}).$$

But clearly $\widehat{B_{\mathbf{1}}^{(d)}}(\mathbf{1}) = \widehat{C_d}(\mathbf{0}) \neq 0$, which proves the claim. $\square$

Proposition 5.6. *Functions $B_{\boldsymbol{y}}$ form a basis of V_N.*

Proof. By forming the $2^N \times 2^N$ transformation matrix by using representations

$$B_{\boldsymbol{y}} = \sum_{\boldsymbol{z} \in \mathbb{F}_2^N} \widehat{B_{\boldsymbol{y}}}(\boldsymbol{z}) W_{\boldsymbol{z}}$$

we immediately notice that the matrix $B_{\boldsymbol{yx}} = \widehat{B_{\boldsymbol{y}}}(\boldsymbol{x})$ is lower-triangular having no zeros in the diagonal. Therefore the transformation matrix is invertible. $\square$

Basis $\{B_{\boldsymbol{y}} \mid \boldsymbol{y} \in \mathbb{F}_2^N\}$ will be called *hybrid basis* of V_N.

5.3.2 Discrete Chebyshev Polynomials

Choose $r \in \{0, 1, \ldots, N\}$ and consider expression

$$D^{(r)}(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in S_r} B_{\boldsymbol{y}}(\boldsymbol{x})$$

analogous to (5.2.1). Again it is easy to see that $D^{(r)}(\boldsymbol{x})$ depends only on the Hamming weight $x = \mathrm{wt}(\boldsymbol{x})$. In fact, we can write

$$\begin{aligned} D^{(r)}(\boldsymbol{x}) &= \sum_{\boldsymbol{y}\in S_r} \binom{\mathrm{wt}(\boldsymbol{y})}{|\mathrm{supp}(\boldsymbol{y}) \cap \mathrm{supp}(\boldsymbol{x})|} \chi_{\boldsymbol{y}}(\boldsymbol{x}) \\ &= \sum_{i=0}^{r} \sum_{\substack{\boldsymbol{y}\in S_r \\ |\mathrm{supp}(\boldsymbol{x})\cap\mathrm{supp}(\boldsymbol{y})|=i}} \binom{r}{i}(-1)^i \\ &= \sum_{i=0}^{r}(-1)^i \binom{r}{i}\binom{N-x}{r-i}\binom{x}{i}. \end{aligned}$$

Definition 5.1. Polynomial

$$D_r(x) = \sum_{i=0}^{r}(-1)^i \binom{r}{i}\binom{N-x}{r-i}\binom{x}{i} \tag{5.3.7}$$

is called the *rth Discrete Chebyshev polynomial* on interval $[0, N]$. If there is no danger of confusion, the interval $[0, N]$ is not mentioned explicitly.

Clearly D_r has degree at most r, and after finding the Binomial representation

$$D_r(x) = \sum_{i=0}^{r}(-1)^i \binom{r+i}{r}\binom{N-i}{r-i}\binom{x}{i} \tag{5.3.8}$$

we can see that D_r has degree exactly r. By using (5.1.12) and (5.3.7) we can write

$$\begin{aligned} D_r(x) &= \sum_{i=0}^{r}(-1)^{r-i} \binom{r}{r-i}\binom{N-x}{i}\binom{x}{r-i} \\ &= (-1)^r \sum_{i=0}^{r} \binom{r}{i}\binom{x-N+i-1}{i}\binom{x}{r-i} \\ &= (-1)^r \sum_{i=0}^{r} \binom{r}{i}\Delta^{r-i}\binom{x+i-N-1}{r}\Delta^{i}\binom{x}{r}, \end{aligned}$$

which, by Leibniz' rule (5.1.4) means that

$$D_r(x) = (-1)^r \Delta^r \left(\binom{x-N-1}{r}\binom{x}{r} \right). \tag{5.3.9}$$

Since each application of Δ decreases the degree by one, we can again see that D_r is a polynomial having degree r. Moreover, it is easy to see that, if $l \le r$, then

$$\Delta^{r-l}\left(\binom{x}{r}\binom{x-N-1}{r} \right)$$

is a polynomial having degree $r+l$ and zeros at

$$\{0,1,\ldots,l-1\}\cup\{N+1,N+2,\ldots,N+l\}.$$

We can use the above observation to prove the following:

Proposition 5.7. *If g is a polynomial having degree less than r, then*

$$\sum_{i=0}^{N} D_r(i)g(i)=0.$$

Proof. By using (5.1.5) and the knowledge on zeroes above, we can write

$$\begin{aligned}
&\sum_{i=0}^{N} D_r(i)g(i)\\
=\;& (-1)^r\sum_{i=0}^{N}\Delta^r\left(\binom{i}{r}\binom{i-N-1}{r}\right)g(i)\\
=\;& (-1)^{r+1}\sum_{i=0}^{N-1}\Delta^{r-1}\left(\binom{i+1}{r}\binom{i+1-N-1}{r}\right)\Delta g(i)\\
=\;& (-1)^{r+1}\sum_{i=1}^{N}\Delta^{r-1}\left(\binom{i}{r}\binom{i-N-1}{r}\right)\Delta g(i-1),
\end{aligned}$$

and repeatedly using the above reduction we finally end up at

$$\sum_{i=0}^{N} D_r(i)g(i)=-\sum_{i=r-1}^{N}\Delta\left(\binom{i}{r}\binom{i-N-1}{r}\right)\Delta^{r-1}g(i-r+1)\quad(5.3.10)$$

Since we assumed that g as degree at most $r-1$, $\Delta^{r-1}g$ is a constant, say C. Now

$$\begin{aligned}
&\sum_{i=0}^{N} D_r(i)g(i)\\
=\;& -C\sum_{i=r-1}^{N}\Delta\left(\binom{i}{r}\binom{i-N-1}{r}\right)\\
=\;& -C\left(\binom{N+1}{r}\binom{0}{r}-\binom{r-1}{r}\binom{r-1-N-1}{r}\right)=0,
\end{aligned}$$

as claimed. □

From the previous proposition it follows that the discrete Chebyshev polynomials form an orthogonal basis for P_N (the vector space of all polynomials with degree no more than N) with respect to the inner product defined by

$$\langle P_1 \,|\, P_2 \rangle_C = \sum_{i=0}^{N} P_1(i)^* P_2(i).$$

We define L_2-norm in P_N by

$$\|P\|_2 = \sqrt{\langle P \,|\, P \rangle_C},$$

and it is of course natural to resolve the values

$$\|D_r\|_2^2 = \langle D_r \,|\, D_r \rangle_C,$$

but these can be found by substituting $g(x) = D_r(x)$ in (5.3.10) and once more applying the "partial integration" to get

$$\langle D_r \,|\, D_r \rangle_C = \sum_{i=r}^{N} \binom{i}{r} \binom{i-N-1}{r} \Delta^r D_r(i-r).$$

It is easy to see that $\Delta^r D_r = \binom{2r}{r}$ is a constant, so

$$\langle D_r \,|\, D_r \rangle_C = \binom{2r}{r} \sum_{i=r}^{N} \binom{i}{r} \binom{i-N-1}{r} = \binom{2r}{r} \binom{N+1+r}{2r+1}, \tag{5.3.11}$$

as easily verified, see the Appendix.

Representation (5.3.7) can be used to reveal an interesting fact about the discrete Chebyshev polynomials. By substituting $N-x$ in (5.3.7) shows us that

$$\begin{aligned} D_r(N-x) &= \sum_{i=0}^{r} (-1)^i \binom{r}{i} \binom{x}{r-i} \binom{N-x}{i} \\ &= \sum_{i=0}^{r} (-1)^{r-i} \binom{r}{r-i} \binom{x}{i} \binom{N-x}{r-i} \\ &= (-1)^r \sum_{i=0}^{r} (-1)^i \binom{r}{i} \binom{N-x}{r-i} \binom{x}{i} \\ &= (-1)^r D_r(x), \end{aligned}$$

which is to say that the discrete Chebyshev polynomials (on interval $[0, N]$) of even (resp. odd) degree are symmetric (resp. antisymmetric) with respect to $N/2$.

The *recurrence relations* are mathematically always interesting, so it is worth studying them. The recurrence relation for the discrete Chebyshev polynomials can be found by using the standard techniques.

Theorem 5.3. *For $r \geq 2$, the discrete Chebyshev polynomials satisfy the following recurrence relation*

$$r^2 D_r = (2r-1) D_1 D_{r-1} - (N+r)(N-r+2) D_{r-2}.$$

Proof. Using (5.3.7) we see that $D_1(x) = N - 2x$. Representation (5.3.9) easily reveals that $D_r(x)$ has $(-1)^r \frac{1}{r!}\binom{2r}{r}$ as the leading coefficient (in the standard power representation). It follows that the polynomials $r^2 D_r$ and $(2r-1)D_1 D_{r-1}$ have identical leading coefficients and therefore polynomial $r^2 D_r - (2r-1) D_1 D_{r-1}$ has degree at most $r-1$. Because of that, we have a representation

$$r^2 D_r - (2r-1) D_1 D_{r-1} = \sum_{i=0}^{r-1} \alpha_i D_i \tag{5.3.12}$$

for some coefficients α_i.

Since each D_r is orthogonal to any g with degree less than r, taking inner product of both sides of (5.3.12) with any D_k such that $k \leq r-3$ implies that $\alpha_i = 0$ for each $i \leq r-3$. Hence

$$r^2 D_r - (2r-1) D_1 D_{r-1} = \alpha_{r-1} D_{r-1} + \alpha_{r-2} D_{r-2} \tag{5.3.13}$$

for some numbers α_{r-1} and α_{r-2}. From (5.3.13) we can deduce that

$$\alpha_{r-2} = \langle -(2r-1) D_1 D_{r-1} \,|\, D_{r-2} \rangle \left\| D_{r-2} \right\|^{-2}.$$

To compute the inner product above, we first notice that

$$\langle -(2r-1) D_1 D_{r-1} \,|\, D_{r-2} \rangle = \langle -(2r-1) D_1 D_{r-2} \,|\, D_{r-1} \rangle,$$

and that the degree of $-(2r-1) D_1 D_{r-2}$ is $r-1$, so we have a representation

$$-(2r-1) D_1 D_{r-2} = \beta_{r-1} D_{r-1} + \beta_{r-2} D_{r-2} + \ldots + \beta_0 D_0. \tag{5.3.14}$$

Clearly

$$\langle -(2r-1) D_1 D_{r-2} \,|\, D_{r-1} \rangle = \beta_{r-1} \left\| D_{r-1} \right\|^2,$$

and comparing the leading coefficients in (5.3.14) gives that

$$\beta_{r-1} = -2(2r-1)(r-1) \binom{2r-4}{r-2} \binom{2r-2}{r-1}^{-1},$$

which finally gives us

$$\alpha_{r-2} = -(N+r)(N-r+2).$$

To find α_{r-1} we notice that representation (5.3.7) shows that $D_r(0) = \binom{N}{r}$ and substituting $x = 0$ in (5.3.14) gives that $\alpha_{r-1} = 0$, and the claim follows immediately. $\square$

Yet another interesting property can be found by using the discrete analogues of the standard techniques.

Theorem 5.4. *The discrete Chebyshev polynomials $D_r(x)$ satisfy the following difference equation:*

$$\begin{aligned} & (x+2)(x-N+1)\Delta^2 D_r(x) + (2x - r^2 - r - N + 2)\Delta D_r(x) \\ - \; & r(r+1)D_r(x) = 0. \end{aligned}$$

Proof. Denote

$$V_r(x) = \binom{x}{r}\binom{x-N-1}{r}.$$

It is easy to verify that

$$A(x)\Delta V_r(x) = B(x)V_r(x), \tag{5.3.15}$$

where $A(x) = (x-r+1)(x-N-r)$ and $B(x) = r(2x-N-r+1)$. Taking discrete derivatives of order $r+1$ of the both sides of (5.3.15) by using Leibniz' rule (5.1.4) we get

$$\begin{aligned} & \sum_{k=0}^{r+1} \binom{r+1}{k} \Delta^{r+1-k} A(x+k) \Delta^k \Delta V_r(x) \\ = \; & \sum_{k=0}^{r+1} \binom{r+1}{k} \Delta^{r+1-k} B(x+k) \Delta^k V_r(x). \end{aligned}$$

Because $\deg(A) = 2$ and $\deg(B) = 1$, there are only three summands in the left hand side (corresponding to values $k \in \{r+1, r, r-1\}$) and two summands in the right hand size (corresponding to values $k \in \{r+1, r\}$. Evaluating the summands yields the claim straightforwardly. □

Notice that the statement of Theorem 5.4 can be also written as

$$\begin{aligned} \Delta\big((x+1)(x-N)\Delta D_r(x)\big) &= r(r+1)(\Delta D_r(x) + D_r(x)) \\ &= r(r+1)D_r(x+1). \end{aligned}$$

By denoting $f(x) = (x+1)(x-N)\Delta D_r(x)$ and using (5.1.2) we can conclude that

$$f(y) = \sum_{x=-1}^{y-1} \Delta f(x) = r(r+1) \sum_{x=-1}^{y-1} D_r(x+1) = r(r+1) \sum_{x=0}^{y} D_r(x),$$

which is to say that

$$(x+1)(x-N)\Delta D_r(x) = (r+1) \sum_{k=0}^{x} D_r(k).$$

5.4 Counterparts from Calculus

5.4.1 The Gradient, Paths

Definition 5.2. Let $f \in V_N$ be any function (recall from Chapter 4 that V_N is the vector space of all functions $f : \mathbb{F}_2^N \to \mathbb{C}$) and $B = \{\boldsymbol{e}_1, \dots, \boldsymbol{e}_N\}$ the natural basis of $\mathbb{F}_2^N$. We define the *gradient* of f as

$$\nabla f = (\Delta_{\boldsymbol{e}_1} f, \dots, \Delta_{\boldsymbol{e}_N} f) \in V_N^N.$$

An element of V_N^N is referred as to a *vector field*.

Definition 5.3. A *path* P in $\mathbb{F}_2^N$ is a sequence $\boldsymbol{x}_0$, $\boldsymbol{x}_1$, …, $\boldsymbol{x}_k$ such that $\boldsymbol{x}_{i+1} - \boldsymbol{x}_i \in B$ for each $i \in \{0, 1, \dots, k-1\}$. That is, a member $\boldsymbol{x}_i \in \mathbb{F}_2^N$ of the path P can be obtained from the previous member $\boldsymbol{x}_{i-1}$ by flipping a single bit. The *length* of the path $P : \boldsymbol{x}_0$, $\boldsymbol{x}_1$, …, $\boldsymbol{x}_k$ is defined to be k. We also say $\boldsymbol{x}_0$ and $\boldsymbol{x}_k$ are the *starting* and *ending* points of the path, and that the path P *connects* points $\boldsymbol{x}_0$ and $\boldsymbol{x}_k$.

If $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_2^N$ are any two vectors, it is clear that there exists a path $P : \boldsymbol{x} = \boldsymbol{x}_0$, $\boldsymbol{x}_1$, …, $\boldsymbol{x}_k = \boldsymbol{y}$ connecting points $\boldsymbol{x}$ and $\boldsymbol{y}$ such that $k = \mathrm{wt}(\boldsymbol{x} - \boldsymbol{y})$. Notice that there are $k!$ shortest paths connecting $\boldsymbol{x}$ and $\boldsymbol{y}$, if $k = \mathrm{wt}(\boldsymbol{x} - \boldsymbol{y})$.

5.4.2 Path Integrals

Definition 5.4. Let $P : \boldsymbol{x}_0$, $\boldsymbol{x}_1$, …, $\boldsymbol{x}_k$ be a path in $\mathbb{F}_2^N$ such that $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{e}_{j_i}$. Let also $g = (g_1, \dots, g_N) \in V_N^N$ be a vector field. We define the *path integral* of g along P to be

$$\int_P g = \sum_{i=0}^{k-1} g_{j_i}(\boldsymbol{x}_i).$$

Definition 5.5. Let $P : \boldsymbol{x}_0$, $\boldsymbol{x}_1$, …, $\boldsymbol{x}_k$ be a path of length k and $Q : \boldsymbol{y}_0$, $\boldsymbol{y}_1$, …, $\boldsymbol{y}_l$ a path of length l such that $\boldsymbol{x}_k = \boldsymbol{y}_0$. Then the *concatenation* of paths P and Q is defined as path $PQ : \boldsymbol{x}_0$, $\boldsymbol{x}_1$, …, $\boldsymbol{x}_k$, $\boldsymbol{y}_1$, …, $\boldsymbol{y}_l$ of length $k + l$.

The proof of the following lemma is straightforward.

Lemma 5.5. *If g is a vector field and P and Q are paths such that the ending point of P is the starting point of Q, then*

$$\int_{PQ} g = \int_P g + \int_Q g.$$

Definition 5.6. A vector field g is *conservative* , if

$$\int_{P_1} g = \int_{P_2} g$$

for any paths P_1 and P_2 which have $\boldsymbol{x}$ and $\boldsymbol{y}$ as the starting and ending points, respectively. If g is conservative and P a path having $\boldsymbol{x}$ and $\boldsymbol{y}$ as starting and ending points, we also denote

$$\int_P g = \int_{\boldsymbol{x}}^{\boldsymbol{y}} g.$$

Proposition 5.8. *Let $f \in V_N$. Then the gradient ∇f is a conservative vector field.*

Proof. Let $P : \boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ be a path such that $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{e}_{j_i}$. Then

$$\begin{aligned}
\int_P \nabla f &= \sum_{i=0}^{k-1} \Delta_{\boldsymbol{e}_{j_i}} f(\boldsymbol{x}_i) \\
&= \sum_{i=0}^{k-1} \big(f(\boldsymbol{x}_i + \boldsymbol{e}_{j_i}) - f(\boldsymbol{x}_i)\big) \\
&= \sum_{i=0}^{k-1} \big(f(\boldsymbol{x}_{i+1}) - f(\boldsymbol{x}_i)\big) \\
&= \sum_{i=1}^{k} f(\boldsymbol{x}_i) - \sum_{i=0}^{k-1} f(\boldsymbol{x}_i) \\
&= f(\boldsymbol{x}_k) - f(\boldsymbol{x}_0),
\end{aligned}$$

which shows that $\int_P \nabla f$ depends only on the ending points of the path P. $\square$

Corollary 5.3. *If $\nabla f \in V_N^N$ is identically zero, then $f \in V_N$ is a constant function.*

Proof. Since ∇f is a conservative vector field, function

$$F(\boldsymbol{x}) = \int_{\boldsymbol{0}}^{\boldsymbol{x}} \nabla f = f(\boldsymbol{x}) - f(\boldsymbol{0}). \tag{5.4.1}$$

is well-defined for each $\boldsymbol{x} \in \mathbb{F}_2^N$. On the other hand, since $\nabla f = 0$ everywhere, we have that $F(\boldsymbol{x}) = 0$ for each $\boldsymbol{x} \in \mathbb{F}_2^N$. By equation (5.4.1) it follows that $f(\boldsymbol{x}) = f(\boldsymbol{0})$ for each $\boldsymbol{x} \in \mathbb{F}_2^N$. $\square$

Remark 5.2. Equation

$$\int_{\boldsymbol{0}}^{\boldsymbol{x}} \nabla f = f(\boldsymbol{x}) - f(\boldsymbol{0}). \tag{5.4.2}$$

is referred as to the *Stoke's theorem* hereafter.

Corollary 5.4. *If the gradient ∇f is known, then f can be reconstructed, up to an additive constant.*

Proof. Let $F(\boldsymbol{x})$ be defined as in 5.4.1. Then F and f differ only by an additive constant, hence F can be viewed as a reconstruction of f. On the other hand, if $g \in V_N$ is another function such that $\nabla g = \nabla f$, then $\nabla(f-g) = 0$ and hence $f - g$ is a constant function by Corollary 5.3. □

There also exists a counterpart of a well-known fact from traditional calculus.

Lemma 5.6. *Let $g = (g_1, \ldots, g_N) \in V_N^N$ be a vector field. Then $g = \nabla f$ for some $f \in V_N$ if and only if $\Delta_{\boldsymbol{e}_i} g_j = \Delta_{\boldsymbol{e}_j} g_i$ for each i, $j \in \{1, \ldots, N\}$.*

Proof. Assume first that $g = \nabla f$ for some $f \in V_N$. Then

$$\Delta_{\boldsymbol{e}_i} g_j = \Delta_{\boldsymbol{e}_i} \Delta_{\boldsymbol{e}_j} f = \Delta_{\boldsymbol{e}_j} \Delta_{\boldsymbol{e}_i} f = \Delta_{\boldsymbol{e}_j} g_i$$

by Corollary 4.2.

Let us then assume that the identity $\Delta_{\boldsymbol{e}_i} g_j = \Delta_{\boldsymbol{e}_j} g_i$ holds for any $i \neq j$. Using the Fourier representations

$$\begin{aligned} g_i &= \sum_{\boldsymbol{y} \in \mathbb{F}_2^N} \widehat{g}_i(\boldsymbol{y}) W_{\boldsymbol{y}} \\ g_j &= \sum_{\boldsymbol{y} \in \mathbb{F}_2^N} \widehat{g}_j(\boldsymbol{y}) W_{\boldsymbol{y}} \end{aligned}$$

and Example 4.3 we find out that

$$\begin{aligned} \Delta_{\boldsymbol{e}_j} g_i &= -2 \sum_{\boldsymbol{y}_j = 1} \widehat{g}_i(\boldsymbol{y}) W_{\boldsymbol{y}} \\ \Delta_{\boldsymbol{e}_i} g_j &= -2 \sum_{\boldsymbol{y}_i = 1} \widehat{g}_j(\boldsymbol{y}) W_{\boldsymbol{y}} \end{aligned}$$

Since the two functions above were assumed to be equal, we must have $\widehat{g}_i(\boldsymbol{y}) = 0$ whenever $\boldsymbol{y}_i = 0$. By Lemma 4.4 there exists a function $f^{(i)} \in V_N$ such that $g_i = \Delta_{\boldsymbol{e}_i} f^{(i)}$. It remains to be shown that the same function $f^{(i)}$ can be chosen for each coordinate i, and this can be proven as follows:

We will construct a function $f \in V_N$ by defining $f(\boldsymbol{0}) = 0$ and $f(\boldsymbol{e}_i) = g_i(\boldsymbol{0})$ for each basis vector $\boldsymbol{e}_i$. So far f has been defined on $S_0 \cup S_1$, and

$$\Delta_{\boldsymbol{e}_i} f(\boldsymbol{0}) = f(\boldsymbol{e}_i) - f(\boldsymbol{0}) = g_i(\boldsymbol{0}).$$

Assume now that f has been defined on $S_0 \cup S_1 \cup \ldots \cup S_k$, where $k \geq 1$, and that $\Delta_{\boldsymbol{e}_i} f(\boldsymbol{x}) = g_i(\boldsymbol{x})$ whenever $\mathrm{wt}(\boldsymbol{x}) \leq k - 1$ and $i \notin \mathrm{supp}(\boldsymbol{x})$. We will extend the definition of f also onto S_{k+1} as follows:

If $\boldsymbol{x} \in S_{k+1}$, we can write

$$\boldsymbol{x} = \sum_{i \in \mathrm{supp}(\boldsymbol{x})} \boldsymbol{e}_i.$$

Then, for any $s \in \operatorname{supp}(\boldsymbol{x})$, we denote

$$\boldsymbol{x}_s = \sum_{i \in \operatorname{supp}(\boldsymbol{x}) \setminus \{s\}} \boldsymbol{e}_i.$$

Thus $\boldsymbol{x}_s \in S_k$ can be written as $\boldsymbol{x} = \boldsymbol{x}_s + \boldsymbol{e}_s$, and, by the hypothesis, $f(\boldsymbol{x}_s)$ has already been defined. Now, we simply define

$$f(\boldsymbol{x}) = f(\boldsymbol{x}_s) + g_s(\boldsymbol{x}_s), \tag{5.4.3}$$

but we still have to show that $f(\boldsymbol{x})$, as defined in 5.4.3, is independent of $s \in \operatorname{supp}(\boldsymbol{x})$ chosen. For that purpose, choose $r \in \operatorname{supp}(\boldsymbol{x})$ different from s, and consider vector

$$\boldsymbol{x}_{rs} = \sum_{i \in \operatorname{supp}(\boldsymbol{x}) \setminus \{r,s\}} \boldsymbol{e}_i.$$

Clearly $\boldsymbol{x}_{rs} \in S_{k-1}$, $\boldsymbol{x}_{rs} + \boldsymbol{e}_s = \boldsymbol{x}_r$, and $\boldsymbol{x}_{rs} + \boldsymbol{e}_r = \boldsymbol{x}_s$. Now

$$\begin{array}{lrcl}
 & f(\boldsymbol{x}_s) + g_s(\boldsymbol{x}_s) & = & f(\boldsymbol{x}_r) + g_r(\boldsymbol{x}_r) \\
\Longleftrightarrow & f(\boldsymbol{x}_{rs} + \boldsymbol{e}_r) + g_s(\boldsymbol{x}_s) & = & f(\boldsymbol{x}_{rs} + \boldsymbol{e}_s) + g_r(\boldsymbol{x}_r) \\
\Longleftrightarrow & f(\boldsymbol{x}_{rs} + \boldsymbol{e}_r) - f(\boldsymbol{x}_{rs}) + g_s(\boldsymbol{x}_s) & = & f(\boldsymbol{x}_{rs} + \boldsymbol{e}_s) - f(\boldsymbol{x}_{rs}) + g_r(\boldsymbol{x}_r) \\
\Longleftrightarrow & g_r(\boldsymbol{x}_{rs}) + g_s(\boldsymbol{x}_s) & = & g_s(\boldsymbol{x}_{rs}) + g_r(\boldsymbol{x}_r) \\
\Longleftrightarrow & g_s(\boldsymbol{x}_{rs} + \boldsymbol{e}_r) - g_s(\boldsymbol{x}_{rs}) & = & g_r(\boldsymbol{x}_{rs} + \boldsymbol{e}_s) - g_r(\boldsymbol{x}_{rs}) \\
\Longleftrightarrow & \Delta_{\boldsymbol{e}_r} g_s(\boldsymbol{x}_{rs}) & = & \Delta_{\boldsymbol{e}_s} g_r(\boldsymbol{x}_{rs}).
\end{array}$$

The last line is true because of the assumption. Now if $\boldsymbol{x} \in S_k$ and $i \notin \operatorname{supp}(\boldsymbol{x})$, we have that $\boldsymbol{x} + \boldsymbol{e}_i \in S_{k+1}$ and it follows that

$$f(\boldsymbol{x} + \boldsymbol{e}_i) = f(\boldsymbol{x}) + g_i(\boldsymbol{x}),$$

which implies that

$$\Delta_{\boldsymbol{e}_i} f(\boldsymbol{x}) = g_i(\boldsymbol{x}).$$

On the other hand, if $\boldsymbol{x} \in S_k$ but $i \in \operatorname{supp}(\boldsymbol{x})$, we have that $\boldsymbol{x} + \boldsymbol{e}_i \in S_{k-1}$. Then

$$\begin{aligned}
\Delta_{\boldsymbol{e}_i} f(\boldsymbol{x}) &= f(\boldsymbol{x} + \boldsymbol{e}_i) - f(\boldsymbol{x}) \\
&= f(\boldsymbol{x} + \boldsymbol{e}_i) - f(\boldsymbol{x} + \boldsymbol{e}_i + \boldsymbol{e}_i) \\
&= -\Delta_{\boldsymbol{e}_i} f(\boldsymbol{x} + \boldsymbol{e}_i) = -g_i(\boldsymbol{x} + \boldsymbol{e}_i).
\end{aligned}$$

We have already seen that g_i can be expressed as $g_i = \Delta_{\boldsymbol{e}_i} f^{(i)}$, and therefore

$$\Delta_{\boldsymbol{e}_i} g_i = \Delta_{\boldsymbol{e}_i} \Delta_{\boldsymbol{e}_i} f^{(i)} = -2\Delta_{\boldsymbol{e}_i} f^{(i)} = -2g_i$$

by Corollary 4.3. Hence

$$\begin{aligned}
\Delta_{\boldsymbol{e}_i} f(\boldsymbol{x}) &= -g_i(\boldsymbol{x} + \boldsymbol{e}_i) \\
&= -g_i(\boldsymbol{x} + \boldsymbol{e}_i) + g_i(\boldsymbol{x}) - g_i(\boldsymbol{x}) \\
&= -\Delta_{\boldsymbol{e}_i} g_i(\boldsymbol{x}) - g_i(\boldsymbol{x}) \\
&= 2g_i(\boldsymbol{x}) - g_i(\boldsymbol{x}) = g_i(\boldsymbol{x}).
\end{aligned}$$

□

Chapter 6

Approximations

By $\mathbb{R}_{\geq 0}$ we understand the set of all non-negative real numbers. Let $\|\cdot\| : V_N \to \mathbb{R}_{\geq 0}$ be a norm, ϵ is a positive number, and $f \in V_N$. We say that a function $g \in V_N$ *ϵ-approximates* f or that g *approximates* f *with threshold* ϵ, if

$$\|f - g\| \leq \varepsilon.$$

In this chapter we will study the following question: if $f \in V_N$, is it possible to approximate f by using another function g (such that $\deg(g) < \deg(f)$) with some threshold ϵ.

The norms which we will study here are basically the L_2-norm and L_∞-norm, which are formally defined as follows:

$$\|f\|_2 = \sqrt{\langle f \,|\, f \rangle} = \sqrt{\sum_{\boldsymbol{x} \in \mathbb{F}_2^N} |f(\boldsymbol{x})|^2},$$

and

$$\|f\|_\infty = \max_{\boldsymbol{x} \in \mathbb{F}_2^N} \{|f(\boldsymbol{x})|\}.$$

It is easy to see that the following inequalities hold:

$$\begin{aligned} \|f\|_2 &\leq \sqrt{2^N}\,\|f\|_\infty\,, \\ \|f\|_\infty &\leq \|f\|_2\,. \end{aligned}$$

Remark 6.1. For a constant function $f(\boldsymbol{x}) = 1$ for each $\boldsymbol{x}$ the first inequality above becomes actually an equality, and so does the second one for the natural basis functions $T_{\boldsymbol{y}}$.

6.1 Easy Restrictions for L_∞-norm

In this section we concentrate only on L_∞-norm, and the Boolean functions are assumed to have range $\{-1, 1\}$. We will first find some ultimate bounds for the threshold, and for that purpose, we begin with the following easy lemma.

Lemma 6.1. *If $f : \mathbb{F}_2^N \to \mathbb{Z}$ is an integer-valued function, then a nonzero $\widehat{f}(\boldsymbol{y})$ satisfies $\left|\widehat{f}(\boldsymbol{y})\right| \geq \frac{1}{\sqrt{2^N}}$. Moreover, if $f : \mathbb{F}_2^N \to \{-1, 1\}$ is a Boolean function, then $\left|\widehat{f}(\boldsymbol{y})\right| \geq \frac{2}{\sqrt{2^N}}$ for a nonzero $\widehat{f}(\boldsymbol{y})$.*

Proof. A Fourier coefficient $\widehat{f}(\boldsymbol{y})$ can be expressed as

$$\widehat{f}(\boldsymbol{y}) = \frac{1}{\sqrt{2^N}} \sum_{\boldsymbol{x} \in \mathbb{F}_2^N} f(\boldsymbol{x}) \chi_{\boldsymbol{x}}(\boldsymbol{y}),$$

but for an integer-valued function f, each summand in the above sum is also an integer. Therefore, $\widehat{f}(\boldsymbol{y}) \in \frac{1}{\sqrt{2^N}}\mathbb{Z}$, and the claim follows. If f is a Boolean function, then also each summand is either -1 or 1, and if $M_{\boldsymbol{y}}$ is the number of -1's in the above sum, then

$$\widehat{f}(\boldsymbol{y}) = \frac{1}{\sqrt{2^N}}(2^N - 2M_{\boldsymbol{y}}) = \frac{2}{\sqrt{2^N}}(2^{N-1} - M_{\boldsymbol{y}}),$$

hence $\widehat{f}(\boldsymbol{y}) \in \frac{2}{\sqrt{2^N}}\mathbb{Z}$, and the claim follows. □

If $f : \mathbb{F}_2^N \to \{-1, 1\}$ is a Boolean function, we always assume here that $\epsilon < 1$ to exclude the trivialities.

Remark 6.2. The above lower bound for the absolute values of nonzero Fourier coefficients is reachable for some quite naturally defined Boolean functions. Consider, for instance so-called *or-function* defined as

$$f(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{x} = \boldsymbol{0}, \\ -1, & \text{if } \boldsymbol{x} \neq \boldsymbol{0}. \end{cases}$$

Then a straightforward computation gives

$$\widehat{f}(\boldsymbol{y}) = \begin{cases} \frac{1}{\sqrt{2^N}}(2 - 2^N), & \text{if } \boldsymbol{y} = \boldsymbol{0}, \\ \frac{2}{\sqrt{2^N}}, & \text{if } \boldsymbol{y} \neq \boldsymbol{0}. \end{cases}$$

If g approximates f, then also the distance between the Fourier coefficients of g and f can be bounded. The following lemma gives a quantitative version of this fact.

Lemma 6.2. *If g ϵ-approximates f, then $\left|\widehat{f}(\boldsymbol{y}) - \widehat{g}(\boldsymbol{y})\right| \leq \sqrt{2^N}\epsilon$.*

Proof. By straightforward computation,

$$\begin{aligned} \left|\widehat{f}(\boldsymbol{y}) - \widehat{g}(\boldsymbol{y})\right| &= \left|\sum_{\boldsymbol{x} \in \mathbb{F}_2^N} (f(\boldsymbol{x}) - g(\boldsymbol{x})) W_{\boldsymbol{x}}(\boldsymbol{y})\right| \\ &\leq \sum_{\boldsymbol{x} \in \mathbb{F}_2^N} |f(\boldsymbol{x}) - g(\boldsymbol{x})| \frac{1}{\sqrt{2^N}} \leq \sqrt{2^N}\epsilon, \end{aligned}$$

which was to be shown. □

A restriction for approximations is given in the following proposition

Proposition 6.1 (Exponentially Precise Approximation). *If g approximates a Boolean function f with threshold $\epsilon < \frac{2}{2^N}$, then* $\deg(g) \geq \deg(f)$.

Proof. Let $d = \deg(f)$. Then there is a $\boldsymbol{y}$ such that $\mathrm{wt}(\boldsymbol{y}) = d$ and $\widehat{f}(\boldsymbol{y}) \neq 0$. By Lemma 6.1, $\left|\widehat{f}(\boldsymbol{y})\right| \geq \frac{2}{\sqrt{2^N}}$, and by Lemma 6.2,

$$\left|\widehat{g}(\boldsymbol{y}) - \widehat{f}(\boldsymbol{y})\right| \leq \sqrt{2^N}\epsilon < \frac{2}{\sqrt{2^N}}.$$

Therefore,

$$\begin{aligned} |\widehat{g}(\boldsymbol{y})| &= \left|\widehat{g}(\boldsymbol{y}) - \widehat{f}(\boldsymbol{y}) + \widehat{f}(\boldsymbol{y})\right| \\ &\geq \left|\widehat{f}(\boldsymbol{y})\right| - \left|\widehat{g}(\boldsymbol{y}) - \widehat{f}(\boldsymbol{y})\right| \\ &> \frac{2}{\sqrt{2^N}} - \frac{2}{\sqrt{2^N}} = 0. \end{aligned}$$

Thus $\widehat{g}(\boldsymbol{y}) \neq 0$ and hence $\deg(g) \geq d = \deg(f)$. □

On the other hand, the following example shows that there are functions that are *not approximable with any threshold* $\epsilon < 1$.

Example 6.1. Let $\boldsymbol{y} \in \mathbb{F}_2^N$ and $f : \mathbb{F}_2^N \to \{-1, 1\}$ be defined as

$$f(\boldsymbol{x}) = \sqrt{2^N} W_{\boldsymbol{y}}(\boldsymbol{x}).$$

If now $g \in V_N$ ϵ-approximates f, for some $\epsilon < 1$, then, by Lemma 6.2

$$|\widehat{g}(\boldsymbol{y})| \geq \left|\widehat{f}(\boldsymbol{y})\right| - \left|\widehat{g}(\boldsymbol{y}) - \widehat{f}(\boldsymbol{y})\right| \geq \sqrt{2^N} - \sqrt{2^N}\epsilon > 0,$$

and therefore $\widehat{g}(\boldsymbol{y}) \neq 0$, which implies that $\deg(g) \geq \deg(f) = \mathrm{wt}(\boldsymbol{y})$.

Remark 6.3. By choosing $\boldsymbol{y} = \mathbf{1}$ (all components 1) in the previous example, we have so-called *parity function*, which has value -1 if and only if $\boldsymbol{x}$ has an odd weight. Recall from the previous chapters that a function with this property can have only a constant speed-up by quantum computers.

6.2 Easy Restrictions for L_2-norm

The problem for finding good approximations with respect to L_2-norm has been mainly resolved long ago (at least for those parts we are here interested in), and this section is included here merely to make some comparisons between L_2 and L_∞ -approximations.

If $f : \mathbb{F}_2^N \to \{-1, +1\}$ is a Boolean function, we define

$$\delta = \begin{cases} 1, & \text{if } |\{\boldsymbol{x} \mid f(\boldsymbol{x}) = 1| > |\{\boldsymbol{x} \mid f(\boldsymbol{x}) = -1\}|, \text{ and} \\ -1, & \text{otherwise.} \end{cases}$$

We define the *trivial approximation* of a Boolean function f to be the function $g : \mathbb{F}_2^N \to \mathbb{C}$ for which $g(\boldsymbol{x}) = \delta$ for each $\boldsymbol{x} \in \mathbb{F}_2^N$.

Now if $f : \mathbb{F}_2^N \to \mathbb{C}$ is a Boolean function and g its trivial approximation, we have that

$$\begin{aligned} \|f - g\|_2^2 &= \sum_{\boldsymbol{x} \in \mathbb{F}_2^N} |f(\boldsymbol{x}) - g(\boldsymbol{x})|^2 \\ &= \sum_{f(\boldsymbol{x}) = \delta} |f(\boldsymbol{x}) - g(\boldsymbol{x})|^2 + \sum_{f(\boldsymbol{x}) \neq \delta} |f(\boldsymbol{x}) - g(\boldsymbol{x})|^2 \\ &\leq \frac{1}{2} \cdot 2^N \cdot 4 = 2^{N+1}. \end{aligned}$$

Notice that $\|f - g\|_\infty = 2$ always holds if g is the trivial approximation of a Boolean function f.

One of the most important results concerning L_2-approximations can be summarized as follows: best L_2-approximations of a function $f \in V_N$ can be found by ignoring the "high-order -terms" in the Fourier representation of f. To be more precise we present this fact in terms of a well known inequality:

Proposition 6.2. *Let*

$$f = \sum_{\boldsymbol{y} \in \mathbb{F}_2^N} \widehat{f}(\boldsymbol{y}) W_{\boldsymbol{y}}$$

and

$$f_1 = \sum_{\substack{\boldsymbol{y} \in \mathbb{F}_2^n \\ \mathrm{wt}(\boldsymbol{y}) \leq d}} \widehat{f}(\boldsymbol{y}) W_{\boldsymbol{y}}.$$

If $g \in V_n$ is any function of degree d, then

$$\|f - f_1\|_2 \leq \|f - g\|_2 .$$

Proof. Since $\deg(g) \leq d$, $\widehat{g}(\boldsymbol{y}) = 0$ whenever $\mathrm{wt}(\boldsymbol{y}) > d$. By Parseval's identity, we have

$$\begin{aligned} \|f - g\|_2^2 &= \sum_{\boldsymbol{y} \in \mathbb{F}_2^N} \left| \widehat{f}(\boldsymbol{y}) - \widehat{g}(\boldsymbol{y}) \right|^2 \\ &= \sum_{\substack{\boldsymbol{y} \in \mathbb{F}_2^N \\ \mathrm{wt}(\boldsymbol{y}) \leq d}} \left| \widehat{f}(\boldsymbol{y}) - \widehat{g}(\boldsymbol{y}) \right|^2 + \sum_{\substack{\boldsymbol{y} \in \mathbb{F}_2^N \\ \mathrm{wt}(\boldsymbol{y}) > d}} \left| \widehat{f}(\boldsymbol{y}) \right|^2 \\ &\geq \sum_{\substack{\boldsymbol{y} \in \mathbb{F}_2^N \\ \mathrm{wt}(\boldsymbol{y}) > d}} \left| \widehat{f}(\boldsymbol{y}) \right|^2 = \|f - f_1\|_2^2 , \end{aligned}$$

so $\|f - g\|_2 \geq \|f - f_1\|_2$, as claimed. □

Example 6.2. Consider again the or-function f of Remark 6.2. Notice that already the trivial approximation f_1 which is a constant -1 gives

$$\|f - f_1\|_2 = 2.$$

The best approximating function of f having degree at most d is then given by

$$g = \frac{1}{\sqrt{2^N}}(2 - 2^N)W_{\mathbf{0}} + \frac{2}{\sqrt{2^N}} \sum_{0<\mathrm{wt}(\boldsymbol{y})\leq d} W_{\boldsymbol{y}},$$

and

$$\|f - g\|_2^2 = \sum_{\mathrm{wt}(\boldsymbol{y})>d} \frac{4}{2^N} = \frac{4}{2^N} \sum_{i=d+1}^{N} \binom{N}{i}.$$

Since $\sum_{i=d+1}^{N} \binom{N}{i} \leq 2^N$ always, the approximation is of course better than the trivial one, but not substantially better unless d is large: If $d < N/2$, then the sum of the binomial coefficients is at least 2^{N-1}, and

$$\|f - g\|_2 \geq \sqrt{2}.$$

Let us then consider how good this approximation is with respect to L_∞-norm. The correct value $f(\mathbf{0}) = 1$, but clearly

$$\begin{aligned} g(\mathbf{0}) &= \frac{1}{2^N}(2 - 2^N) + \frac{2}{2^N} \sum_{i=1}^{d} \sum_{\boldsymbol{y}\in S_i} 1 \\ &= -1 + \frac{2}{2^N} \sum_{i=0}^{d} \binom{N}{i}, \end{aligned}$$

which shows that if we want $g(\mathbf{0})$ to be closer to 1 than to -1, we must have $d > N/2$. In fact, the formula 8.2.5 in the appendix implies that if we want $g(\mathbf{0}) \geq \epsilon$ for some fixed $\epsilon > 0$, then we must choose $d \geq N/2 + c_\epsilon\sqrt{N}$.

Thus, the approximation which is best with respect to L_2-norm cannot be good with respect to L_∞-norm, unless $d \geq N/2 + c_\epsilon\sqrt{N}$.

However, as we will see in the next chapter, it is possible to obtain approximations for the or-function having degree $c_\epsilon\sqrt{N}$, which are good with respect to L_∞-norm.

6.3 The OR-function

In this chapter, we will find some bounds for approximating a symmetric function OR. These bounds have been found in [21] by using the ordinary Chebyshev polynomials. Here we will illustrate how the same bounds can be found, in a stronger form, by using the discrete Chebyshev polynomials.

The OR-function $f : \mathbb{F}_2^N \to \{0, 1\}$ on defined as

$$f(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \mathrm{wt}(\boldsymbol{x}) = 0, \\ -1, & \text{if } \mathrm{wt}(\boldsymbol{x}) > 0. \end{cases}$$

Instead of function f, we will consider function $g = \frac{1}{2}f + \frac{1}{2}$, which has value 1 if $\mathrm{wt}(\boldsymbol{x}) = 0$, and value 0 otherwise. It is clear that an approximation for g yields an approximation for f and vice versa.

Now that g is a symmetric function, we can, by Corollary 5.1, instead of g, study a polynomial polynomial $P(x)$ defined as

$$P(x) = \begin{cases} 1, & \text{if } x = 0, \\ 0, & \text{if } x \in \{1, 2, \ldots N\}. \end{cases}$$

Polynomial P can be represented by using the discrete Chebyshev polynomials (recall Section 5.3.2) as

$$P = \sum_{n=0}^{N} P_n D_n,$$

and each coefficient P_n is revealed easily by taking the inner products of both sides:

$$\langle P \,|\, D_m \rangle_C = P_m \langle D_m \,|\, D_m \rangle_C = P_m \left\| D_m \right\|_2^2,$$

so

$$P_n \left\| D_n \right\|_2^2 = \langle P \,|\, D_n \rangle_C = \sum_{l=0}^{N} P(l) D_n(l) = D_n(0),$$

and therefore

$$P_n \left\| D_n \right\|_2^2 = \binom{N}{n},$$

by (5.3.7) or (5.3.8). By Equation (5.3.11) we have that

$$\left\| D_n \right\|_2^2 = \binom{2n}{n} \binom{N+n+1}{2n+1},$$

which implies directly that

$$\begin{aligned} P_n^2 \left\| D_n \right\|_2^2 &= \frac{N!^2}{n!^2 (N-n)!^2} \frac{n!^2}{(2n)!} \frac{(2n+1)!(N-n)!}{(N+n+1)!} \\ &= (2n+1) \frac{N!^2}{(N-n)!(N+n+1)!} \\ &= \frac{N!^2}{(2N+1)!} (2n+1) \binom{2N+1}{N-n}. \end{aligned}$$

Now if Q is an approximation of P having degree $d \leq N/2$, we can express Q as

$$Q = \sum_{n=0}^{N} Q_n D_n,$$

where $Q_n = 0$ whenever $n \geq d+1$. Using these notations we can write

$$\begin{aligned}
\|P - Q\|_2^2 &= \sum_{n=0}^{N} |P_n - Q_n|^2 \|D_n\|_2^2 \\
&= \sum_{n=0}^{d} |P_n - Q_n|^2 \|D_n\|_2^2 + \sum_{n=d+1}^{N} |P_n - Q_n|^2 \|D_n\|_2^2 \\
&\geq \sum_{n=d+1}^{N} |P_n - Q_n|^2 \|D_n\|_2^2
\end{aligned}$$

with equality if and only if $P_n = Q_n$ for each $0 \leq n \leq d$.

Let us assume hereafter that $Q_n = P_n$ if $n \leq d$, and $Q_n = 0$, if $n \geq d+1$, i.e., Q is the best degree d approximation of P with respect to L_2-norm on polynomials. It is then plain to see that

$$\begin{aligned}
\|P - Q\|_2^2 &= \sum_{n=d+1}^{N} |P_n|^2 \|D_n\|^2 \\
&= \frac{N!^2}{(2N+1)!} \sum_{n=d+1}^{N} (2n+1) \binom{2N+1}{N-n} \\
&= \frac{N!^2}{(2N+1)!} \sum_{n=0}^{N-d-1} (2N-2n+1) \binom{2N+1}{n}.
\end{aligned}$$

Denoting

$$A(n) = \sum_{k=0}^{n} \binom{2N+1}{k}$$

we can write the above sums as

$$\begin{aligned}
\|P - Q\|_2^2 &= \frac{N!^2}{(2N+1)!} \sum_{n=0}^{N-d-1} (2N-2n+1)(A(n) - A(n-1)) \\
&= \frac{N!^2}{(2N+1)!} \Big(\sum_{n=0}^{N-d-1} (2N-2n+1) A(n) \\
&\qquad - \sum_{n=1}^{N-d-1} (2N-2n+1) A(n-1) \Big) \\
&= \frac{N!^2}{(2N+1)!} \Big((2d+3) A(N-d-1) + 2 \sum_{n=0}^{N-d-2} A(n) \Big).
\end{aligned}$$

Estimation (8.2.5) in the appendix implies that

$$\sum_{i \le \frac{2N+1}{2} - a\sqrt{2N+1}} \binom{2N+1}{i} < \frac{2^{2N}}{4a^2},$$

so choosing a such that $n = \frac{2N+1}{2} - a\sqrt{2N+1}$ shows that

$$A(n) \le \frac{(2N+1)2^{2N}}{(2N-2n+1)^2}. \tag{6.3.1}$$

It follows that

$$\begin{aligned}
\sum_{n=0}^{N-d-2} A(n) &\le 2^{2N}(2N+1) \sum_{n=0}^{N-d-2} \frac{1}{(2N-2n+1)^2} \\
&= 2^{2N}(2N+1) \sum_{n=d+2}^{N} \frac{1}{(2n+1)^2} \\
&\le 2^{2N}(2N+1) \int_{d+1}^{\infty} \frac{1}{(2t+1)^2}\, dt \\
&= 2^{2N}(2N+1) \frac{1}{2(2d+3)}.
\end{aligned}$$

Substituting $n = N - d - 1$ in (6.3.1) we learn that

$$A(N-d-1) \le 2^{2N} \frac{2N+1}{(2d+3)^2},$$

which, together with the above estimates, implies that

$$\begin{aligned}
\|P - Q\|_2^2 &\le \frac{N!^2}{(2N+1)!}\Big(2^{2N}\frac{2N+1}{2d+3} + 2^{2N}\frac{2N+1}{2d+3}\Big) \\
&= \frac{N!^2 \cdot 2^{2N}}{(2N)!} \frac{2}{2d+3},
\end{aligned}$$

which is at most

$$\sqrt{\frac{\pi}{2}} \cdot \frac{2\sqrt{2N+1}}{2d+3}$$

by the Wallis inequality (see Appendix). For each $N \ge 1$ we obviously have

$$\sqrt{2N+1} \le \sqrt{3}\sqrt{N},$$

so

$$\sqrt{\frac{\pi}{2}} \cdot \frac{2\sqrt{2N+1}}{2d+3} < \sqrt{\frac{3\pi}{2}} \frac{\sqrt{N}}{d},$$

which shows that $\|P-Q\|_2 \le \epsilon$ whenever

$$\sqrt{\frac{3\pi}{2}}\frac{\sqrt{N}}{d} \le \epsilon^2,$$

which happens if and only if

$$d \ge \frac{1}{\epsilon^2}\sqrt{\frac{3\pi}{2}}\sqrt{N}. \tag{6.3.2}$$

We have now obtained the result that is d satisfies inequality (6.3.2), then

$$\|P-Q\|_2 \le \epsilon,$$

but it is clear that

$$\|P-Q\|_\infty \le \|P-Q\|_2\,,$$

so the approximation Q also satisfies $|Q(i)-P(i)| \le \epsilon$ for each $i \in \{0,1,\ldots,N\}$, provided that (6.3.2) holds.

Remark 6.4. It is possible to improve the coefficient of the above estimate: Inequality (8.2.7) in the appendix implies that

$$A(n) \le \frac{105(2N+1)^4}{(2N+1-2n)^8}2^{2N}$$

Using this instead of (8.2.5) implies that

$$\|P-Q\|_2^2 < \sqrt{\frac{\pi}{2}}\cdot\frac{840}{7}\Big(\frac{1}{\sqrt{2}}\frac{\sqrt{N+1}}{d}\Big)^7,$$

which shows that $\|P-Q\|_2 < \epsilon$, if

$$d \ge \Big(\sqrt{\frac{\pi}{2}}\cdot\frac{840}{7}\Big)^{\frac{1}{7}}\frac{1}{\sqrt{2}}\epsilon^{-\frac{2}{7}}\sqrt{N+1} \approx 1.44717\cdot\epsilon^{-\frac{2}{7}}\sqrt{N+1}.$$

Choosing $\epsilon = 1/3$ gives approximately $1.9808\sqrt{N+1}$ as the degree which allows us to approximate OR-function with threshold $\frac{1}{3}$.

The approximation degree $O(\sqrt{N})$ is optimal for OR-function (up to the multiplicative coefficient), as demonstrated by Nisan and Szegedy in [21]. The proof in article [21] was based on Markov's inequality, cf. [3]:

Theorem 6.1. *If P is a real polynomial of degree d such that $m \le P(x) \le M$ for each $x \in [0,N]$, then*

$$\left|P'(x)\right| \le \frac{d^2(M-m)}{N}$$

holds in the interval $[0,N]$.

By using the above theorem, it is quite easy to deduce that if p is a polynomial that approximates (unless otherwise stated, we choose $\epsilon = \frac{1}{3}$ as the approximation threshold hereafter) any non-constant polynomial $f : \{0, 1, \ldots, N\} \to \{0, 1\}$ in the interval $[0, N]$, then $\deg(p) = \Omega(\sqrt{N})$.

Much more precise information about the approximation degrees is available:

Theorem 6.2 (R. Paturi [23]). *Let $p : \{0, 1, \ldots, N\} \to \{0, 1\}$ be a real polynomial and*

$$\Gamma(p) = \min\{|2k - N + 1| \mid p(k) \neq p(k+1), 0 \leq k \leq N - 1\}.$$

Then the approximation degree of p is $\Theta\big(\sqrt{N(N - \Gamma(p))}\big)$.

Despite of the previous theorems, it would be mathematically desirable to find the bounds for approximation degrees with better constants than those ones that can be found in the proofs of the previous theorems.

Numerical computations suggest the following.

Conjecture 1. *If P is a real polynomial of degree d, then*

$$\frac{1}{N} \sum_{l=0}^{N-1} |P(l)| \geq \frac{1}{d+1} \max_{l \in \{0,1,\ldots,N-1\}} |P(l)|$$

If the previous conjecture were true, then it could be used to establish so-called discrete version of Markov's inequality ([3]):

Theorem 6.3. *If the above conjecture is true, P is a real polynomial of degree d, and $m \leq P(l) \leq M$ for each $l \in \{0, 1, \ldots, N\}$, then*

$$\max_{l \in \{0,1,\ldots,N-1\}} |\Delta P(l)| \leq \frac{d^2(M - m)}{N},$$

where $\Delta P(l) = P(l+1) - P(l)$ is the discrete derivative of P.

Proof. Without loss of generality we can assume that $P(0) \leq P(1)$, for otherwise we could consider $-P$ instead of P. We first divide the set $\{0, 1, 2, \ldots, N\}$ into sets $I_0 = \{0, 1, \ldots, n_1 - 1\}$, $I_1 = \{n_1, n_1 + 1, \ldots, n_2 - 1\}$, $\ldots$, $I_k = \{n_k, n_k + 1, \ldots, N\}$ of consecutive integers as follows: n_1 is chosen such that

$$P(0) < P(1) < \ldots < P(n_1 - 1),$$

but $P(n_1 - 1) \geq P(n_1)$, i.e., n_1 is the first point where the strict growth of P ceases (notice that here we are interested only of the values of P at integer points). Then n_2 is chosen such that

$$P(n_1) > P(n_1 + 1) > \ldots > P(n_2 - 1),$$

but $P(n_2-1) \leq P(n_2)$, i.e, n_2 is first to indicate the end of strict decreasing. Then n_3 is chosen to indicate the end of strict growth, etc.

Notice that choosing the points like this allows also some of the sets be singletons. We will now show that each interval induced by sets I_j contain a zero of polynomial ΔP. Assume that $2 \mid j$ (case $2 \nmid j$ is treated similarly) and that I_j is not singleton. Then

$$P(n_j) < P(n_j+1) < \ldots < P(n_{j+1}-1),$$

and $P(n_{j+1}-1) \geq P(n_{j+1})$. But this is to say that $\Delta P(n_{j+1}-1) \leq 0$. Since I_j is not a singleton, it contains point $n_{j+1}-2$, and $P(n_{j+1}-2) < P(n_{j+1}-1)$, which is to say that $\Delta P(n_{j+1}-2) > 0$. It follows that there must be a zero of ΔP in $(n_{j+1}-2, n_{j+1}-1]$.

If I_j is singleton, then necessarily $n_{j+1} = n_j + 1$ and $P(n_j+1) = P(n_j)$, which is to say that $\Delta P(n_j) = 0$. As a conclusion we get that for each I_j, there is a zero of ΔP, and therefore $k \leq \deg(\Delta P) = d - 1$.

Now if $l \in \{1, \ldots, n_1 - 1\}$, then $P(l-1) < P(l)$, i.e, $\Delta P(l-1) > 0$, and if $l \in \{n_1, n_1+1 \ldots, n_2-1\}$, then $P(l-1) \geq P(l)$, i.e., $\Delta P(l-1) \leq 0$. Continuing in the same way, we see that

$$\begin{aligned}
& \sum_{l=0}^{N-1} |\Delta P(l)| = \sum_{l=1}^{N} |\Delta P(l-1)| \\
= \;& \sum_{l=1}^{n_1-1} \Delta P(l-1) - \sum_{l=n_1}^{n_2-1} \Delta P(l-1) + \ldots + (-1)^k \sum_{l=n_k}^{N} \Delta P(l-1) \\
= \;& (P(n_1-1) - P(0)) - (P(n_2-1) - P(n_1-1)) \\
& + \ldots + (-1)^k (P(N) - P(n_k-1)) \\
\leq \;& (k+1)(M-m) \leq d(M-m).
\end{aligned}$$

By the above Conjecture, we have that

$$\sum_{l=0}^{N-1} |\Delta P(l)| \geq \frac{N}{d} \max_{l \in \{0,1,\ldots,N-1\}} |P(l)| .$$

Combining the two above estimates yields the claim directly. □

The above result (based on an unproven conjecture) could be used directly for finding a general bound for the approximation degree.

Theorem 6.4. *Let P be a polynomial that ϵ-approximates a non-constant function $f : \{0, 1, \ldots, N\} \to \{0, 1\}$. If Conjecture 1 holds, we have*

$$\deg P \geq \sqrt{\frac{1-2\epsilon}{1+2\epsilon} N}.$$

Proof. Let $d = \deg(P)$. Since P approximates f within threshold ϵ, we have $-\epsilon \leq P(l) \leq 1 + \epsilon$ for each $l \in \{0, 1, \ldots, N\}$. Since f is not constant, there must be some k such that $P(k)$ lies in the proximity of 0 and $P(k+1)$ in the proximity of 1 or vice versa. In any case, $|\Delta P(k)| \geq 1 - 2\epsilon$ for some $k \in \{0, 1, \ldots, N-1\}$. By the previous theorem we have that

$$1 - 2\epsilon \leq \max |\Delta P(k)| \leq \frac{d^2(1+2\epsilon)}{N},$$

and the claim follows directly. $\square$

Chapter 7

Open Questions

Many problems still remain open, and the list below is only to mention a few.

Problem 1. *Is Conjecture 1 true?*

Problem 2. *A more demanding version of Conjecture 1 can be stated as follows: It P is a real polynomial which satisfies $-1 \leq P(l) \leq 1$ for each $l \in \{0,1,2\ldots,N\}$, then*

$$\left|(\Delta P(l))^2((l+1+a)(N-l+b))\right| \leq C \deg(P)^2$$

for each $l \in \{0,1,2\ldots,N-1\}$, where $a,b \in [0,1]$ and C are constants. Is this true?

Problem 3. *Is it true that the cutting the hybrid basis representation*

$$f = \sum_{\boldsymbol{y}\in\mathbb{F}_2^N} f_{\boldsymbol{y}} B_{\boldsymbol{y}}$$

always gives the "near-optimal" approximation with respect to L_∞-norm? That is, given a threshold $\epsilon = \frac{1}{3}$, we write

$$f_1 = \sum_{\substack{\boldsymbol{y}\in\mathbb{F}_2^N \\ \mathrm{wt}(\boldsymbol{y})\leq d}} f_{\boldsymbol{y}} B_{\boldsymbol{y}}$$

for smallest d for which $\|f-f_1\|_\infty \leq \epsilon$. Is it then true that there exists a constant C independent of N such that if $\|f-g\|_\infty \leq \|f-f_1\|_\infty$, then $\deg(g) \geq Cd = C\deg(f_1)$*?*

Problem 4. *How relevant is a thesis like this one?*

Chapter 8

Appendix

8.1 Some Formulae on Binomial Coefficients

An identity

$$\binom{N}{k} = \binom{N-1}{k} + \binom{N-1}{k-1} \tag{8.1.1}$$

is easy to verify by a direct calculation, but a recursive use of (8.1.1) leads into a an interesting equation, known as *Vandermonde's convolution*.

Proposition 8.1 (Vandermonde's convolution). *If l is a nonnegative integer, then*

$$\binom{N}{k} = \sum_{j=0}^{l} \binom{l}{j}\binom{N-l}{k-j}.$$

Proof. For $l = 0$ the claim is trivial and for $l = 1$ the claim is exactly the identity (8.1.1). Assuming the claim true for some l, we can write, by using (8.1.1)

$$\begin{aligned}
\binom{N}{k} &= \sum_{j=0}^{l} \binom{l}{j}\binom{N-l}{k-j} \\
&= \sum_{j=0}^{l} \binom{l}{j}\binom{N-l-1}{k-j} + \sum_{j=0}^{l} \binom{l}{j}\binom{N-l-1}{k-j-1} \\
&= \sum_{j=0}^{l} \binom{l}{j}\binom{N-l-1}{k-j} + \sum_{j=1}^{l+1} \binom{l}{j-1}\binom{N-l-1}{k-j} \\
&= \binom{N-(l+1)}{k} + \sum_{j=1}^{l} \binom{l+1}{j}\binom{N-(l+1)}{k-j} + \binom{N-(l+1)}{k-(l+1)} \\
&= \sum_{j=0}^{l+1} \binom{l+1}{j}\binom{N-(l+1)}{k-j},
\end{aligned}$$

which proves the claim. □

The proof of equation (5.3.11).

$$
\begin{aligned}
\sum_{i=r}^{N} \binom{i}{r}\binom{i-N-1}{r} &= \sum_{i=r}^{N} \Delta\binom{i}{r+1}\cdot\binom{i-N-1}{r} \\
= -\sum_{i=r}^{N}\binom{i+1}{r+1}\cdot\Delta\binom{i-N-1}{r} &= -\sum_{i=r}^{N}\Delta\binom{i+1}{r+2}\cdot\binom{i-N-1}{r-1} \\
= \dots &= (-1)^r\sum_{i=r}^{N}\Delta\binom{i+r}{2r+1}\cdot\binom{i-N-1}{r-r} \\
&= (-1)^r\binom{N+1+r}{2r+1}.
\end{aligned}
$$

□

Theorem 8.1 (Wallis inequality).

$$\pi N \le \left(\frac{2^{2N}(N!)^2}{(2N)!}\right)^2 \le \pi\frac{2N+1}{2}.$$

The derivation of Wallis inequality is well known, but it is included here for the sake of completeness.

Proof. Let n be a nonnegative integer and define

$$I_n = \int_0^{\frac{\pi}{2}} \sin^n t\, dt.$$

Clearly $I_0 = \frac{\pi}{2}$, $I_1 = 1$, and for $n \ge 2$ we can evidently write

$$I_n = -\int_0^{\frac{\pi}{2}} \sin^{n-1} t \frac{d}{dt}\cos t\, dt,$$

which, by partial integration, gives

$$
\begin{aligned}
I_n &= -\Big/_{0}^{\frac{\pi}{2}} \sin^{n-1} t\cos t\, dt + (n-1)\int_0^{\frac{\pi}{2}} \sin^{n-2} t\, \cos^2 t\, dt \\
&= (n-1)\int_0^{\frac{\pi}{2}} \sin^{n-2} t(1-\sin^2 t)\, dt. \\
&= (n-1)(I_{n-2}-I_n),
\end{aligned}
$$

which implies that

$$I_n = \frac{n-1}{n} I_{n-2}. \tag{8.1.2}$$

For an even $n = 2k$ recursive use of (8.1.2) gives

$$\begin{aligned} I_{2k} &= \frac{2k-1}{2k} \cdot I_{2k-2} \\ &= \frac{2k-1}{2k} \cdot \frac{2k-3}{2k-2} \cdot \ldots \cdot \frac{3}{4} \cdot \frac{1}{2} \cdot I_0 \\ &= \frac{2k}{2k} \cdot \frac{2k-1}{2k} \cdot \frac{2k-2}{2k-2} \cdot \frac{2k-3}{2k-2} \cdot \ldots \cdot \frac{4}{4} \cdot \frac{3}{4} \cdot \frac{2}{2} \cdot \frac{1}{2} \cdot \frac{\pi}{2} \\ &= \frac{(2k)!}{2^{2k}(k!)^2} \cdot \frac{\pi}{2}. \end{aligned}$$

Similarly, for an odd $n = 2k+1$ we get

$$I_{2k+1} = \frac{2^{2k}(k!)^2}{(2k+1)(2k)!}.$$

The equations above reveal that

$$I_{2k} I_{2k+1} = \frac{\pi}{2(2k+1)},$$

and using (8.1.2) once more gives that

$$I_{2k} I_{2k-1} = \frac{\pi}{4k}.$$

Now that $I_n \leq I_{n+1}$, we have

$$\frac{1}{I_{2k} I_{2k-1}} \leq \frac{1}{I_{2k}^2} \leq \frac{1}{I_{2k} I_{2k+1}}. \tag{8.1.3}$$

substituting the formulae obtained above into (8.1.3) gives

$$\frac{4k}{\pi} \leq \left(\frac{2}{\pi}\right)^2 \left(\frac{2^{2k}(k!)^2}{(2k!)}\right)^2 \leq \frac{2(2k+1)}{\pi},$$

which yields the claim immediately. □

Wallis inequality gives directly an estimation for the binomial coefficient $\binom{2N}{N}$:

$$\frac{2^{2N}}{\sqrt{\pi(N+\frac{1}{2})}} \leq \binom{2N}{N} \leq \frac{2^{2N}}{\sqrt{\pi N}}$$

8.2 Partial Sums of the Binomial Coefficients

Substituting $x = 1$ in expression

$$(1+x)^N = \sum_{i=0}^{N} \binom{N}{i} x^i \tag{8.2.1}$$

gives us a familiar equality

$$2^N = \sum_{i=0}^{N} \binom{N}{i},$$

whereas differentiating both sides of (8.2.1) shows that

$$N(1+x)^{N-1} = \sum_{i=1}^{N} i \binom{N}{i} x^{i-1}, \tag{8.2.2}$$

and a substitution $x = 1$ gives

$$\frac{N}{2} 2^N = \sum_{i=0}^{N} i \binom{N}{i}.$$

On the other hand, multiplying (8.2.2) by x, differentiating, and substituting $x = 1$ shows that

$$\frac{N(N+1)}{4} 2^N = \sum_{i=0}^{N} i^2 \binom{N}{i}. \tag{8.2.3}$$

Continuing the same procedure, we can find a closed form for

$$\sum_{i=0}^{N} i^k \binom{N}{i}$$

for each $k \geq 0$. Multiplying (8.2.2) by a, (8.2.2) by b, and (8.2.3) by c, and adding the equalities together gives us

$$(a + b\frac{N}{2} + c\frac{N(N+1)}{4})2^N = \sum_{i=0} (a + bi + ci^2) \binom{N}{i}. \tag{8.2.4}$$

We attempt to choose a, b, and c in such a way that $a + bx + cx^2$ would be negative everywhere except in interval $[\frac{N}{2} - k, \frac{N}{2} + k]$. It turns out that choice $a = \frac{4k^2 - N^2}{4}$, $b = N$, and $c = -1$ will do. Then f also attains its maximum at $x = \frac{N}{2}$, where $f(\frac{N}{2}) = k^2$. With these choices, the left hand side of (8.2.4) becomes

$$\frac{4k^2 - N}{4} 2^N,$$

and therefore

$$\frac{4k^2 - N}{4} 2^N < \sum_{\frac{N}{2} - k < i < \frac{N}{2} + k} k^2 \binom{N}{i},$$

which implies that

$$\sum_{\frac{N}{2} - k < i < \frac{N}{2} + k} \binom{N}{i} > (1 - \frac{N}{2k^2}) 2^N. \tag{8.2.5}$$

By applying the same procedure we can also establish the following inequalities:

$$\sum_{\frac{N}{2}-k<i<\frac{N}{2}+k} \binom{N}{i} > (1 - \frac{3N^2}{16k^4})2^N, \tag{8.2.6}$$

$$\sum_{\frac{N}{2}-k<i<\frac{N}{2}+k} \binom{N}{i} > (1 - \frac{15N^3}{64k^6})2^N, \tag{8.2.7}$$

when $N \geq 6$, and

$$\sum_{\frac{N}{2}-k<i<\frac{N}{2}+k} \binom{N}{i} > (1 - \frac{105N^4}{256k^8})2^N. \tag{8.2.8}$$

Index

Bibliography

[1] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf: *Quantum lower bounds by polynomials*. In Proceedings of 39th IEEE FOCS, 352–361 (1998).

[2] Charles H. Bennett: *Logical Reversibility of Computation*, IBM Journal of Res. Development 17, 525–532 (1973).

[3] E. W. Cheney: *Introduction to Approximation Theory*, McGraw-Hill (1966).

[4] D. Deutsch: *Quantum theory, the Church-Turing Principle and the Universal Quantum Computer*, Proceedings of the Royal Society of London A 400, 97–117 (1985).

[5] D. Deutsch: *Quantum computational networks*, Proceedings of the Royal Society of London A 425, 73–90 (1989).

[6] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser. *A limit on the speed of quantum computation in determining parity*, quant-ph/9802045. xxx.lanl.gov Online Preprint (1998).

[7] Richard P. Feynman: *Simulating Physics with Computers*, International Journal of Theoretical Physics 21:6/7, 467–488 (1982).

[8] Lov K. Grover: *A fast quantum mechanical algorithm for database search*, Proceedings of STOC, 212–219 (1996).

[9] Mika Hirvensalo: *The Reversibility in Quantum Computation Theory*, Proceedings of the 3rd International Conference Developments in Language Theory – DLT'97, Ed.: Symeon Bozapalidis, Aristotle University of Thessaloniki, 203–210 (1997).

[10] Mika Hirvensalo: *On Quantum Computation*, Licentiate's thesis, University of Turku (1997).

[11] Mika Hirvensalo: *Copying quantum computer makes NP-complete problems tractable*, Proceedings of MCU'98, Vol II, edited by M. Margenstern (1998).

[12] Mika Hirvensalo: *Quantum Computing*, Springer (2001).

[13] Mika Hirvensalo: *An Introduction to Quantum Computing*, In G. Păun, G. Rozenberg, A. Salomaa (eds.): Current Trends in Theoretical Computer Science – Entering the 21st Century, World Scientific, 643–663 (2001).

[14] Mika Hirvensalo: *Some Open Problems Related to Quantum Computing*, Bulletin of the EATCS 74, 154–170 (2001).

[15] Mika Hirvensalo: *Quantum Computing – Facts and Folklore*, International Journal of Natural Computing 1, 135–155 (2002).

[16] Mika Hirvensalo: *Computing with Quanta – Impacts of Quantum Theory on Computation*, Theoretical Computer Science 287:1, 267–298 (2002).

[17] Mika Hirvensalo: *Universality and Quantum Computing*, Bulletin of the EATCS 78, 199–203 (2002).

[18] M. Y. Lecerf: *Récursive insolubilité de l'équation générale de diagonalisation de deux monomorphismes de monoïdes libres* $\varphi x = \psi x$, Comptes Rendus 257 2940–2943 (1963).

[19] W. E. Milne: *Numerical Calculus.* Princeton University press, second printing (1950).

[20] Isaac Newton: *Philosophiae naturalis principia mathematica* (1687). Reprinted at Harvard University Press (1972).

[21] Noam Nisan and Mario Szegedy: *On the Degree of Boolean Functions as Real Polynomials.* Computational Complexity 4:4, 301–313 (1994).

[22] C. H. Papadimitriou: *Computational Complexity*, Addison-Wesley Publishing Company, Inc. (1994).

[23] R. Paturi: *On the Degree of Polynomials that Approximate Symmetric Boolean Functions.* Proceedings of STOC'92, 468–474 (1992).

[24] Peter W. Shor: *Algorithms for Quantum Computation: Discrete Log and Factoring*, Proceedings of the 35th Annual Symposium on the Foundations of Computer Science, 116–123 (1994).

[25] J. T. Schwartz: *Fast probabilistic Algorithms for verification of polynomial identities.* J. Assoc. Computing Machinery 27, 701–717 (1980).

[26] Hans-Ulrich Simon: *A tight* $\Omega(\log\log n)$*-bound on the time for parallel RAM's to compute nondegenerated Boolean functions*, Lecture Notes in Computer Science 158, 439–444 (1983).

[27] B. L. van der Waerden: *Sources of quantum mechanics*, North-Holland (1967).

[28] Ronald de Wolf: *Quantum Computing and Communication Complexity*, Ph.D thesis, university of Amsterdam (2001).