The Discrete Algebra of the Fourier Transform

Gabriel Peyré
CNRS & DMA
PSL University, École Normale Supérieure
gabriel.peyre@ens.fr

November 1, 2020

Forewords

There are many books on the Fourier transform; however, few are aimed at a multidisciplinary audience. Writing a book for engineers with algebraic concepts is a real challenge, as much, if not more, than writing an algebra book that puts the finger on the applications of the theories encountered. This is the challenge that this book has attempted to meet. Thus, each reader will be able to create a “à la carte” program and draw from statements or computer programs specific elements to establish their knowledge in the field, or apply them to more concrete problems.

The presentation is intentionally very detailed and requires little prior knowledge, mentioned at the beginning of the chapters concerned. The reader may occasionally need some advanced notions on finite groups as well as some familiarity with group actions. I did not hesitate to repeat the important definitions and notations. For example, the notion of convolution, approached from many angles (abelian group, signal processing, non-commutative group), is each time placed in its context. Thus, the different paragraphs, although following a logical progression, have a real unity and can be read in a non-linear fashion.

The first chapter uses the language of group theory to explain the main concepts and demonstrate the statements that will be used later. The second chapter applies the results obtained to various problems, and constitutes a first contact with fast algorithms (*Walsh transform* for example). The third chapter is an exposition on the discrete Fourier transform. Even if it reinvests the results of the first chapter, it can be read for example by a computer scientist wishing to understand the mechanisms of the algorithms of discrete transforms. The fourth chapter presents various applications of the discrete Fourier transform, and constitutes an essential complement to the previous chapter, to fully understand the mechanisms involved as well as their use in practical situations. The fifth chapter presents more original ideas and algorithms around the Fourier transform, giving rise to numerous applications. The sixth chapter requires some more advanced knowledge, especially some familiarity with finite field theory. It studies valued transforms in a finite field, and presents applications to corrective codes. The last two chapters (the most difficult), are of a more algebraic nature, and propose to generalize the constructions already carried out in the case of finite non-commutative groups. The seventh chapter exposes the theory of linear representations. The eighth and final chapter applies this theory in both theoretical (study of the simplicity of groups) and practical (spectral analysis) fields.

A number of computer programs are presented; they are written in Matlab for the most part, and in Maple for those which require algebraic manipulations (calculations in finite fields, etc.). Although these are commercial softwares, free software with very similar syntax exists, such as for instance Scilab, Octave and Mupad. The choice of a particular language to implement the algorithms is obviously debatable, but the use of Matlab and Maple seems quite natural, these softwares allowing to quickly test the written programs. We can then translate them into a compiled and faster language, such as C or C++.

Contents

Chapter 1

Fourier transform on a finite group

In this first chapter, we will approach the study of the Fourier transform from an original angle, that of group theory. Fourier theory, whether viewed from an algebraic point of view or not, consists above all in the analysis of functions. Here we must take the word analysis literally, in its etymological sense. It's about breaking down complex data into a simpler form. It will therefore be a question of constructing a systematic means to obtain this decomposition, and this is precisely where the Fourier tools come into play. To achieve this decomposition efficiently, we must still use a certain amount of a priori information about the functions we are studying. This first chapter will relate to the study of functions on a finite group; the "elementary bricks" decomposition carried out by the Fourier analysis will result from the symmetries inherent in the group structure.

The most elementary framework to carry out this project is that of finite commutative groups, since one does not have to worry about either the regularity of the functions encountered, or the convergence of the manipulated series (since they are finite!). Of course, one will be tempted to cry scandal as the work then accomplished seems simplistic compared to the "analytic" theory of Fourier series. However, this study makes it possible to bring new points of view and to pose new questions which will be addressed in the next chapters.

- How can the study of the Fourier transform on a finite group help us to understand the construction of the continuous Fourier transform?
- How does the Fourier transform on finite groups join the discrete Fourier transform?
- What uses can we make of the Fourier transform on a finite group? How to build efficient algorithms, and how to implement them?
- Finally, what happens to this theory when we try to apply it to non-commutative groups? This question will motivate the introduction of new tools, described in detail in the last two chapters.

It is to this set of questions that we will try to answer. The methods implemented are multiple, they often borrow from several mathematical disciplines.

There are many references to the subject of duality on finite groups. There is of course the book of J.P.Serre [60], but also for example that of Warusfel [69], for a more detailed presentation. For an introduction to the Fourier transform on a commutative group, we can look at the work of Dym and Mac Kean [28].

1.1 Dual of a finite group

The aim of this book is to study, from an algebraic point of view, the functions with complex values whose starting space is a finite group denoted by G. It is a question of making maximum use of the properties of the group to obtain interesting functional decompositions. The basic idea of this chapter, the one which will guide our reflections until the end of this book, consists in studying how we can represent a function on a group G. The generally most common way to think about a function $f : G \to \mathbb{C}$ is to consider the set of its values $f(g)$ for $g \in G$. The major disadvantage of this representation is that it does not exploit the structure of our group G at all. In a way, it is a universal representation, which does not depend at all on the group that one has chosen. To study a function efficiently, it seems logical to build a new representation that exploits the symmetries that can be found in a group G. The simplest example of these symmetries is the cyclic character of the group $\mathbb{Z}/n\mathbb{Z}$, but we can of course consider more complex constructions.

1.1.1 Definitions

To understand how a function can be more or less simple to represent, we will first approach the simplest functions, those which do not oppose any resistance to the structure of the starting group G. We are therefore going to focus on the functions that carry the structure of the group. These functions are the morphisms of the group G into a group of the arrival set, that is, a subgroup of $\mathbb{C}^*$. We will therefore introduce the appropriate definitions.

Definition 1 (Characters and dual of a group)**.** *Let G a finite group. By definition, a characterχ is a morphism from the group G into the multiplicative group $\mathbb{C}^*$. We denote by $\hat{G}$ the set of characters, which we call the dual of G.*

$\hat{G}$ is a group for multiplication of applications. It is recalled that this multiplication is defined as follows.

$$\forall (\chi_1,\, \chi_2) \in \hat{G}^2, \quad \chi_1\chi_2 : x \mapsto \chi_1(x)\chi_2(x).$$

We will see, in particular in Paragraph 1.2.3, that the duality over a finite group has many points in common with the duality between vector spaces. In the next chapter, more precisely in Paragraph 2.3.1, we will even see that in certain cases, this rapprochement can become an identity between the two structures (of group and of vector space). The duality on a finite group then makes it possible to demonstrate interesting linear properties. In the meantime, let's start by studying the image of a character $\chi \in \hat{G}$.

Proposition 1. *Let G be a finite group of cardinal $|G| = n$. The elements of $\hat{G}$ are in fact the morphisms of G in the group of the n^{ith} roots of the unit,*

$$\mathbb{U}_n = \left\{ \exp\left(\frac{2\imath k\pi}{n}\right) \Bigm\backslash 0 \le k < n \right\}.$$

In particular,

$$\forall g \in G, \quad |\chi(g)| = 1, \quad \chi(g^{-1}) = \chi(g)^{-1} = \overline{\chi(g)},$$

where we denote by $|z|$ the modulus of a complex number z, and $\overline{z}$ its conjugate.

Proof. Denote by 1 the neutral element of G. Note that for any element $g \in G$, we have $g^n = 1$. This therefore implies, for all $\chi \in \hat{G}$, that $\chi(g)^n = \chi(g^n) = 1$, which means that χ is at values in the set of n^{th} roots of the unit. □

Remark 1. It follows that in particular, $\hat{G}$ is a finite group (because there is only a finite number of maps of G in $\mathbb{U}_n$, which are finite sets) , commutative. Moreover, every element $\chi \in \hat{G}$ is constant over the conjugation classes of G, since

$$\forall (g,\, h) \in G^2, \quad \chi(h^{-1}gh) = \chi(h)^{-1}\chi(g)\chi(h) = \chi(e)\chi(g) = \chi(g). \tag{1.1}$$

Definition 2 (Function space on G)**.** *We denote by $\mathbb{C}[G]$ the set of functions of G in $\mathbb{C}$. The notation $\mathbb{C}[G]$ will be explained in Paragraph 1.4.2. It is a vector space on $\mathbb{C}$. We define a Hermitian dot product, by*

$$\forall (f,\, g) \in \mathbb{C}[G]^2, \quad \langle f,\, g\rangle \overset{\text{def.}}{=} \frac{1}{|G|}\sum_{x\in G} f(x)\overline{g(x)}. \tag{1.2}$$

We also define a norm $\|\cdot\|_2$ on $\mathbb{C}[G]$ by $\|f\|_2^2 \overset{\text{def.}}{=} \langle f,\, f\rangle$.

Remark 2. The Hermitian product that we have just defined on $\mathbb{C}[G]$ presents strong similarities with the one that can be defined between two functions of $L^2(\mathbb{R})$ as follows:

$$\forall (f,\, g) \in L^2(\mathbb{R})^2, \quad \langle f,\, g\rangle = \int_{\mathbb{R}} f(x)\overline{g(x)}\mathrm{d}x.$$

The main difference is the change of the sum to integral. One of the common properties of these two dot products is translational invariance. Indeed, if we denote by $T_y(f)$ the function $x \in G \mapsto f(xy) \in G$ (or its continuous analogue $T_y(f) = f(\cdot + y)$), we have

$$\langle T_h(f),\, T_h(g)\rangle = \langle f,\, g\rangle.$$

This property will be constantly used thereafter, among other things to demonstrate the orthogonal relations between characters.

In order to study the functions of $\mathbb{C}[G]$, we will introduce a canonical basis. The decomposition in this base corresponds to the standard way of representing a function of a set in $\mathbb{C}$.

Proposition 2. *A basis of* $\mathbb{C}[G]$ *is given by the following* $(\delta_g)_{g\in G}$ *functions:*

$$\delta_g(h) \overset{\text{def.}}{=} \left\{ \begin{array}{lll} 1 & \textit{if} & h = g \\ 0 & \textit{if} & h \neq g \end{array} \right. . \tag{1.3}$$

In particular, $\mathbb{C}[G]$ *is a vector space of dimension* $n = |G|$ *on* $\mathbb{C}$.

Proof. We immediately check that the family is orthonormal for the Hermitian product (1.2). As these functions are not zero, this implies that they form a free family of $\mathbb{C}[G]$. The fact that this family is also a generator comes from the canonical decomposition

$$\forall f \in \mathbb{C}[G], \quad f = \sum_{g\in G} f(g)\delta_g, \tag{1.4}$$

which ends the demonstration. □

This proposition allows to immerse G in $\mathbb{C}[G]$ in a canonical way by $g \mapsto \delta_G$. Moreover, we have seen that any function $f \in \mathbb{C}[G]$ decomposes in the base $\{\delta_g\}_{g\in G}$, this is what the equation expresses (1.4). This decomposition is apparently very simple. We will however see in Paragraph 1.4.3, with the notion of *convolution*, that it does not facilitate calculations at all. This is why we are going to look for a database which has the following two properties.

- It must be easy to use (the decomposition in this base must be easy to calculate).
- It must have interesting properties for the algebraic operations that we want to use (linear combination, product, and product of convolution of functions).

The character family, formed from the elements of $\hat{G}$, might be a good candidate; it remains to be shown that she possesses the required qualities.

1.1.2 Dual of a cyclic group

Before launching into the general study of duality on any group, let's take the time to see how all this behaves in the simplest case, that of a cyclic group (whose archetype is $\mathbb{Z}/n\mathbb{Z}$, for a given integer n). In fact, this example is of prime importance, on the one hand because in practice, it is the structure that we meet most often (we will see in chapter 3 that unidimensional calculations in signal processing use the structure of $\mathbb{Z}/n\mathbb{Z}$), and secondly because we will use these results to demonstrate the general case.

Proposition 3 (The cyclic case)**.** *Let* $G = \{1,\, g_0,\, g_0^2, \dots,\, g_0^{n-1}\}$ *be a cyclic group of cardinal* n *and generator* g_0*. Let* ω *be a primitive root* n^{th} *of the unit, for example* $\omega = e^{\frac{2\imath\pi}{n}}$*. The elements of* $\hat{G}$ *are of the form, for* $j \in \{0, \dots, n-1\}$,

$$\chi_j : \left\{ \begin{array}{ccc} G & \longrightarrow & \mathbb{C}^* \\ g = g_0^k & \longmapsto & (\omega^j)^k = e^{\frac{2\imath\pi jk}{n}} \end{array} \right. .$$

In particular, we have $G \simeq \hat{G}$.

Proof. To determine a character $\chi \in \hat{G}$, we need to calculate the value of $\chi(g_0^k)$, for $k \in \{0, \ldots, n-1\}$, which gives

$$\chi(g_0^k) = (\omega^j)^k = \omega^{jk}.$$

In this equality, we have noted $\omega^j \overset{\text{def.}}{=} \chi(g_0)$, with $0 \le j \le n-1$, since, as we have seen in the proposition 1, this quantity is a n^{th} root of the unit. So our character $\chi \in \hat{G}$ is indeed one of $\{\chi_0, \ldots, \chi_{n-1}\}$. Conversely, we see that, for $j \in \{0, \ldots, n-1\}$, the maps χ_j are indeed morphisms from G into $\mathbb{C}^*$, so are indeed elements of $\hat{G}$.
Finally, if we identify the elements of $\mathbb{Z}/n\mathbb{Z}$ and their representatives in $\{0, \ldots, n-1\}$, we define an application $\psi : j \mapsto \chi_j$ from $\mathbb{Z}/n\mathbb{Z}$ in $\hat{G}$. We saw that this application was surjective. On the other hand, this application is injective (it suffices to evaluate $\chi_j = \psi(j)$ in g_0) and it is a morphism (elementary verification). It is therefore an isomorphism and therefore $\hat{G}$ is isomorphic to $\mathbb{Z}/n\mathbb{Z}$, itself isomorphic to G. $\square$

The figure 1.1 shows the first four characters of the group $\mathbb{Z}/12\mathbb{Z}$. On the x-axis, we have denoted $\{0, \ldots, 11\}$ the representatives of the group $\mathbb{Z}/12\mathbb{Z}$, and the character values are denoted $*$. The top line shows the real parts of the characters, and the bottom line the imaginary parts. We can see that the points are regularly spaced along the curves of equations $y = \cos\left(\frac{2\pi}{N}x\right)$ and $y = \sin\left(\frac{2\pi}{N}x\right)$.

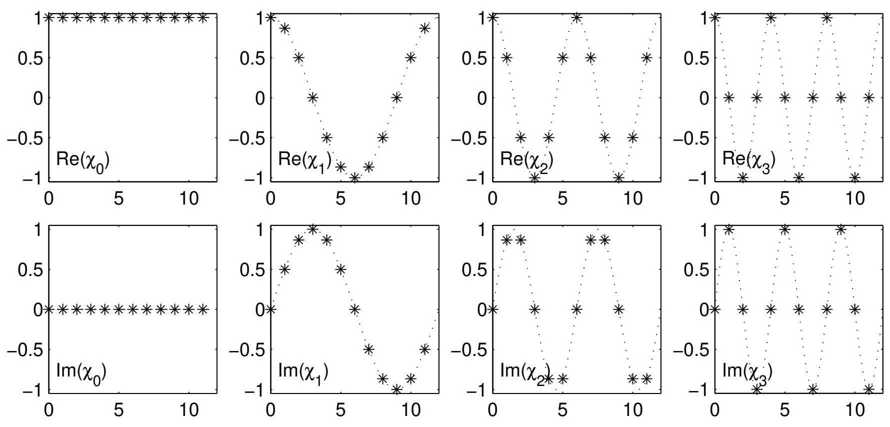

Figure 1.1: The first four characters of the group $\mathbb{Z}/12\mathbb{Z}$

Remark 3. We can already notice that this isomorphism is not canonical, since it depends on the choice of the primitive root of the unit ω chosen. This phenomenon is recurrent in the study of duality (we find it in the linear duality between vector spaces), the dual not being canonically isomorphic to the starting group. All this will be specified later, in particular in Paragraph 1.2.3.

Remark 4. Any cyclic group is isomorphic to $\mathbb{Z}/n\mathbb{Z}$ for $n = |G|$. In a way, the study of the duality on the group $\mathbb{Z}/n\mathbb{Z}$ sums up the one that we can do on any other cyclic group. In the remainder of the presentation, we will consider groups built from the elementary bricks that are groups of the type $\mathbb{Z}/n\mathbb{Z}$. This is why we must keep in mind the structure of duality on these particularly simple groups. By applying the proposition 3, we obtain the isomorphism that must be retained:

$$\forall n \in \mathbb{N}^*, \quad \widehat{\mathbb{Z}/n\mathbb{Z}} \simeq \mathbb{Z}/n\mathbb{Z}.$$

We will see that in fact this property extends to any finite commutative groups.

Before closing this paragraph, notice that we have $|\hat{G}| = |G| = \dim_{\mathbb{C}}(\mathbb{C}[G])$. We even have a stronger property.

Proposition 4. *Let G be a cyclic group. $\hat{G}$ forms an orthonormal basis of $\mathbb{C}[G]$, which means that*

$$\forall (p,\, q) \in \{0, \ldots,\, n-1\}^2, \quad \langle \chi_p,\, \chi_q \rangle = \delta_p^q,$$

where we define the symbol of Kronekerδ_p^q as follows:

$$\delta_p^q = \left\{ \begin{array}{lll} 0 & \text{if} & p \neq q \\ 1 & \text{if} & p = q \end{array} \right. .$$

Proof. We can assume that $G = \mathbb{Z}/n\mathbb{Z}$.
We write $\hat{G} = \{\chi_i\}_{i=0}^{n-1}$, with $\chi_i(k) = \omega^{ik}$, where ω is a primitive $n^{\text{ième}}$ root (according to the proposition 3).
We then have

$$\forall (p,\, q) \in \{0, \ldots,\, n-1\}^2, \quad \langle \chi_p,\, \chi_q \rangle = \frac{1}{n} \sum_{i=0}^{n-1} (\omega^{pq})^i = \delta_p^q \tag{1.5}$$

(we obtain the last equality by summing the geometric series of reason ω^{p-q}).
The family $\hat{G} = \{\chi_i\}_{i=0}^{n-1}$ is therefore orthonormal, and therefore in particular free. To conclude that it does indeed form a base, it suffices to notice that its cardinality is equal to the dimension of $\mathbb{C}[G]$, since we have seen that $|G| = |\hat{G}|$. □

Remark 5. The proof of the orthogonality of the characters in the general case of an abelian group is hardly more complicated, and will be exposed to the proposition 6. However, the proof that we have just made is essential since it is at the base of the results of the one-dimensional Fourier transform, which will be presented in the Section 3.1.

1.2 Dual of an abelian group

Our goal is to extend the result we have just demonstrated ($\hat{G}$ is an orthonormal basis of $\mathbb{C}[G]$) to any finite abelian group. To achieve this, we will use a purely algebraic approach, which uses a character extension theorem. Then, we will establish a stronger result, namely that we have in fact an isomorphism between $\hat{G}$ and G, property that once again we have proved in the case of cyclic groups .

1.2.1 Algebraic approach

The following lemma is the main result in the study of the structure of $\hat{G}$.

Lemma 1 (character extension)**.** *Let G be a commutative finite group and $H \subset G$ a subgroup. Any χ character of H can be extended into a G character.*

Proof. We perform a recurrence on $[G : H] = |G/H|$ the index of H in G. The property being trivial for $[G : H] = 1$, since $G = H$; we therefore assume $[G : H] > 1$, which allows us to take $x \in G$ such that $x \notin H$. Let $K = \langle H,\, x \rangle$ be the group generated by x and H. Let n be the smallest integer such that $x^n \in H$. Any element $z \in K$ is written uniquely as $z = yx^k$ with $y \in H$ and $k \in \{0, \ldots,\, n-1\}$. Indeed, if $yx^k = y'x^{k'}$, with $0 \leq k \leq k' \leq n-1$, then we have $x^{k-k'} \in H$ and $kk' < n$, therefore necessarily $kk' = 0$ by definition of n.
Analysis: Suppose we have an extension $\tilde{\chi}$ of χ.
Let $\zeta = \chi(x)$. We need $\zeta^n = \chi(x^n) = \chi(1) = 1$. So ζ must be a n^{th} root of the unit. We then have, necessarily, if $z \in K$ is written $z = yx^k$ with $y \in H$ and $0 \leq k \leq n-1$,

$$\tilde{\chi}(z) = \tilde{\chi}(yx^k) = \chi(y)\zeta^k. \tag{1.6}$$

Summary: let ζ be a n^{th} root of the unit. Let us define, for $z \in K$ decomposed as previously in the form $z = yx^k$, the extension $\tilde{\chi}$ by the equation (1.6). This is to show that (1.6) does indeed define an element of $\hat{K}$. The uniqueness of the decomposition shows that the definition is not ambiguous. To show that it is indeed a morphism, it suffices to take $h = yx^k$ and $h' = y'x^{k'}$ two elements of K, and to distinguish two cases.

– If $0 \le k + k' \le n - 1$, then we have
$\tilde{\chi}(hh') = \tilde{\chi}(yy'x^{k+k'}) = \chi(yy')\zeta^{k+k'} = \chi(y)x^k\chi(y')x^{k'} = \tilde{\chi}(h)\tilde{\chi}(h')$.

– If $n \le k + k' \le 2n - 1$, we can come back to the previous case,
$\tilde{\chi}(hh') = \tilde{\chi}(yy'x^n x^{k+k'-n}) = \chi(y)\chi(y')\chi(x^n)\zeta^{k+k'-n} = \tilde{\chi}(h)\tilde{\chi}(h')$.

The multiplicativity property of degrees tells us that

$$[G : H] = [G : K][K : H], \quad \text{with} \quad [K : H] > 1.$$

So we have $[G : K] < [G : H]$. We can with the induction hypothesis extend $\tilde{\chi}$ to G. $\square$

As the (arbitrary) choice of the root n^{th} of the unit ζ shows, the character extension is of course not unique. However, it is this result which will allow us to demonstrate that G and $\hat{G}$ have the same cardinality. To do this, let's start by translating the result of character extension in terms of a quotient group.

Lemma 2. *We denote by $\rho : \hat{G} \twoheadrightarrow \hat{H}$ the restriction morphism and $j : \widehat{G/H} \hookrightarrow \hat{G}$ the extension morphism, defined by*

$$j : \left\{ \begin{array}{ccc} \widehat{G/H} & \longrightarrow & \hat{G} \\ \chi & \longmapsto & \tilde{\chi} \end{array} \right. \qquad \textit{with} \quad \tilde{\chi}(x) \overset{\text{def.}}{=} \chi(xH).$$

We have the exact sequence:

$$\{1\} \to \widehat{G/H} \overset{j}{\hookrightarrow} \hat{G} \overset{\rho}{\twoheadrightarrow} \hat{H} \to \{1\}.$$

Proof. By the lemma 1, ρ is surjective.
Moreover, if we consider $\chi \in \mathrm{Ker}(\rho)$, then $H \subset \mathrm{Ker}(\chi)$, and therefore by the universal property of the quotient, there exists a unique $\tilde{\chi} \in \widehat{G/H}$ such that $\chi(x) = \tilde{\chi}(xH)$, i.e. $j(\tilde{\chi}) = \chi$.
Conversely, an element of $\Im(j)$ is trivial over H, which shows that we have $\mathrm{Ker}(\rho) = \Im(j) = \widehat{G/H}$. $\square$

Corollary 1. *Let G be a finite commutative group. Then $\hat{G}$ is of the same order as G.*

Proof. We reason by induction on $n = |G|$. For $n = 1$, the result is trivial because $\hat{G} = \{1\}$, where we denote 1 the trivial character on G (i.e. the function which has any element associatinG1). Let therefore $n \ge 2$, which allows to consider a non-trivial cyclic group $H \subset G$. If $H = G$, we can use the study conducted in Section 1.1.2 on cyclic groups to conclude. Otherwise, we see by the induction hypothesis that $|\hat{H}| = |H|$ and $|\widehat{G/H}| = |G/H|$, and the lemma 2 shows that $|\hat{G}| = |\hat{H}||\widehat{G/H}|$.
We therefore deduce that $|\hat{G}| = |H||G/H| = |G|$. $\square$

1.2.2 Isomorphism theorem

The groups G and $\hat{G}$ therefore have the same cardinality. Although this result is sufficient for the remainder of the presentation, we can nevertheless give a more precise result, in this case explaining an isomorphism between G and $\hat{G}$. To do this, we will use the result obtained for cyclic groups in Section 1.1.2, and we will return to it using the structure theorem of abelian groups. A demonstration of this important result can be found in Artin [3]. We recall the statement of the theorem, without giving a proof.

Théorem 1 (Structure theorem of abelian groups)**.** *Let G be a finite abelian group. There exist strictly positive integers $n_1, \dots, n_r$ uniquely determined such that n_k divides n_{k+1}, and such that we have the isomorphism*

$$G \simeq \mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2\mathbb{Z} \times \cdots \times \mathbb{Z}/n_r\mathbb{Z}.$$

Corollary 2 (Isomorphism theorem)**.** *Let G be a finite commutative group. Then $\hat{G}$ is isomorphic to G. In particular, G and $\hat{G}$ have the same order.*

Proof. It suffices to notice that if G and H are two commutative finite groups, we have $\widehat{G \times H} \simeq \hat{G} \times \hat{H}$. Indeed, if we denote by $i_G : G \to G \times H$ and $i_H : H \to G \times H$ the canonical injections, then the application

$$\Phi : \left\{ \begin{array}{ccc} \widehat{G \times H} & \longrightarrow & \hat{G} \times \hat{H} \\ \chi & \longmapsto & (\chi \circ i_G,\, \chi \circ i_H) \end{array} \right.$$

is an isomorphism. It is trivially injective and, for $(\chi_1,\, \chi_2) \in \hat{G} \times \hat{H}$, the application $\chi : (g,\, h) \mapsto \chi_1(g)\chi_2(h)$ checks $\chi \in \widehat{G \times H}$ and $\Phi(\chi) = (\chi_1,\, \chi_2)$.
We conclude then using the structure theorem 1 as well as the remark 4. □

Remark 6. There is absolutely nothing canonical about the $G \simeq \hat{G}$ isomorphism that we have just updated. Indeed, the latter totally depends on arbitrary choices to describe the structure of the group, as it is given by the theorem 1. Indeed, if we keep the notations of this theorem, each choice of an element of order n_1 sent on $(1, 0, \ldots, 0) \in \mathbb{Z}/n_1\mathbb{Z} \times \cdots \times \mathbb{Z}/n_r\mathbb{Z}$ allows to build a new isomorphism. We must compare this phenomenon with the isomorphism of vector spaces $E \simeq E^*$ which is achieved via the (arbitrary) choice of a basis. Finally, we can add that even the isomorphism $\mathbb{Z}/\hat{n}\mathbb{Z} \simeq \mathbb{Z}/n\mathbb{Z}$ is not canonical, since it depends on the choice of a primitive root of the unit, as explained in the proposition 3.

1.2.3 The bidual

We saw in Paragraph 1.2.2 that the isomorphism $G \simeq \hat{G}$ was not canonical. However, still by analogy with duality in linear algebra, we can be interested in the study of *bidual*. We will see that, in this case, we have a canonical isomorphism with the starting group.

Definition 3 (Bidual). *We have constructed the dual $\hat{G}$ of a finite commutative group G, which in turn is a finite commutative group. We can associate it with its dual which we will denote by $\hat{\hat{G}}$, the bidual of G.*

Proposition 5 (Canonical isomorphism). *We have a canonical isomorphism $G \simeq \hat{\hat{G}}$, which is given by application*

$$\Phi : \left\{ \begin{array}{ccc} G & \longrightarrow & \hat{\hat{G}} \\ g & \longmapsto & (\Phi(g) : \chi \mapsto \chi(g)) \end{array} \right. . \tag{1.7}$$

Proof. First of all, we see that Φ is indeed a morphism of groups. As G and $\hat{\hat{G}}$ have the same cardinality (indeed, a group and its dual have the same cardinal, and we apply this result on the one hand to the group G, on the other hand to the group $\hat{G}$), it suffices to show that Φ is injective. SayinG$g \in \mathrm{Ker}(\Phi)$ means that $\forall \chi \in \hat{G},\ \chi(g) = 1$. To conclude that $g = 1$ it suffices to show, for $h \in G$ other than 1, a character $\chi \in \hat{G}$ such as $\chi(h) \neq 1$. To construct this character, we can consider the group $H \subset G$ generated by $h \neq 1$. As it is cyclic, with a cardinality greater than 1, we know how to construct a character χ_0 such that $\chi_0(h) \neq 1$ (in Section 1.1.2 we have enumerated all the characters of a cyclic group). The lemma 1 shows that we can extend χ_0 into a character $\chi \in \hat{G}$ which still checks $\chi(h) \neq 1$ since $\chi(h) = \chi_0(h) \neq 1$. □

Remark 7. We find a phenomenon similar to the one we encounter on vector spaces of *finite* dimension with the canonical isomorphism $E \simeq E^{**}$ which is defined by the same way as in (1.7). Of course, this remark no longer holds if the vector space is of dimension *infinite*, or if the group is *infinite*. We are first of all obliged to introduce continuity constraints on the applications we are considering, and even under these conditions, it rarely happens that the dual is isomorphic to the starting structure. A good example is given in Paragraph 4.1.1. We will indeed see that the dual of the *torus*$\mathbb{R}/2\pi\mathbb{Z}$ is isomorphic to $\mathbb{Z}$. The exercise 2 proposes to treat the case of an infinite non-commutative group, $SO(3)$.

1.2.4 Orthogonality relations

One can extend without great difficulty the orthogonality of the characters obtained in the cyclic case (proposition 4) to the case of any abelian group. Let's start by proving a lemma which will be very useful later.

Lemma 3. *Let G be a finite abelian group. For $\chi \in \hat{G}$, we have*

$$\sum_{g \in G} \chi(g) = \begin{cases} 0 & \text{if} \quad \chi \neq 1 \\ |G| & \text{if} \quad \chi = 1 \end{cases}. \tag{1.8}$$

Proof. If $\chi = 1$, then the property to be demonstrated is of course verified. So suppose that $\chi \neq 1$. Let $t \in G$ such that $\chi(t) \neq 1$. We then have

$$\chi(t) \sum_{g \in G} \chi(g) = \sum_{g \in G} \chi(tg) = \sum_{h \in G} \chi(h),$$

where we made the change of variable $h = tg$ in the last sum (which is valid because $g \mapsto tg$ is a bijection of G). We therefore deduce that

$$(\chi(t) - 1) \sum_{g \in G} \chi(g) = 0 \quad \Longrightarrow \quad \sum_{g \in G} \chi(g) = 0.$$

Which ends the demonstration. □

Proposition 6 (Orthogonality of characters)**.** *Let G be a finite commutative group. Then $\hat{G}$ is an orthonormal family of elements, i.e .:*

$$\forall (\chi_1, \chi_2) \in \hat{G}^2, \quad \langle \chi_1, \chi_2 \rangle = \begin{cases} 0 & \text{if} \quad \chi_1 \neq \chi_2 \\ 1 & \text{if} \quad \chi_1 = \chi_2 \end{cases}.$$

Proof. We note $\chi \overset{\text{def.}}{=} \chi_1 \overline{\chi_2} = \chi_1 \chi_2^{-1}$ (the $\chi_2(s)$ are of module 1, so it comes $\overline{\chi_2(s)} = \chi_2(s)^{-1}$). We have

$$\langle \chi_1, \chi_2 \rangle = \frac{1}{|G|} \sum_{g \in G} \chi(g).$$

It only remains to notice that if $\chi_1 = \chi_2$, then $\chi = 1$, and that if not, $\chi \neq 1$. We finish by applying the lemma 3. □

Corollary 3. *Let G be a commutative finite group. Then $\hat{G}$ is an orthonormal basis of $\mathbb{C}[G]$.*

Proof. The fact that $\hat{G}$ is an orthogonal family implies in particular that it is a free family of $\mathbb{C}[G]$. Since G and $\hat{G}$ have same cardinality, which is also the dimension of $\mathbb{C}[G]$ as $\mathbb{C}$ -vector space, it is a basis. □

We therefore carried out the program that we had set for ourselves, by explaining a base of the space of functions $\mathbb{C}[G]$ at the same time simple (as we will see it in the Section 3.2, the properties of the roots of the unit will allow rapid calculations of the projections on our basis), and with interesting algebraic properties (which will be used among others in Paragraph 1.4.3).

Once these orthogonal relations between characters have been demonstrated, we can prove other relations, which are in a way "dual".

Proposition 7. *Let g and h be two elements of G. We then have*

$$\sum_{\chi \in \hat{G}} \chi(g) \overline{\chi(h)} = \begin{cases} 0 & \text{if} \quad g \neq h \\ |G| & \text{if} \quad g = h \end{cases}.$$

Proof. It is just a matter of applying the proposition 6 to the abelian group $\hat{G}$. For g and $h \in \hat{\hat{G}}$, we then obtain

$$\sum_{\chi \in \hat{G}} g(\chi)\overline{h(\chi)} = \left\{ \begin{array}{lll} 0 & \text{if} & g \neq h \\ |\hat{G}| = |G| & \text{if} & g = h \end{array} \right. . \tag{1.9}$$

We saw in the previous paragraph that we can in fact canonically identify an element $g \in \hat{\hat{G}}$ to an element $\tilde{g} \in G$ by setting $g(\chi) = \chi(\tilde{g})$. If we rewrite the equation (1.9) using these new notations, we get exactly the formula we want. □

Remark 8. We can represent the characters of a group G in the form of a square matrix $M = \{m_{ij}\}_{1 \leq i,\, j \leq n}$ of size $n \overset{\text{def.}}{=} |G|$. Each line represents the values of a character. More precisely, if we write $G = \{g_1, \ldots, g_n\}$ and $\hat{G} = \{\chi_1, \ldots, \chi_n\}$, then we set $m_{ij} = \chi_i(g_j)$. In this framework, the proposition 6 states orthogonality relations between the rows of the matrix, while the proposition 7 states orthogonality relations between the columns of the matrix.

1.3 Dual of a non-commutative group

After having carried out the study of duality on a finite abelian group, we may want to extend these results to the case of finite non-commutative groups. However, we will see that the beautiful mechanics that we have just developed very quickly fail, even on extremely common groups like symmetrical groups. We will then see that this situation is general, since we are going to show that for any noncommutative group, we are systematically confronted with a lack of characters.

1.3.1 Example of the symmetric group

The fact that $\hat{G}$ is isomorphic to G, and even that $|\hat{G}| = |G|$ fails when G is no longer commutative. We will see it on a concrete example, the symmetric group $\mathfrak{S}_n$. Let us first recall the definition of the *signature* as well as some fundamental properties.

Definition 4 (Signature)**.** *We consider the decomposition of a permutation $\sigma \in \mathfrak{S}_n$ in product of disjoint cycles. It is recalled in fact that such a decomposition exists and is unique down to the order of the factors. To demonstrate this, we can look at Lang [40]. If $\sigma \in \mathfrak{S}_n$ breaks down into the product of k disjoint cycles, we set*

$$\epsilon(\sigma) \overset{\text{def.}}{=} (-1)^{n-k}.$$

This definition is unambiguous, and to verify that it is indeed a morphism, we return to the definition in terms of transpositions using the following lemma.

Lemma 4. *Let $\sigma \in \mathfrak{S}_n$ and τ be a transposition. So we have $\epsilon(\sigma\tau) = -\epsilon(\sigma)$.*

Proof. We denote by τ the transposition $(a,\, b)$. To prove the lemma, we must count the number of cycles in each decomposition and consider two cases. First, if a and b occur in the same cycle c of the decomposition of σ. So, $\sigma\tau$ will have the same decomposition, except for the cycle c which will be split in two. In the second case, we suppose that a and b occur in two disjoint cycles c_1 and c_2 in the writing of σ. In this case, writing $\sigma\tau$ will have one cycle less, since the cycles c_1 and c_2 will be combined. In both cases, the numbers of cycles involved in the writes of σ and $\sigma\tau$ differ by one, which proves the lemma. □

We can then prove the following fundamental property.

Proposition 8. *We suppose that $\sigma \in \mathfrak{S}_n$ is written as the product of p transpositions. We then have $\epsilon(\sigma) = (-1)^p$.*

Proof. We prove this property by an induction over the length of the decomposition in transposition, and by using the previous lemma. □

We must insist on the fact that this property does not allow to directly define the signature ϵ, because the decomposition in transposition is not unique. One is obliged to use the decomposition in disjoint cycles. Once this construction work is done, we are able to determine the dual of $\mathfrak{S}_n$.

Proposition 9. *The only non-trivial character in $\mathfrak{S}_n$ is the signature ϵ.*

Proof. Let χ be a character in $\mathfrak{S}_n$. As the transpositions generate $\mathfrak{S}_n$, it suffices to determine the values that χ can take on the transpositions. Now we see that two transpositions $\tau_1 = (a,\, b)$ and $\tau_2 = (c,\, d)$ of $\mathfrak{S}_n$ are always conjugate. Indeed, we construct a permutation G in $\mathfrak{S}_n$ such that $g(a) = c$, $g(b) = d$.
We have $\tau_2 = g\tau_1 g^{-1}$. This implies that χ, which is constant on the conjugation classes (as we saw in the equation (1.1)), takes one and the same value on all transpositions. Since $\chi(\tau_1^2) = \chi(\tau_1)^2 = 1$, we have $\chi(\tau_1) = +1$ or $\chi(\tau_1) = -1$. So necessarily, a non-trivial character χ must check $\chi(\tau_1) = -1$. Moreover, this condition is sufficient, subject to its existence, to determine χ.
Conversely, we have established the existence of a non-trivial character: the signature. It is therefore the only one. □

We therefore see that we have $\hat{\mathfrak{S}_n} \simeq \mathbb{Z}/2\mathbb{Z}$. This study made in the case of the symmetrical group can be generalized; this is what we will see in the next paragraph.

1.3.2 Use of the derived group

We can in fact describe precisely the dual of a group in terms of a derived group, then apply this description to find the dual of the symmetric group $\mathfrak{S}_n$. Let us start by recalling the definition as well as the main properties of the derived group.

Definition 5 (Derived group)**.** *Let G be a group, we denote by $[x,\, y] \overset{\text{def.}}{=} x\,y\,x^{-1}\,y^{-1}$ the switch associated with the pair $(x,\, y) \in G^2$. We denote by $D(G)$ the group generated by the switches of G, which we call derived group of G. i.e. $D(G) \overset{\text{def.}}{=} \langle [x,\, y]\ ;\ (x,\, y) \in G^2 \rangle$.*

Proposition 10 (Properties of the derived group)**.** *We have $D(G) \lhd G$ (i.e. $D(G)$ is distinguished in G), and $G/D(G)$ is a commutative group. Moreover, $D(G) = \{1\}$ if and only if G is commutative.*

Proof. If $\varphi \in Aut(g)$ is an automorphism of G, we have

$$\forall (x,\, y) \in G^2, \quad \varphi([x,\, y]) = [\varphi(x),\, \varphi(y)],$$

so that the switches are kept by the automorphisms. The same is therefore true of the derived group which is generated by these switches. In particular, $D(G)$ is conserved by interior automorphisms, which is the definition of a distinguished subgroup.
If we denote by $\overline{x}$ and $\overline{y}$ the classes of x and y elements of $G/D(G)$, then $[x,\, y] \overset{\text{def.}}{=} x\,y\,x^{-1}\,y^{-1}$ is an element of $D(G)$, so $\overline{x}\,\overline{y}\,\overline{x}^{-1}\,\overline{y}^{-1} = 1$ in $G/D(G)$, which means that $\overline{x}$ and $\overline{y}$ commute.
The last property is clear with the definition of the derived group. □

Proposition 11. *Let G be a finite group. We have $\hat{G} \simeq G/D(G)$.*

Proof. We can introduce the following morphism:

$$\Phi : \begin{cases} \hat{G} & \longrightarrow \quad L/\hat{D}(G) \\ \chi & \longmapsto \quad \overline{\chi} \end{cases},$$

where $\overline{\chi}$ is defined by $\overline{\chi}(\overline{x}) \overset{\text{def.}}{=} \chi(x)$, where we have noted $\overline{x}$ the class of x in $G/D(G)$. This element $\overline{\chi} \in L/\hat{R}(g)$ is well defined. Indeed, as $\mathbb{C}$ is commutative, for any switch $[x,\, y]$ we have $\chi([x,\, y]) = [\chi(x),\, \chi(y)] = 1$. Thus, the definition of $\overline{\chi}(\overline{x})$ does not depend on the chosen representative.
This morphism Φ is trivially injective, since $\forall x \in G,\ \chi(x) = \overline{\chi}(\overline{x})$. In addition, we can explicitly construct an antecedent for an element $\chi_1 \in G/\hat{D}(G)$, it suffices to construct the character χ defined by the equality

$\chi(x) = \chi_1(\overline{x})$.
We have therefore shown that $\hat{G} \simeq G/\hat{D}(G)$. But as $G/D(G)$ is commutative (proposition 10), we can use the isomorphism theorem 2 and conclude that it is isomorphism $G/\hat{D}(G) \simeq G/D(G)$, which ends the proof of this proposition. □

Remark 9. In fact, the property that we used in the proof is that a morphism which is trivial on switches passes to the quotient by $D(G)$, which leads to the following commutative diagram:

$$\begin{tikzcd} G \arrow[r, "\chi"] \arrow[d, "\pi"] & \mathbb{C}^* \arrow[d, equal] \\ D(G) \arrow[r, "\overline{\chi}"] & \mathbb{C}^* \end{tikzcd} .$$

Let us now show that we find the description of the dual of the group $\mathfrak{S}_n$ obtained in the previous Section 1.3.1. Let us first recall that we denote by $\mathfrak{A}_n$ the subgroup of even permutations, that is to say

$$\mathfrak{A}_n \overset{\text{def.}}{=} \{\sigma \in \mathfrak{S}_n \ : \ \epsilon(\sigma) = 1\},$$

where ϵ denotes the signature. $\mathfrak{A}_n$ is a distinguished subgroup of $\mathfrak{S}_n$, since $\mathfrak{A}_n = \mathrm{Ker}(\epsilon)$, and ϵ is a morphism (with values in $\{-1, 1\}$). But first of all, here is a lemma which specifies the structure of the group $\mathfrak{A}_n$.

Lemma 5. *For $n \geq 3$, $\mathfrak{A}_n$ is generated by cycles of length 3.*

Proof. The first thing to notice is that $\mathfrak{S}_n$ is generated by the transpositions $(1, i)$ for $i = 2 \ldots n$. This is evident by noting that for $i \neq j$, we have $(i, j) = (1, i)(1, j)(1, i)$. Now we just have to realize that an element of $\mathfrak{A}_n$ can only be generated by an even number of transpositions. We therefore see that $\mathfrak{A}_n$ is generated by the elements of the form $(1, i)(1, j) = (1, i, j)$ which are 3-cycles. □

Proposition 12 (Case of the symmetric group)**.** *For $n \geq 3$, we have $D(\mathfrak{S}_n) = \mathfrak{A}_n$. So we have $\hat{\mathfrak{S}_n} \simeq \mathfrak{S}_n/\mathfrak{A}_n \simeq \mathbb{Z}/2\mathbb{Z}$.*

Proof. Since ϵ is a character, we have $D(\mathfrak{S}_n) \subset \mathfrak{A}_n$. As for $n \geq 3$, $\mathfrak{A}_n$ is generated by 3-cycles, it suffices to show that every 3-cycle is a commutator to show the inverse inclusion. For any 3-cycle $\sigma = (a, b, c)$ we have $\sigma^2 = (a, c, b)$ which is still a three cycle. As two cycles of the same length are conjugated in $\mathfrak{S}_n$ (classic result which is the subject of lemma 12), we can find an element $\tau \in \mathfrak{S}_n$ such that $\sigma^2 = \tau\sigma\tau^{-1}$. So we have $\sigma = [\tau, \sigma]$ and we are done.
The fact that $\mathfrak{S}_n/\mathfrak{A}_n \simeq \mathbb{Z}/2\mathbb{Z}$ results from the passage to the quotient of $\epsilon : \mathfrak{S}_n \to \{-1, 1\}$ by $\mathfrak{A}_n$ which is the kernel of this morphism. □

Remark 10. The solution to get around this problem of "missing" characters is to introduce the notion of linear representation, which generalizes the notion of character ($\hat{G}$ consists of the characters of representations of dimension 1). In a way, a non-commutative group does not have enough representations in dimension 1, and it is necessary to pass to the higher dimensions. All this will be the subject of the chapter 7.

1.4 Fourier transform

The main idea of the following paragraphs is to formalize in an algebraic way the Fourier transform using the group structure of the starting set. We will find many similarities with the Fourier transform on $\mathbb{R}$ (the integrals being replaced by finite sums), and the useful properties of the Fourier transform (for example those related to the convolution product) will be explained in terms of group morphisms (in this case finite).

1.4.1 Fourier coefficients and Fourier transform

This paragraph presents the construction of the Fourier coefficients then of the Fourier transform, within the framework of a finite abelian group. It is simply a question of exploiting the property of orthogonality of the characters which we have just demonstrated.

Definition 6 (Fourier coefficients)**.** *For $f \in \mathbb{C}[G]$ we define, for $\chi \in \hat{G}$, the Fourier coefficient$c_f(\chi)$ by*

$$\forall \chi \in \hat{G}, \quad c_f(\chi) \overset{\text{def.}}{=} \langle f, \chi \rangle.$$

This therefore allows to define the application c:

$$c : \left\{ \begin{array}{ccc} \mathbb{C}[G] & \longrightarrow & \mathbb{C}[\hat{G}] \\ f & \longmapsto & c_f \end{array} \right. .$$

In practice, we often use a notation other than that of the Fourier coefficients, by introducing the *Fourier transform.*

Definition 7 (Fourier transform)**.** *The application Fourier transform, noted $\mathcal{F}$, is defined by*

$$\mathcal{F} : \left\{ \begin{array}{ccc} \mathbb{C}[G] & \longrightarrow & \mathbb{C}[\hat{G}] \\ f & \longmapsto & \hat{f} \end{array} \right. , \tag{1.10}$$

where $\hat{f}$ is defined by

$$\forall \chi \in \hat{G}, \quad \hat{f}(\chi) \overset{\text{def.}}{=} |G| c_f(\overline{\chi}) = \sum_{x \in G} f(x)\chi(x).$$

This definition is in fact very natural, as the proposition 2 will show. Figure 1.2 shows the values of the Fourier transform of a "bell" function f defined on $\mathbb{Z}/17\mathbb{Z}$. On the x-axis, we have noted the indices $i \in \{-8, \dots, 0, \dots, 8\}$ of the characters χ_i (the indices are taken in $[-8, 8]$ rather than $[0.16]$ to make the designs look nicer). On the ordinate, we find the values of the Fourier transform $\hat{f}(\chi_i)$. We can verify that the central value (for $i = 0$) is indeed the sum of the values of the function f.

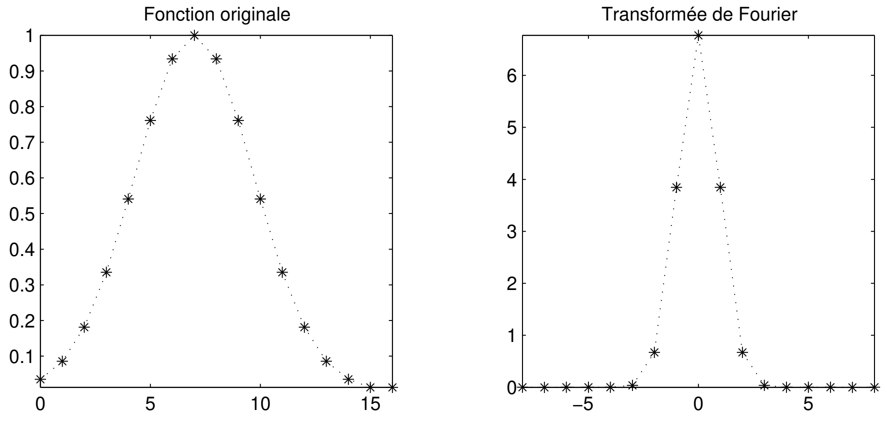

Figure 1.2: Example of Fourier transform

Remark 11. The morphisms c and $\mathcal{F}$ are of course linear, so they are morphisms of vector spaces from $\mathbb{C}[G]$ into $\mathbb{C}[\hat{G}]$. They are in fact *isomorphisms* of vector spaces, and to demonstrate this we will use the following Fourier inversion formula.

Proposition 13 (Inversion formula)**.** *For $f \in \mathbb{C}[G]$, we have the inversion formula*

$$f = \sum_{\chi \in \hat{G}} c_f(\chi)\chi = \frac{1}{|G|} \sum_{\chi \in \hat{G}} \hat{f}(\chi)\chi^{-1}. \tag{1.11}$$

Proof. The equation (1.11) results immediately from the fact that $\hat{G}$ is an orthonormal basis of $\mathbb{C}[G]$, by decomposing f in this based. □

Proposition 14 (Fourier isomorphism)**.** *c and $\mathcal{F}$ are isomorphisms of vector spaces from $\mathbb{C}[G]$ into $\mathbb{C}[\hat{G}]$.*

Proof. Let us show that c is injective. If $c_f = 0$, then the inversion formula (1.11) shows that $f = 0$. As G and $\hat{G}$ have the same cardinal (proposition 1) the spaces $\mathbb{C}[G]$ and $\mathbb{C}[\hat{G}]$ have the same dimension $|G| = |\hat{G}|$. We therefore conclude that c is indeed an isomorphism. The reasoning is the same for $\mathcal{F}$. □

Remark 12. In reality, $\mathcal{F}$ is more than an isomorphism of vector spaces, since it also preserves a very particular algebra structure, the one defined by the convolution product. All this is the subject of Paragraph 1.4.3.

For now, let's continue to state the formulas that we get by using the decomposition in the base of characters.

Proposition 15 (Plancherel formula)**.** *For $(f,\, g) \in \mathbb{C}[G]^2$ we have the following formulas:*

$$\sum_{s \in G} f(s)\overline{g(s)} = |G| \sum_{\chi \in \hat{G}} c_f(\chi)\overline{c_g(\chi)} \tag{1.12}$$

$$= \frac{1}{|G|} \sum_{\chi \in \hat{G}} \hat{f}(\chi)\overline{\hat{g}(\chi)}. \tag{1.13}$$

Proof. By decomposing f and g in the form $f(s) = \sum_{\chi \in \hat{G}} c_f(\chi)\chi(s)$ as well as $g(s) = \sum_{\chi \in \hat{G}} c_g(\chi)\chi(s)$, he comes

$$\sum_{s \in G} f(s)\overline{g(s)} = |G|\langle f,\, g\rangle = |G| \sum_{(\chi_1,\, \chi_2) \in \hat{G}^2} c_f(\chi_1)\overline{c_g(\chi_2)}\langle \chi_1,\, \chi_2\rangle.$$

Hence the equation (1.12) using the orthogonality relations between characters. We prove in the same way the equation (1.13). □

Remark 13. (**Link with theory L^2**). This formula is in every way similar to the formula that we obtain for the Fourier transform of two functions of $L^2(\mathbb{R})$. It translates the conservation of the scalar product (up to a constant) by the Fourier transform, since we can rewrite it in the form:

$$\forall (f,\, g) \in \mathbb{C}[G]^2, \quad \langle f,\, g\rangle = \frac{1}{|G|}\langle \hat{f},\, \hat{g}\rangle,$$

the second scalar product being of course that of $\mathbb{C}[\hat{G}]$. This will be explained in detail in Section 4.1, where we discuss the link between the Fourier transform on the vectors of $\mathbb{C}^N$ (called *transform discrete*) and the Fourier transform *continue*.

1.4.2 Algebra of an abelian group

G always designates a finite abelian group. Since the beginning of this talk, we have denoted $\mathbb{C}[G]$ the (vector) space of functions from G in $\mathbb{C}$. We can give it an algebra structure thanks to the product of functions defined as follows:

$$\forall (f_1,\, f_2) \in \mathbb{C}[G]^2,\ \forall g \in G, \quad (f_1 \cdot f_2)(g) \overset{\text{def.}}{=} f_1(g)f_2(g). \tag{1.14}$$

However, it is not this structure that will be useful for the following, but rather the one defined by the convolution product. Indeed, as we will see, the convolution product closely depends on the structure of the group G considered, unlike the term-to-term product defined by the equation (1.4.3). We will thus see in Paragraph 1.4.3 that the Fourier transform behaves in a very pleasant way for the convolution product.

We have already seen (to the proposition 2) that a basis of the vector space $\mathbb{C}[G]$ is given by the functions $\{\delta_g\}_{g\in G}$, where the function δ_G, for $g \in G$, checks $\delta_g(g) = 1$ and $\delta_g(h) = 0$ for $h \in G$ such that $g \neq h$. Similarly, we have seen that a function $f \in \mathbb{C}[G]$ decomposes in the base $\{\delta_g\}_{g\in G}$ in

$$f = \sum_{g\in G} f(g)\delta_g.$$

We then inject the group G into the vector space $\mathbb{C}[G]$ via the application

$$j : g \in G \mapsto \delta_g \in \mathbb{C}[G].$$

This allows to define (by structure transport) a multiplication noted $*$ between the elements $\{\delta_g\}_{g\in G}$:

$$\forall (g, h) \in G^2, \quad \delta_g * \delta_h \overset{\text{def.}}{=} \delta_{gh}.$$

All that remains is to extend this multiplication to $\mathbb{C}[G]$ by bilinearity in order to provide $\mathbb{C}[G]$ with an algebra structure. This product is called the convolution product, and we can easily calculate the formula giving the expression of a product of two functions.

Definition 8 (Convolution product). *For f_1 and f_2 two functions of $\mathbb{C}[G]$, the convolution product $f_1 * f_2$ is given by*

$$\forall g \in G, \quad (f_1 * f_2)(g) \overset{\text{def.}}{=} \sum_{\substack{(h,\, k)\, \in\, G^2 \\ hk\, =\, g}} f_1(h)f_2(k) = \sum_{h\in G} f_1(h)f_2(h^{-1}g). \tag{1.15}$$

Remark 14. For $f \in \mathbb{C}[G]$ we have

$$\forall (g, h) \in G^2, \quad (f * \delta_g)(h) = f(hg^{-1}).$$

Thus the convolution by an element of G (that is to say the convolution by a function δ_g, using the identification) corresponds to a translation of the function. These properties will be explained again in the simple framework of $G = \mathbb{Z}/n\mathbb{Z}$ in Section 3.3. Let us content ourselves with stating the first properties of the convolution product.

Proposition 16. *The convolution product is commutative, associative, and the map $(f_1, f_2) \mapsto f_1 * f_2$ is bilinear. We thus endow the vector space $\mathbb{C}[G]$ with an algebra structure.*

Proof. The commutativity is easily verified by making the change of variable $h' = h^{-1}g$ in the sum of the equation (1.15). Associativity can be demonstrated by hand, or by using the 2 theorem. The rest is easy. ☐

Before continuing, let's see "graphically" what a convolution product looks like. Figure 1.3 shows the convolution product with itself of a "gate" function, on $\mathbb{Z}/16\mathbb{Z}$. On the x-axis, we noted $\{0, \ldots, 15\}$ of the representatives of $\mathbb{Z}/16\mathbb{Z}$. This is the first time that we approach this kind of figures. These can be a bit confusing and the results aren't necessarily obvious. There are several possibilities.

– We can do the calculation by hand, and check that we get a "triangle" function.
– We can expect the Section 3.3, which studies in detail the discrete cyclic convolution. We will then be able to do the calculations with Matlab, using the FFT algorithm.
– We can read the Section 4.5, which explains the link between the convolutional calculation and the multiplication of polynomials modulo $X^n - 1$.

The fact that the convolution product is defined by extension of the product of the elements of G allows us to state the following proposition.

Proposition 17 (Algebra morphism). *Let $\rho : G \to \mathbb{C}^*$ be a group morphism. There is a unique way to extend it to an algebra morphism $\tilde{\rho} : \mathbb{C}[G] \to \mathbb{C}$.*

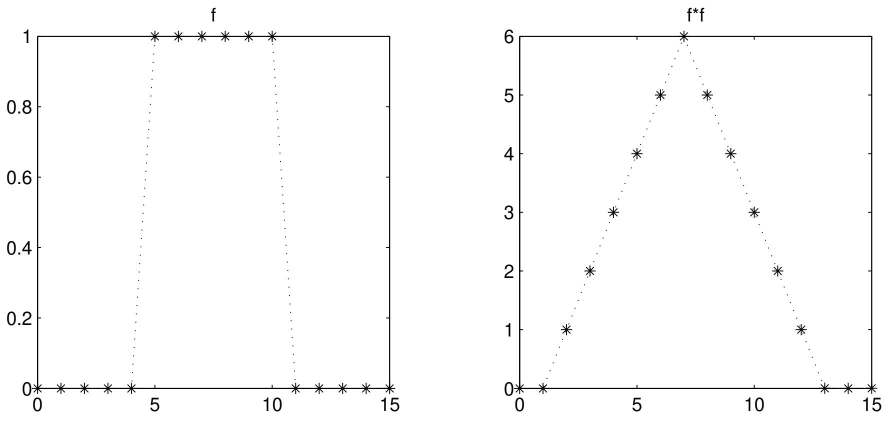

Figure 1.3: Example of convolution calculation

Proof. Indeed, the construction of $\mathbb{C}[G]$ tells us that $\tilde{\rho}$ is only determined by the data of the values of $\tilde{\rho}(\delta_g)$, for $g \in G$. Now in the identification of G as the canonical basis of $\mathbb{C}[G]$, we have $\tilde{\rho}(\delta_g) = \rho(g)$, which shows the uniqueness of the construction. It suffices then to show that the constructed morphism is indeed an algebra morphism. By definition of the convolution product, we can be satisfied with showing the conservation of the product over the elements $\{\delta_g\}_{g\in G}$, which is equivalent to the fact that ρ is a morphism of group. □

This proposition tells us that there is a perfect correspondence between morphisms of groups of G in $\mathbb{C}^*$ and morphisms of algebras of $\mathbb{C}[G]$ in $\mathbb{C}$.

Remark 15. (**Probabilistic interpretation**). The convolution product, which was introduced as the linear extension of a group operation, has a very important probabilistic interpretation. Let X and Y be two *independent* random variables with values in a commutative finite group G. We denote by P_X and P_Y the corresponding probability distributions, that is to say $\forall g \in G,\ P_X(g) = \mathbb{P}(\{X = g\})$. The fundamental result is that the probability distribution of the random variable $X + Y$ is the convolution product of the distributions of X and Y. This is therefore written $P_{X+Y} = P_X * P_Y$. This theorem extends to continuous (with values in $\mathbb{R}$) and discrete (with values in $\mathbb{Z}$) variables, provided that the appropriate convolution product is used. This result is very easy to show (the reader can verify it immediately), and we can consult [49] on the applications of the convolution product in probability (in the continuous and discrete framework). The exercises 9 and 10 study the use of the Fourier transform on a finite group to solve probability problems.

1.4.3 Convolution and Fourier transform

Let G be a commutative finite group of order n. The following proposition shows that the definition of the Fourier transform is in fact very natural.

Proposition 18 (Algebra morphism)**.** *Let $\chi \in \hat{G}$. The application*

$$\mathcal{F}_\chi : \left\{ \begin{array}{ccc} \mathbb{C}[G] & \longrightarrow & \mathbb{C} \\ f & \longmapsto & \hat{f}(\chi) \end{array} \right.$$

corresponds to the unique way to extend the group morphism χ into an algebra morphism.

Proof. The uniqueness results directly from the proposition 17. It only remains to show on the elements δ_G that $\mathcal{F}_\chi$ corresponds to χ, which is trivial:

$$\forall g \in G, \quad \mathcal{F}_\chi(\delta_g) = \sum_{x\in G} \delta_g(x)\chi(x) = \chi(g).$$

This property, which a posteriori justifies the introduction of the Fourier transform, is of capital importance, and we can summarize it in the form of the following convolution theorem.

Théorem 2 (Convolution and Fourier transform)**.** *For f and g two functions of $\mathbb{C}[G]$ we have*

$$\widehat{f * g} = \hat{f} \cdot \hat{g} \quad \textit{and} \quad c_{f*g} = |G| c_f \cdot c_g, \tag{1.16}$$

where we denote by $\cdot$ the term-to-term product of two functions. The Fourier transform $\mathcal{F}$ is therefore an algebra isomorphism from $(\mathbb{C}[G], *)$ to $(\mathbb{C}[\hat{G}], \cdot)$.

This convolution property is undoubtedly the property of the most used Fourier transform, since it allows to change a rather complex problem (the computation of a convolution of two functions) into a more simple (the calculation of the product term by term). The occurrences of this principle of simplification will be numerous throughout the book, whether they are theoretical studies (calculation of circulating determinant, Poisson's formula, etc.) or much more applied (filtering, product of large integers, decoding corrective codes, etc.).

1.5 Exercises

Exercise 1 (Circulating determinant)**.** *Let G be a cyclic group. We fix $f \in \mathbb{C}[G]$. We want to calculate the determinant of endomorphism*

$$\Phi^f \left\{ \begin{array}{ccc} \mathbb{C}[G] & \longrightarrow & \mathbb{C}[G] \\ u & \longmapsto & f * u \end{array} \right. .$$

We will often meet this type of application, in particular in Paragraph 4.2.1, where it will be question of filtering.

1. *Explain why the elements $\chi \in \hat{G}$ are eigenvectors of Φ^f. What are the associated eigenvalues?*
2. *What is the matrix A of the endomorphism Φ^f in the base $\{\delta_g\}_{g\in G}$ of $\mathbb{C}[G]$? Deduce from the previous question an expression of $\det(A)$.*
3. *By judiciously choosing the group G and the application f, show that we have*

$$\det \begin{pmatrix} a_0 & a_1 & a_2 & \dots & a_{n-1} \\ a_{n-1} & a_0 & a_1 & \dots & a_{n-2} \\ vdots & \vdots & \vdots & & \vdots \\ a_1 & a_2 & a_3 & \dots & a_0 \end{pmatrix} = \prod_{i=0}^{n-1} \left(\sum_{j=0}^{n-1} a_j \omega^{ij} \right),$$

   where $(a_0, \dots, a_{n-1}) \in \mathbb{C}^n$, and $\omega \overset{\text{def.}}{=} e^{\frac{2\imath\pi}{n}}$ (A such determinant is called circulating determinant).
4. *After having read the chapter 3 devoted to the discrete Fourier transform and the FFT algorithm, propose a fast implementation of the circulating determinant calculus.*

Exercise 2 (Dual of $SO(3)$)**.** *We denote by $SO(3)$ the group of real, orthogonal 3×3 matrices with determinant 1. It corresponds to the rotations of $\mathbb{R}^3$. We want to show that $SO(3)$ has no non-trivial character.*

1. *Show that two rotations of the same angle are conjugate.*
2. *Let χ be an element of the dual of $SO(3)$. For $g \in SO(3)$, show that $\chi(g)$ depends only on the angle of g.*
3. *We denote by r_α the angle rotation α around $(1, 0, 0)$, and s_α the angle rotation α around $(0, 1, 0)$. We consider $t_\beta \overset{\text{def.}}{=} r_\alpha s_\alpha^{-1}$. Show that t_β is a rotation of some angle β, and that when α traverses $[0, \pi]$, then β does the same.*

4. Deduce that $\chi = 1$.

Exercise 3 (Enumeration of solutions)**.** *Let G be a finite abelian group, and a function $\varphi : G^n \to G$. For $h \in G$, we denote by $N(h)$ the number of n -uples $(g_1, \ldots, g_n)$ such that $\varphi(g_1, \ldots, g_n) = h$. Show that we have*

$$N(h) = \frac{1}{|G|} \sum_{g_1 \in G} \cdots \sum_{g_n \in G} \sum_{\chi \in \hat{G}} \chi\left(\varphi(g_1, \ldots, g_n)\right) \overline{\chi}(h).$$

Exercise 4 (Indicator functions)**.** *Let G be a finite abelian group and $A \subset G$. We denote by f_A the indicator function of A.*

1. Show that

$$\|f_A\|_2 = \sqrt{\frac{|A|}{|G|}} \qquad et \qquad \hat{f}_A(\chi_0) = |A|,$$

where we noted χ_0 the trivial character.

2. We assume that $|A| \leq \frac{1}{2}|G|$. We define

$$\Phi(A) \overset{\text{def.}}{=} \max\left\{ |\hat{f}_A(\chi)| \ : \ \chi \in \hat{G},\ \chi \neq \chi_0 \right\}. \tag{1.17}$$

Show that we have

$$\sqrt{\frac{|A|}{2}} \leq \Phi(A) \leq |A|. \tag{1.18}$$

3. We take the case where $|A| > \frac{1}{2}|G|$. Show that we have $\Phi(A) = \Phi(G\backslash A)$, where we have denoted $G\backslash A$ the complement of A in G. Deduce a lowering of $\Phi(A)$ similar to (1.18).

4. Show that if α is an automorphism of G, then $\Phi(\alpha(A)) = \Phi(A)$.

Intuitively, the closer $\Phi(A)$ is to the lower bound, the more the elements of A are evenly distributed in G. The exercise 9 studies and quantifies this phenomenon. We can see an analogy with the study of the Fourier transform of a continuous function: the lower the high frequency Fourier coefficients, the more "smooth" the function. These indicator functions will be used in Section 6.3 in the context where A is used as corrective code. Once again, it is the spectral properties of f_A that will be used to study the "geometry" of the set A.

Exercise 5 (Equations on a finite abelian group)**.** *This exercise uses the notations and results of exercise 4. It is taken from the review article by Babai*[4].

1. We consider $A_1, \ldots, A_k \subset G$, and we study the equation

$$x_1 + \cdots + x_k = a \qquad \textit{with} \quad x_i \in A_i, \quad i = 1, \ldots, k. \tag{1.19}$$

Explain how we can be reduced to the case $a = 0$. We denote by n the number of solutions of (1.19), in the case $a = 0$. Using the result of the exercise 3, show that

$$N = \frac{1}{|G|} \sum_{\chi \in \hat{G}} \sum_{x_i \in A_i} \chi(x_1 + \cdots + x_k) = \frac{|A_1| \cdots |A_k|}{|G|} + R,$$

with

$$R \overset{\text{def.}}{=} \frac{1}{|G|} \sum_{\chi \neq \chi_0} \prod_{i=1}^{k} \hat{f_{A_i}}(\chi).$$

2. *We assume that $k = 3$. To show that*
$$|R| \leq \frac{\Phi(A_3)}{|G|} \sum_{\chi\in\hat{G}} |\hat{f_{A_1}}(\chi)||\hat{f_{A_2}}(\chi)| \leq \Phi(A_3)\sqrt{|A_1||A_2|},$$
where Φ is defined by the equation (1.17) (we can use the Cauchy-Schwartz inequality). Show, using the result of the exercise 4, question 4., that this is still valid when $a \neq 0$.

3. *Deduce that if*
$$\frac{\Phi(A_3)}{|A_3|} < \frac{\sqrt{|A_1||A_2|}}{|G|},$$
then the equation $x_1 + x_2 + x_3 = a$, with $x_i \in A_i$, $i = 1, 2, 3$, has at least one solution.

This surprising result therefore tells us that if at least one of the three sets A_i is well distributed, and if the three sets are sufficiently large, then the considered equation has at least one solution. The exercise 13 applies this result to $G = \mathbb{F}_q$ to study Fermat's theorem on finite fields.

Exercise 6 (Heisenberg group)**.** *Let G be a finite abelian group. We denote by $\mathcal{U}$ the group of complex numbers of modulus 1. We denote by $\mathcal{H}(g) \overset{\text{def.}}{=} \mathcal{U} \times G \times \hat{G}$ provided with the operation*
$$(\lambda, x, \chi) \cdot (\mu, y, \tau) = (\lambda\mu\tau(x), xy, \chi\tau)$$
the Heisenberg group associated with G.

1. *Show that we define a group structure in this way. In particular, what is the neutral element and what is the inverse of a generic element $(\lambda, x, \chi) \in \mathcal{H}(g)$?*

2. *Show that we can define an action of $\mathcal{H}(g)$ on $\mathbb{C}[G]$ by setting*
$$\forall f \in \mathbb{C}[G],\ \forall(\lambda, x, \chi) \in \mathcal{H}(g), \quad (\lambda, x, \chi) \cdot f : z \mapsto \lambda\chi(z)f(xz).$$

3. *For $(\lambda, x, \chi) \in \mathcal{H}(g)$ and $f \in \mathbb{C}[G]$, we define respectively the expansion operators, translation, and modulation by*
$$D_\lambda(f)(z) = \lambda f(z), \qquad T_x(f)(z) = f(xz), \qquad M_\chi(f)(z) = \chi(z)f(z).$$
Express the action of $\mathcal{H}(g)$ on $\mathbb{C}[G]$ in terms of these three operators. How do these three operators behave with respect to the Fourier transform defined in the equation (1.10)? What is the link with the continuous Fourier transform on $\mathbb{R}$?

4. *By canonically identifying G to $\hat{\hat{G}}$ as described in Paragraph 1.2.3, how the product on $\mathcal{H}(\hat{G}) = \mathcal{U}\times\hat{G}\times G$ defined? How to define an action of $\mathcal{H}(\hat{G})$ on $\mathbb{C}[\hat{G}]$?*

5. *We define the function*
$$\alpha : \left\{ \begin{array}{ccc} \mathcal{H}(g) & \longrightarrow & \mathcal{H}(\hat{G}) \\ (\lambda, x, \chi) & \longmapsto & (\lambda\chi^{-1}(x), \chi, x^{-1}) \end{array} \right. .$$
Show that α is an isomorphism of groups, and that we have
$$\forall f \in \mathbb{C}[G],\ \forall(\lambda, x, \chi) \in \mathcal{H}(g), \quad \mathcal{F}\left((\lambda, x, \chi) \cdot f\right) = \alpha(\lambda, x, \chi) \cdot \mathcal{F}(f),$$
where $\mathcal{F}$ denotes the Fourier transform defined in the equation (1.10).

6. *We suppose that $\Phi : \mathbb{C}[G] \to \mathbb{C}[G]$ commutes with the action of $\mathcal{H}(g)$, that is say that*
$$\forall f \in \mathbb{C}[G],\ \forall(\lambda, x, \chi) \in \mathcal{H}(g), \quad \Phi((\lambda, x, \chi) \cdot f) = (\lambda, x, \chi) \cdot \Phi(f).$$
Show that there exists $r \in \mathbb{C}^*$ such that $\forall f \in \mathbb{C}[G]$, $\Phi(f) = rf$. We can reason on the matrix of Φ expressed in the base $\{\delta_g\}_{g\in G}$ of $\mathbb{C}[G]$, or then use Schur's lemma 13. What happens if $\Phi : \mathbb{C}[G] \to \mathbb{C}[\hat{G}]$ switches the actions of $\mathcal{H}(g)$ and $\mathcal{H}(\hat{G})$?

Exercise 7 (Fourier transform and orthogonalization)**.** *We consider a function $f \in L^2(\mathbb{R})$. We denote by τ_r the translation of r on $L^2(\mathbb{R})$, i.e. $\tau_r(f) = f(\cdot - r)$.*

1. *Show that the family $\{\tau_n(f)\}_{n\in\mathbb{Z}}$ is orthonormal if and only if*

$$\text{for a.e. } \omega \in \mathbb{R}, \quad \sum_{k\in\mathbb{Z}} |\hat{f}(\omega + 2k\pi)|^2 = 1,$$

where we have denoted $\hat{f} \in L^2(\mathbb{R})$ the Fourier transform of f, defined, for the functions $f \in L^1(\mathbb{R})$ by

$$\text{for a.e. } \omega \in \mathbb{R}, \quad \hat{f}(\omega) \overset{\text{def.}}{=} \int_{\mathbb{R}} f(x) e^{-\imath\omega x} \mathrm{d}x,$$

and extended by density to $L^2(\mathbb{R})$ whole.

2. *We assume that there exists $A > 0$ such that*

$$\text{for a.e. } \omega \in \mathbb{R}, \quad A \leq \sum_{k\in\mathbb{Z}} |\hat{f}(\omega + 2k\pi)|^2.$$

Show that if we denote by φ the function of $L^2(\mathbb{R})$ such that

$$\text{for a.e. } \omega \in \mathbb{R}, \quad \hat{\varphi}(\omega) \overset{\text{def.}}{=} \frac{\hat{f}(\omega)}{\left(\sum_{k\in\mathbb{Z}} |\hat{f}(\omega + 2k\pi)|^2\right)^{1/2}},$$

then the family $\{\tau_n(\varphi)\}_{n\in\mathbb{Z}}$ is orthonormal (equalities are to be considered for almost all ω).

The following exercise 8 proposes to study the same orthogonalization problem, but within the framework of a finite abelian group.

Exercise 8 (Orthogonalization on an abelian group)**.** *This exercise is inspired by the article by Bernardini and Kovacevic[7]. Let V be a $\mathbb{C}$ -vector space of dimension n, endowed with a Hermitian product $\langle \cdot, \cdot \rangle$. Let G be a finite abelian group of unit transformations of V, and let $b \in V$. We say that b is orthonormal for the action of G on V if the set $G_b \overset{\text{def.}}{=} \{Ab \,:\, A \in G\}$ is orthonormal. This means that*

$$\forall (x, y) \in G_b^2, \quad \langle x, y \rangle = \delta_x^y.$$

1. *We note*

$$\psi_b : \left\{ \begin{array}{ccc} G & \longrightarrow & \mathbb{C} \\ A & \longmapsto & \langle Ab, b \rangle \end{array} \right. .$$

Show that b is orthonormal for the action of G if and only if $\hat{\psi}_b \equiv 1$, i.e. if and only if $\forall \chi \in \hat{G}$, $|G|\langle \mathcal{U}_\chi b, b \rangle = 1$, where we noted

$$\forall \chi \in \hat{G}, \quad \mathcal{U}_\chi \overset{\text{def.}}{=} \frac{1}{|G|} \sum_{A\in G} \chi(A) A \quad \in \mathcal{L}(V, V).$$

2. *Show that the operators $\mathcal{U}_\chi$ are orthogonal projectors, and that they are two by two orthogonal, that is to say*

$$\forall (\chi_1, \chi_2) \in \hat{G}^2, \quad \mathcal{U}_{\chi_1} \mathcal{U}_{\chi_2} = \left\{ \begin{array}{lll} 0 & \text{if} & \chi_1 \neq \chi_2 \\ \mathcal{U}_{\chi_1} & \text{if} & \chi_1 = \chi_2 \end{array} \right. ,$$

and $\mathcal{U}_\chi^* = \mathcal{U}_\chi$ (A^* denotes the assistant of A).

3. *We assume that $\hat{\psi}_b$ does not vanish. We notice*

$$\tilde{b} \overset{\text{def.}}{=} \sum_{\chi \in \hat{G}} \frac{1}{\sqrt{\hat{\psi}_b(\chi)}} \mathcal{U}_\chi b,$$

where $\sqrt{\hat{\psi}_b(\chi)}$ denotes one of the two possible roots. Show that $\tilde{b}$ is orthonormal for the action of G.

4. *What connection can we make with the exercise 7?*

For a more in-depth study of this method in the case of cyclic groups $\mathbb{Z}/n\mathbb{Z}$, we can look at the exercise 7.

Exercise 9 (Distribution of probability)**.** *Let G be a finite abelian group, and $P : G \to \mathbb{R}^+$ the distribution function of a probability law on G, which means that $\sum_{g \in G} P(g) = 1$. We denote by U the uniform distribution, that is to say $U(g) = \frac{1}{|G|}$ for all $g \in G$. We denote by χ_0 the trivial character of G.*

1. *Calculate $\hat{P}(\chi_0)$ as well as $\hat{U}(\chi)$, for $\chi \in \hat{G}$. Deduce an expression of $\|P - U\|_2^2$.*

2. *Show that we have*

$$\forall g \in G, \quad \left| P(g) - \frac{1}{|G|} \right|^2 \le \frac{1}{|G|} \sum_{\chi \ne \chi_0} |\hat{P}(\chi)|^2.$$

In a way, the quantity $\|P - U\|_2^2$ measures the uniformity of the distribution P, and as we have already seen for the characteristic functions (exercise 4), this is characterized by the Fourier coefficients $\hat{P}(\chi)$, for $\chi \ne \chi_0$.

Exercise 10 (Random walk)**.** *We consider a random walk on $\mathbb{Z}/n\mathbb{Z}$ cons trui te as follows. The random variable $X_k \in \mathbb{Z}/n\mathbb{Z}$ designates a position on the circle $\mathbb{Z}/n\mathbb{Z}$ at the instant $k \in \mathbb{N}$. The displacement between the instant k and $k+1$ is given by a transition probability $p_{i,j} \overset{\text{def.}}{=} \mathbb{P}(X_{k+1} = j \,|\, X_k = i)$.*

1. *We denote by $p^{(k)} : G \to [0,\,1]$ the probability distribution of X_k, i.e. for $0 \le i < n$, $p^{(k)}(i) = \mathbb{P}(X_k = i)$. We can also write $p^{(k)}$ in the form of a vector of size n. Show that $p^{(k+1)} = Pp^{(k)}$, where we have denoted P the transition matrix $\{p_{i,j}\}_{0 \le i,j \le n-1}$. Deduce that $p^{(k)} = P^k p^{(0)}$, where $p^{(0)} = \{1, 0, \dots, 0\}$ is the initial distribution.*

2. *Let $0 < p < 1$. We consider the simplest random walk, given by*

$$\begin{cases} p_{i,i-1} = p, & p_{i,i+1} = 1 - p, \\ p_{i,j} = 0 \quad \textit{if} & i \notin \{i-1,\, i+1\} \end{cases}.$$

Show that we then have $Px = v * x$, where $*$ denotes the product of convolution and $v = \{0, p, 0, \dots, 0, 1-p\}$. How can we find this result by considering sums of random variables, and by using the remark 15?

3. *Express the Fourier transform $\hat{p^{(k)}}$ from $\hat{p^{(0)}}$. Deduce that if n is odd, then*

$$p^{(k)} \overset{k \to +\infty}{\longrightarrow} u \qquad \textit{where} \qquad u \overset{\text{def.}}{=} \{1/n, \dots, 1/n\}.$$

What happens if n is even?

4. *Generalize this result to a translation invariant random walk, that is to say such that*

$$p_{i,j} = v_{j-i} \qquad \textit{with} \qquad v \in ([0,\,1])^n.$$

The figure 1.4 shows the progression of the probability distribution $p^{(k)}$ in the case of two random walks. We see that the second step converges more quickly towards the uniform distribution (we can make the link with the size of the Fourier coefficients).

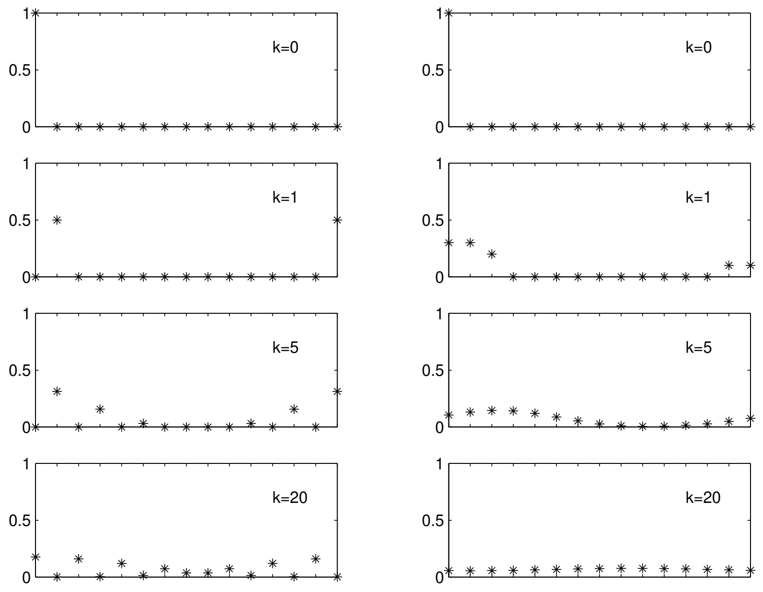

Figure 1.4: Random walks for the transition probabilities $\{0, 1/2, 0, \dots, 0, 1/2\}$ (left) and $\{0.3, 0.3, 0.2, 0, \dots, 0, 0.1, 0.1\}$ (right)

Exercise 11 (Discrete uncertainty principle). *Let G be a finite group, and $f \in \mathbb{C}[G]$ a non-zero function. We want to show that we have*

$$\left|\operatorname{Supp}(f)\right| \times \left|\operatorname{Supp}\left(\hat{f}\right)\right| \geq |G|, \tag{1.20}$$

where $|\operatorname{Supp}(f)|$ denotes the size of the support of f.

1. *We consider, first of all, the group $G = \mathbb{Z}/n\mathbb{Z}$. Show that if f has p non-zero elements, then $\hat{f}$ cannot have p consecutive zeros. Deduce the equation (1.20). This result was demonstrated first by Donoho and Stark*[27].

2. *We return to the general case of a finite abelian group G.*
   We denote by $M \overset{\text{def.}}{=} \sup\{f(x) \,:\, x \in G\}$ and $\|f\|_2^2 \overset{\text{def.}}{=} \langle f, f\rangle$. To show that

$$\|f\|_2^2 \leq \frac{M^2}{|G|}|\operatorname{Supp}(f)| \qquad \textit{et} \qquad M \leq \frac{1}{|G|}\sum_{\chi \in \hat{G}} |\hat{f}(\chi)|.$$

   Using the Cauchy-Schwartz inequality, deduce that

$$M^2 \leq \frac{|\operatorname{Supp}(\hat{f})|}{|G|}\|\hat{f}\|_2^2, \qquad \textit{then,} \qquad M^2 \leq \|f\|_2^2|\operatorname{Supp}(\hat{f})|.$$

   To conclude that

$$\|f\|_2^2 \leq \frac{\|f\|_2^2}{|G|}|\operatorname{Supp}(f)| \times |\operatorname{Supp}(\hat{f})|.$$

3. *Let $H \subset G$, a subgroup and $f_H \in \mathbb{C}[G]$, its characteristic function. Show that we have*

$$\hat{f_H} = |H| f_{H^\sharp},$$

   where we denote by $H^\sharp \subset \hat{G}$ the orthogonal of H. Deduce that the function f reaches the bound of the equation (1.20). In fact, we can show that any function which reaches this bound is related to such a function f_H by a translation and a dilation. This is demonstrated in the article by Matusiak[**?**].

This conclusion on the localization of temporal and frequency supports may seem negative at first glance: it prohibits us from constructing signals well localized both in time and in frequency. However, it can be used to advantage, for example to build efficient corrective codes, as the proposition 58 will show.

Chapter 2

Applications of duality on a finite group

To better understand the theory of characters over a commutative group, it must be applied in situations where it is really useful. The aim of this chapter is therefore to understand, through examples, why this theory is so powerful. We will thus demonstrate without too much effort formulas that may seem complex, at least for someone who is approaching them for the first time. The best example is the quadratic reciprocity formula, the proof of which is based on the use of two types of characters.

2.1 Gaussian sums

The idea behind the discussion of this paragraph is very simple. It is a matter of better understanding *finite fields*, and more precisely, of perceiving more clearly how the two structures which make up such a field (multiplicative group and additive group) can co-exist, and influence each other mutually. The main tool will of course be the *duality* on an abelian group, and the idea to develop will be the combination of two types of characters. It is precisely to combine these characters that we are going to introduce the notion of *Gaussian sum*, which is presented in Paragraph 2.1.3.

The main reference for this talk is the book of Lidl and Niederreiter [44], which constitutes a true encyclopedia of finite fields. We can also read with great interest the very nice presentation of Langevin [41], which links many related subjects, such as Gaussian sums, corrective codes, and number theory.

2.1.1 Quadratic residues

Before embarking on the study of duality over a finite field, let us give a simple example which will justify the introduction of character theory. Consider the following equation, which has integer unknowns:

$$x^2 = 2y^2 + 7k \qquad \text{with} \quad (x,\, y,\, k) \in \mathbb{Z}^3. \tag{2.1}$$

To solve it almost trivially, we just need to replace it with its counterpart modulo the prime number 7, and use the field structure of $\mathbb{Z}/7\mathbb{Z}$ which allows us to perform divisions:

$$(x/y)^2 = 2 \mod 7,$$

for $x \neq 0$. All that remains is to list the squares of $\{0, 1, \ldots, 6\}$, taken modulo 7. We easily get $\{0,\, 1,\, 4,\, 2,\, 2,\, 4,\, 1\}$. We therefore conclude that the non-zero solutions of the equation are given by

$$\frac{x}{y} = 3 + 7k' \quad \text{and} \quad \frac{x}{y} = 4 + 7k', \quad \text{for} k' \in \mathbb{Z}.$$

Obviously, now that we know the square roots of 2 in $\mathbb{Z}/7\mathbb{Z}$, we can consider the following factorization of the equation (2.1):

$$(x-3y)(x-4y) = 0 \mod 7,$$

which of course leads to the same result.

This naive approach is obviously to be avoided in the case of study of large numbers. We are led to consider, for a prime number p the *Legendre symbol*, defined as follows:

$$\forall n \in \mathbb{Z}, \quad \left(\frac{n}{p}\right) = \begin{cases} 0 & \text{si} n \text{is divisible by} p \\ 1 & \text{si} n \text{is a modulo square} p \\ -1 & \text{si} n \text{is not a modulo square} p \end{cases}.$$

It is obvious that we can restrict the study of this symbol to only the elements of $\mathbb{Z}/p\mathbb{Z}$, which we usually denote by $\mathbb{F}_p$. The capital remark for what follows is that the application

$$\eta : \begin{cases} \mathbb{F}_p^* & \longrightarrow & \{-1, 1\} \\ n & \longmapsto & \left(\frac{n}{p}\right) \end{cases} \tag{2.2}$$

is a character of the multiplicative group $\mathbb{F}_p^*$, since we have

$$\left(\frac{n_1 n_2}{p}\right) = \left(\frac{n_1}{p}\right)\left(\frac{n_2}{p}\right).$$

This property follows directly from the following lemma.

Lemma 6 (Euler's formula)**.** *Let p be an odd prime number. An element $x \in \mathbb{F}_p^*$ is a square if and only if $x^{\frac{p-1}{2}} = 1$. An element $x \in \mathbb{F}_p^*$ is not a square if and only if $x^{\frac{p-1}{2}} = -1$. Consequently, we have the formula of Euler*

$$\forall x \in \mathbb{F}_p^*, \quad \left(\frac{x}{p}\right) = x^{\frac{p-1}{2}}.$$

Proof. We consider the multiplicative group $\mathbb{F}_p^*$. As p is odd, the morphism $x \mapsto x^2$ has as its kernel the subgroup $\{1, -1\}$, of so that the elements which are modulo p squares form a subgroup of $\mathbb{F}_p^*$ of cardinal $\frac{p-1}{2}$.
If $x = y^2$, then $x^{\frac{p-1}{2}} = y^{p-1} = 1$. So the $\frac{p-1}{2}$ quadratic residues modulo p are all roots of the polynomial $X^{\frac{p-1}{2}} - 1$, which cannot have more than $\frac{p-1}{2}$ roots. The residuals are therefore exactly these roots, that is to say the elements x such that $x^{\frac{p-1}{2}} = 1$.
To conclude the demonstration, it suffices to note that $\left(x^{\frac{p-1}{2}}\right)^2 = 1$ implies that $x^{\frac{p-1}{2}}$ is an element of $\{-1, 1\}$. The quadratic non-residues are characterized by $x^{\frac{p-1}{2}} = -1$, which completes the proof of Euler's formula. ☐

The aim of this chapter is to demonstrate the important property of these Legendre characters, which we call the quadratic reciprocity formula. It relates the fact of being a square modulo p to that of being a square modulo q. In his time, *Euler* had already noticed, by calculating many cases by hand, that for two first distinct odd numbers p and q, we have $\left(\frac{p}{q}\right) = \left(\frac{q}{p}\right)$, except in the case where p and q are both of the form $4k-1$. This result is however far from obvious, and it was necessary to wait *Gauss* to obtain a complete proof of this result.

Thanks to this formula, we will be able to easily calculate the Legendre character, by successively applying inversions of the symbol $\left(\frac{q}{p}\right)$ (with the reciprocity formula) and reductions of q modulo p (because the symbol depends only on the class of n modulo p).

2.1.2 Additive and multiplicative characters

In the previous chapter, we were interested in commutative finite groups. We are now going to impose a more rigid structure on our group, since we are going to be interested in a finite field $\mathbb{F}_q$, with $q = p^r$ where p is a prime number. It is a feature field p, and it can be seen as a finite dimensional vector space r over its prime field $\mathbb{F}_p$. A more detailed description of finite fields will be given in Section 6.1, when constructing a Fourier transform with values in a finite field.

On our body, we can identify two group structures. First of all we can consider $\mathbb{F}_q$ as an additive group (in fact a vector space on $\mathbb{F}_p$). Then, we can also consider the multiplicative group $\mathbb{F}_q^* \overset{\text{def.}}{=} \mathbb{F}_q - \{0\}$, which is a cyclic group of order $q-1$. This leads to considering two types of characters.

Definition 9 (Additive and multiplicative characters)**.** *The elements of $\hat{\mathbb{F}_q}$ are called additive characters. These are therefore the morphisms*

$$\psi : (\mathbb{F}_q, +) \longrightarrow (\mathbb{C}^*, *).$$

The elements of $\hat{\mathbb{F}_q^*}$ are called multiplicative characters. These are therefore the morphisms

$$\chi : (\mathbb{F}_q^*, *) \longrightarrow (\mathbb{C}^*, *).$$

The easiest characters to determine are the multiplicative characters, that is, the elements of $\hat{\mathbb{F}_q^*}$. Indeed, the group $\mathbb{F}_q^*$ is cyclic. So let ζ be a generator of this group, so that we have $\mathbb{F}_q^* = \{1, \zeta, \zeta^2, \dots, \zeta^{q-2}\}$. We can then enumerate the $q-1$ multiplicative characters

$$\forall j = 0, \dots, q-1, \quad \chi_j : \left\{ \begin{array}{ccc} \mathbb{F}_q^* & \longrightarrow & \mathbb{F}_q^* \\ \zeta^k & \longmapsto & e^{\frac{2\imath\pi}{q-1} jk} \end{array} \right. .$$

We thus obtain a complete description of the dual group $\hat{\mathbb{F}_q^*}$, and we see of course that we have $\hat{\mathbb{F}_q^*} \simeq \mathbb{F}_q^*$. This description is not canonical, in the sense that it requires the (arbitrary) choice of a primitive root ζ.

Regarding the additive group $\mathbb{F}_q$, the situation is a little more complex, since this group is not cyclic. However, as an additive group, $\mathbb{F}_q$ is in fact isomorphic to the product group $(\mathbb{Z}/p\mathbb{Z})^r$. As $\mathbb{Z}/p\mathbb{Z}$ is a cyclic (additive) group, it will be relatively easy to list the additive characters of $\mathbb{F}_q$. However, in order to produce as little effort as possible, and to simplify the description of the dual, we will introduce the following notion.

Definition 10 (Application trace)**.** *Let K be a finite field, containing a sub-body k of cardinality s. We denote by $t \overset{\text{def.}}{=} [K : k]$ the dimension of K as k-vector space, so that $|K| = s^t$. Let $\alpha \in K$. We define the application trace from K to k as follows:*

$$\mathrm{Tr}_{K/k}(\alpha) \overset{\text{def.}}{=} \alpha + \alpha^s + \cdots + \alpha^{s^{t-1}}.$$

In the following, we will be interested in the bodies $k = \mathbb{F}_p$ and $K = \mathbb{F}_q$ (we therefore fix $s = p$ and $t = r$). When there is no risk of confusion, we will simply write Tr *instead of* $\mathrm{Tr}_{\mathbb{F}_q/\mathbb{F}_p}$.

We recall some important properties of finite fields, which we will find demonstrated in the book of Perrin [52].

Proposition 19 (Properties of finite fields)**.** *Let K be a finite field of characteristic p.*

(i) *Let k be a subfield of K of cardinal s An element $x \in K$ belongs to k if and only if $x^s = x$.*

(ii) *The application $\Phi : x \mapsto x^p$ is a morphism, called morphism by Frobenius. The iterates $\Phi^k : x \mapsto x^{p^k}$ are also morphisms.*

Let's see the main properties of this application.

Proposition 20 (Track properties)**.** *The trace of K over k is a non-zero k -linear form with values in k.*

Proof. The first thing to show is that for $\alpha \in K$, we have $\mathrm{Tr}_{K/k}(\alpha) \in k$. It suffices to show that we have $x^s = x$. Using the linearity of the morphism $x \mapsto x^s$ (which is an iterated from Frobenius), it comes

$$\mathrm{Tr}_{K/k}(\alpha)^s = \alpha^s + \alpha^{s^2} + \cdots + \alpha^{s^t}.$$

Since K^* is a group of cardinal $s^t - 1$, we have $\forall \alpha \in K^*,\ \alpha^{s^t-1} = 1$, hence the desired result, since $\forall \alpha \in K,\ \alpha^{s^t} = \alpha$.
Using the Frobenius morphism and the fact that $\lambda^s = \lambda$ for a scalar $\lambda \in k$, it is clear that l The trace application is indeed k -linear. It remains to show that it is not trivial, i.e. there exists an element $\alpha \in K$ such that $\mathrm{Tr}_{K/k}(\alpha) \neq 0$. Now, if $\mathrm{Tr}_{K/k}(\alpha) = 0$, this means that α is root of the polynomial

$$P(X) \overset{\text{def.}}{=} X + X^s + \cdots + X^{s^{t-1}},$$

which is of degree s^{t-1}. This polynomial therefore has at most s^{t-1} roots, and as K a s^t elements, there does exist a certain $\alpha \in K$ such that $P(\alpha) \neq 0$. $\square$

We can now provide a full description of the additive characters of $\mathbb{F}_q$. To do this, we introduce a so-called *canonical* character, in the sense that it is independent of the way in which we construct the body $\mathbb{F}_q$.

Definition 11 (Canonical character)**.** *We define the additive character canonical ψ_1, element of $\hat{\mathbb{F}_q}$ by*

$$\psi_1 : \left\{ \begin{array}{ccc} \mathbb{F}_q & \longrightarrow & \mathbb{C}^* \\ x & \longmapsto & e^{\frac{2\imath\pi}{p}\mathrm{Tr}(x)} \end{array} \right. .$$

The following theorem explains the construction of other characters from this canonical character.

Proposition 21. *Let, for $a \in \mathbb{F}_q$, the application*

$$\psi_a : \left\{ \begin{array}{ccc} \mathbb{F}_q & \longrightarrow & \mathbb{C}^* \\ x & \longmapsto & \psi_1(ax) \end{array} \right. .$$

It is an additive character, $\psi_a \in \hat{\mathbb{F}_q}$, and conversely, any additive character is written that way.

Proof. It is obvious that we have well constructed characters in this way. Let us show that they are all different.
Since the trace is non-identically zero, the canonical character is non-trivial. If we consider two elements $a \neq b$ of $\mathbb{F}_q$, then we can find another element $c \in \mathbb{F}_q$ such that

$$\frac{\psi_a(c)}{\psi_b(c)} = \psi_1\left((ab)c\right) \neq 1.$$

So we have $\psi_a \neq \psi_b$. This means that the number of characters of type ψ_a is equal to q. Now we know, with Corollary 1, that $|\hat{\mathbb{F}_q}| = |\mathbb{F}_q| = q$. We have therefore constructed all the characters well. $\square$

Remark 16. (**Trivial character**). We have thus constructed an isomorphism between $\mathbb{F}_q$ and its dual $\hat{\mathbb{F}_q}$ by the application $a \mapsto \psi_a$. We denote by $\psi_0 = 1$ the trivial additive character. It should not be confused with the trivial multiplicative character χ_0, since the latter is not defined in 0. We will see later that we often extend the multiplicative characters $\chi \in \hat{\mathbb{F}_q^*}$ by setting $\chi(0) = 0$, which removes any ambiguity between the two trivial characters.

Remark 17. (**Character extension**). Let K be a finite super-field of $\mathbb{F}_q$, which can be written in the form $K = \mathbb{F}_{q^t}$, where $t \overset{\text{def.}}{=} [K : \mathbb{F}_q]$. We can easily check that for $\beta \in K$, we have

$$\mathrm{Tr}_{K/\mathbb{F}_p}(\beta) = \mathrm{Tr}_{\mathbb{F}_q/\mathbb{F}_p}\left(\mathrm{Tr}_{K/\mathbb{F}_q}(\beta)\right).$$

If we denote by μ_1 the canonical character of K, this implies that

$$\mu_1(\beta) = \psi_1(\mathrm{Tr}_{K/\mathbb{F}_q}(\beta)). \tag{2.3}$$

Before detailing the properties of additive and multiplicative characters, let us give a basic example.

Example 1 (Quadratic character). Let q be an odd integer. We define a multiplicative character $\eta \in \hat{\mathbb{F}_q^*}$ as follows:

$$\forall x \in \mathbb{F}_q^*, \quad \eta(x) \overset{\text{def.}}{=} \begin{cases} 1 & \text{si} x \text{is a square in} \mathbb{F}_q \\ -1 & \text{otherwise} \end{cases}.$$

We can easily see that we have $\eta = \chi_{\frac{q-1}{2}}$. Moreover, in the case where $q = p$ is a prime number, we have $\eta(x) = \left(\frac{x}{p}\right)$, which means that η is the symbol of *Legendre.*

We will now recall the orthogonality properties of the characters demonstrated in Paragraph 1.2.4, by stating them for the additive and multiplicative characters.

Proposition 22 (Properties of additive characters)**.** *Let a and b be elements of $\mathbb{F}_q$. We then have*

$$\sum_{x\in\mathbb{F}_q} \psi_a(x)\overline{\psi_b(x)} = \begin{cases} 0 & si \quad a \neq b \\ q & si \quad a = b \end{cases} \tag{2.4}$$

$$\sum_{x\in\mathbb{F}_q} \psi_a(x) = 0 \quad si a \neq 0 \tag{2.5}$$

$$\sum_{x\in\mathbb{F}_q} \psi_x(a)\overline{\psi_x(b)} = \begin{cases} 0 & si \quad a \neq b \\ q & si \quad a = b \end{cases}. \tag{2.6}$$

Proposition 23 (Properties of multiplicative characters)**.** *Let a and b be elements of $\mathbb{F}_q^*$ and let χ and τ be two elements of $\hat{\mathbb{F}_q^*}$. We then have*

$$\sum_{x\in\mathbb{F}_q^*} \chi(x)\overline{\tau(x)} = \begin{cases} 0 & si \quad \chi \neq \tau \\ q-1 & si \quad \chi = \tau \end{cases} \tag{2.7}$$

$$\sum_{x\in\mathbb{F}_q^*} \chi(x) = 0 \quad si \chi \neq \chi_0 \tag{2.8}$$

$$\sum_{\chi\in\hat{\mathbb{F}_q^*}} \chi(a)\overline{\chi(b)} = \begin{cases} 0 & si \quad a \neq b \\ q-1 & si \quad a = b \end{cases}. \tag{2.9}$$

Remark 18. As we have already explained in paragraph 1.2.4, we can represent the characters of a finite abelian group in the form of a matrix (each line represents one character). In this framework, the equations (2.4) and (2.7) represent orthogonal relations between the rows of the matrix, and the equations (2.6) and (2.9) represent orthogonality relations between the columns of the matrix.

2.1.3 Gaussian sums

We can now define the important object of this chapter, which makes the connection between the additive and multiplicative characters of a finite field.

Definition 12 (Gaussian sums)**.** *Let $\chi \in \hat{\mathbb{F}_q^*}$ and $\psi \in \hat{\mathbb{F}_q}$ of the multiplicative and additive characters respectively. We define the Gaussian sum $G(\chi, \psi)$ associated with these two characters, by*

$$G(\chi, \psi) \overset{\text{def.}}{=} \sum_{x\in\mathbb{F}_q^*} \psi(x)\chi(x). \tag{2.10}$$

This Gauss sum, which brings into play the two structures of the finite field, is in fact very close to the Fourier transform, as we could define it in equation (1.10). Indeed, let us recall the definition of the Fourier transform on the multiplicative group $\mathbb{F}_q^*$:

$$\forall f \in \mathbb{C}[\mathbb{F}_q^*], \quad \mathcal{F}_{\text{mul}}(f) : \begin{cases} \hat{\mathbb{F}_q^*} & \longrightarrow & \mathbb{C} \\ \chi & \longmapsto & \sum_{x\in\mathbb{F}_q^*} f(x)\chi(x) \end{cases}.$$

We can thus write the Gauss sum of the equation (2.10) as the multiplicative Fourier transform of an additive character. More precisely, we have

$$\forall \psi \in \hat{\mathbb{F}}_q,\ \forall \chi \in \hat{\mathbb{F}}_q^*, \quad G(\chi,\, \psi) = \mathcal{F}_{\text{mul}}(\psi)(\chi).$$

However, in the rest of the presentation, we will be led to consider the opposite point of view, that is to say that we will rather be interested in the Fourier transform on the group additive. This is why we extend a multiplicative character $\chi \in \hat{\mathbb{F}}_q^*$ into a function $\tilde{\chi} \in \mathbb{C}[\mathbb{F}_q]$ by setting $\tilde{\chi}(0) = 0$. Under these conditions, we can also see a Gauss sum as the additive Fourier transform of a multiplicative character. Recall the definition of the additive transform:

$$\forall f \in \mathbb{C}[\mathbb{F}_q], \quad \mathcal{F}_{\text{add}}(f) : \left\{ \begin{array}{ccc} \hat{\mathbb{F}}_q & \longrightarrow & \mathbb{C} \\ \psi & \longmapsto & \sum_{x \in \mathbb{F}_q} f(x)\psi(x) \end{array} \right. . \tag{2.11}$$

We then obtain the remarkable formula

$$\forall \psi \in \hat{\mathbb{F}}_q,\ \forall \chi \in \hat{\mathbb{F}}_q^*, \quad G(\chi,\, \psi) = \mathcal{F}_{\text{add}}\left(\tilde{\chi}\right)(\psi)\,. \tag{2.12}$$

We will therefore take care that the function $\tilde{\chi}$ corresponds to the character χ extended into 0.

As an application of these observations, we can decompose a multiplicative character into an additive Fourier series.

Proposition 24. *Let $\chi \in \hat{\mathbb{F}}_q^*$. We have*

$$\chi = \frac{1}{q} \sum_{\psi \in \hat{\mathbb{F}}_q} G(\chi,\, \overline{\psi})\psi.$$

Proof. By applying to the function $\tilde{\chi}$ the Fourier series decomposition formula, proposition 13, we obtain

$$\tilde{\chi} = \sum_{\psi \in \hat{\mathbb{F}}_q} \langle \tilde{\chi},\, \psi \rangle \psi.$$

It only remains to notice that $\langle \tilde{\chi},\, \psi \rangle \overset{\text{def.}}{=} \frac{1}{q}\mathcal{F}_{\text{add}}\left(\tilde{\chi}\right)\left(\overline{\psi}\right) = \frac{1}{q}G(\chi,\, \overline{\psi})$ to conclude. □

In practice, we are often unable to simply calculate the values of these Gauss sums. The only (trivial) mark-up available is $|G(\chi,\, \psi)| \leq q - 1$. However, the proposition 26 will give us the value of its modulus. Let us start by stating a series of more or less obvious properties of Gauss sums.

Proposition 25 (Properties of Gaussian sums)**.** *We recall that p is the characteristic of the field $\mathbb{F}_q$, that is to say that $q = p^r$. So, if we write $\chi \in \hat{\mathbb{F}}_q^*$ and $\psi \in \hat{\mathbb{F}}_q$:*

(i) *for a and $b \in \mathbb{F}_q$, we have $G(\chi,\, \psi_{ab}) = \overline{\chi(a)}G(\chi,\, \psi_b)$.*

(ii) *$G(\chi,\, \overline{\psi}) = \chi(-1)G(\chi,\, \psi)$.*

(iii) *$G(\overline{\chi},\, \psi) = \chi(-1)\overline{G(\chi,\, \psi)}$.*

Proof. Let us prove (i):

$$G(\chi,\, \psi_{ab}) = \sum_{x \in \mathbb{F}_q^*} \chi(x)\psi_b(ax) = \chi(a^{-1}) \sum_{y \in \mathbb{F}_q^*} \chi(y)\psi_b(y),$$

by performing the change of variable $ax = y$.
Property (ii) is obtained from (i) by taking $b = -1$.
Property (iii) follows from (ii) in switching to conjugation and using the fact that $\chi(-1) \in \mathbb{R}$. □

Proposition 26 (Calculation of Gaussian sums)**.** *We keep the notations of the definition 12. We then have*

$$G(\chi,\,\psi) = \begin{cases} q-1 & \textit{si} \quad \chi = \chi_0 \quad \textit{and} \quad \psi = \psi_0 & \textit{(case 1)} \\ -1 & \textit{si} \quad \chi = \chi_0 \quad \textit{and} \quad \psi \neq \psi_0 & \textit{(case 2)} \\ 0 & \textit{si} \quad \chi \neq \chi_0 \quad \textit{and} \quad \psi = \psi_0 & \textit{(case 3)} \end{cases}.$$

In the other cases, we have $|G(\chi,\,\psi)| = q^{1/2}$. In addition, we have

$$G(\chi,\,\psi)G(\overline{\chi},\,\psi) = q\chi(-1), \quad \textit{for} \quad \chi \neq \chi_0 \quad \textit{and} \quad \psi \neq \psi_0 \tag{2.13}$$

Proof. Case 1 is trivial.
Case 2 results immediately from the equation (2.5) (the term $\psi(0) = 1$ is missing in the sum).
Case 3 results, on the other hand, of the equation (2.8).
Finally, for the general case, we will exploit the fact that the function $\psi \mapsto G(\chi,\,\psi)$ is the Fourier transform (additive) of the function $\tilde{\chi}$ extended into 0, as we have already noticed in the equation (2.12). Using Plancherel's formula (1.12), we get

$$\langle G(\chi,\,\cdot),\, G(\chi,\,\cdot)\rangle = q\langle\tilde{\chi},\,\tilde{\chi}\rangle = q. \tag{2.14}$$

So let's choose an additive character $\psi = \psi_a \in \hat{\mathbb{F}_q}$. We can rewrite the equation (2.14) as follows:

$$\frac{1}{q}\sum_{b\in\mathbb{F}_q} G(\chi,\,\psi_{ab})\overline{G(\chi,\,\psi_{ab})} = q.$$

It only remains to use the result of the proposition 25, (i), to conclude

$$\frac{1}{q}\sum_{b\in\mathbb{F}_q} |\chi(b)|^2|G(\chi,\,\psi_a)|^2 = |G(\chi,\,\psi_a)|^2\langle\chi,\,\chi\rangle = |G(\chi,\,\psi_a)|^2 = q.$$

We can now prove the equality (2.13). By using the proposition 25, (iii), we obtain

$$G(\chi,\,\psi)G(\overline{\chi},\,\psi) = \chi(-1)|G(\chi,\,\psi)|^2.$$

We get the desired result using the fact that $|G(\chi,\,\psi)| = q^{1/2}$. □

These properties clearly show the importance of knowing $\chi(-1)$. A priori, we only know that $\chi(-1) \in \{-1,\,1\}$. The following proposition will tell us more.

Proposition 27. *Let χ be a multiplicative character, and let m be its order in $\hat{\mathbb{F}_q^*}$, i.e. the smallest positive integer k such that $\chi^k = \chi_0$. Then $\chi(-1) = -1$ if and only if m is even and $\frac{q-1}{m}$ is odd.*

Proof. The first thing to notice is that since $\chi^{q-1} = \psi_0$, we have $m|q-1$. Moreover, since χ has values in the set of m^{th} roots of the unit, the value -1 can only appear if m is even. So the statement of this proposition does have a meaning, which is reassuring.
We denote by g_0 a generator of $\mathbb{F}_q^*$, which implies that $\chi(g_0)$ is a root m^{th} primitive of the unit (because $\chi(g_0)$ is an element of order m in the group of complex numbers of modulus 1). Then, if m is even (therefore q is necessarily odd), we have

$$\chi(-1) = \chi\left(g_0^{(q-1)/2}\right) = \zeta^{(q-1)/2}.$$

So we have $\chi(-1) = -1$ if and only if $\zeta^{(q-1)/2} = \zeta^{m/2}$, that is to say $(q-1)/2 \equiv m/2$ modulo m. This is equivalent to $(q-1)/m \equiv 1$ modulo 2, which means that $(q-1)/m$ is odd. □

2.1.4 Quadratic reciprocity

The aim of this paragraph is to study more closely the quadratic character, to finally demonstrate the famous formula of quadratic reciprocity. In order to achieve this, we will use the additive Fourier transform $\mathcal{F}_{\text{add}}$, defined by the equation (2.11).

At $\mathcal{F}_{\text{add}}$, we'll prefer to use an endomorphism of $\mathbb{C}[\mathbb{F}_q^*]$, for convenience. This will be noted $T : \mathbb{C}[\mathbb{F}_q^*] \to \mathbb{C}[\mathbb{F}_q^*]$. It is defined as follows:

$$\forall f \in \mathbb{C}[\mathbb{F}_q^*], \quad Tf : \left\{ \begin{array}{ccc} \mathbb{F}_q^* & \longrightarrow & \mathbb{F}_q^* \\ a & \longmapsto & \sum_{x \in \mathbb{F}_q^*} f(x)\psi_a(x) \end{array} \right. .$$

The difference compared to the additive Fourier transform is due to little, since we have

$$\forall f \in \mathbb{C}[\mathbb{F}_q^*], \quad Tf(a) = \mathcal{F}_{\text{add}}\left(\tilde{f}\right)(\psi_a),$$

where we extended f to 0 by $\tilde{f}(0) = 0$. The usefulness of this operator with respect to the additive Fourier transform is that it makes the formulas in which the Gauss sums intervene simpler. The operator T is in fact nothing more than $R\mathcal{F}_{\text{add}}P$, where P is the embedding of $\mathbb{C}[\mathbb{F}_q^*]$ in $\mathbb{C}[\mathbb{F}_q]$ already described and R the surjection of the vector space $\mathbb{C}[\mathbb{F}_q]$ in $\mathbb{C}[\mathbb{F}_q^*]$. Thus, by taking the equation (2.12), we obtain

$$\forall \chi \in \hat{\mathbb{F}_q^*}, \ \forall \psi \in \hat{\mathbb{F}_q}, \quad T\chi(x) = G(\chi, \psi_x) = \overline{\chi}(x)G(\chi, \psi_1). \tag{2.15}$$

The last equality was obtained thanks to the proposition 25, property (i).

In the remainder of the discussion, we will restrict ourselves to the case where $q = p$, so that we will work in the body $\mathbb{F}_p$. In this case, the operator T is expressed as follows:

$$\forall f \in \mathbb{C}[\mathbb{F}_p^*], \quad Tf : \left\{ \begin{array}{ccc} \mathbb{F}_p^* & \longrightarrow & \mathbb{F}_p^* \\ a & \longmapsto & \sum_{x \in \mathbb{F}_p^*} f(x)\zeta^{ax} \end{array} \right. ,$$

where we noted $\zeta \overset{\text{def.}}{=} e^{\frac{2\imath\pi}{p}}$. We are now going to demonstrate a lemma which makes the link between the operator T and the quadratic character η.

Lemma 7. *Let $\eta \in \hat{\mathbb{F}_p^*}$ be the quadratic character on the field $\mathbb{F}_p$. We then have*

$$\det(T) = (-1)^{\frac{p-1}{2}} \imath^{\frac{(p-1)(p-3)}{4}} p^{\frac{p-3}{2}} G(\eta, \psi_1).$$

Proof. It is about writing the matrix of T in the base of the multiplicative characters $\{\chi_0, \ldots, \chi_{p-2}\}$. The only two characters that are real values are χ_0 and η. We can group the other characters by pairs $(\chi, \overline{\chi})$, and using the equation (2.15), we obtain a matrix of the type

$$\begin{pmatrix} G(\chi_0, \psi_1) & & & & & & & \\ & G(\eta, \psi_1) & & & & & & \\ & & 0 & G(\chi_1, \psi_1) & & & & \\ & & G(\overline{\chi_1}, \psi_1) & 0 & & & & \\ & & & & \ddots & & & \\ & & & & & & 0 & G\left(\chi_{\frac{p-3}{2}}, \psi_1\right) \\ & & & & & & G\left(\overline{\chi_{\frac{p-3}{2}}}, \psi_1\right) & 0 \end{pmatrix}.$$

We know, with the proposition 26 (case 2), that $G(\chi_0, \psi_1) = -1$. The value of $G(\eta, \psi_1)$ is currently unknown. It only remains to calculate the sub-determinants of size 2

$$\det \begin{pmatrix} 0 & G(\chi, \psi_1) \\ G(\overline{\chi}, \psi_1) & 0 \end{pmatrix} = -G(\chi, \psi_1)G(\overline{\chi}, \psi_1) = -\chi(-1)p.$$

For this calculation, we used the equality (2.13). We therefore obtain the value of the determinant

$$\begin{aligned}\det(T) &= -G(\eta,\, \psi_1)(-p)^{\frac{p-3}{2}} \prod_{j=1}^{\frac{p-3}{2}} \chi_j(-1)\\ &= (-1)^{\frac{p-1}{2}} p^{\frac{p-3}{2}} G(\eta,\, \psi_1) \prod_{j=1}^{\frac{p-3}{2}} \chi_j(-1).\end{aligned}$$

However, $\chi_j(-1) = \chi_1(-1)^j = (-1)^j$, which provides an evaluation of the product on the right

$$\prod_{j=1}^{\frac{p-3}{2}} \chi_j(-1) = (-1)^{1+2+\cdots+\frac{p-1}{3}} = (-1)^{\frac{(p-1)(p-3)}{8}} = \imath^{\frac{(p-1)(p-3)}{4}}.$$

This corresponds well to the announced result. □

We can now state an important result. It is a question of calculating the sums of Gauss bringing into play the quadratic character. It was proved by *Gauss*, and allowed him to provide, in 1807, the 6rd of his 8 proofs of the quadratic reciprocity formula.

Proposition 28 (Signs of Gaussian sums)**.** *Let p be an odd prime number. We denote by η the quadratic character of $\mathbb{F}_p$. So*

$$G(\eta,\, \psi_1) = \begin{cases} p^{1/2} & si p \equiv 1 \mod 4 \\ \imath p^{1/2} & si p \equiv 3 \mod 4 \end{cases}.$$

Proof. Like $\eta = \overline{\eta}$, by applying the equality (2.13), it comes

$$G(\eta,\, \psi_1)^2 = \eta(-1)p.$$

We can then use the proposition 27, and see that

$$\eta(-1) = \begin{cases} 1 & \text{si} p \equiv 1 \mod 4 \\ -1 & \text{si} p \equiv 3 \mod 4 \end{cases}.$$

We thus obtain almost the desired result, that is to say

$$G(\eta,\, \psi_1) = \begin{cases} \epsilon_p\, p^{1/2} & \text{si} p \equiv 1 \mod 4 \\ \epsilon_p\, \imath p^{1/2} & \text{si} p \equiv 3 \mod 4 \end{cases} \qquad \text{with} \epsilon_p \in \{+1,\, -1\}.$$

The whole difficulty lies in determining the signs, that is to say of ϵ_p (quantity which depends a priori on p). Let's start by rewriting the last equality in a more compact way:

$$G(\eta,\, \psi_1) = \epsilon_p \imath^{\frac{(p-1)^2}{4}} p^{1/2}.$$

This equality comes simply from the fact that

$$\imath^{\frac{(p-1)^2}{4}} = \begin{cases} 1 & \text{si} p \equiv 1 \mod 4 \\ \imath & \text{si} p \equiv 3 \mod 4 \end{cases}.$$

We are going to be able to use the calculation of $\det(T)$ that we have just carried out in lemma 7. By inserting the value of $G(\eta,\, \psi_1)$ in this determinant, it comes

$$\det(T) = \epsilon_p (-1)^{\frac{p-1}{2}} \imath^{\frac{(p-1)(p-3)}{4}} \imath^{\frac{(p-1)^2}{4}} p^{\frac{p-3}{2}} p^{\frac{1}{2}} \tag{2.16}$$

$$= \epsilon_p (-1)^{\frac{p-1}{2}} \imath^{\frac{(p-1)(p-2)}{2}} p^{\frac{p-2}{2}}. \tag{2.17}$$

To determine the sign that appears in this expression, we are going to calculate the determinant in another base, that of the Dirac functions $\{\delta_1, \dots, \delta_{p-1}\}$. Since we have $T\delta_k(x) = \zeta^{xk}$, we get

$$\det(T) = \det\left(\zeta^{jk}\right)_{1\le j,k\le p-1} = \prod_{1\le m<n\le p-1} (\zeta^n - \zeta^m).$$

The last equality is obtained by calculating a determinant of *Vandermonde*. By going to the half angle, that is to say by setting $\mu \overset{\text{def.}}{=} e^{\frac{\imath\pi}{p}}$, it comes

$$\begin{aligned}\det(T) &= \prod(\mu^{2n} - \mu^{2m}) = \prod_{m<n} \mu^{m+n}(\mu^{nm} - \mu^{mn}) \\ &= \prod \mu^{n+m} \prod \imath \prod 2\sin\left(\frac{\pi(nm)}{p}\right),\end{aligned}$$

where the sign $\prod$ means $\prod_{m<n}$. You have to evaluate the three products that appear in this expression. Regarding the product on the right, it is positive, and this is sufficient for what we want to do with it:

$$\prod 2\sin\left(\frac{\pi(nm)}{p}\right) = A > 0.$$

For the middle product, it suffices to notice that

$$\sharp\{(m, n) \ : \ 1 \le m < n \le p-1\} = \frac{(p-1)(p-2)}{2}$$

(we can make a drawing and count the number of points with integer coordinates inside a triangle). So we get

$$\prod \imath = \imath^{\frac{(p-1)(p-2)}{2}}.$$

As for the product on the left, we use the following calculation:

$$\begin{aligned}\sum_{1\le m<n\le p-1} n+m &= \sum_{n=2}^{p-1}\sum_{m=1}^{n-1} n+m = \frac{3}{2}\sum_{n=1}^{p-2} n(n-1) \\ &= \frac{3}{2}\left(\frac{(p-2)(p-3)(2p-3))}{6} + \frac{(p-1)(p-2)}{2}\right) \\ &= \frac{p(p-1)(p-2)}{2},\end{aligned}$$

from where

$$\prod_{1\le m<n\le p-1} \mu^{n+m} = \mu^{\frac{p(p-1)(p-2)}{2}} = (-1)^{\frac{(p-1)(p-2)}{2}}.$$

We finally obtain the expression of the determinant of T:

$$\det(T) = (-1)^{\frac{p-1}{2}} \imath^{\frac{(p-1)(p-2)}{2}} A \quad \text{with} A > 0.$$

By comparing this expression to the equation (2.17), we see that $\epsilon_p = +1$. □

Remark 19. This result generalizes to the case of an arbitrary $\mathbb{F}_q$ field, that is to say for $q = p^s$. We can indeed state:

$$G(\eta, \psi_1) = \begin{cases} (-1)^{s-1} q^{1/2} & \text{si} q \equiv 1 \mod 4 \\ (-1)^{s-1}\imath^s q^{1/2} & \text{si} q \equiv 3 \mod 4 \end{cases}.$$

The proof of this result goes through the proof of a lemma that we find in the book of Lidl and Niederreiter [44].

After these somewhat computational proofs, we are finally able to prove the famous quadratic reciprocity formula. It was stated by *Legendre* in 1788, and demonstrated for the first time by *Gauss* in 1801. To date, there are several hundred different proofs. Franz Lemmermeyer has collected a large quantity of them, and presents the first part of the history of this formula in [31].

Théorem 3 (Quadratic reciprocity)**.** *For all distinct odd prime numbers p and r, we have*

$$\left(\frac{p}{r}\right)\left(\frac{r}{p}\right) = (-1)^{\frac{(p-1)(r-1)}{4}}. \tag{2.18}$$

Proof. Let η be the quadratic multiplicative character of $\mathbb{F}_p$, and ψ_1 the canonical additive character. With the previous theorem, we know that

$$G(\eta,\, \psi_1)^2 = (-1)^{\frac{p-1}{2}} p \overset{\text{def.}}{=} \tilde{p}.$$

We now denote by $G \overset{\text{def.}}{=} G(\eta,\, \chi_1)$. We have

$$G^r = \left(G^2\right)^{\frac{r-1}{2}} G = \tilde{p}^{\frac{r-1}{2}} G.$$

In the following, we will perform our calculations in the ring R of algebraic integers, that is to say of complex numbers which are roots of unit polynomials with integer coefficients. As the characters are sums of roots of the unit, the values of the Gaussian sums are algebraic integers, so $G \in R$. We denote by (r) the ideal generated by r in R. Since the quotient ring $R/(r)$ is of characteristic r, we obtain

$$G^r = \left(\sum_{x\in\mathbb{F}_p^*} \eta(x)\psi_1(x)\right)^r = \sum_{x\in\mathbb{F}_p^*} \left(\eta(x)^r \psi_1(x)^r\right) \mod (r).$$

As $\eta(x)^r = \eta(x)$ (because $\eta(x) \in \{-1,\, 1\}$ and r is odd) and $\psi_1(x)^r = \psi_r(x)$, we get

$$G^r = \sum_{x\in\mathbb{F}_p^*} \eta(x)\psi_r(x) = G(\eta,\, \psi_r) = \psi(r)G \mod (r).$$

For the last equality, we used the result (i) of the proposition 25. So we have

$$G^r = \eta(r)G = \tilde{p}^{\frac{r-1}{2}} G \mod (r).$$

By multiplying this equality by G, and using the fact that $G^2 = \tilde{p}$, we obtain the following equality:

$$\tilde{p}^{\frac{r-1}{2}} \tilde{p} = \eta(r)\tilde{p} \mod (r).$$

This tie is actually a tie over $\mathbb{Z}/r\mathbb{Z}$. As p and r are coprime, we can simplify by $\tilde{p}$, to get

$$\tilde{p}^{\frac{r-1}{2}} = (-1)^{\frac{(p-1)(r-1)}{4}} p^{\frac{r-1}{2}} = \eta(r) \mod r.$$

As we have already pointed out, we have $\eta(r) = \left(\frac{r}{p}\right)$. Moreover, with Euler's formula (lemma 6), we have $p^{\frac{r-1}{2}} = \left(\frac{p}{r}\right)$. So we actually have the following equality:

$$(-1)^{\frac{(p-1)(r-1)}{4}} \left(\frac{p}{r}\right) = \left(\frac{r}{p}\right) \mod r.$$

Since both members of this equality are in fact valued in $\{-1,\, 1\}$, and $r \geq 3$, this equality is in fact valid on $\mathbb{Z}$, which is the conclusion we wanted to reach. □

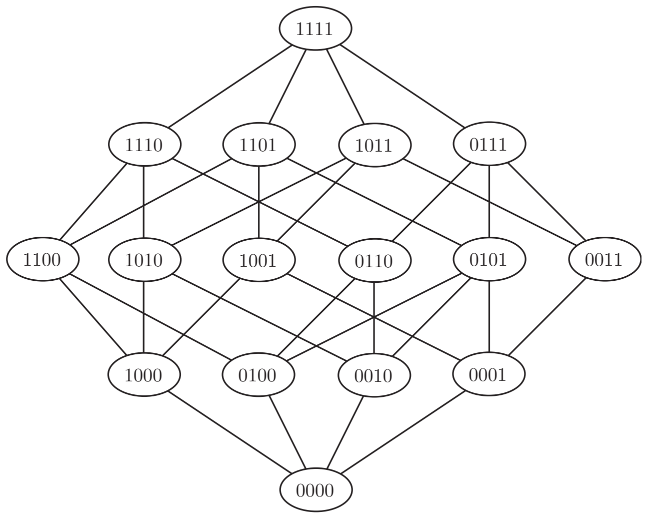

Figure 2.1: Boolean cube $(\mathbb{F}_2)^4$

2.2 Walsh transform

Before presenting, in the next chapter, a series of fast algorithms for computing Fourier transforms on a cyclic group, we will describe a transform which also has a fast algorithm. This is the transform of *Walsh*. Behind this name hides in fact a rewriting of the Fourier transform on an abelian group, in a very particular case, that of the group $G = (\mathbb{Z}/2\mathbb{Z})^k$. This group is often called *boolean cube*, and we can see a drawing of the cube of dimension 4 in the figure 2.1.

2.2.1 Presentation

By closely following the proof of the corollary 2, we can easily construct the dual of a group which is written as a product of elementary cyclic groups. Indeed, each character will be expressed as a product of the different characters of the elementary groups. In the case of the group $(\mathbb{Z}/2\mathbb{Z})^k$, this is extremely simple, since the only non-trivial character of the group $\mathbb{Z}/2\mathbb{Z} = \{0, 1\}$ is defined through

$$\chi_\epsilon(0) = 1 \qquad \chi_\epsilon(1) = -1.$$

To each element $a = \{a_0, \dots, a_{k-1}\} \in (\mathbb{Z}/2\mathbb{Z})^k$ we can therefore assign a character

$$\chi_a : \left\{ \begin{array}{ccc} (\mathbb{Z}/2\mathbb{Z})^k & \longrightarrow & \{-1, 1\} \\ x = \{x_0, \dots, x_{k-1}\} & \longmapsto & (-1)^{a_0 x_0}(-1)^{a_1 x_1} \cdots (-1)^{a_{k-1} x_{k-1}} = (-1)^{\langle a,\, x\rangle} \end{array} \right. . \tag{2.19}$$

where we have denoted $\langle a, x\rangle$ the canonical bilinear form over $(\mathbb{Z}/2\mathbb{Z})^k$ defined by

$$\langle a,\, x\rangle \overset{\text{def.}}{=} \sum_{i=0}^{k-1} a_i x_i.$$

In practice, we represent the elements of the group $(\mathbb{Z}/2\mathbb{Z})^k$ as integers between 0 and $2^k - 1$, by assimilating the element $x \in G = (\mathbb{Z}/2\mathbb{Z})^k$ with the integer $\sum_{i=0}^{k-1} x_i 2^i$. This allows to see the characters χ_a (where a can be seen as an element of G or as an integer of $\{0, \dots, 2^k - 1\}$) as vectors of size 2^k filled with -1 and 1.

Example 2. Consider the group $(\mathbb{Z}/2\mathbb{Z})^3$, of cardinal 8. We can represent its character table as a square matrix of order 8 , denoted W_8, whose line i represents the values of the character χ_i, that is to say that

$(W_8)_{ij} = \chi_i(j)$. Here is the table:

$$W_8 \overset{\text{def.}}{=} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix}. \tag{2.20}$$

The figure 2.2 shows the Walsh matrix W_{64}, where we have made the entries equal to 1 white, and those equal to -1 in black.

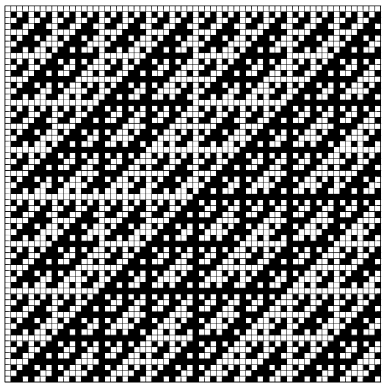

Figure 2.2: Walsh matrix W_{64}

We can then define the *Walsh transform.*

Definition 13 (Walsh transform)**.** *We define the Walsh transform $\mathcal{W}_k(f)$ of a complex vector $f = \{f[0], \dots, f[2^k - 1]\}$ of size 2^k by*

$$\forall i \in \{0, \dots, 2^k - 1\}, \quad \mathcal{W}_k(f)[i] \overset{\text{def.}}{=} \sum_{j=0}^{2^k-1} f[j]\chi_i(j) = 2^k \langle f, \chi_i \rangle. \tag{2.21}$$

Remark 20. (**Link with the Fourier transform**). We should also note $f : (\mathbb{Z}/2\mathbb{Z})^k \to \mathbb{C}$ the function corresponding to the vector f, that is to say the vector corresponding to the decomposition of f in the basis of Dirac functions $\{\delta_x\}_{x \in G}$. Then the computation of $\mathcal{W}(f)$ is that of a Fourier transform, since

$$\mathcal{W}(f)[i] = 2^k \langle f, \chi_i \rangle = \hat{f}(\chi_i),$$

where we have denoted $\hat{f} : \hat{G} \to \mathbb{C}$ the Fourier transform of f. In particular, we will therefore be able to state without effort the inversion formula for the Walsh transform.

If we represent the frequencies $i \in \{0, \dots, 2^n - 1\}$ on the boolean cube of the figure 2.1, then the frequencies located in bottom of the diagram are often referred to as *low frequencies.* We find many analogies with the Fourier spectra already encountered (Fourier transforms on a finite group, continuous transform, etc.). Thus, the exercise 60 proposes to use the spectral properties of the Walsh transform to perform boolean predictions (this is related to the *theory of learning*) .

The operator $\mathcal{W}_k : \mathbb{C}^{2^k} \to \mathbb{C}^{2^k}$ is a linear map whose matrix in the canonical basis is W_{2^k}, the table of characters in the group $(\mathbb{Z}/2\mathbb{Z})^k$. We thus have $\mathcal{W}_k(f) = W_{2^k} f$. The inversion formula obtained from the proposition 13, gives us the following information.

Proposition 29. *The Walsh transform is invertible, and its inverse is* $\frac{1}{2^k}\mathcal{W}_k$. *From a matrix point of view, this means that the matrix* $W_{2^k} = \{w_{ij}\}$, *defined by* $w_{ij} = (-1)^{\langle i,\, j\rangle}$, *check* $W_{2^k}W_{2^k} = 2^k Id$.

The Walsh matrix makes it possible to study concrete problems, for example in statistics, as shown by the exercise 14.

2.2.2 Fast calculation algorithm

One of the advantages of the Walsh transform is that we have a fast algorithm to calculate it. Indeed, although the equation that defines it may seem a bit complicated, it can be broken down in a simple way. This will make it possible to implement a recursive calculation algorithm, much more efficient than the naive evaluation of the sum which defines the transform. We will study in the next chapter the construction of *the FFT algorithm*, and we will find exactly the same ideas, which are at the base of the algorithmic "philosophy" called *divide and conquer*.

Rather than spending time on the analysis of such an algorithm (its cost, its implementation, etc.), we will simply describe the recurrence equation that we will put intoœ work. Discussions on the efficiency of such an implementation are postponed to the next chapter, about *the FFT algorithm*. Here is the famous idea which is the basis of our algorithm. We need to rewrite the equation (2.21), as follows:

$$\mathcal{W}_k(f)[i] = \sum_{j=0}^{2^{k-1}-1} f[j](-1)^{\sum_{p=0}^{k-2} j_p i_p} + (-1)^{i_{k-1}} \sum_{j=0}^{2^{k-1}-1} f[j+2^{k-1}](-1)^{\sum_{p=0}^{k-2} j_p i_p}.$$

To write this expression in a simpler way, we introduce the vectors f_1 and f_2 of length 2^{k-1} defined as follows:

$$\forall j \in \{0, \ldots, 2^{k-1}-1\}, \quad f_1[j] = f[j] \quad \text{and} \quad f_2[j] = f[j+2^{k-1}].$$

Similarly, we will write $\mathcal{W}_k(f)_1$ (respectively $\mathcal{W}_k(f)_2$) to denote the 2^{k-1} prime (respectively last) indices of the transformed vector $\mathcal{W}_k(f)$. We then have the recurrence equation

$$\mathcal{W}_k(f)_1 = \mathcal{W}_{k-1}(f_1) + \mathcal{W}_{k-1}(f_2)$$
$$\mathcal{W}_k(f)_2 = \mathcal{W}_{k-1}(f_1) - \mathcal{W}_{k-1}(f_2).$$

From a matrix point of view this decomposition is written in the form

$$W_{2^k} = \begin{pmatrix} W_{2^{k-1}} & W_{2^{k-1}} \\ W_{2^{k-1}} & -W_{2^{k-1}} \end{pmatrix}.$$

The decomposition of W_{2^k} found corresponds to a tensor product structure, as specified in the exercise 18. This decomposition quite naturally gives rise to a very fast calculation algorithm, named *FWT* for **F** ast **W** alsh **T** ransform. More precisely, if we count the number of additions necessary to calculate the Walsh transform of a vector of size $n = 2^k$, we see that we obtain $k = \log_2(n)$ recursive calls, with each time n additions or subtractions. Hence a cost of $n\log_2(n)$ operations. This is a substantial gain over the naive implementation of the (2.21) equation, which requires n^2 operations. The program Matlab of this algorithm is presented in paragraph B.1.

2.2.3 Use of the Walsh transform

The main interest of the Walsh transform is that it allows to decompose any function from $\{0, \ldots, 2^k-1\}$ into $\mathbb{C}$ on the orthogonal basis of the characters of $(\mathbb{F}_2)^k$. More precisely, we have

$$\forall i \in \{0, \ldots, 2^k-1\}, \quad f[i] = \frac{1}{2^k} \sum_{j=0}^{2^{k-1}} \mathcal{W}_k(f)[j]\chi_j(i).$$

This transform is very quick to use, because it only uses addition and subtraction (no multiplication). In addition, we have seen that we have a formidably efficient algorithm for recursively calculating the Walsh transform . However, the Walsh transform has a weak point of size: it has no pleasant property with respect to the convolution of the functions of $\{0, \ldots, 2^k - 1\}$ in $\mathbb{C}$, unlike the Fourier transform, as we saw in paragraph 1.4.3. Indeed, the Walsh transform is a Fourier transform on the binary cube $(\mathbb{Z}/2\mathbb{Z})^k$, not at all on the cyclic group $\mathbb{Z}/2^k\mathbb{Z}$. It is mainly because of this that we will be led in the next chapter to study more closely the Fourier transform on a cyclic group. This will allow us to build an algorithm to compute convolutions on $\mathbb{Z}/2^k\mathbb{Z}$ extremely fast.

The exercise 16 shows how we can use the Walsh transform to achieve *signal compression*. He also proposes to extend the Walsh transform to the two-dimensional frame.

Finally, the Walsh transform allows us to study *boolean functions*, and in particular their *non-linearity*. As this study supposes to consider functions with values in the finite-field $\mathbb{F}_2$, it is approached only at the end of the chapter 6, at exercise 59. In the same vein, the exercise 60 introduces probabilistic notions to study Boolean functions and their learning.

2.3 Poisson formula

We have seen, in particular in Paragraph 1.2.3 devoted to bidual, the great similarity between duality on a finite abelian group, and duality on a vector space. In this paragraph, we will give another embodiment of this fact, in this case by studying the notion of orthogonality between a group and its dual. The central point of this approach is the *Poisson formula*. We will thus see that if we apply this formula in the case of a group which is also a vector space (on a finite field), we obtain very powerful relations, called identities of *MacWilliams*.

2.3.1 The formula on a finite abelian group

Before stating the Poisson formula on a finite group, it is necessary to clarify the notion of orthogonality. Let us start by briefly recalling the theory of orthogonality on a vector space. For a fuller description, we can refer to the work of Ramis, Dechamps and Odoux [54].

If E is a finite dimensional k-vector space, we denote by $E^* \overset{\text{def.}}{=} \operatorname{Hom}(E, k)$ its dual, which is made up of linear forms. We classically define a bilinear form on $E^* \times E$, which we call the hook of duality, as follows:

$$\forall (f, x) \in E^* \times E, \quad \langle x, f\rangle \overset{\text{def.}}{=} f(x).$$

We then define, for a part $A \subset E$, its orthogonal

$$A^\perp \overset{\text{def.}}{=} \{f \in E^* \ : \ \forall x \in A, \ \langle x, f\rangle = 0\}. \tag{2.22}$$

We verify that it is a vector subspace of E, and we have $E^\perp = \{0\}$. Similarly, we define the orthogonal of $B \subset E^*$ by

$$B^0 = \{x \in E \ : \ \forall f \in B, \ \langle x, f\rangle = 0\}.$$

It is a vector subspace of E^*, and we have $(E^*)^0 = \{0\}$. Note that these properties are still true in infinite dimension, but we must use the axiom of choice. In *finite* dimension, the maps $F \mapsto F^\perp$ and $G \mapsto G^0$ are reciprocal bijections between subspaces of E and E^*. They reverse inclusion.

Once in memory these linear notions of orthogonality, it is natural to introduce the following definition:

Definition 14 (Orthogonal of a subgroup). *Let G be a finite abelian group, and $H \subset G$ a subgroup. We denote by $H^\sharp$ the orthogonal of H which is the subgroup of $\hat{G}$ defined as follows:*

$$H^\sharp \overset{\text{def.}}{=} \left\{\chi \in \hat{G} \ : \ \forall h \in H, \ \chi(h) = 1\right\}.$$

We have already seen during the demonstration of the lemma 2, that any trivial character of $\hat{G}$ over H is uniquely identified with an element of $\widehat{G/H}$, and vice versa. More precisely, we have an isomorphism $H^\sharp \simeq \widehat{G/H}$. We therefore see that $H^\sharp$ is a subgroup of $\hat{G}$ of cardinality $|G|/|H|$. For example, we have $G^\sharp = \{1\}$. In addition, the application $H \mapsto H^\sharp$ reverses inclusions. Here again, the resemblance to the duality between the vector spaces is striking.

We can now state the *Poisson formula* in the framework of finite abelian groups.

Théorem 4 (Poisson formula)**.** *Let G be a finite abelian group, and $H \subset G$ a subgroup. So, for $f : G \to \mathbb{C}$, that is to say $f \in \mathbb{C}[G]$, we have*

$$\forall g \in G, \quad \sum_{h \in H} f(gh) = \frac{|H|}{|G|} \sum_{\chi \in H^\sharp} \hat{f}(\overline{\chi})\chi(g). \tag{2.23}$$

We denote by $\hat{f} : \hat{G} \to \mathbb{C}$ the Fourier transform of f.

Proof. To simplify the notations, we consider S a system of representatives of G/H in G. We denote by $\overline{g}$ the image of $g \in S$ in G/H (that is to say the image by canonical projection). Let's start by defining a function $\tilde{f}$ on G/H by

$$\forall g \in S, \quad \tilde{f}(\overline{g}) \overset{\text{def.}}{=} \sum_{h \in H} f(gh).$$

This amounts to replacing f by an invariant function under the translation by elements of H (we "periodize"). We see that this function is defined without ambiguity, since, if we consider another representative g' of the class gH, we have $g' = gh'$ with h' an element of H, and therefore, $\tilde{f}(\overline{g}) = \tilde{f}(\overline{g'})$. We can therefore decompose the function $\tilde{f} \in \mathbb{C}[G/H]$ into a Fourier series:

$$\forall g \in S, \quad \tilde{f}(\overline{g}) = \sum_{\chi \in \widehat{G/H}} \langle \tilde{f}, \chi \rangle \chi(\overline{g}). \tag{2.24}$$

We can then explain the value of the Fourier coefficients:

$$\langle \tilde{f}, \chi \rangle = \frac{1}{|G/H|} \sum_{g \in S} \tilde{f}(\overline{g})\overline{\chi(\overline{g})} = \frac{|H|}{|G|} \sum_{g \in S} \sum_{h \in H} f(gh)\overline{\chi(\overline{g})}. \tag{2.25}$$

Noticing that the application

$$\begin{cases} S \times H & \longrightarrow & G \\ (g, h) & \longmapsto & gh \end{cases}$$

is a bijection and using the fact that for $\chi \in \widehat{G/H}$, $\chi(\overline{gh}) = \chi(\overline{g})$, we can rewrite the sum (2.25) in the form

$$\langle \tilde{f}, \chi \rangle = \frac{|H|}{|G|} \sum_{g \in G} f(g)\overline{\chi(g)} \overset{\text{def.}}{=} \frac{|H|}{|G|} \hat{f}(\overline{\chi}).$$

All that remains is to report the value of these coefficients in the equation (2.24) and notice that the expression of $\tilde{f}(\overline{g})$ gives us the left side of the Poisson formula. □

By applying the equation (2.23) with $g = 1$, we obtain the form in which the formula is often written:

$$\sum_{h \in H} f(h) = \frac{|H|}{|G|} \sum_{\chi \in H^\sharp} \hat{f}(\chi). \tag{2.26}$$

In order to better understand the Poisson formula, we will give another proof, which uses only linear algebra arguments on the vector space $\mathbb{C}[G]$. Before giving this proof, let us study more precisely the space $\mathbb{C}[G/H]$.

If we denote by $\pi : G \to G/H$ the canonical projection, we can define an application

$$\pi^* : \left\{ \begin{array}{ccc} \mathbb{C}[G/H] & \longrightarrow & \mathbb{C}[G] \\ f & \longmapsto & f \circ \pi \end{array} \right. .$$

π^* is in fact an injective linear map (because π is surjective). The space $\mathbb{C}[G/H]$ is therefore identified with a vector subspace of $\mathbb{C}[G]$, which is in fact formed by constant functions on each of the classes on the left modulo H. To demonstrate this fact, it suffices to note that π^* can be inverted on its image as follows:

$$(\pi^*)^{-1} \left\{ \begin{array}{ccc} \Im(\pi^*) & \longrightarrow & \mathbb{C}[G/H] \\ f & \longmapsto & \tilde{f} \end{array} \right. ,$$

where we denote by $\tilde{f}(\overline{x})$ (for $x \in G$) the value of f on the class to the left of x modulo H. This identification being made, we can give a new proof of the Poisson formula.

Proof. As before, we fix a function $f \in \mathbb{C}[G]$. Recall that the space $\mathbb{C}[G]$ is endowed with an algebra structure thanks to the convolution product $*$, whose expression is given to the equation (1.15). We can then consider a filtering operator

$$\Phi^f : \left\{ \begin{array}{ccc} \mathbb{C}[G] & \longrightarrow & \mathbb{C}[G] \\ \varphi & \longmapsto & f * \varphi \end{array} \right. .$$

We will see in the Section 4.2, why we call such operators filter operators. By using the identification between the functions of $\mathbb{C}[G/H]$ and the functions of $\mathbb{C}[G]$ constants on the classes on the left gH, we can show that $\mathbb{C}[G/H]$ is a subspace of $\mathbb{C}[G]$ stable by Φ^f. Indeed, if we give ourselves φ constant on the classes on the left, and $g' = gh'$, for $g \in G$ and $h' \in G$, we obtain

$$\Phi^f(\varphi)(g') = \sum_{x \in G} f(gh'x^{-1})\varphi(x) = \sum_{x \in G} f(g(xh'^{-1})^{-1})\varphi(xh'^{-1}) = \Phi^f(\varphi)(g).$$

This being shown, we can therefore consider $\tilde{\Phi}^f$, the restriction of Φ^f to $\mathbb{C}[G/H]$. The trick to finding the Poisson formula is to calculate the trace of $\tilde{\Phi}^f$. We have an obvious basis of $\mathbb{C}[G/H]$, namely $\mathcal{B} \stackrel{\text{def.}}{=} \{\delta_g\}_{g \in S}$ (we always denote S a system of representative of G/H). We recall that δ_g is the function which is worth 1 in g, and 0 everywhere else. In this base, the expression of the operator trace is simple to calculate, since the calculation of the images of the base vectors gives

$$\forall s \in S, \quad \tilde{\Phi}^f(\delta_s) : g \mapsto \sum_{x \in S} \sum_{h \in H} f(gx^{-1}h^{-1})\delta_s(x),$$

because we must remember that δ_s is seen as a constant function on the classes on the left. As a result, we get

$$\forall s \in S, \quad \tilde{\Phi}^f(\delta_s) : g \mapsto \sum_{h \in H} f(gs^{-1}h^{-1}).$$

The trace is therefore calculated effortlessly:

$$\operatorname{Tr}\left(\tilde{\Phi}^f\right) = \sum_{s \in S} \sum_{h \in H} f(ss^{-1}h^{-1}) = \frac{|G|}{|H|} \sum_{h \in H} f(h).$$

Up to a constant, we get the left side of the Poisson formula. To obtain the right-hand side, it will suffice to calculate the trace of $\tilde{\Phi}^f$ in another base. And of course, we are going to reinvest the work done in the previous chapter by choosing the orthonormal basis of the characters of G/H, that is to say the elements of $H^\sharp$. The advantage is that the characters behave in a particularly pleasant way vis-à-vis the convolution. Indeed, the convolution theorem 2, allows to show that the elements of $H^\sharp$ are the eigenvectors of the operator $\tilde{\Phi}^f$, since

$$\forall \chi \in \widehat{G/H} = H^\sharp, \quad \tilde{\Phi}^f(\chi) = \hat{f}(\chi)\chi.$$

The exercise 1, question 1, details the proof of this. It only remains to exploit the fact that the trace of the operator $\tilde{\Phi}^f$ is equal to the sum of its eigenvalues, to conclude that

$$\operatorname{Tr}\left(\tilde{\Phi}^f\right) = \sum_{\chi \in H^\sharp} \hat{f}(\chi).$$

We therefore find the simplified Poisson formula (2.26). To obtain the complete equation (2.23), it suffices to apply the obtained equation to the function $f_g : x \to f(gx)$, for $g \in G$, since we have

$$\forall \chi \in \hat{G}, \quad \hat{f}_g(\chi) = \chi(g^{-1})\hat{f}(\chi).$$

This reasoning shows moreover that the equations (2.26) and (2.23) are completely equivalent. □

2.3.2 Application to MacWilliams identities

In this paragraph, we will use the Poisson formula as part of the group G equal to $(\mathbb{Z}/2\mathbb{Z})^k = (\mathbb{F}_2)^k$. We are therefore going to place ourselves in the framework where we carried out the study of the Walsh transform (Section 2.2), and we will use the same notations. The interest of such a group is that it is in addition a vector space, in this case on the finite field $\mathbb{F}_2$. In reality, the situation is very simple, since the notion of subgroup coincides with that of vector subspace (the group operation corresponds to the addition of vectors, and since the field has only two elements, the multiplication of a vector by a scalar is an operation which trivially respects the subgroup structure). We therefore do not lose information by translating the statements resulting from duality over a group (Poisson's formula) into the language of linear algebra. We will see that in this framework, the notions of duality and orthogonality are the same for the two structures.

All the study made in this paragraph generalizes without modification to the case of the vector space K^k, where K is any finite field. The exercise 19 takes this construction step by step. However, to stay within the framework developed for the Walsh transform, we will restrict ourselves to the case of the space $(\mathbb{F}_2)^k$.

Remember that we have a complete description of the dual $\hat{F_2^k}$, since at each element $a = \{a_0, \ldots, a_{k-1}\} \in (\mathbb{Z}/2\mathbb{Z})^k$ we match a character

$$\chi_a : \left\{ \begin{array}{ccc} (\mathbb{Z}/2\mathbb{Z})^k & \longrightarrow & \{-1, 1\} \\ x = \{x_0, \ldots, x_{k-1}\} & \longmapsto & (-1)^{\langle a, x\rangle} \end{array} \right. .$$

This allows us to calculate, for a group $H \subset G$, the orthogonal $H^\sharp$. Indeed, to say that $\chi_a \in H^\sharp$ is equivalent to

$$\forall h \in H, \quad \chi_a(h) = (-1)^{\langle a, h\rangle} = 1.$$

This therefore means that

$$\forall h \in H, \quad \langle a, h\rangle = 0 \Leftrightarrow a \in H^\perp, \tag{2.27}$$

where we have denoted $H^\perp$ the orthogonal of H when we consider H as a vector subspace of $(\mathbb{F}_2)^k$. This orthogonal can of course be seen as the orthogonal for the canonical symmetric bilinear form on $(\mathbb{F}_2)^k$. We can also see it as the orthogonal in the sense of duality (definition (2.22)), if we have identified the vector space $(\mathbb{F}_2)^k$ and its dual by identifying the canonical base with its dual base. However, be careful that the orthogonal space $H^\perp$ has no reason to be an additional of H, for example, in $(\mathbb{F}_2)^4$, the vector $(1, 1, 1, 1)$ is orthogonal to itself. The exercise 88 precisely studies the cases where the space H coincides with its dual (it uses the language of error correcting codes and group actions).

In the end, if we identify the elements $a \in G$ and $\chi_a \in \hat{G}$, we obtain the remarkable property that $H^\sharp = H^\perp$. We can now state the Poisson formula in terms of vector spaces.

Proposition 30 (Vector Poisson formula)**.** *Let H be a vector subspace of $(\mathbb{F}_2)^k$. Let f be a function $f : (\mathbb{F}_2)^k \to \mathbb{C}$. We then have the two relations*

$$\sum_{a\in H} f(a) = \frac{|H|}{2^k} \sum_{u\in H^\sharp} \hat{f}(\chi_u). \tag{2.28}$$

$$\sum_{a\in H^\sharp} f(a) = \frac{1}{|H|} \sum_{u\in H} \hat{f}(\chi_u). \tag{2.29}$$

Remember that

$$\hat{f}(\chi_u) \overset{\text{def.}}{=} \sum_{x\in(\mathbb{F}_2)^k} (-1)^{\langle u,\, x\rangle} f(x).$$

Proof. The first equation is the exact translation of the Poisson formula, with the identification $H^\perp = H^\sharp$ that we have updated. For the second equation, it suffices to replace in the first H by its orthogonal $H^\perp$. Like $|H^\perp| = \frac{|(\mathbb{F}_2)^k|}{|H|}$, we get the desired result. □

We will now be able to apply the Poisson formula for combinatorial purposes. The goal of this study is to better understand the structure of the space $(\mathbb{F}_2)^k$ when it is provided with a somewhat particular distance, which we call the distance of *Hamming*. For now, it is above all a computational exercise, which will reveal some rather spectacular relationships. However, we will see in Section 6.3, that all of this has applications in the study of binary corrective codes. But let us content ourselves, first of all, with defining this famous distance.

Definition 15 (Hamming distance)**.** *Let x and $y \in (\mathbb{F}_2)^k$. We define the Hamming distance $d(x,\, y)$ between these two vectors as follows:*

$$d(x,\, y) \overset{\text{def.}}{=} w(xy) \quad \textit{with} \quad w(z) = \sharp\left\{i = 0, \ldots, k-1 \ : \ z_i \neq 0\right\}.$$

We call $w(z)$ the weight of the vector z.

The figure 2.1 represents the group $(\mathbb{F}_2)^4$ where we have connected the elements at a distance of 1. In order to study the structure of a subspace H with respect to the distance d, we are interested in the weight distribution of the words that form H. The following definitions are then introduced.

Definition 16 (Enumerator polynomial)**.** *Let H be a vector subspace of $(\mathbb{F}_2)^k$. We denote by $A_H \in \mathbb{Z}[X,\, Y]$ the enumerator polynomial of weight of H, which is defined by*

$$A_H(X,\, Y) \overset{\text{def.}}{=} \sum_{c\in H} X^{kw(c)} Y^{w(c)} = \sum_{i=0}^{k} A_i X^{ki} Y^i,$$

where we denote by A_i the number of vectors of H of weight i.

The following relation, discovered by MacWilliams, relates the weights of the words in the space H with the weights of the words of its orthogonal.

Théorem 5 (MacWilliams identity)**.** *Let H be a vector subspace of $(\mathbb{F}_2)^k$. We then have*

$$A_{H^\perp}(X,\, Y) = \frac{1}{|H|} A_H(X+Y,\, XY).$$

Proof. Let x and y be two fixed complex numbers. We then define the function $f \in \mathbb{C}[(\mathbb{F}_2)^k]$ by

$$\forall a \in (\mathbb{F}_2)^k, \quad f(a) \overset{\text{def.}}{=} x^{kw(a)} y^{w(a)}.$$

In order to apply the Poisson formula, we need to calculate $\hat{f}$:

$$\forall a \in (\mathbb{F}_2)^k, \quad \hat{f}(\chi_a) \overset{\text{def.}}{=} \sum_{t \in (\mathbb{F}_2)^k} x^{kw(t)} y^{w(t)} (-1)^{\langle t,\, a \rangle}.$$

Using the fact that $w(t) = \sum_{i=0}^{k-1} t_i$ in $\mathbb{N}$, where $t_i \in \{0,\, 1\}$, we obtain

$$\forall a \in (\mathbb{F}_2)^k, \quad \hat{f}(\chi_a) = \sum_{t \in (\mathbb{F}_2)^k} \prod_{i=0}^{k-1} x^{1-t_i} y^{t_i} (-1)^{a_i t_i} = \prod_{i=0}^{k-1} \sum_{t_i=0}^{1} x^{k-t_i} y^{t_i} (-1)^{a_i t_i}.$$

If $a_i = 0$, the inner sum is worth $x + y$, while if $a_i = 1$, we find xy. So we have

$$\forall a \in (\mathbb{F}_2)^k, \quad \hat{f}(\chi_a) = (x+y)^{kw(a)} (xy)^{w(a)}.$$

We can now apply the Poisson formula (2.29). The equality of the theorem is thus true if we consider the values of the polynomials whatever the point $(x,\, y) \in \mathbb{C}^2$. It is therefore also true as a polynomial equality over $\mathbb{Z}[X,\, Y]$. □

This identity, in addition to its certain aesthetic interest, constitutes the main tool for the combinatorial study of corrective codes. The Section 6.4, reformulates the identity of MacWilliams within the framework of code theory, and explains the multiple applications which result from it.

2.3.3 The continuous Poisson formula

The Poisson formula we just used on a finite abelian group actually has a similar statement in the context of continuous functions defined on $\mathbb{R}$. To make this statement pleasant, we will define the continuous Fourier transform as follows:

$$\forall f \in L^1(\mathbb{R}), \forall x \in \mathbb{R}, \quad \hat{f}(x) \overset{\text{def.}}{=} \int_{\mathbb{R}} f(t) e^{-2\imath\pi tx} \mathrm{d}t. \tag{2.30}$$

We can then state the Poisson formula on $\mathbb{R}$. It will be noted with great interest that its proof is largely similar to that made in the framework of finite groups. The only difficulty of the continuous case lies in the problems of convergence of the manipulated series, which obliges us to impose more restrictive assumptions on the functions that we analyze.

Théorem 6 (Continuous Poisson formula). *Let $f \in L^1(\mathbb{R})$ be a continuous function such that*

$$\exists M > 0,\ \exists \alpha > 1, \quad |f(x)| \leq M(1+|x|)^{-\alpha}, \tag{2.31}$$

$$\sum_{n=-\infty}^{+\infty} |\hat{f}(n)| < +\infty. \tag{2.32}$$

Under these assumptions, we have

$$\sum_{n=-\infty}^{+\infty} f(n) = \sum_{n=-\infty}^{+\infty} \hat{f}(n). \tag{2.33}$$

Proof. We start by periodizing the function f by introducing the function

$$\forall x \in \mathbb{R}, \quad f_1(x) \overset{\text{def.}}{=} \sum_{n=-\infty}^{+\infty} f(x+n)$$

If we restrict ourselves to the compact $\{x \in \mathbb{R},\ |x| \leq A\}$, for $A > 0$, the hypothesis (2.31) allows to affirm, for $|n| \geq 2A$,

$$|f(x+n)| \leq M(1+|x+n|)^{-\alpha} \leq M(1+|n|-A)^{-\alpha} \leq M(1+|n|/2)^{-\alpha},$$

which defines the general term of a convergent series. We therefore conclude that on any compact, the series which defines f_1 is normally convergent, so that f_1 is continuous. Moreover, we can easily see that f_1 is 1-periodic, since

$$f_1(x+1) = \sum_{n=-\infty}^{+\infty} f(x+n+1) = \sum_{n'=-\infty}^{+\infty} f(x+n') = f_1(x),$$

where we have made the change of variable $n' = n+1$ (authorized by the absolute convergence of the series). We can therefore calculate its Fourier coefficients:

$$\forall m \in \mathbb{Z}, \quad c_m(f_1) = \int_0^1 f_1(t) e^{-2\imath m\pi t} \mathrm{d}t = \sum_{n\in\mathbb{Z}} \int_0^1 f(t+n) e^{-2\imath m\pi t} \mathrm{d}t,$$

where the inversion between the sum and the integral is justified by the normal convergence of the series of general term $f(t+n)e^{-2\imath m\pi t}$ for $t \in [0,1]$. We can thus continue the calculations to obtain, by changing the variable $u = t+n$,

$$\forall m \in \mathbb{Z}, \quad c_m(f_1) = \sum_{n\in\mathbb{Z}} \int_n^{n+1} f(u) e^{-2\imath m\pi u} \mathrm{d}u = \int_{-\infty}^{+\infty} f(u) e^{-2\imath m\pi u} \mathrm{d}u = \hat{f}(m)$$

(the last equality is justified by Lebesgue's dominated convergence theorem, because $x \mapsto f(x)e^{-2\imath m\pi x} \in L^1(\mathbb{R})$). We therefore note, with the hypothesis (2.32) that the Fourier series associated with f_1 absolutely converges. By additionally using the fact that f_1 is a continuous function, we therefore conclude that it is the sum of its Fourier series. We can therefore write

$$\forall x \in \mathbb{R}, \quad f_1(x) = \sum_{m\in\mathbb{Z}} c_m(f_1) e^{2\imath m\pi x} = \sum_{m\in\mathbb{Z}} \hat{f}(m) e^{2\imath m\pi x}.$$

In the end, we therefore obtain equality

$$\forall x \in \mathbb{R}, \quad \sum_{n\in\mathbb{Z}} f(x+n) = \sum_{m\in\mathbb{Z}} \hat{f}(m) e^{2\imath m\pi x},$$

which gives the desired Poisson formula by making $x = 0$. □

Remark 21. (**Link with Poisson's formula on a finite group**). The continuous Poisson formula that we have just demonstrated is in all points similar to the formula (2.23), valid on a finite abelian group . Indeed, in the continuous case, we must consider the group $G = \mathbb{R}^1$, which is the real line provided with the addition, as well as the discrete subgroup $\mathbb{Z} \subset \mathbb{R}$. The quotient group is nothing other than the circle $\mathbb{R}/\mathbb{Z} \simeq S^1$. Moreover, we will see in Paragraph 4.1.1, that we have a complete description of the characters of the circle, since they correspond to the exponentials $e_n : t \mapsto e^{2\imath n\pi t}$. We therefore have an explicit isomorphism $\hat{S^1} \simeq \mathbb{Z}$. Therefore, we can write the Poisson formula in the continuous case in the form

$$\sum_{n\in\mathbb{Z}} f(n) = \sum_{e_n \in \hat{\mathbb{R}/\mathbb{Z}}} \langle f,\, e_n \rangle.$$

So up to the factor $\frac{|H|}{|G|}$, this formula is in every way similar to (2.23).

One of the many applications of Poisson's formula concerns the function *Theta* of *Jacobi*, which is defined as follows.

Definition 17 (Jacobi Theta Function)**.** *We define the function Theta by*

$$\forall t > 0, \quad \theta(t) \overset{\text{def.}}{=} \sum_{n=-\infty}^{+\infty} e^{-\pi n^2 t}.$$

Before stating the functional equation that θ satisfies, we need to prove a classical lemma on the Fourier transform of a Gaussian.

Lemma 8. *Let g_t, for $t > 0$, be the Gaussian defined by*

$$\forall x \in \mathbb{R}, \quad g_t(x) = e^{-\frac{x^2}{2t}}.$$

So we have

$$\forall x \in \mathbb{R}, \quad \hat{g}_t(x) = \sqrt{2\pi t} e^{-2\pi^2 t x^2}$$

where we have kept the definition of the Fourier transform (2.30).

Proof. The transform of the function g_t is written

$$\hat{g}_t(x) = \int_{\mathbb{R}} e^{-\frac{u^2}{2t}} e^{-2\imath\pi u x} \mathrm{d}u.$$

Using the derivation theorem under the sum sign, we see that the function obtained is C^1, and that we have

$$\frac{\mathrm{d}\hat{g}_t}{\mathrm{d}x}(x) = -2\imath\pi \int_{\mathbb{R}} u e^{-\frac{u^2}{2t}} e^{-2\imath\pi u x} \mathrm{d}u = -4\pi^2 x t \hat{g}_t(x),$$

where the last equality is obtained by integration by parts. By solving the differential equation of which g_t is the solution, we obtain

$$\hat{g}_t(x) = \hat{g}_t(0) e^{-2\pi^2 t x^2}.$$

It only remains to calculate the value of $\hat{g}_t(0) = \sqrt{t} I$, with $I = \int_{\mathbb{R}} e^{-x^2/2} \mathrm{d}x$. To do this, it is advisable to switch to polar coordinates when calculating I^2:

$$I^2 = \int_{\mathbb{R}} \int_{\mathbb{R}} e^{-\frac{x^2+y^2}{2}} \mathrm{d}x \mathrm{d}y = 2\pi \int_0^{+\infty} r e^{-\frac{r^2}{2}} \mathrm{d}r = 2\pi.$$

By putting all these results end to end, we get the announced Fourier transform. $\square$

Here is finally the identity of Jacobi on the function θ.

Théorem 7 (Identity of Jacobi)**.**

$$\forall t > 0, \quad \theta(t) = \frac{1}{\sqrt{t}} \theta\left(\frac{1}{t}\right) \tag{2.34}$$

Proof. It suffices to apply Poisson's formula to the function g_t. It is obvious that g_t satisfies the hypotheses of the theorem 6. We thus obtain the following equality:

$$\sum_{n \in \mathbb{Z}} g_t(n) = \sum_{n \in \mathbb{Z}} e^{-\frac{n^2}{2t}} = \sum_{n \in \mathbb{Z}} \hat{g}_t(n) = \sqrt{2\pi t} \sum_{n \in \mathbb{Z}} e^{-2\pi^2 t n^2}.$$

It is nothing other than the identity that we were trying to prove, evaluated in $2\pi t$. $\square$

One of the interests of this identity is to allow to calculate the function θ for small values of the parameter t, and this with a very high precision. For example, for $t = 0.001$, the right hand side of (2.34) instantly provides the result with a precision greater than that of Matlab in double precision (which is given by the command `eps = 2.2204e-016`), and this with simply the index term $n = 0$ in the sum. If we naively use the left side of the identity, we observe a relatively slow geometric decrease in error.

2.4 Exercises

Exercise 12 (Geometric proof of quadratic reciprocity)**.** *This exercise is taken from an article by Laubenbacher*[42]*. We will detail step by step the proof of quadratic reciprocity that Eisenstein gave in 1844. This proof is quite original since it mainly uses arguments of a geometric nature. In the rest of this exercise, we denote by* $[x]_p$ *the remainder of the Euclidean division of* x *by* p *(which is always an element of* $\{0, \ldots, p-1\}$*, even if* $x \leq 0$*), and* $\lfloor x \rfloor$ *the integer part of* x*. We consider two distinct odd prime numbers* p *and* r*, and we want to prove the quadratic reciprocity formula* (2.18)*.*

1. *We denote by* $A \overset{\text{def.}}{=} \{2, 4, 6, \ldots, p-1\}$ *and* $B \overset{\text{def.}}{=} \{[ra]_p \ : \ a \in A\}$*. Show that we have*
$$A = \left\{ \left[(-1)^b b\right]_p \ : \ b \in B \right\}.$$

2. *Deduce that modulo* p*, we have the following equality:*
$$\prod_{b \in B} b = r^{\frac{p-1}{2}} \prod_{a \in A} a = r^{\frac{p-1}{2}} (-1)^{\sum_{b \in B} b} \prod_{b \in B} b \mod p,$$
since
$$\left(\frac{r}{p}\right) = (-1)^{\sum_{b \in B} b}.$$

3. *Prove the following equality:*
$$\sum_{a \in A} ra = p \sum_{a \in A} \left\lfloor \frac{ra}{p} \right\rfloor + \sum_{b \in B} b.$$
To deduce
$$\left(\frac{r}{p}\right) = (-1)^{\sum_{a \in A} \left\lfloor \frac{ra}{p} \right\rfloor}.$$

4. *In order to give a geometric meaning to this equation, we construct the figure* 2.3*. Show that no point with integer coordinates lies on* $]AB[$*. Show then that the number of even abscissa points in the triangle* ABD *is equal to* $\sum_{a \in A} \left\lfloor \frac{ra}{p} \right\rfloor$*.*

5. *We consider an entire abscissa* $a > \frac{p}{2}$*. Show that, modulo 2, the number of points of abscissa* a *located below* (AB) *(marked* $+$ *in the figure) is equal to the number of points of the same abscissa but located above* (AB) *(marked* $\times$*). Show that this number is also equal to the number of abscissa points* pa *located below* (AB) *(noted* $\bullet$*). Conclude that* $\sum_{a \in A} \left\lfloor \frac{ra}{p} \right\rfloor$ *is equal, modulo 2, to the number of points with integer coordinates in the interior of the triangle* AHK*.*

6. *By swapping the roles of* p *and* r*, then counting the points in the rectangle* $ALHK$*, deduce the quadratic reciprocity law.*

Exercise 13 (Fermat's theorem on a finite field)**.** *This exercise uses the notations and results of the exercise* 5*. Let* $q = p^r$ *where* p *is a prime number. We want to show that if* k *is an integer such that* $q \geq k^4 + 4$*, then Fermat's equation on* $\mathbb{F}_q$
$$x^k + y^k = z^k \qquad x, y, z \in \mathbb{F}_q^* \tag{2.35}$$
has a solution. To do this, we will use the Fourier transform on the group $(\mathbb{F}_q, +)$*.*

1. *Let* $k | q-1$*. Show that there exists a unique subgroup* H_k *of index* k *in* $\mathbb{F}_q^*$ *and that*
$$H_k = \left\{ x^k \ : \ x \in \mathbb{F}_q^* \right\}.$$

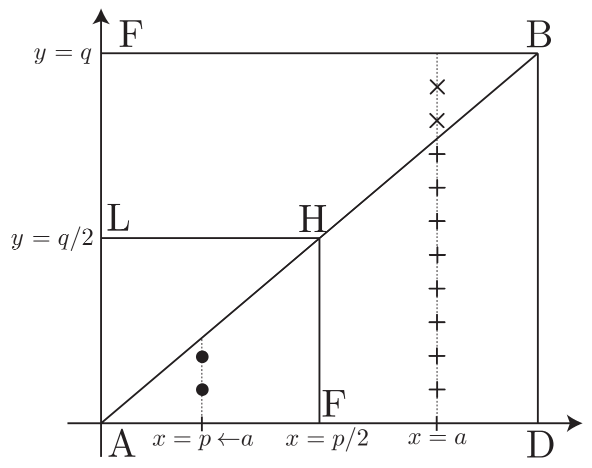

Figure 2.3: Geometric proof of the quadratic reciprocity law

2. *We denote by $\chi_0, \ldots, \chi_{k-1}$ the multiplicative characters of the quotient group $\mathbb{F}_q^*/H_k$. They are canonically extended into multiplicative characters of $\mathbb{F}_q^*$ by composing by the canonical projection. Show then that for any additive character ψ we have*

$$\hat{f_{H_k}}(\psi) = \frac{1}{k}\sum_{i=0}^{k-1} G(\chi_i, \psi).$$

Using the proposition 26, show then that $\Phi(H_k) < \sqrt{q}$, where Φ is defined by the equation (1.17).

3. *Let $A_1, A_2 \subset G$. We denote by N and N' respectively the number of solutions of the equations*

$$x + y = z^k, \qquad \text{with} \quad x \in A_1,\ y \in A_2,\ z \in \mathbb{F}_q^*, \tag{2.36}$$
$$x + y = u, \qquad \text{with} \quad x \in A_1,\ y \in A_2,\ u \in H_k. \tag{2.37}$$

Show that $N = kN'$, then show that

$$\left| N - \frac{|A_1||A_2|(q-1)}{q} \right| < k\sqrt{|A_1||A_2|q}. \tag{2.38}$$

We can start by proving a similar inequality for N' using the result of the exercise 5, question 2.

4. *If we denote by $l_i \overset{\text{def.}}{=} \frac{q-1}{|A_i|}$, then show that if $q \geq k^2 l_1 l_2 + 4$ the equation (2.36) admits a solution. In the case where k does not divide $q - 1$, show that*

$$\left\{x^k \ : \ x \in \mathbb{F}_q^*\right\} = \left\{x^d \ : \ x \in \mathbb{F}_q^*\right\},$$

where $d \overset{\text{def.}}{=} \mathrm{GCD}(q-1, k)$. Deduce that the result is valid for any k satisfying $q \geq k^2 l_1 l_2 + 4$.

5. *Using the sets $A_1 = A_2 = H_k$, show that if $q \geq k^4 + 4$, then the equation (2.35) admits at least one solution .*

Exercise 14 (Walsh transform and statistical study)**.** *This exercise is taken from an article by Rockmore and Malsen[48], which presents a technique known as the analysis of Yates, from the name of the statistician who invented this method. Consider the following situation: a farmer wants to know the influence of three parameters on his wheat production. These parameters are the lighting (represented by the variable a), the* *amount of herbicide (variable b), and the amount of fertilizer (variable c). Each of these variables can take two values: high quantity (noted +) and low quantity (noted −). An experimental report is given in the form of the following table grouping the values for the average size of wheat (in centimeters) under the different conditions:*

a	b	c	α_{abc}
$+$	$+$	$+$	69
$-$	$+$	$+$	81
$+$	$-$	$+$	63
$-$	$-$	$+$	77
$+$	$+$	$-$	61
$-$	$+$	$-$	92
$+$	$-$	$-$	54
$-$	$-$	$-$	89

We can therefore represent these results in the form of a function

$$f : \left\{ \begin{array}{ccc} (\mathbb{Z}/2\mathbb{Z})^3 \simeq \{-, +\}^3 & \longrightarrow & \mathbb{R} \\ (a,\, b,\, c) & \longmapsto & \alpha_{abc} \end{array} \right. .$$

In order to analyze these results, we define the 0-order interaction, denoted μ, which is simply the mean:

$$\mu \overset{\text{def.}}{=} \frac{1}{8} \sum_{(a,\, b,\, c) \in \{+,-\}^3} \alpha_{abc}.$$

We then define the interactions of order 1, denoted μ_a, μ_b and μ_c, as corresponding to the effect of a single parameter, the other two being assumed to be constant. For example we have

$$\mu_a \overset{\text{def.}}{=} \frac{1}{4} \sum_{(b,\, c) \in \{+,-\}^2} \alpha_{+bc} - \frac{1}{4} \sum_{(b,\, c) \in \{+,-\}^2} \alpha_{-bc}.$$

In the same vein, define the 3 interactions of order 2, denoted by μ_{ab}, μ_{bc} and μ_{ac}, as well as the interaction of order 3, denoted by μ_{abc}. How can we calculate all these interactions using the Walsh transform? Deduce a fast calculation algorithm. Do the math for the farmer.

Exercise 15 (Haar wavelet)**.** *We denote ψ_0 the function indicator of $[0,\, 1]$ and ψ the function which is worth 1 on $[0,\, \frac{1}{2}[$, -1 on $[\frac{1}{2},\, 1]$, and 0 everywhere else. We then define a series of functions ψ_n by*

$$\forall j \geq 1,\ \forall k \in \{0, \ldots, 2^{j-1} - 1\}, \quad \psi_n(x) \overset{\text{def.}}{=} \psi_{j,k}(x) \overset{\text{def.}}{=} 2^{\frac{j}{2}} \psi\left(2^j x - k\right),$$

where $n = 2^{j-1} + k$. We denote, for $j \geq 0$, F_j the space of functions of $[0,\, 1]$ in $\mathbb{R}$ constants on each of the intervals $I_k \overset{\text{def.}}{=} [k2^{-j},\, (k+1)2^{-j}[$, for $k \in \{0, \ldots,\, 2^j - 1\}$ (we include the point 1 in the last interval).

1. *Show that $\{\psi_n\}_{n=0}^{2^j - 1}$ forms an orthonormal basis of F_j for the usual scalar product of $L^2([0,\, 1])$.*
2. *Let f be a continuous function of $[0,\, 1]$ in $\mathbb{R}$. For $J \geq 0$, we denote by f_J the projection of f on F_J:*

$$f_J \overset{\text{def.}}{=} \sum_{j=0}^{J} \sum_{k=0}^{2^{j-1}-1} \langle f,\, \psi_{j,k} \rangle \psi_{j,k}.$$

Show that f_J converges uniformly on $[0,\, 1]$ to f when $J \to +\infty$. We then define, for $n \geq 0$, the function

$$\tilde{f}_n \overset{\text{def.}}{=} \sum_{m=0}^{n} \langle f,\, \psi_m \rangle \psi_m.$$

Show that $\tilde{f}_n$ converges uniformly to f when $n \to \infty$. Then show that $\{\psi_n\}_{n \in \mathbb{N}}$ forms a Hilbert basis of $L^2([0,\, 1])$. The figure 2.4 represents the decomposition of a function f on the first vectors of the base of ψ_n, which we call the Haar base.

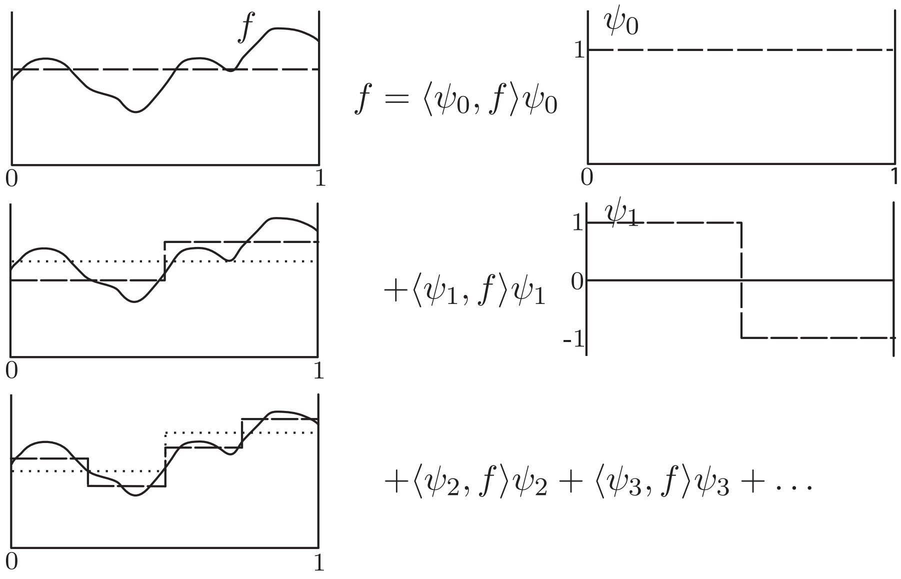
f
$f = \langle \psi_0, f\rangle \psi_0$
ψ_0
1
0
1
0
1
ψ_1
$+\langle \psi_1, f\rangle \psi_1$
1
0
-1
0
1
$+\langle \psi_2, f\rangle \psi_2 + \langle \psi_3, f\rangle \psi_3 + \dots$
0
1

Figure 2.4: Decomposition based on Haar

3. *We introduce, for $j \geq 0$, the functions "stepped" $\varphi_{j,k}(x) \overset{\text{def.}}{=} 2^{\frac{j}{2}} \psi_0\left(2^j x - k\right)$. Show that $\varphi_{j,k}$, for $k \in \{0, \dots, 2^j - 1\}$, forms an orthonormal basis of F_j. We define G_{j-1} the vector space such that $F_j = F_{j-1} \oplus G_{j-1}$ (space of "details"). Show that $\{\psi_{j,k}\}_{k=0}^{2^{j-1}-1}$ is an orthonormal basis of G_{j-1}. Then express the function $\psi_{j,k}$ as a linear combination of $\varphi_{j+1,s}$, $s \in \{0, \dots, 2^j - 1$.*

4. *Let $f \in F_j$. We denote by $x^{(0)} \in \mathbb{R}^{2^j}$ the vector of scalar products $x^{(0)}[k] = \langle f, \varphi_{j,k}\rangle$. How do you calculate them from f? For $i \in \{1, \dots, j\}$, we define vectors $x^{(i)}$ and $d^{(i)}$ of size 2^{ji} , by the relations, for $k \in \{0, \dots, 2^{ji} - 1\}$,*

$$x^{(i)}[k] \overset{\text{def.}}{=} \frac{x^{(i-1)}[2k+1] + x^{(i-1)}[2k]}{\sqrt{2}}$$
$$d^{(i)}[k] \overset{\text{def.}}{=} \frac{x^{(i-1)}[2k+1] - x^{(i-1)}[2k]}{\sqrt{2}}.$$

Intuitively, $x^{(i)}$ represents the trend in the signal $x^{(i-1)}$, and $d^{(i)}$ represents the details. The figure 2.5 symbolizes the succession of calculations to be performed to obtain the vectors $d^{(i)}$ as well as the last coefficient $x^{(j)}[0]$. Show that the operator $x^{(i)} \mapsto (x^{(i+1)}, d^{(i+1)})$ can be seen as an isometry (more precisely a rotation) of $\mathbb{R}^{ji}$. Show that the $x^{(i)}$ all have the same average, and therefore that $x^{(j)}[0]$ represents the average of the original $x^{(0)}$ signal .

$$\begin{tikzcd}
x = x^{(0)} \arrow[r] \arrow[dr] & x^{(1)} \arrow[r] \arrow[dr] & x^{(2)} \arrow[r] & \cdots \arrow[r] \arrow[dr] & x^{(j)} \\
 & d^{(1)} & d^{(2)} & & d^{(j)}
\end{tikzcd}$$

Figure 2.5: Cascade calculation of the decomposition coefficients

5. *Show that we have, for $i \in \{1, \dots, j\}$ and for $k \in \{0, \dots, 2^{ji} - 1\}$,*

$$d^{(i)}[k] = \langle f, \psi_{j-i+1,k}\rangle \qquad \textit{and} \qquad x^{(j)}[0] = \langle f, \psi_0\rangle.$$

Thanks to this algorithm, what is the number of operations necessary to decompose a function of F_j on the basis of ψ_n?

6. *We assume that $n = 2^j$. Show that the operator Γ which to $x \in \mathbb{R}^n$ associates the vector $(d^{(1)}, x^{(2)}, \dots, d^{(j)}, x^{(j)})$ is an isometry of $\mathbb{R}^n$ for the canonical dot product (we put the vectors $d^{(i)}, x^{(i)}, \dots$). Deduce that the application of this operator corresponds to the decomposition of x in an orthonormal basis of $\mathbb{R}^n$ that we will specify. Compare this database to the Walsh database described in paragraph 2.2.1. Compare in terms of complexity the algorithm of decomposition in the base of Haar and that of decomposition in the base of Walsh (FWT algorithm, paragraph 2.2.2).*

Figure 2.6 shows two examples of transforms $y = \Gamma x$. We can see that since the signals are regular, only the coefficients corresponding to the coarse scales (that is to say for j small, the indices on the right of the transform) are large. The Haar base is the simplest example of a wavelet base. The fast wavelet transforma-

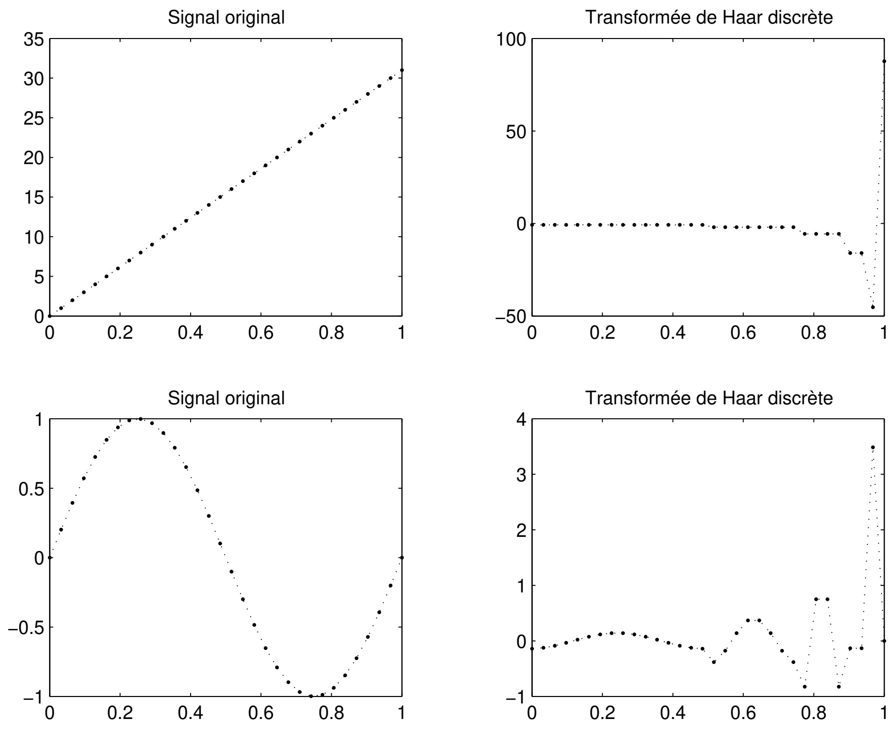

Figure 2.6: Examples of discrete Haar transforms

tion algorithm was introduced by Mallat,[47]. The differences between the Walsh and Haar bases illustrate the transition from the Fourier transform (here on $(\mathbb{F}_2)^k$) to the wavelet transform. The exercise 79 presents the construction of wavelets on finite fields.

Exercise 16 (Image compression). *The goal of this exercise is to properly order Walsh functions to successfully compress 1D and 2D signals.*

1. *Generalize the discrete Walsh transform to the two-dimensional case. We will use the functions*

$$\chi_{i,j}(s, t) = \chi_i(s)\chi_j(t).$$

Write a fast computation algorithm for the 2D Walsh transform.

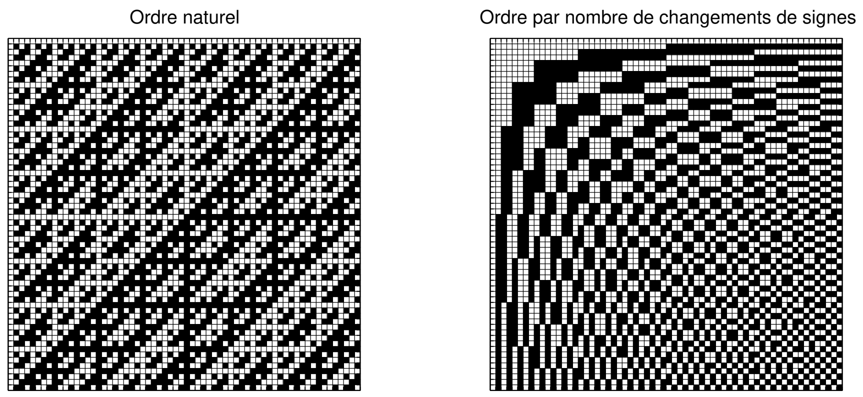

Figure 2.7: Two ways to classify Walsh functions

2. *Show that we can classify the discrete Walsh functions (defined by the equation (2.19)) in increasing order of the number of sign changes. The figure 2.7 shows the Walsh matrices obtained by classifying the functions in the usual order and in the order of the number of sign changes.*

3. *Intuitively, such a classification makes it possible to order the Walsh spectrum from tendencies (low frequencies) to details (high frequencies). Calculate for some functions the spectra obtained with the two classifications, and verify this interpretation. The figure 2.8 shows the spectra of the function represented at the top left of the figure 2.9.*

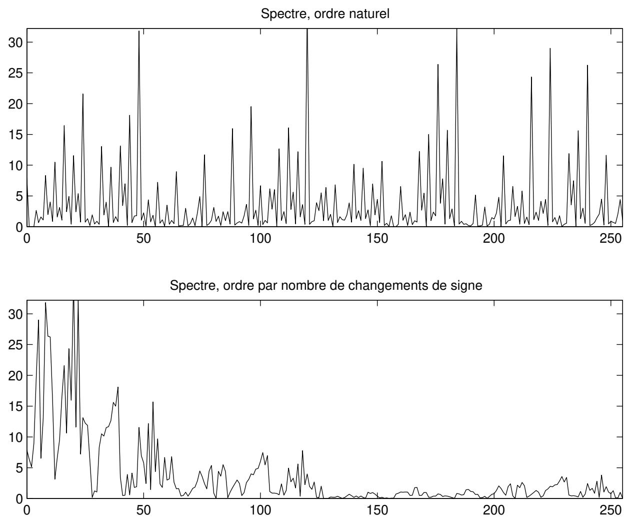

Figure 2.8: Walsh spectrum of a 1D signal

4. *We have therefore classified the Walsh functions according to an order $\chi_{i_0}, \ldots, \chi_{i_N}$. We consider a signal $f \in \mathbb{C}^N$. For $0 \leq n < N$, we construct the function*

$$f_n \overset{\text{def.}}{=} \sum_{k=0}^{n} \langle f,\, \chi_{i_k} \rangle \chi_{i_k}.$$

Explain why this process allows to compress the signal f. Figure 2.9 shows the progressive compression of a signal. The percentage of Walsh coefficients which have been retained is indicated each time. After studying the discrete Fourier transform in chapter 3, we can perform the same process, but with the Fourier spectrum. What are the advantages and disadvantages of each method (calculation time, quality of reconstruction, etc.)?

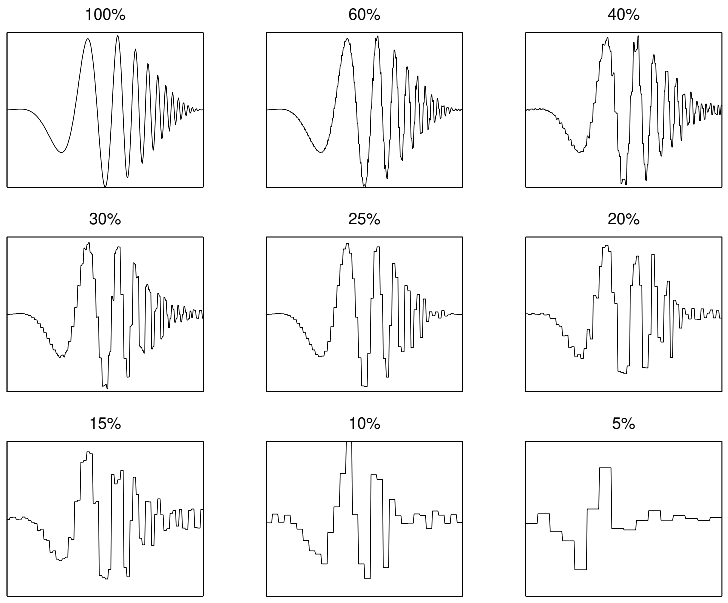

Figure 2.9: Compression of a 1D signal

5. *What classification (s) can we adopt for the functions of Walsh 2D? The figure 2.10 proposes such a classification (from left to right and top to bottom). Apply this classification to compress 2D images. Write a Matlab program to perform this compression. The figure 2.11 shows the progressive compression of an image representing the letter A.*

Exercise 17 (Hadamard matrices)**.** *This exercise makes the connection between the Walsh matrices considered in paragraph 2.2.1 and the quadratic residuals introduced at the beginning of this chapter. A matrix H_n, of size $n \times n$, whose inputs are $+1$ or -1, is called a Hadamard matrix if it satisfies*

$$H_n H_n^\top = n \,\mathrm{Id}_n,$$

where we denote by $H_n^\top$ the transposed matrix of H_n.

1. *Explain why the matrix W_n, defined in the proposition 29, for $n = 2^k$, is a Hadamard matrix.*

2. *Show that if there exists a Hadamard matrix H_n of size $n \times n$, then n is worth 1, or 2, or is a multiple of 4. We can start by showing that we can assume that H_n is normalized, that is to say with 1 on the first row and the first column. Then we will show that if $n \geq 3$, we can assume that the first three lines of H_n are written in the form*

$$\underbrace{\begin{matrix}1&\dots&1\\1&\dots&1\\1&\dots&1\end{matrix}}_{i}\;\;\underbrace{\begin{matrix}1&\dots&1\\1&\dots&1\\-1&\dots&-1\end{matrix}}_{j}\;\;\underbrace{\begin{matrix}1&\dots&1\\-1&\dots&-1\\1&\dots&1\end{matrix}}_{k}\;\;\underbrace{\begin{matrix}1&\dots&1\\-1&\dots&-1\\-1&\dots&-1\end{matrix}}_{l}$$

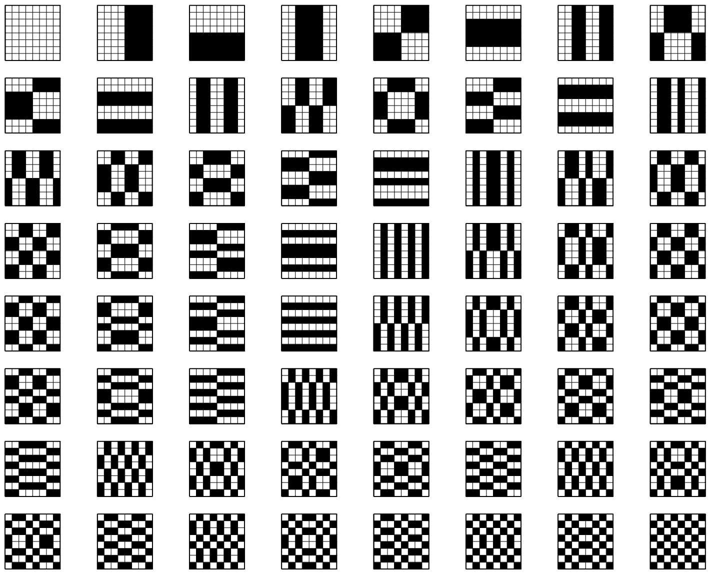

Figure 2.10: Classification of 2D Walsh functions

where the integers i, j, k, and l denote the lengths of each portion (they can optionally be zero). Finally, we will show that we actually have $i = j = k = l$.

3. *The inverse problem, namely the construction of a matrix H_n for a n multiple of 4 given, is very complex. In fact, it is speculated that it is still possible to do so, although it has not yet been proven. Assume that $n = p + 1$, where p is an odd prime number. We also suppose that n is a multiple of 4, and we will show that we can then construct a matrix H_n. We will use the quadratic residue character modulo p, denoted by η, which is defined in the equation (2.2). We define a matrix Q of size $p \times p$ by*

$$Q \overset{\text{def.}}{=} \{\eta(ji)\}_{0 \le i,\, j \le p-1} .$$

Show that Q is anti-symmetric, and that we have $QQ^\top = p \operatorname{Id}_p - J$, where J is the matrix whose all inputs are worth 1. Also show that we have $QJ = JQ = 0$. Such a matrix is called Paley matrix.

4. *We now define the matrix H_n of size $n \times n$ by*

$$H_n \overset{\text{def.}}{=} \begin{pmatrix} 1 & v \\ v^\top & Q - \operatorname{Id}_p \end{pmatrix},$$

where we noted $v = (1, \ldots, 1) \in \mathbb{R}^p$. Show that H_n is a Hadamard matrix. Here is an example for $p = 7$:

$$H_8 \overset{\text{def.}}{=} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & -1 & 1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 & -1 & -1 & -1 & 1 \\ 1 & 1 & 1 & -1 & 1 & -1 & -1 & -1 \end{pmatrix}.$$

5. *Let A be a matrix of size $n \times n$ such that its inputs a_{ij} verify*

$$\forall (i,\, j) \in \{1, \ldots, n\}^2, \quad |a_{ij}| \le 1.$$

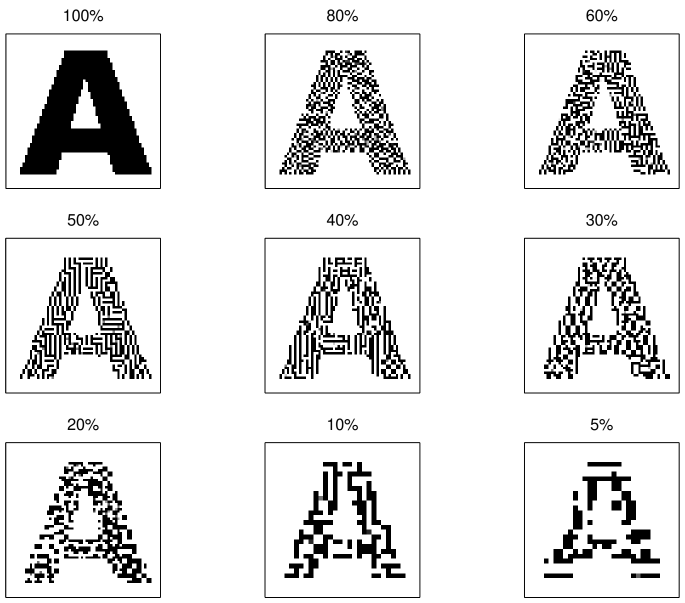

Figure 2.11: Compress a 2D image

Show the Hadamard inequality:

$$|\det(A)| \leq n^{\frac{n}{2}}. \tag{2.39}$$

Show that if there is a Hadamard matrix, then the latter reaches this bound.

The geometric interpretation of the bound (2.39) is very simple. It is a matter of considering a system of n vectors (the columns of the matrix) inside the cube $|x_i| \leq 1$, (where we denote by $\{x_i\}_{i=1}^n$ a coordinate system), enclosing a rectangular parallelepiped of maximum volume. In the case of Hadamard matrices, these vectors are large diagonals of the cube, and therefore have maximum lengths. In addition, they are orthogonal, so as to produce the maximum volume. In dimensions where Hadamard matrices do not exist, it is not possible to produce orthogonal diagonals, even if one thinks that the vectors which minimize (2.39) are close to the large diagonals . This remains an open problem.
The exercise 66 presents an application of Hadamard matrices for the construction of bi-orthogonal corrective codes.

Exercise 18 (Matrix tensor product)**.** *Let A be a square matrix of size s and B a square matrix of size t. We define the tensor product $A \otimes B$ as the matrix of size $s \times t$*

$$A \otimes B \overset{\text{def.}}{=} \begin{pmatrix} a_{11}B & \cdots & a_{1s}B \\ \vdots & & \vdots \\ a_{s1}B & \cdots & a_{ss}B \end{pmatrix}.$$

1. *We assume that A matches $AA^* = s\,\mathrm{Id}_s$. Show that $A^{\otimes n} = A \otimes \cdots \otimes A$ (n products) satisfies $A^{\otimes n}(A^{\otimes n})^* = s^n\,\mathrm{Id}_{s^n}$.*

2. *What is the connection with the Walsh transform?*

3. *Taking inspiration from the fast FWT algorithm, write a fast algorithm that computes the transform $y = A^{\otimes n}x$. How is the inverse transform calculated?*

4. *We take as base matrix*
$$A_\alpha \overset{\text{def.}}{=} \sqrt{2}\begin{pmatrix}\cos(\alpha) & \sin(\alpha)\\ \sin(\alpha) & -\cos(\alpha)\end{pmatrix}.$$
How can the transform $x \mapsto A_\alpha^{\otimes n} x$ be seen as an intermediate Walsh transform?

The figure 2.12 shows the transforms of a "triangle" function for values of α in $[0, \pi/2]$. The ordinary Walsh transform corresponds to the 6th curve. For $\alpha = \pi/2$, we find the original symmetrized signal. We can look

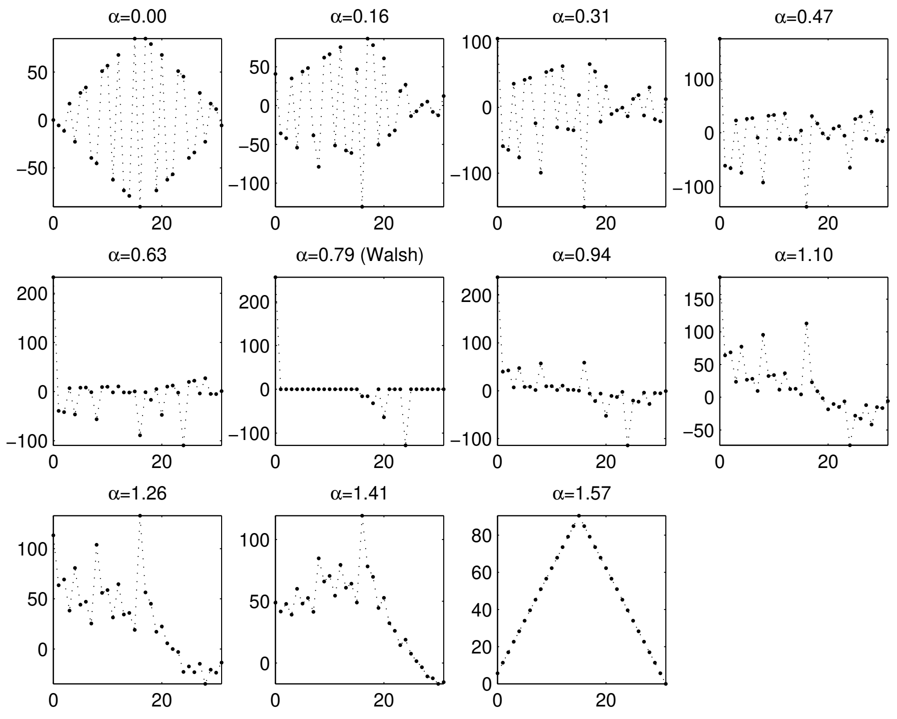

Figure 2.12: Intermediate Walsh transform

at the exercise 86, which uses the theory of linear representations to build a matrix A of size 8.

Exercise 19 (Generalization of the identity of MacWilliams)**.** *In this exercise, we propose to extend the identity of MacWilliams to the case of the vector space $E = \mathbb{F}_q^k$.*

1. *We define the following bilinear form on $E \times E$, with value in $\mathbb{F}_q$:*
$$\forall (a,\, x) \in E^2, \quad \langle a,\, x\rangle \overset{\text{def.}}{=} \sum_{i=0}^{p-1} a_i x_i.$$
Explain how it represents the bilinear form of the duality (bracket of the duality) between the space E and its dual E^* (correctly identified as E).

2. *We denote by χ_1 the canonical additive character of $\mathbb{F}_q$, as defined by the equation (11). Let $a = \{a_0, \ldots, a_{k-1}\} \in (\mathbb{Z}/q\mathbb{Z})^k$. We define*
$$\chi_a : \left\{ \begin{array}{ccc} E & \longrightarrow & \mathbb{C}^* \\ x & \longmapsto & \chi_1(\langle a,\, x\rangle) \end{array} \right. .$$
Explain why the applications χ_a allow to define an isomorphism between E and its dual as an additive group, $\hat{E}$.

3. *Let H be a subgroup of E. Show that it is also a vector subspace. Deduce from the previous question that we can identify the orthogonal of H for the group structure, noted $H^\sharp$, and that for the vector space structure, noted $H^\perp$.*

4. *Demonstrate the identity of MacWilliams within the E space:*

$$A_{H^\perp}(X, Y) = \frac{1}{|H|} A_H(X + (q-1)Y, XY).$$

Exercise 20 (Poisson formula and distributions)**.** *This exercise requires some knowledge of distribution theory, in particular the definition of the Fourier transform of a distribution. Here we take as convention the transform defined in the equation (4.1), which differs by a factor of 2π from that used in (2.30). We denote by Π_s the Dirac comb of steps s, that is to say*

$$\Pi_s \overset{\text{def.}}{=} \sum_{k\in\mathbb{Z}} \delta_{ks}, \tag{2.40}$$

where δ_t is the distribution defined by $\langle \delta_t, \varphi\rangle \overset{\text{def.}}{=} \varphi(t)$ for $\varphi \in \mathcal{C}_0^\infty(\mathbb{R})$ (functions of class $\mathcal{C}^\infty$ with compact support). Show that the Poisson formula (2.33) implies the following equality:

$$\hat{\Pi}_s = \frac{2\pi}{s} \Pi_{\frac{2\pi}{s}}.$$

Exercise 21 (Shannon sampling)**.** *Let $T > 0$. We notice*

$$I_T \overset{\text{def.}}{=} \left[-\frac{\pi}{T}, \frac{\pi}{T}\right] \quad et \quad E_T \overset{\text{def.}}{=} \left\{ f \in L^2(\mathbb{R}) \ : \ \text{Supp}(\hat{f}) \subset I_T \right\}.$$

Let $f \in E_T$. We want to prove Shannon's sampling theorem, which says that f can be reconstructed (interpolated) from the samples $f(nT)$, for $n \in \mathbb{Z}$. More precisely, if we note

$$\text{sinc}_T(t) \overset{\text{def.}}{=} \frac{\sin\left(\frac{\pi t}{T}\right)}{\frac{\pi t}{T}}, \tag{2.41}$$

so we want to show that

$$f(t) \overset{\text{def.}}{=} \sum_{n\in\mathbb{Z}} f(nT)\, \text{sinc}_T(tnT).$$

1. *Show that f is of class $\mathcal{C}^\infty$.*

2. *We denote by f_d the distribution which corresponds to the sampling of f:*

$$f_d \overset{\text{def.}}{=} \sum_{n\in\mathbb{Z}} f(nT)\delta_{nT}.$$

   Using the equality (2.40), show that we have

$$|\omega| \leq \frac{\pi}{T} \Longrightarrow \hat{f}_d(\omega) = \frac{1}{T}\hat{f}(\omega).$$

3. *Calculate the inverse Fourier transform of the indicator function of the interval I_T. Deduce the sampling theorem.*

4. *Show that the family $\{t \mapsto \text{sinc}_T(tnT)\}_{n\in\mathbb{Z}}$ forms an orthogonal base (base de Hilbert) of the space E_T. How is the projection of a function $f \in L^2(\mathbb{R})$ calculated on this space?*

Chapter 3

Discrete Fourier transform

The use of the discrete Fourier transform is at the base of almost all digital digital algorithms. The discovery of the fast transformation algorithm *FFT* (for ***F**ast **F**ourier **T**ransform*, in French *Transformée de Fourier Rapide*) revolutionized the world of signal processing by allowing digital calculations in reasonable time. It was largely this discovery that made it clear that we could work as quickly in the digital world (made up of discrete signals) as in the analog world (made up of continuous signals). Further details on the history of this discovery, and its consequences, can be found in the article by Rockmore [55]. More than a simple particular case of the Fourier transform on a finite group, the discrete Fourier transform has its own language and especially efficient algorithms that are much less obvious than the clear formulas of the previous chapter. This chapter aims in a way to take a tour of the owner; it shows in any case that the multitude of existing FFT algorithms is impressive. But the most important thing, beyond a total understanding of the different variations of the algorithm, is to perceive the strategy of the algorithm, in order to be able to decide, if necessary, which implementation to use.

The FFT algorithm in temporal decimation version is relatively well described (and especially well implemented) in the Numerical Recipes [53]. Regarding the frequency decimation version as well as many improvements, we refer to the book by Brigham [11]. For implementation details in C language, we can look at the book by Arnt [2].

3.1 The language of signal processing

In this paragraph, we will translate the algebraic properties of the Fourier transform into the language of discrete signal theory. First, we will restrict ourselves to a one-dimensional study (to present the algorithms and some applications), then we will make the link between the Fourier transform on an abelian product group and the discrete Fourier transform in dimension two and more.

To fix the ideas, we will consider time signals with complex values. These correspond to functions $\tilde{f} : t \in \mathbb{R} \to \tilde{f}(t) \in \mathbb{C}$. To process this signal digitally, we will only consider a finite number of signal values, and work on these values. We will therefore name the sample of size N of the original signal $\tilde{f}$ the vector $f \overset{\text{def.}}{=} \{f[n]\}_{n=0}^{N-1}$, where the we have denoted $f[n] \overset{\text{def.}}{=} \tilde{f}(t_n)$ the value of the signal f at the instant t_n. The notation in braces is meant to remind that we consider our vectors as samples of a (continuous) signal, but it will happen that we consider these elements as simple vectors of $\mathbb{C}^N$. So that the following analysis is not biased (particularly when reconciling with the continuous transform in Section 4.1), the values of $\{t_n\}_{n=0}^{N-1}$ are assumed to be evenly spaced in an interval $[a, b]$, i.e. $t_n = a + \frac{ba}{N}n$.

Definition 18 (Discrete Fourier transform)**.** *We define the Discrete Fourier transform (abbreviated TFD) of the sample* $f = \{f[n]\}_{n=0}^{N-1}$ *as being the vector* $\hat{f} = \{\hat{f}[k]\}_{k=0}^{N-1} \in \mathbb{C}^N$ *with*

$$\hat{f}[k] \overset{\text{def.}}{=} \sum_{n=0}^{N-1} f[n]\omega_N^{-nk} \qquad fork = 0, \ldots, N-1, \tag{3.1}$$

where we denote $\omega_N = e^{\frac{2\imath\pi}{N}}$ *a primitive* $N^{ième}$ *root of the unit.*
We will also denote $\mathcal{F}(f) \overset{\text{def.}}{=} \hat{f}$*, which allows to define*

$$\mathcal{F} : \left\{ \begin{array}{ccc} \mathbb{C}^N & \longrightarrow & \mathbb{C}^N \\ f & \longmapsto & \mathcal{F}(f) = \hat{f} \end{array} \right. .$$

This notation can lead to confusion with the Fourier transform on a finite group defined by the equation (1.10), however, the great similarity between the two applications (all this is justified a bit below) makes it convenient to use the same notation.

Remark 22. We could have chosen another primitive root of the unit instead of ω_N. This amounts to choosing another generator for the starting group $\mathbb{Z}/N\mathbb{Z}$, and therefore to numbering the elements of f in a different order.

Remark 23. (**Link with the Fourier transform on a finite group**). We have already seen in Section 1.1.2, that the characters $(\chi_k)_{k=0}^{N-1}$ on the cyclic group $\mathbb{Z}/N\mathbb{Z}$ can be defined by

$$\forall s \in \mathbb{Z}/N\mathbb{Z}, \quad \chi_k(s) \overset{\text{def.}}{=} \omega_N^{-ks}. \tag{3.2}$$

We notice that our sample $f \in \mathbb{C}^N$ allows to define a function $f_1 : \mathbb{Z}/N\mathbb{Z} \to \mathbb{C}$, and vice versa. We can make the link between discrete Fourier transform and characters:

$$\hat{f}[k] = \hat{f}_1(\chi_k).$$

We can therefore rewrite the Fourier inversion formula of the proposition 13, in terms of a discrete Fourier transform.

Proposition 31 (Inverse Fourier transform). *We have the following inversion formula:*

$$\forall n = 0, \dots, N-1, \quad f[n] = \frac{1}{N}\sum_{k=0}^{N-1} \hat{f}[k]\omega_N^{nk}. \tag{3.3}$$

Corollary 4. $\mathcal{F} : \mathbb{C}^N \to \mathbb{C}^N$ *is an isomorphism of vector spaces.*

Let us also recall Plancherel's formula.

Proposition 32 (Plancherel formula). *Let* f *and* g *be two samples of size* N*. We have the following formula:*

$$\sum_{i=0}^{N} f[i]\overline{g[i]} = \frac{1}{N}\sum_{i=0}^{N} \hat{f}[i]\overline{\hat{g}[i]}.$$

3.2 Fast Fourier transform

This paragraph is aimed directly at the IT applications of the TFD. It does not require knowledge of group theory. The connections between the FFT algorithm and algebra are discussed in some exercises, for example when studying the *Good-Thomas* 23 method. In parallel with the reading of this chapter, it is of course necessary to have aœ on the algorithms referenced in Paragraph B.3, to make the link between concrete implementation and mathematical formulas.

3.2.1 Presentation of the algorithm

For a signal f of which we know a sample $\{f[n]\}_{n=0}^{N-1}$, the direct calculation of the N coefficients of the discrete Fourier transform

$$\hat{f}[k] \overset{\text{def.}}{=} \sum_{n=0}^{N-1} f[n]e^{-kn\frac{2\imath\pi}{N}} \qquad \text{for} k = 0, \dots, N-1 \tag{3.4}$$

requires $2N^2$ operations (complex addition and multiplication). The *FFT* algorithm allows, by reordering the computations in a dichotomous way, to considerably reduce the computation time by bringing it back to an order of $O(N\log(N))$. Throughout this chapter, we are going to present different versions of the FFT algorithm, starting with the original version, and arguably the simplest, the Cooley and Tukey algorithm. However, we will see that this algorithm has been declined in an almost infinite number of more learned versions than the others, to adapt to different conditions (length of vectors mainly), and to obtain the result always more quickly. Behind a seemingly very simple transformation, TFD, therefore hides a multitude of ideas of a combinatorial and algebraic nature.

Before embarking on a perilous description of the algorithm, let us note a reassuring fact: we will be able to easily reinvest our algorithm to calculate the inverse transform, as specified in the following remark.

Remark 24. (**Inverse transform**). We notice that the inverse transform formula (3.3) can be obtained by replacing ω_N by ω_N^{-1} in the calculation algorithm, then dividing the result by N. Consequently, we can also calculate the inverse discrete Fourier transform in time $O(N\log(N))$, obviously modifying the algorithm used. More synthetically, by considering the sample $\{f_1[n]\}_{n=0}^{N-1}$ defined by

$$\forall n \in \{1,\dots,N-1\},\quad f_1[n] = \frac{1}{N}f[Nn] \qquad f_1[0] \overset{\text{def.}}{=} \frac{1}{N}f[0],$$

we have a writing of the inverse Fourier transform of f in terms of a direct Fourier transform:

$$\mathcal{F}^{-1}(f) = \mathcal{F}(f_1)$$

The algorithm we are about to describe was discovered by Cooley and Tukey in 1965. It allows, when we have a “good” decomposition of the integer N, to calculate the discrete Fourier transform very quickly. We will see in the remainder of the presentation of other algorithms which make it possible to exploit certain less optimal decompositions of N. However, in this first approach of the FFT algorithm, we will assume that $N = 2^p$. This very simple factorization of N will make it possible to use the famous “philosophy” *divide and reign*, by performing a dichotomous progression in the calculation of the TFD. To implement this dichotomy, let us group the terms of the sum of a DFT according to the parity of the indices.
We obtain then, for $k \in \{0,\dots,N-1\}$,

$$\begin{aligned}\hat{f}[k] &= \sum_{n=0}^{N/2-1} f[2n]e^{-2\imath\pi k(2n)/N} + \sum_{n=0}^{N/2-1} f[2n+1]e^{-2\imath\pi k(2n+1)/N} &(3.5)\\ &= \sum_{n=0}^{N/2-1} f[2n]e^{-2\imath\pi kn/(N/2)} + \omega_N^{-k}\sum_{n=0}^{N/2-1} f[2n+1]e^{-2\imath\pi kn/(N/2)}, &(3.6)\end{aligned}$$

where we noted $\omega_N = e^{2\imath\pi/N}$. So if we write

$$f^0 \overset{\text{def.}}{=} \{f[0],\, f[2],\dots,\, f[N-2]\} \tag{3.7}$$

$$f^1 \overset{\text{def.}}{=} \{f[1],\, f[3],\dots,\, f[N-1]\} \tag{3.8}$$

the vectors of even (resp. odd) indices formed from f, we notice that for the $N/2$ first indices $k \in \{0,\,1,\dots,\,N/2-1\}$, the equation (3.6) is written as the sum of two discrete Fourier transforms:

$$\hat{f}[k] = \hat{f}^0[k] + \omega_N^{-k}\hat{f}^1[k]. \tag{3.9}$$

For the indices $k \in \{N/2,\dots,\,N-1\}$, if we write $k' = kN/2$, using the fact that the vectors $\hat{f}^0$ and $\hat{f}^1$ represent samples of period $N/2$, and that $\omega_N^k = -\omega_N^{k'}$ this time we get the difference of two transforms from Fourier:

$$\hat{f}[k] = \hat{f}^0[k'] - \omega_N^{-k'}\hat{f}^1[k']. \tag{3.10}$$

Definition 19 (Some notations)**.** *To summarize all this in a more algorithmic form, let us note*

$$\hat{f}_g \overset{\text{def.}}{=} \left\{\hat{f}[0],\, \hat{f}[1], \ldots,\, \hat{f}[N/2-1]\right\} \tag{3.11}$$

$$\hat{f}_d \overset{\text{def.}}{=} \left\{\hat{f}[N/2],\, \hat{f}[N/2+1], \ldots,\, \hat{f}[N-1]\right\}. \tag{3.12}$$

These are the right and left parts of the transformed vector $\hat{f} = \mathcal{F}(f)$. We will also define the operator $\mathcal{S}_N^x$, for $x \in \mathbb{R}$, which takes a vector $a = \{a_0, \ldots, a_{N-1}\} \in \mathbb{C}^N$ of length N and returns

$$\mathcal{S}_N^x a \overset{\text{def.}}{=} \left\{a_j e^{-xj\frac{2\imath\pi}{N}}\right\}_{j=0}^{N-1} = \left\{a_j \omega_N^{-xj}\right\}_{j=0}^{N-1} \in \mathbb{C}^N. \tag{3.13}$$

We then have the very simple expression of the recurrence that we will use to implement the FFT algorithm:

$$\hat{f}_g = \hat{f}^0 + \mathcal{S}_{N/2}^{1/2} \hat{f}^1 \tag{3.14}$$

$$\hat{f}_d = \hat{f}^0 - \mathcal{S}_{N/2}^{1/2} \hat{f}^1. \tag{3.15}$$

The equations (3.14) and (3.15), also called equations of *Danielson-Lanczos*, express the fact that the Discrete Fourier transform of a signal of length N can be calculated as a function of two signals of length $N/2$, here noted f^0 and f^1. We call this approach *decimation in time* (in English *Decimation In Time*, or *DIT*), as opposed to another approach, the *frequency decimation*, which will be described quickly in paragraph 3.2.6. It is the temporal decimation that will be developed (and optimized) in the next paragraph, but first of all, let's start by presenting a naïve implementation.

Remark 25. (**The butterfly effect**). The operation consisting in mixing two inputs of the even and odd parts of a vector by following the equations (3.14) and (3.15) is called *butterfly scheme* (in English *butterfly scheme*). The figure 3.1 schematically shows the operations performed. It also gives an idea of the cabling to be carried out to perform such an operation directly on a card dedicated to signal processing. Indeed, as shown in figure 3.2, an iteration in the FFT algorithm (here for an input of size 8) is only a succession of butterfly diagrams carried out in cascade .

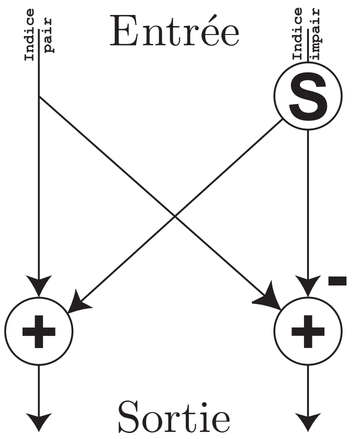

Figure 3.1: Elementary butterfly diagram

The simplest way to implementœ the equations (3.14) and (3.15) is to use a recursive procedure. It is a known fact that a recursive procedure can be written, by means of loops, in a non-recursive fashion (but this process can sometimes be dangerous). We will see in the next paragraph 3.2.5 that the FFT algorithm has

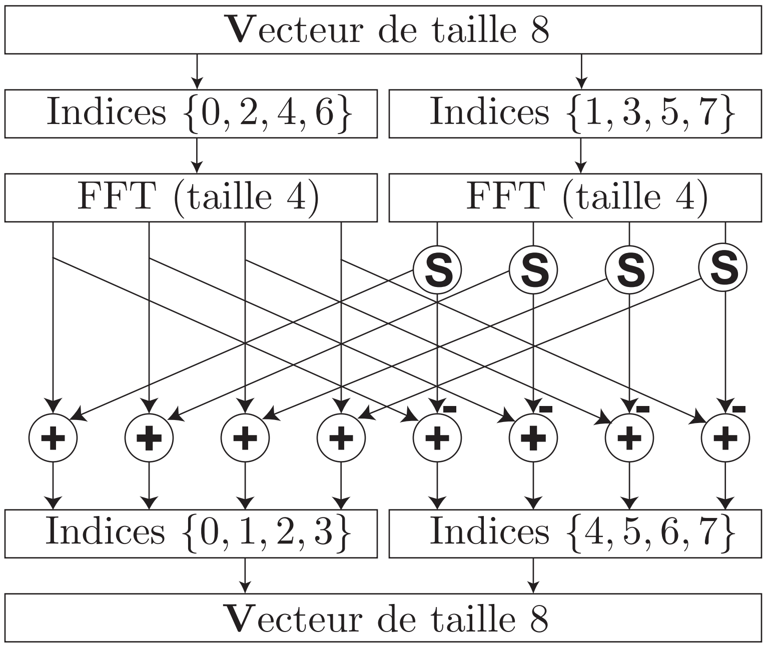

Figure 3.2: An iteration of the FFT algorithm

a lot to gain from being written non-recursively, and not only because of saving time. But for educational purposes, and in order to present some optimizations that can be done on the FFT implementation, we will focus on the recursive implementation written in the B.3 section.

The procedure `fft _rec` therefore takes as input an integer `dir` which is worth $+1$ or -1 depending on whether the Fourier transform is direct or inverse. To simplify the understanding of the code, a procedure `operator _s` has been written to realize the operator $\mathcal{S}_N^x$: it takes as input a vector as well as a real number x (which depends on the sign of the transform).

3.2.2 Cost analysis

Using the recurrence equations (3.14) and (3.15), we easily calculate the cost of the algorithm.

Proposition 33 (Complexity of the FFT)**.** *If we denote by $C(N)$ the cost of the FFT algorithm for an input of length N, then $C(N)$ satisfies the functional equation*

$$C(N) = 2C(N/2) + KN,$$

where K is some constant. In the end, we arrive at the expression $C(N) = KN\log_2(N)$.

Proof. Computing $\hat{f}$ requires computing $\hat{f}^1$ and $\hat{f}^2$ (i.e. $2C(N/2)$ operations), then mixing the two transforms by the butterfly diagram (i.e. KN operations). To reduce to a linear recurrence, it suffices to set $P = \log_2(N)$, and $C'(N) = \frac{C(N)}{N}$ satisfies the functional equation $C'(P) = C'(P-1) + K$. As $C'(0) = 0$, we deduce $C'(P) = KP$, which allows us to conclude. □

The FFT algorithm may seem a bit magical, the fact remains that its discovery made it possible to make many calculations possible by reducing the cost of calculating N Fourier coefficients by $O(N^2)$ for a naive approach to $O(N\log(N))$. Its fairly recent discovery (in the mid-1960s) was a mini revolution: a calculation which until then required two weeks on a computer of the time was suddenly achievable in barely thirty seconds [1].

[1]Source: [53], for N of the order of 10^6

3.2.3 Variations around the algorithm

Before describing a more efficient implementation of the FFT algorithm, let's make some additional remarks, which provide many variations around the proposed recursive implementation.

Remark 26. (**Length of entries**). In the case where the length N of the data of a sample $\{f[n]\}_{n=0}^{N-1}$ is not a power of two, we can make a calculation approximated by completing the sample with a sequence of zeros, to obtain a sample $\{f_1[n]\}_{n=0}^{M-1}$, with $M = 2^p$. Of course, we no longer calculate exactly the same transform, but in the case of an approximate calculation (calculation of continuous transforms, as explained in Section 4.1), this amounts to calculating the transform at slightly different frequencies, which is often acceptable.

Remark 27. (**Calculation basis**). The equations (3.14) and (3.15) which we used to implement the algorithm are the consequence of the sharing of vectors into two sub-vectors of size $N/2$. This is called an FFT in *base 2* (*radix-2* in English). We can think of using another base, for example 4, which leads to considering sums of the four sub-FFTs of length $N/4$. The advantage of such a choice (compared to base 2) is that one avoids making the obvious calculations of the fourth roots of the unit (which are coded simply by subtractions instead of additions in the formulas), which slightly reduces the number of operations to be performed. On the other hand, we must be careful, because the signs are not the same for the direct transform and for the reverse transform. To write the notations, we introduce the sub-vectors f^0, f^1, f^2 and f_3, of length $N/4$, which are constructed from f in considering only the indices congruent respectively to 0, 1, 2 and 3 modulo 4. We also use σ which is worth $+1$ for the direct transform, and -1 for the inverse transform. To discern the different portions of length $N/4$ of the result, we will write $\hat{f}^{(0/4)}$ for the first quarter, etc. Here are the equations:

$$\begin{aligned}
\hat{f}^{(0/4)} &= \mathcal{S}_{N/4}^{0/4}\hat{f}^0 && + \quad \mathcal{S}_{N/4}^{1/4}\hat{f}^1 && + \quad \mathcal{S}_{N/4}^{2/4}\hat{f}^2 && + \quad \mathcal{S}_{N/4}^{3/4}\hat{f}^3 \\
\hat{f}^{(1/4)} &= \mathcal{S}_{N/4}^{0/4}\hat{f}^0 && - \quad \imath\sigma\mathcal{S}_{N/4}^{1/4}\hat{f}^1 && - \quad \mathcal{S}_{N/4}^{2/4}\hat{f}^2 && + \quad \imath\sigma\mathcal{S}_{N/4}^{3/4}\hat{f}^3 \\
\hat{f}^{(2/4)} &= \mathcal{S}_{N/4}^{0/4}\hat{f}^0 && - \quad \mathcal{S}_{N/4}^{1/4}\hat{f}^1 && + \quad \mathcal{S}_{N/4}^{2/4}\hat{f}^2 && - \quad \mathcal{S}_{N/4}^{3/4}\hat{f}^3 \\
\hat{f}^{(3/4)} &= \mathcal{S}_{N/4}^{0/4}\hat{f}^0 && + \quad \imath\sigma\mathcal{S}_{N/4}^{1/4}\hat{f}^1 && - \quad \mathcal{S}_{N/4}^{2/4}\hat{f}^2 && - \quad \imath\sigma\mathcal{S}_{N/4}^{3/4}\hat{f}^3.
\end{aligned}$$

By choosing an arbitrary basis p, and by carrying out analogous calculations, one can handle vectors of size p^s, which can be advantageous. Here is the recurrence formula in general:

Proposition 34. *We keep the notations defined previously, but this time for the computation of a DFT using a* $p \geq 2$ *base. We have the equations*

$$\forall q = 0, \ldots, p-1, \quad \hat{f}^{(q/p)} = \sum_{k=0}^{p-1} e^{-\sigma \frac{2\imath\pi}{p} kq} \cdot \mathcal{S}_{N/p}^{k/p} \hat{f}^k.$$

Remark 28. Of course, this formula is only interesting in practice when we know how to calculate explicitly and simply the factors $e^{\frac{2\imath\pi}{p}kq}$, for example for $p = 2.4,\ 8$. The exercise 24 shows how, by mixing both base 2 and base 4 transforms, we can further optimize the number of operations.

3.2.4 The Cooley-Tukey transformation

We have just seen an FFT algorithm which allows to very quickly calculate the Fourier transform of a vector whose size is 2^p. But what if the size N of the signal is not written in this form? The easy solution, if we just do approximate calculations, is to add zeros to reach a reasonable size, which will of course be the power of 2 immediately after N. But often, we cannot act as directly, and we have to find a finer algorithm, to take advantage of other properties of the integer N. This is how many other versions of the FFT algorithm have emerged since Cooley-Tukey's seminal article. In this chapter, different variants of the algorithm are presented, and some really allow us to get out of bad spots (for example the *Good-Thomas* algorithm or the *split-radix* algorithm, presented in the exercises 23 and 24).

In the case where the number N is an integer that we know how to factorize, there is however a very simple method, which consists in looking more closely at the work carried out by the Cooley-Tukey method in the case where $N = 2^s = 2 \times 2^{s-1}$. Thus, without N necessarily being a power of 2, suppose that we have a factorization $N = pq$. In the case where the integers p and q are coprime, a remarkable algebraic property (the Chinese lemma) makes it possible to optimize the calculations, and gives rise to the Good-Thomas algorithm already mentioned. But for now, let's not worry about such refinements, let's just follow step by step the transformations already done "by hand" in paragraph 3.2.1. Recall the definition of the DFT of a vector $f \in \mathbb{C}^N$:

$$\hat{f}[k] \overset{\text{def.}}{=} \sum_{n=0}^{N-1} f[n]\omega_N^{-kn} \qquad \text{for} \quad k = 0, \dots, N-1. \tag{3.16}$$

The key idea to obtain a factorization of this expression is to perform a change of variables using the following two bijections:

$$\varphi : \begin{cases} \{0, \dots, q-1\} \times \{0, \dots, p-1\} & \longrightarrow & \{0, \dots, N-1\} \\ (a, b) & \longmapsto & ap + b \end{cases}$$
$$\psi : \begin{cases} \{0, \dots, p-1\} \times \{0, \dots, q-1\} & \longrightarrow & \{0, \dots, N-1\} \\ (c, d) & \longmapsto & cq + d \end{cases} .$$

We can indeed rewrite the sum (3.16) in the form

$$\begin{aligned} \hat{f}[\psi(c, d)] &= \sum_{a=0}^{q-1}\sum_{b=0}^{p-1} \omega_N^{-(ap+b)(cq+d)} f[\varphi(a, b)] \\ &= \sum_{b=0}^{p-1} \omega_N^{-b(d+cq)} \sum_{a=0}^{q-1} \omega_q^{-ad} f[\varphi(a, b)]. \end{aligned}$$

If we denote by $f_b[a] \overset{\text{def.}}{=} f[\varphi(a, b)]$ (which corresponds to taking only one column of f, if we represent it in the form of a matrix of size $p \times q$), then we get

$$\hat{f}[\psi(c, d)] = \sum_{b=0}^{p-1} \omega_p^{-cb} \left(\omega_N^{-bd} \hat{f}_b[d]\right). \tag{3.17}$$

We have therefore succeeded in modifying the calculation algorithm to obtain an algorithm operating in 2D, on the size matrix $p \times q$ that constitutes $F \overset{\text{def.}}{=} \{f[\varphi(a, b)]\}_{a,b}$. In fact, if we didn't have the parasitic terms ω_N^{-bd} (often called "twiddle factor" in English, see the exercise 24), we would simply be calculating the two-dimensional DFT of the 2D function F (which can also be considered as an image).

If we count the number of operations necessary to calculate the DFT of f by this method, we obtain $Cpq(p+q)$, where C represents a constant taking into account the calculation time of complex additions and multiplications. But the advantage of the method is that it can be applied recursively to each of the sub-DFTs to be calculated. Thus, if N is factored in the form $p_1 \times p_2 \times \cdots \times p_s$, we obtain a number of operations proportional to $N \sum p_i$. Of course, in the case where $N = 2^s$, we find the traditional FFT algorithm already described in paragraph 3.2.1. However, we see that with a little adaptation, we can easily take into account N admitting more complex decompositions. Be careful, however, not to fall into an excess of optimism: this method will be totally inefficient when N is factored badly. In this case, we must opt for other approaches, such as the one suggested in the 53 exercise. Moreover, when the factorization $N = pq$ has particularities (typically if p and q are coprime), there are more optimized algorithms, like that of *Good-Thomas* presented at l'exercise 23.

3.2.5 Concrete implementation

The naive implementation presented in paragraph 3.2.1 (in the case $N = 2^p$) suffers from many weak points, among which we can note:

– *a recursive structure*: Recursive calls require additional system instructions, which wastes a lot of time.
– *a use of temporary memories*: the explicit computation of the two subvectors f^0 and f^1 of size $N/2$ is obviously an enormous loss of memory (since redundant information is created).

We will see in this paragraph how to implement a routine that solves these two problems at once. The main idea is to rearrange the starting vector. We want the elements of the vector to be arranged so that at each subdivision (in the form of two vectors of half size), the first vector is the $N/2$ first entries, and the second vector is the $N/2$ last (and not the even and odd indices). For an implementation in a classic language (C or C ++ for example), the gain will be enormous: by the use of pointers (or, for the uninitiated, by moving the start of the array), the only memory used by the vector of origin allows to accommodate the two sub-tables.

In the following, we will note the indices in binary form, that is to say

$$i = [i_{p-1} \dots i_0]_b = \sum_{t=0}^{p-1} i_t 2^t.$$

Our goal is to start the algorithm with a vector $g \overset{\text{def.}}{=} \{f[n_p(0)], \dots, f[n_p(N-1)]\}$, where $i \mapsto n_p(i)$ denotes a permutation of the indices. We want that when applying the equation of *Danielson-Lanczos*

$$\hat{g}[k] = \hat{g^0}[k] + \omega_N^{-k}\hat{g^1}[k], \tag{3.18}$$

the vector g^0 is made up of the entries of f with indices $0, \dots, N/2-1$, and that the vector g^1 is made up of the entries of f d'indices $N/2, \dots, N-1$. Thus, dividing g in two is done without having to move values in the memory of the computer. So that this construction still works during recursive calls on g^0 and g^1, these two subvectors are themselves permuted from f^0 and f^1 by n_{p-1}, which meets the same requirements as n_p. This condition, translated on the permutation n_p, is expressed as follows:

$$n_p([i_{p-1} \dots i_0]_b) = i_0 2^{p-1} + n_{p-1}([i_{p-1} \dots i_1]_b).$$

By iterating this equation p times, we find the expression for the permutation n_p:

$$n_p(i) = n_p([i_{p-1} \dots i_0]_b) = \sum_{t=0}^{p-1} i_t 2^{p-1-t}.$$

More concisely, $n_p(i)$ is the transpose of i written in binary. For example, for $N = 8$, if $i = 6$, which is written 110 in binary, then $n_p(i)$ will be written 011, i.e. $n_p(6) = 3$.

In the end, we see that we must classify the elements of the vector according to the reverse binary writing of the indices. This is what the `rev _bits` procedure, described in program 69, does. This procedure requires $O(N)$ operations. For a more refined implementation, we can look at the Numerical Recipes [53]. Figure 3.3 shows the permutation matrix corresponding to n_p, i.e. the matrix $M^{(p)}$ such that $M_{ij}^{(p)} = \delta_i^{n_p(j)}$. The black dots represent the non-zero entries (equal to 1) in the matrix $M^{(p)}$. The exercise 22 proposes to write

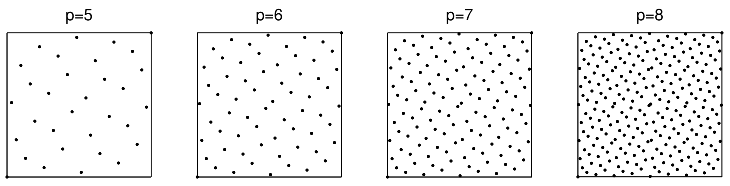

Figure 3.3: Bit inversion matrix

a recursive function to perform the bit reversal. Using the `rev _bits` procedure allows you to write a `fft _dit` function that does not use temporary memory. The end of this procedure replaces the recursive calls with nested `for` loops. The figure 3.4 shows the operations to perform to reverse the inputs of a vector, highlighting the necessary permutations.

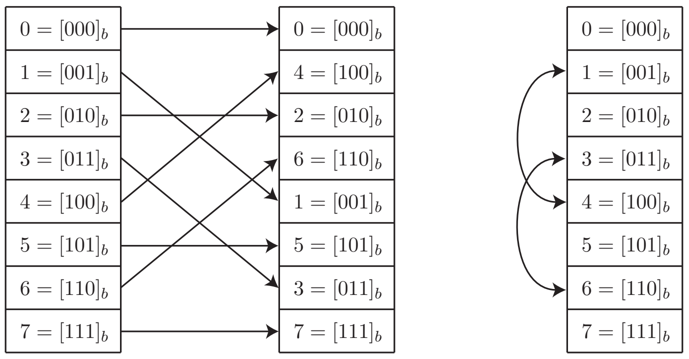

Figure 3.4: Inversion of bits by permutation of inputs

3.2.6 Frequency decimation

We are going to redo the calculations that led to the equations (3.14) and (3.15), but this time by performing a grouping according to the frequencies of the transform. The algorithm that we will obtain will be in a way the symmetric of the "classical" algorithm proposed by Cooley and Tukey. Even if this new implementation will not improve speed of execution, it is important to have in mind the two dual versions of the FFT, just as it is important to master the temporal and frequency properties of the Fourier transform.

In accordance with the notations (3.11), we denote by f_g (resp. F_d) the $N/2$ first entries (resp. $N/2$ last) of the vector f. We have

$$\hat{f}[k] = \sum_{n=0}^{N/2-1} \left(f_g[n] + e^{-kN/2\frac{2\imath\pi}{N}} f_d[n] \right) e^{-nk\frac{2\imath\pi}{N}}.$$

We are therefore led to make a distinction according to the parity of k. By following the notations of the equation (3.7), we consider $(\hat{f}\,)^0$ (resp. $(\hat{f}\,)^1$) the even (resp. odd) part of the transformed vector. Be careful, these vectors should not be confused with $\hat{f}^0$ and $\hat{f}^1$, which are the transforms of the vectors f^0 and f^1. We therefore write, for $k \in \{0, \ldots, N/2-1\}$,

$$(\hat{f}\,)^0[k] = \hat{f}[2k] = \sum_{n=0}^{N/2-1} (f_g[n] + f_d[n])\, e^{-nk\frac{2\imath\pi}{N/2}}$$
$$(\hat{f}\,)^1[k] = \hat{f}[2k+1] = \sum_{n=0}^{N/2-1} e^{-k\frac{2\imath\pi}{N}} (f_g[n] - f_d[n])\, e^{-nk\frac{2\imath\pi}{N/2}}.$$

By using the operator $\mathcal{S}_N^x$ introduced in (3.13), we obtain the following recurrence equations:

$$(\hat{f}\,)^0 = \mathcal{F}(f_g + f_d)$$
$$(\hat{f}\,)^1 = \mathcal{F}\left(\mathcal{S}_{N/2}^{1/2}(f_g - f_d)\right).$$

Contrary to the technique of temporal decimation, we see that the sub-vectors for which we must calculate the Fourier transform are directly obtained from the input vector (just take the left and right parts). On the other hand, the output vector must be composed, according to the parity of the index, either of the values of one transform or of the other. To avoid having to use temporary memory, we will use the same trick as for the temporal decimation, but in the other direction. We will be satisfied with juxtaposing the two transforms, that is to say putting the vectors $(\hat{f}\,)^0$ then $(\hat{f}\,)^1$. To obtain the correct result, it will suffice, at the end of the procedure, to put the frequencies back in the correct order, by calling the `rev _bits` function. We can then write a non-iterative version of the FFT which uses the principle of frequency decimation, it is the procedure `fft _dif` which is written in paragraph 67.

Remark 29. (**Time and frequency**). We can see that the frequency decimation is exactly symmetrical to the temporal decimation. Acting on the indices of the result vector (i.e. on the frequencies) instead of acting on the indices of the input vector results in a reversal of the bits in the final phase of the algorithm .

To conclude, we can already notice the wide variety of variations of the FFT algorithm at our disposal. Many other methods will also be described in the following chapters and exercises. The literature revolving around FFT is gigantic, it is undoubtedly one of the most extensive fields of numerical analysis. Review articles have been written, for example by Burrus [12]. The question is therefore to know which is the best method. Of course, there is no definitive answer, because too many factors come into play, not only concerning the length of the transform and the type of data (real, complex, etc.), but above all the type of architecture (machine, operating system, parallel architecture, memory cache, etc.) and the type of precision desired. If in doubt, it is better to stay on a simple, but robust implementation, even if it means sacrificing a little efficiency.

3.2.7 Matrix writing

If we write the matrix Ω_N of the linear operator $\mathcal{F} : \mathbb{C}^N \to \mathbb{C}^N$ in canonical bases, we get

$$\Omega_N \overset{\text{def.}}{=} \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_N^{-1} & \omega_N^{-2} & \dots & \omega_N^{-(N-1)} \\ 1 & \omega_N^{-2} & \omega_N^{-4} & \dots & \omega_N^{-2(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega_N^{-(N-1)} & \omega_N^{-2(N-1)} & \dots & \omega_N^{-(N-1)(N-1)} \end{pmatrix}. \tag{3.19}$$

This matrix corresponds to a matrix of *Vandermonde*. These matrices occur when we write the linear system corresponding to the search for the unique polynomial of degree N passing through N distinct points. There is nothing surprising about this, since we will see in Section 4.5, that the computation of inverse DFT corresponds to the computation of the coefficients of the interpolation polynomial at points quite particular, the N^{th} roots of the unit.

The formula of the inverse Fourier transform (3.3) results in the fact that the inverse of the matrix Ω_N is the matrix $\frac{1}{N}\Omega_N^*$, where we denote by $M^* \overset{\text{def.}}{=} \overline{M}^\top$ the adjoining matrix of M. This means that the matrix $\frac{1}{\sqrt{N}}\Omega_N$ is unitary, that is to say $\Omega_N \Omega_N^* = N\,\mathrm{Id}_N$. The equations of *Danielson-Lanczos* (3.14) and (3.15) can then be written in the form of a factorization of the matrix Ω_N:

$$\Omega_N \begin{pmatrix} a_0 \\ \vdots \\ a_{N-1} \end{pmatrix} = \begin{pmatrix} \Omega_{N/2} & \Delta_{N/2}\Omega_{N/2} \\ \Omega_{N/2} & -\Delta_{N/2}\Omega_{N/2} \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_{N-2} \\ a_1 \\ \vdots \\ a_{N-1} \end{pmatrix},$$

where we noted $\Delta_{N/2} = \mathrm{diag}(1, \omega_N^{-1}, \dots, \omega_N^{-(N/2-1)})$.

3.3 Circular convolution

We have defined in paragraph 1.4.3, the product of convolution on any abelian group, and we will now apply this definition as well as the convolution theorem 2, in the simple case of a cyclic group, and more precisely by using the language of the discrete Fourier transform which was defined in paragraph 3.1.

3.3.1 Circular convolution

Let us start by recalling the definition of the convolution product as well as the main results already obtained.

Definition 20 (Discrete convolution product). *Let $\{f[n]\}_{n=0}^{N-1}$ and $\{g[n]\}_{n=0}^{N-1}$ two discrete samples (assumed to represent signals sampled at the same times, regularly spaced). We define the convolution product $f * g$ of the two signals by the equation*

$$(f * g)[n] \overset{\text{def.}}{=} \sum_{k=0}^{N-1} f[k]g[nk], \qquad n = 0, \dots, N-1. \tag{3.20}$$

Remark 30. In the equation (3.20), the quantity nk is of course calculated modulo N, which amounts to considering the samples f and g as periodic functions of period N. This formula is the translation of the equation (1.15), in the case of the group $G = \mathbb{Z}/N\mathbb{Z}$, taking care to use an additive notation instead of multiplicative notation. From the perspective of a computer implementation, we can give a more explicit formula:

$$(f * g)[n] \overset{\text{def.}}{=} \sum_{k=0}^{n} f[k]g[nk] + \sum_{k=n+1}^{N-1} f[k]g[n-k+N], \qquad n = 0, \dots, N-1.$$

Proposition 35. *The circular convolution product is commutative, and the mapping $(f, g) \mapsto f * g$ endows $\mathbb{C}^N$ with an algebra structure.*

Proof. The only non-trivial thing to check is the commutativity, which is obtained by making the change of variable $k' = nk$ in the equation (3.20). □

We can now state the convolution theorem 2, in terms of a discrete Fourier transform.

Proposition 36 (Convolution and TFD). *Let $\{f[n]\}_{n=0}^{N-1}$ and $\{g[n]\}_{n=0}^{N-1}$ two discrete samples. We have the convolution formula*

$$\forall n \in \{0, \dots, N-1\}, \quad f \hat{*} g[n] = \hat{f}[n]\hat{g}[n]. \tag{3.21}$$

Proof. For the explanations to be clearer, we will write f_1 and g_1 the functions of $\mathbb{Z}/N\mathbb{Z}$ in $\mathbb{C}$ associated with the samples f and g (which are of size N). We then have, for $n \in \{0, \dots, N-1\}$ (where, in terms of an abelian group, $n \in \mathbb{Z}/N\mathbb{Z}$),

$$\hat{f}[n] = \hat{f_1}(\chi_n) \qquad \text{et} \qquad \hat{g}[n] = \hat{g_1}(\chi_n), \tag{3.22}$$

where we noted $\{\chi_0, \dots, \chi_{N-1}\}$ the characters, that is to say the elements of the dual $\widehat{\mathbb{Z}/N\mathbb{Z}}$ (see the equation (3.2)). Using the theorem 2, for the functions f_1 and g_1 on $G = \mathbb{Z}/N\mathbb{Z}$, we obtain

$$f_1 \hat{*} g_1(\chi_n) = \hat{f_1}(\chi_n)\hat{g_1}(\chi_n).$$

However, we also have

$$f \hat{*} g[n] = f_1 \hat{*} g_1(\chi_n).$$

This thus makes it possible to write, using the equations (3.22),

$$f \hat{*} g[n] = \hat{f_1}(\chi_n)\hat{g_1}(\chi_n) = \hat{f}[n]\hat{g}[n].$$

Remark 31. (**Finite signals and periodization**). The main theoretical difficulty of the discrete Fourier transform is the assimilation between our sample $\{f[n]\}_{n=0}^{N-1}$ and a function f set to $\mathbb{Z}/N\mathbb{Z}$. This assimilation has the advantage of obtaining algebraic formulas at a lower cost such as the result of inversion 31 as well as that of convolution 36. However, this approach implies that our function f, if we look at it as a signal in time is in fact a periodic function, of period N. This goes against the natural intuition which wants us to consider our (finite) signal f as zero outside the interval in which it is defined. It is on this point that we will have to pay attention when we want to calculate the convolution products between two finite signals. It is precisely this problem which is raised in Paragraph 3.3.3 during the study of non-circular convolution.

3.3.2 Calculation with the FFT

A naive implementation of the equation (3.20) leads to a number of operations (complex multiplications and additions) of the order of $O(n^2)$. Indeed, it is necessary to calculate the N values of the convolée, and each time, a sum of N products appears. However, using the convolution formula (3.21) and the inversion formula (3.3), we can write an equation that will turn out to be very useful :

$$f * g = \mathcal{F}^{-1}\left(\hat{f} \cdot \hat{g}\right),$$

where we noted f and $g \in \mathbb{C}^N$ two samples of size N. Thanks to the FFT algorithm, the calculation of the transforms $\hat{f}$ and $\hat{g}$ can be done in a number of operations of the order of $O(N\log(N))$, and the computation of the product $\hat{f} \cdot \hat{g}$ requires of course only N complex multiplications. In the end, we thus manage to calculate a convolution product with a number of operations of the order of $O(N\log(N))$.

3.3.3 Acyclic convolution

We are going to leave for a short time the transformations linked to the group structure of $\mathbb{Z}/N\mathbb{Z}$ to define an operation which does not respect this cyclic structure at all, the acyclic convolution (also called linear convolution), denoted $\star$ (not to be confused with the $*$ of cyclic convolution). The support of a $f \in \mathbb{C}^{\mathbb{Z}}$ signal is defined by

$$\mathrm{Supp}(f) \overset{\text{def.}}{=} \{n \in \mathbb{Z} \ : \ f[n] \neq 0\}.$$

We start by defining the acyclic convolution for two signals $\{f_1[n]\}_{n\in\mathbb{Z}}$ as well as $\{f_2[n]\}_{n\in\mathbb{Z}}$ whose support is assumed to be finite, which means that $\mathrm{Supp}(f_1)$ and $\mathrm{Supp}(f_2)$ are finite sets. We then define the sequence $f_1 \star f_2$ by

$$\forall n \in \mathbb{Z}, \quad f_1 \star f_2[n] = \sum_{k\in\mathbb{Z}} f_1[k] f_2[nk]. \tag{3.23}$$

Note that we have the very useful equation:

$$\mathrm{Supp}(f_1 \star f_2) \subset \mathrm{Supp}(f_1) + \mathrm{Supp}(f_2) \overset{\text{def.}}{=} \{n + p \ : \ n \in \mathrm{Supp}(f_1),\ p \in \mathrm{Supp}(f_2)\}.$$

Linear convolution therefore has nothing to do with cyclic convolution, which is an operation on vectors of $\mathbb{C}^N$ (and results in a vector of $\mathbb{C}^N$). However, by creating from our two sequences, two vectors $\tilde{f_1}$ and $\tilde{f_2}$ of size N sufficiently large, we will see that we can calculate the non-zero values of $f_1 \star f_2$ as some entries of the vector $\tilde{f_1} * \tilde{f_2}$.

Let us start by noticing that the size necessary to store the entries of $f_1 \star f_2$ is $N \overset{\text{def.}}{=} N_1 + N_2 - 1$, where we have noted N_1 and N_2 the sizes supports of f_1 and f_2. We can translate the indices of f_1, which allows us to assume that they are $\{0, \dots, N_1 - 1\}$. This implies that we must perform the same translation on the vector f. So let's start by creating a vector $\tilde{f_1} \in \mathbb{C}^N$ by first copying the non-zero N_1 entries of f_1, then adding zeros. The construction of the vector $\tilde{f_2}$ is a little more difficult, since it is necessary to take into account the negative indices. Copy the entries with positive indices of f_1 into $\tilde{f_2} \in \mathbb{C}^N$, then put enough zeros, then copy the entries with negative indices. More precisely, if we write $\mathrm{Supp}(f_2) = \{-P, \dots, 0, \dots, Q\}$, with $N_2 = Q + P + 1$, then we will have

$$\tilde{f_2} \overset{\text{def.}}{=} \{f_2[0],\ f_2[1], \dots,\ f_2[Q],\ 0, \dots,\ 0,\ f_2[-P], \dots,\ f_2[-1]\} \in \mathbb{C}^N.$$

Once all these transformations have been carried out, we can finally write:

$$\forall n \in \{0, \dots,\ N_1 + Q - 1\}, \quad f_1 \star f_2[n] = \tilde{f_1} * \tilde{f_2}[n].$$

For indices located in the interval $\{-P, \dots, -1\}$, care must be taken because, due to circular convolution, they have been moved in the interval $\{NP, \dots,\ N - 1\}$. However, in practice (for example, for filtering), we only use the indices $\{0, \dots,\ N_1\}$.

Once this transformation is done, we can of course use the algorithm presented in Section 3.3.2 to quickly calculate the convolution. This algorithm, which goes hand in hand with the technique of adding zeros that we have just explained, will allow filtering to be carried out quickly. All this will be explained in detail in paragraph 4.2. It may be noted that when the size of one of the two vectors is much smaller than that of the other, there is a strategy which makes it possible to avoid adding too many zeros at the end of the shorter vector. This method is exposed to the exercise 25.

In the following, we will often directly consider the linear convolution of two vectors of $\mathbb{C}^N$, and in this case, the negative indices will be placed at the end of the vector, (it will therefore be necessary to add zeros between the positive indices and these negative indices to be able to use the FFT algorithm). However, it must be remembered that cyclic and acyclic convolutions give very different results. For example, the figure 3.5 shows a comparison of the two convolutions. Filtering by g performs a kind of "local mean". For the central values of k, more precisely $2 \leq k \leq N-4$, we have $f * g[k] = f \star g[k]$. However, for the edge values, we find different results. Thus, in the majority of applications where the vector x will represent a temporal

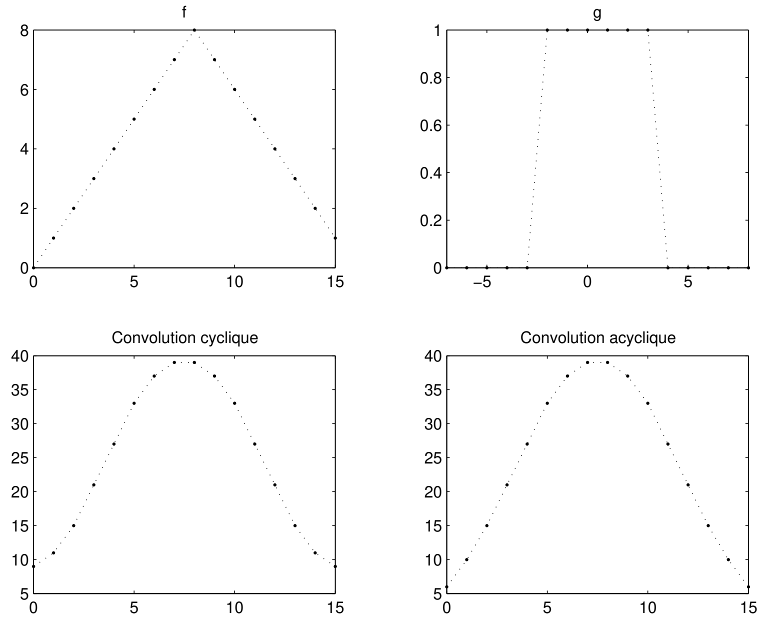

Figure 3.5: Cyclic and acyclic convolutions

signal, acyclic convolution will be preferred, so as not to alter the values on the edges. All this will be covered in detail when explaining the different types of filtering, in Section 4.2.

3.4 In higher dimension

In this paragraph, to simplify the explanations, we will restrict ourselves to calculations of transforms in dimension 2. Generalization to higher dimensions, even if it can be dangerous from the point of view of programming, does not present theoretical difficulties.

3.4.1 Discrete Fourier transform in 2D

Definition 21 (two-dimensional DFT)**.** *A two-dimensional sample is represented by a matrix* $\{f[i,\, j]\} \in \mathbb{C}^{N \times P}$.
The indices are therefore $i \in \{0, \ldots,\, N-1\}$ *and* $j \in \{0, \ldots,\, P-1\}$. *Its discrete Fourier transform is a*

$N \times P$ matrix defined by

$$\hat{f}[k,\, l] \overset{\text{def.}}{=} \sum_{i,\, j} f[i,\, j] e^{-\frac{2\imath\pi}{N} ik} e^{-\frac{2\imath\pi}{P} jl}, \tag{3.24}$$

where $k \in \{0, \dots, N-1\}$ and $l \in \{0, \dots, P-1\}$.

As for the one-dimensional case, we can still make the link with the Fourier transform on an abelian group, by considering the group $G \overset{\text{def.}}{=} \mathbb{Z}/N\mathbb{Z} \times \mathbb{Z}/P\mathbb{Z}$. The characters in this group are χ_{ij}, for $0 \le i < N$ and $0 \le j < P$, defined by

$$\forall (n,p) \in \mathbb{Z}/N\mathbb{Z} \times \mathbb{Z}/P\mathbb{Z}, \quad \chi_{ij}(n,p) \overset{\text{def.}}{=} (\omega_N)^{-in} (\omega_P)^{-jp}.$$

We can therefore translate the equation (3.24) by

$$\forall k \in \{0, \dots, N-1\},\ \forall l \in \{0, \dots, P-1\}, \quad \hat{f}[k,\, l] = \hat{f}(\chi_{kl}),$$

where we noted f both the sample and the associated function $f : G \to \mathbb{C}$.

Once again, we see that the function

$$\mathcal{F} : \left\{ \begin{array}{ccc} \mathbb{C}^{N\times P} & \longrightarrow & \mathbb{C}^{N\times P} \\ f & \longmapsto & \mathcal{F}(f) = \hat{f} \end{array} \right.$$

is an algebra isomorphism of which we explicitly know the reverse.

Proposition 37 (2D inversion formula)**.** *Let $f \in \mathbb{C}^{N\times P}$ be a 2D sample of size $N \times P$. We have the inversion formula*

$$f[i,\, j] = \frac{1}{NP} \sum_{k,\, l} \hat{f}[k,\, l] e^{\frac{2\imath\pi}{N} ik} e^{\frac{2\imath\pi}{P} jl},$$

for $i \in \{0, \dots, N-1\}$ and $j \in \{0, \dots, P-1\}$.

The important point is of course whether we still have a fast algorithm to calculate the DFT in dimension two. The answer is given by a simple rewrite of the equation (3.24), for $k \in \{0, \dots, N-1\}$ and $l \in \{0, \dots, P-1\}$:

$$\hat{f}[k,\, l] = \sum_{i=0}^{N-1} \left(\sum_{j=0}^{P-1} f[i,\, j] e^{-\frac{2\imath\pi}{P} jl} \right) e^{-\frac{2\imath\pi}{N} ik} = \sum_{i=0}^{N-1} \hat{F}_i[l] e^{-\frac{2\imath\pi}{N} ik},$$

where we denote by $F_i \in \mathbb{C}^P$ the vector formed by the i^{th} row of the matrix f, and $\hat{F}_i$ its one-dimensional DFT.

To calculate the TFD in 2D of a f matrix, it is therefore sufficient to calculate the TFD of each of its rows, then to calculate the TFD of the columns of the matrix obtained. In a more synthetic way, we can write matricially:

$$\hat{f} = \mathcal{F}_{1\text{D}} \left(\mathcal{F}_{1\text{D}}(f)^\top \right)^\top,$$

where the operator $\mathcal{F}_{1\text{D}}$ performs the one-dimensional DFT on the rows of a matrix. It is also possible to carry out the calculations in the reverse direction, that is to say first calculate the transform on the columns, then on the rows. Matrix, like $\Omega_N^\top = \Omega_N$, the transformation equation is written $\hat{f} = \Omega_N f \Omega_P$, where Ω_N is defined at the equation (3.19).

Figure 3.6 shows the 2D Fourier transform of an image, which is just another way of representing a 2D sample (the values of the function are represented by levels gray, varying from black for 0 to white for 1). We can intuitively interpret the spectrum obtained. The value of $\hat{f}[i,\, j]$, which we can "read" directly on the image representing the spectrum, corresponds to a certain amount of(two-dimensional) oscillations present in the image. Please note, for the Fourier transform (right image), the large coefficients are shown in black. These oscillations are characterized by a frequency, $\frac{1}{N}\sqrt{i^2+j^2}$, and a direction, that of the vector $(i,\, j)$.

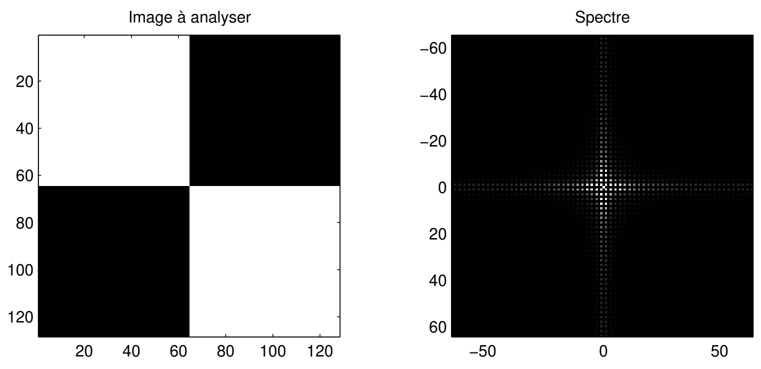

Figure 3.6: 2D Fourier transform

3.4.2 2D convolution

The convolution between two two-dimensional signals is a direct generalization of the cyclic convolution described in Section 3.3.1. Once again, we can keep in memory the definition of convolution over a finite group (defined in paragraph 1.4.3). This is of course to consider the group $G \overset{\text{def.}}{=} \mathbb{Z}/N\mathbb{Z} \times \mathbb{Z}/P\mathbb{Z}$. We can then interpret the functions of $\mathbb{C}[G]$ as images of size $N \times P$, which we would have extended by periodicity along the two axes. Here is the definition of convolution between two two-dimensional signals. We check that this is an immediate translation of the definition given in the framework of finite abelian groups.

Definition 22 (Two-dimensional convolution). *Let f and g be two samples of size $N \times P$. We define their cyclic convolution product $f * g$, which is a matrix of size $N \times P$, as follows:*

$$(f * g)[i,\, j] \overset{\text{def.}}{=} \sum_{k=0}^{N-1} \sum_{l=0}^{P-1} f[k,\, l] g[ik,\, jl], \tag{3.25}$$

for $i \in \{0, \dots, N-1\}$ and $j \in \{0, \dots, P-1\}$. Of course, all the operations on the indices must be carried out modulo N (resp. P) for the indices on the left (resp. On the right).

A naive calculation of the convolution product directly by the formula (3.25) requires $(NP)^2$ operations. In order to quickly calculate such a product, it is necessary to use the algebra morphism property of the Fourier transform on a finite group, which is recalled here in the context of the 2D Fourier transform.

Proposition 38 (2D and TFD convolution). *Let f and g be two samples of size $N \times P$. We have the convolution formula*

$$\forall i \in \{0, \dots, N-1\},\ \forall j \in \{0, \dots, P-1\}, \quad f \hat{*} g[i,\, j] = \hat{f}[i,\, j]\hat{g}[i,\, j]. \tag{3.26}$$

Proof. The proof is the exact copy of that of the proposition 36. It is simply necessary to change the cardinal of the group (which is worth NP and no longer N), and to use an indexing adapted for the characters and the indices of the samples, i.e. $i \in \{0, \dots, N-1\}$ and $j \in \{0, \dots, P-1\}$. □

This theorem suggests, in order to calculate a convolution, to use the technique to which we are starting to be accustomed. First, you have to calculate the DFTs of the two signals you want to combine. Then, we must multiply them point to point, and finally calculate the inverse transform of the signal obtained. Be careful with the fact that to implement this algorithm, it is necessary to move the entries of negative indices in the two signals, so as to have a signal N periodic on the abscissas, P periodic on the ordinates, and with indices $(i,\, j)$ such as $i \in \{0, \dots, N-1\}$ and $j \in \{0, \dots, P-1\}$.

We will see in paragraph 4.2.3, where we will talk about 2D filtering, what are the “intuitive” properties of cyclic convolution, as well as immediate applications to image analysis. We can however give an example of convolution on functions represented by their graph in 3D. Thus the figure 3.7 represents an irregular f function that we have convolated with a g function having the shape of a bump (and integral equal to 1). Convolution has a regularization effect since it realizes a weighted average of the original function in the neighborhood of each point.

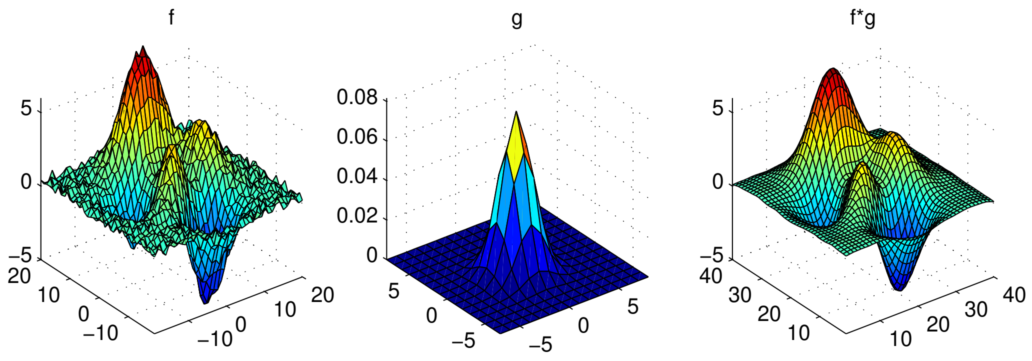

Figure 3.7: 2D Convolution

3.5 Symmetry and discrete transform

In this paragraph, we will give some additional properties of the discrete Fourier transform, and thus create its eigenvectors from given vectors.

3.5.1 Symmetry properties

Let's start by defining different operations on the functions of $\mathbb{Z}/N\mathbb{Z}$ in $\mathbb{C}$.

Definition 23 (Symmetry operator)**.** *Let $f = \{f[0], \ldots, f[N-1]\}$ be a vector of size N which we associates a periodic function f_1, which can be seen as a function $f_1 : \mathbb{Z}/N\mathbb{Z} \to \mathbb{C}$. We define the symmetrized function $f^\sharp$ by:*

$$\forall n \in \{0, \ldots, N-1\}, \quad f^\sharp[n] \overset{\text{def.}}{=} f_1(-n). \tag{3.27}$$

Thus, we have $f^\sharp = \{f[0], f[N-1], f[N-2], \ldots, f[1]\}$.
A vector f is said to be symmetric if it satisfies $f^\sharp = f$. It says anti-symmetric if $f^\sharp = -f$.

Definition 24 (Decomposition)**.** *For $f \in \mathbb{C}^N$, we denote by f_S and f_A the parts symmetric and anti-symmetric of f, defined by the equations*

$$f_S \overset{\text{def.}}{=} \frac{1}{2}\left(f + f^\sharp\right),$$
$$f_A \overset{\text{def.}}{=} \frac{1}{2}\left(f - f^\sharp\right).$$

We have of course the decomposition $f = f_S + f_A$.

Proposition 39 (Symmetry properties)**.** *Let $f \in \mathbb{C}^N$ be a sample. We have the following properties.*

(i) $\mathcal{F}(f^\sharp) = N\mathcal{F}^{-1}(f)$ *as well as* $\mathcal{F}^2(f^\sharp) = Nf$.

(ii) *If f is symmetric, then $\mathcal{F}^2(f) = Nf$ and $\mathcal{F}(f)$ is symmetric.*

(iii) *If f is anti-symmetric, then $\mathcal{F}^2(f) = -Nf$ and $\mathcal{F}(f)$ is anti-symmetric.*

(iv) *If $f \in \mathbb{R}^N$ is symmetric, then $\mathcal{F}(f) \in \mathbb{R}^N$.*

(v) *If $f \in \mathbb{R}^N$ is anti-symmetric, then $\mathcal{F}(f) \in (\imath\mathbb{R})^N$.*

Proof. We prove (i) and (iv):
For (i), we have

$$\mathcal{F}(f^\sharp) = \sum_{k \neq 0} f[-k]\omega_N^{-kn} + f[0] = \sum_{k=0}^{N-1} f[k]\omega_N^{kn} = \mathcal{F}(f)[n].$$

For (iv), if we denote by $\overline{z}$ the conjugate of $z \in \mathbb{C}$, we have

$$\overline{\mathcal{F}(f)[n]} = \sum_k \overline{f[k]} e^{+\frac{2\imath\pi}{N}kn} = \sum_k f^\sharp[k] e^{\frac{2\imath\pi}{N}kn} = \mathcal{F}(f^\sharp)[n] = \mathcal{F}(f)[n].$$

3.5.2 Eigenvalues of the TFD

The study of a linear operator is greatly facilitated by the knowledge of its eigenvalues and the associated eigenvectors. Although the $\frac{1}{\sqrt{N}}\Omega_N$ matrix is arguably the most important unit matrix, finding its eigenvectors is a difficult subject. We will now give a simple way to construct eigenvectors of the DFT.

Theorem-Definition 1. *Let $f \in \mathbb{C}^N$ be a sample. We define*

$$\mathcal{U}_+(f) \overset{\text{def.}}{=} \sqrt{N} f_S + \mathcal{F}(f_S) \quad \text{and} \quad \mathcal{U}_-(f) \overset{\text{def.}}{=} \sqrt{N} f_S - \mathcal{F}(f_S)$$
$$\mathcal{V}_+(f) \overset{\text{def.}}{=} \sqrt{N} f_A + \imath\mathcal{F}(f_A) \quad \text{and} \quad \mathcal{V}_-(f) \overset{\text{def.}}{=} \sqrt{N} f_A - \imath\mathcal{F}(f_A).$$

We then have

$$\mathcal{F}(\mathcal{U}_+(f)) = \sqrt{N}\mathcal{U}_+(f) \quad et \quad \mathcal{F}(\mathcal{U}_-(f)) = -\sqrt{N}\mathcal{U}_-(f)$$
$$\mathcal{F}(\mathcal{V}_+(f)) = -\imath\sqrt{N}\mathcal{V}_+(f) \quad et \quad \mathcal{F}(\mathcal{V}_-(f)) = \imath\sqrt{N}\mathcal{V}_-(f).$$

This means that the vectors $\mathcal{U}_+(f)$, $\mathcal{U}_-(f)$, $\mathcal{V}_+(f)$ and $\mathcal{V}_-(f)$ are eigenvectors of the discrete Fourier transform.

Proof. Let's prove the first equality: $\mathcal{F}(\mathcal{U}_+(f)) = \sqrt{N}\mathcal{F}(f_S) + \mathcal{F}^2(f_S)$. And since f_S is symmetric, we have $\mathcal{F}^2(f_S) = N f_S$, hence the result. □

Remark 32. We can add that the *eigenvalues* that we have just found are the only ones, since the Fourier transform satisfies $\mathcal{F}^4(f) = Nf$. So its eigenvalues are necessarily 4th roots of N^2.

Figure 3.8 shows the different constructed eigenvectors from the function that can be seen on the left of the figure 3.9 (that is to say for $\lambda = 0$). This proposition allows an interesting construction, simply by writing the decomposition of a vector $f \in \mathbb{C}^N$ as a function of the eigenvectors of the Fourier transform:

$$f = \mathcal{U}_+(f) + \mathcal{U}_-(f) + \mathcal{V}_+(f) + \mathcal{V}_-(f).$$

This allows to consider the operator $\sqrt{\mathcal{F}}$ defined as follows:

$$\sqrt{\mathcal{F}}(f) \overset{\text{def.}}{=} N^{1/4}\mathcal{U}_+(f) + \imath N^{1/4}\mathcal{U}_-(f) + (-\imath)^{1/2}N^{1/4}\mathcal{V}_+(f) + \imath^{1/2}N^{1/4}\mathcal{V}_-(f),$$

where we have chosen for $\imath^{1/2}$ a *square root* of $\imath$ (arbitrary choice).

We then have $\sqrt{\mathcal{F}} \circ \sqrt{\mathcal{F}} = \mathcal{F}$: the operator $\sqrt{\mathcal{F}}$ is a square root of the discrete Fourier transform. Likewise, for $\lambda \in \mathbb{R}$, we can thus construct $\mathcal{F}^\lambda$, a $\lambda^{\text{ième}}$ transform of $\mathcal{F}$ (again, the construction n' is absolutely nothing canonical). Figure 3.9 shows different intermediate transforms. For $\lambda = 0.5$ we get $\sqrt{\mathcal{F}}$. The exercise 30 allows to handle a *partial Fourier transform*, which generalizes the construction that we have just performed. We can see, thanks to the example of a Gaussian, that these manipulations correspond to very intuitive notions. For more information on the partial Fourier transform (continuous as well as discrete), one can consult the article of Cariolaro [14]. Finally, the exercise 31 proposes a method to canonically diagonalize the matrix of the TFD.

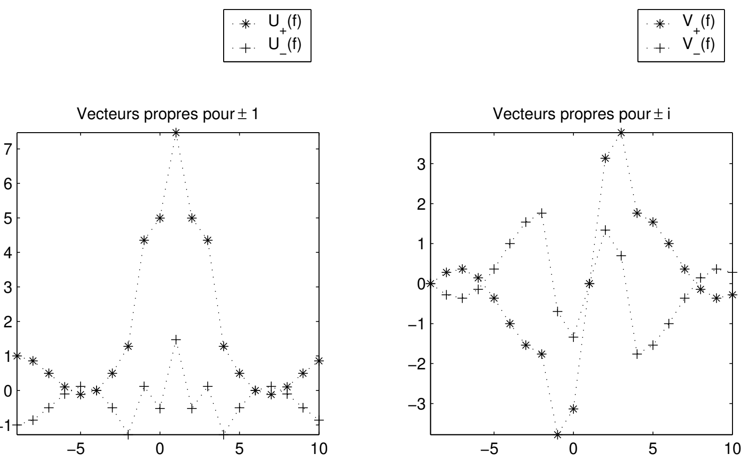

Figure 3.8: Eigenvectors $\mathcal{U}_+(f)$, $\mathcal{U}_-(f)$, $\mathcal{V}_+(f)$ and $\mathcal{V}_-(f)$

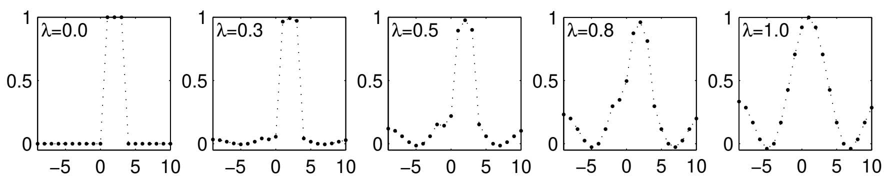

Figure 3.9: Intermediate transformed vectors $\mathcal{F}^\lambda(f)$ for $\lambda \in [0,1]$

3.6 Exercises

Exercise 22 (Bit inversion). *We define, for $n \geq 0$, vectors $u^{(n)}$ of size 2^n, by $u^{(0)} = \{0\}$ and*

$$\forall n > 0,\ \forall k \in \{0, \ldots, 2^n - 1\}, \quad u^{(n)}[k] \overset{\text{def.}}{=} \begin{cases} 2u^{(n-1)}[k] & si k < 2^{n-1} \\ 2u^{(n-1)}[k - 2^{n-1}] + 1 & si k \geq 2^{n-1} \end{cases}.$$

1. *Calculate the value of $u^{(n)}$ for $n = 1, 2, 3$.*
2. *Show that $u^{(n)}$ is in fact the sequence $0, \ldots, 2^n - 1$, classified by considering the reverse binary writes of the inputs.*
3. *Let f be a vector of size 2^n. We denote by $\tilde{f}$ the sequence determined by*

$$\forall k \in \{0, \ldots, 2^n - 1\}, \quad \tilde{f}[k] \overset{\text{def.}}{=} f[u^{(n)}[k]].$$

   What is the use of $\tilde{f}$ when computing the discrete Fourier transform of f?
4. *We denote by f^0 and f^1 the even and odd parts of f. We denote by $\tilde{f}_d$ and $\tilde{f}_g$ the left and right parts of $\tilde{f}$. What relation binds all these vectors?*
5. *Implement in Matlab a recursive algorithm to calculate $\tilde{f}$. Compare its complexity to that of the* `rev _bits` *procedure.*

Exercise 23 (Good-Thomas algorithm). *We saw in paragraph 3.2.4 that the FFT algorithm of Cooley-Tukey generalized without problem if we had an adequate factorization of N, the size of the transformed vector. In this exercise, we will see that if some integers of this factorization are coprime, we can design an even faster algorithm.*

1. *We assume that $N = p\,q$ where p and q are two prime numbers to each other. We recall the Chinese lemma, which says that the application*

$$\varphi \begin{cases} \mathbb{Z}/N\mathbb{Z} & \longrightarrow & \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z} \\ n & \longmapsto & (n \mod p,\, n \mod q) \end{cases}$$

   is an isomorphism of rings. Explain the inverse morphism ψ.

2. *Let $f \in \mathbb{C}^N$. We then define a 2D signal:*

$$\forall k_1 \in \{0, \ldots, p-1\},\ \forall k_2 \in \{0, \ldots, q-1\}, \quad F[k_1,\, k_2] \overset{\text{def.}}{=} f[\psi(k_1,\, k_2)].$$

   Show that we have

$$\hat{f}[(s_1 q + s_2 p) \mod N] = \hat{F}[s_1,\, s_2].$$

3. *Show that when s_1 traverses $\{0, \ldots, p-1\}$ and s_2 traverses $\{0, \ldots, q-1\}$, then $(s_1 q + s_2 p) \mod N$ iterates $\{0, \ldots, N-1\}$. Deduce how we can calculate the Fourier transform of f from that of F, by explaining the change of indices.*

4. *What is the gain compared to a step of the classic FFT algorithm? In particular, what becomes of the operator $\mathcal{S}_N^x$ introduced to the equation (3.13)? Propose a recursive procedure which, following the factorization of N obtained at each step, calls the optimal DFT calculation procedure. In addition to the FFT procedures of Cooley-Tukey and Good-Thomas, we can include the Chirp procedure described in the exercise 53, which is interesting when $N-1$ is a Prime number.*

Exercise 24 (Split-Radix algorithm)**.** *We saw in paragraph 3.2.3 that it was possible to extend the Cooley-Tukey dichotomy method to compute DFTs of length p^r, by grouping inputs by packets of size p. In this exercise, we will show how, by cleverly choosing packets of varying sizes, we can reduce the number of operations. This choice starts from the observation that the Cooley-Tukey algorithm spends a lot of time calculating the operator $\mathcal{S}_N^x$, whereas for some values of N (for example 2 or 4), this last is trivial. In Anglo-Saxon literature, the roots of the unit added by this operator are called "twiddle factors" (literally, "twiddling their thumbs"). We can compare this approach with that of the Good-Thomas algorithm, exercise 23, which in the context of a certain factorization of N, eliminates the operator $\mathcal{S}_N^x$.*

1. *We consider a frequency decimation scheme. Assume that N is a power of 2. It is recalled that the classical DIF algorithm organizes the following grouping:*

$$\left\{\hat{f}[k] \ :\ k = 0, \ldots, N-1\right\} = \left\{\hat{f}[2k] \ :\ k = 0, \ldots, \frac{N}{2} - 1\right\} \bigcup \left\{\hat{f}[2k+1] \ :\ k = 0, \ldots, \frac{N}{2} - 1\right\}.$$

   Explain why there is no interest in touching the first part of this grouping. Regarding the second part, we propose the grouping corresponding to base 4 transforms, that is to say:

$$\left\{\hat{f}[2k+1] \ :\ k = 0, \ldots, \frac{N}{2} - 1\right\} = \left\{\hat{f}[4k+1] \ :\ k = 0, \ldots, \frac{N}{4} - 1\right\} \bigcup \left\{\hat{f}[4k+3] \ :\ k = 0, \ldots, \frac{N}{4} - 1\right\}$$

   Show that this leads to the following transformation formulas:

$$\hat{f}[4k + 2j + 1] = \sum_{n_1=0}^{\frac{N}{4}-1} \omega_{N/4}^{-kn_1} \omega_N^{-n_1(2j+1)} \sum_{n_2=0}^{3} f\left[n_1 + n_2 \frac{N}{4}\right] \omega_4^{-n_2(2j+1)},$$

   for $j = 0,\, 1$ and $k = 0, \ldots, \frac{N}{4} - 1$. Are domestic sums complicated to calculate? Identify the "twiddle factors".

2. *Find other grouping schemes. Why is grouping by 4 advantageous? Calculate the number of operations required each time, and compare with that of the classic DIF diagram, in base 2.*

3. *Transform the algorithms described above to obtain a temporal decimation scheme (grouping of inputs). Describe an iterative implementation of the algorithms, not forgetting the bit reversal procedures allowing to save temporary memories.*

For more information, and a generalization to DFTs of length p^r, we can consult[68].

Exercise 25 (Optimized convolution)**.** *We want to compute the acyclic convolution of two finite sequences f and g of respective sizes N and M. Assume that M is much smaller than N. For simplicity, suppose that the indices of the two sequences start at* 0.

1. *We take $N = pM$. We denote by $f_j \overset{\text{def.}}{=} \{f[k + jM]\}_{k=0}^{M-1}$, for $j = 0, \ldots, p-1$. Show that we have*
$$f \star g[k] = \sum_{j=0}^{p-1} f_j \star g[k - jM].$$

2. *Deduce a quick way to calculate $f \star g$ without having to add $NM - 1$ zeros at the end of g. What is the complexity of the algorithm obtained?*

3. *If N is not a multiple of M, what can be done?*

Exercise 26 (Circulating matrix)**.** *This exercise has strong similarities with the exercise 1 on circulating determinants, with a presentation this time using the convolution product. Let $c \overset{\text{def.}}{=} (c_0, \ldots, c_{N-1})^\top \in \mathbb{C}^N$ be a vector of size N. We define the circulating matrix C which is associated with this vector by*
$$C \overset{\text{def.}}{=} \begin{pmatrix} c_0 & c_{N-1} & c_{N-2} & \ldots & c_1 \\ c_1 & c_0 & c_1 & \ldots & c_2 \\ \vdots & \vdots & \vdots & & \vdots \\ c_{N-1} & c_{N-2} & c_{N-3} & \ldots & c_0 \end{pmatrix}.$$

1. *Let $\{e_1, \ldots, e_N\}$ be the canonical basis of $\mathbb{C}^N$. We consider the matrix R whose columns are $\{e_2, e_3, \ldots, e_N, e_1\}$. We recall that Ω_N designates the Fourier matrix, which is defined by the equation* (3.19)*. Show that we have*
$$\Omega_N R \Omega_N^{-1} = D \qquad \textit{with} \quad D = \operatorname{diag}(1, \omega_N^{-1}, \ldots, \omega_N^{-(N-1)}).$$

2. *Show while we have*
$$\Omega_N C \Omega_N^{-1} = \Delta \qquad \textit{with} \quad \Delta = \operatorname{diag}(\hat{c}[0], \hat{c}[1], \ldots, \hat{c}[N-1]).$$
Deduce that for $x \in \mathbb{C}^N$, we can calculate the product Cx as follows:
$$Cx = \Omega_N^{-1}\left((\Omega_N c) \cdot (\Omega_N x)\right),$$
where we denote by $\cdot$ the product component by component of the matrices.

3. *Show that for $x \in \mathbb{C}$, we have $Cx = c * x$. Using the convolution theorem 36, deduce an immediate proof of the previous question.*

Exercise 27 (Trigonometric interpolation)**.** *Let $f \in \mathbb{C}^N$ be a sample of size $N = 2N_0 + 1$. We define a vector f_0 of size $P = \eta N$ (with $\eta \in \mathbb{N}$ sufficiently large) as follows:*
$$\hat{f}_0 \overset{\text{def.}}{=} \eta \left\{ \hat{f}[0], \hat{f}[1], \ldots, \hat{f}[N_0], 0, \ldots, 0, \hat{f}[N_0 + 1], \ldots, \hat{f}[N-1] \right\}.$$
Show that we have
$$\forall k \in \{0, \ldots, N-1\}, \quad f[k] = f_0[\eta k].$$
Deduce a fast algorithm to interpolate a function by trigonometric polynomials. We can see this algorithm in action in figure 3.10. f

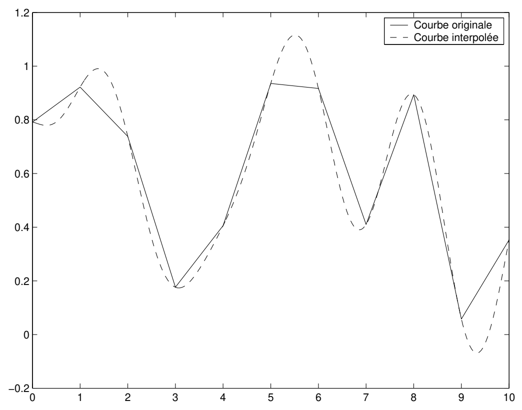

Figure 3.10: Trigonometric interpolation

Exercise 28 (Chebyshev interpolation)**.** *We define the polynomials of Chebyshev by*

$$T_k(X) \overset{\text{def.}}{=} \cos(k \arccos(X)).$$

The figure 3.11 shows the graphical representations of the polynomials T_k for small values of k. These are special cases of Lissajou figures (which are used to study wave phenomena), i.e. parameterized curves of the type $(x = a\cos(kt + c),\, y = b\cos(t))$. We consider a continuous function $f : [-1, 1] \to \mathbb{R}$. We want to

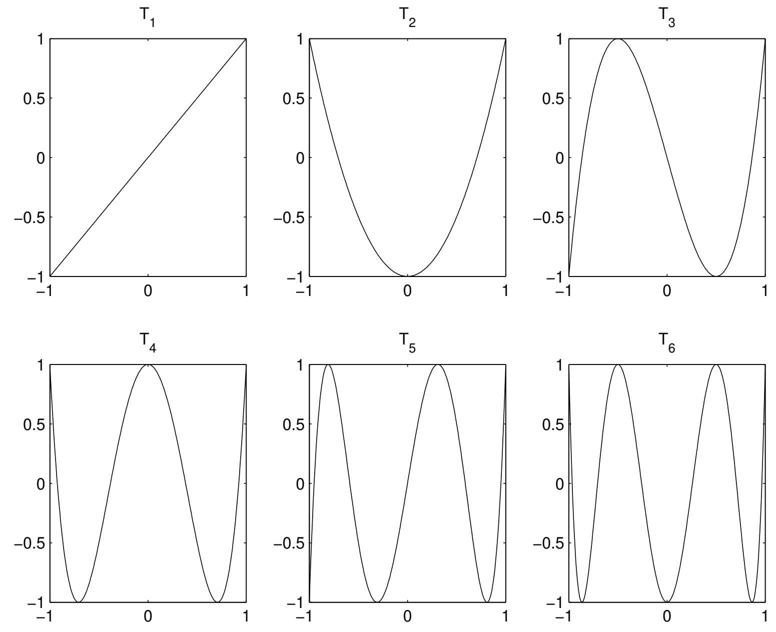

Figure 3.11: Polynomials T_k for $k = 1, \ldots, 6$

interpolate it in N points $\{x_k\}_{k=0}^{N-1}$ by a polynomial P_{N-1} of degree $N-1$, where the x_k are defined by

$$\forall k \in \{0, \ldots, N-1\}, \quad x_k \overset{\text{def.}}{=} \cos\left((k + 1/2)\frac{\pi}{N}\right).$$

1. Show that T_N is indeed a polynomial, by determining a recurrence relation between T_k and T_{k-1}. Show that the roots of T_N are x_k, for $k \in \{0, \dots, N-1\}$.

2. Show that P_{N-1} can be put in the form

$$P_{N-1} = \sum_{k=0}^{N-1} \alpha_k T_k.$$

3. We consider two types of discrete cosine transforms (often denoted DCT-2 and DCT-3 in English, because there are d'others) of a sample $\{f[k]\}_{k=0}^{N-1} \in \mathbb{R}^N$:

$$\mathcal{C}_2(f)[j] \overset{\text{def.}}{=} \sum_{k=0}^{N-1} f[k] \cos\left((k+1/2)\frac{j\pi}{N}\right)$$
$$\mathcal{C}_3(f)[j] \overset{\text{def.}}{=} \frac{1}{2}f[0] + \sum_{k=1}^{N-1} f[k] \cos\left((j+1/2)\frac{k\pi}{N}\right).$$

Show that the inverse of $\mathcal{C}_2$ is $\frac{2}{N}\mathcal{C}_3$. Implement in Matlab these transforms using a DFT of size $4N$ (we can think of making the signal even, then inserting 0 at odd indices). For more information on the cosine transform, we can consult the article of Strang[62]. Note that it is the transform $\mathcal{C}_2$ which is used for the compression of JPEG images.

4. How to calculate the $\{\alpha_k\}_{k=0}^{N-1}$ from $\{f[k] = f(x_k)\}_{k=0}^{N-1}$? Conclude a fast polynomial interpolation algorithm at the points x_k.

The figure 3.12 shows a comparison between the Lagrange interpolation (equidistant points) and the Chebyshev interpolation on a seemingly innocuous function:

$$f(x) \overset{\text{def.}}{=} \frac{1}{\alpha^2 + x^2} \quad \text{for} \quad \alpha \in \mathbb{R}_+^*.$$

For the figure, we have taken $N = 11$, and $\alpha = 0.3$. Try, experimentally, to determine the value α_0 from which the Lagrange polynomial uniformly converges to f when $n \to +\infty$. Chebyshev interpolation is a simple case

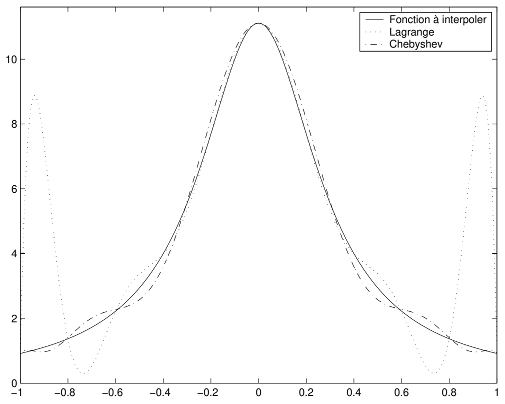

Figure 3.12: Lagrange and Chebyshev interpolation

of spectral method. These methods use decompositions according to orthogonal polynomials to approximate the solutions of partial differential equations. This is an extension of Fourier series decompositions adapted to non-periodic functions. This is all very well described in Boyd[9].

Exercise 29 (Fractional derivation)**.** *Let $f : \mathbb{R} \to \mathbb{R}$ be a function of class $\mathbb{C}^\infty$ rapidly decreasing to infinity. Show that the Fourier transform (defined by the equation (4.1)) of $f^{(n)}$ is*

$$\mathcal{F}(f^{(n)})(\xi) = (-\imath\xi)^{-n}\mathcal{F}(f)(\xi).$$

Explain how this property allows us to define a fractional derivation, ie we can define a derivation for real values of n. Implement a Matlab routine which performs an approximate fractional derivative calculation using the FFT algorithm. The figure 3.13 shows the fractional derivative of a Gaussian obtained by this method, and this for different values of n between 0 and 2.

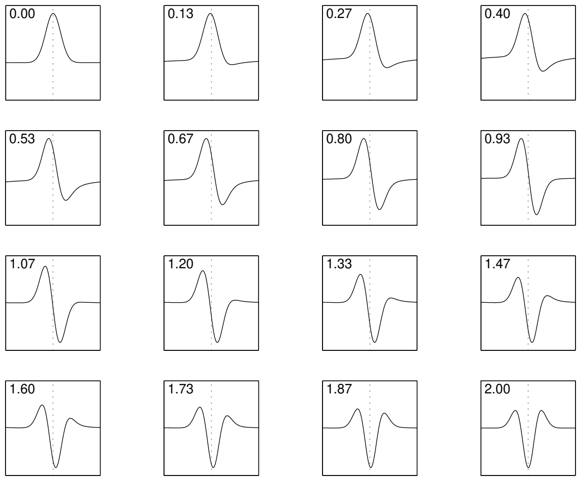

Figure 3.13: Successive fractional derivatives of a Gaussian

Exercise 30 (Intermediate Fourier transform)**.** *We recall that we Note Ω_N the Fourier matrix, which is defined by the equation (3.19). It is a self-joined matrix, so like any normal endomorphism (i.e. which commutes with its adjunct), it diagonalizes to the orthonormal basis of $\mathbb{C}^N$ (which is false in $\mathbb{R}^N$). This means that there exists a unitary P matrix and a diagonal D matrix such that*

$$\Omega_N = PDP^*.$$

1. *What are the entries of D? Check this with Matlab, by using the command* `eig` *which provides the eigenvalues as well as a decomposition according to the eigenvectors. It will be noticed that as the number of distinct eigenvalues is lower than N, the choice of the orthonormal basis of eigenvectors is totally arbitrary.*

2. *We define the matrix Ω_N^α, for $\alpha \in \mathbb{R}$, by*

   $$\Omega_N^\alpha \overset{\text{def.}}{=} PD^\alpha P^*,$$

   where D^α is one power α^{th} of D. We then define intermediate Fourier transforms:

   $$\forall f \in \mathbb{C}^N, \quad \mathcal{F}^\alpha(f) \overset{\text{def.}}{=} \Omega_N^\alpha f.$$

   Show that we have

   $$\forall(\alpha,\beta) \in \mathbb{R}^2, \quad \mathcal{F}^\alpha \circ \mathcal{F}^\beta = \mathcal{F}^{\alpha+\beta} \quad \text{and} \quad \mathcal{F}^1 = \mathcal{F}.$$

3. *The figure 3.14 shows the modulus of the matrix $\mathcal{F}^\alpha$ for a parameter α varying between 0.3 and 1. What do the two white diagonals that we can distinguish represent (we can use the calculation of the matrix Ω_N^2)?*

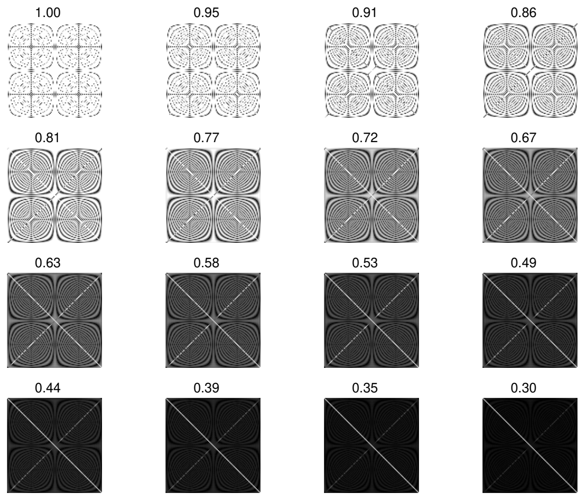

Figure 3.14: Modulus of different partial Fourier transform matrices

4. *Explain why we can construct an infinite number of intermediate transforms. By letting Matlab decide on a factorization of Ω_N, implement the obtained transform, then test it with different values of α and different signals.*

Figure 3.15 shows a panel of intermediate transforms for a Gaussian. The α transformation parameter varies between 0 and 2. Of course, for $\alpha = 2$, we find the original signal (because the Gaussian is symmetrical).

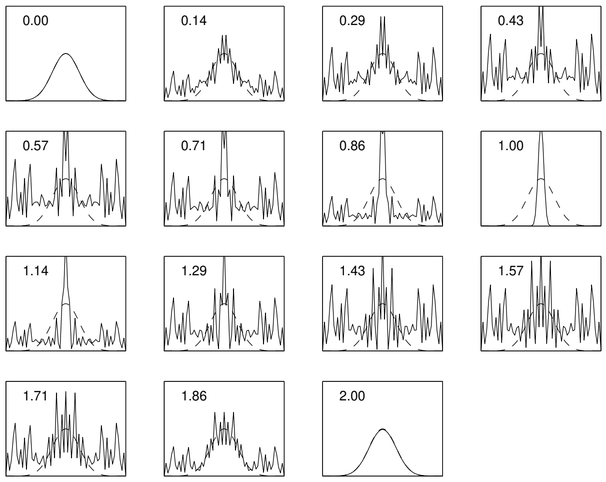

Figure 3.15: Successive partial Fourier transforms of a Gaussian

Exercise 31 (Diagonalization of the TFD)**.** *In the previous exercise, we used Matlab to diagonalize the matrix of the TFD into orthonormal basis. The theoretical construction of an orthonormal basis is not simple, mainly because the eigenvalues have a multiplicity greater than 1, which leaves a potentially infinite choice of decompositions. The goal is therefore to build a canonical process to determine a basis for diagonalization. This exercise is inspired by the article by Dickinson*[26]*. We can also read the article of Candan*[13] *which makes the relation between the matrix S and the discretization of a differential equation.*

1. *We define a matrix $S \in M_N(\mathbb{R})$ as follows:*

$$S \overset{\text{def.}}{=} \begin{pmatrix} C_0 & 1 & 0 & \dots & 1 \\ 1 & C_1 & 1 & \dots & 0 \\ 0 & 1 & C_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \dots & C_{N-1} \end{pmatrix} \quad \textit{where} \quad C_k \overset{\text{def.}}{=} 2\left(\cos\left(\frac{2k\pi}{N}\right) - 2\right).$$

   Explain why S diagonalizes in orthonormal basis.

2. *Show that S and Ω_N, the Fourier matrix, commute, that is, $S\Omega_N = \Omega_N S$. We can decompose S into $S = \Gamma + \Delta$, where Γ is a circulating matrix cleverly chosen so that $\Omega_N \Gamma = \Delta \Omega_N$.*

3. *Show that if f and g are two diagonalizable endomorphisms of $\mathbb{C}^N$ which commute, then there exists a common basis of diagonalization.*

4. *We want to show that the eigenvalues of S are distinct. Let P be the unit endomorphism matrix of $\mathbb{C}^N$ which sends an element f of $\mathbb{C}^N$ to*

$$\forall n \in \{1, \dots, \lfloor (N-1)/2 \rfloor\}, \quad Pf[n] \overset{\text{def.}}{=} \frac{f[n] + f[-n]}{\sqrt{2}},$$
$$\forall n \in \{\lceil (N+1)/2 \rceil, \dots, N-1\}, \quad Pf[n] \overset{\text{def.}}{=} \frac{f[n] - f[-n]}{\sqrt{2}}.$$

   and $Pf[0] = f[0]$. In the case where N is even, we must also add $Pf[N/2] = f[N/2]$. Show that this operator is symmetric and orthogonal, and that it corresponds to the decomposition of f into its symmetric and antisymmetric parts. Then show that PSP^{-1} is a symmetric tridiagonal matrix.

5. *Show that the eigenvalues of a symmetric tri-diagonal matrix with non-zero diagonal elements are distinct. We can use the book of Ciarlet[16] which describes the Givens-Householder method to calculate the eigenvalues of a symmetric matrix. Conclude that the eigenvalues of S are quite distinct.*

6. *Deduce that we have thus built in a canonical way an orthonormal basis of eigenvectors of Ω_N.*

In the figure 3.16 we can see the modulus of the first eigenvectors of the DFT (i.e. those which have the least sign changes) constructed using the method just described. In the figure 3.17 we can see the matrix of the moduli of the eigenvectors of the DFT (the large coefficients are black).

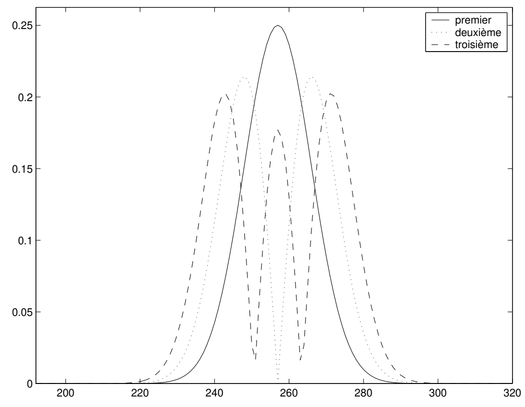

Figure 3.16: Modules of some eigenvectors of Ω_N

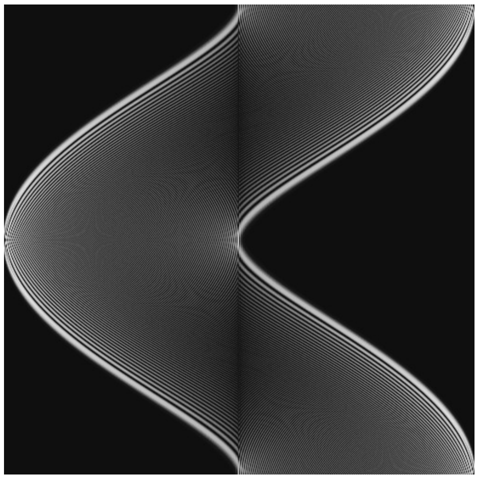

Figure 3.17: Matrix of moduli of orthogonal eigenvectors

Exercise 32 (Orthogonalization on a cyclic group)**.** *This exercise studies in a particular case the notion of orthogonalization introduced in the exercise 8. It is however independent. We consider the finite group $G = \mathbb{Z}/n\mathbb{Z}$, as well as the vector space $\mathbb{C}[G]$ of functions from G in $\mathbb{C}$. For $f \in \mathbb{C}[G]$ and $k \in G$, we define two actions of G on $\mathbb{C}[G]$ by setting*

$$k\top f : x \mapsto f(xk) \qquad \text{and} \qquad k\perp f : x \mapsto \omega^{-kx} f(x),$$

where we noted $\omega_n \overset{\text{def.}}{=} e^{\frac{2\imath\pi}{n}}$.

1. *Show that the operations $\top$ and $\perp$ are related by*

$$\mathcal{F}(k\top f) = k\perp\mathcal{F}(f) \qquad \text{and} \qquad \mathcal{F}(k\perp f) = k\top\mathcal{F}(f).$$

2. *We recall that f is said to be orthonormal under the action of $\top$ if $\{k\top f\}_{k\in G}$ is an orthonormal basis of $\mathbb{C}[G]$. Using the previous question, explain how the orthonormal bases for $\top$ and the orthonormal bases for $\perp$ are related.*

3. *Show that f is orthonormal for $\top$ if and only if $\forall k \in G,\ |\hat{f}[k]| = 1$.*

4. *Let $f \in \mathbb{C}[G]$ be such that $\hat{f}$ does not vanish. We then define $f_0 \in \mathbb{C}[G]$ by*

$$\forall k \in G, \quad \hat{f}_0[k] = \frac{\hat{f}[k]}{|\hat{f}[k]|}.$$

   Show that f_0 is orthonormal for $\top$. Suggest a similar construction for $\perp$.

5. *We now assume that g is orthonormal for $\top$. Let $\varphi \in \mathbb{C}[G]$ be any. We denote, for $k \in G$, $\mathcal{G}(\varphi)[k] \overset{\text{def.}}{=} \langle \varphi,\, k\top g\rangle$ the decomposition coefficients of φ in the orthonormal basis $\{k\top g\}_{k\in G}$. Show that $\mathcal{G}(\varphi) = \frac{1}{n} f * \tilde{g} \overset{\text{def.}}{=} \frac{1}{n}\operatorname{Corr}(\varphi,\, g)$, where $\tilde{g}[k] \overset{\text{def.}}{=} \overline{g[-k]}$, and Corr is by definition the correlation of two vectors (see also the exercise 39 for the correlation of two images). Deduce a fast algorithm for calculating $\mathcal{G}(\varphi)$ in $O(n\log(n))$ operations.*

Figure 3.18 shows two examples of orthogonalization. The top function, which is closer to orthogonality than the bottom one (we see it on the moduli of Fourier coefficients which are far from 1), gives rise to a less oscillating function g. Intuitively, to orthogonalize any function, we need the "to oscillate".

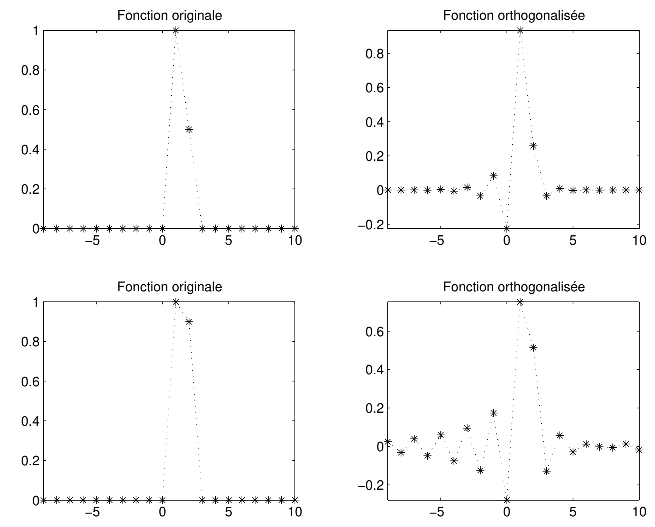

Figure 3.18: Examples of orthogonalization

Chapter 4

Applications of the discrete Fourier transform

We therefore saw, in the previous chapter, that we have an efficient algorithm, the FFT algorithm, to calculate the discrete Fourier transform. From then on, all applications using Fourier theory from near or far will be able to benefit from this algorithmic "find". But this phenomenon goes even further, since many other problems, however far removed from harmonic analysis, will be solved quickly thanks to the FFT algorithm. We will thus see that we can quickly calculate products of large integers, or even approach the solution of the Poisson equation, which may seem somewhat disconnected from the concerns we had until then!

4.1 Link with the Fourier transform on $\mathbb{R}$

This chapter is above all useful to better understand intuitively the discrete Fourier transform, thanks to many analogies with the continuous Fourier transform. It is not there to give the DFT a numerical nature, because it is above all necessary to conceive the discrete transform as an algebraic transformation, with an exact reconstruction formula (the inverse discrete Fourier transform). However, it is true that the FFT algorithm is often used to calculate Fourier coefficients in an approximate way, even if we will quickly see that the corresponding quadrature formula is not very precise.

4.1.1 Continuous Fourier transform

We have just defined, in a way that we could qualify as abstract, the discrete Fourier transform. We can therefore naturally wonder if the latter has any relation to the usual Fourier transform on $\mathbb{R}$. The latter, for a function $f \in L^1(\mathbb{R})$ is defined by the equation

$$\forall \xi \in \mathbb{R}, \quad \hat{f}(\xi) \overset{\text{def.}}{=} \int_{-\infty}^{+\infty} f(t) e^{-\imath \xi t} \mathrm{d}t. \tag{4.1}$$

This functional transformation has a very important meaning, particularly in the field of signal processing. If we consider that the function f represents a continuous signal which propagates in time, the Fourier transform makes it possible to pass from a temporal representation to a frequency representation. The quantity $\hat{f}(\xi)$ intuitively represents how many variations there are at the frequency ξ in f.

Moreover, we can extend by density the Fourier transform to functions $f \in L^2(\mathbb{R})$ of finite energy, i.e. such that $\int_{\mathbb{R}} |f(x)|^2 \mathrm{d}x < +\infty$. Thus, the famous formula of *Parseval*:

$$\forall f \in L^2(\mathbb{R}), \quad \|\hat{f}\|_{L^2} = 2\pi \|f\|_{L^2}$$

can be interpreted as a conservation of energy during the transition from the time domain to the frequency domain. For more details on the construction of the Fourier transform on $\mathbb{R}$, one can consult the book of Rudin [57].

As for the Fourier transform on a finite group, we also have an inversion theorem, under somewhat restrictive assumptions.

Proposition 40 (Inversion formula)**.** *When $f \in L^2$ and $\hat{f} \in L^1$, we have almost everywhere*

$$f(x) = \frac{1}{2\pi} \int_{\mathbb{R}} \hat{f}(\xi) e^{\imath x \xi} \mathrm{d}\xi. \tag{4.2}$$

In fact, we could redo a character theory on the group $(\mathbb{R}, +)$ as it was done on finite abelian groups. Here is for example the determination of the characters of the real line:

Proposition 41 (Characters from $\mathbb{R}$)**.** *We name the character of $(\mathbb{R}, +)$ the continuous morphisms of $\mathbb{R}$ in $\Gamma \overset{\text{def.}}{=} \{z \in \mathbb{C} \;:\; |z| = 1\}$. As usual, we denote by $\hat{\mathbb{R}}$ the group formed by the characters. For $\gamma \in \mathbb{R}$, either*

$$e_\gamma : \left\{ \begin{array}{ccc} \mathbb{R} & \longrightarrow & \mathbb{C}^* \\ t & \longmapsto & e^{\imath \gamma t} \end{array} \right. .$$

So we have $\hat{\mathbb{R}} = \{e_\gamma\}_{\gamma \in \mathbb{R}}$ and the application $\gamma \mapsto e_\gamma$ is an isomorphism between $\mathbb{R}$ and $\hat{\mathbb{R}}$.

Proof. The e_γ are of course elements of $\hat{\mathbb{R}}$. Let χ be a continuous morphism from $\mathbb{R}$ to Γ. We will start by showing that χ is a differentiable function. To do this, it suffices to integrate the homomorphism property in the neighborhood of 0:

$$\int_0^h \chi(s+t)\mathrm{d}t = \chi(s) \int_0^h \chi(t)\mathrm{d}t.$$

As $\chi(0) = 1$, for h small enough, we have $\int_0^h \chi(t)\mathrm{d}t \neq 0$. We therefore have, for a certain h,

$$\chi(s) = \frac{\int_s^{h+s} \chi(t)\mathrm{d}\mathrm{d}tt}{\int_0^h \chi(t)\mathrm{d}t},$$

which clearly defines a continuously differentiable function. Note $\lambda = \chi'(0)$. By factoring the rate of increase, we obtain

$$\forall s \in \mathbb{R}, \quad \chi'(s) = \lim_{t \to 0} \frac{\chi(s+t) - \chi(t)}{h} = \chi(s) \lim_{t \to 0} \frac{\chi(t) - \chi(0)}{h} = \lambda \chi(s).$$

Note that the only function f which checks $f' = \lambda f$ and $f(0) = 1$ is $s \mapsto e^{\lambda s}$. It only remains to show that $\lambda \in \imath\mathbb{R}$. Just use $\chi(s)\overline{\chi(s)} = \chi(0) = 1$. By differentiating this equality in 0, we find $\lambda + \overline{\lambda} = 0$. ☐

Thus, the inversion formula (4.2) is to be compared to the formula (13), on a finite group: we were in a way content to replace the sum finite by an integral. Similarly, we could analyze the decomposition of a function 2π -periodic in Fourier series. This time, it would be to use the characters in the circle $S^1 \overset{\text{def.}}{=} \mathbb{R}/2\pi\mathbb{Z}$, which are the functions

$$\forall n \in \mathbb{Z},\ \forall t \in \mathbb{R}, \quad e_n(t) \overset{\text{def.}}{=} e^{int}.$$

The decomposition formula of a periodic Fourier series function (under good assumptions, and specifying the direction of convergence) is once again an inversion formula, this time with a countable sum. For an introduction to Fourier series and integrals on a group, one can consult the book of Dym and MacKean [28], chapter 4.

4.1.2 Approximate calculation of the Fourier transform on $\mathbb{R}$

Before embarking on approximate calculations of integrals, it is necessary to be aware that the discrete Fourier transform is not limited to such approximations. We can partly explain the "intuitive" properties of the discrete Fourier transform by invoking approximations of the continuous Fourier transform. However, what makes the discrete Fourier transform work so well does not come from its ability to faithfully approach the continuous transform (it is even the opposite), but comes from the fact that it transposes the algebraic properties that we used for the real line (convolution, inversion, translation, ...) in the case of a finite and cyclic domain. It is therefore the algebraic properties of the DFT which make it a powerful tool in numerical analysis, and which make it possible to have simple and fast algorithms. In this paragraph, we will nevertheless explain the connections which connect, in terms of approximate calculations, the two transforms, discrete and continuous.

Of course, the correct method to approximate the continuous transform is to find the value of $\hat{f}$ at certain points. Indeed, we know in practice the signal f only in the form of a sample $\{f[n]\}_{n=0}^{N-1}$, each value $f[n]$ being measured for a value of the parameter x equal to $x_n \overset{\text{def.}}{=} n\Delta$, for $n = 0, \ldots, N-1$. Δ is the discretization step, that is to say the interval (of time if we consider a signal varying in time) between two measurements of the signal f.

In the following, in order to simplify the explanations, it is assumed that N is an even integer. It is clear that given a sample of N values of the signal f, it is futile to want to compute more than N independent values of the Fourier transform $\hat{f}$. This intuitive remark will be confirmed by the following approximate calculation:

$$\forall \xi \in \mathbb{R}, \quad \hat{f}(\xi) \approx \Delta \sum_{n=0}^{N-1} f[n] e^{-\imath \xi x_n}.$$

This approximation is obtained by using the method of the rectangles on the left to calculate in an approximate way the integral (4.1). For this approximation to have a meaning, it is of course necessary that outside the interval $[0,\, N\Delta]$ the function f is otherwise zero, at least very rapidly decreasing.

We see then that by calculating the values of $\hat{f}$ for values of the parameter ξ of the form $\xi_k \overset{\text{def.}}{=} \frac{2k\pi}{N\Delta}$ we obtain a writing particularly pleasant for the approximate calculation of $\hat{f}(\xi_k)$:

$$\hat{f}(\xi_k) \approx \Delta \sum_{n=0}^{N-1} f[n] e^{\frac{-2\imath\pi}{N} kn}. \tag{4.3}$$

As we mentioned before, it makes sense to only calculate N values of the Fourier transform, so we will apply the previous calculation to the points ξ_k for k variant in $\{-N/2+1, \ldots, N/2\}$. One might ask why we don't start at the index $k = -N/2$, but we see that we get the same result for $\xi_{-N/2}$ and for $\xi_{N/2}$ (which conforms to our idea: no need to calculate more than N coefficients). By comparing the expression (4.3) and the definition of the discrete Fourier transform (3.1), we obtain

$$\forall k \in \{0, \ldots, N-1\}, \quad \hat{f}(\xi_k) \approx \Delta \hat{f}[k], \tag{4.4}$$

where we have denoted $\hat{f}[k]$ the $k^{\text{ième}}$ input of the discrete Fourier transform of the vector $\{f[0], \ldots, f[N-1]\} \in \mathbb{C}^N$ (as defined by the equation (3.1)).

However, there is a slight trick in the (4.4) equation. Indeed, the index k varies in $\{-N/2+1, \ldots, N/2\}$, while the discrete transform vector $\{\hat{f}[0], \ldots, \hat{f}[N-1]\}$ has its indices in $\{0, \ldots, N-1\}$. It is nevertheless easy to see that we did not make an error in writing the equality (4.4), since the discrete transform vector can be seen as a periodic function of period N. We can therefore replace the negative frequencies $\{-N/2+1, \ldots, -1\}$ by the frequencies $\{N/2+1, \ldots, N-1\}$: we get a vector whose indices vary between 0 and $N-1$.

Thus, the formula (4.4) tells us that up to a factor Δ, the vector of the discrete Fourier transform $\{\hat{f}[0], \ldots, \hat{f}[N-1]\}$ represents an approximation of the Fourier transform of the signal f, but taken at very specific frequencies: the indices n between 0 and $N/2-1$ correspond to the positive frequencies between 0 and $f_c \overset{\text{def.}}{=} \frac{\pi}{\Delta}$ (excluded), the indices $n = N/2+1, \ldots, N-1$ correspond to the negative frequencies between

$-f_c$ (excluded) and 0 (excluded), while the index $N/2$ corresponds to both the frequency f_c and $-f_c$ (which is normal, since the transformed discrete signal is periodic with period N). We must therefore pay attention to the following two points.

– The transformed vector $\{\hat{f}[0], \ldots, \hat{f}[N-1]\}$ is considered as a periodic data of period N (it is the proper of the Fourier transform on an abelian group, in this case $\mathbb{Z}/N\mathbb{Z}$). This is of course not the case with the function $f : \mathbb{R} \to \mathbb{C}$ which has no reason to be periodic with period $2f_c$.
– Compared to the Fourier transform on $\mathbb{R}$ of the continuous signal f, the frequencies are arranged in the order: negative frequencies, then positive frequencies.

4.1.3 Addition of zeros

There are two basic techniques for influencing how one can calculate, using the discrete transform, different values of the continuous transform.

– We can sample the signal with more or less precision. We have seen that the more precise the sampling is (that is to say the more points we take to represent the analyzed function), the wider the spectrum of the transform. Thus, if one wishes to cover a spectrum twice as large, it suffices to divide the interpolation interval by two. Of course, this also extends the time needed to do the math.
– You can add zeros at the end of the vector. If we are satisfied with the spectrum on which we calculate the transform (more precisely the maximum and minimum frequencies that we can calculate), we may then want to calculate the transform with more precision, for example if we want to draw a curve to graphically represent the transform. In this case, the procedure is simple: it suffices to add zeros at the end of the vector, to add as many calculated intermediate frequencies.

By playing on these two parameters (interpolation precision and addition of zeros), we can calculate "custom" discrete transforms, to have a certain fixed vector size (this can be used to create filters, see Section 4.2), but also for the representation of the transform. The figure 4.1 shows the different results that can be obtained by adjusting both the number of sampling points and the addition of zeros. The functions represented are the modules of the DFT of the indicator function of [0.5, 1], sampled on [0, 1] at regular intervals. Each row uses the same number of sample points, but with more or less zeros added. The exercise 41 is instructive in this regard, since it reinvests all of this in order to create and test a low-pass filter.

4.1.4 Time / frequency duality

The approximate calculation we have just done allows us, via the use of the discrete Fourier transform, to calculate the value of the Fourier transform of a signal for certain frequencies only, precisely those of the form $\frac{2k\pi}{N\Delta}$ for $k \in \{-N/2+1, \ldots, N/2\}$. We therefore notice that the greater the precision with which the computation is carried out (that is to say the discretization step Δ is small), the more the spectrum on which the transform is calculated is spread. This is not an isolated phenomenon, and results from a very strong duality between the starting function and its Fourier transform.

The following result clearly illustrates this duality between the temporal properties of a signal (that is to say the properties of a function f) and its frequency properties (that is to say those of the transformed function $\hat{f}$).

Proposition 42. *Let f be a function of $L^2(\mathbb{R})$. Then f and $\hat{f}$ cannot simultaneously be compactly supported.*

The proof of this result is given in the exercise 33. This property has an immediate discrete analogue, which we can see by considering the discrete impulse δ_0 (the vector which is worth 1 in 0, and which is zero everywhere else). Its discrete Fourier transform has a spectrum that covers the entire interval considered, since it is the exponential function $t \mapsto e^{2\imath\pi t}$ sampled at regular time intervals.

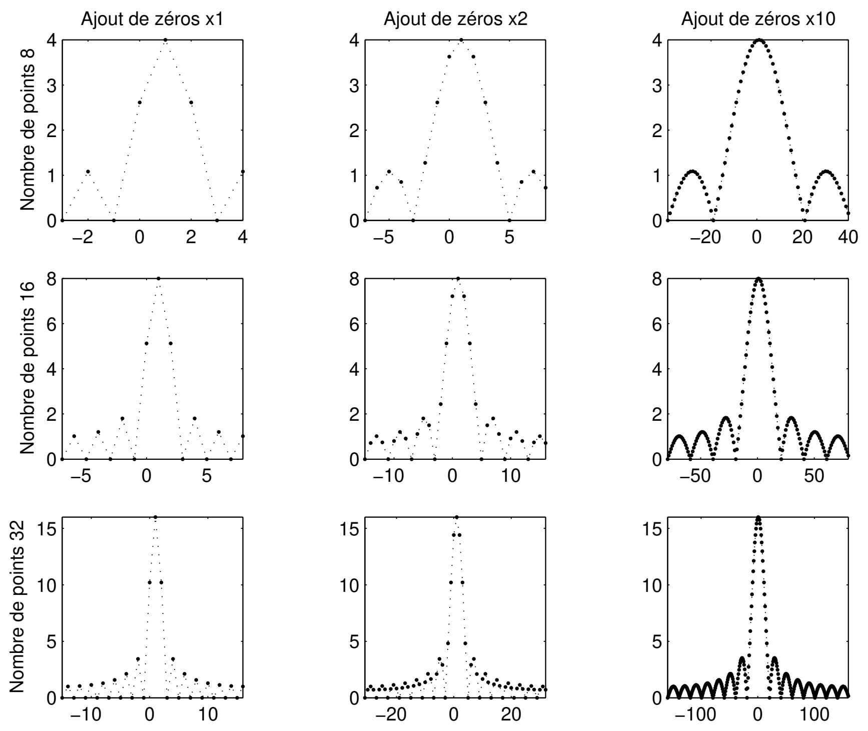

Figure 4.1: Influence of sampling and adding zeros

4.2 Filtering

The notion of filtering a sampled signal is easy to define, but is used in an intensive and varied manner. We will therefore not go around the question, however, it is interesting to link the technique of filtering with tools such as the FFT algorithm and linear convolution.

4.2.1 Linear filters

Let us start by defining the filtering of a theoretical infinite signal, before looking at the practical computation on finite signals. The filtering of a signal $f \in \mathbb{C}^{\mathbb{Z}}$ consists in pairing (acyclically of course) this signal with another signal $g \in \mathbb{C}^{\mathbb{Z}}$ fixed in advance. We thus obtain a filter operator

$$\Phi^g : \left\{ \begin{array}{ccc} \mathbb{C}^{\mathbb{Z}} & \longrightarrow & \mathbb{C}^{\mathbb{Z}} \\ f & \longmapsto & f \star g \end{array} \right. .$$

We say that g is the transfer function of the filter Φ^g.

One of the remarkable properties of linear filters is that they obtain the *impulse response* (that is, how the system represented by the filter reacts to an impulse), which is calculated very simply:

$$\Phi^g(\delta_0) = \delta_0 \star g = g,$$

where we noted δ_0 the impulse in 0, that is to say the sequence which is worth 1 in 0, and which is zero everywhere else (we speak of *discrete Dirac*). In other words, the filter transfer function is nothing more than the impulse response.

Once all these definitions have been exposed, we have to ask ourselves the question of the practical calculation of the filter. If we want to be able to calculate the convolution $f \star g$ in a finite time, a natural assumption is to impose on the impulse response g to be finite (more precisely with finite support). It is a rather radical choice, but the interest is that it will allow us to apply a linear filter to finite signals in a very simple way. It will indeed suffice to use the acyclic convolution techniques already presented in Section 3.3.3. Let us briefly recall the procedure to follow. We suppose that the signal to be filtered is of the type: $f = \{f[0], \ldots, f[N-1]\}$, and that the impulse response, it, is can be defined for negative indices, but is, in all cases, of finite size P. We must:

– add enough zeros to the two vectors so that they reach the size of $N+P-1$.
– move the negative index entries g to the end of the vector (this is customary for cyclic convolutions).
– calculate the convolution *cyclic* by FFT then inverse FFT.
– extract the indices that interest us from the result and put them back in order (if we want to retrieve the entries with negative indices).

One of the main characteristics of the convolution filters that we have just described is that they have a finite impulse response, in the sense that it becomes zero outside the support of g. This is why we often name the convolution filters *finite impulse response filters*, in English *FIR* (for **F** inite **I** mpulse **R** esponse). We will see in Paragraph 5.2.2, that we can easily build filters which do not have this property, but which can still be calculated in a finite time.

In most cases, the transfer function g has limited support in the neighborhood of 0 (that is, only inputs in the neighborhood of 0 are non-zero). A good example is the Gaussian filter, given by the equation

$$\forall k \in \{-N/2, \ldots, N/2\}, \quad g[k] = \frac{1}{M} \exp\left(-\frac{(2k/N)^2}{2t}\right).$$

The parameter M is adjusted so that $\|g\|_1 \overset{\text{def.}}{=} \sum_k |g[k]| = 1$. The parameter t represents the variance of the Gaussian. The smaller t, the more the Gaussian is "picked", so the more g tends towards the Dirac δ_0 (and the filter Φ^g tends towards the identity). On the contrary, the larger t becomes, the more the Gaussian is "spread", and the more the filter smooths the signal. We must pay attention to the fact that the indices of g have been taken from $[-N/2, N/2]$, because we want to realize an acyclic filter. To perform a cyclic filter and / or calculate the filter by FFT, it is of course necessary to report the negative frequencies *at the end* of the vector.

Figure 4.2 shows different Gaussian kernels (top). The left kernel has a variance t so low that it is equal to the Dirac. The bottom line shows the filtering of an irregular signal (shown on the left, since filtering by a Dirac has no effect). We see that the higher the variance, the more regular the filtered signal. This joins the classical theory of continuous convolution: when we filter by a kernel of class $\mathcal{C}^\infty$ (for example a Gaussian), the function instantly becomes of class $\mathcal{C}^\infty$, therefore extremely smooth!

4.2.2 Response types and stability

We have therefore seen that the impulse response defines a filter. However, we can also characterize this filter by other types of responses, including:

– *the frequency response.* It is simply the Fourier transform of the transfer function, i.e. $\sum_{k\in\mathbb{Z}} g[k]e^{\imath kx}$, for $x \in [-\pi, \pi]$. It is a 2π -periodic function, since it is the Fourier series associated with the coefficients $g[k]$. Since the impulse response g is finite, we can calculate it by discrete transform (FFT), by adding a lot of zeros at the end of g to have a very good precision (a quasi-continuous plot of the function, cf paragraph 4.1.3). It represents how the filter operates in the frequency domain. Indeed, by using the convolution theorem 36, we see that the frequency response indicates by what quantity the filtering will multiply the amplitude of each harmonic (i.e. each component of the transformed vector) of the original signal.
– *Step Response.* This is the vector obtained by filtering a step, that is to say the sequence which is zero for negative indices, and which is worth 1 for indices equal to or greater than 0. This response indicates how the filter will react to a discontinuity. For a normally constituted human mind, this is the answer that makes the most sense, since the human eye is above all trained to spot discontinuities. Thus, by observing the step response of a 2D filter, we will have indications on how the filter will transform the contours of the image (where there is a strong variation in intensity).

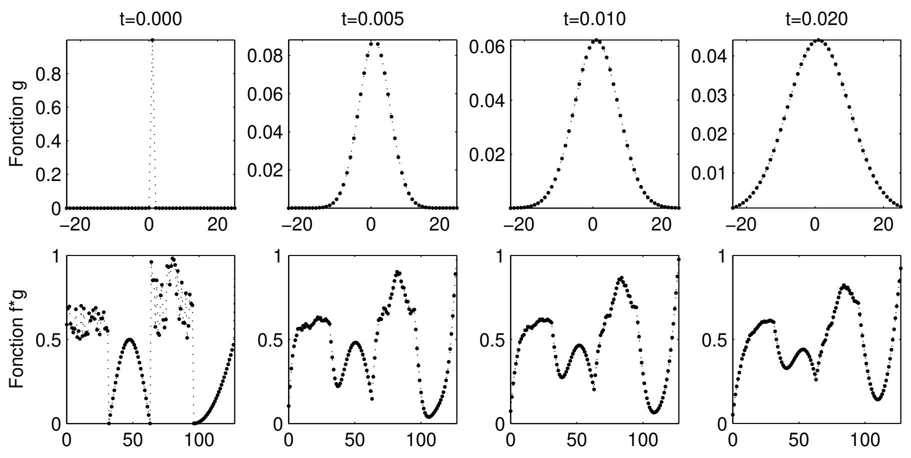

Figure 4.2: Smoothing by a Gaussian

In a pragmatic way, the frequency response is calculated very simply by using the FFT algorithm (by taking care to add enough zeros to obtain sufficient precision, and by putting the negative frequencies back in their place) . For the step response, there are at least two ways to proceed.

– We can give a step at the input of the filter. For linear filtering, it suffices to give as input a constant vector equal to 1, the algorithm being supposed to add enough zeros to avoid circular convolution.
– If we know the impulse response y_0 and we want to calculate the step response y_1, it suffices to notice that the step function is the discrete primitive (that is to say the partial sum), and therefore using the linearity of the filter, it will be the same for both responses. We thus obtain the simple formula

$$\forall n \geq 0, \quad y_1[n] = \sum_{k=-\infty}^{n} y_0[k].$$

Figure 4.3 shows the three different response types for 3 different transfer functions. The first line represents a Gaussian, and we can see that the impulse response is also a Gaussian (which is logical, since the transform of a Gaussian is Gaussian, see the lemma 8). The other two lines show filters with a slower decay, and there are some oscillations in the frequency response. Note that filters are not causal, i.e. transfer functions are defined for negative indices.

Remark 33. (**Time and frequency domain**). Depending on the answer that interests us, we can see a filter as operating in the time domain (even if the signal is not sampled in time but for example in space) or in the frequency domain. In certain applications, it is natural to create the filter according to the temporal properties which one wants to give to the filter (for example to smooth images in a privileged direction). In other applications, we will be interested in the frequency behavior of the filter (how the filter will remove high frequencies, for a low pass filter at the output of a microphone). The remarks on the time / frequency duality made in Paragraph 4.1.4 explain that we cannot have it both ways. For example, if one wishes to create a very precise *bandpass* filter (that is to say with the most compact support possible), it will necessarily be of little use in the time domain, because it will have a very wide impulse response.

Before moving on to the study of two-dimensional filters, let us briefly present an important notion which will be taken up later (during the study of the transform in Z, in Paragraph 5.2). This is the notion of *stability* of a filter, which is defined in a very intuitive way.

Definition 25 (Stability). *A filter Φ^g is said to be stable if for any bounded signal $f \in \mathbb{C}^{\mathbb{Z}}$, the output $\Phi^g(\Phi)$ is also bounded.*

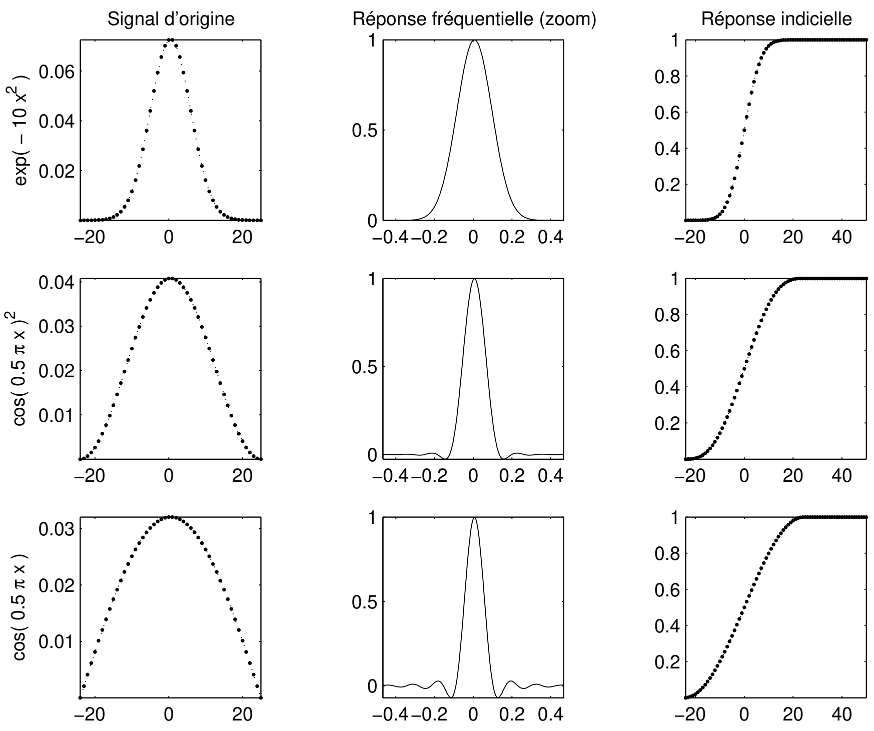

Figure 4.3: Frequency and index responses

A simple calculation shows that

$$\forall n \in \mathbb{Z}, \quad |\Phi^g(f)[n]| \leq \sup_{n\in\mathbb{Z}}(|f[n]|) \sum_{k=-\infty}^{+\infty} |g[k]|.$$

Consequently, for a filter to be stable, it suffices that $g \in \ell^1(\mathbb{Z})$, where we have noted $\ell^1(\mathbb{Z})$ the space of sequences absolutely summable. We can verify that this condition is also necessary, by taking a sequence f such that $f[k]g[-k] = |g[k]|$ (for k such as $g[k] \neq 0$). We then have, if $g \notin \ell^1(\mathbb{Z})$,

$$\Phi^g(f)[0] = \sum_{k=-\infty}^{+\infty} f[k]g[-k] = \sum_{k=-\infty}^{+\infty} |g[k]| = +\infty.$$

If $g \in \ell^1(\mathbb{Z})$, the filter operator Φ^g is a continuous endomorphism of the space of bounded sequences, and its norm is exactly $\|g\|_{\ell^1}$.

The linear filters with finite impulse response which have been considered up to now are therefore always stable. We will see in Paragraph 5.2.2, that it is possible to build filters which do not have this nice property.

4.2.3 2D filtering and image analysis

We can of course consider two-dimensional convolution as it is explained in paragraph 3.4.2. This gives rise to a filter Φ^g, which acts on two-dimensional signals $f : \{0, \ldots, N-1\} \times \{0, \ldots, N-1\} \to \mathbb{R}$. These signals can be represented as images, since in each *pixel* (i, j) of the image, we have a light intensity $f[i, j]$. More precisely, we most often restrict the end set to a finite set, for example $\{0, \ldots, 255\}$. These 256 values represent the famous *levels of gray* which will be displayed on the screen.

To “smooth” an image, we will once again consider transfer functions g tightened around 0, for example a Gaussian:

$$\forall(k,\, l) \in \{-N/2, \dots,\, N/2\}^2, \quad g[k,\, l] = \frac{1}{M} \exp\left(-\frac{(2k/N)^2 + (2l/N)^2}{2t}\right).$$

The parameter M is as in the 1D case chosen such that $\|g\|_1 = 1$, and t is adjusted according to the power of the desired smoothing. Figure 4.4 shows different Gaussian kernels (top row) as well as an image smoothed by these same Gaussian kernels (bottom row).

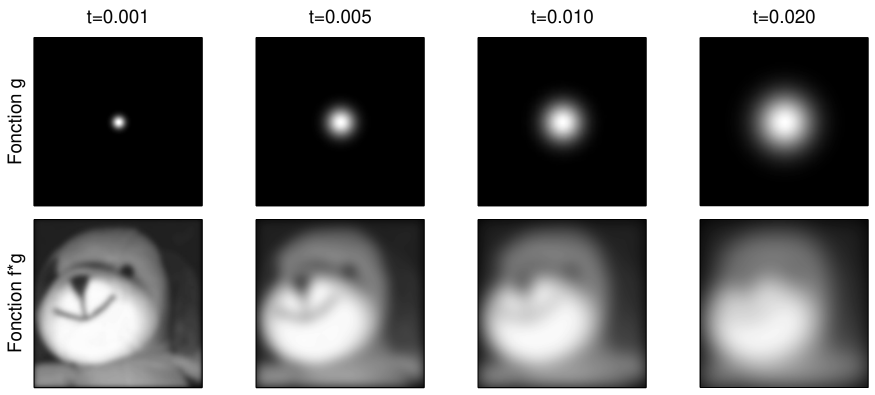

Figure 4.4: Smoothing of an image by a 2D Gaussian

A very common application of image filtering is to soften the noise present in a degraded natural image. This is shown in the figure 4.5, where we can see *Maurice* whose image, of poor quality, has been improved by a Gaussian filter.

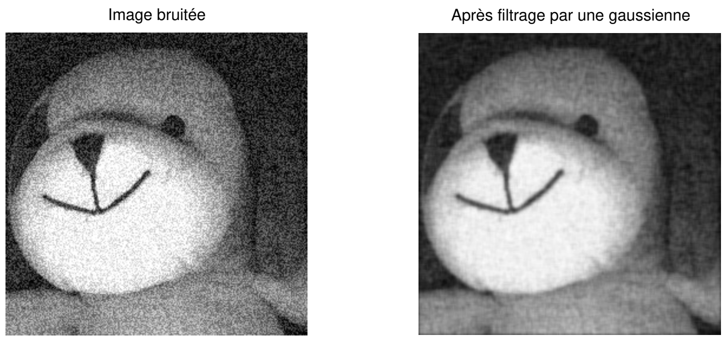

Figure 4.5: Image filtering example

The exercise 39 proposes to go further in image analysis, by applying the correlation calculation to the search for sub-images in a larger image.

4.3 Geometric aspects of filtering

In this section, we will approach the problem of filtering a signal from an original and simple angle, that of plane geometry. Rather than considering the signal studied as a series of values spaced in time, we will use it to describe a polygon drawn in the complex plane. We will then focus on the action of a filter on the shape of this polygon. More precisely, we will see how the successive iterates of the filtered polygon behave.

4.3.1 Polygon filtering

In the rest of this talk, we will consider a polygon with N vertices, Π, which we will see as a vector $\Pi \in \mathbb{C}^N$. Thus, $\Pi[0]$ will represent the first vertex of the polygon, $\Pi[1]$ the second, and so on. Equivalently, we can also consider a polygon as a function $\Pi : \mathbb{Z}/N\mathbb{Z} \to \mathbb{C}$. This description is very convenient, since it adapts well to the concept of closed polygon. Indeed, we consider that the vertex $\Pi[i]$ is linked to the vertex $\Pi[i+1]$, and we naturally want to link the vertex $\Pi[N-1]$ to the vertex $\Pi[0]$. This amounts to considering a N -periodic signal.

We are interested in the action of a circular filter Φ^g on a polygon Π. We will therefore consider the iterated polygons $\Pi^{(k)}$, for $k \geq 0$, which are defined as follows:

$$\begin{cases} \Pi^{(0)} = \Pi \\ \Pi^{(k)} = g * \Pi^{(k-1)} \quad \forall k > 0 \end{cases}, \tag{4.5}$$

where g is the filter transfer function. The natural question is to know if $\Pi^{(k)}$ will tend towards a certain limit polygon, $\Pi^{(\infty)}$, if the iterated polygons will remain bounded, or if on the contrary they will "explode". To carry out this study, it suffices to calculate the Fourier transform of the iteration relation (4.5), and we see that

$$\forall k \geq 0, \quad \hat{\Pi^{(k)}} = (\hat{g})^n \cdot \hat{\Pi}.$$

The study of the convergence of $\Pi^{(k)}$ is therefore very simple in the Fourier domain. Moreover, thanks to the inversion formula of the transform, a convergence in the Fourier domain is equivalent to a convergence of the polygon. Here are the different cases that can arise.

- If $\exists i \in \{0, \dots, N-1\}$ such that $|\hat{g}[i]| > 1$: then the iterated polygons will explode. This corresponds to cases (a) and (b) in figure 4.6.
- If for all i, we have either $|\hat{g}[i]| < 1$ or $\hat{g}[i] = 1$, then the iterated polygons will converge to a polygon $\Pi^{(\infty)}$ which is defined by

$$\forall i = 0, \dots, N-1, \quad \hat{\Pi^{(\infty)}}[i] = \begin{cases} 0 & \text{si} \quad |\hat{g}[i]| < 1 \\ \hat{\Pi}[i] & \text{si} \quad \hat{g}[i] = 1 \end{cases}.$$

  This corresponds to case (c) of figure 4.6.
- If $\forall i \in \{0, \dots, N-1\}$, $|\hat{g}[i]| \leq 1$, but there is a i such that $|\hat{g}[i]| = 1$ and $\hat{g}[i] \neq 1$, then the iterated polygons will not converge, but they will remain bounded. This corresponds to case (d) of figure 4.6. We can notice that if $\hat{g}[i]$ is a root of unity, then the motion is periodic (even if the phenomenon is difficult to study numerically because of the rounding errors).

Two typical examples of polygon filtering can be given. They are shown in figure 4.7.

Example 3. The first corresponds to the filter

$$g = \{1/2,\ 1/2,\ 0, \dots, 0\}.$$

This consists of replacing the polygon Π by the polygon $\Pi^{(1)}$ such that

$$\Pi^{(1)}[i] = \frac{1}{2}\left(\Pi[i] + \Pi[i+1]\right).$$

In a way, this amounts to joining the consecutive midpoints on each side of the polygon. Intuitively (and on the drawing of figure 4.6, on the left), one has the impression that the iterated polygons converge towards the center of the polygon. Let's confirm this by calculation:

$$\hat{g}[k] = \frac{1}{2}\left(1 + e^{-\frac{2\imath\pi}{N}k}\right) = \cos\left(\frac{k\pi}{N}\right) e^{-\frac{\imath\pi}{N}k}.$$

As we have $\hat{g}[0] = 1$ and for $k \geq 1$, $|\hat{g}[k]| < 1$, we therefore conclude that the iterated polygons will converge to the point $\hat{\Pi}[0]$, which corresponds to the center of gravity of the polygon.

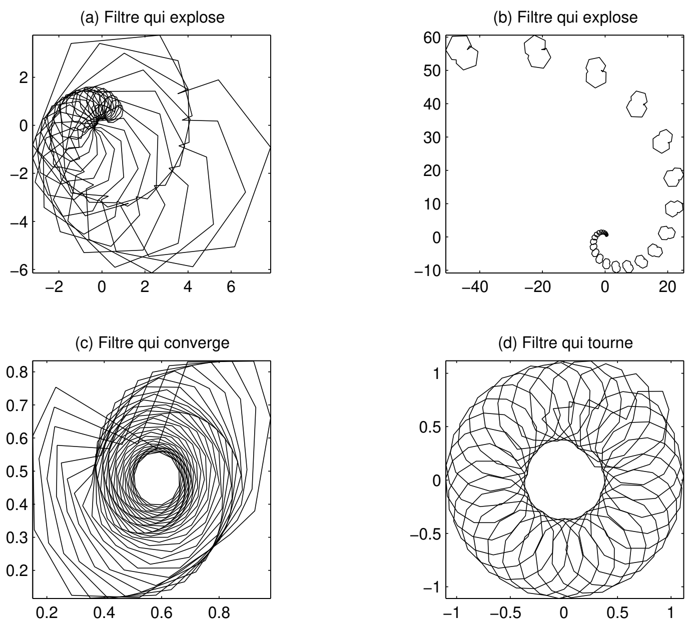

Figure 4.6: Different cases of polygon filtering

Example 4. For the second example, this is a filter which acts in the Fourier domain as follows:

$$\hat{g} = \{0.8,\, 1,\, 0.8, \ldots,\, 0.8\}.$$

The iterated polygons will therefore converge to a polygon $\Pi^{(\infty)}$ such that

$$\Pi^{\hat{(\infty)}} = \{0,\, \hat{\Pi}[1],\, 0, \ldots,\, 0\}.$$

We check that this corresponds to a regular polygon inscribed in a circle of radius $|\hat{\Pi}[1]|$, as we can see in the figure 4.7 (right).

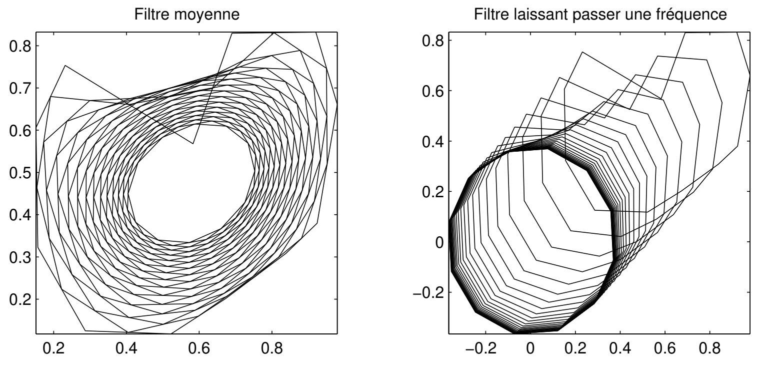

Figure 4.7: Average filtering and frequency pass filtering

4.3.2 Polygonal inequalities

This paragraph is taken from the book by Terras [65]. I wanted to insert it into a more general study of geometrical Fourier analysis, which is the subject of this paragraph. The idea is to use the Fourier transform in order to demonstrate inequalities of Euclidean nature on the polygons. The main tool will be the Plancherel equality 32, which will make it possible to demonstrate the inequalities by passing through the Fourier domain.

As in the previous paragraph, we consider a polygon Π with N sides, which can be seen as a function $\Pi : \mathbb{Z}/N\mathbb{Z} \to \mathbb{C}$. We will define several quantities related to this polygon. First, the sum of the squares of the lengths of the sides:

$$S(\Pi) \overset{\text{def.}}{=} \sum_{i=0}^{N-1} |\Pi[i+1] - \Pi[i]|^2.$$

Then, the sum of the squares of the distances to the center of gravity of the polygon:

$$T(\Pi) \overset{\text{def.}}{=} \sum_{i=0}^{N-1} |\Pi[i] - \hat{\Pi}[0]|^2.$$

Finally, the oriented area of the polygon:

$$A(\Pi) \overset{\text{def.}}{=} \frac{1}{2} \sum_{i=0}^{N-1} \mathcal{I}\left(\overline{\Pi[i]}\Pi[i+1]\right).$$

Proposition 43. *We have the following inequalities:*

(i) $A(\Pi) \leq \frac{1}{2} T(\Pi)$

(ii) $4 \sin^2\left(\frac{\pi}{N}\right) T(\Pi) \leq S(\Pi)$.

Proof. **Inequality (i)**: We introduce the offset operator T defined by the relation $T\Pi[k] = \Pi[k+1]$, which lets write

$$A(\Pi) = \frac{1}{2}\mathcal{I}\left(\sum_{k=0}^{N-1} \Pi[i]\overline{T\Pi[i]}\right).$$

Then just use Plancherel's formula to calculate $A(\Pi)$ in the Fourier domain:

$$A(\Pi) = \frac{1}{2}\mathcal{I}\left(\sum_{k=0}^{N-1} \hat{\Pi}[k]\overline{\mathcal{F}(T\Pi)[k]}\right).$$

We then use the fact that $\mathcal{F}(T\Pi)[k] = e^{\frac{2\imath\pi}{N}k}\hat{\Pi}[k]$ to get

$$A(\Pi) = \frac{1}{2} \sum_{k=0}^{N-1} \sin\left(\frac{2k\pi}{N}\right) |\hat{\Pi}[k]|^2.$$

By noting that $\sin\left(\frac{2k\pi}{N}\right) \leq 1$, we do obtain the desired inequality, after having once again used the Plancherel formula.
Inequality (ii): To do this, let's introduce the filter whose transfer function is equal to $g \overset{\text{def.}}{=} \{-1, 1, 0, \ldots, 0\}$. We can rewrite the quantity $S(\Pi)$ as follows:

$$S(\Pi) = \sum_{k=0}^{N-1} |g * \Pi[k]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |\mathcal{F}(g * \Pi)[k]|^2.$$

For the last tie, we used Plancherel's formula. Let us calculate the modulus of the Fourier transform of g:

$$\forall k = 1, \ldots, N-1, \quad |\hat{g}[k]|^2 = |e^{-\frac{2\imath\pi}{N}k} - 1|^2 = 4\sin^2\left(\frac{k\pi}{N}\right) \geq 4\sin^2\left(\frac{\pi}{N}\right).$$

It only remains to use the convolution property to get, like $\hat{g}[0] = 0$,

$$S(\Pi) = \frac{1}{N}\sum_{k=1}^{N-1} |\hat{g}[k]|^2 |\hat{\Pi}[k]|^2 \geq \frac{1}{N} 4\sin^2\left(\frac{\pi}{N}\right) \sum_{k=1}^{N-1} |\hat{\Pi}[k]|^2.$$

We then conclude by using Plancherel's formula once again:

$$\frac{1}{N}\sum_{k=1}^{N-1} |\hat{\Pi}[k]|^2 = \frac{1}{N}\sum_{k=0}^{N-1} \left|\mathcal{F}\left(\Pi[k] - \hat{\Pi}[0]\right)\right|^2 = T(\Pi).$$

Which ends the demonstration. □

4.3.3 Fourier descriptors

To end this chapter on the applications of Fourier theory to geometry, we will tackle the problem of *pattern recognition*. To be more precise, we want to know if two polygons Π_1 and Π_2 represent the same shape, up to translation, rotation and homothety. We will of course try to compare the two Fourier transforms $\hat{\Pi}_1$ and $\hat{\Pi}_2$.

It is therefore a question, from a transformed vector, of creating a quantity $\mathcal{D}(\Pi)$ characterizing a polygon Π up to translation and similarity. Here are the three operations to perform:

- for the translation: we know that only the component $\hat{\Pi}[0]$ is modified by a translation. In fact, $\hat{\Pi}[0]$ precisely represents the center of gravity of the polygon. We will therefore purely and simply ignore the first entry of the transformed vector.
- for rotation and homothety: we consider a plane similarity with center ω, of angle θ, and of ratio r. This corresponds to the transformation $z \mapsto \omega + re^{\imath\theta}(z - \omega)$. We check that this transformation changes the Fourier coefficients $\hat{\Pi}[k]$ (for $k > 0$) by multiplying them by $re^{\imath\theta}$. Suppose that $\hat{\Pi}[1] \neq 0$ (otherwise, we take another index $k_0 > 0$ such as $\hat{\Pi}[k_0] \neq 0$). To cancel the effect of the homothety, it suffices to consider the quantities $\frac{\hat{\Pi}[2]}{\hat{\Pi}[1]}, \frac{\hat{\Pi}[3]}{\hat{\Pi}[1]}, \ldots, \frac{\hat{\Pi}[N-1]}{\hat{\Pi}[1]}$.
- for the invariant by circular shift of the data: we want the polygon $\Pi' \overset{\text{def.}}{=} T\Pi$ defined by $\Pi' = \{\Pi[1], \Pi[2], \ldots, \Pi[N-1], \Pi[0]\}$ is indistinguishable from the polygon Π, which means that $\mathcal{D}(\Pi) = \mathcal{D}(\Pi')$. We see that $\hat{\Pi'}[k] = e^{\frac{2\imath\pi}{N}k}\hat{\Pi}[k]$. To have this invariance, we will therefore consider the modulus of the quantities already calculated.

After all these observations, we are therefore led to assign to each polygon Π a Fourier descriptor, which we define as follows.

Definition 26 (Fourier descriptor). *Let Π be a polygon with N sides such that $\hat{\Pi}[1] \neq 0$. Its Fourier descriptor $\mathcal{D}(\Pi)$ is a complex vector of size $N-2$ defined as follows:*

$$\mathcal{D}(\Pi) \overset{\text{def.}}{=} \left\{\frac{|\hat{\Pi}[2]|}{|\hat{\Pi}[1]|}, \frac{|\hat{\Pi}[3]|}{|\hat{\Pi}[1]|}, \ldots, \frac{|\hat{\Pi}[N-1]|}{|\hat{\Pi}[1]|}\right\}.$$

We then define the distance between two polygons (we suppose that they verify $\hat{\Pi}_i[1] \neq 0$) Π_1 and Π_2 of the following way:

$$d(\Pi_1, \Pi_2)^2 \overset{\text{def.}}{=} \sum_{k=0}^{N-3} \left(\mathcal{D}(\Pi_1)[k] - \mathcal{D}(\Pi_2)[k]\right)^2.$$

Two polygons Π_1 and Π_2 which are images of each other by a plane similarity therefore verify $d(\Pi_1, \Pi_2) = 0$. However, we should be careful that the converse is false. Indeed, if we choose arbitrary numbers $\theta_0, \ldots, \theta_{N-1}$, then the polygon Π_2 defined by $\hat{\Pi}_2[k] \overset{\text{def.}}{=} e^{\imath\theta_k}\hat{\Pi}_1[k]$ checks $d(\Pi_1, \Pi_2) = 0$. In practice however, the value of d gives a good idea of the similarity between two forms. This is illustrated in the figure 4.8, where the second polygon is close to the first.

4.4 Numerical solution of derivative equations by tials

One of the main utilities of the continuous Fourier transform is the solution of partial differential equations. From a purely formal point of view (almost completely algebraic), it allows to replace a complex problem (a linear differential equation) in a much simpler problem, a polynomial equation. This is because the Fourier transform replaces the derivation with respect to x in multiplication by $\imath x$. Of course, you have to worry about the convergence problems as well as the boundary conditions, but the Fourier transform turns out to be a very powerful theoretical tool to demonstrate the existence of solutions for many equations.

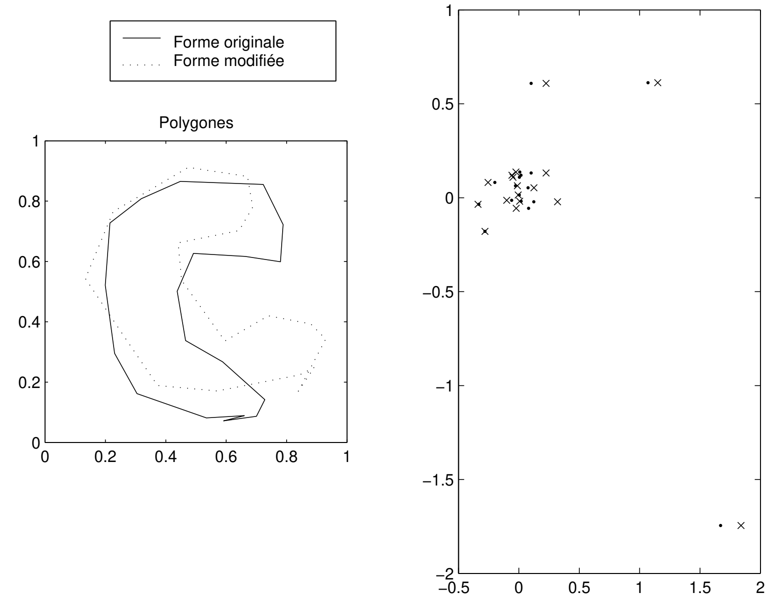

Figure 4.8: Fourier descriptors for two similar forms

From a practical point of view, the numerical computation of the solution of an equation can, at various times of the discretization process, lead to the use of the discrete Fourier transform. This can come from a pure and simple discretization of the continuous Fourier transform (for example for *the heat equation*, paragraph 4.4.2), or from a more ingenious way to simplify and speed up calculations (for example for *Poisson's equation*, paragraph 4.4.3). In the following paragraphs, we will focus on these numerical aspects; The theoretical use of the continuous transform is detailed in particular in the exercise 35.

4.4.1 Calculation of Fourier coefficients

As we saw in paragraph 4.1.2, the FFT algorithm allows to calculate in an approximate way the value of the Fourier transform continue at certain frequencies ξ_k. But in reality, the calculation of $\hat{f}[\xi_k]$ that we carried out corresponds to the approximation of the integral:

$$\hat{f}(\xi_k) \approx \Delta \int_0^{N\Delta} f(t)e^{-\frac{2\imath\pi}{N\Delta}kt}\mathrm{d}t. \tag{4.6}$$

Now, if we denote by f_1 the periodic function of period $N\Delta$ which coincides with f on the interval $[0,\, N\Delta]$, the equation (4.6) corresponds to the calculation of $c_k(f_1)$, the k^{ieme} Fourier coefficient of the function f_1.

Let us summarize all these results by a formula allowing to calculate in an approximate way N Fourier coefficients for a periodic function f of period 1:

$$\forall n \in \{-N/2+1, \dots, 0, \dots, N/2\}, \quad c_n(f) \overset{\text{def.}}{=} \int_0^1 f(t) e^{-2\imath\pi nt} \mathrm{d}t \approx \frac{1}{N} \hat{f}[n],$$

where we denote by $\hat{f}$ the transformed Fourier vector of the vector $\{f(k/N)\}_{k=0}^{N-1}$.

The FFT algorithm will therefore allow the calculation of N Fourier coefficients of a function sampled in N points all at once. Moreover, the techniques of adding zeros and refining the sample make it possible to modulate the number of coefficients calculated for the same sample. The only potential problem lies in the lack of precision (the rectangle method is only of order 1), which can be problematic when the Fourier coefficients decrease rapidly towards 0, as is the case for a very regular function. Two solutions are then possible:

– increase the number of interpolation points.

– use a higher integration method. It is possible to show a discrete Fourier transform with formulas other than that of rectangles (for example that of *Simpson*). The exercise 54, question 2, details this technique.

4.4.2 Application to the heat equation

In this paragraph, we will apply the method described in the previous paragraph, which allows us to calculate all at once a very large number of Fourier coefficients (admittedly with questionable precision). It is a question of solving the heat equation, which historically had a very important role, since it was this which pushed Joseph Fourier to develop his theory, in his article *Theorie Analytique de the Heat* (1822).

We want to solve in an approximate way the equation of heat on the circle $S^1 \overset{\text{def.}}{=} \mathbb{R}/\mathbb{Z}$:

$$\begin{cases} \forall (t,\, x) \in \mathbb{R}_*^+ \times S^1, & \dfrac{\partial u}{\partial t} = \kappa \dfrac{\partial^2 u}{\partial x^2} \\ \forall x \in S^1, & u(0,\, x) = f(x) \end{cases}, \tag{4.7}$$

where the sought solution u is assumed to be sufficiently regular over $\mathbb{R}_*^+ \times S^1$, and continues over $\mathbb{R}^+ \times S^1$. In this paragraph, we are not going to use a finite difference method, unlike what we will do in the 4.4.3 paragraph to solve the Poisson equation. The exercise 34 studies the stability of such a method for the heat equation. Instead, we will explicitly solve the continuous equation, and calculate approximations of the solution by FFT.

This equation reflects the evolution of heat in a circle of length 1, completely isolated, and of which we know the initial temperature distribution. The constant κ translates the conductivity of the material, and without loss of generality, it will be taken equal to $\frac{1}{2}$ in the following. Indeed, we can replace the function u by $(t,\, x) \mapsto u\left(\frac{t}{2\kappa},\, x\right)$, which does not modify the problem. For an exposition on the different applications of series and the Fourier transform to differential equations (and in particular to the heat equation), the book of Dym and MacKean [28] is an excellent source. In the following, we will content ourselves with stating the main results. In particular, the uniqueness result is detailed in the exercise 35.

To begin with, we are looking for a formal solution in the form

$$u(t,\, x) \overset{\text{def.}}{=} \sum_{n\in\mathbb{Z}} c_n(t) e_n(x) \qquad \text{with} \quad e_n(x) \overset{\text{def.}}{=} e^{2\imath\pi nx},$$

since intuitively, the solution must be periodic at t fixed. The coefficients $c_n(t)$ are defined by

$$\forall t > 0, \quad c_n(t) \overset{\text{def.}}{=} \int_0^1 u(t,\, x) e_n(-x) \mathrm{d}x.$$

By doing two integrations by parts, we obtain a differential equation verified by c_n:

$$\begin{aligned}\frac{dc_n}{dt}(t) = \int_0^1 \frac{\partial u}{\partial t}(t,\, x)e_n(-x)\mathrm{d}x &= \int_0^1 \frac{1}{2}\frac{\partial^2 u}{\partial x^2}(t,\, x)e_n(-x)\mathrm{d}x \\ &= -\,2\pi^2 n^2 \int_0^1 u(t,\, x)e_n(-x)\mathrm{d}x = -2\pi^2 n^2 c_n(t).\end{aligned}$$

As $c_n(0) = \hat{f}(n)$ (the n^{th} Fourier coefficient of f), we obtain a formal solution of the heat equation (4.7):

$$\forall (t,\, x) \in \mathbb{R}_*^+ \times S^1, \quad u(t,\, x) = \sum_{n\in\mathbb{Z}} \hat{f}(n)\exp(-2\pi^2 n^2 t)e_n(x). \tag{4.8}$$

The fact that the function u thus defined is indeed the solution of the problem posed for $t > 0$ comes from the fact that we can differentiate under the sum sign because the series of terms $\exp(-2\pi^2 n^2 t)$ is normally convergent for $t \geq \epsilon > 0$, as well as all its derivatives with respect to t. The only thing difficult to show is that we have

$$\|u(t,\, \cdot) - f\|_\infty \overset{t\to 0}{\longrightarrow} 0,$$

that is to say that the initial conditions are well respected. We recall that $\|\cdot\|_\infty$ denotes the uniform norm over S^1. All this is detailed in the exercise 35 at the same time as the proof of the uniqueness of the solution.

Remark 34. (**Continuous filtering**). Intuitively, the passage from $u(0,\, \cdot) = f$ to $u(t,\, \cdot)$ corresponds to the multiplication of the coefficients of Fourier $\hat{f}(n)$ of f by $\exp(-2\pi^2 n^2 t)$. This amounts to filtering f by a Gaussian, that is to say to smooth the starting function. The larger t, the greater the variance of the Gaussian, and therefore the more pronounced the blurring effect induced by the filtering. Ultimately, when $t \to +\infty$, the filtering simply corresponds to averaging the function, so the heat distribution is uniform.

In order to approximate the solution u of the heat equation, we will, for some fairly large N fixed (which we assume to be a power of 2), calculate

$$u_N(t,\, x) \overset{\text{def.}}{=} \sum_{n=-N/2}^{N/2-1} c_n(t)e_n(x).$$

Of course, we will use the technique developed in paragraph 4.4.1 and therefore sample the function f according to a vector $\tilde{f} \overset{\text{def.}}{=} \{f(k/n)\}_{k=0}^{N-1}$. The calculation of the FFT of this vector allows us, up to a factor $\frac{1}{N}$, to calculate approximately N Fourier coefficients of the function f, and therefore to construct the function u_N. All this is detailed in the Matlab programs presented in the B.6 paragraph. The only technical difficulty is that the Fourier coefficients calculated by the FFT are not arranged in the right order, but via the indexing $\{0, \dots,\, N/2-1,\, -N/2, \dots,\, -1\}$. We can see the evolution of the solution over time in the figure 4.9, where the initial data is an indicator function (therefore discontinuous). We can clearly see the regularizing effect of the heat equation: for $t > 0$, the solution becomes smooth, and tends towards a constant function when t tends towards $+\infty$.

Remark 35. (**Discrete filtering**). Solving the heat equation by DFT therefore amounts to performing a discrete filtering with an increasingly strong low-pass filter. In essence, filtering through a symmetrical low pass filter amounts to solving the heat equation for a certain time t. The more the filter is regularizing, the greater t.

4.4.3 Solving the Poisson equation by finite differences

We want to find a function $u : [0,\, 1] \times [0,\, 1] \to \mathbb{R}$ sufficiently regular (of class $\mathcal{C}^2$) which satisfies the equation of *Poisson*:

$$\begin{cases} \forall (x,\, y) \in [0,\, 1] \times [0,\, 1], \quad \Delta u(x,\, y) \overset{\text{def.}}{=} \dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2} = f(x,\, y) \\ \forall x \in [0,\, 1], \quad u(x,\, 0) = u_{x0}(x) \quad \text{and} \quad u(x,\, 1) = u_{x1}(x) \\ \forall y \in [0,\, 1], \quad u(0,\, y) = u_{0y}(y) \quad \text{and} \quad u(1,\, y) = u_{1y}(y) \end{cases} \tag{4.9}$$

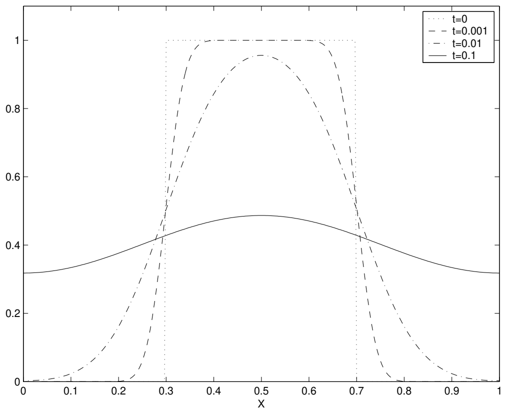

Figure 4.9: Evolution of the solution of the heat equation

where $f : \mathbb{R}^2 \to \mathbb{R}$ is a continuous function known in advance, and the functions u_{x0}, u_{x1}, u_{0y} and u_{1y} are also supposedly known continuous functions (we fix the values of the solution on the edges).

The equation of *Poisson* has very important interpretations, especially in physics. We can note:

- *equation of an elastic membrane*: the surface of the membrane is represented by the equation $z = u(x, y)$. The function f represents the surface amount of forces applied vertically to the surface. Imposing the value of the function u on the edges corresponds to fixing the edges of the surface to a reinforcement.
- *equation of an electric potential*: the quantity $u(x, y)$ represents the value of a surface electric potential at a point (x, y), and the function f takes into account a surface distribution of electric charges.

In the particular case where the function f is zero , we speak of the equation of *Laplace*, and the function u is then called *harmonic function*. Obviously, we show that the real part of a holomorphic function is harmonic. We can even show that locally, the converse is true (and therefore a harmonic function has partial derivatives at any order!). For more details on the theory of harmonic functions, one can consult the book of Rudin [57, p.275].

We propose to approach the solution u of the equation (4.9) by the method of *finite differences*. To do this, we will discretize the square $[0, 1] \times [0, 1]$ according to $N + 1$ points on both directions. We therefore seek an approximate solution $\{U(i, j)\}_{0 \leq i, j \leq N}$, where $U(i, j)$ represents an approximation of $u(ih, jh)$ (we noted $h \overset{\text{def.}}{=} \frac{1}{N}$ the step of the subdivision). By replacing the Laplacian Δu by a discrete approximation, we obtain, for the interior squared points, the following equations, for i and j in $\{1, \ldots, N-1\}$,

$$\frac{1}{h^2}\left\{U(i-1, j) + U(i+1, j) + U(i, j+1) + U(i, j-1) - 4U(i, j)\right\} = F(i, j), \tag{4.10}$$

where we denote by $F(i, j) \overset{\text{def.}}{=} f(ih, jh)$ the right side of the equation. We have of course paid attention to the boundary terms $(i, j = 0, N)$, which are not part of the system since they are fixed once and for all by the boundary conditions.

We would thus be tempted to write the equation (4.10) as a convolution by a filter Φ:

$$U * \Phi = F$$

where $*$ denotes the 2D circular convolution operator, defined in paragraph 3.4.2, and the transfer function is written

$$\Phi \overset{\text{def.}}{=} \frac{1}{h^2}\begin{pmatrix} 1 & & & & \\ & & & \vdots & \\ & & \dots & 0 & \dots \\ & & & \vdots & \\ 1 & & & & \\ -4 & 1 & & & 1 \end{pmatrix}, \tag{4.11}$$

(it should be remembered that the functions considered are N periodic, so that the negative frequencies are pushed to the other end of the table). However, there are at least two objections to this writing.

- The filters operate on a periodic signal, but this is not the case here: the values of the edges have no reason to “re-stick”. Moreover, the equation (4.10) is only valid inside the domain, that is to say for $0 < i,\, j < N$. It does not describe a circular convolution.
- The edge values are taken into account in the filtering, so they are part of the unknowns: on the contrary, we want them to be fixed.

To work around this problem, it suffices to make null U at the ends, that is to say for $i,\, j = 0,\, N$, quite simply by passing in the right-hand side the edge terms. Here is for example an equation obtained for $i = 1$ and $1 < j < N - 1$:

$$\begin{aligned}\frac{1}{h^2}\left\{0 + U(2,\, j) + U(1,\, j+1) + U(1,\, j-1) - 4U(1,\, j)\right\} &= F(i,\, j) - \frac{1}{h^2}u_{0y}(jh)\\ &\overset{\text{def.}}{=} \tilde{F}(i,\, j),\end{aligned}$$

(we used the fact that $U(0,\, j) = u_{0y}(jh)$ the value fixed on the edge $x = 0$). By doing the same for the four edges of the matrix F, we create a new matrix $\tilde{F}$ zero on the edges, and thanks to this manipulation, we can replace the unknown U by an unknown $\tilde{U}$ which is zero on the edges. It remains precisely to solve the problem of the application of the filter on the edges (the equation (4.10)) is only valid inside). To be able to do this, it suffices to extend the functions considered by imparity along the two axes. Indeed, the nullity of the function on the edges as well as the equation (4.10) of the filter shows that the terms symmetrical with respect to an edge must be of opposite signs. For example, we have the equation

$$U(1,\, j) + U(-1,\, j) = \tilde{F}(0,\, j) = 0,$$

hence $U(1,\, j) = -U(-1,\, j)$. We therefore extend the matrices $\tilde{U}$ and $\tilde{F}$ to obtain matrices (always noted in the same way) of size $2N$ odd. Thus, in the (simplistic, but instructive) case where $N = 3$, the matrix $\tilde{F}$ will be written

$$\tilde{F} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \tilde{F}(1,\,1) & \tilde{F}(2,\,1) & 0 & -\tilde{F}(2,\,1) & -\tilde{F}(1,\,1) \\ 0 & \tilde{F}(1,\,2) & \tilde{F}(2,\,2) & 0 & -\tilde{F}(2,\,2) & -\tilde{F}(1,\,2) \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\tilde{F}(1,\,2) & -\tilde{F}(2,\,2) & 0 & \tilde{F}(2,\,2) & \tilde{F}(1,\,2) \\ 0 & -\tilde{F}(1,\,1) & -\tilde{F}(2,\,1) & 0 & \tilde{F}(2,\,1) & \tilde{F}(1,\,1) \end{pmatrix}.$$

With all these new conventions, the equation (4.10), which extends to $0 \le i,\, j \le 2N$ and is also true as a periodic convolution, is written

$$\hat{U} * \tilde{\Phi} = \tilde{F},$$

where $\tilde{\Phi}$ is still defined as in the equation (4.11), but this time is of size $2N$.

The idea is then to take the 2D Fourier transform of the two members, since the latter transforms the convolution product into a simple product, as stated in the proposition 3.3. We get the very nice equation

$$\mathcal{F}\left(\tilde{U}\right) \cdot \mathcal{F}\left(\tilde{\Phi}\right) = \mathcal{F}\left(\tilde{F}\right),$$

where we denote by $\cdot$ the term-to-term product of the two matrices. This equation is now very easy to solve, since we know how to calculate the transform of $\tilde{\Phi}$, as shown by the following lemma.

Lemma 9. *The Fourier transform of the transfer function* Φ *is given by*

$$\mathcal{F}\left(\tilde{\Phi}\right) = -\frac{4}{h^2}\left\{\sin\left(\frac{i\pi}{2N}\right)^2 + \sin\left(\frac{j\pi}{2N}\right)^2\right\}_{0\le i,\, j\le 2N-1}.$$

Proof. It is enough to apply the definition and to make a grouping of terms:

$$\begin{aligned}\mathcal{F}\left(\tilde{\Phi}\right)[i,\, j] &= \sum_{k,\, l}\Phi[k,\, l]e^{\frac{2\imath\pi}{2N}ki}e^{\frac{2\imath\pi}{2N}lj} \\ &= \frac{1}{h^2}\left\{e^{\frac{2\imath\pi}{2N}i} + e^{\frac{2\imath\pi}{2N}j} + e^{-\frac{2\imath\pi}{2N}i} + e^{-\frac{2\imath\pi}{2N}j} - 4\right\} \\ &= -\frac{4}{h^2}\left\{\sin\left(\frac{i\pi}{2N}\right)^2 + \sin\left(\frac{j\pi}{2N}\right)^2\right\}.\end{aligned}$$

So we get

$$G(i,\, j) = \mathcal{F}\left(\tilde{U}\right)(i,\, j) = \frac{\mathcal{F}(\tilde{F})(i,\, j)}{\sin^2\left(\frac{i\pi}{2N}\right) + \sin^2\left(\frac{j\pi}{2N}\right)},$$

(be careful with the division 0/ 0 for $i = j = 0$, but we know that the result is 0). We finish thanks to the calculation of the inverse transform, $\tilde{U} = \mathcal{F}^{-1}(G)$. It only remains to extract the part of the matrix $\tilde{U}$ that interests us (that is to say $0 \le i,\, j \le N$), and to add the terms that we had cut off at the beginning.

The entirety of this method to solve the Poisson equation is taken algorithmically in Matlab in the paragraph B.5. To be able to observe the quality of the approximation, we chose an equation for which we know the solution (in fact, we first choose the solution, then we calculate the right hand side), and we create the boundary functions accordingly . Figure 4.10 shows the solution of a Poisson equation for which we explicitly know a solution, namely $u(x,\, y) = e^{xy}$. The figure 4.11 uses the same equation, but modifies the conditions on two of the edges.

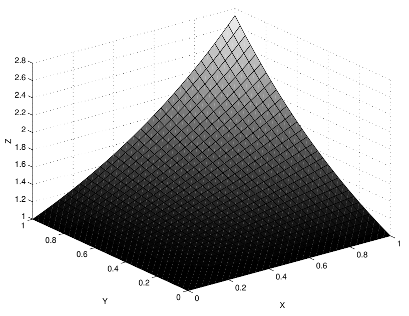

Figure 4.10: Solution of the Poisson equation

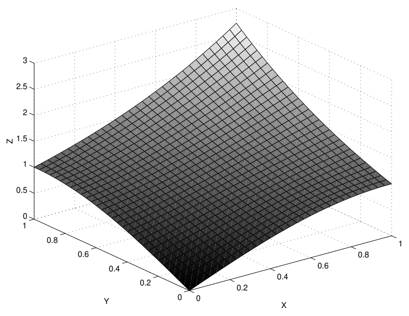

Figure 4.11: Solution modified

Remark 36. (**Case of a quadratic functional**). If we consider an exact quadratic solution, for example the function $u(x,\, y) = x^2 + y^2$, which satisfies the Poisson equation:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 4,$$

one notices that the error obtained during the resolution by a finite difference method is quasi-zero. This is explained by the fact that the approximation of the Laplacian by the equation (4.10) is in fact exact for a polynomial of degree less than two (it is an interpolation formula of order of them).

The exercise 37 takes up this method of resolution by explaining its operation by matrix decompositions.

4.5 Product calculations

Contrary to what one might think after all these applications dedicated to numerical calculation, the FFT algorithm has applications of a much more algebraic nature. It is mainly the convolution property that is used. This algorithm allowed significant advances for certain arithmetic calculations, for example for intensive calculations of large numbers (the search for decimals of π is the best example). We can read about this subject [5] which explains some instructive anecdotes.

4.5.1 Theoretical presentation

Before describing methods using the FFT algorithm presented in paragraph 3.2.1, to calculate products, we can explain in a somewhat theoretical way the approach we are going to follow.

Consider, for $(\xi_0, \ldots, \xi_{N-1}) \in \mathbb{C}^N$, the evaluation morphism

$$\Phi : \left\{ \begin{array}{ccc} \mathbb{C}[X] & \longrightarrow & \mathbb{C}^N \\ P & \longmapsto & (P(\xi_0), \ldots, P(\xi_{N-1})) \end{array} \right. .$$

The kernel of this morphism is the ideal of $\mathbb{C}[X]$ generated by the polynomial $\prod_{k=0}^{N-1}(X-\xi_k)$. In the case where the points ξ_k are distinct, we obtain, by passing to the quotient, a linear isomorphism (since the two spaces have the same dimension N)

$$\tilde{\Phi} : \left\{ \begin{array}{ccc} \mathbb{C}[X]/\prod_{k=0}^{N-1}(X-\xi_k) & \stackrel{\simeq}{\longrightarrow} & \mathbb{C}^N \\ \overline{P} & \mapsto & (\overline{P}(\xi_0), \ldots, \overline{P}(\xi_{N-1})) \end{array} \right. .$$

It is even evidently an isomorphism of algebra, if we endow $\mathbb{C}[X]$ with the product of the polynomials, and $\mathbb{C}^N$ with the product component by component.

Thanks to this evaluation morphism, and its inverse morphism (interpolation), we can draw the following diagram:

$$\begin{tikzcd}
(P_0, \ldots, P_{N-1}), (Q_0, \ldots, Q_{N-1}) \arrow[r, "mult.polynômes", "ON^2)"'] \arrow[d, "évaluation"] & (R_0, \ldots, R_{N-1}) \\
\begin{array}{c}(P(\xi_0), \ldots, P(\xi_{N-1})),\\ (Q(\xi_0), \ldots, Q(\xi_{N-1}))\end{array} \arrow[r, "one-timemult.", "ON)"'] & (R(\xi_0), \ldots, R(\xi_{N-1})) \arrow[u, "interpolation"']
\end{tikzcd} ,$$

which suggests a new way to multiply two polynomials, passing "downwards", namely using the application

$$\Psi : \left\{ \begin{array}{ccc} \mathbb{C}[X] \times \mathbb{C}[X] & \longrightarrow & \mathbb{C}[X]/\prod(X-\xi_k) \\ (P, Q) & \longmapsto & \Phi^{-1}(\Phi(\overline{P}) \cdot \Phi(\overline{Q})) \end{array} \right. ,$$

where we denote by $\overline{P}$ the class of P in the algebra $\mathbb{C}[X]/\prod(X-\xi_k)$. From what we have just said, this morphism therefore calculates the product of the two polynomials modulo the polynomial $\prod(X-\xi_k)$. The natural questions are therefore the following.

- What choice to make for the points $\{\xi_0, \ldots, \xi_{N-1}\}$ so that this new way of calculating a product is fast?
- How to really get the product of the two polynomials, and not just the product modulo a certain polynomial?

In fact, we have already answered these two questions in chapter 3. It suffices to link the TFD to the evaluation of polynomials to reinvest the algorithms already constructed.

Remark 37. The $\tilde{\Phi}$ isomorphism is in fact the canonical isomorphism given by the Chinese theorem:

$$\mathbb{C}[X]/\prod_{k=0}^{N-1}(X-\xi_k) \stackrel{\simeq}{\longrightarrow} \prod_{k=0}^{N-1}\mathbb{C}[X]/(X-\xi_k) \simeq \mathbb{C}^N.$$

Indeed, the $X-\xi_k$ are prime among themselves (because the ξ_k are distinct), therefore the application of the theorem is licit, and the reduction modulo $X-\xi_k$ sends a polynomial P on $P(\xi_k)$.

4.5.2 Multiplication of polynomials modulo $X^N - 1$

The aim of this paragraph is to present the Fourier transform as a transformation (in fact a morphism) on polynomials of fixed degrees. We can then use the algebraic properties of the discrete Fourier transform as well as the FFT algorithm to perform operations on polynomials very quickly. Behind these particularly efficient methods hides a more subtle problem than the simple use of the Fourier transform, since it is the question of the representation of polynomials. Indeed, the Fourier transform makes it possible to juggle between two types of representations, and thus to exploit the strengths of each.

The purpose of this approach being to reinvest the algorithms already presented in chapter 3 (mainly FFT and fast convolution), we hasten to choose wisely the points of evaluation / interpolation ξ_k, by l' occurrence, the N roots $N^{\text{iès}}$ of the unit, i.e. $\xi_k = \omega_N^{-k} = e^{-\frac{2\imath k\pi}{N}}$ for $k = 0, \ldots, N-1$. We see then that the morphism $\tilde{\Phi}$ is in fact the discrete Fourier transform. More precisely, we can rewrite it as follows:

$$\tilde{\Phi} : \begin{cases} \mathbb{C}[X]/(X^N - 1) & \overset{\simeq}{\longrightarrow} & \mathbb{C}^N \\ \overline{P} & \mapsto & \tilde{\Phi}(P) = \mathcal{F}(P_0, \ldots, P_{N-1}) \end{cases},$$

where we denote by $P_0, \ldots, P_{N-1}$ the coefficients of the polynomial $\overline{P}$ (we have chosen the representative of degree less than N). Of course, we used the identity $\prod_{k=0}^{N-1}(X - \xi_k) = X^N - 1$ here. We therefore notice that the computation of the discrete Fourier transform of P (as a vector of $\mathbb{C}^N$) is nothing other than the computation of the values that P takes in them. N roots $N^{\text{ièmes}}$ of the unit, and we find exactly the morphism Φ of the previous paragraph.

The discrete Fourier transform thus makes it possible to juggle between two representations of polynomials of degree $N-1$ (in fact modulo $X^N - 1$):

- *representation by coefficients*: this amounts to considering a polynomial as a vector of $\mathbb{C}^N$. Although very commonly used, this representation has a weak point: it is not at all suitable for calculating the product of two polynomials. While the sum of two polynomials P and Q of degree at most N is calculated in N operations (as shown by the equality $(P+Q)_k = P_k + Q_k$), the product, calculated naively, requires N^2 operations.
- *representation by values*: we give ourselves the polynomial by the values it takes in N distinct points (here taken in a very particular way, the roots N^{ith} of the unit). This representation is much more suited to the computation of the product, since it suffices to do the product of the values of each polynomial.

Take this literally: the FFT algorithm provides a quick and easy way to switch between representations.

To finish, let us recall the equation obtained for the computation of the product of two polynomials modulo $X^N - 1$:

$$P * Q = \mathcal{F}^{-1}\left(\mathcal{F}(P) \cdot \mathcal{F}(Q)\right). \tag{4.12}$$

In this equation, we have confused vector and modulo polynomials $X^N - 1$, and the product $P*Q$ can also be seen as a product of circular convolution between two vectors. The operation performed can be graphically represented. The first two graphs in figure 4.12 show the polynomials defined by $P = 1+2X+X^3-X^4+X^5$ as well as $Q = XX^2 + 2X^3 + 2X^5$. On the abscissa, we put the degrees of monomials $1, \ldots, X^{10}$. We therefore choose to work in $\mathbb{Z}/11\mathbb{Z}$. This makes sense, because the degree of the product $P * Q$ is precisely 10. Thus, when we represent in the graph on the right the product of the two vectors, we find the graphical representation of the product $P * Q$, since the reduction modulo $X^{11} - 1$ did not have effect.

Remark 38. (**Lagrange interpolation**). Algorithms make it possible to calculate the interpolation polynomials in the case where the points ξ_k are not necessarily roots of the unit. There is of course the result of *Lagrange*, which gives an explicit basis of $\mathbb{C}_{N-1}[X]$ (space of polynomials of degree at most $N-1$) in which the computation is done simply. Indeed, if we look for the polynomial P which takes the values y_i at the points x_i, for $i = 0, \ldots, N-1$, then $P = \sum_{i=0}^{N-1} y_i P_i$, where P_i is the i^{rd} *Lagrange polynomial* associated with the points $\{x_i\}_{i=0}^{N-1}$:

$$P_i \overset{\text{def.}}{=} \frac{\prod_{j=0}^{N-1}(X - x_j)}{\prod_{j \neq i}(x_i - x_j)}.$$

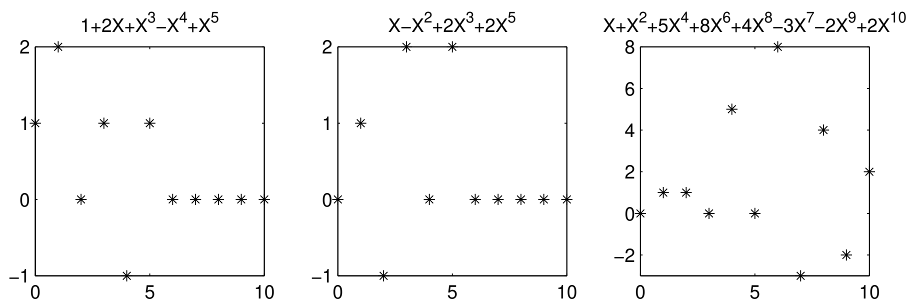

Figure 4.12: Graphical representation of the product of polynomials by convolution

However, the numerical computation of the interpolation polynomial in the base of $\{P_i\}_{i=0}^N$ is not used in practice, because it leads to an accumulation of numerical errors. We prefer to use the *divided differences* technique, which is explained for example in [22].

4.5.3 Multiplication of polynomials

The difficulty with which we are faced when calculating the product of two polynomials P and Q (of degree $N-1$) by the method presented above is that a priori, the product PQ is a polynomial of degree $2N-2$. This polynomial will therefore be reduced modulo X^N-1, which often has a more than undesirable effect ... More precisely, the coefficients of the product are given by the equation

$$\forall n \in \{0,\ldots,\,2N-1\}, \quad (PQ)_n = \sum_{k=0}^{n} P_k Q_{n-k}. \tag{4.13}$$

It is in fact an acyclic convolution product (to be compared to the cyclic product of the equation (4.12)), and we will be able to use the technique already presented in the Section 3.3.3, to calculate it.

This approach (adding zeros at the end of the vector to make the product cyclic) is very intuitive, since it consists in considering the two polynomials as polynomials of degree $2N-1$ (by adding zero coefficients). We can then apply the approach presented in the previous paragraph (ie use a cyclic convolution product, or if you prefer, calculate the product modulo $X^{2N}-1$). Fortunately, this does not change the result, since the polynomial PQ is not affected by the reduction modulo $X^{2N}-1$. In a more theoretical way, this amounts to using the bijectivity of the application

$$\left\{ \begin{array}{ccc} \mathbb{C}_{2N-1}[X] & \longrightarrow & \mathbb{C}[X]/(X^{2N}-1) \\ P & \longmapsto & P \mod X^{2N}-1 \end{array} \right. ,$$

where we denote by $\mathbb{C}_{2N-1}[X]$ the space of polynomials of degree less than or equal to $2N-1$.

4.5.4 Multiplication of large integers

We denote by $(a_0,\ldots,\,a_{N-1})$ the base representation b of a large integer a, that is

$$a = a_0 + a_0 b + \cdots + a_{N-1} b^{N-1}.$$

We notice that the multiplication of two integers a and a' is similar to the calculation of a product of polynomials, with one exception: the integers a_k and a'_k, for $k = 0,\ldots,\,N-1$, must belong to the set $\{0,\ldots,\,b-1\}$. If we want to quickly compute the multiplication of two large integers, we can therefore use the polynomial product technique that we have just exposed in the previous paragraph, followed by a “carry propagation” phase. Matlab programs allowing to achieve all this are gathered in the paragraph B.4.

This approach nevertheless suffers from some weak points, mainly related to the use of floating point calculations (for the classic FFT algorithm), which are subject to rounding errors (while integer calculations are both faster and free of errors). This problem will be solved in chapter 6, thanks to the introduction of the Fourier transform on finite fields and rings.

4.6 Exercises

Exercise 33 (Compactly supported transform)**.** *We have to prove the proposition 42. Let f be a function such that $\mathrm{Supp}(\hat{f}) \subset [-A,\, A]$. Explain why f is of class $\mathcal{C}^\infty$, and calculate its successive derivatives $f^{(n)}(t_0)$, for $t_0 \in \mathbb{R}$, as an integral between $-A$ and A. Using a limited expansion of $t \mapsto e^{\imath t}$ in t_0, deduce an expansion of f. Deduce that $f \neq 0$ cannot vanish over a whole nonempty interval.*

Exercise 34 (Solving the heat equation by finite differences)**.** *We want to solve in an approximate way the equation of heat on the circle, whose formulation we recall, for $\kappa = 1$:*

$$\begin{cases} \forall (t,\, x) \in \mathbb{R}_*^+ \times S^1, & \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \\ \forall x \in S^1, & u(0,\, x) = f(x) \end{cases}, \tag{4.14}$$

where the sought solution u is assumed to be sufficiently regular over $\mathbb{R}_*^+ \times S^1$, and continues over $\mathbb{R}^+ \times S^1$. To do this, we consider a discretization of step $d = \frac{1}{N}$ in space, and of step h in time. This leads us to consider the vectors $u^n \in \mathbb{R}^N$, for $n \geq 0$, supposed to approximate the function u:

$$\forall n \geq 0,\ \forall k \in \{0, \dots,\, N-1\}, \quad u^n[k] \approx u(nh,\, kd).$$

1. *Show that we can, following a discretization of the equation (4.14), consider the difference equation*

$$u^{n+1}[k] - u^n[k] = s\left(u^n[k+1] + u^n[k-1] - 2u^n[k]\right),$$

   for $n \geq 0$ and $k = 1, \dots, N-1$. We have noted $s \overset{\text{def.}}{=} \frac{h}{d^2}$, and by convention, we define $u^n[-1] \overset{\text{def.}}{=} u^n[N-1]$ and $u^n[N] \overset{\text{def.}}{=} u^n[0]$.

2. *Show that this explicit scheme can be written in the form of a convolution. Is it advantageous to compute u^n by iterated convolutions using the FFT algorithm? Propose a Matlab implementation of the chosen algorithm.*

3. *We say that the chosen numerical scheme is stable if, for any u_0 such that $\|u_0\|_\infty \leq 1$, then the approximate solution u^n remains bounded whatever n. Give, using the discrete Fourier transform, a necessary and sufficient condition for the scheme we have just built to be stable.*

4. *We now want to consider non-explicit schemas, that is to say such that u^{n+1} is not not given directly in terms of u^n. We consider the scheme*

$$u^{n+1} - u^n = A * \left(\theta u^{n+1} + (1-\theta) u^n\right),$$

   where θ is a varying parameter in $[0,\, 1]$, and A is the vector such that $A[0] = -2s$, $A[1] = A[-1] = s$, and all other entries are zero. In particular, we notice that for $\theta = 0$ we find the explicit scheme already constructed, and that for $\theta = 1$, we obtain an implicit scheme. Explain how we can solve this equation using the Fourier transform. Then study the problem of the stability of the diagram obtained.

5. *Resume the previous questions in the case of the heat equation in dimension 2, that is to say on $\mathbb{R}^+ \times S^1 \times S^1$. In particular, we will propose an implementation of the implicit algorithms using two-dimensional FFT calculations.*

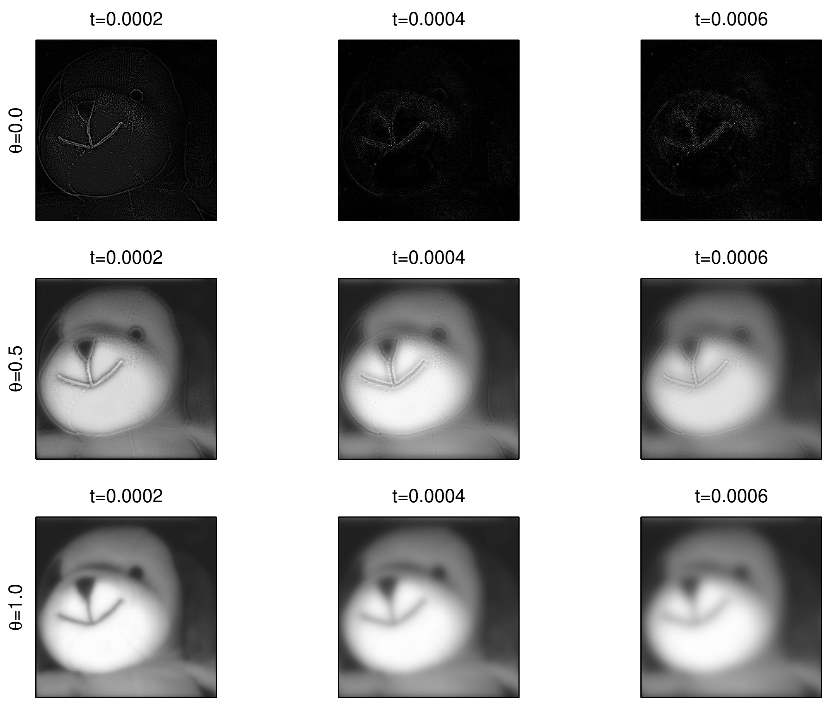

Figure 4.13: Solving the heat equation by finite differences

Figure 4.13 shows the solution of the 2D heat equation by different schemes. Horizontally, each image represents a step of the algorithm. The first line corresponds to $\theta = 0$, and with an image of size 256×256, we verify that we must take h of the order of 10^{-6} so that the diagram either stable (this condition is of course not verified in the figure, where we have $h = 0.0002$). The second line corresponds to $\theta = 0.5$, but with such a large step, we can see that some instabilities appear. Finally, the last line corresponds to $\theta = 1$, and the scheme is very stable.

Exercise 35 (Uniqueness for the heat equation)**.** *This proof is taken from the book of Dym and McKean[28]. We consider the heat equation on the circle (4.14). We want to show that under the f continuous hypothesis, the equation (4.8) does indeed define a solution of the equation, and that it is in fact the only one.*

1. *Show that for $t > 0$, the solution u can be written as a convolution:*

$$u(t, x) = p_t * f(x) \quad \text{with} \quad p_t(x) \overset{\text{def.}}{=} \sum_{n \in \mathbb{Z}} e^{-2\pi^2 n^2 t} e_n(x).$$

2. *In the case where $f \in \mathcal{C}^2(S^1)$, show that we have $\|u(t, \cdot) - f\|_\infty \overset{t \to 0}{\longrightarrow} 0$.*

3. *We want to show that if $f \geq 0$, then $u \geq 0$. We consider the function v defined $v(t, x) = e^{\beta t} u(t, x)$, for a certain parameter β. Show that if we assume that $u(t_0, x_0) < 0$, the function v reaches its minimum $\alpha < 0$ on $[0, t_0] \times S^1$ for a some time $t_1 > 0$ and a position x_1. Show while we have*

$$0 \geq \frac{\partial v}{\partial t}(t_1, x_1) = \alpha\beta + \frac{1}{2}\frac{\partial^2 u}{\partial x^2}(t_1, x_1) \geq \alpha\beta$$

Deduce a contradiction by taking $\beta < 0$.

4. *Using the convolution product that defines u, deduce that p_t is positive. Deduce the maximum principle for the heat equation:*
$$\forall t > 0, \quad \|u(t, \cdot)\|_\infty \le \|f\|_\infty.$$
Show that this ensures the uniqueness of the solution of the heat equation.

5. *Using a sequence of functions $f_n \in \mathcal{C}^2(S^1)$ which converges uniformly to f, deduce that in the case where f is simply continuous, we still have the convergence $\|u(t, \cdot) - f\|_\infty \overset{t\to 0}{\longrightarrow} 0$.*

Exercise 36 (3D electric potential)**.** *We want to generalize the algorithm for solving the Poisson equation described in paragraph 4.4.3 for three-dimensional problems. For example, we want to determine an electric potential u which satisfies the Poisson equation:*

$$\forall (x, y, z) \in [0, 1] \times [0, 1] \times [0, 1], \quad \Delta u(x, y, z) \overset{\text{def.}}{=} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = f(x, y, z),$$

with boundary conditions as well as a user-specified function f. It will of course be necessary to use a three-dimensional Fourier transform, and to think about making the encountered 3D arrays odd. To represent the solution, we can draw equipotential surfaces, i.e. the surfaces of equation $f(x, y, z) = \lambda$ for certain values of the parameter λ .

Exercise 37 (Matrix formulation for the Poisson equation)**.** *This exercise, which offers a more computational explanation of the algorithm for solving the Poisson equation, is partly inspired by the article by Swarztrauber and Sweet[63]. We take the notations of paragraph 4.4.3, and we consider in particular a square matrix U of size $N-1$ (the indices varying from 1 to $N-1$) which is the solution of the finite difference equation (4.10) inside the square $[0, 1] \times [0, 1]$ (that is say without the boundary terms).*

1. *Without taking into account the boundary terms, show that we can write the difference equation in the form*
$$T_{N-1}U + UT_{N-1} = F,$$
where T_{N-1} is the matrix of size $N-1$ with $-2/h^2$ on the diagonal and $1/h^2$ on the sub-diagonal and the over-diagonal.

2. *Since the value of the solution on the edges is assumed to be known, use the same approach as that employed in paragraph 4.4.3 to obtain a modified equation of the type*
$$T_{N-1}\tilde{U} + \tilde{U}T_{N-1} = \tilde{F}. \tag{4.15}$$

3. *Show that the eigenvectors of T_{N-1} are the*
$$V_j \overset{\text{def.}}{=} \left\{ \sin\left(\frac{ij\pi}{N}\right) \right\}_{i=1}^{N-1}.$$
Determine the associated eigenvalues. We denote by V the base change matrix, which means that its columns are the V_j, and D the diagonal matrix whose inputs are the calculated eigenvalues. By noting $U_0 \overset{\text{def.}}{=} V^{-1}\tilde{U}V$ and $\tilde{F}_0 = V^{-1}\tilde{F}V$, deduce that U_0 satisfies l'equation
$$DU_0 + U_0 D = \tilde{F}_0.$$
Solve this equation.

4. *Show that the matrix V is in fact orthogonal, and that the computation of Vx, where X is a matrix of size $N-1$, is equivalent to a calculation of sine transform (that is to say the imaginary part of a certain discrete Fourier transform). Deduce that the computation of $\tilde{U} = VU_0V^{-1}$ is in fact equivalent to the computation of a two-dimensional sine transform (i.e. a transform on the lines followed by'a transform on the columns of a matrix).*

5. *Explain how we can calculate sine transforms (unidimensional then in 2D) thanks to a DFT of double size. How to calculate the inverse transform? Finally, make the connection between the matrix approach proposed in this exercise and the computation of convolution which supported the algorithm proposed in Paragraph 4.4.3.*

Exercise 38 (Image smoothing)**.** *Write a program that smooths a 2D image in grayscale, as shown in the figure 4.5. We can use a Gaussian transfer function, and adapt the parameters to the size of the image.*

Exercise 39 (Correlation and image detection)**.** *This exercise is inspired by an article by Lewis[43], which brings up to date the use of correlation for image detection. Let $f \in \mathbb{R}^{N\times N}$ be an image of size N, and $g \in \mathbb{R}^{P\times P}$ another image, typically much smaller in size than that of f. The question is to determine if the image g is a subimage of f, and if so, to locate its location. For this purpose, we define the distance between f and g*

$$\forall (u,\, v) \in \{0, \ldots,\, N-1\}^2,\ d(f,\, g)[u,\, v]^2 \overset{\text{def.}}{=} \sum_{(x,\, y)\in D(u,\, v)} (f(x,\, y) - g(xu,\, yv))^2,$$

where $D(u,\, v)$ denotes the subset of $\{0, \ldots,\, N-1\}^2$ formed by the pairs $(x,\, y)$ such that $(xu,\, yv) \in \{0, \ldots,\, P-1\}^2$.

1. *What is the intuitive meaning of $d(f,\, g)$? In what circumstance is $d(f,\, g)$ close to zero? In the event that the quantity*

$$P_{u,v}(f) = \sum_{(x,\, y)\in D(u,\, v)} f(x,\, y)^2$$

is almost constant, showing that finding the points where $d(f,\, g)$ is small amounts to maximizing the correlation between f and g

$$\mathrm{Corr}(f,\, g)[u, v] \overset{\text{def.}}{=} \sum_{(x,\, y)\in D(u,\, v)} f(x,\, y)\, g(xu,\, yv).$$

2. *Show that* $\mathrm{Corr}(f,\, g)$ *can be written as an acyclic convolution product. Deduce that we can calculate this correlation quickly using the FFT algorithm.*

3. *We want to correct the defect we introduced by supposing that $P_{u,v}(f)$ is almost constant. We denote by $\tilde{f}_{u,v}$ the average of f over $D(u,\, v)$, and $\tilde{g}$ the average of g. We then define the normalized correlation*

$$\overline{\mathrm{Corr}}(f,\, g)[u,\, v] \overset{\text{def.}}{=} \frac{\sum_{(x,\, y)} (f(x,\, y) - \tilde{f}_{u,v})(g(xu,\, yv) - \tilde{g})}{\left\{\sum_{(x,\, y)} (f(x,\, y) - \tilde{f}_{u,v})^2 \sum_{(x,\, y)} (g(x,\, y) - \tilde{g})^2\right\}^{1/2}},$$

where the sums relate to $(x,\, y) \in D(u,\, v)$. Explain how this quantity actually provides a correction. Do we still have a fast FFT calculation algorithm?

4. *Show that the numerator of* $\overline{\mathrm{Corr}}(f,\, g)$ *is written as a convolution. We denote, for $k = 1,\, 2$, the "sliding sums"*

$$\forall (u,\, v) \in \{0, \ldots,\, N-1\}^2, \quad s_k(u,\, v) \overset{\text{def.}}{=} \sum_{(x,\, y)\in D(u,\, v)} f[x,\, y]^k,$$

with by convention $s_k(u,\, v) = 0$ for $u \geq N$ or $v \geq N$. Show that s_k satisfies the recurrence equation

$$\begin{aligned} s_k(u,\, v) = {} & s_k(u+1,\, v) + s_k(u,\, v+1) - s_k(u+1,\, v+1) \\ & + f(u,\, v) + f(u+P,\, v+P) - f(u,\, v+P) - f(u+P,\, v). \end{aligned}$$

Deduce a fast computation algorithm of s_k (evaluate its complexity), then of $\overline{\mathrm{Corr}}(f,\, g)$.

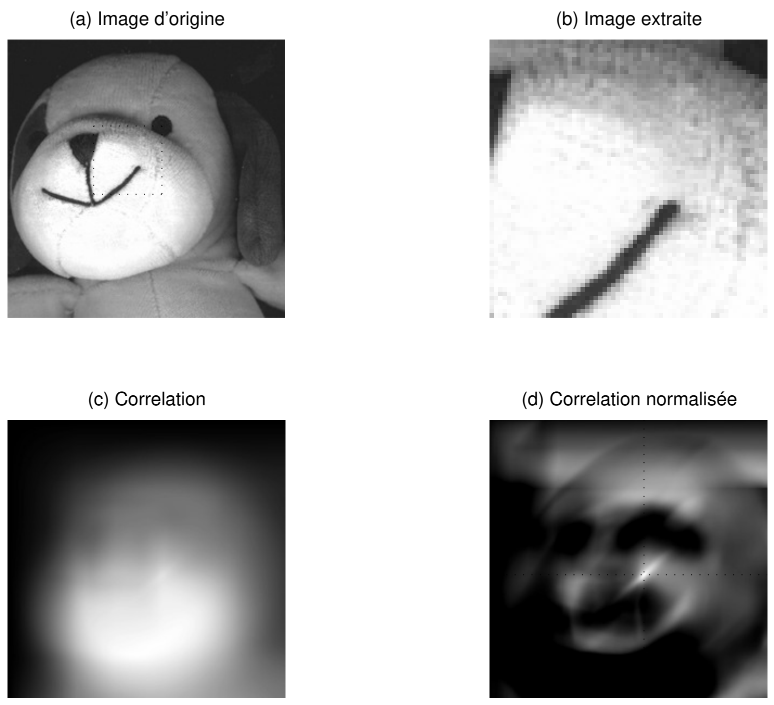

Figure 4.14: Correlation between two images

Figure 4.14 shows an example of application of this method. We can clearly see that the normalized correlation (image (d)) has a much sharper maximum than the non-normalized correlation (image (c)). We can note that[43] *proposes a fast calculation algorithm which was used among other things to perform the readjustments in the movie Forest Gump (1994).*

Exercise 40 (Rotation by FFT)**.** *Let $f \in \mathbb{C}^{N \times N}$ be a two-dimensional signal. We define, for $v = (v_1, v_2) \in \mathbb{R}^2$ and $\lambda \in \mathbb{R}$,*

$$T_v(f)[k,\, l] = f[k - v_1,\, l - v_2], \quad S_\lambda^{(x)}(f)[k,\, l] = f[k - \lambda l,\, l], \quad S_\lambda^{(y)}(f)[k,\, l] = f[k,\, l - \lambda k].$$

1. *Express $\mathcal{F}(T_v(f))$ in terms of $\mathcal{F}(f)$. Deduce a fast algorithm to carry out any translation of an image. By translating each row (resp. Each column) by f, write a fast algorithm to calculate $S_\lambda^{(x)}(f)$ (respectively $S_\lambda^{(y)}(f)$).*

2. *Show that a rotation of angle θ around the origin can be written in the form*

$$\begin{pmatrix} \cos(\theta) & -\sin(\theta) \sin(\theta) & \sin(\theta) \end{pmatrix} = \begin{pmatrix} 1 & \lambda_1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \lambda_2 & 1 \end{pmatrix} \begin{pmatrix} 1 & \lambda_3 0 & 1 \end{pmatrix}.$$

   With the previous question, deduce a fast algorithm for rotating an image $f \in \mathbb{C}^{N \times N}$ around the origin.

3. *What are the advantages and disadvantages of this algorithm? How to solve them? How to rotate an image around its center?*

The figure 4.15 shows several rotations of an image around its center.

Exercise 41 (Low pass filter)**.** *We want to create a low pass convolution filter.*

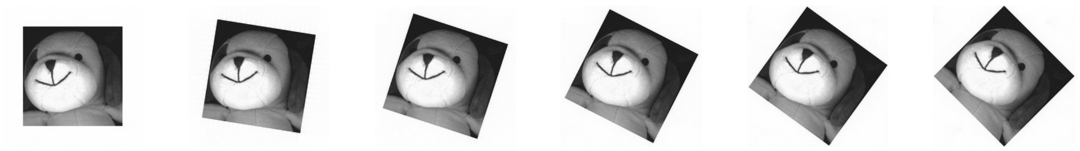

Figure 4.15: Rotation of an image by FFT

1. *Write a Matlab program which builds a vector f, of size N, such that the filter Φ^f keeps the low frequencies $N/2$, and removes the $N/2$ high frequencies.*

2. *Represent with great precision the continuous Fourier transform of the impulse response f (in other words the frequency response of the filter). What do we see?*

3. *Considering a less brutal cut in the conserved / rejected frequencies, go back to the previous questions and comment on the results. In particular, imagine a family of filters f_ϵ, $\epsilon \in [0, 1]$, with a sharp cutoff for $\epsilon = 0$ and soft for $\epsilon = 1$.*

Figure 4.16 shows three different filters, with increasingly softer cuts. We can also see the discrete filter transforms, and their continuous Fourier transforms.

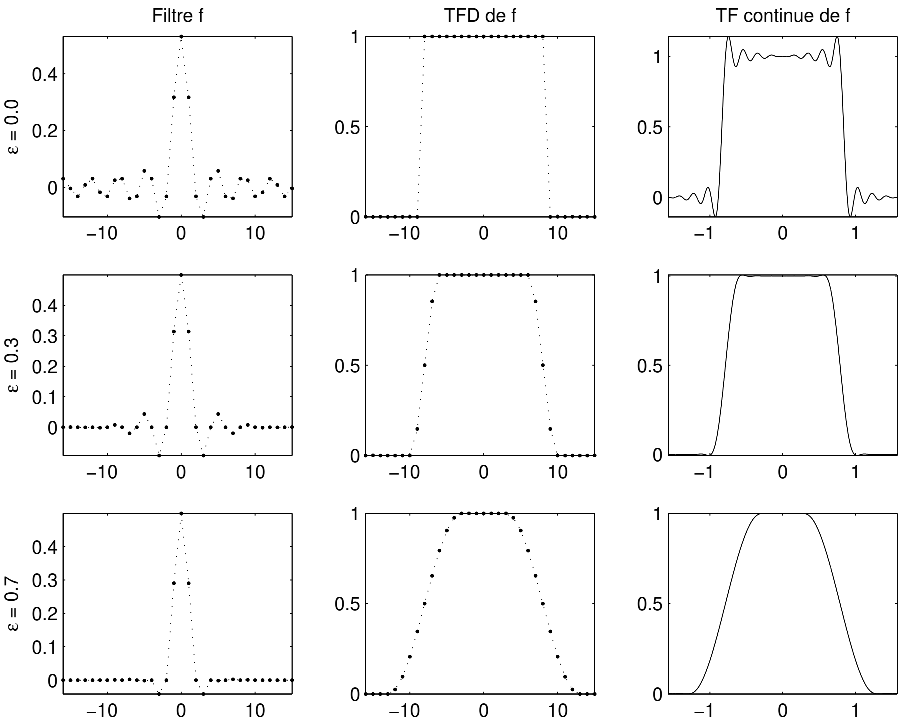

Figure 4.16: Lowpass filters for different values of ϵ

Exercise 42 (Integer iterations). *We consider the following experiment: we have n children in a circle, and we give them each an even, arbitrary number of candies. The game consists in having each child give his neighbor on the right half of his candy, and to iterate the process.*

1. *We arrange for the children to have an even number of candies each time they share. For this, an external person distributes, after each iteration, a candy to each child with an odd number of candies. Show that after a finite number of iterations, all the children have the same number of candies.*

2. *If we allow fractional parts of candies, show how we can translate this experiment into a convolution calculus. Deduce that after a potentially infinite number of iterations, all children have the same number of candies.*

3. *Study the first two questions for different sharing rules. For example, what if each child gives half of their candy to their neighbor on the left, and the other half to their neighbor on the right?*

Exercise 43 (Karatsuba algorithm)**.** *We are going to explain the construction of a recursive polynomial multiplication algorithm . It uses a technique called divide and reign, often used in algorithms, see for example the book of Cormen*[20] *for other examples. We consider two polynomials P and Q of degree n over a field K. We denote by $k \overset{\text{def.}}{=} \lfloor (n+1)/2 \rfloor$.*

1. *We write the polynomials P and Q in the form*

$$P(X) = P_0(X) + X^k P_1(X) \quad \textit{and} \quad Q(X) = Q_0(X) + X^k Q_1(X),$$

where the polynomials P_0, Q_0 are of degree less than k, and the polynomials P_1, Q_1 are of degrees less than k or $k+1$, depending on the parity of n. Show that the product $P(X)Q(X)$ can be in the form

$$P(X)Q(X) = R_0(X) + X^k R_1(X) + X^{2k} R_2(X).$$

Specify the value of the polynomials involved in this equality.

2. *Show that the polynomial R_1 can be calculated using only one multiplication, but on the other hand 4 additions.*

3. *Implement a recursive algorithm using the decomposition we have just performed at each step. Prove that the complexity of this algorithm is $On^{\log_2(3)})$. For which values of n this algorithm is preferable to the algorithm using the FFT described in paragraph 4.5.3?*

Exercise 44 (Spline and filtering)**.** *This exercise requires some knowledge of Fourier series. If $\{f[k]\}_{k\in\mathbb{N}}$ is a sequence of $\ell^2(\mathbb{Z})$, we define its Fourier transform by*

$$\forall x \in \mathbb{R}, \quad \hat{f}(x) \overset{\text{def.}}{=} \sum_{k\in\mathbb{Z}} f[k] e^{-\imath kx}.$$

It is a 2π -periodic function, which we can assimilate to a function of $L^2(\mathbb{R}/2\pi\mathbb{Z})$.
Let $u : \mathbb{R} \to \mathbb{R}$ a continuous function which decreases fast enough to $\pm\infty$. We assume that we actually know sampled values of u, denoted by $u_d[k] \overset{\text{def.}}{=} u(k)$, for $k \in \mathbb{Z}$. We want to interpolate these values in one of the two following forms:

$$v(x) \overset{\text{def.}}{=} \sum_{k\in\mathbb{Z}} u_d[k] \varphi(xk) \tag{4.16}$$

$$v(x) \overset{\text{def.}}{=} \sum_{k\in\mathbb{Z}} a[k] \psi(xk), \tag{4.17}$$

the functions φ and ψ being given in advance, with a little support. We assume of course that the interpolation is exact, ie $\forall k \in \mathbb{Z}$, $v(k) = u(k)$. The sequence $a[k]$, $k \in \mathbb{Z}$, is unknown, and we will have to determine it. The (4.16) *interpolation scheme corresponds to direct interpolation, while* (4.17) *corresponds to indirect interpolation.*

1. *We denote by* $\psi_d[k] \overset{\text{def.}}{=} \psi(k)$ *the sampled sequence of* ψ*. We suppose that*

$$\forall \xi \in \mathbb{R}, \quad \hat{\psi}_d(\xi) \neq 0.$$

Show then that the indirect interpolation problem admits a solution c uni que, given by the relation

$$\forall \xi \in \mathbb{R}, \quad \hat{c}(\xi) = \frac{\hat{u}(\xi)}{\hat{\psi}_d(\xi)}.$$

How can we reduce this interpolation to a direct interpolation? What problem are we encountering?

2. *We define the B-spline function of order* n*, denoted* β^n *by*

$$\beta^0 = 1_{[-\frac{1}{2},\frac{1}{2}]} \quad \textit{and} \quad \forall n > 0, \quad \beta^n = \beta^0 * \beta^{n-1}.$$

What is the support for β^n*? Do these functions allow you to define a direct interpolation scheme? Indirect? The figure 4.17 shows the first 4 spline functions* β^n*.*

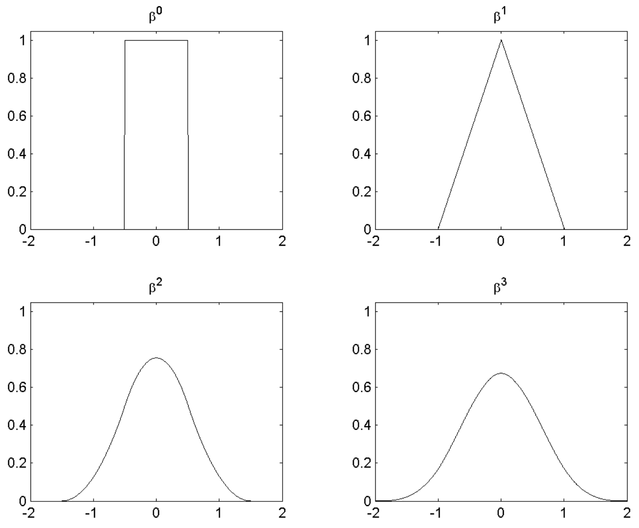

Figure 4.17: Basic spline functions

3. *Calculate the value of* $\hat{\beta^n}$ *(continuous Fourier transform). We denote by* β^n_d *the sequence sampled from* β^n*. Calculate the value of* $\hat{\beta^n_d}$ *(Fourier series), and show that this function does not vanish (we will have to distinguish according to the parity of* n*).*

4. *Deduce an expression of the function* β^n_{card} *which allows to perform an indirect interpolation from spline functions. What is its support? Show that we have the following convergence, in* $L^2(\mathbb{R})$*:*

$$\beta^n_{card} \overset{n\to+\infty}{\longrightarrow} \text{sinc}, \quad \textit{where} \quad \text{sinc}(x) \overset{\text{def.}}{=} \frac{\sin(\pi x)}{\pi x}.$$

When $n \to \infty$*, what kind of interpolation do we perform? The figure 4.18 shows a comparison between the cardinal functions corresponding to the spline interpolation of degree 3 (i.e.* β^3_{card} *) and Shannon interpolation (i.e.* sinc*). We see that the spline function has a lot less "bounces".*

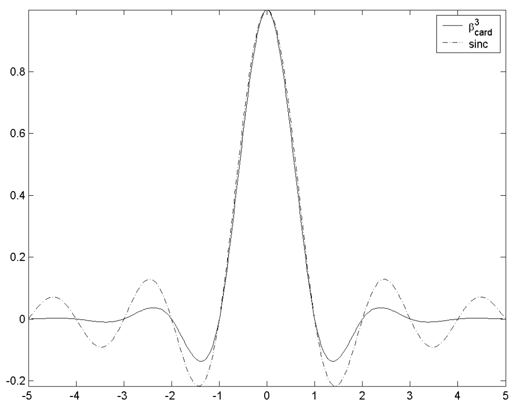

Figure 4.18: Comparison between spline and cardinal sine

5. *Writing c as a convolution, explain how we can calculate an approximate value of it using the FFT algorithm. What is the downside of this method? In the exercise 52, we will see how we can calculate c by much more efficient recursive calculations.*

There are more classical methods to calculate interpolation by cubic splines. For example, in[16]*, Ciarlet decomposes the function sought in a suitable basis of polynomials, and solves a tridiagonal linear system. Compare this method with the filtering method proposed in this exercise. In practice, we consider only a finite number of values $u_d[k]$, for $k \in \{0, \ldots, K-1\}$. One can then show that one loses the uniqueness of the solution, but that one can impose conditions on the edge to remedy the problem. The exercise 52 proposes to study this problem for cubic splines. Figure 4.19 shows the interpolation by cubic splines, with two types of boundary conditions:*

- *Free splines: we impose that the derivative of the interpolating function vanishes at the edges.*
- *Splines "not-a-knot": we do not impose conditions on the edge points, but we impose that the third derivative of v be continuous in 1 and $K-2$.*

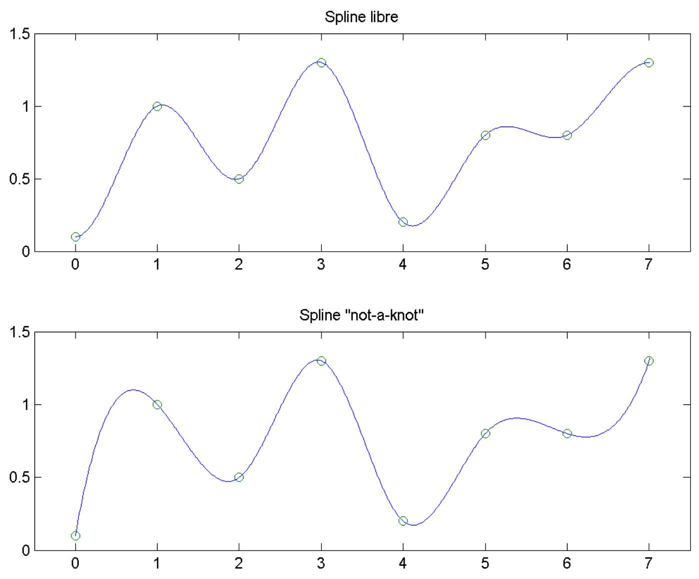

Figure 4.19: Interpolation by cubic splines

Chapter 5

Extension of the notion of Fourier transform

s

This chapter brings together many related notions of the Fourier transform, as well as direct applications of these developments. We will thus have to define new transformations, among others *the Hartley transform*, the *transform in Z*, and the *fractional Fourier transform.* Most often, it is a question of finding algorithms to overcome certain weaknesses of the DFT (for example the *Hartley transform*), or to extend certain notions in a natural way (the *transformed into Z* for example). We will study in which cases these transformations are more efficient or more suitable than the discrete Fourier transform, and which applications can benefit from the fast algorithms obtained. The two important points to keep in mind when working with such transformations are:

– They do not correspond to approximate calculations. These are exact formulas, which very often have an inversion formula. These transforms can be useful for certain numerical computations (for example the approximate computation of a Fourier transform or of an infinite convolution), but they are above all algebraic calculations.
– They have fast calculation algorithms. It is these algorithms which give all its value to a transform, and which make it usable in an intensive way. These algorithms are direct consequences of the algebraic nature of transforms, and only reflect certain algebraic symmetries and invariances.

5.1 Hartley transform

One of the disadvantages of the discrete Fourier transform is that it requires calculations with complex numbers, which is not suitable for calculation of convolutions with real signals. Indeed, unnecessary complex multiplications and additions (more expensive than real multiplications and additions) are performed, and rounding errors are only amplified. We will define a transform, called *Hartley transform*, which allows, like the DFT, to calculate convolution products, but which only involves calculations with real numbers. A very complete reference on the *Hartley transform* and its applications is Bracewell [10].

5.1.1 Definition and first properties

Definition 27 (Hartley transform)**.** *Let* $f = \{f[n]\}_{n=0}^{N-1} \in \mathbb{C}^N$*. We define its discrete Hartley transform* $\mathcal{H}(f) \in \mathbb{C}^N$ *by*

$$\forall k \in \{0, \ldots, N-1\}, \quad \mathcal{H}(f)[k] \overset{\text{def.}}{=} \sum_{n=0}^{N-1} f[n] \operatorname{cas}\left(\frac{2\pi}{N} nk\right), \tag{5.1}$$

where we noted $\operatorname{cas}(x) \overset{\text{def.}}{=} \sin(x) + \cos(x) = \sqrt{2}\cos\left(x - \frac{\pi}{4}\right)$.

Remark 39. The discrete Hartley transform has its continuous analogue, namely, for a function $f \in L^1(\mathbb{R})$, the function

$$\mathcal{H}(f) : s \to \int_{\mathbb{R}} f(x) \operatorname{cas}(2\pi s x) \mathrm{d}x.$$

Most of the statements valid in the discrete case have an analogous formulation for the continuous case, and it is left to the reader to state them.

Proposition 44 (Inversion formula)**.** *$\mathcal{H}$ is an isomorphism from $\mathbb{R}^N$ into $\mathbb{R}^N$. More precisely, for $f \in \mathbb{R}^N$, we have $\mathcal{H}^2(f) = Nf$, which means that the inverse of the Hartley transform is $\mathcal{H}^{-1} = \frac{1}{N}\mathcal{H}$.*

Proof. We will use, for n and $n' \in \{0, \ldots, N-1\}$, the orthogonality relation

$$\sum_{k=0}^{N-1} \operatorname{cas}\left(\frac{2\pi}{N} nk\right) \operatorname{cas}\left(\frac{2\pi}{N} n'k\right) = N\delta_n^{n'}, \tag{5.2}$$

where $\delta_n^{n'}$ is 1 if $n = n'$ and 0 otherwise. We denote by $\omega = e^{\frac{2\imath\pi}{N}}$, hence

$$v_n[k] \overset{\text{def.}}{=} \operatorname{cas}\left(\frac{2\pi}{N} nk\right) = \frac{1}{2}\left(\omega^{nk}(1-\imath) + \omega^{-nk}(1+\imath)\right).$$

We therefore calculate

$$\langle v_n, v_{n'} \rangle = \frac{(1-\imath)^2}{4} \sum_k \omega^{(n+n')k} + \frac{(1+\imath)^2}{4} \sum_k \omega^{(n+n')k} + \frac{(1-\imath)(1+\imath)}{2} \sum_k \omega^{(n-n')k}. \tag{5.3}$$

The first two sums are opposite, and the last is $N\delta_n^{n'}$. To obtain the inversion formula, we note that $\mathcal{H}(f)[n] = \langle f, v_n \rangle$, hence

$$\mathcal{H}(\mathcal{H}(f))[n] = \sum_{k=0}^{N-1} \mathcal{H}(f)[k] v_n[k] = \sum_{k=0}^{N-1} f[k] \langle v_k, v_n \rangle = Nf[n].$$

Proposition 45. *We have the following relations between the discrete Fourier transform and the Hartley transform of a vector $f \in \mathbb{R}^N$:*

$$\begin{aligned} \mathcal{H}(f) =& \mathcal{R}(\mathcal{F}(f)) + \mathcal{I}(\mathcal{F}(f)) \\ \mathcal{F}(f) =& \mathcal{H}(f)_S - \imath \mathcal{H}(f)_A. \end{aligned} \tag{5.4}$$

We have noted, for $g \in \mathbb{R}^N$, g_S and g_S the symmetric and anti-symmetric parts of g, introduced in the definition 24.

Remark 40. The relations (5.4) imply, in the case where we restrict ourselves to real vectors, a one-to-one correspondence between discrete Fourier transform and Hartley transform. How to explain that N complex numbers (for the DFT) can contain as much information as only N real numbers? In reality, there is no contradiction, we just have to remember that in the case of a real signal, the vector $\mathcal{F}(f)$ is the conjugate of $\mathcal{F}(f^\sharp)$ (where $f^\sharp$ is defined at the equation (3.27)), there is therefore information redundancy (exactly twice as much information). For a real signal, the Hartley transform is much more economical (whether in terms of computation time or memory space) than the Fourier transform, since we will handle exactly half the information. It is this quality that we will exploit in the following paragraph.

5.1.2 Fast Hartley transform

As for the DFT, we have a fast algorithm to calculate the Hartley transform. This algorithm has been described in detail by Ullmann [66]. To understand it, we are going to perform a slicing of the transform, as we have already done during the study of the FFT algorithm. It is of course a question of exploiting the (algebraic) symmetries of the transformation, as well as the properties of the function cas. Once all this is in place, we will naturally see a recursive algorithm appear.

This algorithm uses a property of temporal decimation. To obtain the corresponding recurrence equation, we will proceed as we have already done for the discrete Fourier transform. We must decompose the sum that defines $\mathcal{H}(f)$, for $f \in \mathbb{C}^N$, as follows:

$$\mathcal{H}(f)[k] = \sum_{n=0}^{N/2-1} f[2n] \operatorname{cas}\left(\frac{2\pi}{N} 2nk\right) + \sum_{n=0}^{N/2-1} f[2n+1] \operatorname{cas}\left(\frac{2\pi}{N}(2n+1)k\right). \tag{5.5}$$

Let us use the notations of (3.7). We recognize in the left sum a Hartley transform of length $N/2$, more precisely $\mathcal{H}(f^0)$. The right sum poses a problem, but we can overcome this difficulty by using the following property of the function cas.

Proposition 46. *We have, for $(\alpha, \beta) \in \mathbb{R}^2$,*

$$\operatorname{cas}(\alpha+\beta) = \operatorname{cas}(\alpha)\cos(\beta) + \operatorname{cas}(-\alpha)\sin(\beta). \tag{5.6}$$

Proof. This property is demonstrated very simply by using the well-known trigonometric identities of the functions cos and sin. □

Using this property, we can rewrite the second sum of the equation (5.5) to get

$$\mathcal{H}(f)[k] = \mathcal{H}(f^0) + \cos\left(\frac{2\pi}{N}k\right)\mathcal{H}(f^1)[k] + \sin\left(\frac{2\pi}{N}k\right)\mathcal{H}(f^1)[-k].$$

We then define the operator χ_N^x, for $x \in \mathbb{R}$, as follows:

$$\forall a \in \mathbb{C}^N, \quad \chi_N^x a \overset{\text{def.}}{=} \left\{ a[j]\cos\left(\frac{2\pi jx}{N}\right) + a^\sharp[j]\sin\left(\frac{2\pi jx}{N}\right)\right\}_{j=0}^{N-1} \in \mathbb{C}^N. \tag{5.7}$$

We recall that $a^\sharp$ is the symmetrized vector, equation (3.27). We are now going to split the vector $\mathcal{H}(f)$ into its left and right parts, denoted $\mathcal{H}(f)_g$ and $\mathcal{H}(f)_d$. The relationship

$$\operatorname{cas}\left(\frac{2\pi}{N}(n+N/2)\right) = -\operatorname{cas}\left(\frac{2\pi}{N}n\right)$$

allows to obtain a very simple writing of the sought recurrence equation:

$$\mathcal{H}(f)_g = \mathcal{H}(f^0) + \chi_{N/2}^{1/2}\mathcal{H}(f^1), \tag{5.8}$$
$$\mathcal{H}(f)_d = \mathcal{H}(f^0) - \chi_{N/2}^{1/2}\mathcal{H}(f^1). \tag{5.9}$$

These equations make it possible to immediately implement a fast calculation algorithm which we call *FHT* for *Fast Hartley Transform*. The `fht` procedure performs this algorithm by recursive calls, and its program can be found in Section B.2.

Remark 41. The recurrence equations (5.8) and (5.9) show that the computation of $\mathcal{H}(f)$ actually requires the computation of two transforms of half size. However, because of the inverted term $a^\sharp$ present in the operator $\chi_N^x(a)$, it is difficult to use a butterfly scheme as for the FFT algorithm. In order to write an iterative algorithm, Ullmann, in [66], shows how we can do the computation by a double butterfly diagram, using four inputs.

We can show that a butterfly transformation of the FHT algorithm requires four multiplications and six real additions. With regard to the FFT algorithm, we find one addition and two complex multiplications, ie four additions and six multiplications. However, the loop performed to calculate the butterfly transforms of the FFT algorithm runs from 0 to $N-1$, while for the FHT algorithm, it is a loop between 0 and $N/2-1$. In the end, the FHT algorithm has the double advantage of requiring half the number of operations, and of using half the memory (we do not handle complex numbers).

Remark 42. (**Zero padding**). We have already explained in Paragraph 4.1.3, that it was possible to represent fairly faithfully the continuous DFT of a signal finished by *zero padding*. It is of course possible to do the same with the Hartley transform. The FHT algorithm, combined with a zero-padding procedure, therefore makes it possible to simply calculate a continuous Hartley transform. The more zeros are added, the more intermediate values of the transform are calculated, and the better the calculation precision. The figure 5.1 shows this process on a simple (trianglular) signal, and allows to compare the Fourier spectrum (in fact its real part) and its Harley spectrum. Since the Hartley spectrum is the difference between the real and imaginary parts of the Fourier spectrum, the similarities are not accidental!

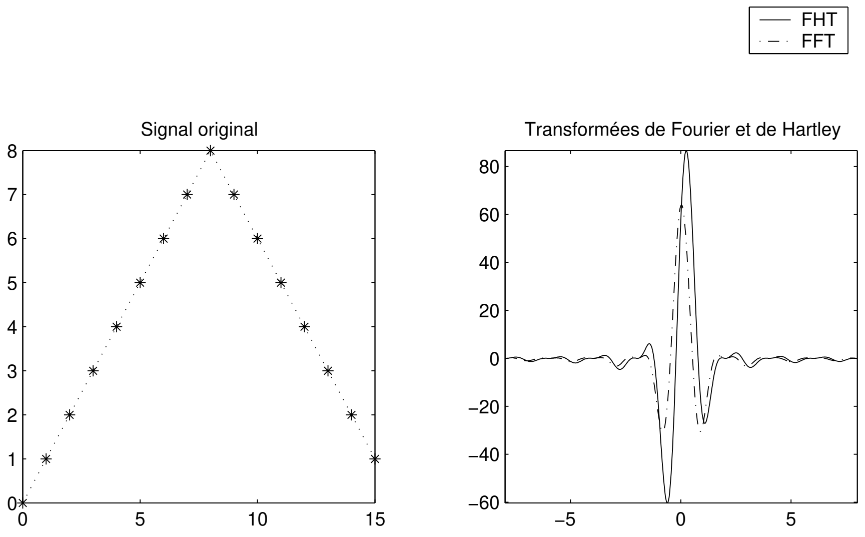

Figure 5.1: Comparison between the Hartley spectrum and the Fourier spectrum

5.1.3 Convolutional calculation by Hartley transform

The equations (5.4) show that it is possible to calculate a DFT of a real signal by means of a Hartley transform calculation, which avoids having to resort to multiplications and complex additions. In the same vein, one can establish a formula which makes it possible to calculate a product of convolution of two real signals using only Hartley transforms.

Proposition 47 (Convolution and Hartley transform)**.** *Let a and b be two real vectors of size N. We have*

$$\mathcal{H}(a*b) = \frac{1}{2}\left\{\mathcal{H}(a)\mathcal{H}(b) - \mathcal{H}(a)^\sharp\mathcal{H}(b)^\sharp + \mathcal{H}(a)\mathcal{H}(b)^\sharp + \mathcal{H}(a)^\sharp\mathcal{H}(b)\right\},$$

that is, for $n = 0, \dots, N-1$,

$$\mathcal{H}(a*b)[n] = \frac{1}{2}\left\{c[n](d[n]+d[-n]) + c[-n](d[n]-d[-n])\right\},$$

where we noted $c \overset{\text{def.}}{=} \mathcal{H}(a)$ and $d \overset{\text{def.}}{=} \mathcal{H}(b)$. We must of course consider the index $-n$ as taken modulo N.

Proof. After inverting the summations in the expression of $\mathcal{H}(a*b)[k]$ we find

$$\mathcal{H}(a*b)[n] = \sum_{l=0}^{N-1} a[l] \sum_{k=0}^{N-1} b[k] \operatorname{cas}\left(\frac{2\pi}{N} n(k+l)\right).$$

Then it suffices to use the relation (5.6) to obtain

$$\mathcal{H}(a*b)[n] = \mathcal{H}(b)[n] \sum_{l=0}^{N-1} a[l] \cos\left(\frac{2\pi}{N} nk\right) + \mathcal{H}(b)^{\sharp}[n] \sum_{l=0}^{N-1} a[l] \sin\left(\frac{2\pi}{N} nk\right).$$

To conclude, all that remains is to express the functions cos and sin using the function cas as follows:

$$\begin{cases} \cos(x) = \operatorname{cas}(x) + \operatorname{cas}(-x) \\ \sin(x) = \operatorname{cas}(x) - \operatorname{cas}(-x) \end{cases}.$$

The program `fht _convol`, which can be found in the Section B.2, realizes this convolution of real signals using the FHT algorithm. It is more economical, both in terms of computation time, and in terms of memory space used.

Remark 43. (**Auto correlation**). In the case where the signals a and b are equal, we can slightly optimize the implementation of the convolution calculation algorithm by writing

$$\mathcal{H}(a*a)[k] = c[k]c[-k] + \frac{1}{2}(c[k]^2 - c[-k]^2).$$

5.2 Z transform and applications

Our objective here is to give a relatively general framework to study transforms which have properties similar to those of the discrete Fourier transform, and in a way, generalize it (like for example the vector Z transform in Section 5.3, and the fractional Fourier transform in Section 5.4). To do this, we will be interested in the generating functions, which we will call transform in Z. After the definition of this transform, we will present an immediate application of the transform in Z to the construction of filters defined by recurrence equations.

The work of Wich [70] gathers a lot of information on the transform in Z. The link with generating series and holomorphic functions is discussed at length in the book of Demengel [24].

5.2.1 Definition and formal properties

Our goal is to make the link between transform in Z and whole series. We will therefore define the notion of transform in Z within the framework of signals of potentially infinite size. In practice, we will use the transform on signals of finite size, which allows us to not have to worry about possible convergence problems. We will therefore consider a sequence $f = \{f[n]\}_{n\in\mathbb{Z}} \in \mathbb{C}^{\mathbb{Z}}$. Only the study of the transfer function of a recursive filter (in Paragraph 5.2.2) requires the use of an infinite series. However, even then we will have an exact expression of the transform (as a rational fraction), which will save us from any problems.

Definition 28 (Transform to Z)**.** *Let $f \in \mathbb{C}^{\mathbb{Z}}$. We call transformed into Z of f the function*

$$\mathcal{Z}(f) : \begin{cases} D & \longrightarrow & \mathbb{C} \\ z & \longmapsto & \sum_{n=-\infty}^{+\infty} f[n] z^{-n} \end{cases}, \tag{5.10}$$

where D is the (possibly empty) set of points where the series converges.

Remark 44. (**Z transform and holomorphic functions**). The theory of *Laurent series* shows that the transform in Z is in fact defined at inside a crown of the type

$$C_\alpha^\beta \overset{\text{def.}}{=} \{z \in \mathbb{C};\ \alpha < |z| < \beta\} \qquad \text{for} 0 \leq \alpha < \beta,$$

where β can optionally be $+\infty$. We can refer for example to the book of Cartan [15] for a complete study of the decomposition of a holomorphic function in integer series and in Laurent series. The function $\mathcal{Z}(f)$ is therefore holomorphic inside its convergence disk. It is also often said that $\mathcal{Z}(f)$ is the generating series associated with the sequence f. As we will see later (by considering recursive filters), this notion makes it possible to represent in an elegant way certain sequences which are defined by induction. For an interesting talk on the resolution of recurrences by generating series, we can look at the reference work of Donald Knuth [36]. A simpler discussion can be found in [33]. The book [71] is an original work on the subject.

Example 5. Here are some simple examples.

1. Let the sequence $f \in \mathbb{C}^\mathbb{Z}$ be defined by $f[n] \overset{\text{def.}}{=} 0$ for $n < 0$ and $f[n] \overset{\text{def.}}{=} 1$ otherwise. We have

$$\mathcal{Z}(f)(z) = \sum_{n \geq 0} z^{-n} = \frac{1}{1 - z^{-1}},$$

the series being convergent in $C_1^{+\infty}$.

2. For $(a, b) \in \mathbb{C}^2$, we define the sequence $f \in \mathbb{C}^\mathbb{Z}$ by $f[n] \overset{\text{def.}}{=} a^n$ for $n \geq 0$ and $f[n] \overset{\text{def.}}{=} b^n$ otherwise. We have

$$\mathcal{Z}(f)(z) = \sum_{n \geq 0} \left(\frac{a}{z}\right)^n + \sum_{n \geq 0} \left(\frac{z}{b}\right)^n - 1 = \frac{1}{1 - a/z} + \frac{1}{1 - z/b} - 1,$$

the sum being convergent in $C_{|a|}^{|b|}$.

Proposition 48 (Properties of the Z transform)**.** *We denote by f and g two sequences of $\mathbb{C}^\mathbb{Z}$. Besides linearity, here are the important properties of the Z transform:*

(i) *linear convolution:*

$$\mathcal{Z}(f \star g) = \mathcal{Z}(f)\mathcal{Z}(g),$$

the series defining $\mathcal{Z}(f \star g)$ being at least convergent on the intersection of the convergence rings of $\mathcal{Z}(f)$ and $\mathcal{Z}(g)$ (if it is not empty).

(ii) *derivation:*

$$\mathcal{Z}\left(\{nf[n]\}_{n \in \mathbb{Z}}\right)(z) = -z\frac{d}{dz}\mathcal{Z}(f)(z).$$

Proof. We will prove the most important property, the convolution property (i). The absolute convergence on the intersection of the convergence domains allows us to write, for z in this intersection:

$$\begin{aligned}
\mathcal{Z}(f \star g)(z) = \sum_{n=-\infty}^{+\infty} (f \star g[n])\, z^{-n} &= \sum_{n=-\infty}^{+\infty} \sum_{m=-\infty}^{+\infty} f[n]g[nm]z^{-n} \\
&= \sum_{m=-\infty}^{+\infty} f[m]z^{-m} \left(\sum_{n=-\infty}^{+\infty} g[nm]z^{-(nm)} \right) \\
&= \mathcal{Z}(f)(z)\mathcal{Z}(g)(z).
\end{aligned}$$

Absolute convergence justifies the inversion of the two summations. □

Remark 45. There are also formulas allowing to integrate the function $\mathcal{Z}(f)$. They require the use of curvilinear integrals, as well as possible precautions on the behavior at infinity of f.

5.2.2 Recursive filters

We have already seen in Section 4.2, the definition of linear filters, as well as their use to modify signals appropriately. We are going to define here a new type of filters, named *recursive filters*, which also allow us to perform interesting operations on signals. The Z transform is very often used to represent the transfer function of recursive filters. It is indeed very practical for at least two reasons:

– the representation in the form of a function will allow the use of algebraic operations to modify the filter (sum, product, derivative, etc.).

– the complex representation in the form of module and argument (that is to say in polar coordinates) will allow you to create filters from scratch in an intuitive way.

Excellent books are available on digital signal processing in general, and using the Z transform for creating filters. We can cite among others [61] as well as [17].

Definition 29 (Recursive filter)**.** *Let $a = \{a_0, \dots, a_k\}$ and $b = \{b_1, \dots, b_l\}$ two vectors of complex numbers. They allow to define the recursive filter Φ_a^b operating on sequences $x \in \mathbb{C}^{\mathbb{N}}$ by defining $y = \Phi_a^b(x) \in \mathbb{C}^{\mathbb{N}}$ by*

$$\begin{aligned}\forall n \in \mathbb{Z}, \quad y[n] = a_0 x[n] + a_1 x[n-1] + \cdots + a_k x[nk] \\ + b_1 y[n-1] + \cdots + b_l y[nl].\end{aligned} \tag{5.11}$$

Remark 46. The equation (5.11) therefore defines the filter by induction. The filter uses, for the computation of $y[n]$, only already computed values of y as well as inputs of the vector x of index less than n. In accordance with the terminology already used (for convolutional filters), we say that the filter is causal. We thus obtain a simple algorithm to evaluate the action of the filter on a signal x. In the following, only x filters with finite support will be considered, and all the signals involved (in particular x and y) will be indexed from zero. Note that the computation of the first inputs of the vector y requires the knowledge of $y[-1]$, $y[-2], \dots, y[-l]$, as well as $x[n-1]$, $x[n-2], \dots, x[-k]$. By convention, it will be assumed, unless explicitly stated to the contrary, that these entries are null.

Remark 47. (**Link with convolution filters**). We have already defined, in Section 4.2, finite convolution filters. Note that in the case where the vector b is zero, the recursive filter is in fact a finite convolution filter. Recursive filters can be seen, from a theoretical point of view, as a generalization of linear convolution filters. In fact, we will see even in the following study, by the calculation of the transform in Z, that the recursive filters are filters of convolution, but of which the impulse response is infinite. This is why they are called *IIR* filters in the English literature (abbreviation of **I** nfinite **I** mpulse **R** esponse). From a pragmatic point of view, it is not the same: the uses of these two types of filtering are different, mainly because of the properties inherent both in their implementation, and in their impulse responses. Unlike a linear convolutional filter which can be computed quickly by FFT, a recursive filter will generally be limited to only a few recurrence terms (meaning that k and l will often be assumed to be small). These filters therefore make it possible, without having to calculate convolutions, to obtain very long responses (in theory infinite) for very low costs, of the order of ON) (where N denotes the size filtered vectors), assuming that k and l are small in front of N. On the other hand, the fact of having only a small number of coefficients at its disposal to create these filters makes them less manageable. Finally, it should be noted that the calculation of a filter by induction facilitates the propagation of rounding errors (since they are added as the calculations progress). To stop this phenomenon, we are often obliged to do double precision calculations.

The idea of this paragraph is to use the Z transform to define a recursive filter in a nicer way than by the recurrence equation (5.11). To do this, let's calculate the Z transform of this equation. After grouping the terms, we get

$$\mathcal{Z}(y)(z) = H(z)\mathcal{Z}(x)(z),$$

with

$$H(z) \overset{\text{def.}}{=} \frac{a_0 + a_1 z^{-1} + \cdots + a_k z^{-k}}{1 - b_1 z^{-1} - \cdots - b_l z^{-l}}.$$

We want to assert that H is the transform in Z of a certain transfer function h, in order to write the filter Φ_a^b as being the convolution filter Φ_h. Indeed, suppose that we succeeded in finding such a h. Using the convolution result 48 (i), we get

$$\mathcal{Z}(y) = H \cdot \mathcal{Z}(x) = \mathcal{Z}(h \star x) = \mathcal{Z}(h) \cdot \mathcal{Z}(x).$$

We can see that $H = \mathcal{Z}(h)$. The problem is to know if knowing H makes it possible to determine such h, in other words, if it is possible to calculate the inverse of the transform in Z. The answer is far from obvious. We know, thanks to *Cauchy formulas*, that a holomorphic function defined on a crown C_α^β admits a unique series expansion of Laurent inside this crown. However, it remains to be seen which crown we are dealing with. According to the choice of the domain of convergence, one obtains a sequence h different. So we need more information about the filter. By taking the Z transform of the recurrence equation (5.11), we kind of have "forgot" the boundary conditions, i.e. the values of $x[-1], \ldots, x[-k]$ as well as $y[-1], \ldots, y[-l]$. In practice, we have in fact simple information which makes it possible to find the domain of convergence which corresponds to the recurrence equation, and thus to find the sequence h starting from H.

- If we consider the recurrence equation written in (5.11), we see that the considered filter is causal (i.e. - say that the determination of $y[n]$ depends only on entries with indices less than n of $x[n]$). This implies that the impulse response is zero for negative indices, which results in the fact that the domain of convergence is outside a circle (it suffices to use the convergence criterion for a whole series, since l'we actually have an entire series in $1/z$).
- Most often, we want the filter to be stable in addition to being causal, which implies, as we will see in the proposition 49, that the domain of convergence must contain the unit circle.

Here is an example that shows the importance of specifying the domain of convergence.

Example 6 (Causality and domain of convergence). We consider a signal y which satisfies the difference equation

$$y[n] = \alpha y[n-1] + x[n], \tag{5.12}$$

where $x \in \mathbb{C}^\mathbb{Z}$ is the input signal. One thus obtains for the impulse response the transform in Z following:

$$H(z) = \frac{1}{1 - \alpha z^{-1}}.$$

In a natural way, the equation (5.12) defines a causal filter, and we can try to find its impulse response by the method we have just explained. The function H must therefore be considered as defined outside the circle of radius $|\alpha|$. We then obtain, by expanding the whole series fraction as a function of z^{-1},

$$\forall z \in C_{|\alpha|}^{+\infty}, \quad H(z) = \sum_{n \geq 0} \alpha^n z^{-n}.$$

Hence the value of the impulse response h:

$$\forall n < 0, \quad h[n] = 0,$$
$$\forall n \geq 0, \quad h[n] = \alpha^n.$$

It is therefore indeed a causal filter. If we want it to be stable, we check that we must also assume that $|\alpha| < 1$. However, another case can occur, if we consider that the function H is defined on $C_0^{|\alpha|}$. This time you have to do the following:

$$\forall z \in C_0^{|\alpha|}, \quad H(z) = \frac{-z/\alpha}{1 - z/\alpha} = \sum_{n<0} -\alpha^n z^{-n}.$$

We thus obtain the expression of h:

$$\forall n < 0, \quad h[n] = -\alpha^n,$$
$$\forall n \geq 0, \quad h[n] = 0.$$

The filter is therefore anti-causal, and stability dictates that $|\alpha| > 1$. The fact that it is anti-causal can be explained simply by rewriting the recurrence equation (5.12) in an inverse form (propagation of calculations in the other direction):

$$y[n-1] = \frac{1}{\alpha} y[n] - \frac{1}{\alpha} x[n].$$

Once we have succeeded in obtaining the value of h, we see that the filter Φ_a^b is indeed a convolution filter, but which remains a little special, since its impulse response, in the domain of the transform in Z, can be put in the form of a rational fraction.

We have just seen how we can represent the transfer function of a recursive filter thanks to its Z transform. This representation has the advantage of offering a compact and simple writing. This allows efficient calculations on filters, more simply than allowed by the representation in the form of convolution exposed in Paragraph 4.2.1. Before studying the use of the Z transform for the creation of new filters, let's see the relationship between the rational fraction representing a filter, and the stability of this filter.

Proposition 49 (Poles and stability). *Let $H(z) = \frac{A(1/z)}{B(1/z)}$ be the Z transform of the transfer function of a filter Φ_a^b (which is therefore causal). Then, this filter is stable if and only if all the poles of H are located inside the unit circle Γ.*

Proof. We have already said that for a causal filter, the series that defines H converges outside a certain circle, and the converse is true (it suffices to consider the expansion in $1/z$ of a holomorphic function defined in the neighborhood of infinity). Since Φ_a^b is of course causal, this remark applies. In addition, we saw in Paragraph 4.2.2, that an impulse response filter h was stable if and only if $\|h\|_{\ell^1} < +\infty$. If h is absolutely summable, we have the markup

$$\forall z \in \Gamma, \quad |H(z)| \leq \sum_{n=-\infty}^{+\infty} |h[n]z^{-n}| = \sum_{n=-\infty}^{+\infty} |h[n]| \overset{\text{def.}}{=} \|h\|_{\ell^1}.$$

We thus see that the condition $\|h\|_{\ell^1} < +\infty$ implies that the series which defines H is absolutely convergent on the circle Γ. The converse is true. The fact that $\|h\|_{\ell^1} < +\infty$ is therefore equivalent to the fact that the convergence region contains everything outside the circle Γ. However, the region of convergence cannot contain a pole. All the poles of the rational fraction H must therefore have a modulus strictly less than 1. Once again, the converse is true. □

5.2.3 Application to the construction of filters

In this paragraph, we will focus on using the Z transform for the creation of digital recursive filters. The process for creating a filter is to decide where the zeros and poles of the transfer function are located. As we have already seen, the poles must be contained in the unit circle for the filter to be causal and stable. For example, we can choose the locations identified in the figure 5.2, which leads to the expression of the transfer function:

$$H(z) = \frac{(z - e^{\imath\pi/4})(z - e^{-\imath\pi/4})}{(z - 0.9e^{\imath\pi/4})(z - 0.9e^{-\imath\pi/4})} \approx \frac{1 - 1.414z + z^2}{0.810 - 1.273z - z^2}.$$

This Z transform is shown in figure 5.3. From this expression, we immediately deduce the coefficients of the associated recursive filter:

$$\begin{array}{ccc} a_0 = 1 & a_1 \approx -1.414 & a_2 = 1 \\ b_0 \approx 1.273 & b_1 \approx -0.810. & \end{array}$$

Therefore, if we want to calculate the frequency response, two methods are available to us:

– calculate the frequency response directly from H: it suffices to consider the value of the transfer function on the unit circle, i.e. the function $\xi \mapsto H(e^{2\imath\pi\xi})$, for $\xi \in [0, 1]$. By inverse Fourier transform, the impulse response is deduced therefrom (the approximate calculation is done by sampling the frequency response, then by inverse FFT).

– calculate the impulse response using the filter recurrence equation and applying it for the impulse δ_0. We can then use a (discrete) Fourier transform to approximate the frequency response. To have sufficient precision, it will be necessary to calculate a fairly long approximate impulse response.

Figure 5.4 shows the impulse and frequency responses of the filter. They were calculated directly from the transfer function H shown in figure 5.3. In paragraph 5.3.1, we will present a fast calculation algorithm to

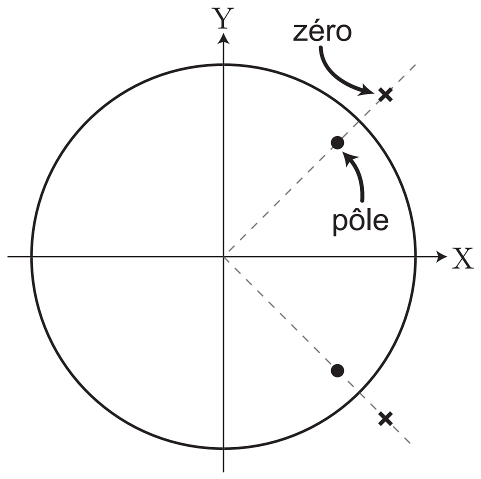

Figure 5.2: Positioning of poles and zeros

determine the value of the Z transform on some contours, and we will calculate the impulse response from the recurrence equation (5.11).

The previous example is far from being as anecdotal as it seems. Indeed, by breaking down the rational fraction H, we will be able to reduce ourselves to the case of simple filters, that is to say with at most two poles and two zeros. Here are two steps we can take.

– **Decomposition into products.** We can factorize the numerators and denominators into polynomials of degree 1 or 2 on $\mathbb{R}[X]$ (respectively of degree 1 over $\mathbb{C}[X]$). We thus obtain the writing of the filter Φ_a^b in the form of a cascade of filters:

$$\Phi_a^b = \Phi_{\alpha_1}^{\beta_1} \circ \cdots \circ \Phi_{\alpha_p}^{\beta_p},$$

where each α_i and β_i represents the coefficients of a polynomial of degree at most 2 (respectively at most 1). The filter Φ_a^b therefore corresponds to the serialization of a series of recursive filters of order at most 2 (respectively at most 1).

– **Decomposition into simple elements.** We can decompose the fraction H into the sum of simple elements over $\mathbb{R}[X]$ (respectively $\mathbb{C}[X]$). We then obtain the decomposition

$$\Phi_a^b = \Phi_{\alpha_1}^{\beta_1} + \cdots + \Phi_{\alpha_p}^{\beta_p},$$

where each α_i and β_i represents the coefficients of a polynomial of degree at most 2 (respectively at most 1).

In the case where we carry out decomposition on $\mathbb{C}[X]$, even if the signals are real, we will have to do the convolution calculations with complex numbers. Each of these decompositions provides a new way to implement the recursive filter, in addition to the naive implementation of the (5.11) equation. The 52 exercise applies these two methods to calculate the coefficients of an interpolation by splines.

5.2.4 Reconciliation with analog filtering

Before going into the details of the implementation of a discrete Z transform, we will try to establish a connection between the digital recursive filters and the analog filters. Analog filters are in a way the ancestors of modern digital filters, but are still used in many situations. It is therefore interesting to understand why recursive filters (which perform discrete transformations) make it possible to replace analog filters (which perform continuous transformations) within the "modern" framework of digital signal processing. Without going into the description of analog filtering, let's just say that it's a matter of passing a DC signal through

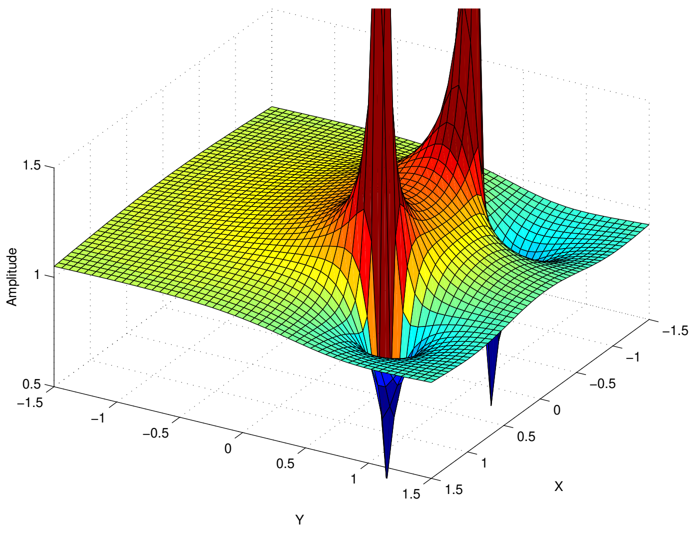

Figure 5.3: Z transform of the filter

a set of electronic components, so that the output signal is linked to the input signal by a linear differential equation. The digital filter then behaves like a dynamic system governed by a differential equation.

The difference equation (5.11) is in fact the analog discrete of the differential equations that dynamical systems must satisfy. We can take the example of a *circuit RLC* (see the diagram in figure 5.5). We then have the following differential equation which connects V_e and V_s, the input and output voltages:

$$\frac{dV_e}{dt} = \frac{1}{RC}V_s + \frac{dV_s}{dt} + \frac{L}{R}\frac{d^2V_s}{dt^2}. \tag{5.13}$$

We can consider this circuit as an analog filter. As we have developed the Z transform to study discrete filters, we will introduce another generalization of the Fourier transform, continued this time, to study analog filters. This is the *Laplace transform*, which is defined as follows.

Definition 30 (Laplace transform)**.** *For a function $f : \mathbb{R} \to \mathbb{C}$, we formally define its Laplace transform through*

$$\forall s \in D, \quad \mathcal{L}(f)(s) \overset{\text{def.}}{=} \int_{t\in\mathbb{R}} f(t)e^{-st}\mathrm{d}t.$$

The function $\mathcal{L}(f)$ is defined on a domain D where the integral converges.

By taking precautions on the fields of definition of the functions considered, one can define the transfer function of the analog filter, in the field of Laplace:

$$K(s) \overset{\text{def.}}{=} \frac{\mathcal{L}(V_s)(s)}{\mathcal{L}(V_e)(s)} = \frac{s}{\frac{1}{RC} + s + \frac{L}{R}s^2}.$$

We have simply calculated the Laplace transform of the two members of the equation (5.13) here. We used the fact that the Laplace transform transforms the derivation into the multiplication by the parameter s.

We are now interested in the problem resulting from the discretization, at regular time intervals Δ, of the studied signals. We obtain discrete signals $\tilde{V}_e$ and $\tilde{V}_s$, which satisfy the difference equation

$$\begin{aligned}\frac{1}{\Delta}\left(\tilde{V}_e[n] - \tilde{V}_e[n-1]\right) &= \frac{1}{RC}\tilde{V}_s[n] + \frac{1}{\Delta}\left(\tilde{V}_s[n] - \tilde{V}_s[n-1]\right)\\ &\quad + \frac{L}{R\Delta^2}\left(\tilde{V}_s[n] + \tilde{V}_s[n-2] - 2\tilde{V}_s[n-1]\right).\end{aligned}$$

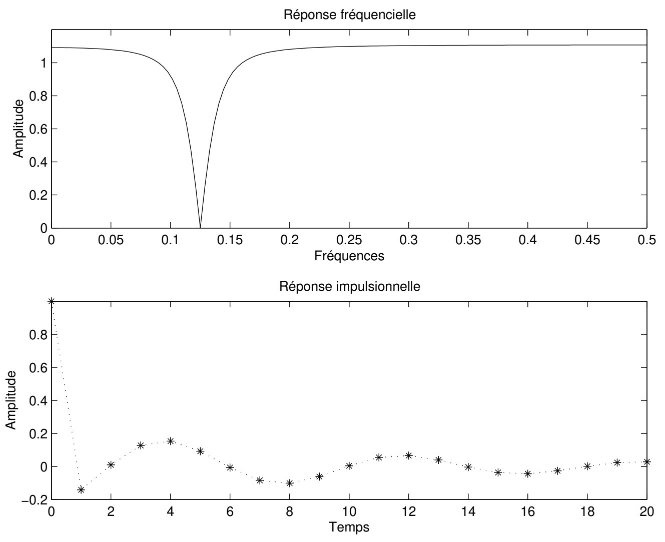

Figure 5.4: Frequency and impulse response of the filter

The resolution of the original differential equation is thus replaced by an explicit finite difference scheme. Of course, one could have chosen other methods to calculate in an approximate way the derivatives involved. That would have led to a slightly different equation. The exercise 51 proposes to calculate some finite difference equations for an integrating analog circuit.

From a purely discrete point of view, we obtain a recursive filter, which can be calculated using a computer (and no longer an electrical circuit as was the case for the RLC filter). We can then calculate the transfer function in the domain of the Z transform to study this filter:

$$H(z) \overset{\text{def.}}{=} \frac{1-z}{\left(\frac{\Delta}{RC}+1+\frac{L}{\Delta R}\right)-\left(1+\frac{2L}{R\Delta}\right)z-\frac{2L}{R\Delta}z^2}.$$

The Z transform is in a way the tool that allows us to study recursive filters, while the Laplace transform allows us to study their continuous cousins, the analog filters. Thus, the principles of construction of digital filters by the use of the transform in Z (placement of poles and zeros, placing in series of filters, etc.) also apply to the creation of analog filters, provided that use the Laplace transform.

5.3 Vectorial Z Transform

The presentation we have just made of the Z transform is above all theoretical. In order to actually calculate the values of a transformed function in Z, we need to sample and do a finite number of calculations. This is what we will do in this paragraph, by defining a new transform, which we call Z *vector* transform. The resulting algorithm is called the *chirp* algorithm. It was discovered for the first time, in the (more restricted) framework of the TFD by Bluestein, and is well explained in the article [64]. Some aspects of programming the Z transform are discussed in [2].

5.3.1 Discrete calculation algorithm

The Z transform, even operating on discrete and finite samples, nonetheless remains a function of the complex variable z. The fact is that a computer does not know how to work directly with such functions (except for software such as MAPLE which can do certain formal operations). We therefore need a way to numerically evaluate the value of the Z transform at certain points, and this quickly. To build this algorithm,

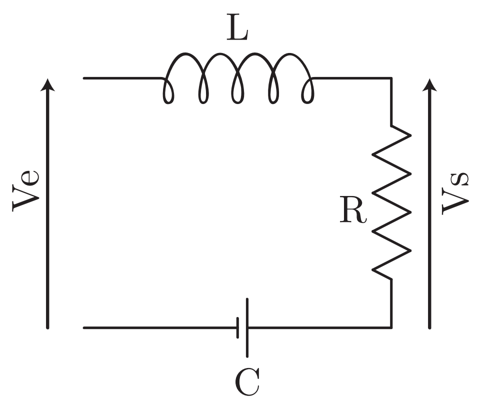

Figure 5.5: RLC circuit

we will introduce a transform dedicated to the computation of $\mathcal{Z}(f)$ in a sufficient number of points (as many as there are points in the original sample).

Definition 31 (Transform to vector Z)**.** *We fix $z \in \mathbb{C}$. For a vector $f \in \mathbb{C}^N$, we define the transformed into vector Z (at the point z) by*

$$\mathcal{G}_z(f) \overset{\text{def.}}{=} \{\mathcal{Z}(f)(z^n)\}_{n=0}^{N-1} = \left\{ \sum_{k=0}^{N-1} f[k] z^{-kn} \right\}_{n=0}^{N-1}. \tag{5.14}$$

Remark 48. The vector obtained can be seen as the calculation of the value that the Z transform takes along a curve drawn in the complex plane. If the point z is taken from modulus 1, this curve will be the unit circle, otherwise, it will be a spiral.

Let $z \in \mathbb{C}$ fixed. To build an efficient computation algorithm for $\mathcal{G}_z(f)$, we will use the relation

$$\forall (n,\, k), \quad nk = \frac{1}{2}\left(n^2 + k^2 - (nk)^2\right).$$

Applying it to the definition equation (5.14), we get

$$\mathcal{G}_z(f)[n] = z^{-\frac{n^2}{2}} \sum_{k=0}^{N-1} f[k] z^{-\frac{k^2}{2}} z^{\frac{(nk)^2}{2}} = z^{-\frac{n^2}{2}} (\tilde{f} \star g)[not],$$

where we denote by g the vector defined by

$$\forall k \in \{-N+1,\, N-1\}, \quad g[k] \overset{\text{def.}}{=} z^{-\frac{k^2}{2}},$$

and f the vector

$$\forall k \in \{0, \ldots, N-1\}, \quad \tilde{f}[n] \overset{\text{def.}}{=} f[k] z^{-\frac{k^2}{2}}.$$

Be careful that convolution is a linear convolution between a vector of size N and a vector of size $2N-1$. Using the method described in Section 3.3.3, one can compute an acyclic convolution very quickly by replacing it with a larger cyclic convolution. More precisely, the vectors to be combined being of size N and $2N-1$, it is in theory necessary to calculate a cyclic convolution of size $3N-2$. In fact, to use a Cooley-Tukey "classic" FFT algorithm, we add zeros to reach a size $M = 2^k$ just after $3N-2$. However, we can do much better (size $2N-1$) by exploiting the fact that $g[k] = g[-k]$. This is explained in the exercise 50 and gives rise to the procedure Matlab `czt` (for **C** hirp **Z** **T** ransform) (see the correction of the exercise 50). We can thus calculate the vector Z transform in a time of the order of $ON\log(N)$).

The "chirp" approach therefore consists in replacing a transform calculation by a convolution calculation. Another trick (using a finite field) allows you to achieve a similar result (when N is a prime number). This is the subject of the exercise 53.

To finish, let's use the computation algorithm we just built to draw vector Z transforms of a recursive filter. We have chosen the filter whose poles and zeros are placed on the figure 5.2. We calculated the impulse response of the filter by directly using the recurrence equation (5.11). We have chosen two contours, which correspond respectively to $z = e^{\frac{2\imath\pi}{N}}$ (unit circle) and $z = 1.001e^{\frac{2\imath\pi}{N}}$ (spiral). The first contour is used to calculate the impulse response (we find the figure 5.4). Indeed, calculating the transform in Z for $z = e^{\frac{2\imath\pi}{N}}$ amounts to calculating a DFT (with a substantial time saving if N is not a power of 2). For the second contour, on the other hand, we see that the second "jump" is less marked, because the spiral is farther from the second pole than the unit circle is.

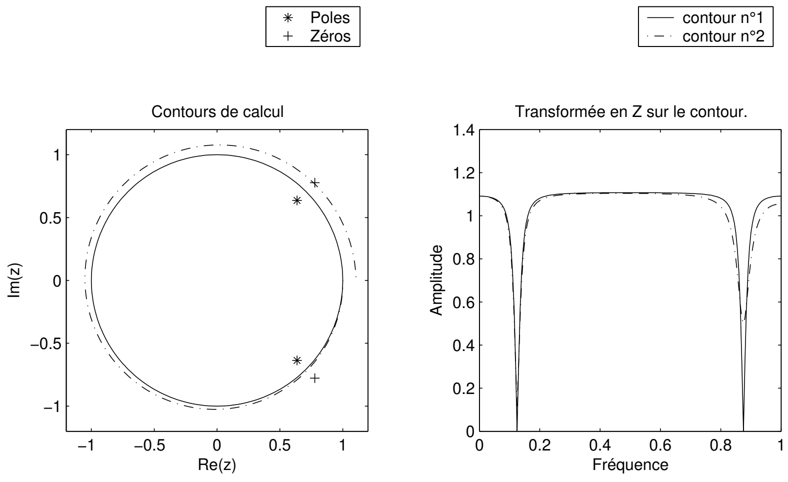

Figure 5.6: Transformed into Z along two contours

5.3.2 Applications to the discrete Fourier transform

This paragraph makes the connection between the vector Z transform and the DFT. In particular, we are going to see how this approximation makes it possible to perform DFT calculations in the case where the length N of the signals is not a composite number of the type $N = 2^p$ (case where the FFT algorithm is very effective).

We can indeed see the discrete Fourier transform as a particular case of vector Z transform. For that, we choose $z = \omega_N \overset{\text{def.}}{=} e^{\frac{2\imath\pi}{N}}$ and we obtain, for a vector $f \in \mathbb{C}^N$,

$$\mathcal{F}(f) = \mathcal{G}_{\omega_N}(f).$$

However, one of the strengths of the *chirp transform* algorithm presented in the previous paragraph is that it can be applied to any positive integer N. Unlike the FFT algorithm, it is not restricted to only integers N for which we know a "good" factorization (the simplest example is $N = 2^p$), as explained in Paragraph 3.2.4. We can even apply the *chirp transform* algorithm for transforms whose length N is a prime number, whereas in this case it is impossible to reduce the computation time by an FFT approach!Of course, this algorithm requires a number of additional calculations, among others:

– addition of zeros to transform acyclic convolution into circular convolution. In fact, we are going to calculate FFTs of length $M = 2^k$ just after $2N - 1$.

– calculation of two FFTs (or even three taking into account the vector g) to calculate a circular convolution. However, in the case where we have to calculate a DFT of length N (and where we cannot replace these calculations by a larger transform), this algorithm constitutes an advantageous alternative compared to the calculation naive. However, it should be kept in mind that in a good number of applications, one can be satisfied with computing a transform at the frequencies $\{k/N'\}_{k=-N'/2}^{N'/2-1}$ rather than $\{k/N\}_{k=-N/2}^{N/2-1}$, and therefore this approach is to be avoided!

Remark 49. The worst case that can arise for the *chirp* algorithm for calculating a DFT is $2N-1 = 2^p+1$. We must indeed calculate 3 FFT of size $2^{p+1} \approx 4N$ for the calculation of the convolution (we have $N' = 2^{p+1}$ and we must double the size because the convolution is not circular). We therefore see that we carry out about 12 times more calculations than for an FFT of size 2^p ...

5.4 Fractional Fourier transform

In this section, we will study the *fractional Fourier transform*. It is simply a question of considering intermediate frequencies when evaluating the sum that defines the DFT. Certainly, we lose many properties of the discrete Fourier transform (convolution, inversion, etc.), since we no longer use the exponential characters $e_n : k \mapsto e^{\frac{2\imath\pi}{N}kn}$. However, we will see that we have a fast calculation algorithm, which makes this transform easy to use. A relatively complete presentation of the fractional Fourier transform is given in [6].

5.4.1 Definition and calculation algorithm

Here is the definition, very natural, of this new transform.

Definition 32 (Fractional Fourier transform)**.** *Let $\alpha \in \mathbb{R}$. We define the fractional Fourier transform $G(f, \alpha)$ of a vector $f \in \mathbb{C}^N$ by*

$$\forall k \in \{0, \ldots, N-1\}, \quad G(f, \alpha)[k] \overset{\text{def.}}{=} \sum_{n=0}^{N-1} f[n] e^{-\alpha \frac{2\imath\pi}{N} kn}.$$

Remark 50. (**Link with DFT**). We see that if $\alpha = 1$, we find the discrete Fourier transform. For $\alpha = -1$, we find the inverse discrete Fourier transform (up to a factor of $1/N$). It is in this sense that the fractional transform generalizes the usual TFD.

In order to build a computational algorithm, we will make the link with the transform in Z, defined by the equation (5.14). We indeed note that in the case where $z = e^{\frac{2\imath\alpha\pi}{N}}$, the two transforms coincide. By using the chirp transformation algorithm in Z, we will therefore be able to calculate the fractional Fourier transform quickly.

The fractional Fourier transform does not really have a simple intuitive meaning. We will see in the next paragraph that it allows to calculate intermediate values of the Fourier transform of a discrete signal. It will thus prove to be effective for analyzing signals whose periodicity is unknown. However, the fractional Fourier transform is not what one could call a partial transform, as we were able to define in Paragraph 3.5.2, and in the exercise 30. Indeed, the composite of two fractional transforms with $\alpha = \frac{1}{2}$ does not give back the classical Fourier transform.

The figure 5.7 shows the fractional Fourier transforms $G(f, \alpha)$ of a step function for several values of α. The exercise 55 proposes to calculate the fractional Fourier transforms of an image.

5.4.2 Analysis of signals with non-integral periodicity

The discrete Fourier transform is a powerful tool for analyzing physical data collected by sensors or other more or less sophisticated methods. However, all this analysis made using the Fourier transform assumes that the recorded signal is periodic, and above all that its period is a divisor of the length over which the signal was sampled. However, in most applications, we are far from being able to know *a priori* this period. In some

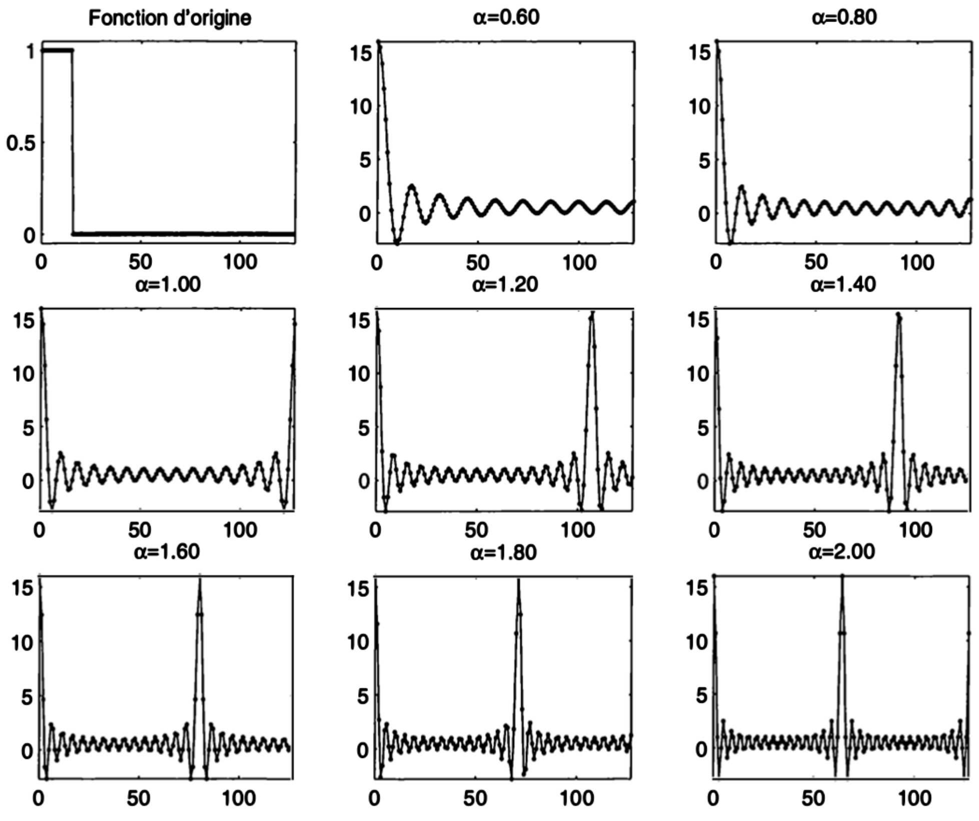

Figure 5.7: Fractional Fourier transforms of a 1D function

cases, information is available on the value of this period. A good example is meteorological data. We know that the period of rotation of the earth around the sun is 365.2422$days.However, eveninthisfavorablecase, thedataacquiredis$
365 $dataperyear.Asaresult, thesignalspectrumisgoingtobeunusuallycomplicated, muchmoresothanifwehadbeenabletoobtai$
365.2422 samples per year.

We are therefore faced with a double problem.

– How to determine, from a given spectrum, the true period of the signal?

– Once this period is known, how to modify the original spectrum so that it corresponds to data sampled according to the correct period?

We need to build an algorithm that automates these two tasks, and does the math quickly. We will see that the use of the fractional Fourier transform and its fast algorithm solves this problem.

To get an idea of how to find the period, it is interesting to study what happens on a monochromatic sample, that is, on a sinusoid. Either the signal

$$f = \left\{ e^{\beta k \frac{2\imath\pi}{N}} \right\}_{k=0}^{N-1},$$

where β is a real number. If β is not an integer, we are in the presence of a signal that we know to be periodic (of period N/β), but the sampling of which does not reflect the periodicity. The figure 5.8 shows the spectrum obtained for $\beta = 4.63$. We have plotted both the discrete Fourier transform and the continuous Fourier transform (which is approximated by adding a significant number of zeros at the end of the signal before calculating the DFT).

Although the spectrum is not equal to that which a monochromatic wave should have (we should obtain

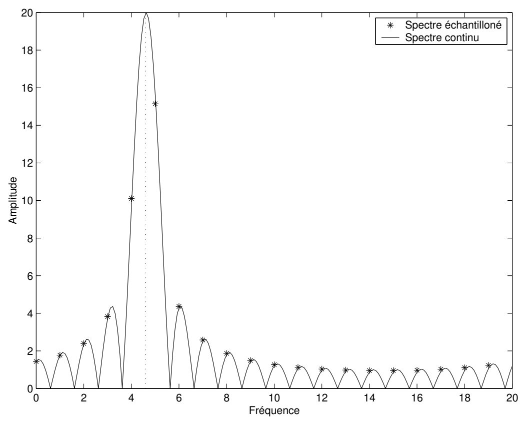

Figure 5.8: Spectrum of a badly sampled sinusoid

a single line placed at the frequency 4.63), it presents a maximum from which the values of the transform decrease. In the drawing, it is easy to determine the exact value of the abscissa of this maximum, which corresponds to the sought frequency and which allows us to determine the period. However, it is much too expensive to calculate the continuous Fourier transform to have a good approximate value of the period. Without knowing this continuous Fourier transform, we are nevertheless able to determine the period to within one unit. Here we see that this sought frequency is between 4 and 5. Let us denote by b the integer immediately less than β. To calculate β with more precision, we will calculate in a finer way the spectrum of f in the interval $[b,\, b+1]$. Let δ therefore be a subdivision step. We are looking for the value of the Fourier transform of f at the intermediate frequencies $b,\, b+\delta, \ldots,\, b+m\delta \geq b+1$, which amounts to doing the calculation:

$$\forall k \in \{0, \ldots,\, m\}, \quad \hat{f}(b+k\delta) = \sum_{n=0}^{N-1} f[n] e^{-\frac{2\imath\pi}{N} n(b+k\delta)} = G(\tilde{f},\, \delta)[k],$$

where $\tilde{f}$ is the vector

$$\tilde{f} = \left\{ f[n] e^{-\frac{2\imath\pi}{N} nb} \right\}_{n=0}^{N-1}.$$

The figure 5.9 (a) shows a periodic signal (noisy) whose sampling length is unfortunately not chosen to be multiple of the period. In figure (b), we can see the spectrum of this signal, which has a peak at $b = 2$. Figure (c) zooms in on the frequency window [2, 3], and we can see precisely that $\beta = 2.23$.

Now that we have precisely calculated the frequency β which determines the period, we would like to modify the spectrum so that this frequency β takes the place of a frequency actually calculated by the DFT, by the occurrence the frequency b. We hope that the coefficients will decrease more quickly, as is the case with a well sampled monochromatic wave. Let $\alpha = \frac{b}{\beta}$, which is slightly smaller than 1. Let r be the integer closest to $N\alpha$. In order for the frequency β to become the frequency b, we need to calculate a Fourier transform by multiplying the frequencies by $\frac{1}{\alpha}$, which leads to calculate

$$\forall k \in \{0, \ldots,\, r-1\}, \quad F_k \overset{\text{def.}}{=} \hat{f}\left(\frac{k}{\alpha}\right) = \sum_{n=0}^{r-1} f[n] e^{-\frac{2\imath\pi}{N\alpha} kn} = G\left(f,\, \frac{r}{N\alpha}\right)[k],$$

where care must be taken to complete the vector f with zeros so that it reaches the length r. In the case where $N\alpha = r$ is an integer, and the signal f is monochromatic as we have already defined, we obtain

$$\forall k \in \{0, \ldots,\, r-1\}, \quad \hat{f}\left(\frac{k}{\alpha}\right) = \sum_{n=0}^{r-1} e^{\frac{2\imath\pi}{N}\beta n} e^{-\frac{2\imath\pi}{r} nk} = r\delta_b^k.$$

We therefore obtain the desired result, namely a perfect correction of the spectrum. If $N\alpha$ is not an integer, the correction is however not perfect; we can calculate the error obtained for a monochromatic wave:

$$\forall k \in \{0,\ldots,r-1\},\ k \neq b, \quad F_k \overset{\text{def.}}{=} \left|\hat{f}\left(\frac{k}{\alpha}\right)\right| = \left|\frac{1-e^{-\frac{2\imath\pi}{N\alpha}n(bk)}}{1-e^{-\frac{2\imath\pi}{N\alpha}(bk)}}\right|$$
$$= \left|\frac{sin\left(\frac{\pi s(bk)}{N\alpha}\right)}{\sin\left(\frac{\pi(bk)}{N\alpha}\right)}\right|,$$

where we noted $s \overset{\text{def.}}{=} r - N\alpha$. By noting that $|s| < \frac{1}{2}$, we see that

- we have $F_b = r$, as in the case where $N\alpha$ is integer.
- when k is close to b, in the first order, we have $|F_k| \equiv |s|$ which is bounded by $\frac{1}{2}$, so much smaller than F_b.
- When k is far from b, on the other hand, the situation is less favorable (the denominator of $|F_k|$ may become small). However, using the fact that

$$\forall x \in [0,\pi/2], \quad \frac{2x}{\pi} \leq \sin(x) \leq x$$

  we show that $|F_k|$ remains bounded by $\frac{N}{\pi\beta}$ or $\frac{N}{\pi(N-\beta)}$.

In the case where the signal is not monochromatic, the decrease will of course be weaker, and the improvement less visible. Figure 5.9 (d) shows the effect of this frequency adjustment, and we observe a marked decrease in the values of the transform outside the peak at the frequency 2. We have obtained a frequency representation of the analyzed function that is much more efficient and able to be used for subsequent processing.

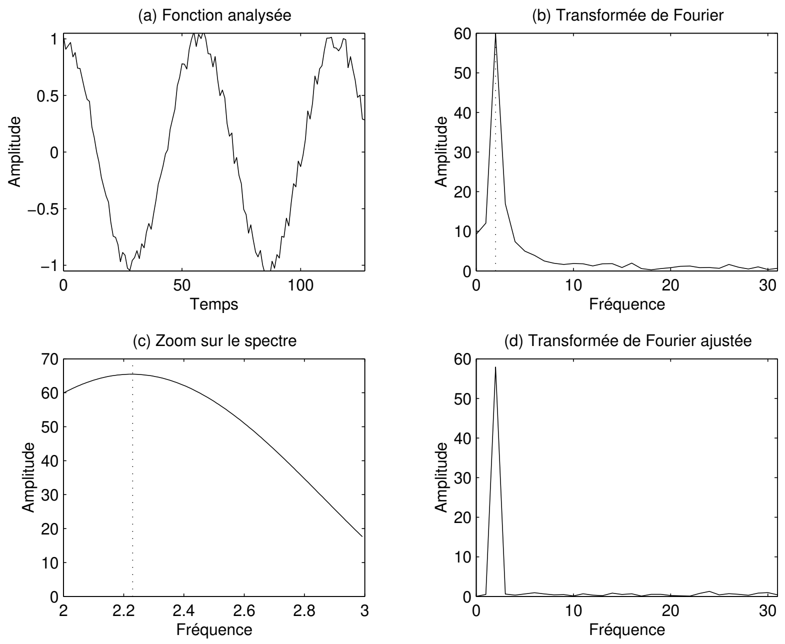

Figure 5.9: Spectrum adjustment by fractional Fourier transform

5.5 Exercises

Exercise 45 (Eigenvectors and Hartley transform)**.** *What are the eigenvalues of the Hartley transform defined in Section 5.1? Inspired by the construction of Paragraph 3.5.2, propose a method to obtain eigen-*

vectors for each of the eigenvalues. Based on what was done in the paragraph, deduce the construction of an intermediate Hartley transform $\mathcal{H}^\lambda$. The figure 5.10 shows different intermediate transforms. We can make the comparison with the figure 3.9.

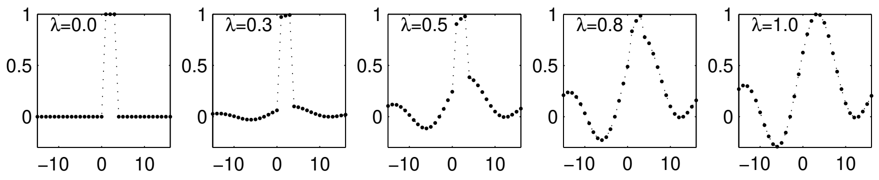

Figure 5.10: Real parts of intermediate transforms $\mathcal{H}^\lambda(f)$ for $\lambda \in [0, 1]$

Exercise 46 (Generalized Hartley transform)**.** *Noting that*

$$\operatorname{cas}(x) = \sqrt{2}\cos\left(x - \frac{\pi}{4}\right), \tag{5.15}$$

we can define, for $f \in \mathbb{R}^N$, a Hartley transform generalized by

$$\forall n \in \{0, \ldots, N-1\}, \quad \mathcal{H}_\lambda(f)[n] \overset{\text{def.}}{=} \sum_{k=0}^{n-1} f[k] \cos\left(\frac{2\pi}{N} nk + \lambda\right),$$

where λ is an actual parameter in $[0, \pi[$. For $\lambda \notin \left\{0, \frac{\pi}{2}\right\}$, show that this transformation is bijective, and that its inverse is given by

$$(\mathcal{H}_\lambda)^{-1} = \frac{2}{\sin(2\lambda)} \mathcal{H}_{\frac{\pi}{2}-\lambda}. \tag{5.16}$$

Figure 5.11 shows the generalized transform of the triangular signal in figure 5.1 (left), for λ varying between 0 and 2π.

Exercise 47 (Interpolation and Hartley transform)**.** *In the exercise 27, we saw how we can interpolate a sampled signal thanks to the discrete Fourier transform. Still using the zero padding technique, explain why the discrete Hartley transform also allows you to interpolate sampled values. Show that in fact the interpolations obtained by Fourier and by Hartley are the same.*

Exercise 48 (Hartley transform 2D)**.** *Let $f \in \mathbb{R}^{N_1 \times N_2}$. We define its two-dimensional Hartley transform $\mathcal{H}(f)$ by*

$$\mathcal{H}(f)[n_1, n_2] \overset{\text{def.}}{=} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} f[k_1, k_2] \operatorname{cas}\left(\frac{2\pi}{N_1} k_1 n_1\right) \operatorname{cas}\left(\frac{2\pi}{N_2} k_2 n_2\right),$$

where $n_1 \in \{0, \ldots, N_1 - 1\}$ and $n_2 \in \{0, \ldots, N_2 - 1\}$. Show that we have the following inversion formula:

$$f = \frac{1}{N_1 N_2} \mathcal{H}(\mathcal{H}(f)).$$

How to quickly calculate this transform using the FHT algorithm?

Exercise 49 (Hartley transform and finite fields)**.** *Extend the definition of the Hartley transform within the framework of finite fields (then rings in which we have a root $n^{\text{ième}}$ main). If ζ represents a n^{ith} root of the unit, we can use $\frac{\zeta^2+1}{2\zeta}$ to replace $\cos\left(\frac{2\pi}{n}\right)$. Write the corresponding FHT algorithm.*

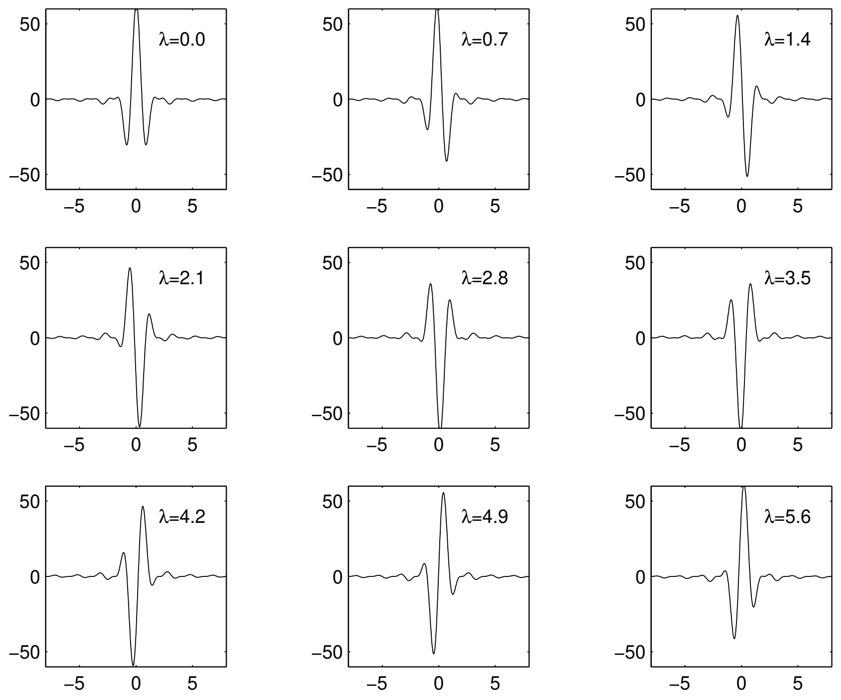

Figure 5.11: Generalized Hartley transform for different values of λ

Exercise 50 (Chirp transformation and Toeplitz matrices)**.** *In this exercise, we use the description of the chirp algorithm for the calculation of the Z transform, in order to find a matrix formulation. Recall the definition of the chirp algorithm. It consists in calculating the transform in Z of a vector $f \in \mathbb{C}^N$ via a vector $\{y[n]\}_{n=0}^{N-1}$ defined by*

$$\forall n \in \{0, \ldots, N-1\}, \quad y[n] \overset{\text{def.}}{=} z^{\frac{n^2}{2}} \mathcal{G}_z(f)[n] = \sum_{n=0}^{N-1} h[k]g[nk],$$

where we noted

$$\forall k \in \{0, \ldots, N-1\}, \quad h[k] \overset{\text{def.}}{=} f[k] z^{-\frac{k^2}{2}},$$

and

$$\forall k \in \{-N+1, N-1\}, \quad g[k] \overset{\text{def.}}{=} z^{-\frac{k^2}{2}}.$$

1. *Show that y can be calculated as a matrix product $y = Gh$, where G is what we call a matrix of Toeplitz, i.e. a matrix constant along each of its diagonals. For example, show that for $N = 3$, we have*

$$G = \begin{pmatrix} g[0] & g[1] & g[2] \\ g[1] & g[0] & g[1] \\ g[2] & g[1] & g[0] \end{pmatrix}.$$

2. *In general, a Toeplitz matrix T of size $m \times n$ is written*

$$T \overset{\text{def.}}{=} \begin{pmatrix} t_{m-1} & t_m & \ldots & t_{m+n-2} \\ t_{m-2} & t_{m-1} & \ldots & t_{m+n-3} \\ \vdots & \ldots & \ldots & \vdots \\ t_0 & \ldots & \ldots & t_{m-1} \end{pmatrix}.$$

It is therefore entirely determined by its first column, which we denote in the form $t_c \overset{\text{def.}}{=} (t_0, \dots, t_{n-1})^\top$ and its first row denoted $t_l \overset{\text{def.}}{=} (t_m, \dots, t_{m+n-2})^\top$ (we do not take the element t_{m-1}). We consider the vector

$$c \overset{\text{def.}}{=} (t_c, 0, \dots, 0, t_l)^\top \in \mathbb{C}^M.$$

where M is the power of two immediately after $n + m - 1$. We denote by C circulating matrix associated with c, as defined in exercise 26. Where can we find the matrix T inside the matrix C? How to calculate a product Tx, where $x \in \mathbb{C}^n$, using the matrix C? Deduce an algorithm allowing to calculate Cx quickly, using the FFT algorithm.

3. *Apply the previous construction to the G matrix. Show for example that in the case $N = 3$, we obtain the following matrix C:*

$$C = \begin{pmatrix} g[0] & g[1] & g[2] & 0 & 0 & 0 & g[2] & g[1] \\ g[1] & g[0] & g[1] & g[2] & 0 & 0 & 0 & g[2] \\ g[2] & g[1] & g[0] & g[1] & g[2] & 0 & 0 & 0 \\ 0 & g[2] & g[1] & g[0] & g[1] & g[2] & 0 & 0 \\ 0 & 0 & g[2] & g[1] & g[0] & g[1] & g[2] & 0 \\ 0 & 0 & 0 & g[2] & g[1] & g[0] & g[1] & g[2] \\ g[2] & 0 & 0 & 0 & g[2] & g[1] & g[0] & g[1] \\ g[1] & g[2] & 0 & 0 & 0 & g[2] & g[1] & g[0] \end{pmatrix}.$$

Deduce an algorithm allowing to calculate y, then $\mathcal{G}_z(f)$ quickly.

Exercise 51 (Quadrature methods and recursive filters)**.** *We consider an integrating circuit, which connects the input voltage $x(t)$ and output voltage $y(t)$ by the equation*

$$y(t) = \int_0^t x(\tau)\mathrm{d}\tau.$$

What is the transfer function of this filter in the Laplace domain? We want, from this analog filter, to create a digital recursive filter. We consider the following quadrature methods, for a function $f : \mathbb{R} \to \mathbb{R}$ continue:

$$(M_0) \qquad \int_0^1 f(t)\mathrm{d}t \approx f(0), \tag{5.17}$$

$$(M_1) \qquad \int_0^1 f(t)\mathrm{d}t \approx \frac{1}{2}\left(f(0) + f(1)\right), \tag{5.18}$$

$$(M_2) \qquad \int_0^1 f(t)\mathrm{d}t \approx \frac{1}{6}f(0) + \frac{2}{3}f(1/2) + \frac{1}{6}f(1). \tag{5.19}$$

These methods are called the rectangles on the left method, the trapezoids method, and the Simpson method, respectively. Considering a discretization of the signals x and y of steps Δ, give the recurrence equations obtained by using each of the three methods. What are the associated transfer functions (for the Z transform)?

Exercise 52 (Spline and recursive filters)**.** *We use the notations of the exercise 44. This is to explain how we can calculate the interpolation coefficients $c[k]$ by means of a recursive filter. We will mainly focus on the example of cubic splines, and we leave it to the reader to practice on other examples, then to generalize the method. To use an expression of β_d^n, we can refer to the article of Unser[67].*

1. *Calculate β_d^3, then show that its transform in Z is worth*

$$\mathcal{Z}(\beta_d^3)(z) = \frac{z + 4 + z^{-1}}{6}.$$

2. *Recall that we have $c = \Phi_d^3 * u_d$, where Φ_d^3 is defined by $\hat{\Phi}_d^3 = 1/\hat{\beta}_d^3$. Decompose the rational fraction $\mathcal{Z}(\Phi_d^3)$ into simple elements. How can we calculate the coefficients c of the indirect interpolation using two recursive filters?*

3. *We assume that we know the signal $u_d[k]$ for $k \in \{0, \ldots, K-1\}$. Show that we can organize the calculations of c as follows:*

$$\begin{cases} \forall k \in \{1, \ldots, K-1\}, & c^+[k] = u_d[k] + b_1 c^+[k-1] \\ \forall k \in \{0, \ldots, K-2\}, & c^-[k] = u_d[k] + b_1 c^-[k+1] \\ \forall k \in \{0, \ldots, K-1\}, & c[k] = b_0(c^+[k] + c^-[k] - u_d[k]) \end{cases} .$$

We will specify the values of the constants b_0 and b_1. What values should $c^+[0]$ and $c^-[K-1]$ be given? It will be possible for example to ensure that the produced signal can be prolonged by mirror symmetry in $K-1$ and 0 (to avoid discontinuities).

4. *Resume the two previous questions by exploiting a product decomposition of the form*

$$\mathcal{Z}(\Phi_d^3) = \frac{A}{(1 - \alpha z^{-1})(1 - \alpha z)}.$$

Exercise 53 (Chirp transformation and finite field)**.** *Let p be a prime number. We denote by g a generator of the group $\mathbb{F}_p^*$.*
A function $f : \{0, \ldots, p-1\} \to \mathbb{C}$ will be considered as a function defined on $\mathbb{F}_p$. Show that we can write the Fourier transform of f as follows:

$$\forall b \in \{0, \ldots, p-2\}, \quad \hat{f}(g^{-b}) = f(0) + \sum_{a=0}^{p-2} f(g^a) \omega_p^{-g^{ab}},$$

where $\omega_p = e^{\frac{2\imath\pi}{p}}$. Deduce that we can calculate the Fourier transform of f using a convolution on $\mathbb{Z}/(p-1)\mathbb{Z}$. Specify the vectors involved, and implement this algorithm.

Exercise 54 (Calculations approximated by fractional Fourier transform)**.** *Let a continuous function $f : \mathbb{R} \to \mathbb{R}$ be assumed to be supported in $[-a/2, a/2]$. We have a regular sampling $f[k] = f(x_k)$ with $x_k = f((kN/2)a/N)$ for $k = 0, \ldots, N-1$.*

1. *We want to evaluate the continuous Fourier transform of f around a frequency ζ, more precisely at the points $y_k = \zeta + \frac{2\pi}{a}(kN/2)\gamma$, where γ is the desired precision. If we want to calculate the $\hat{f}(x_k)$ using the FFT algorithm, what is the size of the transform to calculate (we will assume that $\gamma \in \mathbb{Q}$)? Show that we can perform this calculation more economically using the fractional Fourier transform.*

2. *We now want to increase the precision of the calculations. Using the Simpson quadrature method (exercise 51, method (M_2)), explain how to modify the method of the previous question.*

Exercise 55 (Fractional Fourier transform of an image)**.** *Propose a 2D fractional Fourier transform which extends the 1D transform by tensor product (we can use the construction of the 2D Hartley transform, exercise 48). Write the corresponding Matlab function, and test the transform on several images. The figure 5.12 shows the transforms of the indicator function of a disk.*

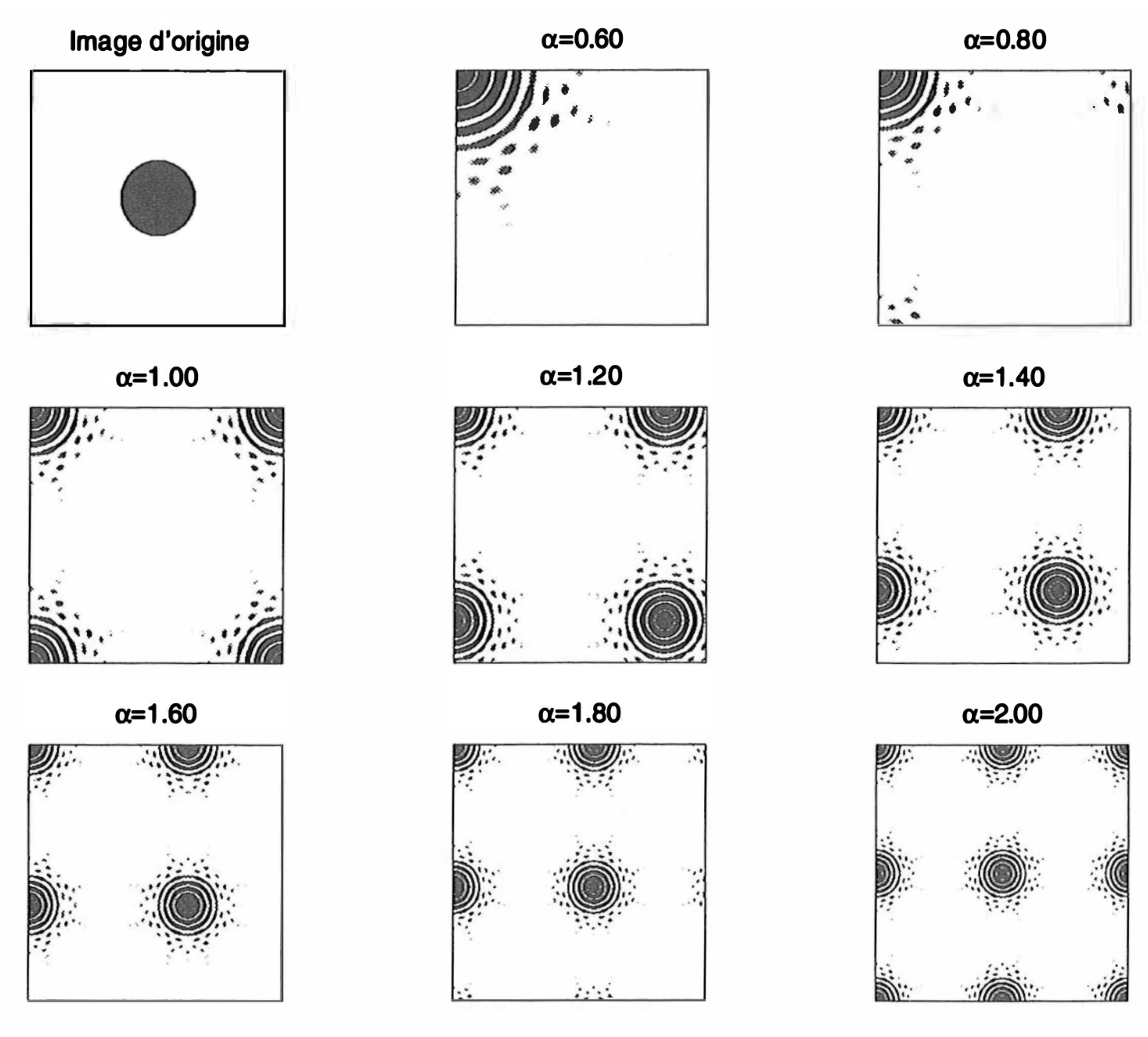

Figure 5.12: Fractional Fourier transforms of a 2D function

Chapter 6

Fourier transform with values in a finite field

We have already encountered finite fields many times, in particular in chapter 2. However, we limited ourselves to exploiting them as *start* domains of the functions we wanted to analyze. However, it is very common to manipulate data with values in a finite set, which we can often provide with a finite field structure. The most striking example is binary data, which can be modeled by functions with values in $\mathbb{F}_2 \simeq \{0, 1\}$. In this chapter, we will present many similar situations, and we will see how Fourier tools naturally extend to such field structures.

6.1 Finite field calculations

Since the beginning of the talk, we have limited ourselves to the study of functions with values in the field $\mathbb{C}$ of the complexes, and we are particularly interested in the morphisms of G (abelian finite group) in the group multiplicative $\mathbb{C}^*$. However, most of the results remain valid if we consider the morphisms of G in any commutative field. In this section, we will take a closer look at the case of finite fields. Not only will we take the results already stated in the previous chapters, but we will particularize them and explain why and how to perform the calculations in a finite field.

It is a question of carrying out our calculations modulo a prime number p. We must be careful not to believe that we are going to restrict ourselves to the only field $\mathbb{F}_p \overset{\text{def.}}{=} \mathbb{Z}/p\mathbb{Z}$. For example, in Paragraph 6.1.4, we are going to place ourselves in a larger field, of the type $\mathbb{F}_{p^r}$, where $r \geq 1$, to define a transform of arbitrary length.

6.1.1 Fourier transform on a finite field

We therefore fix a prime integer p. All our calculations will be carried out modulo p. To construct a valued transform in a finite field, we need a *primitive root* n^{th} of unity. Let's clarify what this means.

Definition 33 (Primitive root)**.** *Let K be a field. An element $\zeta \in K$ is called root of unit if it is a finite order element of K^*, that is, if there is an integer $s > 0$ such that $\zeta^s = 1$.*
An element of order n of K^* is called primitive root n^{th} of the unit, this which means that $\zeta^n = 1$ and that for any s such that $0 < s < n$, we have $\zeta^s \neq 1$.

In a finite field K of cardinal $q = p^r$, any element of K^* is necessarily a root $(q-1)^{\text{th}}$ of the unit. Of course, the existence of a root n^{th}, for any n, is not assured. Indeed, the order of any element of K^* is a divisor of $q-1$. Conversely, if $n|q-1$, i.e. $-1 = nk$, then, if we denote by ω a generator of K^*, $\zeta = \omega^k$ is a primitive n^{th} root.

For simplicity, we'll assume that $q = p$ is a prime number. We also admit that we have ζ, a n^{th} primitive root of the unit over $\mathbb{F}_p$. It is then very simple to define the Fourier transform with values in the field $\mathbb{F}_p$.

Definition 34 (Transform over $\mathbb{F}_p$)**.** *For a vector $f \in (\mathbb{F}_p)^n$, we define the Fourier transform:*

$$\forall j \in \{0, \dots, n-1\}, \quad \mathcal{F}(f)[j] = \hat{f}[j] \overset{\text{def.}}{=} \sum_{k=0}^{n-1} f[k]\zeta^{-kj}. \tag{6.1}$$

The Fourier transform on $\mathbb{F}_p$ has exactly the same properties as the classical Fourier transform; here they are briefly recalled.

Proposition 50 (Properties)**.** *$\mathcal{F}$ is an algebra isomorphism from $((\mathbb{F}_p)^n, *)$ to $((\mathbb{F}_p)^n, \cdot)$, where we denote $*$ the product of circular convolution and $\cdot$ the product component by component. Its inverse is given by*

$$\mathcal{F}^{-1}(f)[n] = n^{-1} \sum_{k=0}^{n-1} f[k]\zeta^{kn}. \tag{6.2}$$

The Fourier transform modulo p has a definite advantage: all the calculations are done with integers (certainly modulo p, but in the end, we always carry out additions and multiplications of integers). There is thus no numerical error likely to taint the results of calculations. On the other hand, in calculations requiring high precision, the use of a conventional FFT can lead to errors. When, for example, we want to calculate the product of two large integers (using the technique presented in Paragraph 4.5.4), it is very important to minimize the numerical errors, since we want, at final, find whole values (we are actually going to round to the nearest whole number). For example, in [5], the author explains that in double precision, past 10 million decimal places, the FFT algorithm, used for calculations of integer products, gave bad results because of rounding errors. This is why algorithms for calculations on finite fields (and more generally on $\mathbb{Z}/m\mathbb{Z}$) are at the heart of computer systems requiring high precision integer calculations.

Beyond all these advantages, we must keep in mind that the result obtained by transforming on a finite field no longer has any meaning "physics". Indeed, the Fourier transform with values in $\mathbb{C}$ represents an approximation of the continuous Fourier transform on $\mathbb{R}$, which is of course far from being the case for the transform on a finite field. We can consider the transformation performed in complexes as a means of passing from a temporal representation of the data to a frequency representation, whereas there is no such obvious interpretation for the transformation carried out on $\mathbb{F}_p$.

6.1.2 A special case

The problem we face in building this transform is finding a primitive n^{th} root of unity. To fully understand the difficulties that we encounter, let's start by showing a case where everything goes well, but which, as we will see, is very restrictive.

We assume that we have chosen $n = p - 1$. We know that the multiplicative group $\mathbb{F}_p^* = \mathbb{F}_p - \{0\}$ is a finite cyclic group. Consequently, it has a generator element: let us denote the ζ. By definition, we therefore have

$$\mathbb{F}_p^* = \{1, \zeta, \zeta^2, \dots, \zeta^{n-1}\} \quad \text{and} \zeta^n = 1.$$

We have therefore exhibited a primitive n^{th} root of unity, but at the cost of a particular choice for the value of n. A naive algorithm to determine a generator of the multiplicative group $(\mathbb{Z}/n\mathbb{Z})^*$ consists in trying all the elements of the group one by one. A more efficient algorithm is presented in the book by Cohen [18].

6.1.3 Cyclotomic bodies

In the previous paragraph, we built a Fourier transform of size n on $\mathbb{F}_p$ in a very specific case, which imposed a relation between the integers n and p. However, one may wish to choose these two parameters independently, for example in the frequent case where one wishes to produce DFTs of very large size, much more than p. The problem is that there is no reason for the field $\mathbb{F}_p \overset{\text{def.}}{=} \mathbb{Z}/p\mathbb{Z}$ to contain primitive n^{th} roots. We are therefore going to have to work in an extension of $\mathbb{F}_p$, that is to say a certain $\mathbb{F}_{p^r}$ for $r \in \mathbb{N}^*$.

To carry out the search for this extension, we need to study the irreducible factors of the polynomial $X^n - 1$, since these will be the minimal polynomials of a possible primitive root. Let us recall without demonstrating what these irreducible factors are on the field $\mathbb{Q}$.

Theorem-Definition 2 (Cyclotomic polynomials)**.** *We denote by Φ_n the $n^{\text{ième}}$ cyclotomic polynomial, defined as follows:*

$$\Phi_n(X) \overset{\text{def.}}{=} \prod_{k\in(\mathbb{Z}/n\mathbb{Z})^*} \left(X - e^{\frac{2\imath\pi k}{n}}\right).$$

Cyclotomic polynomials verify

$$X^n - 1 = \prod_{d|n} \Phi_d(X). \tag{6.3}$$

They have coefficients in $\mathbb{Z}$, and moreover, they are irreducible in $\mathbb{Q}[X]$.

For a proof of this theorem, as well as a very clear construction of finite fields, we can look at the book of Demazure [23].

Example 7. The equation (6.3) provides an algorithm which allows to determine by induction the cyclotomic polynomials, of which here are some examples:

$$\begin{aligned}
\Phi_1(X) &= X - 1,\\
\Phi_2(X) &= X + 1,\\
\Phi_3(X) &= X^2 + X + 1,\\
\Phi_6(X) &= X^2 - X - 1.
\end{aligned}$$

The Maple 1 program calculates a cyclotomic polynomial. For more on this, see the exercise 56.

Program 1 Computing a cyclotomic polynomial

```
with (numtheory):  cyclotomic(105, X);
```

$1 + X - X^8 + X^2 - X^5 - 2\,X^7 + X^{35} - X^{28} + X^{48} + X^{46} - X^{43} - 2\,X^{41} - X^{40} - X^{39} + X^{36} + X^{34}$
$+ X^{33} + X^{31} - X^{26} - X^{24} - X^{22} - X^{20} + X^{17} + X^{16} + X^{15} + X^{14} + X^{12} - X^9 - X^6 + X^{47}$
$- X^{42} + X^{32} + X^{13}$

The polynomials Φ_n being with coefficients in $\mathbb{Z}$, we can therefore reduce them modulo p and regard them as elements of $\mathbb{F}_p[X]$. Although they are irreducible on $\mathbb{Q}$, there is no reason why they are still irreducible on $\mathbb{F}_p$. More precisely, it is the following theorem which will allow us to construct the extension of fields which we need.

Proposition 51 (Cyclotomy on $\mathbb{F}_p$)**.** *We assume* GCD$(n, p) = 1$*. Let r be the order of p in the group $(\mathbb{Z}/n\mathbb{Z})^*$, i.e. the smallest integer t such that $p^t = 1$ modulo n. Then the cyclotomic polynomial $\Phi_n(X)$ decomposes in $\mathbb{F}_p[X]$ into the product of irreducible polynomials of degree r, all different.*

Proof. Since n is prime with p, the polynomial $X^n - 1$ is prime with its derivative nX^{n-1}, so it does not have a multiple root in an extension. With the relation (6.3), we therefore deduce that the polynomial Φ_n has no multiple factor. It is therefore sufficient to show that any irreducible factor of Φ_n is of degree r.
Let P be an irreducible factor of degree s. We denote by $K = F_p[X]/(P)$ the residual field. Its cardinality is $|K| = p^s$. The image ζ of the undetermined X in K is a primitive root of unity, since P is an irreducible factor of $X^n - 1$ and ζ is root of P. Since every element x of K^* satisfies $x^{p^s-1} = 1$, in particular, $\zeta^{p^s-1} = 1$, and the definition of a primitive root n^{th} implies that $n|p^s - 1$. By definition of the order r of p in $(\mathbb{Z}/n\mathbb{Z})^*$, we therefore have $r \leq s$.
Let us show the inverse inequality. Like $n|p^r - 1$, we have $p^r - 1 = \lambda n$, hence $\zeta^{p^r-1} = (\zeta^n)^\lambda = 1$, so $\zeta^{p^r} = \zeta$. Let k denote the subfield of K formed by the roots of the equation $X^{p^r} = X$. It contains ζ which generates K^*, so it is actually equal to K integer. Since $X^{p^r} - X$ has at most p^r distinct roots, we get that $|K| = |k| \leq p^r$, so $s \leq r$ which is the inequality sought. □

Remark 51. In the case where the integers n and p are not prime to each other, n is a multiple of p, that is $n = mp^t$, where this time m is prime with p. We then write

$$X^n - 1 = (X^m)^{p^t} - 1^{p^t} = (X^m - 1)^{p^t}.$$

We can now apply the previous proposition to the polynomial $X^m - 1$ to find a decomposition of $X^n - 1$.

Remark 52. In fact, the proposition 51 is still valid on a field of the type $\mathbb{F}_q$ where $q = p^t$ (without modification of the proof). The polynomial Φ_n decomposes in $\mathbb{F}_q[X]$ into a product of irreducible polynomials of degree r, where r is the order of q in $(\mathbb{Z}/n\mathbb{Z})^*$. We must be careful: looking at the polynomial Φ_n on $\mathbb{F}_p$ amounts to reducing its modulo p coefficients (and not modulo q!). However, since there are more elements in $\mathbb{F}_q$ than in $\mathbb{F}_p$, it is possible that some irreducible factors of Φ_n on $\mathbb{F}_p$ decompose on $\mathbb{F}_q$.

If we denote by P an irreducible factor of Φ_n, the field $K \overset{\text{def.}}{=} \mathbb{F}_p[X]/(P)$ is therefore the extension of $\mathbb{F}_p$ that we are looking for. It is a field of cardinal p^r, and we can see it as a vector space of dimension r on $\mathbb{F}_p$, simply by considering its elements as polynomials of degree at most $r-1$. We generally write $\mathbb{F}_{p^r}$ this field, but it should not be confused with $\mathbb{Z}/p^r\mathbb{Z}$: in fact, in K, we continue to perform the modulo addition p, and the multiplication is more complex since it corresponds to the multiplication of the polynomials modulo P. Finally, let us note that we have thus found a primitive root of unity. It suffices to consider the class of the indeterminate X in K: it is in fact a root that we will denote by ζ of P in K. We can then see the field K as $\mathbb{F}_p[\zeta]$, the field generated by ζ on $\mathbb{F}_p$. We name this field the *cyclotomic field of index* n on $\mathbb{F}_p$.

Example 8. If we consider the polynomial

$$\Phi_{15}(X) = 1 - X + X^3 - X^4 + X^5 - X^7 + X^8 \in \mathbb{Z}[X].$$

Since the order of 2 in $(\mathbb{Z}/15\mathbb{Z})^*$ is 4, Φ_{15} decomposes over $\mathbb{F}_2$ as a product of 2 irreducible degree 4 polynomials:

$$\Phi_{15}(X) = (1 + X + X^4)(1 + X^3 + X^4) \ \in \mathbb{F}_2[X].$$

A decomposition field of Φ_{15} is therefore for example

$$\mathbb{F}_2[X]/(1 + X + X^4) \simeq \mathbb{F}_{2^4} \simeq \mathbb{F}_2[\alpha],$$

where we denote by α a root of $1 + X + X^4$ (i.e. the class of the indeterminate X in the quotient field $\mathbb{F}_2[X]/(1 + X + X^4)$).

In a practical way, to build this field, it is first necessary to calculate the polynomial Φ_n (for example by using by induction the relation (6.3)), then to find a factor irreducible from Φ_n to $\mathbb{F}_p$. To do this, we can use Maple, the whole process being detailed in the first program of paragraph C.1. An algorithm allowing to factorize polynomials over a finite field is the *Berlekamp* algorithm. It is detailed in the book of Demazure [23].

6.1.4 Transform on a cyclotomic field

Thanks to the decomposition of the polynomial Φ_n over the field $\mathbb{F}_p$, we succeeded in constructing an extension of $\mathbb{F}_p$, denoted $K \overset{\text{def.}}{=} \mathbb{F}_{p^r}$, in which there is a root n^{th} primitive of the unit, α. In addition, knowing its minimal polynomial P allows us to calculate with the elements of K (which are vectors of r elements of $\mathbb{F}_p$), since it suffices to use the polynomial multiplication modulo P.

By placing ourselves in the field K, we therefore define the transform of a vector $f \in K^n$ by the equation (6.1). This time, $\mathcal{F}(f)$ is also a vector of K^n. In practice, we often use this transform for data which are elements of $(\mathbb{F}_p)^n$, but of course we must keep in mind that the result has no reason to stay in $\mathbb{F}_p$. We find the same phenomenon as for the computation of the (classical) DFT of a real vector, for which we are obliged to place ourselves in the $\mathbb{C}$ extension of the field of real numbers. Recall that if we denote the elements of $\mathbb{F}_{p^r}$ as vectors (rather than polynomials), an element of the base field $\mathbb{F}_p$ is distinguished by the fact that only the first component of the vector is not zero.

This construction of a fast transform algorithm for an arbitrary length N will be used in Section 6.3. It will in fact be a question of decoding certain corrective codes using the Fourier transform. We will see that the use of a super-field (which may seem a little artificial for the moment) then becomes very natural to find the set of codes verifying certain properties (we call them *cyclic codes*).

6.1.5 Effective calculations

We will be able to use the Fourier transform with values in a finite field to do convolution calculations. As all the calculations are carried out modulo p, we will have to choose an integer p large enough to be able to retrieve results of calculations in $\mathbb{Z}$, and not modulo p.

The best example of the use of this Fourier transform is the computation of products of large integers, written in base b. The paragraph 4.5.4, explains how to calculate the product of two integers using an acyclic convolution. We can easily determine a bound on the value of the inputs of a linear convolution of two integers represented in base b:

$$|f \star g[k]| = \left| \sum_{l=0}^{n-1} f[l]g[kl] \right| \leq \sum_{l=0}^{n-1} |f[l]||g[kl]| \leq n(b-1)^2.$$

In order to be able to calculate exactly the product of two integers, $|f \star g[k]| < p$. We will therefore be interested in choosing $p > n(b-1)^2$.

To finish, it is necessary to say a few words about the practical implementation of the algorithm for calculating the transform. Most often, we will consider that n is of the form 2^k, and we will use the dichotomous FFT algorithm presented in Section 3.2. The details of the algorithm remain unchanged. Among other things, if ζ is the root of order 2^k sought, then ζ^2 will be the root of order 2^{k-1}, etc. Of course, it is necessary to replace the operations in the field of the complexes by operations in the finite field $K = \mathbb{F}_{p^r}$. We must therefore choose a way to represent the elements of the field K, and know how to multiply them. If we have looked for the primitive root ζ as indicated in Paragraph 6.1.3, we have a very natural way of doing it. Indeed, we have at the same time the minimal polynomial P of ζ, of degree r, and we can therefore represent the elements of the field as polynomials of degree at most $r-1$ (c'that is, by vectors of size r in the computer memory). The addition is then carried out component by component, while the multiplication is carried out as a multiplication of polynomials modulo P. This multiplication is calculated in the classic way, then by taking the remainder of the Euclidean division by P.

The program Maple of the C.1 section, performs step by step the different steps leading to the construction of a transform on a finite field. It starts with the factorization of Φ_n on $\mathbb{F}_p$, and contains a recursive FFT algorithm. It makes full use of the Maple facilities, which eliminates the need to worry about operations in the field $\mathbb{F}_{p^r}$. Finally, it will be noted that it precedes the program on *corrective codes BCH* (paragraph C.4), the latter using the procedures for calculating transforms on a finite field.

6.2 Calculations on a ring

It is natural, after having presented the Fourier transform with values in a finite field, to try to extend this notion to any commutative ring. This generalization is not at all free, since it will make it possible to calculate in any ring $\mathbb{Z}/m\mathbb{Z}$ at a lower cost (the calculation of sums and products is done quite simply modulo m). We will thus obtain an FFT algorithm much simpler than the one built in the previous paragraph, which required the passage in an over-field and the computation of Euclidean divisions to perform the products in this field. Of course, just like in the simplistic example of Paragraph 6.1.2, where we had $n = p - 1$, we will once again have limitations on the choice of n or p. However, we will see that by considering rings of the type $\mathbb{Z}/2^r\mathbb{Z}$, we can build a very simple FFT algorithm, where the primitive root search becomes trivial.

6.2.1 Main roots of the unit

In this section, we consider more general structures, namely arbitrary commutative rings. In practice, we mainly use the ring $\mathbb{Z}/m\mathbb{Z}$, where m is a positive integer, which allows calculations to be done in a simple way using a computer (addition and multiplication modulo m). A problem arises when one wishes to define a Fourier transform using the equation (6.2) on any ring A: the transformation thus defined does not have no reason to be bijective. This is due to the presence of divisors of zero in the ring A. Recall that a divisor of zero $x \neq 0$, is an element of A such that there exists a $y \neq 0$ verifying $xy = 0$. For example, in the ring $\mathbb{Z}/6\mathbb{Z}$, we have the equality $2 \cdot 3 = 0$, so the elements 2 and 3 are divisors of zero. To overcome this problem, we will have to impose additional restrictions on the choice of the root $n^{\text{ième}}$ ζ. By following the notations of Demazure [23], we will introduce the concept of *main root of the unit.*

Definition 35 (Main root of unit)**.** *Let A be any commutative ring. An element $\zeta \in A$ is called the main n^{th} root of the unit if:*

(i) *we have $\zeta^n = 1$ (in particular, ζ is invertible).*

(ii) *for any $i \in \{1, \ldots, n-1\}$, the element $1 - \zeta^i$ is not a divisor of zero in A. This means that if $a\zeta^i = a$, then $a = 0$.*

Remark 53. The fact that $\zeta^i - 1$ is not a divisor of zero for $i = 1, \ldots, n-1$ implies in particular that $\zeta^i \neq 1$, so a principal root is a primitive root. We see that if the ring is integral (and a fortiori if it is a field), the notions of primitive $n^{\text{ième}}$ root and of main $n^{\text{ième}}$ root coincide. On the other hand, if we consider for example the ring $\mathbb{Z}/15\mathbb{Z}$, we see that the element 2 is a primitive root $4^{\text{ième}}$ of the unit, but it is not not principal, since $2^2 - 1 = 3$ is a divisor of zero.

We are therefore going to find a main root of the unit, for example by carrying out an exhaustive search among the elements of A^*. As this calculation is done once and for all, it is not important to have a fast algorithm. We can then define the discrete Fourier transform with values in A in the usual way.

Definition 36 (Fourier transform in a ring)**.** *Let A be a commutative ring, and $\zeta \in A^*$ a main n^{th} root. For a vector $f \in A^n$, we define the Fourier transform $\mathcal{F}$ as well as another map denoted $\tilde{\mathcal{F}}$ by*

$$\forall j \in \{0, \ldots, n-1\}, \quad \mathcal{F}(f)[j] \overset{\text{def.}}{=} \sum_{k=0}^{n-1} f[k]\zeta^{-kj}, \tag{6.4}$$

$$\forall j \in \{0, \ldots, n-1\}, \quad \tilde{\mathcal{F}}(f)[j] \overset{\text{def.}}{=} \sum_{k=0}^{n-1} f[k]\zeta^{kj}. \tag{6.5}$$

In order to study the relations between these two transforms, we need the following lemma.

Lemma 10. *Let A be a commutative ring and ζ a main n^{th} root of unity. We then have*

$$\sum_{i=0}^{n-1} \zeta^{ki} = \begin{cases} n & \text{sik} = 0 \mod n \\ 0 & \text{otherwise} \end{cases}.$$

Also, n is not a divisor of zero in the ring.

Proof. We will prove the following polynomial equality:

$$X^n - 1 = \prod_{i=0}^{n-1} (X - \zeta^i).$$

Denote by $P(X) = X^n - 1$. We have $P(1) = 0$, so P is written $(X-1)Q(X)$, where Q is a unit polynomial (indeed, like the polynomial $X - 1$ is unitary, we can realize the Euclidean division of P by $X - 1$ and see that the remainder is zero). As ζ is also the root of P, we see that $(\zeta - 1)Q(\zeta) = 0$, and the fact that $\zeta - 1$

is not a divisor of zero allows us to conclude that $Q(\zeta) = 0$. We start again with Q, which we write in the form $(X - \zeta)R(X)$. We then have $P(\zeta^2) = (\zeta^2 - 1)\zeta(\zeta - 1)R(\zeta^2)$. The fact that $\zeta^2 - 1$ is not a divisor of zero allows us to state that $R(\zeta^2) = 0$. We continue in this way until finding the announced factorization. By removing the factor $X-1$, we find

$$X^{n-1} + \cdots + X + 1 = \prod_{i=1}^{n-1} \left(X - \zeta^i\right).$$

Hence, by evaluating the previous equality in $X = 1$,

$$n1_A = \prod_{i=1}^{n-1} \left(1 - \zeta^i\right).$$

This shows that n is not a divisor of zero in A. Evaluating the same equality again, but for $X = \zeta^k$, we obtain the announced equality. □

We can then state the main result.

Théorem 8 (Properties of the transform on a ring)**.** *We denote by $*$ the product of circular convolution and $\cdot$ the product component by component. $\mathcal{F}$ is an algebra morphism from $(A^n, *)$ into $(A^n, \cdot)$. We have the connections*

$$\forall f \in A^n, \quad \mathcal{F}(\tilde{\mathcal{F}}(f)) = nf \qquad \text{and} \qquad \tilde{\mathcal{F}}(\mathcal{F}(f)) = nf.$$

The morphisms $\mathcal{F}$ and $\tilde{\mathcal{F}}$ are injective. Moreover, if $n1_A$ is invertible, then the map $\mathcal{F}$ is an inverse isomorphism $n^{-1}\tilde{\mathcal{F}}$.

Proof. The property of morphism does not present any difficulty, the proof is identical to that carried out for the value transform in $\mathbb{C}$, theorem 2.
We will calculate $\mathcal{F}(\tilde{\mathcal{F}}(f))$:

$$\mathcal{F}(\tilde{\mathcal{F}}(f))[n] = \sum_i \zeta^{-in} \sum_j f[j]\zeta^{ij} = \sum_j f[j] \sum_i \zeta^{i(jn)}.$$

Then it suffices to use the previous lemma 10 with $k = jn$ to conclude.
The injectivity of $\mathcal{F}$ results simply from the fact that n is not no divisor of zero in A, because if $\mathcal{F}(f) = 0$, then $\tilde{\mathcal{F}}(\mathcal{F}(f)) = nf = 0$, so $f = 0$. □

In the framework of the ring $A \overset{\text{def.}}{=} \mathbb{Z}/m\mathbb{Z}$, the conditions under which we can construct a Fourier transform become simpler. The exercise 57 details the steps which make it possible to demonstrate the following proposition:

Proposition 52. *Let $m = p_1^{k_1} \times \cdots \times p_r^{k_r}$ where the p_i are distinct prime numbers. We can construct an invertible Fourier transform of size n on the ring $A \overset{\text{def.}}{=} \mathbb{Z}/m\mathbb{Z}$ if the following conditions are satisfied.*

(i) $\mathrm{GCD}(n, m) = 1$.

(ii) *If m is written as, then n divides* $\mathrm{GCD}(p_1 - 1, \ldots, p_r - 1)$.

Condition (i) makes it possible to invert n in A, and condition (ii) makes it possible to construct a main n^{th} root. In the following paragraph, we will particularize this study to certain classes of integers m, in order to build a fast computation algorithm of the FFT type.

6.2.2 Implementation of an FFT algorithm

We want to implement a dichotomous FFT algorithm on any ring A. We therefore suppose that $n = 2^s$, and the problem, in order to be able to implement the algorithm, is first of all to find a $(2^s)^{\text{th}}$ principal root of the unit. But when the second recursive call is launched, it will be necessary to find a root $(2^{s-1})^{\text{ième}}$ principal, then a root $(2^{s-2})^{\text{ième}}$, etc. If we want the FFT algorithm to be really useful, this search for a main root must take as little time as possible.

The following proposition, whose proof is the object of the exercise 58, will allow us to construct this main root ζ.

Proposition 53. *Let A be a commutative ring. For ζ to be a principal root $(2^k)^{ième}$, it is necessary and sufficient that 2 is not a divisor of zero in the ring, and more than $\zeta^{2^{k-1}} = -1$.*

We will choose an odd m integer so that 2 is invertible in $A \overset{\text{def.}}{=} \mathbb{Z}/m\mathbb{Z}$ (in particular, 2 will not be a divisor of zero). Moreover, by simply taking $\zeta = 2$, we see that $\zeta^{2^{k-1}} = -1$ in A by assigning m the value $2^{2^{k-1}} + 1$. We have indeed found a root $(2^k)^{\text{ième}}$ of the unit, in this case $\zeta = 2$, moreover, as we see by repeating the exercise 58, ζ^2 will then be an $(2^{k-1})^{\text{ith}}$ root of the main unit, then we will use ζ^4, etc.

Remark 54. Numbers of the form 2^{2^n} are of great importance, and are called *Fermat numbers*. We denote by $\text{Fer}_n \overset{\text{def.}}{=} 2^{2^n}$ the n^{th} Fermat number. We can easily see that any prime number of the form 2^k is in fact a Fermat number. We can see that $\text{Iron}_0 = 3$, $\text{Iron}_1 = 5$, $\text{Iron}_2 = 17$, $\text{Iron}_3 = 257$ and $\text{Fer}_4 = 65537$ are all prime, unfortunately Fer_5 is not ...

In conclusion, this method, although less flexible in terms of the choice of n, is much easier to implementœ than the Fourier transform in a cyclotomic field . In addition, it requires significantly fewer calculations. As a complete program is better than long speeches, we can refer to the paragraph C.2, where the FFT algorithm of length n on the ring $\mathbb{Z}/m\mathbb{Z}$ with $m = 2^{2^{s-1}}$ is implemented in Maple.

6.3 Application to correction codes

This paragraph is a modest introduction to the theory of *corrective codes*; it is above all a question of making the reader want to seek additional information in the proposed bibliography. The goal is to apply the tools introduced from the beginning of this chapter, on the one hand to better understand the conditions involved in the construction of codes, and on the other hand to obtain fast and efficient decoding algorithms. First, the definitions of the main concepts are given, highlighting the theory of cyclic codes. Then, it is a question of reinvesting as well as possible the knowledge which one possesses on the cyclotomic bodies and on the discrete Fourier transform, to arrive, at the end of the presentation, to build codes and algorithms.

The classical theory of error correcting codes covers a wide variety of subjects, which makes it particularly attractive (for example to illustrate an aggregation lesson). First of all, it deals with finite geometry (since it is basically a matter of manipulating balls in a finite space). Then, the combinatorial aspect of the codes presents a good number of remarkable properties, mainly around the relations between them the various parameters of the codes. Finally, there will be a lot of discussion of the theory of bodies, which constitutes the heart of this talk.

A very good reference on corrective code theory is the book by Papini and Wolfman [50]. That of Demazure [23] constitutes a very nice presentation, with among other things a very efficient *BCH* code decoding algorithm.

6.3.1 Notion of corrective code

The notion of *coding* of information is not new: it is even intimately linked to human activity, to the natural need to communicate by various but not always very reliable means. The best example is human language, which is very complex, and which perfectly meets the need to correct errors. After all, why use such complicated natural languages, when one could use words certainly much shorter, with a much simpler syntax? One of the explanations consists in saying that these natural languages make it possible to understand each other better (provided you master the basics). Indeed, the diversity of the words used reduces the risk of error during a conversation, and the rigidity of the grammar rules makes this communication less sensitive to hazards (ambient noise, bad pronunciation, etc.). In a way, all this helps to make the words that make up the language very different from each other, so that they can be easily distinguished. If so, if a communication error occurs, it will be relatively easy for the speaker to find the meaning of the original message.

This first example gives all the key points of a theory of error correcting codes. It is about finding a way to encode certain information so as to make it less sensitive to errors during communication. We can split the coding process as follows:

– the transformation of information (which can be a thought, a sound, a DNA sequence, etc.) into a series of symbols. In the theory which will follow, we will not be interested in the meaning of this series of symbols. It is in a way a layer of abstraction that will allow us to set a way of representing the information that we want to process.

– the actual mathematical encoding. It is this part of the coding process that will interest us. This involves modifying the series of symbols in an appropriate manner so as to make it as less sensitive as possible to transmission errors. This transformation will require the addition of redundant information.

The aim of corrective code theory is therefore to use algebraic structures and algorithms so that:

– the redundant information added to the message during encoding is as low as possible, for a fixed number of corrected errors.

– coding and especially decoding algorithms are fast and efficient.

To achieve this, we are going to impose more or less rigid (algebraic) structures on the set of manipulated words (which we call the code).

The first choice to make is that of the alphabet that we will use to write the words of the messages. The original information (for example a message that is to be transmitted over a network) will thus be transformed, during the first coding step, into a series of symbols taken from this alphabet. A major example of an alphabet is that used to encode the genetic information of the human code. In this case, the information to be encoded is DNA (**A** cide **D** ésoxyribo **n** nucleic), and the symbols are 4 “nitrogenous bases”, which we symbolically notes A, T, G and C. This first coding example is symptomatic of the mathematical construction that we are going to carry out. It is a question of replacing information (which has a certain meaning for the one who transmits it) into a series of fixed symbols. Recent studies moreover discuss the existence of corrective codes in the sequence of nitrogenous sequences of DNA, see for example [45].

In order to make the following mathematical analysis pleasant, we are going to take as symbols elements of certain algebraic sets. The choice we are going to make, and which may appear arbitrary, is that of a finite field. In practice, this choice is not that restrictive. For example, the binary encoding of elements in the field $\mathbb{F}_2 = \mathbb{Z}/2\mathbb{Z}$ is ideal for the way information is stored in computer memory. In the same vein, the elements of a field of the type $\mathbb{F}_{2^r}$ can be stored in the form of vectors of r bits. The second choice to be made is which of the words we are actually going to send over a network. They will all have the same length, which we will denote by n. So we are going to group the elements of the alphabet sent in packets of n to obtain n -uples $(x_0, \dots, x_{n-1})$, with $x_k \in \mathbb{F}_q$. In the following, we will therefore assume that these words will be elements of the vector space $(\mathbb{F}_q)^n$, where $q \overset{\text{def.}}{=} p^r$ with p a prime number.

We want to use certain algebraic structures to give rise to more and more efficient codes. However, there are many codes which are used very frequently and which do not require complex algebraic constructions. Here are two examples.

Example 9 (Parity bit). This is arguably the simplest and most widely used code. The information we want to encode consists of a sequence of n bits, which we will therefore denote in the form $a = a_0 a_1 \dots a_{n-1}$, where $a_k \in \{0.1\}$. The encoding operation consists of adding a $(n+1)^{\text{ith}}$ bit a_n defined as follows:

$$a_n \overset{\text{def.}}{=} a_0 + \cdots + a_{n-1} \mod 2.$$

In fact, we arrange so that the sum of all the bits of the transmitted word is always an even number. It is then noted that this code makes it possible, on reception, to detect an error due to the modification of only one bit, since then the sum of the bits will be odd. However, if two simultaneous errors occur, the code will be unable to notice them. Besides, it is pointless to want to correct an error with such a simple code. We will therefore have to build more complex codes (with more redundant information added), to obtain performance that meets our requirements. However, this first example is fundamental, on the one hand because in many cases it is sufficient (if the communication is very reliable, as is the case for the storage in the main memory of a computer), and'on the other hand because it can be superimposed on an already existing code to make it more efficient.

Example 10 (ISBN code). The code *ISBN* (for **I** nternational **S** tandard **B** ook **N** umber) is a unique identifier assigned to each book. It is a series of 9 decimal digits followed by another number between 0 and 10 (we then note 10 in the form of the Roman numeral X). If we write down the ISBN of a book $d_0 \dots d_9$, the tenth digit is calculated so as to satisfy the equality

$$10d_0 + 9d_1 + \cdots + 2d_8 + d_9 = 0 \mod 11.$$

In general, we group the d_i into 4 packages separated by hyphens, as for example in ISBN 0-201-89683-4, whose meanings are as follows.

– The first group is made up of a single digit, which characterizes the publisher's country (0 for England, 2 for France, 3 for Germany, etc.).
– The second group characterizes the editor, and its length can vary from 2 to 7 digits (the fewer digits, the larger the editor).
– The third group characterizes the book, and its length can vary from 1 to 6 digits.
– The fourth and last group corresponds to the number d_9 supposed to detect possible errors.

We can easily see that this code makes it possible to detect an error, as well as an exchange between two consecutive digits. However, it cannot correct errors or detect certain digit inversions. This coding is very clever, as most of the time ISBN codes are entered by humans, who often tend to reverse two consecutive digits.

There are many other examples of corrective codes in everyday life. We could cite the *Traveler's Checks American Express* numbers, the *bar codes*, and of course the *compact discs*, which use much more complex codes (to counter any scratches on their area).

Our words are therefore vectors of the space $(\mathbb{F}_q)^n$. To study the properties of our code $\mathcal{C} \subset (\mathbb{F}_q)^n$, it is interesting to introduce a notion of proximity between the words which compose it, to be able to consider this code from a geometric angle. We are led to consider a distance over the set of words. Here is the one that is most commonly used.

Definition 37 (Hamming distance). *We define the weight of a word $x \in (\mathbb{F}_q)^n$, denoted by $w(x)$, as the number of non-zero entries in x. The Hamming distance between two words x and y of $(\mathbb{F}_q)^n$ is defined by $d(x, y) = w(xy)$.*

We can easily see that d is indeed a distance, that is, for elements x, y, and z of $(\mathbb{F}_p)^n$:

(i) $x \neq y \Longrightarrow d(x, y) > 0$;

(ii) $d(x, y) = d(y, x)$;

(iii) $d(x, y) \leq d(x, z) + d(z, y)$.

This distance therefore corresponds to the number of entries which differ between two words. For example, the illustration 6.1 shows the Hamming distance on $(\mathbb{F}_2)^3$, each edge of the cube connecting two points at distance 1.

We could have chosen another distance, but it turns out that the Hamming distance is both simple enough to allow efficient calculations, and precise enough to fully account for the efficiency of the codes.

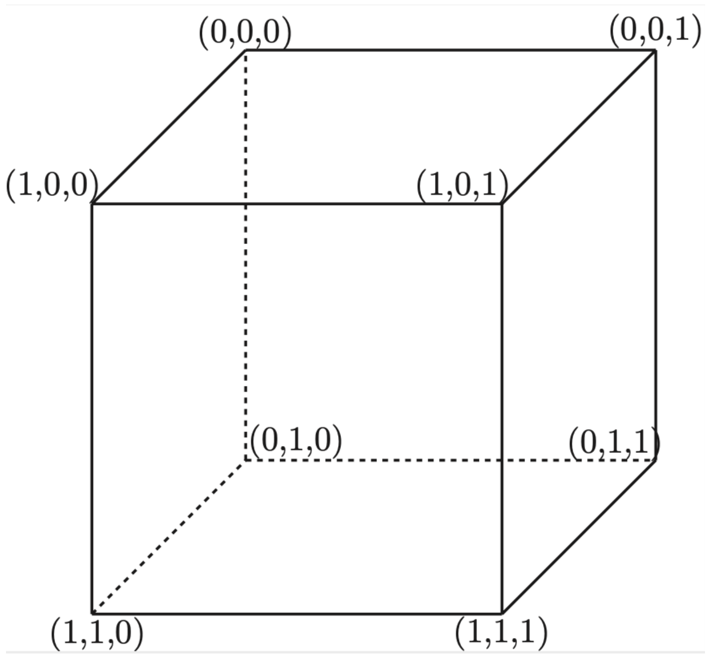

Figure 6.1: Hamming distance over $(\mathbb{F}_2)^3$

6.3.2 Presentation of linear codes

After the choice of the alphabet (which is therefore a finite field $\mathbb{F}_q$), the second hypothesis that we will do concerns the set of words that can be encoded. Indeed, it is obvious that the encoding operation will add information to the original message, therefore the set of "valid" words (i.e. the words that we will be able to produce by encoding) will not occupy the entire space $(\mathbb{F}_q)^n$. Intuitively, we can see that the more space there is between valid words (i.e. code words), the more we will be able to distinguish them from each other, and therefore the more will be easy to spot any errors.

The search for a set $\mathcal{C} \subset (\mathbb{F}_q)^n$ presenting, for a given size $|\mathcal{C}|$, the best weight distribution (in a sense to be specified, but intuitively, such that the words are as far apart as possible) is an extremely difficult problem. Thus, this research is closely linked to that of the most compact possible stacking of spheres, and to the famous *Kissing Number* problem. For more details, refer to the article by Elkies [29]. It is possible to define a set of constants characterizing with more or less finesse the distribution of the words of a code.

Definition 38 (Distribution functions)**.** *We denote by $\{A_i\}_{i=0}^n$ the distribution of weights of a code $\mathcal{C} \subset (\mathbb{F}_q)^n$:*

$$\forall i = 0, \dots, n, \quad A_i \overset{\text{def.}}{=} \#\{x \in \mathcal{C} \ : \ w(x) = i\}. \tag{6.6}$$

We denote by $\{B_i\}_{i=0}^n$ the distance distribution of $\mathcal{C}$:

$$\forall i = 0, \dots, n, \quad B_i \overset{\text{def.}}{=} \frac{1}{|\mathcal{C}|}\#\{(x, y) \in \mathcal{C} \ : \ d(x, y) = i\}. \tag{6.7}$$

It should be noted that the couples (x, y) are considered ordered, that is to say $(x, y) \neq (y, x)$.

Remark 55. We see that we have $B_0 = 1$, and that

$$A_0 + \cdots + A_n = B_0 + \cdots + B_n = |\mathcal{C}|.$$

Moreover, if u is any vector of $(\mathbb{F}_q)^n$, the codes $\mathcal{C}$ and $\mathcal{C} + u$ have the same weight distribution. This is why in practice we assume that $0 \in \mathcal{C}$, even if $\mathcal{C}$ is not linear.

We will come back to the calculation and the study of these distributions in the Section 6.4. To quantify this distribution in a simpler way, we introduce a very useful notion, that of *minimal distance.*

Definition 39 (Minimum distance)**.** *We denote by d the minimum distance of the code $\mathcal{C}$ considered, which is defined as follows:*

$$d \overset{\text{def.}}{=} \min \{d(x,\, y) \ : \ x \neq y \in \mathcal{C}\}\,.$$

Intuitively, we realize that the greater this minimum distance, the more efficient the code will be, since the words will be more distant from each other. This choice is very partial, and in a way very pessimistic, since it only takes into account the smallest distance. However, it gives us conditions under which we are sure to correct an error, which is specified by the following definition.

Definition 40 (Correction capability)**.** *The correction capacity t of a code $\mathcal{C}$ is the maximum number of errors that it can correct. More precisely, if the sender sends a code word $x \in \mathcal{C}$ through a network, and the receiver receives a word x' (which may be different from the word sent), then it must be able to find the original word if $d(x,\, x') \leq t$.*

This means that the balls of radius t (for the Hamming distance) whose center is a code word must be disjoint from each other, or, in other words, that the words must be at a distance d'at least $2t+\, 1$ from each other. We therefore obtain the following result.

Proposition 54. *A $\mathcal{C}$ code is t -corrector (i.e. it has a correction capacity of at least t) if the minimum distance between two distinct words is greater than or equal to $2t+$ 1. The correction capacity of the code is $t = \lfloor (d-1)/2 \rfloor$, where we have noted $\lfloor x \rfloor$ the integer part of a real x.*

To greatly simplify the search for efficient codes (for example with a large minimum distance d), we are going to impose a very restrictive structure on the words of the code.

Definition 41 (Linear code)**.** *A linear code $\mathcal{C}$ of size n and dimension m on $\mathbb{F}_q$ is a vector subspace of dimension m of $(\mathbb{F}_q)^n$. If we consider a matrix G (called generator matrix) whose columns form a basis of $\mathcal{C}$, we have*

$$\mathcal{C} = \{Gx \ : \ x \in (\mathbb{F}_q)^m\}\,.$$

Note that there is no uniqueness in the choice of G. Certainly, even an optimal linear code will at best be as good as the best nonlinear code (i.e. any code), and in practice it will be much worse. However, restricting ourselves to vector subspaces will make our research much more fruitful, and will also bring its share of efficient decoding algorithms. For example, the linearity property of $\mathcal{C}$ allows us to calculate the minimum distance d much more simply:

$$d \overset{\text{def.}}{=} \min \{d(x,\, y) \ : \ x \neq y \in \mathcal{C}\} = \min \{w(x) \ : \ x \neq 0 \in \mathcal{C}\}\,.$$

In the same way, in the linear case, one notes that the distributions of weight and distance coincide. Unless explicitly stated otherwise, it is now assumed that the code $\mathcal{C}$ is a linear code, of size n and dimension m.

The first phase of the encoding operation consists in transforming the original message containing certain information into a word of the code $\mathcal{C}$. This operation must be bijective. In the case of a linear code, it is very easy to achieve this. The simplest way to operate is to simply consider that our messages are "small" vectors of $(\mathbb{F}_q)^m$, and that we send them on "large" vectors of $(\mathbb{F}_q)^n$, simply by multiplying them on the left by the matrix G. The matrix G is not chosen in a canonical way, but the choice of another base leads to a code having approximately the same properties (one speaks about isomorphic code).

Remark 56. (**Control matrix**). A code of size n and dimension m on $\mathbb{F}_q$ can be seen as the kernel of a matrix H of size $(nm) \times n$. We call this matrix a *control matrix* of the code $\mathcal{C}$; it allows to check if a vector $x \in (\mathbb{F}_q)^n$ belongs to the code since $x \in \mathcal{C} \Leftrightarrow Hx = 0$. There is no uniqueness in the choice of G.

Example 11 (Repetition code). Take the simple example of the repeat code. It consists of repeating, for example, 4 times a symbol $x \in \mathbb{F}_2$. The only two words in the code are (0000) and (1111). Generator G and control H matrices are for example

$$G = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \qquad H = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}.$$

The dual code (concept specified a little later, definition 45) is made up of the vectors $x \in (\mathbb{F}_2)^4$ such as $\langle x, [1111] \rangle = 0$. So it's simply the code of adding a parity bit to $x \in \mathbb{F}_2^3$. It is noted that it suffices, except for a transposition, to exchange the generator and control matrices. On this subject, we can see the exercise 61.

Example 12 (Hamming code of length 7). We consider the code of size 7 and dimension 4 on $\mathbb{F}_2$ whose generator matrix is

$$G \overset{\text{def.}}{=} \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

We can explain the 16 elements that make up the code:

$x^\top$	$(Gx)^\top$	$w(Gx)$	$x^\top$	$(Gx)^\top$	$w(Gx)$
(0000)	(0000000)	0	(1000)	(1101000)	3
(0100)	(0110100)	3	(0010)	(0011010)	3
(0001)	(0001101)	3	(1100)	(1011100)	4
(1010)	(1110010)	4	(1001)	(1100101)	4
(0110)	(0101110)	4	(0101)	(0111001)	4
(0011)	(0010111)	4	(1110)	(1000110)	3
(1101)	(1010001)	3	(1011)	(1111111)	7
(0111)	(0100011)	3	(1111)	(1001011)	4

As we can see, each non-zero word of the code has a weight greater than 3, so the correction capacity of this code is 1. In addition, it has an interesting property: the 4 row vectors of the matrix G are deduced from each other by circular permutation. Consequently, the whole code is invariant by circular permutation. In the following, we will be interested in the algebraic properties of such codes, and we will see that we have very practical tools to build them, and, in certain cases, to decode them. The exercise 62 proposes to generalize the construction which has just been done, to give birth to a widely used family of codes, the *Hamming codes*.

To end this paragraph, here is an important relationship between the various parameters of a code, which clearly shows the choice to be made between correction capacity and redundancy of the information transmitted.

Proposition 55 (Singleton bound)**.** *Let $\mathcal{C}$ be a linear correction code of length n, of dimension m, and of minimum distance d. We then have*

$$d \leq n + 1 - m. \tag{6.8}$$

Proof. Let E be the subspace of $(\mathbb{F}_q)^n$ formed by vectors whose last $m-1$ components are zero. It is a space of dimension $n - m + 1$.

We have $\dim(\mathcal{C}) + \dim(E) = n + 1$ which implies that $\mathcal{C} \cap E \neq \{0\}$. So there exists a non-zero x element in $\mathcal{C}$ whose last $m-1$ components are zero, therefore which satisfies $w(x) \leq n + 1 - m$. We therefore have the desired relationship from the definition of the minimum distance. □

Remark 57. Codes which verify $d = n + 1 - m$ are called *MDS codes* (for **M** aximum **D** istance **S** eparable, in English in the text), they are therefore optimal for the Singleton bound. The exercise 67 studies these codes in detail.

6.3.3 Cyclic codes

The problem with linear codes is that, in the general case, rapid decoding algorithms are not available, or else the latter require the storage of decoding tables which quickly become enormous for codes of respectable size. To remedy this problem, we will impose an additional structure on the linear codes.

Definition 42 (Cyclic code)**.** *A code $\mathcal{C}$ of size n on $\mathbb{F}_p$ is said to be cyclic if it is stable by circular shift, that is to say*

$$\forall a = (a_0, \dots, a_{n-1})^\top \in \mathcal{C}, \quad \tilde{a} \overset{\text{def.}}{=} (a_{n-1}, a_0, \dots, a_{n-2})^\top \in \mathcal{C}.$$

A very convenient way to represent the words of a cyclic code is to consider them as elements of the $\mathbb{F}_p$ -algebra $\mathcal{A}$ of dimension n that is $\mathbb{F}_p[X]/(X^n - 1)$. We therefore consider that a word a is in fact a polynomial of degree at most $n-1$ (we choose a modulo representative $X^n - 1$), denoted $a_0 + a_1X + \cdots + a_{n-1}X^{n-1}$. We then notice that

$$\tilde{a} = a_{n-1} + a_0X + \cdots + a_{n-2}X^{n-1} = Xa + a_{n-1}(X^n - 1).$$

So modulo $X^n - 1$ (that is, in $\mathcal{A}$), we have $\tilde{a} = Xa$. The code $\mathcal{C}$ is stable by multiplying by X. As it is also a vector space, by linearity, we deduce that it is in fact stable by multiplication by any polynomial $P \in \mathcal{A}$. This means that it is an ideal of the ring $\mathcal{A}$. We know that the ideals of $\mathcal{A}$ are in bijection with the ideals of $\mathbb{F}_p[X]$ which contain the ideal generated by $X^n - 1$. Since the ring of polynomials $\mathbb{F}_p[X]$ is principal, an ideal of $\mathbb{F}_p[X]/(X^n - 1)$ is generated by a unique unit polynomial which must additionally be a divisor of $X^n - 1$.

If we denote by P the generator polynomial of the code $\mathcal{C}$, we have, denoting $s \overset{\text{def.}}{=} \deg(P)$,

$$\mathcal{C} = \{PQ \mod X^n - 1 \ : \ Q \in \mathbb{F}_p[X]\} = \{PQ \ : \ \deg(Q) \leq ns - 1\}.$$

The code $\mathcal{C}$ is therefore of length n and of dimension ns. The encoding operation is even simpler than for a linear code. The information that we want to transmit, instead of being contained in a vector of size ns, is this time represented by a polynomial of degree at most $ns - 1$ (but it's the same thing) . To obtain a word of the code that we are going to send over a network, it suffices to multiply this polynomial by the generator polynomial P.

We can now make a connection with the ideas that we introduced during the study of the Fourier transform on a cyclic group, and more particularly during the research rapid polynomial multiplication techniques (Section 4.5). We have already noticed that the multiplication by a polynomial modulo $X^n - 1$ corresponds to the computation of a cyclic convolution of vectors of size n (the equation (4.12), shows very clearly the reconciliation). We can write the encoding operation of a vector $x \in (\mathbb{F}_p)^{ns}$ as the circular convolution $x * y$ where we denote y the vector of the coefficients of the polynomial P . You should add zeros at the end of each of the vectors so that they reach the size n. The generator matrix G of the code therefore corresponds to a circulating matrix as we explain in the exercise 26.

In Section 4.5, we were only interested in calculations for polynomials of $\mathbb{C}[X]$. But this limitation was only justified because we did not have a Fourier transform on a field other than $\mathbb{C}$. Thanks to the construction of the 6.1.4 paragraph, this limitation is lifted, and we are able to perform a Fourier transform with values in the field $\mathbb{F}_p$ (even if the latter, let us recall- le, requires to pass in a larger field, which one noted $\mathbb{F}_{p^r}$). So by taking the formula for calculating the cyclic convolution product (4.12), we see that we can quickly perform the coding operation, of course using an FFT algorithm to perform the Fourier transform.

6.3.4 Construction of BCH codes

In this paragraph, we will present a class of cyclic codes named *BCH codes*, from the name of their inventors, Bose, Chaudhuri and Hocquenghem. The construction makes full use of the decomposition of cyclotomic polynomials explained in Paragraph 6.1.3. The major advantage of these codes, besides their simple description using the Fourier transform, is that we explicitly have a lower bound of their correction capacity. This allows the code parameters to be adjusted as needed. Moreover, we will see in Paragraph 6.3.5 that we have an efficient decoding algorithm, which makes these codes usable in a practical way.

We are going to build a generator polynomial of a cyclic code using the cyclotomic fields presented in Paragraph 6.1.3. Indeed, since we are interested in the divisors of the polynomial $X^n - 1$, it is natural to consider the modulo p behavior of the cyclotomic polynomials Φ_n. In the following, we suppose that $\mathrm{GCD}(n, p) = 1$ (see the remark 51 in the opposite case). Recall that if we denote by r the order of p in the multiplicative group $(\mathbb{Z}/n\mathbb{Z})^*$, then $K \overset{\text{def.}}{=} \mathbb{F}_{p^r}$ is a breaking field of the polynomial Φ_n. This therefore allows us to choose $\alpha \in K$ an $n^{\text{ième}}$ primitive root of unity (such a choice, let us recall, results from the choice of an irreducible factor of Φ_n modulo p, which is of degree r). We will then choose the generator polynomial P in the form

$$P = \prod_{i \in I} (X - \alpha^i), \tag{6.9}$$

where I is a subset of $\{1, \ldots, n-1\}$ to be determined. Indeed, in order to obtain a cyclic code on $\mathbb{F}_p$, the polynomial P must still have coefficients in $\mathbb{F}_p$, and not simply in K. Here is a simple but important lemma which gives us a way to see if a polynomial belongs to $\mathbb{F}_p[X]$.

Lemma 11. *A polynomial $Q \in K[X]$ belongs to $\mathbb{F}_p[X]$ if and only if it satisfies $Q(X^p) = Q(X)^p$.*

Proof. We know that an element $y \in K$ belongs to the prime sub-field $\mathbb{F}_p$ if and only if it satisfies $y^p = y$ (use the arguments of the proof of the proposition 51 considering the roots of $X^p X$). Using the *Frobenius morphism* $x \mapsto x^p$ from K in K, we see that if we denote by $Q = a_0 + \cdots + a_k X^k$, we have

$$Q(X)^p = (a_0 + a_1 X + \cdots + a_k X^k)^p = a_0^p + a_1^p X^p + \cdots + a_k^p (X^p)^k. \tag{6.10}$$

Thus, to say that $a_i^p = a_i$ for $i = 0, \ldots, k$ therefore amounts to saying that $Q(X)^p = Q(X^p)$. □

The following proposition then gives us a criterion which will allow us to effectively choose the set I.

Proposition 56. *The polynomial P belongs to $\mathbb{F}_p[X]$ if and only if I is stable by multiplication by p modulo n.*

Proof. If $P \in \mathbb{F}_p[X]$ and if β is a root of P, we see with the previous lemma that $P(\beta^p) = P(\beta)^p = 0$, consequently the set I is stable by multiplication by p.
Conversely, if I is stable by multiplication by p, then

$$P(X)^p = \prod_{i \in I} (X - \alpha^i)^p = \prod_{i \in I} (X^p - \alpha^{ip}) = \prod_{i \in I} (X^p - \alpha^i) = P(X^p),$$

so we have $P \in \mathbb{F}_p[X]$. □

We finally have in hand all the tools necessary to define the BCH codes.

Definition 43 (BCH codes)**.** *We call code BCH of assigned distance δ a cyclic code whose generator polynomial is obtained by the equation (6.9) , where I denotes the smallest cyclotomic class (i.e. the smallest set stable by multiplication by p modulo n) containing the set of indices $\{1, \ldots, \delta - 1\}$.*

Before going further in the investigation of the properties of these codes, we will give a simple way to calculate the generator polynomial once we know the decomposition on $\mathbb{F}_p$ of the cyclotomic polynomial Φ_n. To do this, we will consider the simplest cyclotomic classes that are, i.e. the classes generated by a single element $k \in \{0, \ldots, n-1\}$:

$$I_k \overset{\text{def.}}{=} \{k, kp, \ldots, kp^{s-1}\}, \tag{6.11}$$

where we denote by s the smallest integer such as $kp^s = k$ (in the case where $k = 1$, we have of course $s = r$ degree of P). More elegantly, we can consider the equivalence relation $\sim$ on $\mathbb{Z}/n\mathbb{Z}$:

$$\forall (x,\, y) \in (\mathbb{Z}/n\mathbb{Z})^2, \quad (x \sim y) \Leftrightarrow \left(\exists i,\ x = yq^i\right).$$

The cyclotomic classes are then the equivalence classes of $\mathbb{Z}/n\mathbb{Z}$ for this relation, and form a partition of $\{0, \dots, n-1\}$. We notice that the polynomial

$$P_k \overset{\text{def.}}{=} \prod_{i \in I_k} (X - \alpha^i) \tag{6.12}$$

is, according to the proposition 56, an irreducible polynomial of $\mathbb{F}_p[X]$ of degree s admitting α^k as root. Consequently, it is the minimal polynomial of α^k. The following description of the generator polynomial of the code is then easily obtained.

Proposition 57. *The polynomial P generator of an assigned distance BCH code δ is the PPCM of the polynomials $P_1, \dots, P_{\delta-1}$ defined by the equation (6.12).*

Example 13. Here is an example of the use of cyclotomic classes, in the case of codes on the field $\mathbb{F}_2$. We want to build codes of length $n \overset{\text{def.}}{=} 2^5 - 1 = 31$, so we have here $r = 5$. Here is the list of irreducible factors of $X^n - 1$ on $\mathbb{F}_2$:

Polynomial:	Cyclotomic class:
$P_0 \overset{\text{def.}}{=} X + 1$	$I_0 \overset{\text{def.}}{=} \{0\}$
$P_1 \overset{\text{def.}}{=} 1 + X + X^2 + X^4 + X^5$	$I_1 \overset{\text{def.}}{=} \{1, 2, 4, 8, 16\}$
$P_3 \overset{\text{def.}}{=} 1 + X^3 + X^5$	$I_3 \overset{\text{def.}}{=} \{3, 6, 12, 24, 17\}$
$P_5 \overset{\text{def.}}{=} 1 + X + X^2 + X^3 + X^5$	$I_5 \overset{\text{def.}}{=} \{5, 10, 20, 9, 18\}$
$P_{11} \overset{\text{def.}}{=} 1 + X + X^2 + X^3 + X^4 + X^5$	$I_{11} \overset{\text{def.}}{=} \{11, 22, 13, 26, 21\}$
$P_{14} \overset{\text{def.}}{=} 1 + X^2 + X^5$	$I_{14} \overset{\text{def.}}{=} \{14, 28, 25, 19, 7\}$
$P_{15} \overset{\text{def.}}{=} 1 + X + X^3 + X^4 + X^5$	$I_{15} \overset{\text{def.}}{=} \{15, 30, 29, 27, 23\}$

In accordance with what we advised to do to build a cyclotomic field, we arbitrarily choose an irreducible factor of higher degree, for example P_1, and we denote α one of its roots, which will therefore be a primitive root of unity, and which can be seen as an element of $K \overset{\text{def.}}{=} \mathbb{F}_{2^5}$. The polynomial P_0 is thus the minimal polynomial of 1 associated with the class I_0, the polynomial P_3 the minimal polynomial of α^3 associated with the class I_3, etc. To build a BCH code, it suffices to judiciously choose certain cyclotomic classes, and to multiply the corresponding polynomials between them, to obtain the generator polynomial of the code. For example, if we choose the classes I_1 and I_3, we obtain the cyclotomic class $\{1, 2, 3, 4, 6, 8, 12, 16, 17, 24\}$, and we see that it is the smallest class containing $\{1, 2, 3, 4\}$. Therefore, the polynomial

$$P_1(X)P_2(X) = 1 + X + X^2 + X^3 + X^5 + X^6 + X^8 + X^9 + X^{10}$$

generates an assigned distance BCH code 5. More substantial examples are easily constructed from the Maple program presented in Section C.4, and which allows you to construct BCH codes of arbitrary parameters. Before studying the relationships that can exist between δ and the correctness of the code, we will introduce a matrix formalism which will naturally lead to the use of the Fourier transform.

The polynomial P that we have just built is the polynomial of $\mathbb{F}_p[X]$ of lowest degree having for root $\alpha, \alpha^2, \dots, \alpha^{\delta-1}$. Consequently, the polynomials $Q \in \mathbb{F}_p[X]$ which constitute the code are therefore the polynomials of degree less than n which satisfy

$$\forall i \in \{1, \dots, \delta-1\}, \quad Q(\alpha^i) = 0. \tag{6.13}$$

Matrix, if we denote by $q = \{q_0, \dots, q_{n-1}\}$ the vector made up of the coefficients of Q, the previous equation is equivalent to

$$Aq = 0 \qquad \text{with} \qquad A \overset{\text{def.}}{=} \left\{\alpha^{ij}\right\}_{i=0,\dots,\,\delta-1}^{j=0,\dots,\,n-1} \in M_{\delta-1,n-1}(K).$$

We can also use the language of the discrete Fourier transform. We denote by $\mathcal{F}$ the transform obtained with the root $n^{\text{ième}}$ primitive α^{-1}, as defined in the equation (6.2) (taking $\zeta = \alpha^{-1}$). The condition (6.13) then becomes

$$\forall i \in \{1, \dots, \delta - 1\}, \quad Q(\alpha^i) = \hat{q}[i] \overset{\text{def.}}{=} \mathcal{F}(q)[i] = 0.$$

In other words, the code is now defined in terms of spectral conditions, more precisely, by the nullity of $\delta - 1$ components of the transformed vector. By using the notion of control matrix introduced in the remark 56, we can therefore say that the $\delta - 1$ lines of the Fourier matrix constitute a control matrix for the BCH code considered .

Remark 58. (**Duality and minimum distance**). Intuitively, we begin to understand what effect the use of a high δ can have. By forcing the transformed vector to have a very reduced support (by the nullity conditions that we have just explained), we also force the original vector (that is to say a word of the code) to have a lot of non-zero components. This is in agreement with the principle of discrete uncertainty, such as it is stated in the exercise 11 (the result extends without difficulty to the case of the Fourier transform with value in a finite field). As a result, we will obtain a large minimum distance for the code, which will therefore have a high correction rate. Once again, we see the principle of duality that we mentioned in Paragraph 4.1.4 appear. Let us clarify all this by giving a result which gives us a lower bound on the code correction rate.

Proposition 58 (Minimum distance of a BCH code)**.** *The minimum distance of the constructed $\mathcal{C}$ code is at least δ.*

Proof. It is a matter of showing that the ball with center 0 and radius $r \overset{\text{def.}}{=} \delta - 1$ contains only 0. Let $Q \in \mathcal{C}$, which is therefore a polynomial of degree at most $n - 1$. We assume that it is in this ball, so it has at most r non-zero coefficients. If we write it in the form

$$Q(X) = a_1 X^{b_1} + \cdots + a_r X^{b_r},$$

the fact that it belongs to $\mathcal{C}$ implies

$$\forall i \in \{1, \dots, r\}, \quad Q(\alpha^i) = a_1 \alpha^{ib_1} + \cdots + a_r \alpha^{ib_r}.$$

This means that the vector $a \overset{\text{def.}}{=} \{a_1, \dots, a_r\} \in \mathbb{C}^r$ satisfies the linear system $Ma = 0$, where $M \overset{\text{def.}}{=} \left\{\alpha^{ib_j}\right\}_{1 \leq i,j \leq r}$. We see that it is a matrix of *Vandermonde*, and as the α^{b_j} are distinct, it is invertible. This therefore implies that $a = 0$, which had to be demonstrated. ☐

In the following, to simplify the explanations, we will assume that $\delta = 2t + 1$, so that the code is at least t -corrector, since then $\lfloor(\delta - 1)/2\rfloor = t$. We can notice that in the case where $p = 2$, we can always assume that $\delta = 2t + 1$, since the part $\{1, \dots, \delta - 1\}$ can be assumed to be stable by multiplying by $p = 2$.

6.3.5 Fourier transform decoding

One of the advantages of the BCH codes that we have just constructed is that we have simple and fast algorithms to decode them. For example a method using *the extended Euclidean algorithm* is presented in the book of Demazure [23]. In this paragraph, we will present another algorithm, based on the description of the code in terms of a discrete Fourier transform.

Assume that we have just received a code word $x' = x + \epsilon$, where x represents the original word, and ϵ the transmission error. Our goal is to find the word x, or equivalently, to determine the error ϵ, which we write in the form

$$\epsilon(X) \overset{\text{def.}}{=} \epsilon_0 + \epsilon_1 X + \cdots + \epsilon_{n-1} X^{n-1}.$$

ϵ is unknown, but in order to have a chance to solve this problem, we still assume that it checks $w(\epsilon) \leq t$. We recall that we have assumed that $\delta = 2t + 1$, so that the proposition 58 assures us that the problem is well posed. But we need to find a way to solve it effectively.

We know, by the definition of the code in terms of Fourier transform, that

$$\forall i \in \{1, \ldots, 2t\}, \quad \hat{\epsilon}[i] = \hat{x'}[i] - \hat{x}[i] = \hat{x'}[i].$$

We therefore know how to calculate $2t = \delta - 1$ coefficients of the vector $\hat{\epsilon}$. The others still have to be calculated, in order to be able, by inverse Fourier transform, to find the error ϵ. To do this, we introduce an intermediate unknown, another polynomial.

Definition 44 (Error locator polynomial)**.** *We denote*

$$J \overset{\text{def.}}{=} \{i \in \{0, \ldots, n-1\} \; : \; \epsilon_i \neq 0\}.$$

We have already assumed that $\mathrm{Card}(J) \leq t$*, since* $w(x) \leq t$*. We call error locator polynomial, and we denote by* σ*, the polynomial*

$$\sigma(Z) \overset{\text{def.}}{=} \prod_{i \in J} \left(1 - \alpha^i Z\right) \overset{\text{def.}}{=} 1 + \sigma_1 Z + \cdots + \sigma_t Z^t.$$

The error locator polynomial is therefore a polynomial of degree at most t, comprising t a priori unknown coefficients. The inverses of the roots of σ correspond to α^i, where i is the position of an error in the transmitted word.

We notice that the polynomials ϵ and $\hat{\sigma}$ have an orthogonality property, in the sense that

– if $s \in J$, $\hat{\sigma}[s] \overset{\text{def.}}{=} \sigma(\alpha^{-s}) = 0$ and $\epsilon[s] \neq 0$.

– if $s \notin J$, $\hat{\sigma}[s] \overset{\text{def.}}{=} \sigma(\alpha^{-s}) \neq 0$ and $\epsilon[s] = 0$.

We can summarize this by the equation $\hat{\sigma} \cdot \epsilon = 0$, where we denote by $\cdot$ the multiplication coefficient by coefficient of the polynomials. Using the convolution theorem (which is still valid for a transform over a finite field, as explained by the proposition 50), we obtain by passing to the Fourier transform l'convolution equation

$$\mathcal{F}^{-1}(\hat{\sigma} \cdot \epsilon) = \sigma * \mathcal{F}^{-1}(\epsilon) = \frac{1}{N} \sigma * \hat{\epsilon}^{\sharp} = 0.$$

We recall that $\hat{\epsilon}^{\sharp}$, considered as a vector, is obtained according to the definition 23. So this is the vector $\{\hat{\epsilon}[0], \hat{\epsilon}[n-1], \ldots, \hat{\epsilon}[1]\}$. Then it suffices to replace the convolution of vectors by the modulo $Z^n - 1$ multiplication of the polynomials, to obtain a fairly complex polynomial equation:

$$\left(1 + \sigma_1 Z + \cdots + \sigma_t Z^t\right) \left(\hat{\epsilon}_0 + \hat{\epsilon}_{n-1} Z + \cdots + \hat{\epsilon}_1 Z^{n-1}\right) \quad \mod Z^n - 1 = 0.$$

We can replace this polynomial equation in two systems of linear equations:

$$(\mathcal{S}_1) \left\{ \begin{array}{lllllllllll} \hat{\epsilon}_0 & + & \hat{\epsilon}_1 \sigma_1 & + & \hat{\epsilon}_2 \sigma_2 & + & \ldots & + & \hat{\epsilon}_t \sigma_t & = & 0 \\ \hat{\epsilon}_{n-1} & + & \hat{\epsilon}_0 \sigma_1 & + & \hat{\epsilon}_1 \sigma_2 & + & \ldots & + & \hat{\epsilon}_{t-1} \sigma_t & = & 0 \\ \ldots & & & & & & & & & & \\ \hat{\epsilon}_{n-i} & + & \hat{\epsilon}_{n-i+1} \sigma_1 & + & \hat{\epsilon}_{n-i+2} \sigma_2 & + & \ldots & + & \hat{\epsilon}_{t-i} \sigma_t & = & 0 \\ \ldots & & & & & & & & & & \\ \hat{\epsilon}_{t+1} & + & \hat{\epsilon}_{t+2} \sigma_1 & + & \hat{\epsilon}_{t+3} \sigma_2 & + & \ldots & + & \hat{\epsilon}_{2t+1} \sigma_t & = & 0 \end{array} \right.$$

$$(\mathcal{S}_2) \left\{ \begin{array}{lllllllllll} \hat{\epsilon}_t & + & \hat{\epsilon}_{t+1} \sigma_1 & + & \hat{\epsilon}_{t+2} \sigma_2 & + & \ldots & + & \hat{\epsilon}_{2t} \sigma_t & = & 0 \\ \ldots & & & & & & & & & & \\ \hat{\epsilon}_2 & + & \hat{\epsilon}_3 \sigma_1 & + & \hat{\epsilon}_4 \sigma_2 & + & \ldots & + & \hat{\epsilon}_{t+2} \sigma_t & = & 0 \\ \hat{\epsilon}_1 & + & \hat{\epsilon}_2 \sigma_1 & + & \hat{\epsilon}_3 \sigma_2 & + & \ldots & + & \hat{\epsilon}_{t+1} \sigma_t & = & 0 \end{array} \right.$$

As we know the values of $\hat{\epsilon}_1, \ldots, \hat{\epsilon}_{2t}$, the system $(\mathcal{S}_2)$ allows us to calculate $\sigma_1, \ldots, \sigma_t$ in a very simple way (the system is actually triangular). We can now use the system $(\mathcal{S}_1)$ to find $\hat{\epsilon}_0$, $\hat{\epsilon}_{2t}, \ldots, \epsilon_{n-1}$, since we have nt equations for only $n - 2t$ unknowns.

Remark 59. Once we have calculated $\sigma_1, \ldots, \sigma_t$ (by solving the system $\mathcal{S}_2$), we have another alternative. Indeed, since we know σ, it is possible to test, for $i = 0, \ldots, n-1$, if $\sigma(\alpha^{-i}) = 0$, and thus to detect the positions of the errors. The system $\mathcal{S}_1$ being very easy to solve, the two methods are however equivalent.

This decoding method is implemented using Maple in Paragraph C.4. It uses the routines defined in Paragraph C.1, to perform Fourier transforms with value in $\mathbb{F}_p$.

6.4 Correction codes and duality on a finite abelian group

In this last part, we will see how the tools developed in the previous chapters can be useful to study the characteristics of a code. Thus, the notions of orthogonality, of duality on a finite group, and of course the various transformations which are linked to all these notions (Walsh and Fourier transform among others) will intervene in turn.

The fundamental book on the combinatorial study of corrective codes is that of MacWilliams and Sloane [46]. This paragraph takes up the main results on duality, by exposing them through the language which is now familiar to us, that of group algebras $\mathbb{C}[G]$ and characters. It is important to take a look at the proposed exercises, the exercise 66 for example, proposes a construction of very efficient nonlinear codes.

6.4.1 Enumerator polynomials of weight

In the following, we will restrict ourselves to the study of binary codes, but all the results given extend to the case of codes defined on any finite field $\mathbb{F}_q$. The appropriate additive characters must be used, and we refer to the exercise 19 to obtain the corresponding MacWilliams formula. In the following pages, we will consider $\mathcal{C}$, a linear code on $\mathbb{F}_2$, of size n and dimension m.

Definition 45 (Orthogonal of a code)**.** *We recall that we have a canonical non-degenerate symmetric bilinear form on $(\mathbb{F}_2)^n$, already introduced in Paragraph 2.3.2:*

$$\forall (x,\, y) \in (\mathbb{F}_2)^n \times (\mathbb{F}_2)^n, \quad \langle x,\, y\rangle \overset{\text{def.}}{=} \sum_{i=0}^{n-1} x_i y_i. \tag{6.14}$$

We denote by $\mathcal{C}^\perp$ the orthogonal code of $\mathcal{C}$ for this bilinear form, that is to say:

$$\mathcal{C}^\perp \overset{\text{def.}}{=} \{x \in (\mathbb{F}_2)^n \ : \ \forall y \in \mathcal{C},\ \langle x,\, y\rangle = 0\}\,.$$

It is therefore a code of size n and dimension nm.

Remark 60. (**Dual code**). We also speak of *dual code* to denote $\mathcal{C}^\perp$. This name is very natural, since we have already seen in Section 2.3 the similarities (and even the identity in the case of $(\mathbb{F}_2)^n$) between the notions of orthogonality , duality over a vector space, and duality over a finite abelian group.

Remark 61. (**Auto-dual codes**). It is important to emphasize that, even if we still have $\dim(\mathcal{C})+\dim(\mathcal{C}^\perp) = n$, a code and its dual are usually not additional. It sometimes happens that we have $\mathcal{C} \subset \mathcal{C}^\perp$, and we speak of code *auto-orthogonal*. When we have $\mathcal{C}^\perp = \mathcal{C}$, we say that the code is *auto-dual*. This notion is studied in more detail in the exercise 88. The fact that the generator matrix can at the same time serve as a control matrix makes it possible to simplify the decoding procedures.

Definition 46 (Enumerator polynomial)**.** *We denote by $W_\mathcal{C} \in \mathbb{Z}[X,\, Y]$ the enumerator polynomial of weight of $\mathcal{C}$, which is defined by*

$$W_\mathcal{C}(X,\, Y) \overset{\text{def.}}{=} \sum_{i=0}^{n} A_i X^{ni} Y^i,$$

where we noted $\{A_i\}_{i=0}^n$ the weight distribution of $\mathcal{C}$, defined by the equation (6.6).

The fundamental result for the determination of the enumerator polynomial is the identity of *MacWilliams*, already demonstrated in the theorem 5, and which we recall here.

Théorem 9 (MacWilliams identity)**.** *We have*

$$W_{\mathcal{C}^\perp}(X,\, Y) = \frac{1}{2^m} W_\mathcal{C}(X+Y,\, XY).$$

Several exercises propose using all these tools in order to obtain information of a combinatorial nature on the corrective codes. The exercise 63 proposes to calculate the enumerator polynomials for the Hamming codes. The 67 exercise studies the distribution of the weights of words in the *MDS* codes (ie those which are optimal for the Singleton bound). Finally, the exercise 88 studies auto-dual codes, using invariant theory techniques to exploit MacWilliams identities.

6.4.2 Algebra of a group and corrective codes

All these combinatorial techniques are therefore very useful for analyzing the structure of a linear code. However, they fall short when it comes to studying a nonlinear code, i.e. studying the distribution of the words of a set $\mathcal{C} \subset (\mathbb{F}_2)^n$. We recall that the essential notion to study a nonlinear code is not the weight distribution $\{A_i\}_{i=0}^n$ but the distance distribution $\{B_i\}_{i=0}^n$, defined by the equation (6.7). In the nonlinear case, these two distributions do not coincide, and it can be very complex to determine them. However, we will see that by using the Fourier transform on the algebra $\mathbb{C}[(\mathbb{F}_2)^n]$, we can obtain a lot of information about $\mathcal{C}$. For convenience, we denote by $G \overset{\text{def.}}{=} (\mathbb{F}_2)^n$, which can be seen as an additive group. We recall that $\mathbb{C}[G]$ denotes the algebra of functions from G in $\mathbb{C}$. The characters of G are denoted, for $a \in G$,

$$\chi_a : \left\{ \begin{array}{ccc} G & \longrightarrow & \mathbb{C}^* \\ x & \longmapsto & (-1)^{\langle a,\, x\rangle} \end{array} \right. .$$

We recall the definition of the Fourier transform of $f \in \mathbb{C}[G]$, already given in (1.10):

$$\hat{f} : \left\{ \begin{array}{ccc} G & \longrightarrow & \mathbb{C} \\ a & \longmapsto & \sum_{x\in G} \chi_a(x) f(x) \end{array} \right. .$$

With these definitions in mind, let's explain how we can represent a set of vectors of $(\mathbb{F}_2)^n = G$ as a function of $\mathbb{C}[G]$.

Definition 47 (Indicator function)**.** *Let $\mathcal{C} \subset G$ (which is not necessarily a linear code). We define the indicator function of $\mathcal{C}$ by*

$$f_{\mathcal{C}} \overset{\text{def.}}{=} \sum_{x\in\mathcal{C}} \delta_x.$$

This is the function that is worth 1 over $\mathcal{C}$, and zero everywhere else. We can thus identify the subsets of G (that is to say the arbitrary codes) with functions of $\mathbb{C}[G]$.

These indicator functions are studied in detail in the exercise 4. This exercise makes in particular the link between the spectral properties of the function $f_{\mathcal{C}}$ and the "regularity" of $\mathcal{C}$. Let us dwell for a moment on the case of linear codes. The question that naturally arises is whether there is a relationship between the indicator function of $\mathcal{C}$ and that of $\mathcal{C}^\perp$. We have already seen in the equation (2.27), that

$$x \in \mathcal{C}^\perp \quad \Leftrightarrow \quad \forall t \in \mathcal{C},\ \langle x,\, t\rangle = 0 \quad \Leftrightarrow \quad \forall t \in \mathcal{C},\ \chi_x(t) = 1. \tag{6.15}$$

This property is fundamental; it will allow us to calculate the Fourier transform of the function $f_{\mathcal{C}}$.

Proposition 59 (Transform of an indicator function)**.** *Let $\mathcal{C}$ be a linear code. We then have*

$$\hat{f_{\mathcal{C}}} = |\mathcal{C}| f_{\mathcal{C}^\perp}.$$

Proof. If $x \in \mathcal{C}^\perp$, then the equation (6.15) tells us that

$$\hat{f_{\mathcal{C}}}(x) = \sum_{t\in\mathcal{C}} \chi_x(t) = \sum_{t\in\mathcal{C}} 1 = |\mathcal{C}|.$$

Likewise, if $x \notin \mathcal{C}^\perp$, the equation (6.15) tells us that there exists $t_0 \in \mathcal{C}$ such that

$$\chi_x(t_0) \neq 1, \quad \text{i.e.} \quad \chi_x(t_0) = -1.$$

So we get

$$-\hat{f_{\mathcal{C}}}(x) = \sum_{t\in\mathcal{C}} \chi_x(t_0)\chi_x(t) = \sum_{t\in\mathcal{C}} \chi_x(t+t_0) = \hat{f_{\mathcal{C}}}(x),$$

the last equality resulting from the fact that $\mathcal{C}$ is a vector subspace of G (or an additive subgroup, changing the vocabulary). So we have

$$x \notin \mathcal{C}^\perp \Longrightarrow \hat{f_{\mathcal{C}}}(x) = 0.$$

Our goal is to extend the notion of duality to any sets $\mathcal{C}$. It would be tempting to study the space formed by vectors orthogonal to $\mathcal{C}$. However, the construction which will make it possible to obtain information on $\mathcal{C}$ is more complex. Indeed, it is a question of constructing a dual function of the indicator function $f_{\mathcal{C}}$. For the sake of generality, we are going to define the dual function of any element of $\mathbb{C}[G]$, from the moment when its mean is not zero.

Definition 48 (Dual function)**.** *Let $f \in \mathbb{C}[G]$ such that $M_f \overset{\text{def.}}{=} \sum_{x\in G} f(x) \neq 0$. We define the dual function of f, denoted $f^\perp$ by $f^\perp \overset{\text{def.}}{=} \frac{1}{M_f}\hat{f}$.*

We can therefore state the following important result.

Proposition 60. *If $\mathcal{C}$ is a linear code, we have $(f_{\mathcal{C}})^\perp = f_{\mathcal{C}^\perp}$.*

Proof. It suffices to notice that for $f = f_{\mathcal{C}}$, we have $M_f = |\mathcal{C}|$. It only remains to apply the proposition 59. □

Still with the idea of extending the notions specific to linear codes to more general codes, let us define the weight enumerator polynomial of a function.

Definition 49 (Enumerator polynomial of a function)**.** *Let $f \in \mathbb{C}[G]$.*
For $i = 0, \dots, n$, we define

$$A_i \overset{\text{def.}}{=} \sum_{w(x)=i} f(x),$$

as well as the enumerator polynomial of weight of f:

$$W_f(X, Y) \overset{\text{def.}}{=} \sum_{i=0}^{n} A_i X^{ni} Y^i.$$

We see that this polynomial generalizes the one defined in 46, since we have, for $\mathcal{C}$ a linear code,

$$W_{\mathcal{C}}(X, Y) = W_{f_{\mathcal{C}}}(X, Y).$$

The question is whether the identities of MacWilliams are still valid. The answer is yes, and we can reformulate these identities with the vocabulary of corrective codes.

Théorem 10 (MacWilliams identity for functions)**.** *Let $f \in \mathbb{C}[G]$ such that $M_f \neq 0$. We then have*

$$W_{f^\perp}(X, Y) = \frac{1}{M_f} W_f(X + Y, XY). \tag{6.16}$$

Proof. We have, using the definition of $f^\perp$,

$$\begin{aligned} W_{f^\perp}(X, Y) &= \sum_{x\in G} \frac{1}{M_f} \left(\sum_{y\in G} \chi_x(y) f(y) \right) X^{nw(x)} Y^{w(x)} \\ &= \frac{1}{M_f} \sum_{y\in G} f(y) \sum_{x\in G} \chi_x(y) X^{nw(x)} Y^{w(x)}. \end{aligned}$$

However, we have already calculated the internal sum during the proof of MacWilliams' theorem 5:

$$\sum_{x\in G} \chi_x(y) X^{nw(x)} Y^{w(x)} = (X + Y)^{nw(y)} (XY)^{w(y)},$$

and we get to the desired result. □

We see that if we apply the identity (6.16) to the indicator function of a linear code, we find the identity of MacWilliams for linear codes.

6.4.3 Combinatorial study of arbitrary codes

We will now see how we can apply all these constructions performed on the algebra $\mathbb{C}[G]$ to the study of a code. So let $\mathcal{C} \subset (\mathbb{F}_2)^n$ be any set. We can see $\mathcal{C}$ as any corrective code, not necessarily linear.

Definition 50 (Distance function). *The distance function $D_\mathcal{C} \in \mathbb{C}[G]$ is defined by:*

$$D_\mathcal{C} \overset{\text{def.}}{=} \frac{1}{|\mathcal{C}|} f_\mathcal{C} * f_\mathcal{C},$$

where $*$ denotes the convolution product of the functions over G, as we have defined in the equation (1.15).

This function can be calculated explicitly:

$$D_\mathcal{C} = \frac{1}{|\mathcal{C}|} \sum_{x\in\mathcal{C}} \sum_{y\in\mathcal{C}} \delta_{x+y}. \tag{6.17}$$

In particular, we see that if $\mathcal{C}$ is a linear code, we have $D_\mathcal{C} = f_\mathcal{C}$. In the following, we denote the enumerator polynomial of weight of $D_\mathcal{C}$ in the form

$$W_{D_\mathcal{C}} = \sum_{i=0}^{n} D_i X^{ni} Y^i.$$

We then have the following proposition, which allows us to simply obtain information on the distribution of the words of $\mathcal{C}$ with respect to each other.

Proposition 61 (Distance distribution). *$\{D_i\}_{i=0}^n$ represents the distance distribution of $\mathcal{C}$, that is:*

$$D_i = B_i \overset{\text{def.}}{=} \frac{1}{|\mathcal{C}|} \# \left\{ (x,\, y) \in \mathcal{C}^2 \ :\ d(x,\, y) = i \right\},$$

where $d(x,\, y)$ denotes the Hamming distance between x and y.

Proof. The equation (6.17) can be put in the form

$$D_\mathcal{C} = \sum_{i=0}^{n} \frac{1}{|\mathcal{C}|} \sum_{d(x,\,y)=i} \delta_{xy}. \tag{6.18}$$

Now we have, by definition, $D_i = \sum_{w(z)=i} D_\mathcal{C}(z)$. Which therefore gives, using (6.18),

$$D_i = \frac{1}{|\mathcal{C}|} \sum_{d(u,\,v)=i} 1,$$

which is exactly the desired result. □

Remark 62. We can notice that the reasoning carried out to prove the proposition 61 uses the fact that, for $(x,\, y) \in (\mathbb{F}_2^n)^2$, we have $x + y = xy$. To extend this proposition to an arbitrary finite field $\mathbb{F}_q$, it is necessary to introduce, for $f \in \mathbb{C}[(\mathbb{F}_q)^n]$ the symmetrized function $\tilde{f} : x \mapsto f(-x)$. We must then define the distance function as follows:

$$D_\mathcal{C} \overset{\text{def.}}{=} \frac{1}{|\mathcal{C}|} f_\mathcal{C} * \tilde{f}_\mathcal{C}.$$

We check without problem that the proposition 61 is still valid, like the rest of the results of this paragraph.

In the linear case, we therefore know that the enumerator polynomial of $D_{\mathcal{C}}^{\perp} = D_{\mathcal{C}^{\perp}}$ will represent the distance distribution of the dual code $\mathcal{C}^{\perp}$. It is therefore natural to study the generalization of this process to nonlinear codes. This therefore means studying the enumerator polynomial of the function $D_{\mathcal{C}}^{\perp}$, which a priori has no reason to have interesting properties. To calculate this function, notice that, according to (6.17),

$$M_{D_{\mathcal{C}}} = \sum_{x \in \mathcal{C}} \sum_{y \in \mathcal{C}} \frac{1}{|\mathcal{C}|} = |\mathcal{C}|. \tag{6.19}$$

The theorem 10 allows us to calculate, from $W_{D_{\mathcal{C}}}$, the polynomial of the dual function. For now, let's just write it down as

$$W_{D_{\mathcal{C}}^{\perp}} = \sum_{i=0}^{n} B_i' X^{ni} Y^i. \tag{6.20}$$

We can notice that, by definition of the dual function and using (6.19), the B_i' are

$$B_i' = \frac{1}{|\mathcal{C}|} \sum_{w(x)=i} \hat{D}_{\mathcal{C}}(x),$$

although it is less easy to use this expression rather than the result of the 10 theorem. A priori, the numbers B_i' have no particular property. In particular, there is no reason why the B_i' represent the distance distribution of a code. Indeed, $\mathcal{C}$ not being linear, the dual code $\mathcal{C}^{\perp}$ is not defined: everything therefore relies on the *function* dual $f_{\mathcal{C}}^{\perp}$. For example, B_i' has no reason to be whole!However, here is a simple result that clarifies things.

Proposition 62. *The B_i' are positive rational numbers.*

Proof. It suffices to use the convolution theorem 2 to rewrite the B_i' in the form

$$B_i' = \frac{1}{|\mathcal{C}|^2} \sum_{w(x)=i} \mathcal{F}(f_{\mathcal{C}} * f_{\mathcal{C}})(x) = \frac{1}{|\mathcal{C}|^2} \sum_{w(x)=i} \hat{f}_{\mathcal{C}}(x)^2 \geq 0.$$

This apparently innocuous property allows us to demonstrate a very fine inequality on the maximum size of codes of size n and minimum distance d. This development requires the introduction of the polynomials of *Krawtchouk*, which are defined at the start of the exercise 67. The exercise 68 details the steps that allow this inequality to be demonstrated.

Example 14. The *codes of Hadamard* are defined in the exercise 66. We consider the example of the code $\mathcal{A}_8$, calculated thanks to the quadratic residuals modulo 7. The vectors which compose it are given to the equation (6.22). It is a simplex code containing the zero vector, so its weight distribution is equal to its distance distribution:

$$A_0 = B_0 = 1, \quad A_4 = B_4 = 7 \quad \text{and} \quad \forall i \notin \{0, 4\}, \quad A_i = B_i = 0.$$

We therefore obtain the following enumerator polynomial of weight:

$$W_{D_{\mathcal{A}_8}}(X, Y) = X^7 + 7X^3Y^4.$$

The dual distance distribution is calculated as follows:

$$\begin{aligned} W_{D_{\mathcal{A}_8}^{\perp}} &= \frac{1}{8} W_{D_{\mathcal{A}_8}}(X+Y, XY) = \frac{1}{8}\left((X+Y)^7 + 7(X+Y)^3(XY)^4\right) \\ &= X^7 + 7X^4Y^3 + 7X^3Y^4 + Y^7. \end{aligned}$$

Which gives

i	1	3	4	7
B_i'	1	7	7	1

.

For a Hadamard code H_{12}, we get

$$\begin{aligned} W_{D^{\perp}_{\mathcal{A}_{12}}}(X,\,Y) &= \frac{1}{12} W_{D_{\mathcal{A}_{12}}}(X+Y,\,XY) \\ &= \frac{1}{12}\left((X+Y)^{11} + 11(X+Y)^5 (XY)^6\right) \\ &= X^{11} + \frac{55}{3}Y^3X^8 + \frac{110}{3}Y^4X^7 + \frac{88}{3}Y^5X^6 + \frac{88}{3}Y^6X^5 \\ &\quad + \frac{110}{3}Y^7X^4 + \frac{55}{3}Y^8X^3 + Y^{11}. \end{aligned}$$

Which gives

i	1	3	4	5	6	7	8	11
B'_i	1	$18\frac{1}{3}$	$39\frac{2}{3}$	$29\frac{1}{3}$	$29\frac{1}{3}$	$36\frac{2}{3}$	$18\frac{1}{3}$	1

.

We see that we have $B'_i \geq 0$ and that

$$\sum_{i=0}^{n} B'_i = \frac{512}{3} = \frac{2^{11}}{|\mathcal{A}_{12}|}.$$

This is quite normal, since $\mathcal{C}$ is a code that contains 0 and that

$$\mathcal{W}_{D^{\perp}_{\mathcal{C}}}(1,\,1) = \frac{1}{|\mathcal{C}|}\mathcal{W}_{D_{\mathcal{C}}}(2,\,0) = \frac{2^n}{|\mathcal{C}|}.$$

6.5 Exercises

Exercise 56 (Cyclotomic polynomials)**.** *Using Maple, show that the smallest values of n for which Φ_n has a coefficient equal to ± 1, ± 2, ± 3, ... are*

0, 105, 385, 1365, 1785, 2805, 3135, 6545, 6545, 10465, 10465,
10465, 10465, 10465, 11305, 11305, 11305, 11305, 11305,
11305, 11305, 15015, 11305, 17255, 17255, 20615, 20615, ...

Exercise 57 (Existence of a principal root)**.** *This exercise details the steps of the proof of the proposition 52. It is a question of finding a condition so that we can construct a Fourier transform on $\mathbb{Z}/m\mathbb{Z}$, where m is written in the form $m = p_1^{k_1} \times \cdots \times p_r^{k_r}$ and the p_i are distinct prime numbers. We recall Euler's theorem: if x is invertible in $\mathbb{Z}/s\mathbb{Z}$, then $x^{\Phi(s)} = 1 \mod s$. We have denoted $\Phi(s)$ the Euler function, i.e. the number of invertibles in $\mathbb{Z}/s\mathbb{Z}$.*

1. *Explain why there is a n^{rd} principal root in $\mathbb{F}_p$, for p prime, if and only if $n|p-1$. Let ζ be such a root, which we assimilate to its representative in $\{0, \ldots, p-1\}$, seen as a subset of $\mathbb{Z}/p^r\mathbb{Z}$.*
2. *We go to $\mathbb{Z}/p^r\mathbb{Z}$, and we write $\zeta_0 \overset{\text{def.}}{=} \zeta^{p^{r-1}}$. Show that ζ is invertible in $\mathbb{Z}/p^r\mathbb{Z}$ then that $\zeta_0^{p-1} = 1$.*
3. *Show that in $\mathbb{F}_p$, and for $s = 1, \ldots, n-1$,*
$$\zeta^s - 1 = \left(\zeta^{p^{r-1}}\right)^s - 1 = \zeta_0^s - 1.$$
Deduce that $\zeta_0^s - 1$ is prime with p^r and therefore that ζ_0 is a principal n^{th} root of the unit in $\mathbb{Z}/p^r\mathbb{Z}$.
4. *Using the Chinese theorem, conclude.*

Exercise 58 (Principal dyadic roots)**.** *We want to prove the proposition 53, which gives a simple criterion to find a main $(2^s)^{th}$ root in a commutative ring A.*

1. *Show that for $\zeta \in A$ to be a $(nm)^{ith}$ principal root of the unit, it is necessary and sufficient that ζ^m be a rootnth principal, and that ζ^n is a root m^{th} principal.*

2. *What are the main square roots of unity ? Specify under what condition they exist.*

3. *Show by induction on k the proposition 53*

Exercise 59 (Boolean functions)**.** *This exercise uses the Walsh transform defined in Section 2.2 . A function $\tilde{f} : (\mathbb{F}_2)^n \to \mathbb{F}_2$ is called a boolean function with n arguments. In a practical way, we can also represent such a function by the real function $f \overset{\text{def.}}{=} (-1)^{\tilde{f}}$ with values in $\{-1,\, 1\}$, which allows to calculate the transform of Walsh $\mathcal{W}(f)$:*

$$\forall k \in (\mathbb{F}_2)^n, \quad \mathcal{W}(f)(k) \overset{\text{def.}}{=} \sum_{t \in (\mathbb{F}_2)^n} f(t)(-1)^{\langle t,\, k\rangle}.$$

In the following, we will juggle these two types of representations f and $\tilde{f}$. A boolean function $\tilde{f}$ is said to be affine if it is written

$$\forall x \in (\mathbb{F}_2)^n, \quad \tilde{f}(x) = \tilde{f}_{a,b}(x) \overset{\text{def.}}{=} \langle x,\, a\rangle + b,$$

where $a \in (\mathbb{F}_2)^n$, $b \in \mathbb{F}_2$, and $\langle \cdot,\, \cdot\rangle$ denotes the canonical bilinear form over $(\mathbb{F}_2)^n$, already encountered at the equation (6.14). We will use the distance $d(f,\, g)$ between two Boolean functions, which is, by definition, the Hamming distance (definition 37) between the vectors $V(f) = \{\tilde{f}(x)\}_{x \in \mathbb{F}_2^n}$ and $V(g) = \{\tilde{g}(x)\}_{x \in \mathbb{F}_2^n}$ We then define the non-linearity of $\tilde{f}$ by

$$N(f) \overset{\text{def.}}{=} \inf\left\{d(f,\, f_{a,b}) \;:\; a \in (\mathbb{F}_2)^n,\; b \in \mathbb{F}_2\right\}.$$

1. *Explain why any Boolean function $\tilde{f}$ can uniquely take the following polynomial form:*

$$\forall x \in (\mathbb{F}_2)^n, \quad \tilde{f}(x) = b + a_0x_0 + \cdots + a_{n-1}x_{n-1} + \\ a_{01}x_0x_1 + a_{02}x_0x_2 + \cdots + a_{0\cdots n-1}x_0\cdots x_{n-1}.$$

   Intuitively justify the term non-linearity.

2. *Show that we have*

$$N(f) = 2^{n-1} - \frac{1}{2}\max\left\{|\mathcal{W}(f)(k)| \;:\; k \in (\mathbb{F}_2)^n - \{0\}\right\}.$$

   Deduce a fast method of calculating $N(f)$ which uses the FWT algorithm.

3. *Show that we have*

$$N(f) \leq 2^{n-1} - 2^{\frac{n}{2}-1}.$$

   We assume that n is even. Show that a function $\tilde{f}$ reaches the previous bound if and only if for all $k \in (\mathbb{F}_2)^n$, $|\mathcal{W}(f)(k)| = 2^{\frac{n}{2}}$. In Anglo-Saxon literature, they are called "bent functions". They were first introduced by Rothaus in[56].

4. *For $u \in (\mathbb{F}_2)^n$ and $v \in (\mathbb{F}_2)^m$, we set $w = (u,\, v) \in (\mathbb{F}_2)^{n+m}$. Let $\tilde{f}$ and $\tilde{g}$ be functions of n and m variables. We define $\tilde{h}$ a function of $n+m$ variables by $\tilde{h}(w) = \tilde{f}(u) + \tilde{g}(v)$. Show that $\tilde{h}$ is bent if and only if $\tilde{f}$ and $\tilde{g}$ are.*
   Show that $\tilde{f}_0(u_1,\, u_2) = u_1u_2$ is bent. Deduce the existence of bent functions for n even.

5. *We call the code of Reed-Muller of order 1 in n variables (noted $R(1,\, n)$) the vector subspace of the boolean function space formed by $f_{a,\,b}$, for $a \in \mathbb{F}_2^n$ and $b \in \mathbb{F}_2$. What are the size and minimum distance of this code? The encoding procedure consists, from the pair $(a,\, b)$, in producing the truth table $V(f_{a,\,b})$, that is to say the vector $\{f_{a,\,b}(u)\}_{u \in (\mathbb{F}_2)^n} = F_{a,\,b}$. Suggest a fast coding algorithm. For $F = V(f) \in (\mathbb{F}_2)^{2^n}$, what is the pair $(a,\, b)$ such that $d(f,\, f_{a,\,b})$ is minimal? Deduce a fast decoding algorithm.*

The determination of the least linear functions in the case where n is odd is an open problem. Strongly non-linear functions are widely used in cryptography. A full discussion of this topic is the article by Pasalic and Johansson[51].

Exercise 60 (Learning boolean functions)**.** *In this exercise, we keep the notations of the exercise 59. We propose, using some notions of probability, to perform Boolean predictions on a function f using only an approximate knowledge of its Walsh transform. This theory was originally developed by Kushilevitz and Mansour in*[37]*. We denote by $\mathbb{P}$ the uniform probability distribution over $(\mathbb{F}_2)^n$, that is to say $\forall x \in (\mathbb{F}_2)^n$, $\mathbb{P}(x) = 2^{-n}$. By representing a boolean function $\tilde{f}$ by the real function $f = (-1)^{\tilde{f}}$, we can then calculate the expectation of f:*

$$E[f] \overset{\text{def.}}{=} \frac{1}{2^n} \sum_{x\in(\mathbb{F}_2)^n} f(x) = \frac{1}{2^n}\mathcal{W}(f)(0).$$

Finally, we recall the bound of Chernoff-Hoeffding, which we can find in[49]*. Let $X_1, \ldots, X_m$ be independent variables identically distributed such that X_i either with values in $[-1, 1]$, $E[X_i] = p$ and $S_m = \sum_{i=1}^m X_i$. So we have*

$$\mathbb{P}\left[\left|\frac{S_m}{m} - p\right| \geq \lambda\right] \leq 2e^{-\frac{\lambda^2 m}{2}}.$$

In the following, we consider f an unknown Boolean function, of which we only suppose to be able to access $f(x_k)$ samples, where the x_k are drawn at random independently according to a uniform distribution. Our goal is to give, with as little information as possible, a good estimate of f. We say that we "learn" the function f.

1. *Suppose that one of the coefficients $a_\beta = \langle f, \chi_\beta\rangle = 2^{-n}\mathcal{W}(f)(\beta)$ is very large. What is the quadratic error $E[(fh)^2]$ that we make by replacing f by $h \overset{\text{def.}}{=} a_\beta\chi_\beta$?*

2. *A priori, h has no reason to be a Boolean function. We therefore replace h by $h_0 = \operatorname{sign}(h)$. Show that we then have*
$$\mathbb{P}\left(f(x) \neq h_0(x)\right) \leq E[(fh)^2].$$

3. *We are therefore interested in the approximate computation of $a_\beta = E[f\chi_\beta]$. We propose to use m samples $f(x_k)$ and to calculate the mean value:*
$$\tilde{a_\beta} \overset{\text{def.}}{=} \frac{1}{m}\sum_{k=1}^m f(x_k)\chi_\beta(x_k). \tag{6.21}$$
We therefore wish to approach the unknown function f by $\varphi_0 \overset{\text{def.}}{=} \operatorname{sign}(\tilde{a_\beta}\chi_\beta)$. Show that if $m \geq \frac{2}{\lambda^2}\ln\left(\frac{2}{\delta}\right)$ then
$$\mathbb{P}\left(|a_\beta - \tilde{a_\beta}| \geq \lambda\right) \leq \delta.$$
Deduce that we have, with a probability of at least $1-\delta$, the following increase of the error probability:
$$\mathbb{P}\left(f(x) \neq \varphi_0(x)\right) \leq 1 - a_\beta^2 + \lambda^2.$$

4. *We now want to approach a whole class of functions possessing weak guillhigh frequencies coefficients. We say that a boolean function f has a degree (α, d) if*
$$\sum_{w(s)>d} a_s^2 \leq \alpha,$$
where $w(s)$ is the Hamming weight of a word $s \in (\mathbb{F}_2)^n$. For all s such as $w(s) \leq d$, we compute $\tilde{a_s}$ by the equation (6.21). We then consider the function
$$\varphi_0 \overset{\text{def.}}{=} \operatorname{sign}(\varphi) \quad \textit{where} \quad \varphi \overset{\text{def.}}{=} \sum_{w(s)\leq d} \tilde{a_s}\chi_s.$$

Show that if we choose $m \geq \frac{2n^d}{\epsilon} \ln\left(\frac{2n^d}{\delta}\right)$, then we have

$$\mathbb{P}\left(f(x) \neq \varphi_0(x)\right) \leq \alpha + \epsilon$$

with a probability of at least $1 - \delta$.

We can note that the exercise 86 uses representation theory to find an orthonormal basis of $\mathbb{C}[(\mathbb{F}_2)^n]$ which differs from the basis of Walsh. This makes it possible to consider other methods for learning a Boolean function.

Exercise 61 (Generator and control matrices)**.** *Let $\mathcal{C}$ be a linear code of dimension m and size n on $\mathbb{F}_q$.*

1. *What is the relationship between the control matrices and the generator matrices of $\mathcal{C}$ and $\mathcal{C}^\perp$?*
2. *We now suppose that the generator matrix of $\mathcal{C}$ is in systematic form, that is to say that*

$$G = \begin{pmatrix} \mathrm{Id}_m \\ A \end{pmatrix} \quad \textit{with} \quad A \in (\mathbb{F}_q)^{(nm)\times m}.$$

   What are the advantages of such a form? How is a control matrix of $\mathcal{C}$ written?

3. *Show that any linear code $\mathcal{C}$ is equivalent to a systematic code. We say that two codes are equivalent if they differ only in the order of the symbols forming each word of the code (they have the same characteristics, in particular, the same weight distribution).*

Exercise 62 (Hamming codes)**.** *We call Hamming code on $\mathbb{F}_2$ any code $\mathcal{C}$ of length $n = 2^k - 1$ admitting as a matrix of controls a H matrix defined as follows: columns of H are all vectors of $(\mathbb{F}_2)^k - \{0\}$.*

1. *Show that its dimension is $m = 2^k - 1 - k$ and that its minimum distance is 3. How to decode an error?*
2. *By resuming the construction of BCH codes in the case where $q = 2$ and $n = 2^k - 1$, show that we can thus define a cyclic code which is a Hamming code (we will consider the cyclotomic bodies $K = \mathbb{F}_{2^k}$, and we will use the fact that if α is a primitive n^{rd} root, then α^i iterates over all K^*).*
3. *Prove that the code thus constructed is perfect, in the sense that the balls of radius 1 (the correction capacity), whose center is a word of the code, form a partition of $(\mathbb{F}_q)^n$ (here with $q = 2$ and $n = 2^k - 1$). Explain why the code defined in the example 12 is indeed a Hamming code.*
4. *Show that the dual code of $\mathcal{C}$ is a simplex code, that is to say that the distance between any two words of the code is constant. How much is this distance worth?*
5. *How to generalize the construction of Hamming codes to any finite field (we will think of using vectors representing vector lines)? Show in particular that its dimension is $\frac{q^k-1}{q-1} - k$, its size $\frac{q^k-1}{q-1}$, and its minimum distance 3.*

Exercise 63 (Enumerator polynomials and Hamming codes)**.** *We denote by H the matrix of size $k \times 2^k - 1$ having for columns all the binary representations of integers between 1 and $2^k - 1 = n$.*

1. *Explain why the code $\mathcal{C}$, of which H is a control matrix, is a Hamming code, as defined in exercise 62. Calculate the weight enumerator polynomial of the Hamming code of size 7 described in the example 12.*
2. *What is the generator matrix of the code $\mathcal{C}^\perp$? Show that each column of this matrix has a Hamming weight equal to 2^{k-1}.*
3. *Deduce that the weight enumerator polynomial of $\mathcal{C}$ is written*

$$W_{\mathcal{C}} = \frac{1}{2^k}\left((X+Y)^n + n(XY)^{\frac{n+1}{2}}(X+Y)^{\frac{n-1}{2}}\right).$$

4. *What is the number of words of weight 1 and 2? Does this agree with the results of exercise 62? Show that the number of words of weight 3 is* $\frac{1}{6}n(n-1)$.

5. *Show that the distribution of the weights is symmetric, that is,* $A_{ni} = A_i$.

Exercise 64 (Extended Hamming code)**.** *We note* $\mathcal{C}$ *the code extended Hamming code of size 8. It is the code obtained by adding a parity bit to the Hamming code of size 7 presented in the example 12. This means that all the vectors* $x \in \mathcal{C}$ *verify, modulo 2,* $\sum_{k=0}^{7} x_i = 0$. *What are the parameters of this code? List the words that compose it. Calculate its enumerator polynomial, and show that this code is self-dual.*

Exercise 65 (Repetition code)**.** *We denote by* $\mathcal{C}$ *the pure repetition code. It consists of replacing an element* $x \in \mathbb{F}_q$ *by the vector of* $(\mathbb{F}_q)^n$ *whose inputs are* x.

1. *What are its parameters (dimension, minimum distance)? What is its enumerator polynomial?*

2. *Identify the dual code* $\mathcal{C}^{\perp}$. *What is its enumerator polynomial? In which case (s) is this code auto-dual, i.e.* $\mathcal{C} = \mathcal{C}^{\perp}$*?*

Exercise 66 (Hadamard codes)**.** *The Hadamard matrices are defined in the exercise 17. Let* H_n *be such a matrix, of size* $n \times n$. *It is assumed to be normalized, that is, the entries in the first row and in the first column are equal to* 1. *We define* $\tilde{H}_n$ *the matrix obtained from* H_n *by replacing the* 1 *by* 0 *and the* -1 *by* 1. *We then define two codes:*

– *The code* $\mathcal{A}_n$, *whose words are the lines of* $\tilde{H}_n$ *from which the first column has been removed.*

– *The code* $\mathcal{B}_n$, *which is made up of the union of the words of* $\mathcal{A}_n$ *and their complements (we set* 1 *the entries equal to* 0, *and vice versa).*

1. *Are these codes linear (we can distinguish according to the construction of* H_n*)?*

2. *Show that two distinct lines of* $\tilde{H}_n$ *have* $\frac{n}{2}$ *common entries and* $\frac{n}{2}$ *different entries. What are the parameters of these two codes (size, number of elements, and minimum distance)? Deduce that* $\mathcal{A}_n$ *is a simplex code, that is to say that the distance between any two words of the code is constant.*

Here are for example the words of two codes $\mathcal{A}_8$ *generated by the method of quadratic residuals (known as Paley) and Walsh matrices (each line represents a word):*

Quadratic residuals:	*Walsh matrix:*
(0000000)	(0000000)
(1001011)	(1010101)
(1100101)	(0110011)
(1110010)	(1100110)
(0111001)	(0001111)
(1011100)	(1011010)
(0101110)	(0111100)
(0010111)	(1101001)

(6.22)

Exercise 67 (MDS codes)**.** *We consider a code of size* n *and dimension* m *on* $\mathbb{F}_2$. *We denote by* d *its minimum distance, and we recall that the code is said MDS if there is equality in the Singleton bound (6.8), i.e.* $d = n+$ *1-million. We denote by* A_i *the number of words of weight* i *in* $\mathcal{C}$, *and* A'_i *the number of words of weight* i *in* $\mathcal{C}^{\perp}$.

1. *We define the polynomials of Krawtchouk* P_k, *by*

$$\forall k = 0, \ldots, n, \quad P_k(x) \overset{\text{def.}}{=} \sum_{j=0}^{k} (-1)^j C_x^j C_{nx}^{kj}, \tag{6.23}$$

where the binomial coefficient C_x^j, for $j \in \mathbb{N}$, is defined by

$$C_x^j \overset{\text{def.}}{=} \frac{x(x-1)\cdots(x-j+1)}{j!}.$$

Show that we have

$$A'_k = \frac{1}{|\mathcal{C}|}\sum_{i=0}^{n} A_i P_k(i).$$

2. *Show that we have the following equalities:*

$$\forall k = 0, \ldots, n, \quad \sum_{i=0}^{nk} C_{ni}^k A_i = 2^{mk}\sum_{i=0}^{k} C_{ni}^{nk} A'_i.$$

We can think of formally differentiating the polynomial $P(1, Y)$ with respect to Y.

3. *We now assume that the code $\mathcal{C}$ is MDS. Explain why we have $A_i = 0$ for $1 \le i \le nm$, as well as $A'_i = 0$ for $1 \le i \le m$. Deduce that we have*

$$\forall k = 0, \ldots, m-1, \qquad \sum_{i=n-m+1}^{nk} C_{ni}^k A_i = C_n^k(2^{mk} - 1).$$

4. *Explain why the preceding identities uniquely determine the weight distribution of $\mathcal{C}$. For example, give the minimum number of non-zero weight words in an MDS code.*

Exercise 68 (Bound of linear programming)**.** *We denote by $R(n, d)$ the maximum cardinal of a code of size n and minimum distance d on $\mathbb{F}_2$. This quantity is extremely difficult to estimate in general, and we will see that by using the results of MacWilliams, we can give an upper bound, called the limit of linear programming (because this quantity appears as a solution of an optimization problem d'a linear form under linear constraints).*

1. *Let $\mathcal{C}$ be a binary code of size n. We denote by P_k the $k^{\text{ième}}$ Krawtchouk polynomial, which is defined by the equation (6.23). We denote by B_i the distribution of $\mathcal{C}$, show that we have*

$$\forall k \in \{0, \ldots, n\}, \quad \sum_{i=0}^{n} B_i P_k(i) \ge 0.$$

2. *Deduce that we have*

$$R(n, d) \le \max\left\{ \sum_{i=0}^{n} \tilde{B}_i \;\Big\backslash\; (\tilde{B}_0, \ldots, \tilde{B}_n) \in E_n^d \right\}.$$

The set E_n is defined as follows:

$$E_n^d \overset{\text{def.}}{=} \left\{ (1, 0, \ldots, 0, x_{d+1}, \ldots, x_n) \in \mathbb{R}_+^{n+1} \;\Big\backslash\; \forall k = 0, \ldots, n, \ \sum_{i=0}^{n} x_i P_k(i) \ge 0 \right\}.$$

Chapter 7

Linear representations of finite groups

This chapter deals with representation theory, which allows the notion of character and Fourier transform to be extended to non-commutative groups. This theory manages to make the connection between several fields of mathematics and to use tools specific to one discipline to solve problems formulated in the language of another:

– general algebra: the initial problem is that of studying an abstract group.
– linear algebra: the goal is to "geometrically" realize our group as a group of linear transformations. The use of matrix tools will make it possible to carry out calculations on our group, and to obtain precise information (resolubility, simplicity, conjugation classes, etc.).
– geometry: the abstract study of a geometry amounts to the study of the invariants for a given group action. Most of the actions considered are linear and representation theory naturally comes into play.

The notion of linear representation, although rather complex at first glance, is in fact at the center of many problems in the aggregation program: actions of groups, equivalent matrices, finite groups, Hermitian spaces (the space $\mathbb{C}[G]$ is naturally provided with such a structure), dimension of vector spaces, enumeration, permutation group, stable subspaces, duality, distinguished subgroups (study of simplicity).

Regarding representation theory, the main reference in French is the book of JPSerre [59]. The proof of the orthogonality of characters is quite computational, and will only be approached in the next chapter. We can look at the English reference book of Fulton and Harris [34] to find the one made here. In addition, [35] fully explains Fourier's theory on $\mathbb{C}[G]$, [1] gives a good number of character tables of classical groups, just like [19]. The history of the representation theory of finite groups is explained in the two articles of Lam [38] and [39].

7.1 First definitions

This first part is devoted to the very definition of the notion of representation, but also (and above all) to the implementation of the problematic already stated: the search, apart from isomorphism, for irreducible representations. We therefore give the definition of the notion of isomorphism of representations, as well as the details of the notions related to that of irreducibility. To highlight these definitions, many examples are exposed, whether they are fundamental (in the sense that they lead to constructions used later) or only informative (allowing to make "by hand" calculations without use the tools developed later).

7.1.1 Linear representations

Definition 51 (Linear representation)**.** *Let V be a K-vector space of finite dimension n. A linear representation of a group G in V is the data of a morphism $\rho : G \to GL(V)$. This corresponds to the data of a linear group action of G over V, noting $\forall (g, v) \in G \times V,\ gv = \rho(g)(v)$. We also say that V is a G-module (this terminology will be explained later).*

Definition 52 (Faithful representation). *A representation ρ on a vector space V is said to be faithful if ρ is injective. We also say that G acts faithfully on V.*

Example 15. The examples that must be kept in mind are geometric in nature: a faithful group action allows to realize an abstract group as a group of transformations (most often unitary or orthogonal, cf. proposition 66) of a vector space. For example, we can see the symmetric group $\mathfrak{S}_4$ as the set of isometries which keep a cube. This identification establishes a representation of the abstract group $\mathfrak{S}_4$ on the vector space $\mathbb{R}^3$ as a group of orthogonal transformations.

We have already seen in Paragraph 1.4.2, the definition as well as the main properties of the algebra of an abelian group (on the field $\mathbb{C}$ of the complexes). These definitions extend without difficulty to a noncommutative group and to any field K. We will see that this algebra allows us to define the notion of representation in another way, by using the language of modules.

Definition 53 (Algebra of a group). *We assume given a vector space V over a field K of dimension $|G|$ whose base is indexed by G, c'that is, a base of the form $\{e_g\}_{g\in G}$. We can then define a structure of K-algebra on V by the equality $e_g * e_h = e_{gh}$ which extends by bilinearity to the whole space. We denote $K[G]$ this algebra, and we often identify $g \in G$ and $e_g \in K[G]$ so that we speak of the group algebra G , and that G canonically injects itself into $K[G]$ by $g \mapsto e_g$.*

The main utility of the algebra $K[G]$ which encompasses our group G is that it satisfies a *universal property*, in the sense that any representation over G uniquely extends into a representation over $K[G]$. Let us first define the notion of algebra representation.

Definition 54 (Algebra representation). *A representation of an associative K-algebra A is the data of a vector space V on K of finite dimension and of a morphism of K-algebra $\rho : A \to \mathrm{End}(V)$.*

We can easily see that a representation of a group $\rho : G \to GL(V)$ uniquely extends linearly into an algebra representation $\tilde{\rho} : K[G] \to \mathrm{End}(V)$. So the representations of $K[G]$ correspond exactly to the representations of G. In fact, any statement concerning representations of G has an equivalent in terms of the algebra of the group G. It is simply a choice of language, each having preferences for this or that formulation.

Remark 63. (**Representations and $K[G]$- modules**). The definition of a ρ representation of the algebra $K[G]$ corresponds exactly to the definition of a $K[G]$- module to the left. The external multiplication of a vector $v \in V$ by a "scalar" $\lambda \in K[G]$ is then given by $\lambda \cdot v \overset{\text{def.}}{=} \rho(\lambda)(v)$. Conversely, the data of such a structure unambiguously defines an algebra representation. Thus, the notions of group representation, algebra representation, and $K[G]$- module are totally equivalent. In the following, we will meet the notion of G-morphism, which will correspond to the morphisms for the structure of G-module.

As we have already noticed in Paragraph 1.4.2, in the case of a finite group, the algebra of a group is identified in a natural (and canonical) way with the space of functions of G in the field K considered, endowed with a product called the convolution product. Let us recall these constructions within the framework of any finite group G and of any field K.

Remark 64. (**Functions over G and group algebra**). If we denote, for $g \in G$, δ_g the function which is worth 1 in g and 0 elsewhere, then we can decompose a function $f : G \to K$ in database $\{\delta_g\}_{g\in G}$:

$$f = \sum_{g\in G} f(g)\delta_g. \tag{7.1}$$

This identifies f with the element

$$\sum_{g\in G} f(g)e_g \quad \in K[G].$$

We will therefore identify $K[G]$ with the space of functions from G in K, whose base is given by $\{\delta_g\}_{g\in G}$.

Let us recall the formula which gives the multiplication on the algebra $\mathbb{C}[G]$, and which we call, using the vocabulary of functional analysis, *convolution product.*

Definition 55 (Convolution product)**.** *The multiplication over $K[G]$ is defined by extending the multiplication in G. In a way, we know how to multiply among themselves the elements of the base $\{\delta_g\}_{g\in G}$ (remembering that an element $\delta_g \in K[G]$ s' identifies with $g \in G$), and by bilinearity of the multiplication, we can thus calculate the product of two elements by decomposing them as in (7.1). Thus, for $(f,\, g) \in K[G]^2$ we obtain*

$$\forall s \in G, \quad (f * g)(s) \overset{\text{def.}}{=} \sum_{hk=s} f(h)g(k) = \sum_{h\in G} f(h)g(h^{-1}s).$$

This multiplication is called the product of convolution. The neutral element for this operation is δ_1 (which is identified with the neutral element 1 of G), and so we must not confuse $*$ with the component by component multiplication of the functions of G in K (usually denoted by $\cdot$), for which the neutral element is the constant function 1.

7.1.2 Basic examples

Regular representation

This representation is the most basic, and also the most important. It is by breaking down the regular representation into sub-representations that we will obtain a good deal of information on the group G, as we will see in Paragraph 7.4.2.

Definition 56. *The regular representation on the left is the representation of G on the vector space $K[G]$ defined by the morphism*

$$\forall s \in G, \quad \rho_r(s) : \begin{cases} K[G] & \longrightarrow & K[G] \\ f & \longmapsto & \delta_s * f \end{cases}.$$

Remark 65. We can, as it is explained in the definition 54 extend the regular representation into an algebra representation defined on $K[G]$ whole. We see that we simply obtain the algebra structure of $K[G]$ (for the convolution product). This means that the $K[G]$- modulus given by $K[G]$ itself corresponds to the regular representation.

Proposition 63. *Regular representation is faithful.*

Proof. If $\rho(g) = \mathrm{Id}$, then $\rho(g)(\delta_e) = \delta_e$, i.e. $\delta_g * \delta_e = \delta_g = \delta_e$, so $g = e$. □

The sum representation

The simplest operation that can be done between two representations is the direct sum, which is defined as follows.

Definition 57 (Sum representation)**.** *For two representations ρ_V and ρ_W respectively on V and W, we define a representation $\rho_{V\oplus W}$ on $V \oplus W$ by*

$$\forall g \in G,\ \forall (v,\, w) \in V \times W, \quad \rho_{V\oplus W}(g)((v,\, w)) \overset{\text{def.}}{=} \rho_V(g)(v) + \rho_W(g)(w).$$

The notion of sum is to be compared to the notion of decomposability which is approached in Paragraph 7.1.3. The question is to know under what condition a representation can be written as the sum of two others. We will see (with the theorem 11) that this is simply linked to the fact that the representation admits or not sub-representations.

The representation of morphisms

The product representation (in the sense of *tensor product* of vector spaces) does not will not be discussed. However, we can use, instead, the notion of representation on the vector space of morphisms.

Definition 58 (Representation of morphisms)**.** *For two representations ρ_V and ρ_W respectively on V and W, we define a representation $\rho_{\mathcal{L}(V,W)}$ on $\mathcal{L}(V, W)$ (space of linear maps of V in W) by*

$$\forall g \in G,\ \forall f \in \mathcal{L}(V, W), \quad \rho_{\mathcal{L}(V,W)}(g)(f) \overset{\text{def.}}{=} \rho_W(g) \circ f \circ \rho_V(g^{-1}).$$

Proposition 64. *We thus define a representation.*

Proof. We denote by $\rho \overset{\text{def.}}{=} \rho_{\mathcal{L}(V,W)}$. First of all, notice that $\rho(g)$ is indeed a linear map. The proposition results simply from the calculation, for $f \in \mathcal{L}(V, W)$ and for $\forall(g, h) \in G^2$:

$$\begin{aligned}\rho(gh)(f) &= \rho_W(gh) \circ f \circ \rho_V((gh)^{-1})\\ &= \rho_W(g)\left\{\rho_W(h) \circ f \circ \rho_V(h^{-1})\right\}\rho_V(g^{-1})\\ &= \rho(g)\left\{\rho(h)(f)\right\}.\end{aligned}$$

The dual representation

In the same order of ideas as for the representation of morphisms, we can define a dual representation.

Definition 59 (Dual representation)**.** *For a ρ representation on V, we define a ρ^* representation on V^* on dual of V per*

$$\forall g \in G, \quad \rho^*(g) \overset{\text{def.}}{=} \rho(g^{-1})^t,$$

where we denote by $\varphi^\top \in \mathcal{L}(V^*)$ the transposed operator of $\varphi \in \mathcal{L}(V)$.

Remark 66. We can see that this definition corresponds to the representation of the morphisms $\rho_{\mathcal{L}(V,K)}$ (where K is considered as a space of dimension 1), since $V^* = \mathcal{L}(V, K)$, with the trivial representation $\rho_K(g) = \text{Id}$.

Definition 60 (Duality hook)**.** *We note, for $f \in V^*$ and for $x \in V$,*

$$\langle x,\, f\rangle_{(E,E^*)} \overset{\text{def.}}{=} f(x).$$

We call this application the hook of duality, which is a bilinear form on $E \times E^*$.

We easily show that the dual representation that we have just constructed has an interesting behavior with respect to the duality hook.

Proposition 65. *The dual representation on V^* keeps the hook of the duality, that is to say:*

$$\forall g \in G,\ \forall (f, x) \in E^* \times E, \quad \langle x,\, \rho_{V^*}(g)(f)\rangle_{(E,E^*)} = \langle \rho_V(g^{-1})(x),\, f\rangle_{(E,E^*)}.$$

An action on polynomials

Definition 61. *Let G be a finite subgroup of $GL_n(K)$. If we denote, for $A \in G$, A^{-1} in the form $(a_{i,j})_{1\leq i,j\leq n}$, we define an action linear from G on $K[X_1, \ldots, X_n]$ by defining $\rho(A)(P)$ the polynomial obtained by the substitution of X_i by $\sum_{j=1}^n a_{j,i}X_j$. We denote symbolically $\rho(A)(P)(X) = P(A^{-1} \cdot X)$.*

If we consider this action on $K[X_1, \ldots, X_n]$ as a whole, we get a representation of infinite dimension. However, it is easy to see that this action respects the degree of the polynomials. We can therefore restrict this action to the subspace $K_s[X_1, \ldots, X_n]$ made up of polynomials of degree less than or equal to s. It is a finite dimensional space, and this gives rise to a finite dimensional linear representation. But we can also consider the space $K_s^0[X_1, \ldots, X_n]$ of homogeneous polynomials of degree s (including the zero polynomial), which provides a second family of finite dimensional representations . It is this point of view which will be adopted to prove the *Molien's theorem*, in the exercise 74.

Finally, note that the theory of invariant polynomials under this action is very important. The exercise 73 proposes to demonstrate a fundamental result of this theory. Within the framework of the study of corrective codes, it is these representation theory tools which allow classifying the *auto-dual* codes. An instance of this approach can be found in the exercise 88.

Representation of degree 1

We consider the case where $K = \mathbb{C}$. A representation of degree 1 is simply a morphism of G in the multiplicative group $\mathbb{C}^*$ (we identify $GL_1(\mathbb{C})$ and $\mathbb{C}^*$). It is therefore a character of G as defined in chapter 1. We thus find the classical theory of duality on a finite group. We already know that if G is abelian, the characters form a basis of the space $\mathbb{C}[G]$ of functions from G in $\mathbb{C}$. In the following, we will extend the notion of character, and we will see that this notion has the expected properties.

– In the case where G is commutative, this new notion does not add anything new: we only find the characters already defined. Intuitively, we know that the dimension 1 is sufficient to study commutative groups.
– In the non-commutative case, the addition of "new" characters makes it possible to develop a Fourier theory generalizing the theory already developed in the commutative case.

First of all, let us be satisfied with the following remark:

Remark 67. If ρ is a representation of G on a vector space V, then the application which to $s \in G$ associates $\det(\rho(s))$ is a representation of degree 1 on $\mathbb{C}$ (that is to say a character in the sense in which we have already defined it).

7.1.3 Irreducible representations

The notion of irreducible representation is very intuitive. As in any construction, we look for the "basic bricks", those with which we will be able to reconstruct the whole building (here all the representations). Our tool is, as we defined in Paragraph 7.1.2, the sum of representations. The intuitive definition of a "building block" is that it must be minimal (in the sense of including non-zero subspaces). Is this definition compatible with the construction by sum of new representations? This is what the theorem 11 will specify, in the case of an algebraically closed field. In fact, we will gradually leave the generality of the constructions of the previous paragraph, to restrict ourselves to the case of the body $K = \mathbb{C}$, for which there is already a lot to do. However, we will try, as far as possible, to mention the results that remain valid under weaker assumptions.

Definition 62 (Isomorphic representations)**.** *Two representations ρ and ρ' of the same group G respectively on two K-vector spaces V and V' are called isomorphs if there is an isomorphism of vector spaces $\tau : V \to V'$ such that for all $s \in G$, $\tau \circ \rho(s) = \rho'(s) \circ \tau$, which identifies the two representations.*

The notion of isomorphism of representations defines an equivalence relation on the representations of a given group G. In the following, we will focus on the equivalence classes for this relation. We are now going to give the definitions which make it possible to explain the notions of "basic bricks".

Definition 63 (Sub-representations)**.** *If a ρ representation of G on V admits a vector subset $W \subset V$ stable by all $\rho(s) \in GL(V)$, it induces a representation ρ_W on W called sub-representation.*

Remark 68. Using the language of $K[G]$- modules, we see that a pre sen ta tion is nothing but a sub $K[G]$-module, and that an isomorphism of representations is an isomorphism of $K[G]$- modules.

Definition 64 (Irreducible representations)**.** *A representation on a space V is called irreducible if it admits exactly two sub-representations sen tations: $\{0\}$ and V whole .*

Definition 65 (Indecomposable representation)**.** *A representation over a space V is called indecomposable if each time we have an isomorphism of representations $V \simeq W_1 \oplus W_2$, then $W_1 = \{0\}$ or $W_2 = \{0\}$.*

Remark 69. (**Irreducibility and indecomposability**). It is evident that an irreducible representation is in particular indecomposable, since a decomposition of V in the form $V \simeq W_1 \oplus W_2$ non-trivial gives rise to two sub-representations. We will now turn to the reciprocal question. To solve this problem, it is necessary to know if, given a non-trivial sub-representation W_1 of V, we can find another sub-representation W_2 such as $V \simeq W_1 \oplus W_2$. This exactly means finding an additional W_1 stable under the G share. The exercise 69 shows that in general this reciprocal aspect has no reason to be true. However, under certain restrictive assumptions on the base field, we will see that we can demonstrate the equivalence between irreducibility and indecomposability.

Important: From now on, unless explicitly stated otherwise, we will work in the body of $K = \mathbb{C}$ complexes.

Proposition 66 (Unit representation). *Let ρ be a representation of a finite group G on a space V. Then ρ leaves the following Hermitian product invariant:*

$$\langle x, y \rangle_G \overset{\text{def.}}{=} \sum_{s \in G} \langle \rho(s)(x), \rho(s)(y) \rangle,$$

where we noted $\langle \cdot, \cdot \rangle$ any Hermitian product over V.

Proof. The fact that $\langle \cdot, \cdot \rangle_G$ is invariant by G is trivial:

$$\langle \rho(g)(x), \rho(g)(y) \rangle_G \overset{\text{def.}}{=} \sum_{s \in G} \langle \rho(sg)(x), \rho(sg)(y) \rangle = \langle x, y \rangle_G.$$

The only important point is that $\langle \cdot, \cdot \rangle_G$ is indeed a Hermitian product, as the sum of Hermitian products. This is valid because we are working on the body $\mathbb{C}$ of the complexes. $\square$

Remark 70. (**Unit representation**). This result is equivalent to the fact that the M_s matrices of $\rho(s)$ are unitary in an orthonormal basis for $\langle \cdot, \cdot \rangle_G$, that is, verify $M_s M_s^* = \mathrm{Id}$, where M_s^* is the adjoining matrix. We say that M_s is a *unit matrix representation.*

Théorem 11 (Irreducibility and indecomposability). *A representation ρ on V is irreducible if and only if it is indecomposable.*

Proof. Let W_1 be a non-trivial sub-representation of V. As we have already pointed out, to prove the equivalence, it suffices to find an additional of W_1 stable by G. We can then consider the invariant Hermitian product $\langle \cdot, \cdot \rangle_G$, and choose W_2 the orthogonal of W_1. By conservation of the scalar product, the image by G of a vector orthogonal to W_1 is still orthogonal to W_1: W_2 is quite stable under the action of G. $\square$

Remark 71. The above demonstration uses the existence of a stable Hermitian product. It is therefore not valid on a body other than $\mathbb{C}$. We can however propose another approach, which allows to prove the theorem 11 in the case of a field K such that its characteristic does not divide $|G|$. Here is a second demonstration:

Proof. There is another way to build a stable supplement of W_1. Indeed, consider an arbitrary additional W_2, and denote by π the projection on W_1 associated with the decomposition $V = W_1 \oplus W_2$. We can then define an endomorphism π_0 as follows:

$$\pi_0 \overset{\text{def.}}{=} \frac{1}{|G|} \sum_{g \in G} \rho(g) \circ \pi \circ \rho(g^{-1}) \ ;$$

this is valid because $\mathrm{Car}(K)$ does not divide $|G|$. We can then see that π_0 is an image projector W_1, and even better, that its kernel $W_1 \overset{\text{def.}}{=} \mathrm{Ker}(\pi_0)$ is stable by the action of G (we can easily be checked by hand). By virtue of the properties of the fixtures, we have $V = W_1 \oplus W_2$. This construction, which may seem a little magical, is in fact very natural, and will become clear once the notions of G-morphism (definition 68), and especially of *Reynolds operator* R_G (definition 69), since we have in fact constructed $\pi_0 = R_G(\pi)$. $\square$

Remark 72. The theorem 11 means that if ρ is reducible, the matrices of $\rho(s)$ are written as a diagonal of two blocks in a base well chosen, which corresponds well to the notion of reducibility (and to the sum representation). Here is now the result which assures us that the "base bricks" which are the irreducible representations allow us to reconstruct all the representations.

Proposition 67. *Any representation can be written as a sum of irreducible representations.*

Proof. We reason by induction on the dimension of V the vector space of our representation. A space of dimension 1 is irreducible. If V is a space of dimension greater than 1 irreducible, and the proof is finished. Otherwise, V admits a non-trivial stable subspace W, and with the corollary 11, we can find W_0 stable such that we have $V = W \oplus W_0$. By applying the induction hypothesis to W and W_0, we prove what was required. $\square$

Remark 73. This writing is not unique, but we will see that it is unique "up to isomorphism", in the sense that if we have two decompositions of W in the form $W_1 \oplus \cdots \oplus W_r$ and $W'_1 \oplus \cdots \oplus W'_{r'}$, then $r = r'$, and even if it means reordering the indices, there are isomorphisms $W_i \simeq W'_i$.

7.1.4 The symmetric group

Before getting to the heart of the matter, namely the introduction of useful tools for studying representations, let us take a look at a very important group, the symmetric group $\mathfrak{S}_n$. We will try to identify "by hand" the main characteristics of its representations, failing to be able to give an exhaustive description.

Definition 66 (Representation by permutation)**.** *Let $\mathfrak{S}_n$ be the group of permutations of a set of n elements, identified with the set $\{1, \ldots, n\}$. Let V be a vector space of dimension n, of which we choose a base $\mathcal{B} = \{e_1, \ldots, e_n\}$. $\mathfrak{S}_n$ acts on V by permutation of the elements of the base $\mathcal{B}$, which allows to define a morphism $\rho_p : \mathfrak{S}_n \to GL(V)$. We will therefore note, for $\sigma \in \mathfrak{S}_n$, $\rho_p(\sigma)$ the corresponding endomorphism, i.e. such that $\rho_p(e_i) = e_{\sigma(i)}$. We denote by M_σ the associated permutation matrix, which is the matrix of $\rho_p(\sigma)$ in the base $\mathcal{B}$. This matrix has only one 1 per row and per column, and 0s everywhere else. Moreover, only $M_{\mathrm{Id}} = \mathrm{Id}$ has only 1s on the diagonal (this fact will be used for the example 17).*

Remark 74. (**Link with regular representation**). Any group G of cardinal n is injected into the group $\mathfrak{S}_n$ of the permutations of the set $\{1, \ldots, n\}$. To see it, it suffices to number the elements of $G = \{g_1, \ldots, g_n\}$ and to consider, for $h \in G$, the permutation

$$j(h) : \left\{ \begin{array}{ccc} \{1, \ldots, n\} & \longrightarrow & \{1, \ldots, n\} \\ k & \longmapsto & k' \end{array} \right. \qquad \text{where } k' \text{is such that} h g_k = g_{k'}.$$

We then have the following commutative diagram:

$$\begin{tikzcd} G \arrow[r, "\rho_r"] \arrow[d, "j"] & GL(V) \arrow[d, equal] \\ \mathfrak{S}_n \arrow[r, "\rho_p"] & GL(V) \end{tikzcd} .$$

In the same order of ideas, when we study the representation ρ_p of $\mathfrak{S}_n$ on a space of dimension n (which we can assume on any field K) , we can be interested in the determination of the similarity classes of the matrices of $\rho_p(\sigma)$, $\sigma \in \mathfrak{S}_n$. Let us start by characterizing the conjugation (ie similarity) classes in the symmetric group $\mathfrak{S}_n$.

Lemma 12 (Conjugation classes in $\mathfrak{S}_n$)**.** *Two elements of $\mathfrak{S}_n$ are in the same conjugation class if and only if their decomposition into disjoint cycles has the same number of cycles of a given length.*
In particular, there are as many classes of conjugations in $\mathfrak{S}_n$ than of partitions of n of the type

$$n = k_1 + k_2 + \cdots + k_p \quad \textit{with} \quad k_1 \geq k_2 \geq \cdots \geq k_p > 0.$$

Proof. First of all, note that two cycles of the same length are conjugated. Indeed, if we consider $c \overset{\text{def.}}{=} (c_1, \ldots, c_k)$ and $c' \overset{\text{def.}}{=} (c'_1, \ldots, c'_k)$, it suffices to use a permutation $\sigma : c_i \mapsto c'_i$ to see that $c = \sigma^{-1} \circ c' \circ \sigma$. So if two permutations have decompositions with cycles of the same length, they are conjugated.
Conversely, it is obvious that the conjugation of a permutation also conjugates the cycles which compose it, and thus preserves their lengths. □

Théorem 12 (Brauer's theorem)**.** *We place ourselves on a field K of characteristic 0. Two matrices M_σ and $M_{\sigma'}$ are similar if and only if σ and σ' are conjugated in $\mathfrak{S}_n$, that is:*

$$\exists \tau \in \mathfrak{S}_n, \quad \sigma\tau = \tau\sigma'.$$

Proof. This demonstration was communicated to me by Daniel Ferrand.
The reciprocal meaning follows directly from the definition of a representation.
Indeed, if $\exists \tau \in \mathfrak{S}_n,\ \sigma' = \tau^{-1}\sigma\tau$, then $M_{\sigma'} = M_{\tau^{-1}\sigma\tau} = M_\tau^{-1} M_\sigma M_\tau$.
For the direct meaning, we denote by $c_k(\sigma)$ the number of cycles of order k in the decomposition of σ into disjoint cycles . Using the lemma 12, it suffices to show that $\forall k \in \{1, \dots, n\},\ c_k(\sigma) = c_k(\sigma')$. Now any cycle α of order k satisfies $\alpha^k = \mathrm{Id}$, so the characteristic polynomial of M_α is $X^k - 1$. Since M_σ and $M_{\sigma'}$ have the same characteristic polynomial (because they are similar), we have

$$\prod_{k\geq 1} (X^k - 1)^{c_k(\sigma)} = \prod_{k\geq 1} (X^k - 1)^{c_k(\sigma')}. \tag{7.2}$$

For $m \in \mathbb{N}$, we take ζ a root m^{th} of the unit (in an algebraic closure $\overline{K}$ of K) of order m (i.e. primitive). Since we are in characteristic 0, $P = X^k - 1$ and $P' = kX^{k-1}$ are coprime, which means that P is split with single roots in $\overline{K}$. So the multiplicity of ζ in P is 1 if ζ is root of P (that is, if $m|k$), 0 otherwise. By equaling the multiplicities in the equality (7.2), we get

$$\sum_{m|k} c_k(\sigma) = \sum_{m|k} c_k(\sigma').$$

Now suppose that there exists m such that $c_m(\sigma) \neq c_m(\sigma')$. We choose m as large as possible, denote it m_0 (this is possible because $k \mapsto c_k(\sigma)$ is a finite support function). We then have

$$0 = \sum_{m_0|k} c_k(\sigma) - \sum_{m_0|k} c_k(\sigma') = c_{m_0}(\sigma) - c_{m_0}(\sigma'),$$

which is a contradiction, because $c_{m_0}(\sigma) \neq c_{m_0}(\sigma')$. □

After having answered these questions on the classes of similarities linked to the representation by permutation, the problem of determining the representations of $\mathfrak{S}_n$ arises. Since it is pointless to want to determine all of them (it is easy to create new ones by direct sums), the real problem is in fact determining the *irreducible* representations of $\mathfrak{S}_n$. This is a difficult question, and at first we will just give "obvious" representations. The example of the group $\mathfrak{S}_4$ will be fully covered in Paragraph 8.1.4. However, there are precise descriptions of the irreducible representations of $\mathfrak{S}_n$, which are based on the action of this group on *Young arrays*. A full description can be found in Fulton and Harris [34] book.

First of all, is the permutation representation, for $n \geq 2$, irreducible? We can easily see that not, by looking at the subspace

$$H_0 \overset{\text{def.}}{=} \{\lambda(1, \dots, 1)\ :\ \lambda \in \mathbb{C}\} = \mathrm{Span}((1, \dots, 1)). \tag{7.3}$$

We can easily see that this subspace is stable by any permutation of the coordinates, and that it admits an additional also stable by G:

$$H_1 \overset{\text{def.}}{=} \{(x_1, \dots, x_n) \in V\ :\ x_1 + \cdots + x_n = 0\}. \tag{7.4}$$

The representation by permutation ρ_p induces the trivial representation (that is to say, cons tan te equal to the identity) on H_0. The question of the irreducibility of the representation H_1 is tackled in the exercise 75. We will see in Paragraph 8.1.4, that in the example of $\mathfrak{S}_4$, this representation is indeed irreducible. We call *standard representation* the representation induced by ρ_p on H_1. Besides the trivial representation (which is of course irreducible), there remains another representation of degree one, given by the equation

$$\forall \sigma \in \mathfrak{S}_n, \quad \rho_\epsilon(\sigma) = \epsilon(\sigma),$$

where we noted $\epsilon(\sigma)$ the signature of the permutation σ (as it is a representation of degree 1, we noted $\rho_\epsilon(\sigma)$ as a scalar, while it is actually a matrix of size 1). This representation is called the alternate representation. Moreover, we have already seen in chapter 1 that these were the only two representations of degree 1 of $\mathfrak{S}_n$.

7.2 Invariance and representations

A very simple way to create globally stable subspaces under the action of a group G is to look at the set of vectors which are not modified by G, which forms well a subspace. On this invariant subspace, G induces the trivial representation. The capital interest of this subspace is that we have a complete description, and a very simple way to generate elements. The fundamental idea behind the construction made in this paragraph (and behind *Reynolds operator*, which is presented in Paragraph 7.2.3) is that we are on a finite group, and therefore that we are able to *average* the action of our group. This very simple principle, which we have already used to build additional stable ones, will be of constant use in the rest of the presentation, and that is why it is important to formalize it.

7.2.1 Invariant sub-representation

Definition 67 (Invariant sub-representation)**.** *Let ρ be a representation on V. We denote by V^G the subspace of the vectors invariants, that is to say:*

$$V^G = \left\{ v \in V \ : \ \forall s \in G, \ \rho(s)(v) \overset{\text{def.}}{=} sv = v \right\}.$$

This is an under-representation of V.

Example 16. We consider the action of the symmetric group $\mathfrak{S}_n$ by permutation of coordinates, as defined in Paragraph 7.1.2. This action allows to define another action, on $K[X_1, \ldots, X_n]$ this time, via the construction carried out in Paragraph 7.1.2. The invariant representation $K[X_1, \ldots, X_n]^{\mathfrak{S}_n}$ is formed by symmetric polynomials, which means that $P \in K[X_1, \ldots, X_n]^{\mathfrak{S}_n}$ if and only if

$$\forall \sigma \in \mathfrak{S}_n, \quad P(X_{\sigma(1)}, \ldots, X_{\sigma(n)}) = P(X_1, \ldots, X_n).$$

These polynomials, as well as the action of finite groups on the polynomials are studied in detail in the exercises 73 and 74.

Definition 68 (Interleaving operators)**.** *In the case of the representation of morphisms $\rho_{\mathcal{L}(V,W)}$ on $\mathcal{L}(V, W)$ of two representations ρ_V and ρ_W respectively on V and W, we denote by $\mathrm{Hom}_G(V, W)$ the space of invariants. We call these elements interlacing operators or G-morphisms.*

Remark 75. Saying that $f \in \mathcal{L}(V, W)$ is an interleaving operator corresponds to the fact that f checks $\forall s \in G, \ f \circ \rho_V(s) = \rho_W(s) \circ f$, i.e. f switches, for all $s \in G$, the following diagram:

$$\begin{tikzcd}
V \arrow[r, "f"] \arrow[d, "\rho_V(s)"] & W \arrow[d, "\rho_W(s)"] \\
V \arrow[r, "f"] & W
\end{tikzcd}$$

If f is bijective, this corresponds to the fact that f is an isomorphism of representations. In the general case, we speak of G-morphism, or of interlacing operator. Using the language of $K[G]$- modules, an interleaving operator is simply a morphism of $K[G]$- modules.

7.2.2 Schur lemma

This lemma, which appears very simple, is in fact the cornerstone of most of the proofs which will be made in the remainder of the talk.

Lemma 13 (Schur's lemma)**.** *Let $\rho_V : G \to GL(V)$ and $\rho_W : G \to GL(W)$ two irreducible representations of a group G. Let $f \in \mathcal{L}(V, W)$ be an interleaving operator, i.e. $f \in \mathrm{Hom}_G(V, W)$. So*

(i) *if ρ_V and ρ_W are not isomorphic, $f = 0$.*

(ii) *if $f \neq 0$, then f is an isomorphism, the representations are isomorphic, and if we assume $V = W$, $\rho_V = \rho_W$, then f is a dilation.*

Proof. If we assume that $f \neq 0$, then the assumptions show that $\mathrm{Ker}(f)$ is stable by all $\rho_V(s)$. Indeed,

$$\forall x \in \mathrm{Ker}(f), \quad f(\rho_V(s)(x)) = \rho_W(s)(f(x)) = \rho_W(s)(0) = 0,$$

hence $\rho_V(x) \in \mathrm{Ker}(f)$. So since ρ_V is irreducible and $f \neq 0$, $\mathrm{Ker}(f) = \{0\}$. Similarly, we show that $\Im(f)$ is stable by all $\rho_W(s)$, and as ρ_W is irreducible and $f \neq 0$, $\Im(f) = W$. In the end, f is an isomorphism and ρ_V and ρ_W are isomorphic.
To show (ii), as we are working on $\mathbb{C}$-vector spaces, f a au minus an eigenvalue λ. By setting $f' = f - \lambda\,\mathrm{Id}$, we see that $\mathrm{Ker}(f') \neq \{0\}$. By applying the first part of the proof to f' which is still a G-morphism, we have $f' = 0$. □

Remark 76. In the case where we work in a field K not necessarily algebraically closed, we keep the fact that if $f \neq 0$, f is an isomorphism. In particular, if V is irreducible, then $\mathrm{Hom}_G(V, V) \overset{\text{def.}}{=} \mathrm{End}_G(V)$ is a not necessarily commutative field.

Remark 77. We can prove Schur's lemma by using the language of $K[G]$-modules. Indeed, to say that f is an interleaving operator means that f is a morphism of $K[G]$-modules. Now we check that in this case, $\mathrm{Ker}(f)$ and $\Im(f)$ are sub- $K[G]$-moduli respectively of V and W. The irreducibility of these two modules allows us to conclude in the same way.

Corollary 5. *We always consider two irreducible representations of G on V and W. We have $\dim_{\mathbb{C}}(\mathrm{Hom}_G(V, W)) = 1$ if V and W are isomorphic, and $\dim_{\mathbb{C}}(\mathrm{Hom}_G(V, W)) = 0$ otherwise.*

Proof. If $V = W$ or if the representations are not isomorphic, Schur's lemma gives us the result. In the case where the two representations are isomorphic (but not defined on the same space), it suffices to fix g an isomorphism between the two vector spaces. We can then consider $\rho'_W(s) = g^{-1} \circ \rho_W(s) \circ g \in \mathcal{L}(V)$. By applying Schur's lemma to ρ_V and ρ'_W, we see that any interleaving operator between these two representations is written $\lambda\,\mathrm{Id}$. So any interleaving operator between V and W is written λg, and we have $\dim_{\mathbb{C}}(\mathrm{Hom}_G(V, W)) = 1$. □

Remark 78. (**Case of commutative groups**). Once the notion of interleaving operator has been defined, a natural question is whether, for $g \in G$, $\rho(g)$ is an interleaving operator. Now $\rho(g) \in \mathrm{Hom}_G(V)$ is equivalent to

$$\forall h \in G, \quad \rho(g)\rho(h) = \rho(h)\rho(g), \quad \text{i.e.} \quad \rho(ghg^{-1}h^{-1}) = 1.$$

So if $g \in Z(G)$, the center of G, then $\rho(g) \in G$.
In particular, if G is commutative, then $Z(G) = G$, and therefore with Schur's lemma 13, for an irreducible representation ρ of G on V, like the $\rho(g)$, for $g \in G$ are interleaving operators, they are multiples of the identity, which means that ρ is a representation of degree 1. We therefore fall back into the classical theory of duality on a finite commutative group (which is reassuring), and we can fully use the theory developed in the previous chapter. This observation will be demonstrated again with the help of character theory with the corollary 10.

7.2.3 Reynolds operator

We are now going to define the operator which will allow us to average the action of G on a vector space.

Definition 69 (Reynolds operator). *Let ρ be a representation of G on V. We define the operator $R_G \in \mathcal{L}(V, V)$ by*

$$R_G \overset{\text{def.}}{=} \frac{1}{|G|} \sum_{s \in G} \rho(s) \qquad \in \mathcal{L}(V, V).$$

We call it Reynolds operator associated with ρ.

Théorem 13 (Properties of the Reynolds operator)**.** *R_G is a projector on V^G. In particular:*

(i) $V^G = \Im(R_G) = \mathrm{Ker}(R_G - \mathrm{Id})$.

(ii) $\dim_{\mathbb{C}}(V^G) = \mathrm{Tr}(R_G)$.

Proof. (i) Let $y = R_G(x) \in \Im(R_G)$, then for $s \in G$, we have

$$\rho(s)(y) = \frac{1}{|G|}\sum_{g\in G}\rho(s)\rho(g)(x) = \frac{1}{|G|}\sum_{g\in G}\rho(sg)(x) = R_G(x) = y,$$

therefore $y \in V^G$. The converse is obvious:

$$\mathrm{si} x \in V^G,\ \forall s \in G,\ \rho(s)(x) = x, \mathrm{then} x = R_G(x) \in \Im(R_G).$$

To show that R_G is a projector, we have to show that $R_G^2 = R_G$, which is obvious because $\Im(R_G)$ is stable by G. Finally, by virtue of projector theory, $\mathrm{Id} - R_G$ is also a kernel projector $\Im(R_G) = V^G$.

(ii) Property (i) shows us that R_G is a projector on V^G, so in particular $V = \mathrm{Ker}(R_G) \oplus \Im(R_G)$ and by writing the matrix of R_G in a good basis, we deduce (ii).

□

7.2.4 Averaged application

By applying Schur's lemma to the representation of morphisms, we will be able to calculate the dimension of the space of G-morphisms. To clarify the notations, we will introduce the following definition.

Definition 70 (Averaged application)**.** *Let ρ_V and ρ_W be two representations on V and W respectively. We denote by $\rho_{\mathcal{L}(V,W)}$ the representation of morphisms on $\mathcal{L}(V, W)$. For $f \in \mathcal{L}(V, W)$, we denote by $\tilde{f} \overset{\text{def.}}{=} R_G(f) \in \mathcal{L}(V, W)$, which corresponds to the application averaged:*

$$\tilde{f} \overset{\text{def.}}{=} \frac{1}{|G|}\sum_{s\in G}\rho_W(s)\circ f\circ\rho_V(s^{-1}).$$

Proposition 68 (Application to G-morphisms)**.** *We use the notations of the previous definition. We assume that the representations on V and W are irreducible. We have*

$$\dim_{\mathbb{C}}(\mathrm{Hom}_G(V, W)) = \mathrm{Tr}(R_G) = \begin{cases} 1 & \textit{if the representations are isomorphic,} \\ 0 & \textit{otherwise.} \end{cases}$$

Proof. We have seen in the corollary 5 that $\dim_{\mathbb{C}}(\mathrm{Hom}_G(V, W))$ is well worth the right-hand side of the sought equality. Moreover, the theorem 13 (ii), tells us that we have $\mathrm{Tr}(R_G) = \dim_{\mathbb{C}}(\mathrm{Hom}_G(V, W))$. □

In the end, for all $f \in \mathcal{L}(V, W)$, $\tilde{f}$ is a G-invariant map for the linear representation $\rho_{\mathcal{L}(V,W)}$, that is, a G-morphism, $\tilde{f} \in \mathrm{Hom}_G(V, W)$. We kind of have a way to "make" interleaving operators. We will see in the exercise 73 an important application of this technique.

7.3 Characters

In this part, we will define and use the main tool which is used to analyze representations, but also to find representations of certain abstract groups.

7.3.1 Definition and first properties

Definition 71 (Characters). *Let ρ be a representation of a group G on V a $\mathbb{C}$-vector space of dimension n. We associate it with its character χ_ρ defined by $\chi_\rho(s) = \mathrm{Tr}(\rho(s))$ where Tr designates the trace. It is a function of G in $\mathbb{C}$, that is, $\chi_\rho \in \mathbb{C}[G]$.*

Remark 79. We must pay attention to the fact that the characters as we have just defined them are not, in general, morphisms from G in $\mathbb{C}^*$. They are therefore not characters within the meaning of chapter 1.

Knowing the character χ of a representation ρ makes it possible to know, for all $g \in G$ and all $k \in \mathbb{N}$, the value of $\mathrm{Tr}(\rho(g)^k) = \mathrm{Tr}(\rho(g^k))$, and if we write $\{\lambda_1, \dots, \lambda_n\}$ the eigenvalues of $\rho(g)$, so this amounts to knowing, for any $k \in \mathbb{N}$, the value of $S_k = \sum_{i=1}^n \lambda_i^k$. S_k is the k^{ith} Newton sum associated with the eigenvalues of $\rho(g)$, and therefore, by virtue of Newton's relations, these sums allow (with the help of solving d'a linear triangular system) to calculate the values of the $\sigma_i(\lambda_1, \dots, \lambda_n)$, where the σ_i are the elementary symmetric polynomials. Thanks to the coefficients / roots relations, we therefore know the characteristic polynomial P_g of our endomorphism $\rho(g)$, and therefore (by looking for the roots of this polynomial all the same) the eigenvalues $\{\lambda_1, \dots, \lambda_n\}$. In conclusion, knowing the character of a representation is in fact equivalent to knowing all the eigenvalues of all the morphisms associated with the elements of G. Is this sufficient to characterize a representation (at least up to isomorphism)? It is to this question that we will try to answer. But first of all, here is a classical lemma which will be useful for studying the representation of morphisms.

Lemma 14. *Let $u \in \mathcal{L}(W)$ and $v \in \mathcal{L}(V)$ be two linear maps.*
We define $\Phi \in \mathcal{L}(\mathcal{L}(V), \mathcal{L}(W))$ by the equality $\Phi(f) = u \circ f \circ v$.
We then have $\mathrm{Tr}(\Phi) = \mathrm{Tr}(u)\,\mathrm{Tr}(v)$.

Proof. We give ourselves bases $\{e_i\}_{i\in I}$ of V and $\{f_j\}_{j\in J}$ of W, as well as the dual databases $\{e_i^*\}_{i\in I}$ and $\{f_j^*\}_{j\in J}$. We can build a base $\{F_{i,j}\}_{(i,j)\in I\times J}$ of $\mathcal{L}(V, W)$ by

$$\forall x \in V, \quad F_{i,j}(x) \overset{\text{def.}}{=} \langle e_i^*, x\rangle f_j \in W.$$

If the endomorphisms of $\mathcal{L}(V, W)$ are written in matrix form in the bases (e_i) and (f_j), then $F_{k,l} = (\delta_{ik}\delta_{jl})_{(i,j)\in I\times J}$. The element $F_{k,l}^*$ of the dual basis associates with a matrix $(a_{i,j})$ the value $a_{k,l}$. The dual base is thus defined by the property:

$$\forall f \in \mathcal{L}(V, W), \quad \langle F_{i,j}^*, f\rangle = \langle f_j^*, f(e_i)\rangle.$$

So we have

$$\begin{aligned}\mathrm{Tr}(\Phi) &\overset{\text{def.}}{=} \sum_{(i,j)\in I\times J} \langle F_{i,j}^*, \Phi(F_{i,j})\rangle = \sum_{(i,j)\in I\times J} \langle f_j^*, u \circ F_{i,j} \circ v(e_i)\rangle \\ &= \sum_{(i,j)\in I\times J} \langle f_j^*, u(\langle e_i^*, v(e_i)\rangle f_j)\rangle = \sum_{(i,j)\in I\times J} \langle f_j^*, u(f_j)\rangle\langle e_i^*, v(e_i)\rangle \\ &= \mathrm{Tr}(u)\,\mathrm{Tr}(v).\end{aligned}$$

In the following, if ρ_U is a representation on a space U, we will abbreviate the notation χ_{ρ_U} in χ_U. Let's start by giving the obvious properties of the characters.

Proposition 69 (Character properties). *We have the following properties.*

(i) $\chi_\rho(1) = n$.

(ii) $\forall s \in G, \quad \chi_\rho(s^{-1}) = \overline{\chi_\rho(s)}$.

(iii) *$\forall (s,\, t) \in G^2,\ \chi_\rho(tst^{-1}) = \chi_\rho(s)$: we say that χ_ρ is a central function (see paragraph 7.5.2) on G.*

(iv) *If ρ breaks down into a direct sum of two representations ρ_V and ρ_W, then $\chi_\rho \overset{\text{def.}}{=} \chi_{V\oplus W} = \chi_V + \chi_W$.*

(v) *If we denote by $\rho_{\mathcal{L}(V,\,W)}$ the representation of morphisms on $\mathcal{L}(V,\, W)$ of two representations ρ_V and ρ_W, then $\chi_{\mathcal{L}(V,\,W)} = \overline{\chi_{\rho_V}}\chi_{\rho_W}$.*

(vi) *If we denote by ρ^* the dual representation of a ρ representation, then $\chi_{\rho^*} = \overline{\chi_\rho}$.*

(vii) *Two isomorphic representations have the same character.*

Proof. (i) This is obvious because $\mathrm{Tr}(\mathrm{Id}_V) = \dim(V) = n$.

(ii) This comes from the fact that we can take a unit matrix for $\rho(s)$ and from the computation $\chi_\rho(s^{-1}) = \mathrm{Tr}(\rho(s)^{-1}) = \mathrm{Tr}(\rho(s)^*) = \overline{\mathrm{Tr}(\rho(s))}$.

(iii) This is because $\forall (A,\, B) \in GL_n(\mathbb{C}),\ \mathrm{Tr}(BAB^{-1}) = \mathrm{Tr}(A)$.

(iv) If we denote by $\mathcal{B}_V$ a basis of V and $\mathcal{B}_W$ a basis of W, the matrix of $\rho_{V\oplus W}(s)$ is written in the base $\mathcal{B} \overset{\text{def.}}{=} \mathcal{B}_V \cup \mathcal{B}_W$:
$$M(s) = \begin{pmatrix} M_V(s) & 0 \\ 0 & M_W(s) \end{pmatrix},$$
where $M_V(s)$ is the matrix of ρ_V (s) in the base $\mathcal{B}_V$ and $M_W(s)$ that of $\rho_W(s)$ in $\mathcal{B}_W$. From where
$$\chi_{V\oplus W}(s) = \mathrm{Tr}(M(s)) = \mathrm{Tr}(M_V(s)) + \mathrm{Tr}(M_W(s)) = \chi_V(s) + \chi_W(s).$$

(v) This comes from the lemma 14, applied to $u = \rho_W(s)$ and $v = \rho_V(s^{-1})$.

(vi) This comes from the fact that the representation V^* is isomorphic to the representation of morphisms $\mathcal{L}(V, K)$, which allows to use (v), and the fact that $\mathrm{Tr}(f^\top) = \mathrm{Tr}(f)$.

(vii) Same proof as for (iii).

Example 17 (Regular representation). We denote by ρ_r the regular representation to the left of a group G, on a space of dimension $|G| = n$. This representation corresponds to a representation by permutation of the elements of the base $\{\delta_g\}_{g\in G}$ of $\mathbb{C}[G]$. The presence of non-zero entries on the diagonal of the matrix associated with $\rho_r(g)$ corresponds to fixed points for the permutation induced by g. However, the permutation induced by an element different from the neutral element has no fixed point, since
$$\rho_r(g)(\delta_h) = \delta_h \quad \Leftrightarrow \quad gh = h \quad \Leftrightarrow \quad g = e.$$
We therefore have $\chi_r(1) = n$ and $\forall s \neq 1,\ \chi_r(s) = 0$.

7.3.2 Orthogonality relations

The characters are elements (admittedly a little particular) of the space $\mathbb{C}[G]$ of the functions of G in $\mathbb{C}$. An important idea is that we can provide the vector space $\mathbb{C}[G]$ with a Hermitian space structure, and the associated scalar product will prove to be both a computational and theoretical tool very efficient.

Definition 72 (Hermitian product)**.** *If φ and ψ are two functions of G in $\mathbb{C}$, we set*
$$\langle \varphi,\, \psi\rangle \overset{\text{def.}}{=} \frac{1}{|G|}\sum_{t\in G} \varphi(t)\overline{\psi(t)}.$$
$\langle \cdot,\, \cdot\rangle$ is a Hermitian product on the vector space $\mathbb{C}[G]$ of functions from G in $\mathbb{C}$.

We will now reinvest the properties of the Reynolds operator to demonstrate the first important result of this chapter, namely the orthogonality of irreducible characters.

Théorem 14 (Orthogonality relations)**.** *A character family of non-isomorphic two-by-two irreducible representations forms a orthonormal family of the space of functions of G in $\mathbb{C}$, which means that*
– *if χ is the character of an irreducible representation, we have $\langle\chi,\, \chi\rangle = 1$.*
– *if χ and χ' are two characters of non-isomorphic irreducible representations, we have $\langle\chi,\, \chi'\rangle = 0$.*

Proof. Let ρ_1 and ρ_2 be two representations of the considered family, respectively on vector spaces V and W. With the proposition 68, we therefore have $\mathrm{Tr}(R_G) = \delta$, where $\delta = +1$ if the two representations are isomorphic (therefore in fact equal), and 0 otherwise. Gold

$$\mathrm{Tr}(R_G) = \frac{1}{|G|}\sum_{s\in G}\mathrm{Tr}(\rho_{\mathcal{L}(V,\,W)})(s) = \frac{1}{G}\sum_{s\in G}\chi_{\rho_{\mathcal{L}(V,\,W)}}(s).$$

We have seen with the proposition 69, (v), that $\chi_{\mathcal{L}(V,\,W)}(s) = \overline{\chi_V(s)}\chi_W(s)$, so we have

$$\mathrm{Tr}(R_G) = \frac{1}{|G|}\sum_{s\in G}\overline{\chi_V(s)}\chi_W(s) \overset{\text{def.}}{=} \langle\chi_W,\, \chi_V\rangle = \delta.$$

Corollary 6. *There is a finite number of classes of irreducible representations (implied by classes for the relation "to be isomorphic").*

Proof. The characters of the non-isomorphic irreducible representations form a free family, because orthogonal, of $\mathbb{C}[G]$, which is a space of finite dimension on $\mathbb{C}$. Consequently, there is a finite number of characters, therefore a finite number of irreducible representations. Their number is bounded by $\dim(\mathbb{C}[G]) = |G|$. □

7.4 Representations and enumeration

Before going any further in the study of characters, we can draw many interesting conclusions using only the orthogonality of the characters, which we have just demonstrated. In particular, we will be able to respond to the problem of the uniqueness of the decomposition into irreducible representations.

7.4.1 Decomposition of a representation

In the remainder of the exposition, we give ourselves a family of representatives $(V_i)_{i=1}^p$ of the set of irreducible representations on G, each G-module V_i being implicitly linked to a representation $\rho_i \overset{\text{def.}}{=} \rho_{V_i}$. This means that the V_i are pairwise not G-isomorphic, and that any irreducible representation W is G-isomorphic to a unique V_i.

Definition 73 (Dual of a finite group)**.** *We denote by $\hat{G}$ the set of equivalence classes of irreducible representations on G for the isomorphism relation. By abuse of language, we will often write $\hat{G}$ the set $(\rho_i)_{i=1}^p$, which corresponds to choosing a representative in each class. Likewise, we will often write $\sum_{\rho\in\hat{G}}$ instead of $\sum_{i=1}^p$.*

Proposition 70 (Uniqueness of the decomposition)**.** *Let be a representation on V, of character χ_V. Then it decomposes (i.e. is G-isomorphic) into*

$$V \simeq \bigoplus_{i=1}^{p} V_i^{\oplus a_i}, \qquad \textit{with} \quad a_i = \langle\chi_V,\, \chi_i\rangle \tag{7.5}$$

In this relation, we have noted $V_i^{\oplus a_i} \overset{\text{def.}}{=} V_i \oplus \cdots \oplus V_i$ (a_i times).
Moreover, we have $\langle\chi_V,\, \chi_V\rangle = \sum_{i=1}^p a_i^2$.

Proof. We know, from the proposition 67 that the representation over V decomposes into a sum of q irreducible representations $(W_j)_{j=1}^q$:

$$V = W_1 \oplus \cdots \oplus W_q, \tag{7.6}$$

each space W_i being associated with a morphism $\rho_{W_i} : G \to GL(W_i)$. So according to the proposition 69, (iv), we have $\chi_V = \chi_{W_1} + \cdots + \chi_{W_q}$. Since we have $\langle \chi_V,\, \chi_i \rangle = \sum_{j=1}^q \langle \chi_{W_j},\, \chi_i \rangle$, and $\langle \chi_{W_i},\, \chi_i \rangle$ is equal to 1 if W_i is G-isomorphic to W_i, and 0 otherwise, we deduce that $\langle \chi_V,\, \chi_i \rangle$ represents the number of W_j, for $j = 1, \ldots, q$, which are isomorphic to V_i. However, by definition, it is a_i.
Finally, in the (7.6) writing, we can group the W_j isomorphic to V_i, and therefore write $V_i^{\oplus a_i}$ instead. □

Remark 80. It is in this sense that the decomposition of a representation V is unique. Moreover, if we consider an irreducible representation W, it is isomorphic to a certain V_i, and the number of times W occurs in the decomposition (i.e. the number of W_i isomorphic to W) is independent of the decomposition and is worth $\langle \chi_W,\, \chi_V \rangle \overset{\text{def.}}{=} a_i$. In particular, if we have two decompositions $W = W_1 \oplus \cdots \oplus W_r$ and $W = W_1' \oplus \cdots \oplus W_{r'}'$, then $r = r'$, and even if it means reordering the indices, there are isomorphisms $W_i \simeq W_i'$.

Corollary 7. *Two representations are isomorphic if and only if they have the same character. Moreover, a representation on V of character χ_V is irreducible if and only if $\langle \chi_V,\, \chi_V \rangle = 1$.*

Proof. The character entirely determines the decomposition (7.5) according to the elements of $\hat{G}$, therefore determines the isomorphism class.
Moreover, a character is irreducible if and only if its decomposition has only one term, i.e. if there is a $j \in \{1, \ldots, p\}$ such that $a_j = 1$ and if $i \neq j$, then $a_i = 0$. This is equivalent to $\sum_{i=1}^p a_i^2 = 1$. □

7.4.2 Enumeration results

The central point to demonstrate the relations linking the degrees of irreducible representations is the use of the regular representation, since we will see that it contains all the other representations, and that we can even explain its decomposition:

Proposition 71 (Decomposition of regular representation)**.** *We denote by ρ_r the regular representation of a group G, on a vector space V of dimension n. Let χ_r be its character (cf. example 17). We take the notations of Paragraph 7.4.1. The decomposition of the regular representation over V (i.e. in terms of G-isomorphism) is written*

$$V \simeq \bigoplus_{i=1}^p V_i^{\oplus n_i} \qquad \textit{with} \qquad n_i \overset{\text{def.}}{=} \dim_{\mathbb{C}}(V_i) = \langle \chi_r,\, \chi_i \rangle.$$

Proof. According to the proposition 70, the number of times that V_i occurs in the regular representation is

$$a_i \overset{\text{def.}}{=} \langle \chi_r,\, \chi_i \rangle = \frac{1}{|G|} \sum_{s \in G} \chi_r(s^{-1}) \chi_i(s) = \frac{1}{|G|} \chi_r(1) \chi_i(1) = \chi_i(1) = n_i.$$

Corollary 8. *We have the relations:*

(i) $\sum_{i=1}^p n_i^2 = |G|$.

(ii) *For $s \neq 1$, $\sum_{i=1}^p n_i \chi_i(s) = 0$.*

Proof. According to the proposition 71, we have $\chi_r(s) = \sum n_i \chi_i(s)$. We deduce (i) by taking $s = 1$ and (ii) by taking $s \neq 1$. □

Remark 81. Relation (i) makes it possible to determine whether or not all the representations of a given group have been found, and, if applicable, to determine the size of a possible representation missing from the call. In this case, we can use relation (ii) to determine the value of this character (see the example of the group $\mathfrak{S}_4$, paragraph 8.1.4, for an application) .

7.5 Fourier Theory

Before trying to develop a theory of Fourier series similar to that of characters over a commutative group (this will be done in Paragraph 7.5.3), let us start by defining the Fourier transform morphism. This construction is identical to that made in the context of abelian groups in Paragraph 1.4.1, since it consists in extending a representation over a group G to the space of functions of G in $\mathbb{C}$. Indeed, we have already said that a representation of G uniquely extends into an algebra representation over $\mathbb{C}[G]$. Quite naturally, we are in fact building the Fourier transform, and we will see that, as in the case of integrable square functions over a real interval, this Fourier transform is an algebra morphism.

7.5.1 Fourier transform

Definition 74 (Fourier transform)**.** *Let $f \in \mathbb{C}[G]$ be a function of G in $\mathbb{C}$. We define, for ρ a representation of G on a space V, the application $\pi(f)(\rho)$, by*

$$\pi(f)(\rho) = \sum_{s \in G} f(s)\rho(s) \in \mathcal{L}(V, V).$$

This allows to define the transformed Fourier map:

$$\mathcal{F} : \begin{cases} \mathbb{C}[G] & \longrightarrow \quad \bigoplus_{i=1}^{p} \operatorname{End}(V_i) \\ f & \longmapsto \quad \{\pi(f)(\rho_i)\}_{i=1}^{p} \end{cases}, \tag{7.7}$$

where we denote by $\operatorname{End}(V_i) \overset{\text{def.}}{=} \mathcal{L}(V_i, V_i)$ the space of linear maps of V_i in V_i. By abuse of notation, we will write $\mathcal{F}(f)(\rho_i)$ instead of $\pi(f)(\rho_i) = (\mathcal{F}f)_i$ (the $i^{\text{ième}}$ component of $\rho_i(f)$).

Remark 82. (**Fourier transform and representation on $\mathbb{C}[G]$**). This definition is in fact very natural, since we were content to extend the representation ρ defined on G to a representation on the algebra $\mathbb{C}[G]$ (ie in a morphism of algebras from $\mathbb{C}[G]$ into $\operatorname{End}(V)$). Indeed, we can identify $s \in G$ with the element δ_s (this is the canonical identification), and we notice that

$$\forall s \in G, \quad \pi(\delta_s)(\rho) = \rho(s).$$

We want to extend ρ to $\tilde{\rho}$ on $\mathbb{C}[G]$. To define, for $f \in \mathbb{C}[G]$, the value of $\tilde{\rho}(f)$, it suffices to write f in the form $f = \sum_{s \in G} f(s)\delta_s$. The only way to carry out this extension is to use the linearity that the function $\tilde{\rho}$ must have, and to set

$$\tilde{\rho}(f) \overset{\text{def.}}{=} \sum_{s \in G} f(s)\rho(s).$$

As if by chance, this is the definition we took for $\pi(f)(\rho)$!From this remark, we immediately draw the following proposition.

Proposition 72 (Convolution and Fourier transform)**.** *The transformed Fourier map is a morphism of algebras from $(\mathbb{C}[G], *)$ into $\bigoplus_{i=1}^{p} (\operatorname{End}(V_i), \circ)$, in other words,*

$$\forall \rho \in \hat{G},\ \forall (f, g) \in \mathbb{C}[G]^2, \quad \mathcal{F}(f * g)(\rho) = \mathcal{F}(f)(\rho) \circ \mathcal{F}(g)(\rho).$$

Proof. Just use the fact that the application $f \in \mathbb{C}[G] \mapsto \mathcal{F}f(\rho)$ is the only que representation that extends the representation ρ to space $\mathbb{C}[G]$. Now a representation of an algebra is a morphism of algebras, hence the proposition. □

As for the Fourier transform on an abelian group, the map that we have just defined is in fact an algebra isomorphism. This is what we will demonstrate, using the regular representation once again.

Proposition 73 (Bijectivity of the Fourier transform)**.** *The transformed Fourier map is an isomorphism of algebras from $(\mathbb{C}[G], *)$ over $\bigoplus_{i=1}^{p} (\operatorname{End}(V_i), \circ)$.*

Proof. **Injectivity:** Let $f \in \mathbb{C}[G]$ such that $\forall i = 1, \dots, p,\ \mathcal{F}f(\rho_i) = 0$. Let ρ be any representation. We can decompose ρ into the sum of representations $(\rho_i)_{i=1}^p$. This means that on a good basis, the matrix of $\rho(f)$ is formed by diagonal arrays of the matrices $\mathcal{F}f(\rho_i),\ i = 1, \dots, p$, so that it is zero.
Applying this result to the regular representation ρ_r on the space $V = \mathbb{C}[G]$, we get $\rho_r(f_r) = 0$, hence

$$0 = \mathcal{F}f(\rho_r)(\delta_e) \overset{\text{def.}}{=} f * \delta_e = f.$$

Surjectivity: We have already seen in the corollary 8 that $\sum_{i=1}^p n_i^2 = |G|$. Now we know that $\dim_{\mathbb{C}}(\mathbb{C}[G])$ is equal to $|G|$ (because the $(\delta_s)_{s\in G}$ form a canonical basis of $\mathbb{C}[G]$), and that we also have $\dim_{\mathbb{C}}(\mathrm{End}(V_i)) = n_i^2$ (it is a matrix algebra). By equality of dimensions, we deduce that $\mathcal{F}$, which is injective, is bijective. □

We can even go further by explaining the inverse application, thanks to an inversion formula which generalizes the formula already proved in the case of an abelian group, to the proposition 13.

Théorem 15 (Inversion formula)**.** *For $f \in \mathbb{C}[G]$, we have the following inversion formula:*

$$\forall g \in G, \quad f(g) = \frac{1}{|G|} \sum_{\rho_i \in \hat{G}} n_i \ \mathrm{Tr}\left(\rho_i(g^{-1})\, \mathcal{F}f(\rho_i)\right),$$

where n_i is the degree of the representation ρ_i and Tr *denotes the trace.*

Proof. Using the linearity of the two members of the equality, it suffices to prove the proposition in the case where $f = \delta_h$. The right-hand side of the equality then boils down to

$$\frac{1}{|G|} \sum_{i=1}^p n_i \,\mathrm{Tr}(\rho_i(g^{-1})\rho_i(h)) = \frac{1}{|G|} \sum_{i=1}^p n_i \chi_i(g^{-1}h).$$

However, according to the proposition 8, this last quantity is worth 1 if $g^{-1}h = 1$ (that is to say $g = h$), and 0 otherwise. Looking at the right-hand side of the equality, which is worth $\delta_h(g)$, we see that this is what we had to prove. □

7.5.2 Space of central functions

As previously, we assume that we have a family of representatives $(\chi_i)_{i\in I}$ of the characters of the irreducible representations on V, that is to say of $\hat{G}$. We have seen that the characters on a group G have an important property, since they are in the center of $\mathbb{C}[G]$ for the convolution product. They share this property with a larger class of functions, which we call central functions, and which we will study in this paragraph. We will see in particular the primordial result of this chapter, which says that the characters in fact form a basis of this space, and that this basis is even orthonormal.

Let us start by recalling the definitions linked to the action of G on itself by conjugation.

Definition 75 (Conjugation classes)**.** *G acts on itself by conjugation: for an element $g \in G$, the action sends $h \in G$ to ghg^{-1}. The orbits for this action are called the conjugation classes of G. Thus, the class of an element $h \in G$ is*

$$C_h \overset{\text{def.}}{=} \left\{ghg^{-1} \ : \ g \in G\right\}.$$

Two elements are said to be conjugated if they belong to the same conjugation class.

Definition 76 (Central functions space)**.** *A function $\varphi : G \to \mathbb{C}$ is said to be central if it checks*

$$\forall (s,\, g) \in G^2, \quad \varphi(sgs^{-1}) = \varphi(g).$$

We denote by $\mathbb{C}[G]^G$ the set of central functions over G: it is a vector subspace of the space $\mathbb{C}[G]$ of functions of G in $\mathbb{C}$. Each function being constant on each conjugation class, the dimension of $\mathbb{C}[G]^G$ is equal to the number of these same classes (equal, therefore, to the number of orbits for the conjugation action of G on

G). More precisely, if we denote by $\{C_1, \ldots, C_q\}$ the different conjugation classes of G, a basis of the space $\mathbb{C}[G]^G$ is given by the functions $f_{C_1}, \ldots, f_{C_q}$, defined by

$$\forall g \in G, \quad f_{C_i}(g) = \begin{cases} 1 & si g \in C_i \\ 0 & otherwise \end{cases}. \tag{7.8}$$

In fact, the central functions form a very important subspace of the algebra $\mathbb{C}[G]$.

Proposition 74. *The central functions are the functions $f \in \mathbb{C}[G]$ which check $\forall \varphi \in \mathbb{C}[G],\ f * \varphi = \varphi * f$. In other words, $\mathbb{C}[G]^G$ is the center of $\mathbb{C}[G]$ for the convolution product $*$.*

Proof. It suffices to write the definition of the convolution product:

$$(f * \varphi)(g) \overset{\text{def.}}{=} \sum_{h \in G} f(h)\varphi(h^{-1}g) = \sum_{h' \in G} \varphi(h')f(gh'^{-1}),$$

where we made the change of variable $h' = h^{-1}g$ in the summation. We conclude by using, since f is central, the fact that $f(gh'^{-1}) = f(h'^{-1}g)$. □

Remark 83. The notation $\mathbb{C}[G]^G$ is consistent with the theory of group actions, because the central functions can be seen as the invariant elements of $\mathbb{C}[G]$ under the conjugation action by G . Indeed, G acts by conjugation on $\mathbb{C}[G]$ by

$$\forall g \in G,\ \forall f \in \mathbb{C}[G], \quad g \cdot f : x \mapsto f(gxg^{-1}).$$

The central functions therefore form the invariant under-representation of $\mathbb{C}[G]$ under this action of G on $\mathbb{C}[G]$.

Lemma 15. *If $f \in \mathbb{C}[G]^G$ is a central function of G in $\mathbb{C}$, then, for any irreducible representation ρ on a space V of dimension n, $\mathcal{F}f(\rho)$ is a homothety of ratio $\frac{|G|}{n}\langle f, \overline{\chi_\rho}\rangle$.*

Proof. Let us start by noting that $\mathcal{F}f(\rho)$ is an interleaving operator for ρ:

$$\forall s \in G, \quad \rho(s)^{-1}\mathcal{F}f(\rho)\rho(s) = \sum_{t \in G} f(t)\rho(s^{-1})\rho(t)\rho(s) = \sum_{t \in G} f(t)\rho(s^{-1}ts).$$

So, using the change of variable $u = s^{-1}ts$ and using the fact that the function f is central, it comes

$$\forall s \in G, \quad \rho(s)^{-1}\mathcal{F}f(\rho)\rho(s) = \sum_{u \in G} f(sus^{-1})\rho(u) = \sum_{u \in G} f(u)\rho(u) \overset{\text{def.}}{=} \mathcal{F}f(\rho).$$

We then apply the case (ii) of Schur's lemma 13 to see that $\mathcal{F}f(\rho)$ is a dilation of ratio λ. As its trace is worth $n\lambda$, we have

$$n\lambda = \sum_{t \in G} f(t)\operatorname{Tr}(\rho(t)) = \sum_{t \in G} f(t)\chi_\rho(t) \overset{\text{def.}}{=} |G|\langle f, \overline{\chi_\rho}\rangle.$$

Remark 84. This property, demonstrated somewhat computationally, simply reflects the fact that the morphism of algebras $\mathcal{F}$ matches the center of $\mathbb{C}[G]$ (i.e. the central functions) with the center of the algebra $\bigoplus_{i=1}^p \operatorname{End}(V_i)$ (that is to say the elements which induce homothety on each V_i). All these properties will make it possible to refine the result of orthogonality of the characters, demonstrated in the theorem 14.

Théorem 16. $(\chi_\rho)_{\rho \in \hat{G}} = (\chi_i)_{i=1}^p$ *forms an orthonormal basis of space $\mathbb{C}[G]^G$ of central functions on G.*

Proof. The (χ_i) forms an orthonormal family, therefore free, it suffices to show that it is a generator. Saying that the (χ_i) generate $\mathbb{C}[G]^G$ is equivalent (because $f \in \mathbb{C}[G]^G \Leftrightarrow \overline{f} \in \mathbb{C}[G]^G$) to say that the $\overline{(\chi_i)}$ generate $\mathbb{C}[G]^G$. In other words, if we take $f \in \mathbb{C}[G]^G$ orthogonal to $H \overset{\text{def.}}{=} \operatorname{Span}\{\overline{\chi_i}\ :\ i = 1, \ldots, p\}$, we want to show that $f = 0$. Now with the lemma 15, we know that $\mathcal{F}f(\rho_i)$ is a homothety of ratio $\langle f, \overline{\chi_i}\rangle$, therefore is zero, because f is orthogonal to H. This means that the Fourier transform of f is zero, so $f = 0$ thanks to the proposition 73. □

Corollary 9. *The number p of irreducible representations on G non-isomorphic (i.e. the cardinality of $\hat{G}$) is equal to the number of conjugation classes of G.*

Proof. As the functions of $\mathbb{C}[G]^G$ are constant functions on the conjugation classes of G, the dimension of $\mathbb{C}[G]^G$ is equal to the number of these classes . We finish by using the fact that the $(\chi_i)_{i=1}^p$ form a basis of $\mathbb{C}[G]^G$. □

Remark 85. Even if we know that the number of irreducible representations up to isomorphism is the same as the number of conjugation classes, we have, a priori, no means of relating these two types of objects. For example, given a class, we would like to have a way to construct an irreducible representation. Within the framework of the $\mathfrak{S}_n$ group, we know how to do it, but the construction is complicated (refer to the book of Fulton and Harris [34]).

Corollary 10. *G is commutative if and only if all its irreducible representations are of degree 1.*

Proof. If we denote by p the number of conjugation classes, G is commutative if and only if $p = |G|$. Now with the corollary 8, we have $\sum_{i=1}^p n_i^2 = |G|$, therefore G is commutative if and only if $\forall i = 1, \ldots, p,\ n_i = 1$. □

7.5.3 Fourier series

This paragraph is satisfied to synthesize the preceding results in the form of a decomposition formula of a central function in *Fourier series*. We find exactly the same statements as for the Fourier series on an abelian group, while restricting ourselves of course to central functions. In the next chapter, we will extend these Fourier series to any functions of $\mathbb{C}[G]$, but this will require leaving out our characters, however useful!

Definition 77 (Fourier coefficients)**.** *For $f \in \mathbb{C}[G]$ and for ρ_i an irreducible representation, we define the Fourier coefficient of f in ρ_i by*

$$c_f(\rho_i) \overset{\text{def.}}{=} \langle f,\, \chi_i \rangle \overset{\text{def.}}{=} \frac{1}{|G|} \sum_{t \in G} f(t)\overline{\chi_i(t)}, \tag{7.9}$$

where we noted χ_i the character of ρ_i.

Proposition 75 (Fourier series decomposition)**.** *Let $f \in \mathbb{C}[G]^G$ be a central function on G. We have the decomposition of f in Fourier series:*

$$f = \sum_{\rho \in \hat{G}} c_f(\rho)\chi_\rho.$$

Proof. This comes immediately from the fact that the (χ_i) form an orthonormal basis of $\mathbb{C}[G]^G$. □

Proposition 76 (Plancherel formula)**.** *Let f and g be two central functions on G. We have the identity of Plancherel:*

$$\frac{1}{|G|} \sum_{s \in G} f(s)\overline{g(s)} = \sum_{\rho \in \hat{G}} c_f(\rho)\overline{c_g(\rho)}.$$

Proof. By writing $f = \sum_{\chi \in \hat{G}} c_f(\chi)\chi$ as well as $g = \sum_{\chi \in \hat{G}} c_g(\chi)\chi$, he comes

$$\sum_{s \in G} f(s)\overline{g(s)} = |G|\langle f,\, g\rangle = |G| \sum_{(\chi_1,\chi_2) \in \hat{G}^2} c_f(\chi_1)\overline{c_g(\chi_2)}\langle \chi_1,\, \chi_2 \rangle.$$

We therefore obtain the desired equality thanks to the orthogonality relations between the characters. □

In conclusion, let us observe how this decomposition of a central function is explained in terms of base change. We have already seen that the "natural" database in which we readily represent a central function is the $\{f_{C_1}, \ldots, f_{C_p}\}$ base of "plateau" functions . The Fourier series decomposition makes it possible to go from this base, which is not very practical from the computational point of view, to the character base, which has much more interesting properties with respect to convolution. It is precisely this use of characters that will be discussed in the following paragraph.

7.5.4 Fourier transform and characters

In this section, we are now interested in the properties of characters as central elements of the $\mathbb{C}[G]$ algebra. But to study the *projectors* of such an algebra, we have to use certain concepts of a more general scope.

Definition 78 (Central idempotents)**.** *Let $\mathcal{A}$ be an associative algebra of finite dimension over the field $\mathbb{C}$ of the complexes. An element $e \in \mathcal{A}$ is called idempotent central if it matches $e^2 = e$ and $\forall x \in \mathcal{A}$, $e * x = x * e$. A family $\{e_\lambda\}_{\lambda \in L}$ (where L is a finite set) is a system of orthogonal idempotents if it satisfies*

$$\sum_{\lambda \in L} e_\lambda = 1_{\mathcal{A}} \qquad \forall (\lambda,\, \mu) \in L^2, \quad e_\lambda * e_\mu = \begin{cases} e_\lambda & si \lambda = \mu \\ 0 & otherwise \end{cases}.$$

Suppose we have an algebra isomorphism

$$\Phi : \mathcal{A} \xrightarrow{\simeq} \bigoplus_{\lambda \in L} \mathrm{End}(V^\lambda) = \mathcal{B},$$

where the V^λ are finite dimensional vector spaces. So let's note

$$E_\lambda \overset{\text{def.}}{=} 0 \oplus \cdots \oplus 0 \oplus \mathrm{Id}_{V^\lambda} \oplus 0 \oplus \cdots \oplus 0 \in \mathcal{B}.$$

We can easily see that the family $\{E_\lambda\}_{\lambda \in L}$ forms a minimal system of orthogonal idempotents, since we have a complete description of the center of $\mathcal{B}$ (the morphisms which are homothety on each V^λ). Consequently, thanks to Φ, it is very simple to determine a system of idempotents on $\mathcal{A}$, it suffices to consider the $\{e_\lambda\}_{\lambda \in L}$ with $e_\lambda \overset{\text{def.}}{=} \Phi^{-1}(E_\lambda)$.

If we take the case where $\mathcal{A} = \mathbb{C}[G]$, the Fourier transform $\mathcal{F}$ defined by the equation (7.7) will therefore allow us to build our system formed by the $\{e_\lambda\}_{\lambda \in \hat{G}}$. Here, the set L has been taken equal to $\hat{G}$, since to each irreducible representation class λ corresponds an idempotent e_λ. The important point is that we can explicitly calculate these idempotents, using the Fourier inversion formula, proposition 15:

$$\forall \lambda \in \hat{G},\ \forall g \in G, \quad e_\lambda(g) = \mathcal{F}^{-1} E_\lambda(g) = \frac{n_\lambda}{|G|} \chi_\lambda(g^{-1}),$$

where we denote by n_λ the dimension of the representation λ. Recall the two essential properties of our e_λ:

$$\sum_{\lambda \in \hat{G}} e_\lambda = \delta_1 \quad \text{and} \quad \forall (\lambda,\, \mu) \in \hat{G}^2, \quad e_\lambda * e_\mu = \begin{cases} e_\lambda & \text{si} \lambda = \mu \\ 0 & \text{otherwise} \end{cases}.$$

These orthogonal idempotents allow in particular to compute projections on sub-representations. Let ρ_U be a representation of G on a space U. We extend this representation in a natural way into an algebra representation, which we further denote by ρ_U. Using the language of the Fourier transform, we can even write, for $f \in \mathbb{C}[G]$, that $\rho_U(f) = \mathcal{F}(f)(\rho_U)$. For any $\lambda \in \hat{G}$, we then define an endomorphism of U, denoted P_λ as follows:

$$P_\lambda \overset{\text{def.}}{=} \rho_U(e_\lambda) = \mathcal{F}(e_\lambda)(\rho_U).$$

Moreover, we know, with the proposition 70, that the space U decomposes into a direct sum of irreducible representations, and even more precisely:

$$U = \bigoplus_{\lambda \in \hat{G}} V_\lambda^{\oplus a_\lambda} \quad \text{with} \quad a_\lambda = \langle \chi_\lambda,\, \chi_U \rangle \in \mathbb{N},$$

where the V_λ are the spaces associated with the irreducible representations $\lambda \in \hat{G}$.

Definition 79 (Isotypic spaces)**.** *We call $U_\lambda \overset{\text{def.}}{=} V_\lambda^{\oplus a_\lambda}$ the isotypic component of U associated with the irreducible representation λ.*

We can now state the important theorem of this paragraph.

Proposition 77. *P_λ is the projector on U_λ associated with the decomposition $U = \bigoplus U_\lambda$.*

Proof. Let V be an irreducible sub-representation. The construction of e_λ my be that $\mathcal{F}(e_\lambda)(\rho_U)$, restricted to V_μ is the identity if $V \simeq V_\lambda$, and is null otherwise. This means that P_λ is the searched projector. □

7.6 Exercises

Exercise 69 (Irreducibility and indecomposability). *Let K be a field. We consider the representation*

$$\left\{ \begin{array}{ccc} K & \longrightarrow & GL_2(K) \\ x & \longmapsto & \begin{pmatrix} 1 & x \\ 0 & 1 \end{pmatrix} \end{array} \right. .$$

Show that if $K = \mathbb{C}$ or if K is a finite field, the space K^2 is indecomposable but reducible for the considered representation. Deduce that the theorem 11 is false, if the starting group is infinite, or if the body of the arrival space is not algebraically closed.

Exercise 70 (Stationary operators). *Let G be a finite group, and a linear operator $A : \mathbb{C}[G] \to \mathbb{C}[G]$. We denote, for $h \in G$, τ_h the translation operator, that is to say:*

$$\forall f \in \mathbb{C}[G], \quad \tau_h f : g \mapsto f(h^{-1}g).$$

We assume that A commutes with translations, i.e. $A\tau_h = \tau_h A$. Show that there exists $\varphi \in \mathbb{C}[G]$ such that we have

$$\forall f \in \mathbb{C}[G], \quad Af = f * \varphi,$$

where $*$ denotes the convolution product.

Exercise 71 (Irreducible representation). *Let χ_U be the character of a non-trivial 1-dimensional representation, and χ_V that of an irreducible representation. Show that $\chi_U\chi_V$ is the character of an irreducible representation other than ρ_U and ρ_V.*

Exercise 72 (Representation of a product group). *Let G and H be two finite groups. Give representatives of the irreducible representations of the group $G \times H$ as a function of the representations of G and H. What are the corresponding characters?*

Exercise 73 (Action on polynomials). *Let G be a finite subgroup of $GL_n(K)$, where K denotes a feature field 0. In paragraph 7.1.2, we have defined a representation of G on the vector space of polynomials in n indeterminate, $K[X_1, \ldots, X_n]$. Recall that we denote by $K[X_1, \ldots, X_n]^G$ the subspace of invariant polynomials under this action. It is also a sub-ring. We want to show that this sub-ring is generated by a finite number of polynomials.*
In the following, we will need the following notations:

$$\forall \alpha = (\alpha_1, \ldots, \alpha_n) \in (\mathbb{N}^+)^n, \quad X^\alpha \overset{\text{def.}}{=} X_1^{\alpha_1} \cdots X_n^{\alpha_n}.$$

We then denote by $|\alpha| = |\alpha_1| + \cdots + |\alpha_n|$ the degree of the obtained monomial. For convenience, we also note

$$(A \cdot X)^\alpha \overset{\text{def.}}{=} (A \cdot X)_1^{\alpha_1} \cdots (A \cdot X)_n^{\alpha_n}, with (A \cdot X)_i \overset{\text{def.}}{=} a_{i1}X_1 + \cdots + a_{in}X_n.$$

The goal of this exercise is to find a set of polynomials $\{P_1, \ldots, P_s\}$ generator of the ring $K[X_1, \ldots, X_n]^G$. This means that

$$\forall P \in K[X_1, \ldots, X_n]^G, \ \exists Q \in K[Y_1, \ldots, Y_s], \quad P = Q(P_1, \ldots, P_s). \tag{7.10}$$

This theorem is often referred to as Noether's theorem.

1. *In the case where the group G is the symmetric group $\mathfrak{S}_n$, we make G act on the space $K[X_1, \dots, X_n]$ by permutation of the indeterminates. Explain why this action is part of this exercise. Then give generators of the ring of invariants.*

2. *We consider the following subgroups of $GL_2(\mathbb{C})$:*

$$V_4 \overset{\text{def.}}{=} \left\{ \begin{pmatrix} \pm 1 & 0 \\ 0 & \pm 1 \end{pmatrix} \right\} \quad \textit{and} \quad C_2 \overset{\text{def.}}{=} \{\mathrm{Id}, -\mathrm{Id}\}.$$

For each of them, determine the ring of invariants, and give a minimal set of generators. Is the decomposition of a polynomial of $K[X_1, X_2]$ according to these generators unique?

3. *Recall that the Reynolds operator for the action of G on $K[X_1, \dots, X_n]$ is defined by*

$$\forall f \in K[X_1, \dots, X_n], \quad R_G(f)(X) \overset{\text{def.}}{=} \frac{1}{|G|} \sum_{A \in G} f(A \cdot X),$$

where we symbolically denote by $A \cdot X$ the action of A on the undetermined $X_1, \dots, X_n$. We want to show the following result:

$$K[X_1, \dots, X_n]^G \overset{\text{def.}}{=} K\left[R_G(X^\beta)\ ;\ |\beta| \leq |G|\right].$$

Explain why it suffices to show that for any exponent α, $R_G(X^\alpha)$ is expressed as a polynomial in the $R_G(X^\beta)$, $|\beta| \leq |G|$.

4. *We denote*

$$(X_1 + \cdots + X_n)^k = \sum_{|\alpha|=k} a_\alpha X^\alpha,$$

where the a_α are positive integers. Let then $u_1, \dots, u_n$ be undetermined news. We notice

$$\forall A \in G, \quad U_A \overset{\text{def.}}{=} u_1 A_1 \cdot X + \cdots + u_n A_n \cdot X.$$

Show that we have

$$S_k(U_A\ ;\ A \in G) \overset{\text{def.}}{=} \sum_{A \in G} (U_A)^k = \sum_{|\alpha|=k} |G| a_\alpha R_G(X^\alpha) u^\alpha.$$

We recall that any symmetric polynomial of $K[Y_1, \dots, Y_p]$ is written as a function of the p first sums of Newton S_k defined by

$$\forall k \in \{1, \dots, p\}, \quad S_k(Y_1, \dots, Y_p) = \sum_{i=1}^{p} Y_i^k.$$

Using this property for Newton's sums $S_k(U_A \quad A \in G)$, show that there exists a polynomial F with coefficients in K such that

$$\sum_{|\alpha|=k} a_\alpha R_G(X^\alpha) u_\alpha = F\left(\sum_{|\beta|=1} |G| a_\alpha R_G(X^\beta) u^\beta, \dots, \sum_{|\beta|=|G|} |G| a_\alpha R_G(X^\beta) u^\beta \right).$$

Deduce the desired result.

5. *We consider the group*

$$C_4 \overset{\text{def.}}{=} \{\mathrm{Id}, A, A^2, A^3\}, \quad \textit{where} \quad A \overset{\text{def.}}{=} \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}.$$

Use the Reynolds operator to determine the ring of invariants.

The exercise 88, proposes to use Maple to calculate the ring of invariants by the method exposed here.

Exercise 74 (Molien's theorem)**.** *We consider G a finite subgroup of $GL_n(\mathbb{C})$, and we wish to study the action of G on polynomials, as defined in Paragraph 7.1.2. More precisely, to obtain a finite dimensional representation, we consider the restriction of this action to the space $V_s \overset{\text{def.}}{=} K_s^0[X_1, \dots, X_n]$ of homogeneous polynomials of degree s (in which of course includes the zero polynomial). We denote by d_s the dimension of V_s. We denote by $\rho_s : G \to GL(V_s)$ the action thus defined. We recall that a basis of V_s is given by the set of monomials of degree s. For $i \geq 0$, we denote by a_i the maximum number of polynomials of V_s which are homogeneous, invariant, and linearly independent. To study these numbers a_i, we introduce the formal Molien series:*

$$\Phi(\lambda) \overset{\text{def.}}{=} \sum_{n=0}^{\infty} a_i \lambda^i.$$

We want to show Molien's theorem, which states that:

$$\Phi(\lambda) = \sum_{A \in G} \frac{1}{\det(\mathrm{Id} - \lambda A)} \tag{7.11}$$

1. *Show that if a polynomial $P \in V_s$ is invariant under the action of G, then each of its homogeneous components is invariant under G. Explain why $a_s = \dim(V_s^G)$, where we recall that V_s^G denotes the vector subspace of the invariants.*

2. *For $A \in G$, we denote by $A^{[s]}$ the matrix of $\rho_s(A)$ in the base of V_s made up of homogeneous monomials of degree s. The elements of this database will be indexed in the lexicographic order $X_1 < \cdots < X_n$. For example, for $n = 2$:*

$$si \quad A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \quad then \quad A^{[2]} = \begin{pmatrix} a^2 & ac & c^2 \\ 2ab & ad+bc & 2cd \\ b^2 & bd & d^2 \end{pmatrix}.$$

   To show that

$$a_s = \frac{1}{|G|} \sum_{A \in G} \mathrm{Tr}\left(A^{[s]}\right).$$

3. *We denote by $\omega_1, \dots, \omega_n$ the eigenvalues of $A \in G$. What are the eigenvalues of $A^{[s]}$? Deduce that the coefficient in λ^s in $\det(\mathrm{Id} - \lambda A)^{-1}$ is equal to the trace of $A^{[s]}$. Deduce the expression (7.11).*

4. *In the case of the groups G_1 and G_2 encountered in exercise 88, what is the expression of $\Phi(\lambda)$? How does this result simplify the search for generators (in the sense of (7.10)) of $K[X_1, \dots, X_n]^G$?*

Exercise 75 (Cauchy-Frobenius lemma)**.** *Let G be a finite group acting on a finite set X. For $g \in G$, we denote by X_g the set of fixed points of g, that is to say:*

$$X_g \overset{\text{def.}}{=} \{x \in X \ : \ g \cdot x = x\}.$$

1. *We denote by χ_1 the character of the trivial representation over G. Let V be a vector space of dimension $|X|$ whose base is $\{e_x\}_{x \in X}$. This allows to define a representation by permutation $\pi : G \to GL(V)$ by the relations*

$$\forall g \in G,\ \forall x \in X, \quad \pi(g)(e_x) \overset{\text{def.}}{=} e_{g \cdot x}.$$

   Calculate $\langle \chi_1, \chi_\pi \rangle$ using $|X_g|$.

2. *Prove the Cauchy-Frobenius lemma (sometimes attributed to Burnside), namely that $\langle \pi, \chi_1 \rangle$ is equal to the number of orbits of X under the action of G.*

3. *From two types of gemstones, how many different necklaces of 6 stones can a jeweler build? We can use an action of the dihedral group D_6 on the set $X = \{0, 1\}^6$.*

4. *We assume that $|X| \geq 2$ and that the action of G on X is doubly transitive. We recall that $\mathrm{Hom}_G(V)$ denotes the space of interleaving operators, i.e. the $f \in \mathcal{L}(V, V)$ such that $f \circ \pi(g) = \pi(g) \circ f$. Show that $\dim(\mathrm{Hom}_G(V)) = 2$. Deduce the decomposition of G-modules $V = \mathbb{C}1 \oplus W$, where W is irreducible. Conclude that the standard representation of $\mathfrak{S}_n$ is irreducible for $n \geq 2$.*

Exercise 76 (Representation and number theory)**.** *Let G be a finite group. We want to show that the dimensions of the irreducible representations of G are divisors of G. This exercise requires some knowledge of algebraic integers, which can be found at the beginning of Samuel*[58].

1. *Let $\rho : G \to V$ be an irreducible representation, and K a conjugation class of G. We define*
$$f \overset{\text{def.}}{=} \sum_{g \in K} \rho(g) \quad \in GL(V).$$
Show that f is a scaling of ratio $r(\rho, K)$. Determine $r(\rho, K)$ as a function of $\chi_\rho(K)$ (the value of χ_ρ over K), and d_ρ, the dimension of V.

2. *Prove the relation*
$$\frac{|G|}{d_\rho} = \sum_K r(\rho, K)\chi(K^{-1}),$$
the sum relating to all the conjugation classes of G.

3. *Let K, K' and K'' be three conjugation classes of G. We define, for $x \in K''$,*
$$a(K, K', x) \overset{\text{def.}}{=} \mathrm{Card}\left\{(k_1, k_2) \in K \times K' \ : \ x = k_1 k_2\right\}.$$
Show that $a(K, K', x)$ takes a constant value for $x \in K''$. We denote this value by $a(K, K', K'')$.

4. *Show relation*
$$r(\rho, K)\, r(\rho, K') = \sum_{K''} a(K, K', K'') r(\rho, K''),$$
the sum relating to the conjugation classes of G.

5. *Deduce that $\sum_K r(\rho, K)\mathbb{Z}$ is a sub-ring of $\mathbb{C}$ of finite type over $\mathbb{Z}$. Show then that $r(\rho, K)$ are algebraic integers, then that $\frac{|G|}{d_\rho}$ is an algebraic integer. Conclude.*

Exercise 77 (Determinant of a group)**.** *The reader will be able to make the connection between this exercise and the exercise 1 which studies circulating determinants. In a way, the idea is to generalize this notion to any group. This exercise is taken from the talk of Lam*[38]*, which translates into modern terms the discovery of representation theory by Frobenius.*
Let G be a finite group, and $\rho : G \to GL_n(\mathbb{C})$ a representation. We recall that it extends in a unique way into a morphism of algebras $f \mapsto \pi f$ still noted $\rho : \mathbb{C}[G] \to M_n(\mathbb{C})$. We consider a set of indeterminates $\{X_g\}_{g \in G}$. The determinant of the group G is denoted $\Theta(G)$, and it is the determinant of the matrix A of size $|G| \times |G|$, whose entries, indexed by the elements of G, are $A_{g,h} \overset{\text{def.}}{=} X_{gh^{-1}}$. To simplify the notations, we denote
$$X \overset{\text{def.}}{=} \sum_{g \in G} X_g \delta_g,$$
which can be considered as a generic element of the algebra $\mathbb{C}[G]$. Similarly, we define the value of ρ in X:
$$\rho(X) \overset{\text{def.}}{=} \sum_{g \in G} X_g \rho(g),$$
which can be seen as a matrix with coefficients in $\mathbb{C}\left[\{X_g \ : \ g \in G\}\right]$. This allows to define the determinant of the group G in ρ:
$$\Theta_\rho(G) \overset{\text{def.}}{=} \det(\rho(X)),$$
which is therefore a polynomial with $|G|$ indeterminate.

1. *Let ρ_r be the regular representation of G. Show that $\Theta_{\rho_r}(G) = \Theta(G)$.*

2. *Let ρ_U and ρ_V be two representations of G on vector spaces U and V. We denote by $\rho_{U\oplus V}$ the sum representation. To show that*
$$\Theta_{\rho_{U\oplus V}}(G) = \Theta_{\rho_u}(G)\Theta_{\rho_V}(G).$$

3. *We consider a system of representatives of irreducible representations, $\rho_i : G \to GL_{n_i}(\mathbb{C})$, for $i = 1, \ldots, p$. Explain why the morphisms of associated algebras $\rho_i : \mathbb{C}[G] \to M_{n_i}(\mathbb{C})$ are surjective. If we note $\rho_i(X) \overset{\text{def.}}{=} \{\lambda_{jk}(X)\}$ (in matrix form), deduce that the linear forms (in each X_g, $g \in G$) $\lambda_{jk}(X)$, for $1 \leq j,\, k \leq n_i$ are independent.*

4. *Prove that the determinant of the matrices of $M_n(\mathbb{C})$, seen as a polynomial in n^2 variables, is irreducible.*

5. *Explain why we can complete the family $\{\lambda_{jk}(X)\}$ in a base of the space of linear forms by the variables $\{X_g \;:\; g \in G\}$. Deduce that $\Theta_{\rho_i}(G)$ is irreducible.*

6. *Noting that X_1 only appears on the diagonal of $\rho_i(X)$, deduce if we look at $\Theta_{\rho_i}(G)$ as a polynomial in X_1, then its degree term $X_1^{n_i-1}$ is written*
$$\sum_{g\neq 1} \chi_{\rho_i}(g) X_1^{n_i-1} X_g.$$
Deduce that the knowledge of $\Theta_{\rho_i}(G)$ determines ρ_i, then that the $\Theta_{\rho_i}(G)$, for $i = 1, \ldots, p$ are two by two non-proportional.

7. *Conclude that the decomposition of $\Theta(G)$ into irreducible factors on $\mathbb{C}$ is written*
$$\Theta(G) = \prod_{i=1}^{p} \Theta_{\rho_i}(G)^{n_i}.$$

Exercise 78 (Affine group over a finite field)**.** *This exercise introduces important concepts that we will use in the following exercise. Let p be a prime number. We consider the group of invertible affine transformations of $\mathbb{F}_p$, which are of the form*
$$\Phi_{a,b} : \left\{ \begin{array}{ccc} \mathbb{F}_p & \longrightarrow & \mathbb{F}_p \\ x & \longmapsto & ax + b \end{array} \right. ,$$
where $a \in \mathbb{F}_p^*$ and $b \in \mathbb{F}_p$. We denote this group by G_p.

1. *Show that the identification of $\Phi_{a,b}$ with the pair $(b,\, a) \in \mathbb{F}_p \times \mathbb{F}_p^*$ allows to define G_p as a semi-product direct. What is the neutral element? Give the formulas defining the product of two elements of this group as well as the inverse of an element.*

2. *We define an application*
$$\pi : \left\{ \begin{array}{ccc} G_p & \longrightarrow & \mathbb{C}[\mathbb{F}_p] \\ (b,\, a) & \longmapsto & \left(f_{(b,\,a)} : x \mapsto f(a^{-1}(xb))\right) \end{array} \right. .$$
Show that it is in fact a unitary representation (for the usual Hermitian product on $\mathbb{C}[\mathbb{F}_p]$) of the group G_p.

3. *We consider the $E \subset \mathbb{C}[\mathbb{F}_p]$ subspace:*
$$E \overset{\text{def.}}{=} \left\{ f \in \mathbb{C}[\mathbb{F}_p] \;\middle\backslash\; \sum_{x\in\mathbb{F}_p} f(x) = 0 \right\}. \tag{7.12}$$
Show that E is an invariant subspace under the action of π, and that the restriction of π to E defines an irreducible representation.

Exercise 79 (Wavelet transform on $\mathbb{F}_p$)**.** *This exercise is taken from an article by Flornes and his collaborators*[30]. *It is a matter of constructing a wavelet transform on the field $\mathbb{F}_p$, using the language of representation theory. We take the notations from the previous exercise.*
Let $\psi \in \mathbb{C}[\mathbb{F}_p]$, which we will name wavelet. We define, for $f \in \mathbb{C}[\mathbb{F}_p]$ the wavelet transform $\mathcal{W}(f) \in \mathbb{C}[G_p]$:

$$\mathcal{W}(f) : \left\{ \begin{array}{ccc} G_p & \longrightarrow & \mathbb{C} \\ (b,\, a) & \longmapsto & p\,\langle f,\, \psi_{(b,\,a)} \rangle = \sum_{x\in\mathbb{F}_p} f(x)\overline{\psi(a^{-1}(xb))} \end{array} \right. .$$

1. *Express $\mathcal{W}(f)(b,\, a)$ as a function of $\hat{f}$ (the Fourier transform of f on the additive group $\mathbb{F}_p$).*
2. *We assume that $\psi \in E$. Show then that if $f \in E$, we have the inversion formula:*
$$\forall x \in \mathbb{F}_p, \quad f(x) = \frac{1}{c_\psi} \sum_{(b,\,a)\in G_p} \mathcal{W}(f)(b,\, a)\psi_{(b,\,a)}(x),$$
where we noted $c_\psi \overset{\text{def.}}{=} p^2\langle\psi,\, \psi\rangle$. We can think of calculating the Fourier transform of the two members.
3. *Now let ψ be a wavelet such that*
$$(p-1)|\hat{\psi}(0)|^2 = \sum_{k=1}^{p-1} |\hat{\psi}(k)|^2.$$
Show that $\mathcal{W}$ is, up to a constant d_ψ, an isometry of $\mathbb{C}[\mathbb{F}_p]$ on $\mathbb{C}[G_p]$. Show that its inverse is given by the formula
$$\forall x \in \mathbb{F}_p, \quad f(x) = \frac{1}{d_\psi} \sum_{(b,\,a)\in G_p} \mathcal{W}(f)(b,\, a)\psi_{(b,\,a)}(x),$$
with $d_\psi = (p-1)|\hat{\psi}(0)|^2$.
4. *Write a wavelet transform algorithm on $\mathbb{F}_p$, as well as the inverse transform. Graphically represent the results obtained for various wavelets and test functions.*

The figure 7.1 represents some transforms. The right column represents the modulus of $\mathcal{W}(f)(b,\, a)$ (translation b on the abscissa, expansion a on the ordinate), the more black the color, the greater the coefficient.

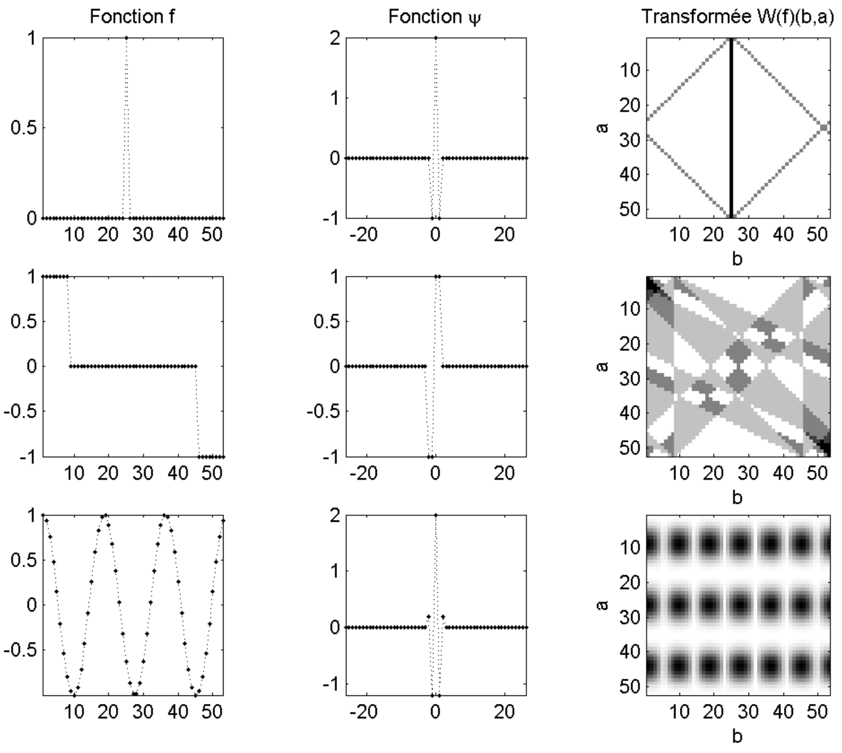

Figure 7.1: Wavelet transform on $\mathbb{F}_{53}$

Chapter 8

Applications of linear representations

Linear representations have many applications, mainly in theoretical algebra. Even in the simple framework of finite groups, this theory allows to demonstrate difficult results. Without going very far in this direction, the second paragraph shows how, starting from the knowledge of the characters of a group (that is to say of information on how our group acts on external objects), we can deduce information about the subgroups that compose it. First of all, and to provide some study material, the first paragraph studies some important finite groups. Finally, the last paragraph, which closes this book, transposes the problem of data analysis within the framework of non-commutative groups.

8.1 Representation of classic groups

The practical application of the theory developed in this chapter involves the study of elementary groups but which intervene constantly in theoretical physics or crystallography as well as in mathematics. We are therefore going to determine the list of irreducible representations of these groups, their characters, by trying to find the different geometric meanings of our groups (groups of isometries of a figure, action on faces, edges, etc.).

8.1.1 Character table

As the characters are constant on the conjugation classes $C_1, \ldots, C_p$ of G, we just need to draw up an array of the values of the characters $(\chi_i)_{i=1}^p$ on these classes. We will therefore consider the quantities $\chi_i(g_j)$, where g_j is a representative of the class C_j. In the following, we always place in first position the trivial representation, so that $\chi_1 = 1$. For convenience, we also indicate the cardinals k_j of the different classes C_j. Finally, we use the fact that $\chi_i(1) = n_i$, to establish a table, which is a square matrix of size p:

$$\begin{array}{c|cccc}
 & 1 & k_2 & \ldots & k_p \\
 & 1 & g_2 & \ldots & g_p \\
\hline
\chi_1 & 1 & 1 & \ldots & 1 \\
\chi_2 & n_2 & \chi_2(g_2) & \ldots & \chi_2(g_p) \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\chi_p & n_p & \chi_p(g_2) & \ldots & \chi_p(g_p)
\end{array}$$

We saw, in Paragraph 7.5.2, that the characters form an orthonormal basis of the space of central-functions. This results, on the character table, by orthogonality relations between the rows of the table, taking care to affect each column j by the weight k_j. In fact, we also have similar relations on the columns of the matrix as specified in the following proposition:

Proposition 78 (Orthogonality of columns)**.** *If we denote by $\chi_\lambda(C_1)$ the value of the character χ_λ on the conjugation class C_1, we have*

$$\sum_{\lambda\in\hat{G}} \chi_\lambda(C_1)\chi_\lambda(C_2) = \begin{cases} \frac{|G|}{|C_1|} & si C_1 = C_2 \\ 0 & otherwise \end{cases} .$$

Proof. Let C be a conjugation class. We recall that we denote by f_C the characteristic function of this class (cf. equation (7.8)). Let us calculate its Fourier coefficients using the defining formula (7.9):

$$\forall \lambda \in \hat{G}, \quad c_{f_C}(\lambda) = \frac{1}{|G|}\sum_{g\in G} f_C(g)\overline{\chi_\lambda(g)} = \frac{|C|}{|G|}\overline{\chi_\lambda(C)}.$$

By taking successively $C = C_1$ then $C = C_2$ in this formula, then by using the formula of *Plancherel*, equation (76), we obtain the desired result. □

8.1.2 Cyclic groups

A cyclic group being commutative, according to the corollary 10, it has only representations of dimension 1, c'that is, characters in the primary sense of the term (morphisms of G in the multiplicative group $\mathbb{C}^*$). Let $G = \{1, g_0, g_0^2, \dots, g_0^{n-1}\}$ be a finite cyclic group of cardinal n and generator g_0. Let $\omega_n = e^{\frac{2\imath\pi}{n}}$. We have already seen that all the elements of $\hat{G}$ are then of the form, for $i \in \{0, \dots, n-1\}$,

$$\chi_i : \begin{cases} G & \longrightarrow & \mathbb{C}^* \\ g = g_0^k & \longmapsto & (\omega_n^i)^k = e^{\frac{2\imath\pi i k}{n}} \end{cases} .$$

In particular, we have $G \simeq \hat{G}$. We can therefore write the table of $\widehat{\mathbb{Z}/n\mathbb{Z}}$, which is a matrix of *Vandermonde*:

	1	$k_2 = 1$	$\dots$	$k_n = 1$
	$g_1 = 0$	$g_2 = 1$	$\dots$	$g_n = n-1$
χ_1	1	1	$\dots$	1
χ_2	1	ω_n	$\dots$	ω_n^{n-1}
χ_3	1	ω_n^2	$\dots$	$\omega_n^{2(n-1)}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
χ_n	1	ω_n^{n-1}	$\dots$	$\omega_n^{(n-1)(n-1)}$

8.1.3 Dihedral groups

Definition 80 (Dihedral group)**.** *We call dihedral group D_n the group of isometries of the plane which keep a regular polygon with n sides. It contains n angle rotations $\frac{k\pi}{n}$, $k = 0, \dots, n-1$ which form a subgroup isomorphic to C_n, as well as n symmetries. If we denote by r the rotation of angle $\frac{2\pi}{n}$ and s one of the symmetries, then we have the relations*

$$r^n = 1 \qquad s^2 = 1 \qquad (sr)^2 = 1.$$

Depending on whether or not an element of D_n belongs to C_n or not, an element of D_n is uniquely written as $s^i r^k$ with $k = 0, \dots, n-1$ and $i = 0, 1$. In addition we have $x \in C_n \Leftrightarrow i = 0$.

Note first of all that we only need to give the values of the different representations and the different characters for the two generators r and s.

Case where n is even:
A representation ρ of degree one (or its character, since it is the same thing) must verify $\psi(s)^2 = 1$, c' that

is, $\psi(s) = \pm 1$. It must also check $\psi(sr)^2 = 1$, so $\psi(r) = \pm 1$ and $\psi(r)^n = 1$. Since n is even, the condition on r is written $\psi(r) = \pm 1$. In the end, we obtain the following 4 representations:

	n r^k	n sr^k
ψ_1	1	1
ψ_2	1	-1
ψ_3	$(-1)^k$	$(-1)^k$
ψ_4	$(-1)^k$	$(-1)^{k+1}$

For representations of degree two, let $\omega_n = e^{\frac{2\imath\pi}{n}}$. We will define for $h \in \mathbb{N}$ a representation on D_n by the formulas

$$\rho_h(r^k) = \begin{pmatrix} \omega_n^{hk} & 0 \\ 0 & \omega_n^{-hk} \end{pmatrix} \qquad \rho_h(s^k) = \begin{pmatrix} 0 & \omega_n^{-hk} \\ \omega_n^{hk} & 0 \end{pmatrix}.$$

We check that these formulas do indeed define a representation. Moreover, we can take $h \in \{0, \dots, n-1\}$, and the representations ρ_h and ρ_{nh} are isomorphic, since

$$\forall g \in G, \quad \rho_h(g) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \rho_{nh}(g) \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}^{-1}.$$

We therefore come to consider only the representations ρ_h for $h = 0, \dots, n/2$. The representation corresponding to the case $h = 0$ is reducible, since the lines $\mathbb{C}(e_1 + e_2)$ and $\mathbb{C}(e_1 - e_2)$ are stable. It is the same for the case $h = n/2$. We can also see that $\chi_{\rho_0} = \psi_1 + \psi_2$ and that $\chi_{\rho_{n/2}} = \psi_3 + \psi_4$, which proves that the representations ρ_0 and $\rho_{n/2}$ are reducible, and allows to know their decomposition. For the other values of h, the representation ρ_h is indeed irreducible. Indeed, if ρ_h admitted a non-trivial under-representation, it would be a straight line, and we see that a straight line stable by $\rho_h(r)$ is necessarily a coordinate axis, which n is not left stable by $\rho_h(sr)$. We can calculate the characters of these $n/2 - 1$ irreducible representations:

	r^k	sr^k
χ_h	$2\cos\left(\frac{2\pi hk}{n}\right)$	0

We can therefore see that these representations are not isomorphic (because their characters are different). To check that all the representations have been constructed in this way, it suffices to calculate the sum of the squares of the degrees of the representations. In total, we get $4 \times 1 + (n/2 - 1) \times 4 = 2n = |D_n|$.

Case where n is odd:
This time, we can only have two representations of degree one:

	n r^k	n sr^k
ψ_1	1	1
ψ_2	1	-1

We define the representations ρ_h as in the case where n is even. For $1 \leq h \leq (n-1)/2$, these representations are irreducible and two by two not isomorphic. Their characters have already been calculated in the n even case. By calculating the sum of the squares of the degrees, we get $2 \times 1 + (n-1)/2 \times 4 = 2n = |D_n|$. We have thus enumerated all the irreducible representations.

8.1.4 The group $\mathfrak{S}_4$

The first thing to do is to determine the conjugation classes of $\mathfrak{S}_4$, the group of permutations of a set with 4 elements, identified at $\{1, 2, 3, 4\}$. To do this, we use the lemma 12, and we thus obtain

– the class of the identity, which corresponds to the decomposition $4 = 1 + 1 + 1 + 1$, that is to say in four 1-rings. It has 1 element.

– the class of transpositions, for example of the element (12), which corresponds to the decomposition $4 = 2 + 1 + 1$. It has 6 elements (choice of 2 elements out of 4 without order, which makes C_4^2).
– the class of the three cycles, for example of the element (123), which corresponds to the decomposition $4 = 3 + 1$. It comprises 8 elements (4 possible choices of 3 elements among 4, and 2 possible cycles by choice).
– the class of the four cycles, for example of the element (1234), which corresponds to the decomposition $4 = 4$. It comprises 6 elements (24 permutations which are grouped together in packets of 4 identical 4-cycles).
– the class of couples of disjoint 2-cycles, for example of the element (12)(34), which corresponds to the decomposition $4 = 2 + 2$. It comprises 3 elements (6 possible choices for the first transposition, and the choice of the second divides the number of possibilities by two).

By the corollary 9, we know that $\mathfrak{S}_4$ admits, up to isomorphism, 5 irreducible representations. We have already determined a certain number of representations in Paragraph 7.1.4:
– the trivial representation, on a space U (of dimension 1), of character $\chi_1 = (1, 1, 1, 1, 1)$ (we thus note the corresponding line in the table of characters. indexes columns in the same order as that used for conjugation classes).
– the alternate representation, on a space V (of dimension 1), which corresponds to the signature and has for character $\chi_\epsilon = (1, -1, 1, -1, 1)$.
– The standard representation, on a space V_s (of dimension 3), whose character χ_s, according to the decomposition found in Paragraph 7.1.4, verifies $\chi_p = \chi_s + \chi_1$ (we noted χ_p the character of the representation by permutation of the elements of a base). However, the value $\chi_p(\sigma)$ corresponds to the number of elements left fixed by σ, which gives $\chi_p = (4, 2, 1, 0, 0)$. In the end, we therefore have $\chi_s = (3, 1, 0, -1, -1)$. We notice that we have

$$\begin{aligned}|G|\langle \chi_s,\, \chi_s\rangle = 1\chi_s(\mathrm{Id})^2 + 6\chi_s((12))^2 \\ + 8\chi_s((123))^2 + 6\chi_s((1234))^2 + 3\chi_s((12)(34))^2 = 24.\end{aligned}$$

Hence $\langle \chi_s,\, \chi_s\rangle = 1$, so according to the corollary 7, the standard representation of $\mathfrak{S}_4$ is irreducible .

For the moment, we obtain a partial character table:

	1 Id	6 (12)	8 (123)	6 (1234)	3 (12)(34)
χ_1	1	1	1	1	1
χ_ϵ	1	-1	1	-1	1
χ_s	3	1	0	-1	-1

There are still two representations to be determined, and using the relation (i) of the corollary 8, we have $n_4^2 + n_5^2 = 13$, where we have noted n_4 and n_5 the degrees of the two representations. We therefore necessarily have a representation of degree 3 and the other of degree 2. The first representation can be obtained through the representation of morphisms on $W \overset{\text{def.}}{=} \mathcal{L}(V_s, V_\epsilon)$ of representations standard and alternating. It is of degree 3, and its character is $\chi_{\mathcal{L}(W,V)} = \chi_W \overline{\chi_V} = (3, -1, 0, 1, -1)$. We notice that it is very different from the characters already determined, and that $\langle \chi_{\mathcal{L}(W,V)},\, \chi_{\mathcal{L}(W,V)}\rangle = 1$, so this representation is indeed irreducible (see the exercise 71 for generalization). To determine the last representation, on a space denoted W' (of dimension 2), we use the relation (ii) of the corollary 8, and we find $\chi_{W'} = (2, 0, -1.0.2)$. In the end, we get the character table:

	1 Id	6 (12)	8 (123)	6 (1234)	3 (12)(34)
χ_1	1	1	1	1	1
χ_ϵ	1	-1	1	-1	1
χ_s	3	1	0	-1	-1
χ_W	3	-1	0	1	-1
$\chi_{W'}$	2	0	-1	0	2

One of the concrete realizations of the group $\mathfrak{S}_4$ is the group of direct isometries keeping a cube. We can see this realization by the action of the group on the four large diagonals of the cube. Consequently, the group also acts by permuting the faces of the same cube, which gives rise to a representation by permutation of the group $\mathfrak{S}_4$, i.e. $\rho_E : \mathfrak{S}_4 \to GL(E)$, where E is a vector space of dimension 6. As for any representation by permutation, the value of $\chi_E(\sigma)$, for $\sigma \in \mathfrak{S}_4$ is equal to the number of faces fixed by the action of σ. Let's identify the different values of this character:

– a rotation of 180 ° on an axis connecting the midpoints of two opposite sides: this permutation exchanges only two diagonals. It corresponds to the class of (12). No face is attached.
– a rotation of 120 ° along a large diagonal: only the diagonal in question is invariant, the others permuting circularly. It corresponds to the class of (123). No face is attached.
– a rotation of 90 ° along a coordinate axis: permute the four diagonals in a circle. It corresponds to the class of (1234). Two faces are fixed.
– a rotation of 180 ° along a coordinate axis: swaps the diagonals two by two. It corresponds to the class of (12)(34). Two faces are fixed.

The character of our representation is therefore given by

	Id	(12)	(123)	(1234)	(12)(34)
χ_E	6	0	0	2	2

We have $\langle \chi_\rho, \chi_\rho \rangle = 3$, so our representation is written as the sum of three irreducible representations. To calculate the decomposition of this representation, it suffices to calculate the different scalar products:

$$\begin{array}{lll} \langle \chi_E, \chi_1 \rangle = 1, & \langle \chi_E, \chi_\epsilon \rangle = 0, & \langle \chi_E, \chi_s \rangle = 0, \\ \langle \chi_E, \chi_W \rangle = 1, & \langle \chi_E, \chi_{W'} \rangle = 1. & \end{array}.$$

We thus obtain the decomposition $E = \mathbb{C} \oplus W \oplus W'$, as the sum of G -modules.

8.2 The question of simplicity

In this paragraph, we will use character theory to obtain information about the structure of our group. We are going to focus on finding distinguished subgroups.

8.2.1 Character core

Let us start with a proposition, which will allow us to characterize the kernel of representations.

Proposition 79. *Let G be a finite group, and $\rho : G \to GL(V)$ a representation, of character χ_V on a space V of dimension d. We denote by $g \in G$ an element of order k. So:*

(i) *$\rho(g)$ is diagonalizable.*

(ii) *χ_V is sum of $\chi_V(1) = \dim(V) = d$ roots k^{iths} of the unit.*

(iii) *$|\chi_V(g)| \leq \chi_V(1) = d$.*

(iv) *$K_{\chi_V} \overset{\text{def.}}{=} \{x \in G : \chi_V(x) = \chi_V(1)\}$ is a distinguished subgroup of G. We call it the kernel of the representation.*

Proof. (i) As $g^k = 1$, we have $\rho(g)^k = \mathrm{Id}$. So the minimal polynomial of $\rho(g)$ divides $X^k - 1$, which is split to single root.

(ii) Let $\omega_1, \ldots, \omega_d$ be the eigenvalues of $\rho(g)$, which are $k^{\text{ìes}}$ roots of l'unit. We have $\chi_V(g) = \omega_1 + \cdots + \omega_d$.

(iii) $|\chi_V(g)| \leq |\omega_1| + \cdots + |\omega_d| = d$.

(iv) If $|\chi_V(g)| = d$, we have equality in the previous triangular inequality. This means that the complex numbers ω_i are positively related on $\mathbb{R}$. Since they are of module 1, they are all equal. If $\chi_V(g) = d$, we necessarily have $\omega_i = 1$, so $\rho(g) = \mathrm{Id}$. So $K_{\chi_V} = \mathrm{Ker}(\rho)$, is indeed a distinguished subgroup.

In the following, we will need the following lemma.

Lemma 16. *Let $N \lhd G$ be a distinguished subgroup of G. Let ρ_U be a representation of G/N on a vector space U. Then there exists a canonical representation of G on U such that the sub-representations of U under the action of G/N are exactly those of U under the action of G.*

Proof. Just ask

$$\forall g \in G, \quad \tilde{\rho}_U(g) \stackrel{\text{def.}}{=} \rho_U \circ \pi(g),$$

where $\pi : G \to G/N$ is the canonical projection. $\tilde{\rho}_U$ defines well the sought representation. □

8.2.2 Use of the character table

Let G be a finite group. We denote by $\hat{G} = \{\rho_1, \ldots, \rho_r\}$ its dual, formed by representatives of non-isomorphic irreducible representations. Here is the result which will allow us to determine the set of distinguished subgroups of a given group.

Proposition 80. *The distinguished subgroups of G are exactly of the type*

$$\bigcap_{i \in I} K_{\chi_i} \qquad \textit{where} \quad I \subset \{1, \ldots, r\}.$$

Proof. Let $N \lhd G$ be a distinguished subgroup. We denote by ρ_U the regular representation of G/N. This therefore means that U is a vector space of dimension equal to $|G/N| = |G|/|N|$, base $\{e_g\}_{g \in G/N}$, and we have $\rho_U(h)(e_g) = e_{hg}$.
We have already seen in the proposition 63, that the regular representation is faithful, therefore ρ_U is injective. Using the 16 lemma, we extend this representation to a $\tilde{\rho}_U : G \to U$ representation. Denote by χ the character of the representation $\tilde{\rho}_U$. We then have the equality $\mathrm{Ker}(\tilde{\rho}_U) = \mathrm{Ker}(\rho_U \circ \pi) = N$, hence $N = K_\chi$.
It does not all that remains is to decompose the representation $\tilde{\rho}_U$ according to the irreducible representations, to obtain

$$\chi = a_1 \chi_1 + \cdots + a_r \chi_r.$$

We therefore have, using point (iii) of the proposition 79,

$$\forall g \in G, \quad |\chi(g)| \leq \sum_{i=1}^{r} a_i |\chi_i(g)| \leq \sum_{i=1}^{r} a_i |\chi_i(1)| = \chi(1).$$

We therefore have the equality $\chi(g) = \chi(1)$ (i.e. $g \in K_\chi$) if and only if we have an equality in the previous triangular inequality . It follows that $\chi(g) = \chi(1)$ if and only if $\forall i,\ a_i \chi_i(g) = a_i \chi_i(1)$. This is ultimately equivalent to

$$\forall i, \quad a_i > 0 \Longrightarrow g \in K_{\chi_i}.$$

We therefore have the desired result, with $I \stackrel{\text{def.}}{=} \{i \ : \ a_i > 0\}$.
Finally, the converse is obvious: indeed, as the K_{χ_i} are distinguished, all subgroup of the type $\cap_{i \in I} K_{\chi_i}$ is also. □

Corollary 11. *G is simple if and only if $\forall i \neq 1$, $\forall g \in G$, $\chi_i(g) \neq \chi_i(1)$.*

Proof. If we assume that there exists $g \in G$, with $g \neq 1$, such that $\chi_i(g) = \chi_i(1)$, then $K_i \subset G$ is a subgroup distinguished non-trivial, so G is not simple.
Conversely, if G is non-simple, then there exists $g \neq 1$ in some distinguished subgroup $N \lhd G$ non-trivial . With the previous proposition, $N = \cap_{i \in I} K_i$, so $g \in K_i$ for $i \in I \subset \{2, \ldots, r\}$. This does mean that $\chi_i(g) = \chi_i(1)$. □

Thanks to the character table, we are therefore able to draw up the list of all the distinguished subgroups of a given G group, and even to determine the inclusion relations between these subgroups. For example, we can consider the group $\mathfrak{S}_4$, whose table was established in Paragraph 8.1.4. We see that it has two distinguished non-trivial subgroups: $\mathrm{Ker}(\chi_\epsilon) = \mathfrak{A}_4$ as well as $\mathrm{Ker}(\chi_{W'}) = \langle (12)(34) \rangle$ (the class of the permutation $(12)(34)$). Furthermore, we see that $\mathrm{Ker}(\chi'_W) \subset \mathrm{Ker}(\chi_\epsilon)$.

8.3 Spectral analysis

We saw in the previous chapter that the family of characters of a finite group constitutes an orthogonal basis of the space of central functions. The fundamental result of this paragraph is the generalization of this result to the space of functions of G in $\mathbb{C}$ as a whole. Of course, it will be necessary to consider another family of functions, which intervenes in a natural way when one tries to calculate in a matrix way the Fourier transform. This method for finding orthonormal bases of a functional space is the basis of spectral analysis on any finite group, which has many applications, especially in statistics.

8.3.1 Orthogonality of coordinate functions

The characters are above all theoretical objects for the search for the representations of a group G (thanks to the orthogonality relations of the rows and columns of the table of characters), and for the study of the group G lui even (study of its simplicity, resolubility, etc.). In a practical way, the fact that they form a basis only of the space of central functions makes them of little use to analyze a function from G in $\mathbb{C}$. To solve this difficulty, we will prefer to use the Fourier transform as defined in the previous paragraph. We will even see that, thanks to a certain matrix formulation, this transform also corresponds to the calculation of a decomposition in an orthogonal base.

We consider as usual a finite group G, and we denote by $\hat{G} = \{\rho_1, \ldots, \rho_p\}$ the representatives of the classes of irreducible representations. Each representation ρ_k is linked to a space V_k of dimension n_k, and these different representations are of course two by two non-isomorphic. We have seen, with the proposition 66, that we could, for each representation ρ_k, find a basis of V_k in which the matrices $R_k(g)$ of endomorphisms $\rho_k(g)$ are unitary. We will denote these matrices in the form $R_k(g) = \{r^k_{ij}(g)\}$. We thus obtain a series of applications:

$$\forall k \in \{1, \ldots, p\},\ \forall (i,\, j) \in \{1, \ldots, n_k\}^2, \quad r^k_{ij} : G \to \mathbb{C}.$$

More precisely, we thus obtain $\sum_{k=1}^{p} n_k^2 = n$ elements of $\mathbb{C}[G]$. The following proposition, which is the heart of the developments which will follow, tells us that these elements are not arbitrary.

Théorem 17 (Orthogonality of coordinate functions)**.** *The r^k_{ij} for $k \in \{1, \ldots, p\}$ and for $(i, j) \in \{1, \ldots, n_k\}^2$, form an orthogonal basis of $\mathbb{C}[G]$. More precisely, we have*

$$\forall (k,\, l) \in \{1, \ldots, p\}^2,\ \forall (a,\, b,\, c,\, d) \in \{1, \ldots, n_k\}^4, \quad \langle r^k_{ab},\, r^l_{cd} \rangle = \delta^c_a \delta^d_b \delta^l_k \frac{1}{n_k}.$$

Proof. It is in fact a question of reformulating the result of Paragraph 7.2.4. Let ρ_k and ρ_l be two irreducible representations. We know that for $f \in \mathcal{L}(U_k,\, U_l)$, the application $\tilde{f} \overset{\text{def.}}{=} R_G(f) \in \mathcal{L}(U_k,\, U_l)$ is an operator interlacing. According to Schur's lemma, it is either a homothety of ratio $\frac{\mathrm{Tr}(f)}{n_k}$ (if $k = l$), or the null morphism (if $k \neq l$).

In the bases we have chosen for U_k and U_l, the morphism f is written in matrix form $\{x_{ij}\}_{i=1\dots d_l}^{j=1\dots n_k}$. Similarly, we write the matrix of $\tilde{f}$ in the form $\{\tilde{x}_{ij}\}_{i=1\dots d_l}^{j=1\dots n_k}$. We can explicitly calculate the value of $\tilde{x}_{ij}$:

$$\tilde{x}_{i_2 i_1} \overset{\text{def.}}{=} \frac{1}{|G|} \sum_{j_1,\, j_2,\, g\in G} r^l_{i_2 j_2}(g) x_{j_2 j_1} r^k_{j_1 i_1}(g^{-1}). \tag{8.1}$$

Let's start with the case where the representations are not isomorphic, i.e. $k \neq l$. The fact that $\tilde{f} = 0$ is equivalent to $\tilde{x}_{i_2 i_1} = 0$, and this whatever the $x_{j_2 j_1}$. The expression for $\tilde{x}_{i_2 i_1}$ defines a linear form in $x_{j_2 j_1}$, which is zero. This therefore means that its coefficients are zero. By noting that $r_{i_1 j_1}(g^{-1}) = \overline{r_{j_1 i_1}(g)}$, we thus obtain, in the case where $k \neq l$,

$$\begin{aligned}&\forall (i_1,\, j_1) \in \{0, \dots,\, n_k\}^2,\ \forall (i_2,\, j_2) \in \{0, \dots,\, n_l\}^2,\\ &\frac{1}{|G|} \sum_{g\in G} \overline{r^k_{j_1 i_1}(g)} r^l_{j_2 i_2}(g) \overset{\text{def.}}{=} \langle r^k_{j_1 i_1},\, r^l_{j_2 i_2}\rangle = 0.\end{aligned}$$

There now remains the case where $k = l$. This time we have $\tilde{f} = \frac{\mathrm{Tr}(f)}{n_i}\,\mathrm{Id}$, hence

$$\forall (i_1,\, j_1) \in \{0, \dots,\, n_k\}^2,\ \forall (i_2,\, j_2) \in \{0, \dots,\, n_l\}^2 \quad \tilde{x}_{i_2 i_1} \overset{\text{def.}}{=} \frac{1}{d_i} \left(\sum_{j_1,\, j_2} \delta^{j_2}_{j_1} x_{j_2 j_1} \right) \delta^{i_2}_{i_1}.$$

By reusing the expression of $\tilde{x}_{i_2 i_1}$ obtained from the equation (8.1), and by equaling the coefficients of the linear form obtained, we have the formula

$$\frac{1}{|G|} \sum_{g\in G} \overline{r^k_{j_1 i_1}(g)} r^l_{j_2 i_2}(g) \overset{\text{def.}}{=} \langle r^k_{j_1 i_1},\, r^l_{j_2 i_2}\rangle = \frac{1}{n_i} \delta^{i_2}_{i_1} \delta^{j_2}_{j_1}.$$

Remark 86. As the characters of the irreducible representations are sums of different coordinate functions, this result simultaneously asserts the orthogonality of the characters, which we have already demonstrated in the theorem 14.

We denote by $I \overset{\text{def.}}{=} \{(k, i, j) \ :\ k = 1, \dots,\, p \text{ and } i,\, j = 1, \dots,\, n_k\}$. The result we have just demonstrated affirms the existence of an orthonormal basis of the space $\mathbb{C}[G]$, which we denote in the form $\{\Delta_{kij}\}_{(k,i,j)\in I}$. We notice that we have of course $|I| = |G|$, which is the dimension of $\mathbb{C}[G]$. These functions are defined as follows:

$$\forall (k,\, i,\, j) \in I, \quad \Delta_{(k,i,j)} \overset{\text{def.}}{=} \sqrt{n_k} r^k_{ij}.$$

8.3.2 Generalized Fourier series

The fundamental result of the previous paragraph therefore provides us with an orthogonal basis of the space of functions from G in $\mathbb{C}$. We can not help but make the comparison with the result already obtained thanks to the theory of characters to the theorem 14. However, it is important to understand that these two constructions have absolutely nothing to do. Characters are canonically defined. They do not depend on the choice of any matrix writing of our representations. It is above all a theoretical tool for obtaining information on representations (for example knowing if a representation is irreducible) or on the group itself(to determine the distinguished sub-groups). On the other hand, one can construct a quantity of orthonormal bases of $\mathbb{C}[G]$ thanks to the coordinate functions. It suffices to apply to the matrices of the different unit representations a change of unit basis. It is therefore a calculating tool. The only case where these two constructions coincide is that of commutative finite groups. Indeed, the irreducible representations of such a group are of dimension 1, and the unique entry of the corresponding matrices is equal (except for a constant) to the characters of the representation. We see moreover that in this particular case, the construction of the coordinated functions, not canonical in the general case, becomes canonical.

We now want to apply the construction we just performed to the analysis of a function $f \in \mathbb{C}[G]$. We therefore suppose that we have an orthonormal basis $\{\Delta_{kij}\}_{(k,i,j)\in I}$. The Fourier coefficients are then defined with respect to this base.

Definition 81 (Fourier coefficients)**.** *For $f \in \mathbb{C}[G]$, we call Fourier coefficients with respect to the base $\{\Delta_{kij}\}_{(k,i,j)\in I}$, and we denote by $c_f(k,\, i,\, j)$ the quantities*

$$\forall (k,\, i,\, j) \in I, \quad c_f(k,\, i,\, j) \overset{\text{def.}}{=} \langle f,\, \Delta_{kij}\rangle.$$

We therefore have the following Fourier development:

$$f = \sum_{(k,i,j)\in I} c_f(k,\, i,\, j)\Delta_{kij}.$$

We can then ask ourselves what link there is between the Fourier coefficients that we have just introduced, and the Fourier transform defined in 7.9. The calculation of the Fourier transform of a function $f \in \mathbb{C}[G]$ is equivalent to the calculation, for any irreducible representation ρ_k, of each coefficient of $\mathcal{F}(f)(\rho_k)$, that is to say

$$\forall (k,\, i\, j) \in I, \quad \mathcal{F}(f)(\rho_k)_{ij} = \sum_{g\in G} f(s)\left(\rho_k(g)\right)_{ij} = c_h(k,\, i,\, j), \tag{8.2}$$

where we noted $h \overset{\text{def.}}{=} \frac{1}{\sqrt{n_k}}\overline{f}$. It can therefore be seen that the calculation of the Fourier coefficients is totally equivalent to that of the calculation of the Fourier transform. By continuing to exploit the analogies between these two concepts, we can also say that the calculation of the transform is similar to a calculation of change of bases. We notice indeed that on condition of replacing f by its conjugate, then to normalize the result (by multiplying it by $\sqrt{n_i}$), the calculation of the Fourier transform (in matrix form) in fact amounts to going from the canonical basis of δ_g to the orthonormal basis of Δ_{kij}.

One of the questions is to know if we have, like the FFT algorithm on abelian groups, a fast calculation algorithm for the Fourier transform on a noncommutative group. We can indeed note that a naive implementation of the equations (8.2) requires $O|G|^2)$ operations. The review article by Rockmore [48] explains that there are such algorithms for large classes of groups, including the symmetric groups discussed in the next paragraph.

8.3.3 The signal representation

The fundamental problem of signal processing is that of the representation (in the first sense of the term) of the data studied. The language of linear algebra allows this problem to be formalized in a concise and elegant way. The signals that we want to analyze can in fact be seen as functions $f : D \to \mathbb{C}$ where D is any a priori domain (for example a square in the case of an image). In the context of computer processing, we have to consider finite D domains. The problem of the representation of a finite signal can then be summed up in the search for a "good" basis of the finite dimensional vector space formed by the functions of D in $\mathbb{C}$. From a practical point of view, the quality of our database will be measured by its ability to simplify our way of understanding the data to be analyzed. In particular, the representation of the data in the new base will have to be simpler, more *hollow* than in the original base.

The first important property that we want for the base sought is to be orthonormal. This makes it possible to have simple analysis and reconstruction formulas, and more robust from a numerical point of view. This is exactly what we did during the various calculations of Fourier transforms already encountered. Secondly, the search for a good basis requires exploiting the symmetries of the domain D. Even if this point may seem unrelated to the efficiency of the basis (a priori, there is no reason for the studied signals to follow the symmetries of the domain), the exploitation of the symmetries is essential to obtain algorithms of quick calculations. For example, if the FFT algorithm is so fast, it is because it fully exploits the symmetry (periodicity) of the set $\mathbb{Z}/n\mathbb{Z}$, which makes it possible to avoid any superfluous calculation as much as possible. . In practice, this property of respect for symmetry is in fact also important for the representation of functions, because most of the "natural" signals respect the regularities of the original domain. The most striking example is the study of stationary musical signals by Fourier series decomposition. We observe, after a few fundamental harmonics, coefficients which decrease very rapidly: the frequency representation of such a signal is much more compact than its temporal representation.

To try to exploit the ideas developed in the previous paragraph, it seems natural to want to provide D with a finite group structure. This usually leaves a great deal of latitude for the choice of an orthonormal basis. On the one hand, there is a multitude of structures, which may be non-isomorphic, and even if two structures are isomorphic, one may be better suited than the other to the signal being studied. On the other hand, we have already explained that the choice of different bases for the computation of the matrices of the irreducible representations gave rise to different orthonormal bases. Thus, the exercise 86 proposes to use representation theory to find an orthonormal basis of the space of functions of $\{0, 1\}^n$ in $\mathbb{C}$. This echoes the exercises 59 and 60 which use the Walsh basis (ie abelian characters) to study Boolean functions. We will now see on a concrete example how to make these choices of structures and bases, and use them to analyze a set of data.

The example we will mention now is taken from the book by Diaconis [25], who was the first to apply representation theory to statistics. For a complete panorama of fast computation algorithms in representation theory, we can refer to the article by Rockmore [48]. We consider the result of a survey where we asked a significant number of people to rank in order of preference the following three places of residence: *city* (proposition 1), *suburb* (proposition 2), *countryside* (proposition 3). Each person answers the survey by giving a permutation of the three propositions. For example, the permutation (2, 3, 1) corresponds to the classification suburb, then countryside, then city. Here are the results of the survey:

city	suburb	countryside	result
1	2	3	242
2	1	3	170
3	2	1	359
1	3	2	28
2	3	1	12
3	1	2	628

The set of results can thus be seen as a function $f : \mathfrak{S}_3 \to \mathbb{N}$, which at each permutation of (1, 2, 3) assigns the number of people having given this permutation as an answer. The problem which arises now is that of the analysis of these results. The permutation with the highest result (in this case (3, 1, 2)) gives us some information about the preferences of the respondents. But to analyze the interactions between the different permutations, a more detailed analysis must be used.

We are therefore going to perform a change of base, and calculate the way in which f decomposes in an orthogonal base obtained thanks to the irreducible representations of the group $\mathfrak{S}_3$. Besides the ρ_1, trivial, and ρ_2, alternating representations, there is an irreducible representation of dimension 2, the standard representation ρ_3. The exercise 82 proposes a geometric method to find the associated orthogonal matrices. Here we offer another basic choice. In this case, if we denote by $\{e_1, e_2, e_3\}$ the canonical basis of $\mathbb{C}^3$, we choose $\{(e_1 - e_2)/\sqrt{2}, (e_1 + e_2 - 2e_3)/\sqrt{6}\}$ for the orthonormal basis of the orthogonal of $e_1 + e_2 + e_3$. The matrices of the representation ρ_3 are written in this base:

$$\rho_3((1, 2, 3)) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad \rho_3((2, 1, 3)) = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix},$$
$$\rho_3((3, 2, 1)) = \frac{1}{2}\begin{pmatrix} 1 & -\sqrt{3} \\ -\sqrt{3} & -1 \end{pmatrix}, \qquad \rho_3((1, 3, 2)) = \frac{1}{2}\begin{pmatrix} 1 & \sqrt{3} \\ \sqrt{3} & -1 \end{pmatrix},$$
$$\rho_3((2, 3, 1)) = \frac{1}{2}\begin{pmatrix} -1 & \sqrt{3} \\ -\sqrt{3} & -1 \end{pmatrix}, \qquad \rho_3((3, 1, 2)) = \frac{1}{2}\begin{pmatrix} -1 & -\sqrt{3} \\ \sqrt{3} & -1 \end{pmatrix}.$$

By calculating the dot products between the function f and the 6 coordinate functions of these representations, we can decompose the functions f as follows:

$$f = \frac{1}{6}\left(1439\rho_1 + 325\rho_2 - 109\rho_{311} - 1640.2\rho_{312} + 493.6\rho_{321} - 203\rho_{322}\right),$$

where we have denoted ρ_{3ij} the coordinate function (i, j) of the matrix representation ρ_3. The first and most important coefficient corresponds to the mean value of the function. It is therefore not very informative. On the other hand, we note that the coefficient of ρ_{312} is clearly greater than all the others. It corresponds to the component of f on the function

$$\rho_{321} = (0,\, 0,\, -0.87,\, 0.87,\, 0.87,\, -0.87),$$

where we listed the values of ρ_{312} on the elements of $\mathfrak{S}_3$ in the same order as that of the poll results. We see that this function actually performs a grouping of the responses in 3 packets of 2 permutations (depending on whether the value is -0.87, 0 or 0.87), each packet being characterized by the choice of the place classified last . The best estimator after the average therefore corresponds here to the choice of the least appreciated place of residence.

8.4 Exercises

Exercise 80 (Orthogonality of characters). *We write $\Phi \overset{\text{def.}}{=} \{\chi_i(C_j)\}_{1\leq i,j\leq p}$ the table of characters. We denote by $K = \operatorname{diag}(k_1, \ldots, k_p)$ the diagonal matrix whose inputs are the cardinals of the conjugation classes. Show that we have $\Phi K \Phi^* = |G|\,\mathrm{Id}$, i.e. the matrix $\frac{1}{\sqrt{|G|}}K^{1/2}\Phi$ is unitary. Deduce from it another proof of the column orthogonality formula, proposition 78.*

Exercise 81 (Representation of the dihedral group). *We consider the dihedral group D_n. Show that it can be realized geometrically as the group formed from the following transformations:*
- *the rotations around the axis Oz and angles $\frac{2k\pi}{n}$, for $k = 0, \ldots, n-1$.*
- *the symmetries with respect to the lines of the plane Oxy forming angles $\frac{k\pi}{n}$ with the axis Ox, for $k = 0, \ldots, n-1$.*

We thus obtain a representation $\rho : D_n \to O_3(\mathbb{R})$. Is it irreducible? Calculate its character, and deduce the decomposition of this representation.

Exercise 82 (Representations of $\mathfrak{S}_3$). *We consider the triangle whose three vertices have for affixes 1, $e^{\frac{2\imath\pi}{3}}$, $e^{-\frac{2\imath\pi}{3}}$, and we fix the base $\{1, \imath\}$ of the complex plane. The group $\mathfrak{S}_3$ acts on the vertices of the triangle by permuting them. Calculate the matrices of two generators of this group (for example (12) and (123)). Deduce the character table of the group $\mathfrak{S}_3$.*

Exercise 83 (Action on the faces of a cube). *As indicated in Paragraph 8.1.4, the group $\mathfrak{S}_4$ can be considered as the group of direct isometries of the cube. It therefore acts by permuting the set of 8 elements formed by the vertices of the cube, which gives rise to a representation of dimension 8. Calculate the character of this representation. Using the character table of $\mathfrak{S}_4$, deduce a decomposition of this representation. Do the same with the permutation of the edges.*

Exercise 84 (Character from $\mathfrak{S}_4$). *We consider the character $\chi_{W'}$ of $\mathfrak{S}_4$ whose table is given by*

	Id	(12)	(123)	(1234)	(12)(34)
$\chi_{W'}$	2	0	−1	0	2

1. *We denote by $\rho_{W'}$ the associated representation. Show that $\rho_{W'}((12)(34)) = \mathrm{Id}$.*

2. *Show that if $H \subset G$ is a distinguished subgroup, then a representation $\rho : G \to GL(V)$ is trivial over H if and only if it is factored by G/H in $\hat{\rho}$:*

$$G \xrightarrow{\pi} G/H \xrightarrow{\hat{\rho}} GL(V)$$

that is, we can identify the representations of G/H with the trivial representations on H.

3. *We denote by H the subgroup of $\mathfrak{S}_4$ generated by $(12)(34)$. Show that $\mathfrak{S}_4/H \simeq \mathfrak{S}_3$. For example, we can consider the action of $\mathfrak{S}_4$ on the opposite faces of a cube.*

4. *Conclude by showing that $\rho_{W'}$ is in fact the standard representation of $\mathfrak{S}_3$.*

Exercise 85 (Representation of a simple group)**.** *Let G be a finite simple non-abelian group. We want to show that G does not have an irreducible representation of dimension 2.*

1. *Start by showing Cauchy's lemma: if p is a prime number which divides $|G|$, then G has an element of order p. To do this, we can consider the set $X = G^p$, as well as the action of the group $\mathbb{Z}/p\mathbb{Z}$ on X:*

$$\left\{ \begin{array}{ccc} (\mathbb{Z}/p\mathbb{Z},\, G) & \longrightarrow & X \\ (k,\, (x_{\overline{0}}, \ldots, x_{\overline{p-1}})) & \longmapsto & (x_{\overline{k}}, \ldots, x_{\overline{p-1+k}}) \end{array} \right. ,$$

to which the equation will be applied to the classes (see the book by Perrin[52]*).*

2. *We assume that G has an irreducible representation $\rho : G \to GL_2(\mathbb{C})$. Assuming the result of the exercise 76, deduce that G has an element t of order 2.*

3. *Show that ρ is in fact valued in $SL_2(\mathbb{C})$. Then show that $\rho(t) \in SL_2(\mathbb{C})$ must be equal to $-\mathrm{Id}$. Deduce that t is in the center of G. Conclude.*

Exercise 86 (Quaternionic group)**.** *We denote by H_8 the quaternionic group, which is formed by 8 elements $\{\pm 1, \pm i, \pm j, \pm k\}$ whose multiplications are given by the sign rule and the formulas*

$$i^2 = j^2 = k^2 = -1, \quad jk = -kj = i, \quad ki = -ik = j, \quad ij = -ji = k.$$

We call $H = \mathbb{R}[H_8]$ the quaternion algebra. For more information on quaternions, see[52].

1. *We denote by $q = a1 + bi + cj + dk$ a generic element of H. Show that the application:*

$$q \mapsto \begin{pmatrix} a & -b & -c & -d \\ b & a & -d & c \\ c & d & a & -b \\ d & -c & b & a \end{pmatrix}$$

allows us to identify H to a subalgebra of $M_4(\mathbb{R})$. Deduce that H is indeed an associative algebra. Also deduce a representation of H_8.

2. *Show that the application*

$$\varphi : q \mapsto M(a + \imath b,\, c - \imath d) \quad \textit{with} \quad M(\alpha, \beta) \overset{\text{def.}}{=} \begin{pmatrix} \alpha & -\overline{\beta} \\ \beta & \overline{\alpha} \end{pmatrix}$$

allows us to identify H to a subalgebra of $M_2(\mathbb{C})$. Deduce a representation of dimension 2 of H_8 on the field of the complexes. Is it unitary?

3. *Calculate the 4 irreducible representations of dimension 1 of H_8. Show that with the representation obtained in the previous question, we have all the irreducible representations. Then give the corresponding orthonormal basis of the space of functions of H_8 in $\mathbb{C}$.*

4. *Explain how H_8 allows to define a non-commutative group structure on the space $\{0, 1\}^3$. Using the result of the exercise 72 describe the representations of the group $\{0, 1\}^{3n}$ seen as the product of the group $\{0, 1\}^3$. Deduce an orthonormal basis from the space of functions of $\{0, 1\}^{3n}$ in $\mathbb{C}$.*

We can compare this construction to that of the Walsh base encountered in Section 2.2, which consisted in using the abelian additive group structure of $\{0, 1\}^n$. We have, in a way, refined the construction to use a non-commutative structure. There are important applications of this type of construction, for example such orthonormal bases make it possible to generalize the technique of learning Boolean functions presented in the exercise 60. It was Boneh, in [8] *who first introduced this process.*

Exercise 87 (Ring of invariants)**.** *We consider G the group of direct isometries of $\mathbb{R}^3$ keeping a cube centered at the origin and whose edges are aligned with the coordinate axes. This exercise does not assume that the isomorphism between G and $\mathfrak{S}_4$ is known. We keep the notations of the exercise 73, and we want to geometrically determine elements of $K[X,\, Y,\, Z]^G$.*

1. *Explain why $X^2 + Y^2 + Z^2 \in K[X,\, Y,\, Z]^G$.*
2. *Show that, if we denote by $f \overset{\text{def.}}{=} XYZ$, then*

   $$\forall A \in G, \quad f(A \cdot (X, Y, Z)) = af(X,\, Y,\, Z), \quad poura \in \mathbb{R}.$$

   Then show that we necessarily have $a = \pm 1$. Conclude that $(XYZ)^2 \in K[X,\, Y,\, Z]^G$.
3. *Similarly, show that the polynomials*

   $$\begin{aligned} & f = (X + Y + Z)(X + YZ)(XYZ)(XYZ) \\ \textit{and} \quad & g = (X^2 - Y^2)(X^2 - Z^2)(Y^2 - Z^2) \end{aligned}$$

   are squared invariant under G.

Exercise 88 (Auto-dual correction codes)**.** *Let $\mathcal{C}$ un linear code on $\mathbb{F}_2$ of size n and dimension k. We denote by $\mathcal{C}^\perp$ its dual code and $W_\mathcal{C}(X,\, Y)$ the weight enumerator polynomial of $\mathcal{C}$. Assume that $\mathcal{C}$ is self-dual, i.e. $\mathcal{C} = \mathcal{C}^\perp$.*

1. *Which relation must n and k verify?*
2. *We denote by A the matrix 2×2 defined by*

   $$A \overset{\text{def.}}{=} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

   Using the identities of MacWilliams 9, show that $W_\mathcal{C}(X,\, Y)$ is invariant under the action of A (as defined in Paragraph 7.1.2).
3. *We denote by G_1 the group generated by A. Write the elements that compose it. Explain why $W_\mathcal{C}(X,\, Y) \in K[X,\, Y]^{G_1}$. Using the result of the exercise 73, show that $K[X,\, Y]^{G_1}$ is generated, in the sense of (7.10), by polynomials*

   $$X + (\sqrt{2} - 1)Y \quad \textit{and} \quad Y(XY).$$

4. *Show that for all $x \in \mathcal{C}$, $w(x)$ is even.*
   Deduce that $W_\mathcal{C}(X,\, Y) \in K[X,\, Y]^{G_2}$, where we noted G_2 the group generated by A and $-\mathrm{Id}_2$.
5. *Write a program Maple which computes generators of the ring of invariants under the action of a given group. Use this program to calculate generators for $K[X,\, Y]^{G_2}$. What are the problems with this method?*

There are more efficient methods to calculate the ring of invariants of a given group. The book of Cox [21] presents the Gröbner bases, and their applications to the theory of invariants.

Appendix A

Correction of exercises

A.1 Correction of the exercises of chapter 1

Correction (exercise 1)

1. Let $\chi \in \hat{G}$. The convolution theorem 2, tells us that $\mathcal{F}(f * \chi) = \hat{f} \cdot \hat{\chi}$. We easily verify that $\hat{\chi} = |G|\delta_{\chi^{-1}}$, where $\delta_{\chi^{-1}} \in \mathbb{C}[\hat{G}]$ is the function that equals 1 in χ^{-1} and 0 otherwise. This allows us to write that $\mathcal{F}(f * \chi) = |G|\hat{f}(\chi^{-1})\delta_{\chi^{-1}}$. It only remains to apply the inverse Fourier transform using the fact that $\mathcal{F}^{-1}(\delta_{\chi^{-1}}) = \frac{1}{G}\chi$. We thus obtain that $\Phi^f(\chi) = f * \chi = \hat{f}(\chi^{-1})\chi$. So χ is an eigenvector for Φ^f, with associated eigenvalue $\hat{f}(\chi^{-1})$.

2. We have $\Phi^f(\delta_g) = f * \delta_g = \sum_{h \in G} f(g^{-1}h)\delta_h$. If we write $\{0, \ldots, n-1\}$ the elements of $G \simeq \mathbb{Z}/n\mathbb{Z}$, we write the matrix $A = \{a_{ij}\}$ in the form $a_{ij} = f(ij)$.
   Moreover, with the previous question, we have the expression of the determinant
$$\det(A) = \prod_{\chi \in \hat{G}} \hat{f}(\chi^{-1}) = \prod_{\chi \in \hat{G}} \hat{f}(\chi).$$

3. Just choose $G = \mathbb{Z}/n\mathbb{Z}$, as well as $\forall i \in \mathbb{Z}/n\mathbb{Z},\ f(i) = a_i$.

4. We can compute $\hat{f}$ in $O(n\log(n))$ operations with the FFT algorithm, and we obtain the determinant by multiplying the inputs of $\hat{f} between them$. The Matlab 2 program does this, on a vector f of size 10 drawn at random.

Program 2 Calculation of circulating determinant

```
n = 10; f = rand (n, 1);
prod(fft(f))
```

Correction (exercise 2)

1. We consider R_1 (resp. R_2) a rotation of unit axis u_1 (resp. u_2), as well as v_1, w_1 (resp. v_2, w_2) an orthonormal basis of the plane $(u_1)^\perp$ (resp. $(u_2)^\perp$). We take Ω the isometry which sends (u_1, v_1, w_1) on (u_2, v_2, w_2). If the two rotations have the same angle, we have $R_1 = \Omega^* R_2 \Omega$.

2. Like $\chi(a^{-1}ba) = \chi(a)^{-1}\chi(b)\chi(a) = \chi(b)$, χ is constant on conjugation classes. With the previous question, $\chi(R)$ depends only on the angle of the rotation R.

Program 3 Calculation of $\mathrm{Tr}(t_\beta)$

```
with  (linalg):
r1:= matrix(3,3, [1,0,0,0, cos (t), sin(t), 0, -sin(t), cos (t)]);
r2:= matrix(3,3, [cos (t), 0, -sin(t), 0,1,0, sin(t), 0, cos (t)]);
tr := trace (r1&*r2);
plot (tr, theta = 0..pi);
```

3. We use the fact that $\mathrm{Tr}(t_\beta) = 1+2\cos(\beta)$. By doing the calculation (with MAPLE, as shown in program 3), we find that $\mathrm{Tr}(r_\alpha s_\alpha^{-1}) = 2\cos(\alpha) + \cos(\alpha)^2$.

   The study of the function $\alpha \mapsto \frac{1}{2}(2\cos(\alpha)+\cos(\alpha)^2-1)$ immediately shows that it is strictly decreasing over $[0,\, \pi]$ and takes all values between 1 and -1.

4. We have therefore, $\forall \beta \in [0,\, \pi],\ \chi(t_\beta) = \chi(r_\alpha)\chi(s_\alpha)^{-1} = 1$. Since a rotation of angle $-\beta$ and axis v can be seen as a rotation of angle β and axis $-v$, the previous equality is still true for $\beta \in [-\pi,\, 0]$. So $\chi = 1$.

Correction (exercise 3) It suffices to use the relation $\delta_0 = \frac{1}{|G|}\sum_{\chi\in\hat{G}} \chi$. We then write that

$$\begin{aligned} N(h) &= \sum_{(x_1,\ldots,\,x_n)\in G^n} \delta_0(\varphi(x_1, \ldots,\, x_n) - h) \\ &= \sum_{(x_1,\ldots,\,x_n)\in G^n} \sum_{\chi\in\hat{G}} \chi(\varphi(x_1, \ldots,\, x_n) - h). \end{aligned}$$

We find the equality requested by noting that

$$\chi(\varphi(x_1, \ldots,\, x_n) - h) = \chi(\varphi(x_1, \ldots,\, x_n))\overline{\chi(h)}.$$

Correction (exercise 4) In the following, we denote by $n = |G|$.

1. We have $\|f_A\|_2^2 = \frac{1}{G}\sum_{x\in A} 1 = \frac{|A|}{|G|}$. Likewise, $\hat{f_A}(\chi_0) = \sum_{x\in G} f_A(x) = |A|$.

2. Using Plancherel's formula, proposition 15, we obtain the equality $\|\hat{f_A}\|_2^2 = n\|f_A\|_2^2 = |A|$. We can increase $\|\hat{f_A}\|_2^2$ as follows:

$$n\|\hat{f_A}\|_2^2 \le |\hat{f_A}(\chi_0)|^2 + (n-1)\Phi(A)^2 = |A|^2 + (n-1)\Phi(A)^2.$$

   We thus obtain $\Phi(A)^2 \ge \frac{1}{n-1}|A|(n-|A|) \ge |A|/2$. The other inequality is trivial, since $|\hat{f_A}(\chi)| \le \sum_{x\in A}|\chi(x)| \le |A|$.

3. We write $B = G\backslash A$. We have $f_B = 1 - f_A$, so $\hat{f_B} = \hat{1} - \hat{f_A} = |G|\delta_{\chi_0} - \hat{f_A}$. We therefore have, for $\chi \ne \chi_0$, $|\hat{f_B}(\chi)| = |\hat{f_A}(\chi)|$, so $\Phi(A) = \Phi(B)$. In the end, we have the inequalities $|G| - |A| \ge \Phi(A) \ge \sqrt{\frac{|G|-|A|}{2}}$

4. If $\chi \ne \chi_0$, then $\hat{f_{\alpha(A)}}(\chi) = \sum_{x\in A} \chi\circ\alpha(x)$. However, the application $\chi \mapsto \chi\circ\alpha$ is a permutation of non-trivial characters. So $\Phi(A) = \Phi(\alpha(A))$.

Correction (exercise 5) In the following, we denote by $n = |G|$.

1. By replacing A_1 by $A_1\backslash a = \{xa \ :\ x \in A_1\}$, we come back to the homogeneous equation $x_1+\cdots+x_n = 0$, with $x_1 \in A_1\backslash a$, which has the same number of solutions as the starting equation.

By applying the result of the exercise 3, with the function $\varphi(x_1, \dots, x_k) = x_1 + \cdots x_k$ and $h = 0$, we get

$$N = \frac{1}{|G|} \sum_{\chi \in \hat{G}} \sum_{x_i \in A_i} \chi(x_1 + \cdots + x_k). \tag{A.1}$$

The last sum can be written

$$\sum_{x_i \in A_i} \chi(x_1 + \cdots + x_k) = \prod_{i=1}^{k} \sum_{x_i \in A_i} \chi(x_i) = \prod_{i=1}^{k} \hat{f_{A_i}}(\chi).$$

The term of the sum (A.1) corresponding to χ_0 gives $\frac{|A_1| \cdots |A_k|}{|G|}$. The other terms give R.

2. For $k = 3$, we have

$$R = \frac{1}{|G|} \sum_{\chi \neq \chi_0} \hat{f_{A_1}}(\chi) \cdot \hat{f_{A_2}}(\chi) \cdot \hat{f_{A_3}}(\chi).$$

Hence the inequality

$$|R| \leq \frac{\Phi(A_3)}{|G|} \sum_{\chi \in \hat{G}} |\hat{f_{A_1}}(\chi)| \cdot |\hat{f_{A_2}}(\chi)|.$$

Using the Cauchy-Schwartz inequality, we get

$$\begin{aligned} \sum_{\chi \in \hat{G}} |\hat{f_{A_1}}(\chi)| \cdot |\hat{f_{A_2}}(\chi)| &\leq \left\{ \left(\sum_{\chi \in \hat{G}} |\hat{f_{A_1}}(\chi)|^2 \right) \left(\sum_{\chi \in \hat{G}} |\hat{f_{A_2}}(\chi)|^2 \right) \right\}^{1/2} \\ &\leq \left(n \|\hat{f_{A_1}}\|^2 n \|\hat{f_{A_2}}\|^2 \right)^{1/2} \\ &= n^2 \|f_{A_1}\| \|f_{A_2}\| = n \sqrt{|A_1||A_2|}. \end{aligned}$$

The translation by a being an automorphism of G, using the result of the exercise 4, question 4, the inequality found is invariant by translation by a.

3. Thanks to question 2, the proposed inequality is a rewriting of $R < \frac{|A_1||A_2||A_3|}{|G|}$. Using the equality proved in question 1, we get $N > 0$.

Correction (exercise 6)

1. We check the associativity of the operation on the given formula. The neutral element is $(1, 1, \chi_0)$, and $(\lambda, x, \chi)^{-1} = (\lambda^{-1}\chi(x), x^{-1}, \chi^{-1})$.

2. We must show the different axioms of a group action (see[52]), in particular, with a slight abuse of notation,

$$\begin{aligned} (\lambda, x, \chi) \cdot [(\mu, y, \tau) \cdot f](z) &= (\lambda, x, \chi) \cdot [\mu \tau(z) f(yz)] \\ &= \lambda \mu \tau(xz) \chi(z) f(xyz) \\ &= [(\lambda, x, \chi)(\mu, y, \tau)] \cdot f(z). \end{aligned}$$

The action of $\mathcal{H}(G)$ on $\mathbb{C}[G]$ is linear (ie the action commutes with the laws of vector space). This is called a linear representation, (see chapter 7).

3. We have

$$(\lambda, x, \chi) \cdot f = D_\lambda \circ M_\chi \circ T_x \circ f.$$

One can, moreover, with this formalism, easily demonstrate the result of the previous question. It suffices to notice that $T_x M_\tau = D_{\tau(x)} M_\tau T_x$, which allows to write

$$(D_\lambda M_\chi T_x)(D_\mu M_\tau T_y) = D_\lambda D_\mu M_\chi (T_x M_\tau) T_y = (D_\lambda D_\mu D_{\tau(x)})(M_\chi M_\tau)(T_x T_y).$$

We can then simplify the terms by using the fact that $\lambda \mapsto D_\lambda$, $\chi \mapsto M_\chi$ and $x \mapsto T_x$ are group morphisms.

4. To simplify the notations, we introduce, for $(\lambda, \chi, x) \in \mathcal{H}(\hat{G})$, the operators of dilation, translation, and modulation by

$$\tilde{D}_\lambda(\varphi)(\tau) = \lambda\varphi(\tau), \qquad \tilde{T}_\chi(\varphi)(\tau) = \varphi(\chi\tau), \qquad \tilde{M}_x(\varphi)(\tau) = \tau(x)\varphi(\tau),$$

where $\tau \in \hat{G}$ and $\varphi \in \mathbb{C}[\hat{G}]$. We then easily show the following relations:

$$\mathcal{F}(T_x f) = \tilde{M}_{x^{-1}}\mathcal{F}(f), \qquad \mathcal{F}(M_\chi) = \tilde{T}_\chi \mathcal{F}(f), \qquad \mathcal{F}(D_\lambda) = \tilde{D}_\lambda \mathcal{F}(f).$$

By analogy with the action of $\mathcal{H}(G)$ on $\mathbb{C}[G]$, we define an action of $\mathcal{H}(\hat{G})$ (whose multiplication remains to be defined!) on $\mathbb{C}[\hat{G}]$ by

$$(\lambda, \chi, x) \cdot f = \tilde{D}_\lambda \tilde{M}_x \tilde{T}_\chi f.$$

To obtain the multiplication law that suits us, we are content to compose the group action with itself:

$$(\tilde{D}_\lambda \tilde{M}_x \tilde{T}_\chi)(\tilde{D}_\mu \tilde{M}_y \tilde{T}_\tau) = (\tilde{D}_\lambda \tilde{D}_\mu \tilde{D}_{\chi(y)})(\tilde{M}_x \tilde{M}_y)(\tilde{T}_\chi \tilde{T}_\tau).$$

Once again, it is the translation/dilation commutation calculation which makes it possible to arrive at the result $\tilde{T}_\chi \tilde{M}_y = \tilde{D}_{\chi(y)} \tilde{M}_y \tilde{T}_\chi$. In the end, we obtain the following law on $\mathcal{H}(\hat{G})$:

$$(\lambda, \chi, x) \cdot (\mu, \tau, y) = (\lambda\mu\chi(y), \chi\tau, xy).$$

In a way, this law was "constructed" so that we get a group action.

5. The fact that α is a morphism results from the following calculation

$$\begin{aligned}\alpha((\lambda\mu\tau(x), xy, \chi\tau)) &= (\lambda\mu\tau(x)(\chi\tau)(x^{-1}y^{-1}, \chi\tau, x^{-1}y^{-1}) \\ &= \big((\lambda\chi^{-1}(x))(\mu\tau^{-1}(y))\chi^{-1}(y), \chi\tau, x^{-1}y^{-1}\big) \\ &= (\lambda\chi^{-1}(x), \chi, x^{-1}) \cdot (\mu\tau^{-1}(y), \tau, y^{-1}).\end{aligned}$$

With the previous question, we have

$$\mathcal{F}(D_\lambda M_\chi T_x f) = \tilde{D}_\lambda \tilde{T}_\chi \tilde{M}_{x^{-1}} \mathcal{F}(f) = D_{\lambda\chi(x^{-1})} M_{x^{-1}} T_\chi \mathcal{F}(f).$$

By translating this equality in terms of group action, we obtain the desired result.

6. We denote, for $g \in \mathcal{H}(G)$, $\rho(g) \in \mathcal{L}(\mathbb{C}[G])$ the linear map $f \mapsto g \cdot f$. Similarly, we note, for $\tilde{g} \in \mathcal{H}(\hat{G})$, $\tilde{\rho}(\tilde{g}) \in \mathcal{L}(\mathbb{C}[\hat{G}])$ linear map $\varphi \mapsto \tilde{g} \cdot \varphi$. ρ and $\tilde{\rho}$ are linear representations of the two finite groups $\mathcal{H}(G)$ and $\mathcal{H}(\hat{G})$. The assumption made on Φ means that $\Phi \circ \rho = \rho \circ \Phi$. By using Schur's lemma 13, we therefore have that Φ is a homothety. If we now assume that Φ commutes with the actions of $\mathcal{H}(G)$ and $\mathcal{H}(\hat{G})$, then Φ interleaves the two representations. Using the corollary 5, since the dimension of the vector space of interleaving morphisms is 1, Φ is written $r\Psi$, where Ψ is a (non-canonical) isomorphism between $\mathbb{C}[G]$ and $\mathbb{C}[\hat{G}]$ fixed (these two spaces have of course the same dimension) and $r \in \mathbb{C}$. As $G \simeq \hat{G}$ (noncanonical), it is even easy to construct an isomorphism of algebras between $\mathbb{C}[G]$ and $\mathbb{C}[\hat{G}]$ (always not canonical).

Correction (exercise 7)

1. We have $\langle \tau_n(f), \tau_p(f)\rangle = f * \tilde{f}(np)$, where we have noted $\tilde{f}(x) = \overline{f}(-x)$. The family $\{\tau_n(f)\}$ is therefore orthonormal if and only if $f * \tilde{f}(n) = \delta_0(0)$. This is written, with the distributions,
$$(f * \tilde{f}) \cdot \Pi_1 = \delta_0, \tag{A.2}$$
where δ_0 is the Dirac at 0. We know that the Fourier transform of $f * \tilde{f}$ is $|\hat{f}|^2$. By taking the Fourier transform of the equation (A.2), and using the convolution property of the Fourier transform (see[57]) we thus find $\frac{1}{2\pi}|\hat{f}|^2 * \hat{\Pi}_1 = 1$. By using the calculation of $\hat{\Pi}_1$ done in exercise 20, we find the desired result.

2. We have $|\hat{\varphi}| \leq A|\hat{f}|$, so $\varphi \in L^2(\mathbb{R})$.
The function φ checks $\sum_{k\in\mathbb{Z}} |\hat{\varphi}(\omega + 2k\pi)|^2 = 1$, so the family $\{\tau_n(\varphi)\}$ is orthonormal.

Correction (exercise 8)

1. The fact that b is orthonormal is equivalent to the fact that $\psi_b = \delta_e$, where e is the neutral element of G. By taking the Fourier transforms of the two members, we find $\hat{\psi_b} = \hat{\delta_e} = 1$, where we have noted 1 the function constant equal to 1. It only remains to notice that $\hat{\psi_b} = |G|\langle \mathcal{U}_\chi b,\, b\rangle$

2. This is to show three things:

   (i) the orthogonality of $\mathcal{U}_{\chi_1}$ and $\mathcal{U}_{\chi_2}$.
   (ii) the idempotence of $\mathcal{U}_{\chi_1}$.
   (iii) that $\mathcal{U}_{\chi_1}$ is self-vice.

   Let's show (i):
$$\begin{aligned}\mathcal{U}_{\chi_1}\mathcal{U}_{\chi_2} &= \frac{1}{|G|^2} \sum_{(U,\,V)\in G^2} UV\chi_1(U)\chi_2(V) \\ &= \frac{1}{|G|^2} \sum_{(U,\,R)\in G^2} R\chi_1(U)\overline{\chi_2(U)}\chi_2(R) \\ &= \frac{1}{|G|^2} \sum_{U\in G} \chi_1(U)\overline{\chi_2(U)} \sum_{R\in G} R\chi_2(R) = \delta_{\chi_1}^{\chi_2}\mathcal{U}_{\chi_2}.\end{aligned}$$
   For the second equality, we used the change of variable $R = UV$, and for the last equality, we used the orthogonality of the characters χ_1 and χ_2.
   To show (ii), the calculation is identical, it suffices to take $\mathcal{U}_{\chi_1} = \mathcal{U}_{\chi_2}$.
   The fact that $\mathcal{U}_{\chi_1}$ is self-adjoint is immediate:
$$(\mathcal{U}_{\chi_1})^* = \sum_{U\in G} U^*\overline{\chi_1(U)} = \sum_{U\in G} U^{-1}\chi_1(U^{-1}) = \mathcal{U}_{\chi_1}.$$
   For the last equality, we simply used the fact that $U \mapsto U^{-1}$ was a permutation on the summation index.

3. To show the orthogonality of $\tilde{b}$, we must perform the following calculation:
$$\begin{aligned}\langle \tilde{b}, \mathcal{U}_\chi(\tilde{b})\rangle &= \langle \sum_{\tau\in\hat{G}} \mathcal{U}_\tau b \frac{1}{\sqrt{\Phi(\tau)}}, \sum_{\tau\in\hat{G}} \mathcal{U}_\tau\mathcal{U}_\chi b\frac{1}{\sqrt{\Phi(\tau)}}\rangle \\ &= \langle \mathcal{U}_\chi b\frac{1}{\sqrt{\Phi(\chi)}}, \mathcal{U}_\chi\mathcal{U}_\chi b\frac{1}{\sqrt{\Phi(\chi)}}\rangle \\ &= \frac{1}{\Phi(\chi)}\langle \mathcal{U}_\chi b, \mathcal{U}_\chi b\rangle = \frac{1}{\Phi(\chi)}\langle b, \mathcal{U}_\chi b\rangle = 1.\end{aligned}$$

For the second equality, we have developed the sums, and we used the orthogonality relations between the $\mathcal{U}_\tau$ demonstrated in the previous question. For the last equality, we used the fact that $\mathcal{U}_\chi$ is self-vice.

4. The exercise 7 is placed in the continuous case and uses the group $\mathbb{R}$ acting unitarily on $L^2(\mathbb{R})$ by translation. It is also about a method of orthogonalization in the field of Fourier.

Correction (exercise 9)

1. We have $\hat{P}(\chi_0) = 1$ and, for $\chi \neq \chi_0$, $\hat{U}(\chi) = 0$. Using Plancherel's formula, proposition 15, we get
$$\|PU\|_2^2 = \frac{1}{|G|}\|\hat{P} - \hat{U}\|_2^2 = \frac{1}{|G|^2}\sum_{\chi \neq \chi_0} |\hat{P}(\chi)|^2.$$

2. It suffices to notice that $\left|P(g) - \frac{1}{|G|}\right| \leq |G|\|PU\|_2^2$.

Correction (exercise 10)

1. We have the relation
$$\mathbb{P}(X_{k+1} = i) = \sum_{j=0}^{n-1} \mathbb{P}(X_k = j)\mathbb{P}(X_{k+1} = i|X_k = j),$$
which means exactly $p^{(k+1)} = Pp^{(k)}$.
By iterating this relation, we get $p^{(k)} = P^k p^{(0)}$.

2. In this particular case, we have $(Px)[i] = (1-p)x[i-1] + px[i+1]$. We recognize a convolution formula, and we have $Px = v * x$. It is also possible to consider independent random variables Y_i with density vector v. We then have $X_k = \sum_{i=1}^k Y_i$. Using the fact that $P_{Y_i + Y_j} = P_{Y_i} * P_{Y_j} = v * v$, we find $p^{(k)} = v * \cdots * v * p^{(0)}$ (k products).

3. Thanks to the convolution theorem 18, we obtain $\hat{p^{(k)}} = \hat{v} \cdot \ldots \cdot \hat{v} \cdot \hat{p^{(0)}}$. We can explicitly calculate $\hat{v}[i] = pe^{\frac{2\imath\pi}{n}} + (1-p)e^{-\frac{2\imath\pi}{n}}$. Since $0 < p < 1$ and k is odd, we have, for $i \neq 0$, $|\hat{v}[i]| < 1$, so $\hat{p^{(k)}} \longrightarrow \delta_{\chi_0}$ when $k \to +\infty$. By taking the inverse transform of this relation, we get $p^{(k)} \longrightarrow u$ when $k \to +\infty$. If k is even, we have $\hat{v}[n/2] = -1$ and the probability does not converge. Intuitively, we see that if we write $\{0, \ldots, n-1\} = P \cup I$ (partition between even and odd), then $p^{(2s)}$ will be ported by P, and $p^{(2s+1)}$ by I, which excludes any convergence (the two sets do not "mix").

4. We have $p^{(k+1)}[i] = \sum_{j=0}^{n-1} v_{ij} p^{(k)}[i] = p^{(k)} * v[i]$. By writing this equation in the Fourier domain, and by iterating it, we see that several situations can arise:
   - If $\exists i,\ |\hat{c}[i]| > 1$, then $p^{(k)}$ will explode. This cannot happen for a probability distribution, since $|\hat{c}[i]| \leq 1$.
   - If $\exists i,\ |\hat{c}[i]| = 1$ and $\hat{c}[i] \neq 1$, then $p^{(k)}$ will not converge.
   - Otherwise, $p^{(k)} \longrightarrow p^\infty$ when $k \to +\infty$, where we have defined p^∞ by $\hat{p^\infty}[i] = \hat{p^{(0)}}[i]$ if $\hat{v}[i] = 1$, and $\hat{p^\infty}[i] = 0$ otherwise.

   We can compare this with the study of polygons (paragraph 4.3.1) which is identical in every way(except that the polygons can explode!). The code Matlab 4 allows, from an initial probability vector `p0`, and from the transition vector `v`, to calculate the probability `pk` at the k^{th} iteration.

Correction (exercise 11) In the following we identify $\widehat{\mathbb{Z}/n\mathbb{Z}}$ with $\mathbb{Z}/n\mathbb{Z}$ and we write $\hat{f}(k)$ for $\hat{f}(\chi_k)$.

Program 4 Calculation of $p^{(k)}$

```
pk = p0;
for i = 1:k
  pk = real(ifft(fft(pk).*fft(v)));
end
```

1. The method is exactly the one used to prove the bound on the minimum distance of BCH codes, proposition 58. Assume that $\text{Supp}(f) = \{a_1, \ldots, a_p\}$. For now, we just assume that $\hat{f}(0) = \cdots = \hat{f}(p-1) = 0$. By noting $\omega = e^{\frac{2\imath\pi}{n}}$, we obtain the system
$$\begin{pmatrix} 1 & 1 & \ldots & 1 \\ \omega^{a_1} & \omega^{a_2} & \ldots & \omega^{a_p} \\ \vdots & \vdots & \ddots & \vdots \\ \omega^{a_1(p-1)} & \omega^{a_2(p-1)} & \ldots & \omega^{a_p(p-1)} \end{pmatrix} \begin{pmatrix} f(a_1) \\ f(a_2) \\ \vdots \\ f(a_p) \end{pmatrix} = 0.$$
The matrix of the system is Vandermonde, it is invertible, which is absurd because the $f(a_i)$ are not all zero. In the general case, if we suppose that p consecutive inputs of $\hat{f}$ are zero, we come back to the previous case by performing a translation on $\hat{f}$ (which amounts to a modulation on f and do not change the support).
It is now easy to see that this implies the principle of uncertainty. Suppose first that $p|N$. We partition $\{0, \ldots, n-1\}$ into n/p blocks of size p. From what we have just shown, on each of these blocks, $\hat{f}$ cannot be zero. Thus, each block contains at least one k such that $\hat{f}(k) \neq 0$ and $|\text{Supp}(\hat{f})| \geq n/p$.
If p does not divide n, we denote by $d = \lceil n/p \rceil$. It is impossible to distribute less than d elements among n places on a circle without leaving two elements with a hole of p places between them. Consequently, we have $|\text{Supp}(\hat{f})| \geq d$ and $|\text{Supp}(f)| \times |\text{Supp}(\hat{f})| \geq dp \geq n$.

2. The first inequality is trivial. For the second, it suffices to use the Fourier inversion formula 13, and write
$$|f(x)| \leq \frac{1}{|G|} \sum_{\chi \in \hat{G}} |\hat{f}(\chi)||\overline{\chi}(x)|,$$
which leads to the desired inequality by taking the maximum of all these inequalities, for $x \in G$.
The preceding inequality is written $M^2 \leq \langle \hat{f}, g \rangle$, where we denote by g the indicator function of $\text{Supp}(f)$. With Cauchy-Schwartz, we get $M^2 \leq \|f\|^2\|g\|^2$, which is the first inequality required. The second inequality is obtained by simply using the Plancherel formula 15. By combining the first equality of question 2, and the equality that we have proved, we find the final inequality.

3. The transform of f_H is studied with the proposition 59.
We therefore have $|\text{Supp}(f_H)| = |H|$ and $|\text{Supp}(f_{H^\sharp})| = |H^\sharp| = |G|/|H|$. The function f_H does indeed reach the established limit.

A.2 Correction of the exercises of Chapter 2

Correction (exercise 12)

1. Each of the residuals $[(-1)^b b]_p$ is clearly even. If there is a duplicate among these residuals, then $(-1)^{ra} ra = (-1)^{ra'} ra' \mod p$, so $a = \pm a' \mod p$, and in the end we have $a + a' = 0 \mod p$. This is impossible because $0 < a + a' < 2p$ and $a + a' \neq p$ (because $a + a'$ is even).

2. By making the product of the elements of B we find $\prod_{b \in B} b = r^{\frac{p-1}{2}} \prod_{a \in A} a \mod p$, since $\text{Card}(A) = \frac{p-1}{2}$. Likewise, by making the product of the elements of the set $\{[(-1)^b b]_p\}$, we find $\prod_{a \in A} a =$

$(-1)^{\sum_{b\in B} b} \prod_{b\in B} b \mod p$. With Euler's criterion, lemma 6, we get $\left(\frac{r}{p}\right) = r^{\frac{p-1}{2}}$, which leads to the desired equality.

3. We notice that $\left\lfloor \frac{ra}{p} \right\rfloor$ is the quotient of the Euclidean division of ra by p, and $[ra]_p$ the remainder. So we have $ra = \left\lfloor \frac{ra}{p} \right\rfloor p + [ra]_b$. By summing over all $a \in A$, we obtain the requested equality. As all a are even and $p = 1 \mod 2$, we have $\sum_{b\in B} b = \sum_{a\in A} \left\lfloor \frac{ra}{p} \right\rfloor \mod 2$. We immediately translate this inequality in terms of the power of -1.

4. If there was a point (x, y) on $]AB[$, with $x \leq r$ and $y \leq p$, then we would have $py = rx$, which n'is not, since p and r are distinct primes. On each horizontal line $x = a$, with a even, i.e. for $a \in A$, the number of points below $[AB]$ is $\left\lfloor \frac{ra}{p} \right\rfloor$. By summing for $a \in A$, we count all the even abscissa points below $[AB]$.

5. We denote by $C_p(F)$ the number of even abscissa points in a figure F, $C_i(F)$ the number of odd abscissa points and $C(F)$ the total number of points.
We denote by n_1 the number of points of abscissa a located below $[AB]$, and n_2 the number of points located above. We have $n_1 + n_2 = r - 1$ which is even, so $n_1 = n_2 \mod 2$. This therefore means that $C_p(HKBD) = C_p(MHB) \mod 2$.
By symmetry with respect to H, n_2 is equal to the number of abscissa points pa located below of $[AB]$. This therefore means that $C_p(MHB) = C_i(ALH) \mod 2$.
We have $C_p(ABD) = C_p(AKH) + C_p(HKDB) \mod 2$ (decomposition of ABD in two). So $C_p(ABD) = C_p(AKH) + C_i(ALH) \mod 2$. By symmetry with respect to $[AB]$, we have $C_i(ALH) = C_i(AKH)$. In the end, we have $C_p(ABD) = C_p(AKH) + C_i(AKH) = C(AKH)$.

6. The previous question tells us that $\left(\frac{r}{p}\right) = (-1)^{C(AKH)}$. By swapping the roles of r and p, we also have $\left(\frac{p}{r}\right) = (-1)^{C(AHL)}$. In the end, we get
$$\left(\frac{r}{p}\right)\left(\frac{p}{r}\right) = (-1)^{C(AKH)+C(AHL)} = (-1)^{C(AKHL)} = (-1)^{\frac{(p-1)(r-1)}{4}}.$$

Correction (exercise 13)

1. This question reflects the subgroup structure of $\mathbb{F}_q^*$, which is a cyclic group, see for example[23]. We show that the only group of index k of $\mathbb{F}_q^*$ is formed by the roots of the polynomial $X^{\frac{q-1}{k}} - 1$, it is therefore H_k.

2. We have
$$\sum_{i=0}^{k-1} G(\chi_i, \psi) = \sum_{x\in\mathbb{F}_q^*} \psi(x) \sum_{i=0}^{k-1} \chi_i(x).$$
By the orthogonality proposition 2.9, applied to the multiplicative characters of the group $\mathbb{F}_q^*/H_k$, $\sum_{i=0}^{k-1} \chi_i(x)$ is 0 if $x \notin H_k$, and k if $x \in H_k$. We thus obtain
$$\sum_{i=0}^{k-1} G(\chi_i, \psi) = k \sum_{x\in H_k} \psi(x) = k\hat{f_{H_k}}(\psi).$$
Let ψ be a non-trivial additive character. Thanks to the proposition 26, we have the upper bound
$$\left|\hat{f_{H_k}}(\psi)\right| \leq \frac{1}{k} \sum_{i=0}^{k-1} |G(\chi_i, \psi)| = \frac{1}{k}\left(1 + (k-1)\sqrt{q}\right) < \sqrt{q}.$$

3. We note $A_3 = H_k$, hence $|A_3| = \frac{q-1}{k}$. Since $\mathbb{F}_q$ has k roots of unity, the equation $z^k = u$ has k roots, and therefore $N = kN'$. Using the result of the exercise 5, question 2, we have
$$\left| N' - \frac{|A_1||A_2||H_k|}{q} \right| < \Phi(H_k)\sqrt{|A_1||A_2|},$$
which gives the desired inequality.

4. Under the hypothesis $q \geq k^2 l_1 l_2 + 4$, we can bound the right side of the equation (2.38):
$$k\sqrt{|A_1||A_2|q} = k|A_1||A_2|\sqrt{\frac{l_1 l_2 q}{(q-1)^2}} \leq |A_1||A_2|\sqrt{\frac{(q-4)q}{(q-1)^2}}.$$
We then show, by a function study, that, for $q \geq 0$, $\frac{(q-4)q}{(q-1)^2} \leq \frac{(q-1)^2}{q^2}$, which allows to have the inequality
$$\left| N - \frac{|A_1||A_2|(q-1)}{q} \right| < \frac{|A_1||A_2|(q-1)}{q},$$
and show that $N > 0$.
In the case where k does not divide $k-1$, we denote by $d = \text{GCD}(q-1,\, k)$. Let $A = \{x^k\}$ and $B = \{x^d\}$ As $d|k$, $k = \lambda d$ for some λ, hence $x^k = (x^\lambda)^d$ and $A \subset B$. Conversely, with Bezout's theorem, $\exists (u,\, v)$ such that $d = u(q-1) + vk$. So we have $x^d = (x^{q-1})^u (x^v)^k = (x^v)^k$, so $B \subset A$. So if $q \geq k^2 l_1 l_2 + 4$, we have $q \geq d^2 l_1 l_2 + 4$, which makes it possible to apply the previous reasoning to the group H_d and shows that the equation considered still has a solution.

5. Considering $A_1 = A_2 = H_d$, where $d = \text{GCD}(k,\, q-1)$, we have $|A_i| = \frac{q-1}{d} \geq \frac{q-1}{k}$. This implies $l_i \leq k$ so $k^2 l_1 l_2 + 4 \leq k^4 + 4$.

Correction (exercise 14) An example of an interaction of order 2:
$$\alpha_{ab} = \frac{1}{4}(\alpha_{+++} + \alpha_{++-} + \alpha_{-+} + \alpha_{---}) - \frac{1}{4}(\alpha_{+-+} + \alpha_{-++} + \alpha_{+-} + \alpha_{-+-}).$$
The 3 order interaction:
$$\alpha_{abc} = \frac{1}{4}(\alpha_{+++} + \alpha_{+-} + \alpha_{-+-} + \alpha_{+-}) - \frac{1}{4}(\alpha_{++-} + \alpha_{+-+} + \alpha_{-++} + \alpha_{---}).$$
By reordering the interactions (to put them in the "de Yates" order), we obtain a matrix writing
$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} \alpha_{+++} \\ \alpha_{-++} \\ \alpha_{+-+} \\ \alpha_{-+} \\ \alpha_{++-} \\ \alpha_{-+-} \\ \alpha_{+-} \\ \alpha_{---} \end{pmatrix} = \begin{pmatrix} 8\mu \\ 4\mu_a \\ 4\mu_b \\ 4\mu_{ab} \\ 4\mu_c \\ 4\mu_{ac} \\ 4\mu_{bc} \\ 4\mu_{abc} \end{pmatrix},$$
where, more compactly $W_8 \tilde{\alpha} = \tilde{\mu}$, with obvious conventions (W_8 is the Walsh matrix, equation (2.20)) . Multiplication can be performed quickly by the FWT algorithm.

Correction (exercise 15)

1. On checks that $h \mapsto 2^{\frac{j}{2}} h(2^j \cdot -k)$ is an isometry of $L^2([0,\,1])$, so in particular $\|\psi_{j,k}\|_2^2 = \|\psi\|_2^2 = 1$.
   If $k_1 \neq k_2$, ψ_{j,k_1} and ψ_{j,k_2} have disjoint supports so $\langle \psi_{j,k_1},\, \psi_{j,k_2}\rangle = 0$. If $j_1 < j_2$, then ψ_{j_1,k_1} is constant on the support of ψ_{j_2,k_2}, and like $\int \psi_{j_2,k_2} = 0$, we still have $\langle \psi_{j_1,k_1},\, \psi_{j_2,k_2}\rangle = 0$.
   Like $\dim(E_j) = 2^j$, the family $\{\psi_n\}_{n=0}^{2^j-1}$ is an orthonormal basis of E_j.

2. Since f is continuous over $[0,\,1]$ which is compact, there exists a uniform continuity modulus C such that $|f(x) - f(y)| \leq C(|xy|)$, with $C(t) \longrightarrow 0$ when $t \to 0$. To conclude on the uniform convergence of f_j, it suffices to notice that there exists $x_k \in I_k$ such that $f_j(I_k) = f(x_k)$ (theorem of intermediate values). For $x \in I_k$, we therefore have
$$|f_j(x) - f(x)| \leq |f_j(x) - f_j(x_k)| + |f(x_k) - f(x)| \leq C(|x - x_k|) \overset{j\to+\infty}{\longrightarrow} 0.$$
   There is still a bit of work to be done to infer the convergence of $\tilde{f}_n$. The complementary term must be checked
$$R_j(x) \overset{\text{def.}}{=} \sum_{k=0}^{2^j-1} |\langle f,\, \psi_{j,k}\rangle||\psi_{j,k}(x)|.$$
   In fact, the scalar products must be finely increased.
   We denote by $A_1 \overset{\text{def.}}{=} [k2^{-j},\, (k+1/2)2^{-j}[$ and $A_2 \overset{\text{def.}}{=} [(k+1/2)2^{-j},\, (k+1)2^{-j}[$. We have
$$|\langle f,\, \psi_{j,k}\rangle| = 2^{j/2}\left|\int_{A_1} f - \int_{A_2} f\right| = 2^{j/2}2^{-j+1}|f(x_1) - f(x_2)| \leq 22^{-j/2}C(2^{-j}).$$
   where $x_1 \in A_1$ and $x_2 \in A_2$. In the end, if we take $x \in I_k$, we get the markup
$$R_j(x) = |\langle f,\, \psi_{j,k}\rangle|2^{j/2} \leq 2C(2^{-j}) \overset{j\to+\infty}{\longrightarrow} 0,$$
   which shows the uniform convergence of $|\tilde{f}_n| \leq |f_J| + |R_J|$ for a certain J.
   We have $\|\tilde{f}_n - f\|_2 \leq \|\tilde{f}_n - f\|_\infty$, therefore, the sequence $\tilde{f}_n$ also converges in the norm L^2, and $\{\psi_n\}$ forms a Hilbert basis.

3. The functions $\varphi_{j,k}$ are the indicator functions of the intervals I_k multiplied by $2^{j/2}$. This family therefore forms a base of F_J. Their supports being disjoint, they are two by two orthogonal. The transformation $h \mapsto 2^{j/2}h(2^j \cdot -k)$ conserving the energy, this basis is orthonormal.
   If we set $g_{j-1} = f_j - f_{j-1}$, we see that $g_{j-1} = \sum_{k=0}^{2^{j-1}-1} \langle f,\, \psi_{j,k}\rangle$. This means that if we write $F_j = F_{j-1} \oplus G_{j-1}$, we have
$$\begin{aligned} F_{j-1} &= \operatorname{Span}\left\{\varphi_{j-1,k} \;:\; k = 0, \ldots, 2^{j-1}-1\right\} \\ \text{et} \quad G_{j-1} &= \operatorname{Span}\left\{\psi_{j,k} \;:\; k = 0, \ldots, 2^{j-1}-1\right\}. \end{aligned}$$
   We have the following relations, for $k = 0, \ldots, 2^{j-1}-1$:
$$\psi_{j,k} = \frac{\varphi_{j,2k} - \varphi_{j,2k+1}}{\sqrt{2}} \quad \text{and} \quad \varphi_{j-1,k} = \frac{\varphi_{j+1,2k} + \varphi_{j+1,2k+1}}{\sqrt{2}}. \tag{A.3}$$

4. The quantity $x^{(0)}[k]$ is simply the value that f takes over the interval I_k, multiplied by $2^{j/2}$.
   To study the operator $\Phi_i : x^{(i)} \mapsto (x^{(i+1)},\, d^{(i+1)})$, we must consider
   – for the starting set, the canonical basis of $\mathbb{R}^N$, $N = 2^{ji}$.
   – for the arrival set, the base "alternate" $\{e_0,\, f_0, \ldots, e_{N/2-1},\, f_{N/2-1}\}$, where we noted $\{e_0, \ldots, e_{N/2-1},\, f_0, \ldots, f_{N/2-1}\}$ the canonical basis of $\mathbb{R}^{N/2} \times \mathbb{R}^{N/2}$.

   In these databases, the matrix of Φ_i is a diagonal of blocks $\frac{1}{\sqrt{2}}\left(\begin{smallmatrix} 1 & 1 \\ -1 & 1 \end{smallmatrix}\right)$, so it's an angle rotation $\pi/2$ on each subspace $\operatorname{Span}(e_i,\, f_i)$.
   We check that $d^{(i)}$ has zero mean, so $x^{(i)}$ has the same mean as $x^{(i+1)}$. Gradually, the average of $x^{(0)}$ is that of $x^{(i)}$ and therefore in the end is worth $x^{(j)}[0]$.

5. We define $\tilde{x}^{(i)}[k] \overset{\text{def.}}{=} \langle f,\, \varphi_{ji,k}\rangle$ and $\tilde{d}^{(i)}[k] \overset{\text{def.}}{=} \langle f,\, \psi_{j-i+1,k}\rangle$. We have $\tilde{x}^{(0)} = x^{(0)}$. This is to show that $\tilde{x}^{(i)}$ and $\tilde{d}^{(i)}$ satisfy the same recurrence equations as $x^{(i)}$ and $d^{(i)}$. This will imply that $\tilde{x}^{(i)} = x^{(i)}$ and $\tilde{d}^{(i)} = d^{(i)}$ These relationships are obvious by taking the dot products of the equations (A.3) with the function f.

6. As Φ_i is an isometry, we have $\|x^{(i)}\|_2^2 = \|x^{(i+1)}\|_2^2 + \|d^{(i+1)}\|_2^2$. By iterating, we find

$$\|f\|_2^2 = \|x^{(0)}\|_2^2 = \sum_{i=1}^{j} \|d^{(i)}\|_2^2 + |x^{(j)}[0]|^2 = \|\Gamma(x^{(0)})\|_2^2.$$

   Which means that Γ is an isometry. This amounts to decomposing the discrete signal $x^{(0)}$ on the basis of $\mathbb{R}^n$ formed by Γe_i, where e_i is the canonical basis of $\mathbb{R}^n$. This base corresponds to the functions $\psi_{j,k}$ sampled with a step of 2^{-j}. We call this base the discrete Haar base.
   Compared to the Walsh base, this base is made up of functions with very compact support. The larger j becomes, the smaller the support is.
   Since the vector $x^{(i)}$ is of size $n/2^i$, the application of the operator Φ_i requires $cn2^{-i}$ operations (c is a constant, representing addition, subtraction, and two divisions by $\sqrt{2}$). The algorithm therefore requires $\sum_{i=0}^{j} cn2^{-i} = 2cn$, that is to say $O(n)$ operations. This is much faster than the Walsh transform, which requires $O(n\log(n))$ operations!

The program 5 implements a Matlab function which realizes the Γ operator.

Program 5 Procedure `haar`

```
function y = haar (x)
n = length(x); j = log2(n); y = [];
for i = 0:j-1
  d = 1/sqrt(2) * (x(1:2:2∧(ji)) - x(2:2:2∧(ji)));
  y = [y; d];
  x = 1/sqrt(2) * (x(1:2:2∧(ji)) + x(2:2:2∧(ji)));
end
y = [y; x];
```

Correction (exercise 16)

1. Let $n = 2^k$. The 2D Walsh transform can be written simply, for $f \in \mathbb{C}^{n\times n}$,

$$\mathcal{W}_k(f)[i,\, j] \overset{\text{def.}}{=} \sum_{s=0}^{n-1}\sum_{t=0}^{n-1} f[s,\, t]\chi_{i,\,j}(s,\, t).$$

   The inverse transform is $\mathcal{W}_k^{-1} = \frac{1}{n^2}\mathcal{W}_k$. The 2D Walsh transform corresponds to applying a 1D transform on the columns, then a 1D transform on the rows. By using the `fwt` function written in the B.1 section, we can define a Matlab function performing the transform, as shown in the program 6.

Program 6 Procedure `fwt2d`

```
function y = fwt2d (x)
n = length(x); y = zeros(n, n);
for (i = 1:n) y(i,:)  = fwt(x(i,:)')'; end ;
for (j = 1:n) y(:,j) = fwt(y(:,j)); end ;
```

2. For clarity, we denote by $\chi_i^{(k)}$ the character χ_i on $(\mathbb{Z}/2\mathbb{Z})^k$, which can be seen as a vector of $\{\pm 1\}^n$. We denote by $n_i^{(k)}$ the number of sign changes of $\chi_i^{(k)}$, that is to say

$$n_i^{(k)} \overset{\text{def.}}{=} \#\left\{s = 1, \ldots, 2^k - 1 \;:\; \chi_i^{(k)}[s] \neq \chi_i^{(k)}[s-1]\right\}.$$

We will show, by induction on k, that $n_i^{(k)}$ provides a new numbering of $\chi_i^{(k)}$, that is to say that $i \mapsto n_i^{(k)}$ is a permutation of $\{0, \ldots, 2^k - 1\}$. For $k = 1$, we have $n_0^{(1)} = 0$ and $n_1^{(1)} = 1$ so the property is true. We assume the property to be true for $i \mapsto n_i^{(k)}$. For $i = 0, \ldots, 2^k - 1$, we have, by denoting (a, b) the concatenation of the vectors a and b,

$$\chi_i^{(k+1)} = \left(\chi_i^{(k)}, \chi_i^{(k)}\right) \quad \text{and} \quad \chi_{i+2^k}^{(k+1)} = \left(\chi_i^{(k)}, -\chi_i^{(k)}\right).$$

This implies the following relationships on sign changes:

$$n_i^{(k+1)} = 2n_i^{(k)} + \epsilon_i^{(k)} \quad \text{and} \quad n_{i+2^k}^{(k+1)} = 2n_i^{(k)} + (1 - \epsilon_i^{(k)}), \tag{A.4}$$

where $\epsilon_i^{(k)} = 1$ if there is a discontinuity in the middle of $\chi_i^{(k+1)}$, which is to say that $\chi_i^{(k)}[2^k - 1] = -1$. With the relations found, it is easy to see that the $n_i^{(k+1)}$ covers all $\{0, \ldots, 2^{k+1} - 1\}$.

3. The Walsh spectrum is calculated using the `fwt` function (see Section B.1). We can then classify the spectrum by number of sign changes. This can be done quickly by calculating the number of sign changes at the same time as the transform using the equations (A.4). Indeed, the quantities $\epsilon_i^{(k)}$ also satisfy an induction equation, $i = 0, \ldots, 2^k - 1$, we have

$$\epsilon_i^{(k+1)} = \epsilon_i^{(k)} \quad \text{and} \quad \epsilon_{i+2^k}^{(k+1)} = 1 - \epsilon_i^{(k)}.$$

For example, the routine Matlab 7 calculates the vector $n^{(k)}$.

Program 7 Procedure `nbr_chgt_signe`

```
function nk = nbr_chgt_signe (n)
p = log2(n); nk = 0; ek = 0;
for k = 1:p
  ek = [ek; 1-ek]; nk = 2*[nk; nk] + ek;
end
```

4. Keeping only the coefficients corresponding to functions with few sign changes, we reconstruct a function with less detail. This has the effect of conserving less information, and therefore allows signal compression.
Walsh transform compression is quick to calculate (only additions and subtractions). On the other hand, it introduces discontinuities in the signal which are often unacceptable.

5. The number of sign changes is not well defined for a 2D function. For a function $\chi_{i,j}$, we denote by n_i the number of sign changes on the x axis, and n_j the number of sign changes on the $yaxis$. We can for example classify the functions in order of increasing $n_i + n_j$, and we rule the equality cases in increasing order of n_j.

Correction (exercise 17)

1. As we have $\mathcal{W}_k^{-1} = \frac{1}{2^k}\mathcal{W}_k = \frac{1}{2^k}\mathcal{W}_k^\top$, we have $W_n W_n^\top = n\,\mathrm{Id}_n$, with $n = 2^k$.

2. The first thing to notice is that if we permute the rows and columns of a Hadamard matrix, then it remains of Hadamard. It is the same if we multiply a row or a column by -1. By multiplying by -1 the columns then the rows which start with -1, we thus put the matrix in normalized form. By permutation on the lines, one can rearrange the first three lines so that they are of the announced form. By the orthogonality properties between these three lines, we obtain the equations

$$i + j + j + l = n, \quad i + j - k - l = 0, \quad i - j + k - l = 0, \quad i - j - k + l = 0.$$

This leads to $i = j = k = l = \frac{n}{4}$, and therefore n is divisible by 4.

3. To begin with, $\eta(ij) = \eta(-1)\eta(ji) = -\eta(ji)$ since $p = 4k-1$ (use the formula d'Euler 6, to see it). So Q is anti-symmetric.
   Let $c \neq 0$, we then calculate
$$\sum_{b=0}^{p-1} \eta(b)\eta(b+c) = \sum_{b=1}^{p-1} \eta(b)\eta(bz) = \sum_{z\neq 1} \eta(z) = 0 - \eta(1) = -1.$$
   We made the change of variable $z = \frac{b+c}{b}$, using the fact that $b \mapsto z$ is a bijection from $\mathbb{F}_q^*$ to $\mathbb{F}_q \backslash \{1\}$.
   If $i = j$, we have $(QQ^\top)_{ii} = \sum_{b=0}^{p-1} \eta(b) = p-1$. If $i \neq j$, we have
$$(QQ^\top)_{ij} = \sum_{k=0}^{p-1} \eta(ki)\eta(kj) = \sum_{b=0}^{p-1} \eta(b)\eta(b+c) = -1,$$
   with $b = ki$ and $c = ij$.
   As $\mathbb{F}_p^*$ contains $\frac{1}{2}(p-1)$ quadratic residues and as many non-residues, each row of Q contains as many $+1$ as there are -1 and $QJ = JQ = 0$.

4. We have
$$H_n H_n^\top = \begin{pmatrix} 1 & v \\ v^\top & Q - \mathrm{Id}_p \end{pmatrix} \begin{pmatrix} 1 & v^\top \\ v & Q^\top - \mathrm{Id}_p \end{pmatrix} = \begin{pmatrix} p+1 & 0 \\ 0 & J + (Q - \mathrm{Id}_p)(Q^\top - \mathrm{Id}_p) \end{pmatrix},$$
   and $J + (Q - \mathrm{Id}_p)(Q^\top - \mathrm{Id}_p) = J + p\,\mathrm{Id}_p - J - Q - Q^\top + \mathrm{Id}_p = (p+1)\,\mathrm{Id}_p$.

5. The quantity $|\det(A)|$ measures the volume of the parallelepiped generated by the column vectors of A. This volume is smaller than the product of the norms of these vectors. If we have $|a_{ij}| \leq 1$, then the norm of a column vector is bounded by $\sqrt{n}$ and we find the Hadamard bound. If A is a Hadamard matrix, we have $\det(A)^2 = \det(AA^\top) = \det(n\,\mathrm{Id}_n) = n^n$.

The procedure Maple 8 builds the Paley matrix H_n, and we can test it with the program 9.

Program 8 Procedure `MatricePaley`
```
MatricePaley:= proc  (p)
local H, Q, i, j;
with  (numtheory):
Q:= Matrix(1..p, 1..p);
for i from 1 to p do
for j from 1 to p do
  Q [i, j]:= legendre(ij, p);
end do :
end do :
H:= Matrix(1..p+1,1..p+1);
H[2..p+1,2..p+1] := Q-Matrix(1..p, 1..p, shape = identity);
H[1..p+1.1] := 1; H[1,1..p+1] := 1;
return  H;
end proc :
```

Program 9 Paley construction test
```
H:= MatricePaley(31);
evalm (H & * transpose (H));
```

Correction (exercise 18)

1. We check that
$$(A\otimes A)(A\otimes A)^* = \begin{pmatrix} \langle A_1,\, A_1\rangle AA^* & \cdots & \langle A_1,\, A_s\rangle AA^* \\ \vdots & & \vdots \\ \langle A_s,\, A_1\rangle AA^* & \cdots & \langle A_s,\, A_s\rangle AA^* \end{pmatrix} = s^2\,\mathrm{Id}_{s^2},$$
where we have denoted A_i the i^{ith} column of A. By induction, we obtain the desired result.

2. For the Walsh transform, we have $W_{2^k} = A^{\otimes k}$ with $A = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$.

3. The Matlab 10 procedure performs the calculation of the transform quickly(of the order of $O(N\log_s(N))$ operations, where N is the size of the vector to transform). We have $x = (A^{\otimes n})^{-1}y = \frac{1}{s^n}(A^*)^{\otimes n}$. The inverse transform is therefore obtained by passing A^* instead of A to the algorithm, and dividing the result by s^n.

Program 10 Procedure `decompose_tensoriel`

```
function y = decompose_tensoriel (x, A)
s = length(A); m = length(x); m0 = m/s;
if  (m==1) y = x; return ; end ;
B = zeros(m0, s); % temporary results
for j = 1:s
  sel = ((j-1) * m0+1):( j * m0);
  B (:,j) = decompose_tensoriel(x(sel), A);
end
y = zeros(m, 1);
for i = 1:s
  sel = ((i-1) * m0+1):( i * m0);
  for j = 1:s
    y(salt) = y(salt) + A(i, j) * B (:,j);
  end
end
```

4. For $\alpha = \pi/4$, we get the ordinary Walsh transform. For $\alpha = \pi/2$, the transform performs symmetry.

Correction (exercise 19)

1. $\langle\cdot,\,\cdot\rangle$ is the canonical bilinear form over $E\times E$. We identify an element $x\in E$ has the linear form $\langle x,\,\cdot\rangle$. This identification works because the form is non-degenerate, and corresponds to the hook of duality.

2. We consider the application $\Phi : a \mapsto \chi_a$. It is a morphism of groups. It is easy to show that it is injective. Indeed, if $\Phi(a) = 0$, this means that for all $x\in E$ we have $\chi_1(\langle x,\, a\rangle) = 1$, and like χ_1 is injective, $\forall x\in E,\ \langle x,\, a\rangle = 0$. Since the bilinear form $\langle\cdot,\,\cdot\rangle$ is non-degenerate, we have $a = 0$. Finally, since E and $\hat{E}$ have the same dimension, the morphism is bijective.

3. If $x\in H$, by the group property, $x+\cdots+x = kx\in H$, then H is stable for the external distribution, so it is a vector space.
We have $H^\sharp = \{\chi_a\ :\ \forall x\in H,\ \chi_a(x) = 0\}$, which is in correspondence by Φ with the set
$$\{a\ :\ \forall x\in H,\ \langle x,\, a\rangle = 0\} = H^\perp.$$

4. The Poisson formula, 30, is still valid on E. The proof of MacWilliams' formula, theorem 5, is still the same. Only the calculation of $\hat{f}(\chi_a)$ is slightly changed, since we have
$$\hat{f}(\chi_a) = (x+(q-1)y)^{kw(a)}(xy)^{w(a)}.$$

Correction (exercise 20) The Poisson formula is written, for a function f of the class of Schwartz $\mathcal{S}(\mathbb{R})$:

$$\sum_{n\in\mathbb{Z}} \hat{f}(n) = \frac{2\pi}{s} \sum_{n\in\mathbb{Z}} f\left(\frac{2n\pi}{s}\right).$$

We check that in the sense of distributions, the left member is equal to $\langle \hat{f},\, \Pi_1\rangle = \langle f,\, \hat{\Pi}_1\rangle$, and that the member right is equal to $\langle \frac{2\pi}{s}\Pi_{\frac{2\pi}{s}},\, f\rangle$. The equality being valid for all f in $\mathcal{S}(\mathbb{R})$, we deduce the requested formula.

Correction (exercise 21)

1. We have, with the Fourier inversion formula,

$$f(x) = \frac{1}{2\pi}\int_{I_T} \hat{f}(\omega) e^{\imath\omega x} \mathrm{d}\omega.$$

With the derivation theorem under the sign of integration, we see that f is of class $\mathcal{C}^\infty$.

2. By applying the result of the exercise 20, we can easily see that

$$\hat{f}_d(\omega) = \frac{1}{T}\sum_{k\in\mathbb{Z}} \hat{f}\left(\omega - \frac{2k\pi}{T}\right).$$

Indeed, we have $f_d = f \cdot \Pi_T$, and by taking the Fourier transform of the two members (in the sense of distributions), we find $\hat{f}_d(\omega) = f * \hat{\Pi}_T(\omega)$. If $n \neq 0$, the support of $\hat{f}\left(\cdot - \frac{n\pi}{T}\right)$ and that of $\hat{f}$ are of empty intersection. This therefore implies the requested result.

3. The result of the previous question can be written $\hat{f}(\omega) = Th_T\hat{f}_d$, where h_T is the indicator of the interval I_T. Its inverse Fourier transform is $\mathcal{F}^{-1}(h_T) = T\,\mathrm{sinc}_T$. By taking the inverse Fourier transform of the relation $\hat{f}(\omega) = Th_T\hat{f}_d$, we find the sampling theorem.

4. We write $g = \mathrm{sinc}_T$. The fact that the functions $\{g(. - nT)\}$ are orthogonal is immediately verified using the Plancherel formula:

$$\langle g,\, g(. - nT)\rangle = \frac{1}{2\pi}\langle \mathcal{F}(g),\, \mathcal{F}(g(. - nT))\rangle = \frac{T^2}{2\pi}\langle h_T,\, h_T(\cdot)e^{-\imath nT\cdot}\rangle = T\delta_0^n.$$

The fact that this family is total results from the sampling theorem, since the reconstruction formula (2.41), converges in the norm L^2.
The projection on this basis corresponds to l'sampling, since we have the relation $\langle f,\, g(. - nT)\rangle = f(nT)$.

A.3 Correction of the exercises of chapter 3

Correction (exercise 22)

1. We have $u^{(1)} = (0,\, 1)$, $u^{(2)} = (0,\, 2,\, 1,\, 3)$ and $u^{(3)} = (0,\, 4,\, 2,\, 6,\, 1,\, 5,\, 3,\, 7)$.

2. Let $k^{(n+1)} = \sum_{i=0}^{n} k_i 2^i$ be an integer of $n+1$ bits. We denote by $\tilde{k}^{(n+1)}$ the integer with the bits reversed. We have

$$\tilde{k}^{(n+1)} = \sum_{i=0}^{n} k_i 2^{ni} = \sum_{i=0}^{n-1} k_i 2^{ni} + k_n = 2\tilde{k}^{(n)} + k_n,$$

which is exactly the recurrence equation verified by $u^{(n)}$.

3. $\tilde{f}$ is useful for building an FFT algorithm that does not use temporary memory, as explained in Paragraph 3.2.5.

4. We have $\tilde{f}_g = \tilde{f^0}$ and $\tilde{f}_d = \tilde{f^1}$.

5. The previous question gives birth to a recursive Matlab function, program 11. This procedure requires $O(N\log(N))$ operations, which is the same complexity as the `rev_bits` procedure.

Program 11 Procedure `rev_bits_rec`

```
function y = rev_bits_rec(x)
n = length(x);
if  (n> 1) y = [rev_bits_rec(x(1:2:n)); rev_bits_rec(x(2:2:n))];
else y = x; end
```

Correction (exercise 23)

1. φ is a ring morphism, and we can explain its inverse. With Bezout's theorem, $\exists (u\,v),\ up + vq = 1$. We then take $\psi(k_1,\,k_2) = k_2 up + k_1 vq \mod N$.

2. We write $\omega_r = e^{\frac{2\imath\pi}{r}}$. We first calculate the 2D transform:
$$\hat{F}[s_1,\,s_2] = \sum_{k_1,\,k_2} f[\psi(k_1,\,k_2)]\omega_p^{-s_1k_1}\omega_q^{-s_2k_2} = \sum_{k_1,\,k_2} f[\psi(k_1,\,k_2)]\omega_N^{-(qs_1k_1+ps_2k_2)}.$$
Let us calculate the following quantity A:
$$A \overset{\text{def.}}{=} \psi(k_1,\,k_2)(s_1q + s_2p) = qk_1s_1(qv) + pk_2s_2(pv) \mod N.$$
We want to show that $A = qs_1k_1 + ps_2k_2 \mod N$. With the Chinese theorem, it suffices to show that this equality is true modulo p and modulo q, which we verify without problem. So we have
$$\begin{aligned}\hat{F}[s_1,\,s_2] &= \sum_{k_1,\,k_2} f[\psi(k_1,\,k_2)]\omega_N^{-\psi(k_1,\,k_2)(s_1q+s_2p)} \\ &= \sum_{k} f[k]\omega_N^{-k(s_1q+s_2p)} = \hat{f}[s_1q + s_2p],\end{aligned}$$
where we just made the change of index $k = \psi(k_1,\,k_2)$.

3. We have $0 \le s_1q + s_2p \le N-1$. Moreover, by the Chinese lemma, the system of two equations $\{n = s_1q \mod p\ ;\ n = s_2p \mod q\}$ has a unique solution modulo N, and the representative of this solution is therefore n.
To calculate the FFT of f, it suffices to calculate the FFT 2D of F, and reorder the indices according to $(s_1,\,s_2) \mapsto s_1q + s_2p \mod N$.

4. The step of the Good-Thomas algorithm presented makes it possible to replace the step of the Cooley-Tukey algorithm explained in Paragraph 3.2.4. This replacement avoids the internal multiplication by ω_N^{-bd} in the equation 3.17, which corresponds to the operator $\mathcal{S}_N^x$. The Chinese lemma has somehow eliminated the "twiddle factors".
The program 12 shows a naive Matlab procedure.

   In reality, it would be necessary to call, rather than `fft2 (m)`, an FFT procedure on the lines (length q) then on the columns (length p) which exploits the decompositions of p and q.

Correction (exercise 24)

Program 12 Procedure `fft_gt`

```
function y = fft_gt (x, p)
n = length(x); q = n/p;
y = zeros(n, 1); m = zeros(p, q);
for i = 0:n-1
  m (mod(i, p) +1, mod(i, q) +1) = x(i+1);
end
m = fft2 (m);
for (s1 = 0:p-1) for (s2 = 0:q-1)
  y(mod(s1 * q + s2*p, n) +1) = m(s1+1, s2+1);
end ; end ;
```

1. The first part of the grouping does not generate "twiddle factors", so there is no need to decompose it. Using the fact that $(n_1, n_2) \mapsto n_1 + n_2 N/4$ is a bijection of the produced set $\{0, \dots, N/4-1\} \times \{0, \dots, 3\}$ on $\{0, \dots, N-1\}$, we can write

$$\hat{f}[4k+2j+1] = \sum_{n_1=0}^{N/4-1} \sum_{n_2=0}^{3} \omega_N^{-(n_1+n_2N/4)(4k+2d-1)} f\,[n_1+n_2N/4].$$

   To conclude with the proposed expression, it suffices to notice that

$$\omega_N^{-(n_1+n_2N/4)(4k+2d-1)} = \omega_{N/4}^{-kn_1} \omega_N^{-n_1(2d+1)} \omega_4^{-n_2(2j+1)}.$$

   The interior sums are DFTs of length 4, and they are trivial to compute since the complex roots used are $\{\pm 1, \pm \imath\}$.

2. We can perform groupings of frequencies by packets of 2^l (the case $l = 1$ corresponds to the classical frequency decimation, and $l = 2$ to question 1). This leads to the following decomposition of the TFD:

$$\hat{f}[2^l k + 2j + 1] = \sum_{n_1=0}^{N/2^l-1} \omega_{N/2^l}^{-kn_1} \omega_N^{-n_1(2j+1)} \sum_{n_2=0}^{2^l} f\left[n_1 + n_2 N/2^l\right] \omega_{2^l}^{-n_2(2j+1)},$$

   for $j = 0, \dots, 2^{l-1}-1$ and $k = 0, \dots, N/2^l - 1$. We can show that the optimal split-radix of size 2^l corresponds to that of question 1 (see[68]). This is because size 4 DFTs do not require any complex multiplication.

3. For a temporal decimation scheme, it is a question of grouping not the frequencies of the transformed vector, but the inputs of the vector to transform. This gives rise to the following decomposition equation:

$$\begin{aligned}\hat{f}\left[n_1 + n_2 \frac{N}{4}\right] =& (-1)^{n_2} \sum_{k=0}^{\frac{N}{2}-1} f[2k] \omega_{N/2}^{n_1 k} + \\ & \sum_{j=0}^{1} \omega_N^{n_1(2j+1)} \omega_4^{n_2(2j+1)} \sum_{k=0}^{\frac{N}{4}} f[4k+2j+1] \omega_{N/4}^{kn_1},\end{aligned}$$

   for $n_1 = 0, \dots, N/4$ and $n_2 = 0, \dots, 3$. The TFD of length N is thus decomposed into the sum of one TFD of length $N/2$ and two TFD of length $\frac{N}{4}$. The analysis of the optimality of this decomposition is identical to that of the frequency decimation version of question 1.

Correction (exercise 25)

1. We have $f = \sum_{j=0}^{p-1} f_j[\cdot - jM]$. By bilinearity of the convolution product, we therefore have $f \star g = \sum_{j=0}^{p-1} f_j \star g[\cdot - jM]$
2. We must therefore calculate the p convolution products $f_j \star g$. Since the two vectors have the size of M, this product can be calculated by FFT by adding only $M-1$ zeros. Assuming that the FFT algorithm requires $cM\log(M)$, the convolution calculation requires $2cM\log(M)+M$ operations, and this calculation is performed p times.
3. If N is not a multiple of M, zeros should be added to f to reach a size equal to the multiple of M just after N.

The Matlab `convol` procedure, program 13, sets up this method. Note that the FFT of the vector g (to which we have added $M-1$ zeros) is stored once and for all in the variable `fg`.

Program 13 Procedure `convol`

```
function y = convol(f, g)
N = length(f); M = length(g); p = N/M;
y = zeros(M + N-1,1);
fg = fft([g; zeros(M-1,1)]);
for j = 0:p-1
  fj = [f((1:M) + j * M); zeros(M-1,1)];
  sel = (j * M+1):  ((j + 2) * M-1); % the indices concerned
  y(sel) = y(sel) + ifft(fft(fj).*fg);
end
```

Correction (exercise 26)

1. We have, for $x = (x_0, \dots, x_{N-1})^\top$,

$$\Omega_N(Dx)[k] = (\Omega_N(x_{N-1}, x_0, \dots, x_{N-2})^\top)[k] = \sum_{i=0}^{N-1} x_{i-1}\omega_N^{-ki}$$
$$= \omega_N^{-k}\sum_{i=0}^{N-1} x_i\omega_N^{-ki} = \omega_N^{-k}(\Omega_N x)[k],$$

which is the requested result.

2. We use the decomposition $C = \sum_{i=0}^{N} c_i R^i$, hence

$$\Omega_N C \Omega_N^{-1} = \sum_{i=0}^{N} c_i \Omega_N R^i \Omega_N^{-1} = \sum_{i=0}^{N} c_i D^i = \Delta.$$

Note that if $y \in \mathbb{C}^N$, $\Delta y = (\Omega_N c)\cdot y$. So we have $\Delta\Omega_N x = (\Omega_N c)\cdot(\Omega_N x)$.

3. The fact that $Cx = c * x$ is immediately true. With the convolution theorem, we have $\mathcal{F}(Cx) = \mathcal{F}(c*x) = \hat{c}\cdot\hat{x}$. As $\hat{x} = \Omega_N x$, we get the same formula.

Correction (exercise 27) Using the Fourier inversion formula 31, we have

$$f_0[\eta k] = \frac{1}{P}\sum_{s=0}^{P-1}\hat{f}_0[s]\omega_P^{s\eta k}$$
$$= \frac{1}{N}\sum_{s=0}^{N_0}\hat{f}[s]\omega_N^{sk} + \frac{1}{N}\sum_{s=N_0+1}^{N-1}\hat{f}[s]\omega_N^{sk-P+N} = f[k].$$

Program 14 Procedure `interp_trigo`

```
function y = interp_trigo(x, eta)
N = length(x); N0 = (N-1)/2; P = N * eta;
f = fft(x);
f = eta * [f(1:N0+1); zeros(PN, 1); f(N0 + 2:N)];
y = real(ifft(f));
```

The program 14 implements the exposed interpolation technique.

Correction (exercise 28)

1. Using the trigonometric relation

$$\cos((k+1)\theta)+\cos((k-1)\theta)=\cos(k\theta)\cos(\theta), \quad \text{with} \quad \theta=\arccos(X),$$

we get the recurrence relation $T_{k+1}=2XT_k-T_{k-1}$. This relation shows that T_k is a polynomial with integer coefficients of degree k.
Since we have $T_N(x_j)=\cos\left(N(j+1/2)\frac{\pi}{N}\right)=0$, we have found the N roots of T_N which is of degree N.

2. As the $\{T_k\}_{k=0}^{N-1}$ have different degrees, they form a space free family of polynomials of degree less than $N-1$. Since they are N, it is a basis of this space.

3. The inversion formula can be checked by hand, but it is quite painful. It is better to come back to orthogonal vectors. We can indeed show quite easily that the vectors

$$v_k=\left\{\lambda_k\sqrt{\frac{2}{N}}\cos\left(\frac{k\pi}{N}\left(n+\frac{1}{2}\right)\right)\right\}_{k=0}^{N-1} \quad \text{with} \quad \lambda_k=\begin{cases}2^{-1/2} & \text{si} k=0\\ 1 & \text{otherwise}\end{cases}$$

form an orthonormal basis of $\mathbb{C}^N$. The article by Strang[62] offers a nice proof that uses the fact that v_k are eigenvectors of a certain symmetric matrix.
The program Matlab 15 implements the transform $\mathcal{C}_2$ via an FFT of size $4N$. The program 16 implements the transform $\mathcal{C}_3$, always via an FFT of size $4N$.

Program 15 Procedure `dct2`

```
function y = dct2(x)
n = length(x);
y(2:2:2*n, 1) = x;
y = [y; zeros(2*n, 1)];
y = real(fft(y)); y = y(1:n);
```

Program 16 Procedure `dct3`

```
function y = dct3(x)
n = length(x);
y = [x; zeros(3*n, 1)];
y = real(fft(y)); y = y(2:2:2*n) - x(1)/2;
```

4. The coefficients $\alpha=(\alpha_0,\ldots,\alpha_{N-1})^\top$ of the polynomial P_{N-1} verify

$$f[j]=\sum_{k=0}^{N-1}\alpha_k T_k(x_j)=\sum_{k=0}^{N-1}\alpha_k\cos\left(k(j+1/2)\frac{\pi}{N}\right)=\mathcal{C}_3(\alpha)+\frac{\alpha_0}{2}. \tag{A.5}$$

There is therefore a slight difficulty because of the term "parasite" $\frac{\alpha_0}{2}$. If we denote by 1 the constant function equal to 1, we have $\mathcal{C}_2(1) = N\delta_0$. We can therefore reverse the equality A.5:

$$\frac{2}{N}\mathcal{C}_2(f) = \frac{2}{N}\mathcal{C}_2(\mathcal{C}_3(\alpha)) + \frac{1}{N}\mathcal{C}_2(\alpha_0 1) = \alpha + \alpha_0\delta_0 = (2\alpha_0,\, \alpha_1, \ldots,\, \alpha_{N-1}).$$

The program 17 shows how, with Matlab we can use the procedure `dct3` to perform a Chebyshev interpolation. Here, we interpolate the function $f(x) = \frac{1}{\alpha^2+x^2}$ known in `n` points (the points x_k). The program draws the interpolated curve by evaluating it in `nn` evenly spaced points.

Program 17 Chebyshev interpolation

```
n = 16; nn = 200; alpha = 0.3;
x = cos (((0:n-1) +1/2)*pi/n)';
f = 1 ./ (alpha∧2 + x.∧2);
coef = 2/n * dct2 (f);
coef(1) = coef(1) * 1/2;
xx = (-1:2/(nn-1):1)';
ff = zeros(nn, 1);
for k = 0:n-1
  ff = ff + coef(k+1) * cos (k * acos (xx));
end
plot (x, f,'o', xx, ff);
```

Correction (exercise 29) We prove the required formula simply by n integrations by parts.
Note that the multiplication by $(\imath\xi)^n$ corresponds to a filter which amplifies the highs frequencies. This is quite logical, since the derivation causes a loss of regularity, like a filter amplifying the high frequencies (slower decrease of the transform, which can be compared to an amplification of the "noise").
The program Matlab 18 proposes a function which realizes a fractional derivative to the order `alpha`. The procedure uses the FFT algorithm, so for it to accurately approximate the derivative of the original function, the sampling must be fine enough, and $\hat{h}$ must be supported in $[-\pi,\, \pi]$ (otherwise, be careful with aliasing, because we no longer meet the assumptions of Shannon's theorem, exercise 21).

Program 18 Procedure `der_frac`

```
function y = der_frac (f, alpha)
n = length(f)/2;
fce = pi * [(0:n)/n, ((-n+1):-1)/n];
f = fft(f);
f = (-i * fce).∧alpha.*f; f = real(ifft(f));
```

Correction (exercise 30)

1. Like $(\Omega_N)^4 = N^2\,\mathrm{Id}_N$, the eigenvalues of Ω_N are $\{\pm\sqrt{N},\, \pm\imath\sqrt{N}\}$.

2. We have $\mathcal{F}^\alpha \circ \mathcal{F}^\beta = PD^\alpha P^* PD^\beta P^* = PD^{\alpha+\beta}P^* = \mathcal{F}^{\alpha+\beta}$.

3. We notice that $\Omega_N^2 = \Delta$ where Δ is the matrix such that $\Delta e_i = e_{-i}$ (signal inversion). This matrix corresponds to the inverted diagonal. The closer α (modulo 4) is to 2, the more this inverted diagonal is preponderant in $\mathcal{F}^\alpha$, and the closer α is to 0, the more the diagonal is dominant.

4. For $N > 4$, there exists an infinity of orthogonal bases of diagonalization of Ω_N (the eigenspaces are of dimension greater than 1). Each different base gives rise to an intermediate transform.

The Matlab 19 procedure computes an intermediate transform for a given value of α.

Correction (exercise 31)

Program 19 Procedure `tfd_interm`

```
function y = tfd_interm (x, alpha)
n = length(x);
f = (0:n-1)'* (0:n-1);
Omega = exp(-2i * f * pi/n);
[V, D] = eig (Omega,'nobalance');
y = V * D∧alpha * ctranspose (V) * x;
```

1. S is a symmetric *real* matrix, which diagonalizes in orthonormal basis.

2. We write $\omega =$. The simplest is to use the following decomposition of S:

$$\begin{pmatrix} -2 & 1 & 0 & \dots & 1 \\ 1 & -2 & 1 & \dots & 0 \\ 0 & 1 & -2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \dots & -2 \end{pmatrix} + \operatorname{diag}\left(2\left(\cos\left(k\frac{2\pi}{N}\right)-1\right) \Big\backslash\ k=0,\dots,N-1\right),$$

which we denote by $S = \Gamma + \Delta$. Γ is a circulating matrix(see exercise 26), the multiplication by this matrix corresponds to the convolution by $v = (-2, 1, 0, \dots, 0, 1)^\top$. As the Fourier transform of v is the diagonal of Δ, we have $\Omega_N\Gamma = \Delta\Omega_N$ (with the convolution theorem). Using the symmetry of S and Ω_N, it comes

$$S\Omega_N = (\Omega_N S)^\top = (\Delta\Omega_N)^\top + (\Omega_N\Delta)^\top = \Delta\Omega_N + \Omega_N\Delta = \Omega_N S.$$

3. We can find the proof of this classical property in [54]. Essentially, this is to use the fact that the eigenspaces of f are stable by g, which results from the fact that if φ and ψ commute, then $\ker(\varphi)$ is stable by ψ. This is exactly what we use to prove *Schur's lemma* 13.

4. A simple calculation shows that this operator is symmetric and orthogonal. For example em ple for $N = 5$, we have

$$P = \frac{1}{\sqrt{2}}\begin{pmatrix} \sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & 0 & -1 \end{pmatrix}.$$

To simplify the explanations, suppose that $N = 2p$. To see that PSP^{-1} is tridiagonal, we must use the canonical basis of $\mathbb{C}^N$, denoted by $\{\delta_0, \dots, \delta_{N-1}\}$, and consider the base $\mathcal{B} \overset{\text{def.}}{=} \{e_0 = \delta_0, e_1, \dots, e_p, f_1, \dots, f_p\}$. We noted $e_i = \frac{1}{\sqrt{2}}(\delta_i + \delta_{-i})$ and $f_i = \frac{1}{\sqrt{2}}(\delta_i + \delta_{-i})$. So we have

$$S(e_0) = S(\delta_0) = C_0\delta_0 + \delta_1 + \delta_{-1} = C_0 e_0 + e_1,$$
$$\forall 1 \le i < p, \quad S(e_i) = \frac{1}{\sqrt{2}}S(\delta_i) + \frac{1}{\sqrt{2}}S(\delta_{-i}) = \frac{1}{\sqrt{2}}(C_i + C_{Ni})e_i + e_{i+1} + e_{i-1},$$
$$\forall 1 \le i < p, \quad S(f_i) = \frac{1}{\sqrt{2}}S(\delta_i) - \frac{1}{\sqrt{2}}S(\delta_{-i}) = \frac{1}{\sqrt{2}}(C_i - C_{Ni})f_i + f_{i+1} + f_{i-1}.$$

We must pay attention that the indices are expressed modulo N, and that for $i = p$, the inequalities are valid on condition of taking the convention $e_{p+1} = e_p$ and $f_{p+1} = f_p$. So the operator S, expressed in the base $\mathcal{B}$ is tridiagonal, which amounts to saying that PSP^{-1} is tridiagonal.

5. The proof uses an eigenvalue separation procedure thanks to *suites of Sturm*. See [16].

6. This construction is totally intrinsic, it does not rely on an arbitrary choice of diagonalization vectors. To build a partial DFT, it is necessary to make a choice in the order of the eigenvectors. In [13] the number of sign changes is used.

The Matlab 20 procedure builds the eigenvector base exposed in this exercise, for a size N given as a parameter.

Program 20 Procedure `vect_propres_tfd`

```
function V = vect_propres_tfd (n)
x = (0:n-1)'* (0:n-1);
Omega = 1/sqrt(n) * exp(2i * x * pi/n);
d = 2*(cos (2*pi/n * (0:n-1)) - 2);
S = diag (d, 0) + diag (ones (n-1,1), 1) + diag (ones (n-1,1),-1);
S (1, n) = 1; S (n, 1) = 1;
[V, D] = eig (S);
```

Correction (exercise 32)

1. Let's only do one of the two calculations:

$$\mathcal{F}(k\top f)[t] = \sum_{s=0}^{n-1} \omega_n^{-st} f[sk] = \omega_n^{-kt} \sum_{r=0}^{n-1} \omega_n^{-rt} f[r] = (k\bot f)[t],$$

where we made the change of summation index $r = sk$.

2. If $\mathcal{B} = \{e_i\}_{i=0}^{n-1}$ is orthonormal for $\bot$, then $\mathcal{F}(\mathcal{B}) = \{\mathcal{F}(e_i)\}_{i=0}^{n-1}$ is orthonormal for $\top$. Of course, $\bot$ and $\top$ are interchangeable.

3. f is orthonormal for $\top$ if and only if, for all k

$$\frac{1}{n} f * \tilde{f}[k] = \frac{1}{n} \sum_{s=0}^{n-1} f[s]\overline{f[sk]} = \delta_0[k],$$

where we noted $\tilde{f}[s] = \overline{f[-s]}$. By taking the Fourier transform of $f * \tilde{f} = \delta_0$, and using the convolution theorem, we find $|\hat{f}|^2 = 1$, where we noted 1 the constant function equal to 1.

4. Obviously $|\hat{f}_0|^2 = 1$, so f_0 is orthonormal for $\top$.
If we want g to be orthonormal for $\bot$, then we apply the previous construction to $f = \hat{g}$, and we consider $g_0 = \mathcal{F}^{-1}(f_0)$.

5. We have $\langle \varphi,\, k\top g\rangle = \frac{1}{n}\sum_{s=0}^{n-1} g[sk]\varphi[s] = \frac{1}{n} f * \tilde{f}[k]$. Since $\mathcal{G}$ is expressed as a convolution, it can be calculated quickly by FFT.

The Matlab 21 procedure allows you to orthogonalize a given vector.

Program 21 Procedure `fft_orthog`

```
function y = fft_orthog (x)
y = fft(x);
if  (min (abs(y))==0) error ('The TFD of xs'' cancels.'); return ; end ;
y = y./abs(y); y = real(ifft(y));
```

A.4 Correction of the exercises of chapter 4

Correction (exercise 33) With the Fourier inversion formula, we have

$$f(t)=\frac{1}{2\pi}\int_{-A}^{A}\hat{f}(\omega)e^{-\imath\omega t}\mathrm{d}\omega.$$

By derivation under the integral sign, we show that f is of class $\mathcal{C}^\infty$.
If $f(t)=0$ for $t\in[c,\,d]$, we would have, at point $t_0=\frac{1}{2}(c+d)$,

$$f^{(n)}(t_0)=\frac{1}{2\pi}\int_{-A}^{A}(-\imath\omega)^n\hat{f}(\omega)e^{-\imath\omega t_0}\mathrm{d}\omega=0.$$

By expanding $t\mapsto\exp(-\imath(t-t_0))$ in the neighborhood of 0, we find e

$$f(t)=\frac{1}{2\pi}\sum_{n=0}^{+\infty}\frac{[-\imath(t-t_0)]^n}{n!}\int_{-A}^{A}\hat{f}(\omega)\omega^n e^{\imath\omega t_0}\mathrm{d}\omega=0,$$

which is absurd. The inversion between $\sum$ and $\int$ is justified by Fubini's theorem.

Correction (exercise 34)

1. We use the finite difference schemes $\frac{\partial u}{\partial t}(t,\,x)\approx\frac{1}{h}(u(t+h,\,x)-u(t+h,\,x))$ and $\frac{\partial^2 u}{\partial x^2}(t,\,x)\approx\frac{1}{d^2}(u(t,\,x+d)+u(t,\,xd)-2u(t,\,x))$, which leads to the stated equation. Of course, the vectors are considered cyclic.

2. We have $u^{n+1}=g*u^n$, with $g=\{1-2s,\,s,\,0,\ldots,\,0,\,s\}$. As the support of g is very small, there is no interest in using the FFT algorithm.

3. As the Fourier transform is an isometry, u^n remains bounded if and only if $\hat{u^n}$ remains bounded. As we have $\hat{u^n}=\mathcal{F}(g*\cdots*g*u^0)=(\hat{g})^n\cdot\hat{u^0}$, this condition is equivalent to $|\hat{g}|\leq 1$. We can calculate this Fourier transform, and we find

$$1-4s\leq\hat{g}[k]=1+2s\left(\cos\left(\frac{2\pi}{N}\right)-1\right)\leq 1$$

The stability condition is therefore written $-1\leq 1-4s$, that is to say $s\leq\frac{1}{2}$. This condition is called the *Courant-Friedrichs-Levy* or *CFL* condition. The Matlab 22 procedure calculates the solution u^t after `t` iterations, by specifying the initial condition (variable `x`) and the precision (variable `s`).

Program 22 Procedure `resolution_explicite`

```
function y = _explicit resolution (x, s, t)
n = length(x); y = x;
for i = 1:t
  for k = 1:n
    y1 (k) = s * y(mod(k-2, n) +1) + s * y(mod(k, n) +1) + (1-2*s) * y(k);
  end
  y = y1;
end
```

4. Using the convolution properties, we get the solution

$$\hat{u^{n+1}}[k]=\frac{1+(1-\theta)\hat{A}[k]}{1-\theta\hat{A}[k]}\hat{u^n}[k]=h\left(\frac{2k\pi}{N}\right)\cdot\hat{u^n},$$

that it is therefore possible to solve in Fourier. We notice that

$$\frac{1-4(1-\theta)s}{1+4\theta s} \leq h(\omega) = \frac{1+2s(1-\theta)(\cos(\omega)-1)}{1-2s\theta(\cos(\omega)-1)} \leq 1,$$

these inequalities being easy to find by a function study, and they are the best possible. The stability condition is therefore written $-1 \geq \frac{1-4(1-\theta)s}{1+4\theta s}$, which is equivalent to $s(1-2\theta) \leq \frac{1}{2}$. So, if $\theta \geq \frac{1}{4}$, the scheme is still stable. If $\theta < \frac{1}{4}$, it is necessary that $s \leq \frac{1}{2(1-2\theta)}$. We find the CFL condition for $\theta = 0$. The Matlab 23 program solves the heat equation by this implicit method. Compared to the 22 procedure, it takes an additional argument, `theta`.

Program 23 Procedure `resolution_implicit`

```
function y = implicit_resolution (x, s, theta, t)
n = length(x); y = x;
A = zeros(n, 1); A(1) = -2*s; A(2) = s; A(n) = s;
fA = fft(A); y = fft(x);
mult = (ones (n, 1) + (1-theta) * fA) ./ (ones (n, 1) -theta * fA);
for (i = 1:t) y = y.*mult; end ;
y = real(ifft(y));
```

5. The 2D equations are the same as long as you consider the filter A such that only the inputs $A[0, 0] = -4s$, $A[\pm 1, 0] = s$ and $A[0, \pm 1] = s$ are non-zero. The stability conditions are the same. The Matlab 24 program solves the heat equation for a given 2D function.

Program 24 Procedure `resolution_implicit_2d`

```
function y = implicit_resolution_2d (x, s, theta, t)
n = length(x); y = x;
A = zeros(n, n); A(1.1) = - 4 * s; A(2.1) = s; A(1.2) = s; A(n, 1) = s; A(1, n) = s;
fA = fft2 (A); y = fft2 (x);
mult = (ones (n, n) + (1-theta) * fA) ./ (ones (n, n) -theta * fA);
for (i = 1:t) y = y.*mult; end ;
y = real(ifft2 (y));
```

Correction (exercise 35)

1. We have

$$u(t, x) = \sum_{n\in\mathbb{Z}} \hat{f}(n) e^{-2\pi^2 n^2 t} e^{2\imath\pi n x} = \int_0^1 \left\{ \sum_{n\in\mathbb{Z}} e^{-2\pi^2 n^2 t} e_n(xy) \right\} f(y)\mathrm{d}y = p_t * f(x),$$

where the convolution is that of $L^1([0, 1])$. We denote by $p_t(x) = \sum_{n\in\mathbb{Z}} e^{-2\pi^2 n^2 t} e_n(x)$.
By normal convergence of the derivatives for $t > 0$, we see that $u \in \mathcal{C}^\infty(S^1 \times \mathbb{R}_*^+)$. Moreover, by derivation under the integral sign, we see that for $t > 0$, u satisfies the partial differential equation of heat.

2. If $f \in \mathcal{C}^2(S^1)$, we have $|\hat{f}(n)| = O(\frac{1}{n^2})$ (see [72] for example), and therefore by dominated convergence, it comes

$$\|u(t, \cdot) - f\|_\infty \leq \sum_{n\in\mathbb{Z}} |\hat{f}(n)| |1 - e^{-2\pi^2 n^2 t}| \overset{n\to+\infty}{\longrightarrow} 0.$$

3. We assume $u(t_0, x_0) < 0$. On $[0, t_0] \times S^1$ which is compact, v reaches its minimum α in (t_1, x_1). On the x axis, since x_1 is an interior point , we have $\frac{\partial u}{\partial x}(t_1, x_1) = 0$ as well as $\frac{\partial^2 u}{\partial x^2}(t_1, x_1) \geq 0$. On the t axis, we can possibly have $t_1 = t_0$, but in all cases, we have $\frac{\partial v}{\partial t}(t_1, x_1) \leq 0$. So we have

$$0 \geq \frac{\partial v}{\partial t}(t_1, x_1) = \beta v(x_1, t_1) + e^{\beta t} \frac{\partial u}{\partial t}(t_1, x_1) \geq \alpha\beta + \frac{1}{2} e^{\beta t} \frac{\partial^2 u}{\partial x^2}(x_1, t_1) \geq \alpha\beta,$$

which is absurd if we take β such that $\alpha\beta > 0$.
If u and $\tilde{u}$ are solution of the same heat problem, then $u - \tilde{u}$ is solution of the heat equation with the initial condition $f = 0$. By the principle of the maximum, it comes $\|u(\cdot, t) - \tilde{u}(\cdot, t)\|_\infty \leq \|f\|_\infty = 0$, so $u = \tilde{u}$.

4. If we didn't have $p_t \geq 0$, we could find a neighborhood $]a, b[$ on which $p_t < 0$. We denote by $x_0 = \frac{1}{2}(a+b)$, and $ba = 2m$. By choosing f regular, with support in $[-m, m]$, and $f > 0$ on $]-m, m[$, we have
$$f * p_t(x_0) = \int_a^b p_t(y) f(x_0 - y)\mathrm{d}y < 0,$$
which is absurd from the previous question. So it comes
$$\|u(\cdot, t)\|_\infty = \|p_t * f\|_\infty \leq \|f\|_\infty \int_0^1 p_t = \|f\|_\infty,$$
since $\int_0^1 p_t = \sum e^{-2\pi^2 n^2 t} \int_0^1 e_n = 1$.

5. We have
$$\|u(t, \cdot) - f\|_\infty \leq \|p_t * f_n - p_t * f\|_\infty + \|p_t * f_n - f_n\|_\infty + \|f_n - f\|_\infty.$$
Let $\epsilon > 0$. We fix N such that $\|f_n - f\|_\infty \leq \epsilon/4$. We also have $\|p_t * f_n - p_t * f\|_\infty \leq \|f_n - f\|_\infty \leq \epsilon/4$. Then, as f is of class $\mathcal{C}^2$, we fix t_0 such that if $t \leq t_0$, we have $\|p_t * f_n - f_n\|_\infty \leq \epsilon/\ 2$.

Correction (exercise 36) The equations are the same, except that we must take for Φ the 3D filter such that $\Phi[\pm 1, 0, 0] = \Phi[0, \pm 1\, 0] = \Phi[0, 0\, \pm 1] = 1$, $\Phi[0, 0, 0] = 6$, and the other entries are zero. The only hard thing to code is the function that makes the data matrix antisymmetric. This is done by the Matlab 25 procedure.

Program 25 Procedure `antisymetrise`

```
function ff = antisymmetry(f)
n = length(f) +1; ff = zeros(2*n, 2*n, 2*n);
for (x = 1:2*n) for (y = 1:2*n) for (z = 1:2*n)
  if mod(x-1, n)==0 | mod(y-1, n)==0 | mod(z-1, n)==0
    ff(x, y, z) = 0;
  else
    sign = 1; nx = x; ny = y; nz = z;
    if  (x>n) sign = -sign; nx = 2*n-x + 2; end
    if  (y>n) sign = -sign; ny = 2*n-y + 2; end
    if  (z>n) sign = -sign; nz = 2*n-z + 2; end
    ff(x, y, z) = sign * f(nx-1, ny-1, nz-1);
  end ;
end ; end ; end ;
```

Correction (exercise 37)

1. The finite difference equation, is written as the sum of two acyclic convolutions, one on the rows and the other on the columns. Multiplication on the left by T_{N-1} performs this convolution on the lines, and the multiplication on the right performs that on the rows.

2. By subtracting the values at the edges (inputs $U_{i,j}$ with $i, j \in \{1, N-1\}$), we check that the acyclic convolutions mentioned in previous question are written as matrix products $T_{N-1}\tilde{U}$ (columns) and $\tilde{U}T_{N-1}$ (rows).

3. Using trigonometric identities, we have, noting $\omega = \frac{\pi}{N}$, for $2 \leq i \leq N-2$,

$$h^2(T_{N-1}V_j)[i] = \sin((i-1)j\omega) - 2\sin(ij\omega) + \sin((i+1)j\omega)$$
$$= -4\sin^2\left(\frac{j\pi}{2N}\right)V_j[i].$$

We check that this result is still valid for $i = 1$ and $i = N-1$. We can therefore diagonalize T_{N-1}, $V^{-1}T_{N-1}V = D$, with $D = \left\{-4\sin^2\left(\frac{j\pi}{2N}\right)\right\}_{1\leq j\leq N-1}$.
By multiplying the equation (4.15), on the left by V^{-1} and on the right by V, we obtain the required equation.

4. As T_{N-1} is symmetric, its eigenvectors are orthogonal and therefore V is an orthogonal matrix. We denote by $\mathcal{S} : \mathbb{C}^{N-1} \to \mathbb{C}^{N-1}$ the sine transform, defined by

$$\mathcal{S}(f)[i] = (Vf)[i] = \sum_{j=1}^{N-1} f[j]\sin\left(\frac{ij\pi}{N}\right).$$

If we write $f_0 = \{0,\, f[0], \ldots,\, f[N-1],\, 0, \ldots, 0\} \in \mathbb{C}^{2N}$, then, for $i = 1, \ldots, N-1$, we have $\mathcal{S}(f)[i] = \mathcal{I}(\hat{f}_0[i])$.

5. We can also consider $\tilde{f} = \{0,\, f[1], \ldots,\, f[N-1],\, 0,\, -f[N-1], \ldots, -f[1]\}$ the signal balanced by imparity. We then have $\mathcal{S}(f)[k] = -2\imath\hat{f}_0[k]$. As V is orthogonal and symmetric, we have $\mathcal{S}^{-1} = \frac{2}{N}\mathcal{S}$. The Matlab 26 procedure performs the 1D sine transform.

Program 26 Procedure `transformed_sinus`

```
function y = transform_sinus (x)
n = length(x);
x = [0; x; 0; -x(n:  -1:1)]; x = fft(x);
y = real(x(2:n+1)/(- 2i));
```

Similarly, if $F \in \mathbb{C}^{(N-1)\times(N-1)}$, we denote by $\tilde{F}$ the corresponding odd function. This time we have $VFV^{-1}[k,\, l] = -4\mathcal{F}(\tilde{F})[k,\, l]$. This method is in fact identical to the one exposed in Paragraph 4.4.3. The Matlab 27 procedure performs the 2D sine transform, but only uses the 1D transform algorithm (on the rows then the columns).

Program 27 Procedure `transformed_sinus_2d`

```
function y = transform_sinus_2d (x)
y = zeros(size (x)); n = length(x);
for (i = 1:n) y(i,:)  = transform_sinus (x(i,:)')'; end ;
for (j = 1:n) y(:,j) = transform_sinus (y(:,j)); end ;
```

Correction (exercise 38) The Matlab 28 procedure calculates a 2D Gaussian filter of size (parameter `n`) and variance (parameter `s`) data. The 29 program applies a Gaussian filter to an image loaded from a file, then draws the image on the screen.

Program 28 Procedure `calculation_filter`

```
function f = calculation_filter (n, s)
x = -1:2/(n-1):1;
[X,Y] = meshgrid (x, x);
f = exp(- (X.^2 + Y.^2)/(2*s));
f = f/sum(sum(f));
```

Correction (exercise 39)

Program 29 Applying a Gaussian filter

```
[im, cm] = imread('put file name here');
n = length(im); s = 0.01;
f = calculation_filter (n, s);
y = filter2 (f, im);
image(y); colormap(cm);
axis off; axis image;
```

1. $d(f,\, g)[u,\, v]$ measure the similarity between g and a portion of the image f whose lower left corner is located at $(u,\, v)$. We have
$$d(f,\, g)[u,\, v] = \mathrm{Corr}(f,\, g)[u,\, v] + P_{u,v}(f) + \|g\|_2^2.$$
Since $\|g\|_2^2$ is constant, minimizing $d(f,\, g)$ amounts to minimizing $\mathrm{Corr}(f,\, g)[u,\, v]$ if $P_{u,v}(f)$ varies little.

2. $\mathrm{Corr}(f,\, g)$ is the acyclic convolution product of f with $\tilde{g}[x,\, y] = g[-x,\, -y]$.

3. We denote by $f_0[x,\, y] = f[x,\, y] - \tilde{f}_{u,v}$ (which we assume null outside of $D(u,\, v)$), and $g_0[x,\, y] = g[x,\, y] - \tilde{g}$. We have
$$\overline{\mathrm{Corr}}(f,\, g)[u,\, v] = \frac{\langle f_0,\, g_0\rangle}{\|f_0\|_2\|g_0\|_2}.$$
Thus, $-1 \leq \overline{\mathrm{Corr}}(f,\, g) \leq 1$, and $\overline{\mathrm{Corr}}(f,\, g) = 1$ if and only if f_0 and g_0 are equal. This indeed gives a notion of resemblance between g and a portion of f, which moreover is insensitive to the affine modifications of the intensity of the two images. The problem is that this quantity is not written as a convolution.

4. The numerator simplifies to $\sum_{(x,\, y)} f[x,\, y]g_0[xu,\, yv]$, since g_0 has mean ne zero . We thus obtain a convolution. The recurrence relation is seen by drawing, the s_k being sums on overlapping squares. The Matlab 30 procedure uses this recurrence to fill, by decreasing indices, s_k. Its complexity is about $6N^2$ operations.

Program 30 Procedure `sum_sliding`

```
function y = sliding_sum(x, P, k)
N = length(x); y = zeros(N, N);
for (u = N: -1:1) for (v = N: -1:1)
  if  (u <N) y(u,v) = y(u,v) + y(u+1, v); end ;
  if  (v <N) y(u,v) = y(u,v) + y(u, v+1); end ;
  if  (u <N & & v <N) y(u,v) = y(u,v) -y(u+1, v+1); end ;
  y(u,v) = y(u,v) + x(u,v) ∧ k;
  if  (u+P<=N) y(u,v) = y(u,v) -x(u + P, v) ∧ k; end ;
  if  (v+P<=N) y(u,v) = y(u,v) -x(u, v + P) ∧ k; end ;
  if  (u+P<=N & & v + P<=N) y(u,v) = y(u,v) + x(u+P, v+P ) ∧ k; end ;
end ; end ;
```

We have $\|f_0\|_2^2 = s_2(u,\, v) - \frac{1}{P^2}s_1(u,\, v)^2$, which can be calculated with are slippery, so very quickly. In addition, $\|g\|_2$ is calculated once and for all. The 31 procedure computes $\overline{\mathrm{Corr}}(f,\, g)$ using this fast algorithm.

Correction (exercise 40)

1. We have
$$\mathcal{F}(T_v(f))[k,\, l] = \omega_N^{-(kv_1+lv_2)}\mathcal{F}(f)[k,\, l] \qquad \text{where} \quad \omega_n = e^{\frac{2\imath\pi}{N}}.$$
The Matlab 32 procedure uses the FFT algorithm to translate an image.

Program 31 Procedure `correlation_normalisee`

```
function y = correlation_normalized (f, g)
N = length(f); P = length(g);
% renormalization
g = g - mean (mean (g)); g = g/norm (g);
% calculating the numerator
ff = zeros(N+P-1, N+P-1); ff(1:N, 1:N) = f;
gg = zeros(N+P-1, N+P-1); gg (1:P, 1:P) = g;
fg = real(ifft2 (fft2 (ff).*conj (fft2 (gg))));
fg = fg (1:N, 1:N);
% calculating the denominator
s1 = sliding_sum(f, P, 1);
s2 = sliding_sum(f, P, 2);
denom = sqrt(s2-1/P ∧ 2*(s1.  ∧ 2));
y = fg./denom;
```

Program 32 Procedure `fft_translation`

```
function y = fft_translation (x, v)
n = length(x);
[s, t] = meshgrid ([0:(n/2-1), - n/2:-1]);
mult = exp(-2i * pi/n * (s * v (1) + t * v (2)));
y = fft2 (x).*mult;
y = real(ifft2 (y));
```

The 33 procedure applies the operator $S_\lambda^{(x)}$ by performing a translation on each line of the image. Note that the indices of the image inputs are taken between $-N/2$ and $N/2-1$, in order to apply $S_\lambda^{(x)}$ to turn an image around its center. We leave it to the reader to write the `fft_transvec_y` procedure himself.

Program 33 Procedure `fft_transvec_x`

```
function y = fft_transvec_x(x, lambda)
n = length(x);
for k = 1:n
  v = x(:,k); trans = lambda * (kn/2-1);
  mult = exp(-2i * pi/n * ([0:(n/2-1), - n/2:-1]'* trans));
  v = fft(v).*mult; y(:,k) = real(ifft(v));
end
```

2. We have
$$\begin{pmatrix}\cos(\theta) & -\sin(\theta)\\ \sin(\theta) & \sin(\theta)\end{pmatrix} = \begin{pmatrix}1 & -\tan(\theta/2)\\ 0 & 1\end{pmatrix}\begin{pmatrix}1 & 0\\ \sin(\theta) & 1\end{pmatrix}\begin{pmatrix}1 & -\tan(\theta/2)\\ 0 & 1\end{pmatrix}.$$
If we assume that an image corresponds to a function from $\mathbb{R}^2$ in $\mathbb{R}$ discretized at points $0, \ldots, N-1$, then the discrete rotation operator is written $R_\theta = S_{\lambda_1}^{(x)} S_{\lambda_2}^{(y)} S_{\lambda_3}^{(x)}$.

3. To rotate an image around its center, it suffices to use, in the algorithms for calculating $S_\lambda^{(x)}$ and $S_\lambda^{(y)}$, points discretization $-N/2, \ldots, N/2-1$ (this is what we did for `fft_transvec_x`). The procedure 34 allows to perform a discrete rotation.

   This algorithm is very fast, since it requires $O(N^2\log(N))$ operations. Moreover, it is bijective, and satisfies $R_{\theta_1}R_{\theta_2} = R_{\theta_1+\theta_2}$. In a way, passing through the DFT is the "natural" way of discretizing a continuous rotation. On the other hand, since the image is considered as a continuous function, the image is found "cut into pieces" after the rotation. To prevent this, you have to add zeros all around. This is what the 35 program does, which made it possible to produce the figure 4.15. We can notice that this program adds 255 around the starting image. This is to get a white edge rather than a black one.

Program 34 Procedure `fft_rotation`

```
function y = fft_rotation (x, theta)
y = fft_transvec_x(x, -tan(theta/2));
y = fft_transvec_y(y, sin(theta));
y = fft_transvec_x(y, -tan(theta/2));
```

Program 35 Rotate an image

```
[im, cm] = imread('put file name here');
n = length(im); p = ceil (n/2*(sqrt(2) -1));
x = ones (n + 2*p, n + 2*p) * 255;
x((p+1):( n + p), (p+1):( n + p)) = im;
nbr = 6; rot = pi/(4 * (nbr-1));
for r = 0:nbr-1
  y = fft_rotation(x, r * rot);
  subplot(1, nbr, r+1);
  image(y); colormap(cm);
  axis off; axis image;
end
```

Correction (exercise 41)

1. The Matlab 36 procedure does this.

Program 36 Procedure `filter_pass_bas`

```
function f = filter_pass_bas(N)
f = (0:N-1)'; f = (f<=N/4) | (f>=3 * N/4);
f = real(ifft(f));
```

2. The Matlab 37 program draws the Fourier transform of a filter, adding zeros.

   We observe oscillations (Gibbs phenomenon), because we try to approach a discontinuous function (the ideal filter is the indicator function of $[-\pi/2, \pi/2]$) by a trigonometric polynomial.

3. We will replace the sudden change from 0 to 1 in the filter of question 1 by a smooth (sinusoidal) progression of length $\epsilon N/2$. The 38 procedure does this.

Correction (exercise 42)

1. At the k^{th} iteration of the process, we write m_k the minimum number of candies a child has, n_k the maximum number, and s_k the number of occurrences of m_k. It is easy to see that $m_{k+1} \geq m_k$ and $n_{k+1} \leq n_k$. Moreover, we also see that if $s_k > 1$, then $s_{k+1} < s_k$, and if $s_k = 1$, then $m_{k+1} > m_k$. Since s_k is bounded by n, every n operations, m_k increases by at least 1, and therefore $|m_k - n_k|$ is equal to 0 after a finite number of operations.

2. We can assimilate the distribution of candies to a probability distribution (by renormalizing it so that its sum is equal to 1), and we are therefore in the situation of the exercise 10. If we write $p^{(k)}$ the distribution of the candies after k iterations, we have $p^{(k)} = v * \cdots * v * p^{(0)}$, with $v = \{1/2, 1/2, 0, \ldots, 0\}$. By taking the Fourier transform, we get $\hat{p^{(k)}} = (\hat{v})^k \cdot \hat{p^{(0)}}$. We calculate $\hat{v}[k] = \frac{1}{2}(1 + e^{-\frac{2\imath k\pi}{N}})$, and we see that for $k \neq 0$, $|\hat{v}[k]| < 1$. So $p^{(k)} \longrightarrow \{m, \ldots, m\}$, where $m = \hat{p^{(0)}}[0]$ (the average number of candies).

3. In the case where each child distributes half on the left and half on the right, there is no convergence, because we can have $m_{k+1} = m_k$. The situation is similar to that encountered in the 10 exercise for $v = \{0, 1/2, 0, \ldots, 0, 1/2\}$ when n is even.

Program 37 Drawing the continuous Fourier transform by zero padding

```
N = 64; P = 1024;
f = filter_pass_bas (N);
ff = [f(1:N/2); zeros(PN, 1); f((N/2+1):N)];
ff = real(fft(ff));
plot (ff); axis tight;
```

Program 38 Procedure `parametrable_filter`

```
function f = parametrable_filter (N, eps)
P1 = floor (eps * N/4); P = 2*P1+1; % P must be odd
t = [1]; if  (P ~ = 1) t = (cos ((0:P-1)'* pi/(P-1))+1)/2; end ;
f = [ones (N/4-P1,1); t; zeros(N/2-P, 1); t (P: -1:1); ones (N/4-P1-1, 1)];
f = real(ifft(f));
```

Correction (exercise 43)

1. We must take $R_0 = P_0Q_0$, $R_1 = P_1Q_0 + P_0Q_1$ and $R_2 = P_1Q_1$.

2. We can perform the following clever calculation: $R_1 = (P_0 + P_1)(Q_0 + Q_1) - R_0 - R_2$.

3. In all, there are $\log_2(n)$ nested recursive calls. At each step, there are 3 recursive calls, so for $k = 0, \ldots, \log_2(n)$, there are 3^k calls in total. The cost of the additions at each step is $c \times 2^{\log_2(n)-k}$ (since the polynomials are of degree $2^{\log_2(n)-k}$). In total, the number of operations is

$$\sum_{k=0}^{\log_2(n)} 3^k c 2^{\log_2(n)-k} = O(n(3/2)^{\log_2(n)}) = O(n^{\log_2(3)}).$$

The Matlab 39 procedure calculates the product of two polynomials of the same size (represented as vectors). It uses the procedure 40 which adds two polynomials of different degrees.

Program 39 Procedure `karatsuba`

```
function r = karatsuba(p, q)
n = length(p) -1;
if  (n==0) r = p * q; return ; end ;
k = floor ((n+1)/2);
p0 = p (1:k); p1 = p ((k+1):( n+1));
q0 = q (1:k); q1 = q ((k+1):( n+1));
r0 = karatsuba(p0, q0); r2 = karatsuba (p1, q1);
r1 = karatsuba(add (p0, p1), add(q0, q1));
r1 = add (r1, -r0); r1 = add(r1, -r2);
r = add (r0, [zeros(k, 1); r1]);
r = add (r, [zeros(2*k, 1); r2]);
```

Correction (exercise 44)

1. The equation $v(k) = u_d[k]$ is written $u_d = a * \psi_d$. Using the convolution theorem for Fourier series (see [72]), we get $\hat{u_d} = \hat{a} \cdot \hat{\psi}_d$. By replacing ψ by the function φ such that

$$\varphi(x) = \sum_{k\in\mathbb{Z}} \Phi_d(k)\psi(xk) \quad \text{with} \quad \hat{\Phi_d} = \frac{1}{\hat{\psi}_d},$$

we come back to a problem of direct interpolation. Even though ψ is compactly supported, φ is generally not compactly supported.

Program 40 Procedure add

```
function r = add (p, q)
m = length(p); n = length(q);
if  (m>=n) r = p + [q; zeros(mn, 1)];
else r = q + [p; zeros(nm, 1)]; end ;
```

2. β^n is supported in $[(-n+1)/2,\ (n+1)/2]$. β^n allows to define a direct interpolation scheme if and only if $\beta^n(k)=\delta_0(k)$, and we check that this is the case only for $n=0$ (constant piecewise interpolation) or $n=1$ (linear interpolation).

3. With the convolution theorem of the Fourier transform, it comes

$$\hat{\beta^n}(\xi)=\left(\frac{\sin(\xi/2)}{\xi/2}\right)^{n+1}.$$

We have $\beta^n_d=\beta^n\cdot\Pi_1$, from where, with the result of the exercise 20, it comes

$$\hat{\beta^n_d}(\xi)=2\pi\hat{\beta^n}*\Pi_{2\pi}(\xi)=2\pi\sum_{p\in\mathbb{Z}}(-1)^{p(n+1)}\left(\frac{\sin(\xi/2)}{\pi p+\xi/2}\right)^{n+1}.$$

As $\hat{\beta^n_d}(\xi)$ is a 2π -periodic function, it suffices to study it on $[0,\ 2\pi[$. If n is odd, all the terms of the sum are positive, and $\hat{\beta^n_d}>0$. If n is even, it's a bit more complicated. The term for $p=0$ of the sum is strictly positive. For the rest, it is necessary to group together the terms corresponding to p and to $-p$ and use the criterion of alternating series.

4. We have $\beta^n_{card}=\Phi^n_d*\beta^n$. In Fourier, we get

$$\hat{\beta^n_{card}}(\xi)=\frac{\left(\frac{\sin(\xi/2)}{\xi/2}\right)^{n+1}}{2\pi\sum_{p\in\mathbb{Z}}\left(\frac{\sin(\xi/2)}{\pi p+\xi/2}\right)^{n+1}}=\frac{1/(2\pi)}{1+\Gamma_n(\xi)},$$

with

$$\Gamma_n(\xi)=\sum_{p=1}^{+\infty}\left[\left(\frac{2\pi p}{\xi}+1\right)^{-n-1}+\left(\frac{2\pi p}{\xi}-1\right)^{n-1}\right].$$

Γ_n has infinite support. As the Fourier transform is an isometry, we want to show the convergence in $L^2(\mathbb{R})$:

$$\hat{\beta^n_{card}}\overset{n\to\infty}{\longrightarrow}\frac{1}{2\pi}1_{[-\pi,\,\pi]}.$$

By parity, we will evaluate only the integrals

$$E_n=\int_0^{\pi}\left(\frac{1}{1+\Gamma_n(\xi)}-1\right)^2\mathrm{d}\xi\qquad\text{and}\qquad F_n=\int_{\pi}^{+\infty}\left(\frac{1}{1+\Gamma_n(\xi)}\right)^2\mathrm{d}\xi.$$

For the first, we have $E_n\le\int_0^{\pi}\Gamma_n^2$, and we use the fact that if $\xi\in[0,\ \pi]$, then we have $t=2\pi/\xi>2$, hence the markup

$$\Gamma_n(\xi)\le 2\left(\frac{2\pi}{\xi}-1\right)^{-n-1}\sum_{p=1}^{+\infty}p^{-n-1}\le 4(t-1)^{-n-1}.$$

By changing the variable $\xi\to t$, we obtain

$$E_n\le\int_2^{+\infty}\left[4(t-1)^{-n-1}\frac{2\pi}{t^2}\right]^2\mathrm{d}t=O(\frac{1}{n}).$$

For the second integral, we use the lower bound $\Gamma_n(\xi) \geq (\xi/\pi)^{n+1}$, hence

$$F_n \leq \int_{\pi}^{+\infty} \left[\frac{1}{1+(\xi/\pi)^{n+1}} \right]^2 \mathrm{d}\xi = O(\frac{1}{n}).$$

In the end, we see that the trigonometric interpolation (that is to say by zero padding, also called Shannon interpolation, as shown in the exercise 21) can be seen as an interpolation spline of infinite degree.

5. We have $c = \Phi_d * u_d$, which is written in Fourier $\hat{c} = \hat{u_d}/\hat{\beta}_d^n$. If we approach the continuous Fourier transform by a discrete Fourier transform, and we calculate $\hat{c}$ by FFT, we will truncate the considered functions, which is very bad (Gibbs oscillations)

A.5 Correction of the exercises of chapter 5

Correction (exercise 45) We have $\mathcal{H}^2 = N\,\mathrm{Id}$, so the eigenvalues of $\mathcal{H}$ are included in $\{\pm\sqrt{N}\}$. For $f \in \mathbb{R}^N$, we define

$$\mathcal{U}_+(f) \overset{\text{def.}}{=} \sqrt{N}f + \mathcal{H}(f) \quad \text{and} \quad \mathcal{U}_+(f) \overset{\text{def.}}{=} \sqrt{N}f - \mathcal{H}(f).$$

We check that they are indeed eigenvectors of $\mathcal{H}$ associated with the eigenvalues $\sqrt{N}$ and $-\sqrt{N}$. The procedure Matlab 41 calculates, using these eigenvectors, an intermediate Hartley transform. We will pay attention to the fact that it has a value in $\mathbb{C}$.

Program 41 Procedure `fht_interm`

```
function y = fht_interm (x, lambda)
N = length(x);
u1 = sqrt(N) * x + fht(x); u2 = sqrt(N) * x-fht(x);
y = (sqrt(N)∧lambda) * (u1 + (-1)∧lambda * u2);
```

Correction (exercise 46) We denote by $\alpha = \frac{2\pi}{N}$, and $v_k^\lambda \in \mathbb{R}^N$ such that $v_k^\lambda[l] = \cos(\alpha kl + \lambda)$. The generalized Hartley transform is written $\mathcal{H}_\lambda(f)[k] = \langle f,\, v_k^\lambda \rangle$. To define an inverse transform, we look for a family of vectors biorthogonal to the v_k^λ in the form $\{v_k^{\lambda'}\}_{k=0}^{N-1}$. So we want $\langle v_k^\lambda,\, v_{k'}^{\lambda'} \rangle = \gamma \delta_k^{k'}$, where $\gamma \in \mathbb{C}$ is a constant. By expanding the dot product, we obtain an expression similar to that of the equation (5.3). To make the first two terms simpler, we impose $e^{\imath(\lambda+\lambda')} = -e^{-\imath(\lambda+\lambda')}$, for example $\lambda + \lambda' = \pi/\ 2$. We then get $\langle v_k^\lambda,\, v_{k'}^{\lambda'} \rangle = \frac{1}{2}\sin(2\lambda)\delta_k^{k'}$, and so in the end it comes to $\mathcal{H}^\lambda \circ \mathcal{H}^{\lambda'} = \frac{1}{2}\sin(2\lambda)\,\mathrm{Id}$.

Correction (exercise 47) Let $f \in \mathbb{R}^N$ with $N = 2N_0+1$ and $g = \mathcal{H}(f)$. We note $\tilde{g} = \{g[0], \ldots,\, g[N_0],\, 0, \ldots,\, 0,\, g[N_0+1],\, \ldots,\, g[N-1]\}$. This allows to calculate, by FHT, the quantities $\mathcal{H}(\tilde{g})[n] = F(nN/P)$, where we noted $F(x) = \frac{1}{N}\sum_{k=0}^{N-1} g[k]\,\mathrm{cas}(2\pi kx/N)$. We can therefore evaluate the function F with the precision we want. With the inversion formula, proposition 44, F interpolates f at points $0, \ldots,\, N-1$. By expressing the functions $x \mapsto \mathrm{cas}(2\pi kx)$ using the complex exponentials $x \mapsto e^{\frac{2\imath kx\pi}{N}}$, we see that F is a trigonometric polynomial of degree at most $N-1$. The trigonometric interpolation of the exercise 27 also uses a trigonometric polynomial of degree at most $N-1$. As it passes a single trigonometric polynomial of degree at most N by N distinct points, we deduce that these two interpolations are the same.

Correction (exercise 48) We can write the Hartley transform 1D as $\mathcal{H}(f)[k] = \langle f,\, \varphi_k^{(N_1)} \rangle$ with $\varphi_k^{(N_1)}[n] = \mathrm{cas}(nk2\pi/N_1)$. The proposition 44 tells us that $\langle \varphi_k^{(N_1)},\, \varphi_{k'}^{(N_1)} \rangle = N_1 \delta_k^{k'}$. Hartley 2D functions are tensor products $\varphi_{(k_1,\,k_2)}^{(N_1,N_2)}[n_1,\, n_2] = \varphi_{k_1}^{(N_1)}(n_1)\varphi_{k_2}^{(N_2)}(n_2)$. They therefore still form an orthogonal system, since

$$\langle \varphi_{(k_1,\,k_2)}^{(N_1,N_2)},\, \varphi_{(k_1',\,k_2')}^{(N_1,N_2)} \rangle = \langle \varphi_{k_1}^{(N_1)},\, \varphi_{k_1'}^{(N_1)} \rangle \langle \varphi_{k_2}^{(N_2)},\, \varphi_{k_2'}^{(N_2)} \rangle = N_1 N_2 \delta_{k_1}^{k_1'} \delta_{k_2}^{k_2'},$$

hence the proposed inversion formula. The calculation algorithm consists in applying the FHT procedure on each row of the matrix $f \in \mathbb{R}^{N_1 \times N_2}$, then on each column. This is what the 42 procedure does.

Program 42 Procedure `fht2d`

```
function y = fht2d (x)
n = length(x); y = zeros(n, n);
for (i = 1:n) y(i,:)  = fht(x(i,:)')'; end ;
for (j = 1:n) y(:,j) = fht(y(:,j)); end ;
```

Correction (exercise 49) Let p be a prime number such that $4|p-1$. We denote by ζ a generator of $\mathbb{F}_p^*$, and $\gamma = \zeta^{\frac{p-1}{4}}$. In the equation (5.1), we replace $\text{cas}(2kl\pi/N)$ by $\frac{1}{2}(\zeta^{kl} + \zeta^{-kl}) + \frac{1}{2\gamma}(\zeta^{kl} - \zeta^{-kl})$. The proof of the proposition 44 is still valid if we replace ω by ζ and $\imath$ by γ. The program Maple 43 performs a Hartley transform on a finite field using a field extension, as explained in Paragraph 6.1.4 for the TFD.

Program 43 Hartley transform on a cyclotomic field

```
  >  with (numtheory):  n := 16:  p := 5:
  >  liste_div := op(Factor( cyclotomic(n,X) ) mod p ):
  >  P := liste_div[1];
  >  alias ( zeta = RootOf(P) ): # primitive nth root
  >  g := 2:  # 2 is a sqrt pof -1 mod 5
```

$$P := X^4 + 3$$

```
  >  cas := proc (x)
  >  1/2*( zeta^(x)*(1-g) + zeta^(-x)*(1+g) );
  >  end  proc :
```

Hartley transform, version $O(n^2)$:

```
  >  Hartley := proc (f)
  >     local  res, formule;
  >     formule := 'f[l+1]*cas((k-1)*l)';
  >     res := [ seq( sum( formule, 'l'=0..n-1 ) mod p , k=1..n)  ];
  >     return (Normal(res) mod p);
  >  end  proc :
```

Test simple :

```
  >  hasard := rand(0..(p-1)):
  >  x := [seq( hasard(), i=1..n )];
  >  y := simplify(Hartley(x));  # Hartley(x) is not anymore in F_p.
  >  evalb( x = Hartley(y)/n mod p ); # should be equal.
```

$$x := [0, 1, 1, 1, 2, 2, 1, 4, 0, 1, 2, 0, 1, 0, 2, 0]$$

$$y := [3, 3\,\zeta^2 + 4 + 4\,\zeta + 3\,\zeta^3, 2 + 4\,\zeta^2, 1 + 4\,\zeta^3 + 3\,\zeta^2, 3, 2\,\zeta^2 + 2\,\zeta + \zeta^3 + 4, 2,$$
$$2\,\zeta^3 + 2\,\zeta^2 + 1, 0, \zeta + 2\,\zeta^3 + 3\,\zeta^2 + 4, 2 + \zeta^2, \zeta^3 + 3\,\zeta^2 + 1, 1, 3\,\zeta + 4\,\zeta^3 + 2\,\zeta^2 + 4,$$
$$2, 2\,\zeta^2 + 3\,\zeta^3 + 1]$$

true

Correction (exercise 50)

1. We must take $G_{i,j} = g[ij] = g[ji]$. The multiplication by G corresponds well to the acyclic convolution which defines y.

2. The T matrix is the block made up of the first m rows and the first n columns. To calculate Tx, we calculate $\tilde{y} = C\tilde{x}$ where $\tilde{x} = (x, 0, \ldots, 0)^\top \in \mathbb{C}^M$, and we extract from $\tilde{y}$ the first m components to find y. We have $\tilde{y} = c * \tilde{x}$, which is quickly calculated by FFT.

3. We must therefore consider $c = \{g[0], \ldots, g[N-1], 0, \ldots, 0, g[N-1], \ldots, g[1]\} \in \mathbb{C}^M$. The Matlab 44 procedure performs the calculation of the vector Z transform by the CZT algorithm.

Program 44 Procedure `czt`

```
function y = czt (x, z)
n = length(x);
g = z.∧(1/2*(0:n-1)'.∧2); h = x./g;
k = ceil (log2(2*n-1)); M = 2∧k;
g = [g; zeros(M-2*n+1.1); g (n:  -1:2)];
h = [h; zeros(Mn, 1)];
y = ifft(fft(g).*fft(h));
y = y(1:n) ./ g (1:n);
```

Correction (exercise 51) We denote by $x_n = f(n)$ the discretized signal, and $y_n^{(i)} \approx \int_0^n f(t)\mathrm{d}t$ the result obtained with the method (M_i). Let $X = \mathcal{Z}(x_n)$ and $Y^{(i)} = \mathcal{Z}(y_n^{(i)})$ the transforms in Z. We denote by $H^{(i)} \overset{\text{def.}}{=} Y^{(i)}/X$ transfer functions. We have

$$H^{(1)}(z) = \frac{1}{z-1}, \quad H^{(2)}(z) = \frac{1}{2}\frac{z+1}{z-1}, \quad H^{(3)}(z) = \frac{1}{2}\frac{1+4z+z^2}{z^2-1}.$$

The figure A.1 shows the frequency responses of these three filters. We see that they amplify the low frequencies a lot (they are not stable), which is normal, since we have made integrators, which smooth the input signal.

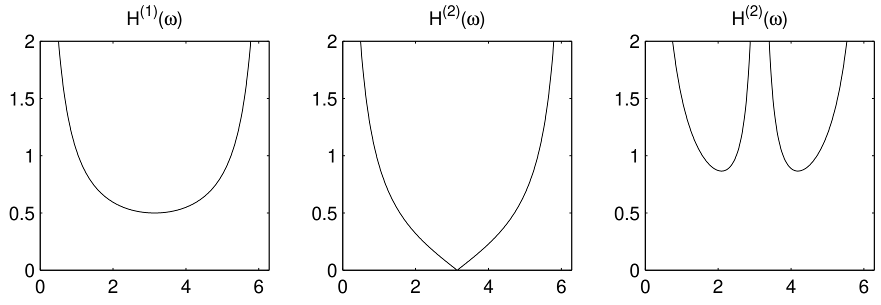

Figure A.1: Frequency responses of the three filters

Correction (exercise 52)

1. We have

$$\beta^3(x) = \begin{cases} 2/3 - |x|^2 + |x|^3/2 & \text{si } 0 \le |x| \le 1, \\ (2-|x|)^3/6 & \text{si } 1 \le |x| \le 2, \\ 0 & \text{otherwise,} \end{cases}$$

which gives $\beta_d^3 = \{\ldots, 0, 1/6, 2/3, 1/6, 0, \ldots\}$.

2. We have the decomposition

$$\mathcal{Z}(\Phi_d^3)(z) = \frac{-6\alpha}{1-\alpha^2}\left(\frac{1}{1-\alpha z^{-1}} + \frac{1}{1+\alpha z} - 1\right) \quad \text{with} \quad \alpha = \sqrt{3} - 2.$$

The fraction in z^{-1} (respectively in z) corresponds to a recursive causal filter (respectively anti-causal) which is stable. To calculate c, we must filter u_d by the two filters, (one according to the increasing indices and the other according to the decreasing indices), add the results, subtract u_d, and multiply the whole by $-6\alpha/(1-\alpha^2)$.

Program 45 Procedure `coef_spline_1`

```
function c = coef_spline_1 (ud)
K = length(ud); alpha = sqrt(3)-2;
b1 = alpha; b0 = -6 * b1/(1-b1∧2);
c1 = zeros(K, 1); c2 = zeros(K, 1);
for i = 2:K
  c1 (i) = ud(i) + b1 * c1 (i-1);
end
for i = (K-1):-1:1
  c2 (i) = ud (i) + b1 * c2 (i+1);
end
c = b0 * (c1 + c2-ud);
```

3. With the previous question, there comes $b_0 = -6\alpha/(1-\alpha^2)$ and $b_1 = \alpha$. We can impose $c^+[0] = c^-[K-1] = 0$. For more complex conditions, we will look at [67]. The Matlab 45 procedure calculates the coefficients of the interpolation by this method.

4. We have the decomposition
$$\mathcal{Z}(\Phi_d^3)(z) = \frac{-6\alpha}{(1-\alpha z^{-1})(1-\alpha z)}.$$
The vector c is therefore obtained by the composition of two filters (one causal, the other anti-causal). The Matlab 46 program uses this second decomposition.

Program 46 Procedure `coef_spline_2`

```
function c = coef_spline_2 (ud)
K = length(ud); alpha = sqrt(3) -2;
c = zeros(K, 1); d = zeros(K, 1);
for i = 2:K
  d (i) = 6 * ud (i) -alpha * d (i-1);
end
c (K) = -6 * alpha/(1-alpha∧2) * (2*d (K) -6 * ud (K));
for i = (K-1):-1:1
  c (i) = alpha * (c (i+1) -d (i+1));
end
```

Correction (exercise 53) Using the fact that $a \mapsto g^a$ is a bijection of $\{0, \ldots, p-2\}$ into $\mathbb{F}_p^*$, we get
$$\hat{f}(g^{-b}) = f(0) + \sum_{x \in \mathbb{F}_p^*} f(x)\omega_p^{-xg^{-b}} = f(0) + \sum_{a=0}^{p-2} f(g^a)\omega_p^{-g^{ab}}.$$
We denote by $\tilde{f}$ and $\tilde{h}$ the vectors of $\mathbb{C}^{p-1}$ defined by $\tilde{f}[k] = f(g^k)$ and $\tilde{h}[k] = \omega_p^{-g^{-k}}$. We check that the definition of $\tilde{h}$ is independent of a translation of k by $p-1$, so $\tilde{h}$ can be seen as a function $(p-1)$ -periodic. Consequently, the expression of $\hat{f}(g^{-b})$ corresponds well to the circular convolution $\tilde{f} * \tilde{h}[b]$. We can therefore calculate a DFT of length p thanks to a convolution of length $p-1$, therefore to 3 DFTs of length $p-1$. This is advantageous because p does not admit a factorization, whereas $p-1$ admits one (it is at least divisible by 2), which allows to use for example the method of Cooley-Tukey, paragraph 3.2.4.
The Matlab 47 procedure uses this method. You must provide it with a generator of $\mathbb{F}_p^*$ in the `g` parameter. It uses an auxiliary function `invmod`, program 48 which computes an inverse modulo p.

Correction (exercise 54)

1. We have
$$\hat{f}(y_k) \approx \frac{a}{N}\sum_{s=0}^{N-1} f(x_s)e^{-\imath x_s y_k} = \frac{a}{N}e^{\imath\delta}\sum_{s=0}^{N-1} \tilde{f}[s]e^{-\frac{2\imath\pi}{N}sk\gamma} = \frac{a}{N}e^{\imath\delta}G(\tilde{f}, \gamma)[k],$$

Program 47 Procedure `fft_chirp`

```
function y = fft_chirp (x, g)
p = length(x); f = zeros(p-1.1); h = zeros(p-1,1);
for k = 0:p-2
  j = mod(g∧k, p); jj = invmod(j, p);
  f(k+1) = x(j+1); h (k+1) = exp(-2i * pi/p * jj);
end
h = ifft(fft(f).*fft(h));
y = zeros(p, 1); y(1) = sum(x);
for k = 0:p-2
  j = mod(g∧k, p); jj = invmod(j, p);
  y(dd+1) = x(1) + h (k+1);
end
```

Program 48 Procedure `invmod`

```
function y = invmod(x, p)
[u, y, d] = gcd(x, p); y = mod(y, p);
```

where $\delta = \frac{N}{2}\left(\zeta - \frac{2\pi}{a}\gamma\frac{N}{2}\right)$ and $\tilde{f}[s] = f[s]e^{-\imath\zeta s}$. If $\zeta = \frac{p}{q}$, we have

$$G(\tilde{f},\, \gamma) = \sum_{k=0}^{N-1} \tilde{f}[s]e^{-\frac{2\imath\pi}{qN}s(pk)}.$$

This can be calculated by adding $N(q-1)$ zeros to the end of f, then calculating a DFT of size Nq. The use of the fractional Fourier transform is very advantageous if q is large.

2. We will define a fractional Fourier transform G_{sim} which uses the Simpson method instead of the rectangle method for the Fourier integral. We suppose that $N = 2N_0 + 1$, and taking care to divide the sums well, we obtain

$$G_{\text{sim}}(f,\, \gamma)[k] = \frac{1}{3}\sum_{s=0}^{N_0-1} f[2s]\omega_N^{-2sk\gamma} + \frac{4}{3}\sum_{s=0}^{N_0-1} f[2s+1]\omega_N^{-(2s+1)k\gamma}$$
$$+\frac{1}{3}\sum_{s=1}^{N_0} f[2s]\omega_N^{-2sk\gamma} = G(g,\, \gamma),$$

where g is defined by $g[0] = \frac{1}{3}f[0]$, $g[N-1] = \frac{1}{3}f[N-1]$ and

$$\begin{cases} g[2k] = \frac{2}{3}f[2k] & \text{for } k = 1, \dots,\, N_0 - 1, \\ g[2k+1] = \frac{4}{3}f[2k+1] & \text{for } k = 0, \dots,\, N_0 - 1. \end{cases}$$

The Matlab 49 procedure uses this method. To calculate $G_{\text{sim}}(f,\, \gamma)$, it must be called with the parameter `alpha` equal to $(\omega_N)^\gamma$.

Program 49 Procedure `czt_simpson`

```
function y = czt_simpson(x, alpha)
N = length(x); N0 = (N-1)/2; y = zeros(N, 1);
y(1) = x(1)/3; y(N) = x(N)/3;
y(2*(1:(N0-1))+1) = 2/3 * x(2*(1:(N0-1))+1);
y(2*(0:(N0-1)) + 2) = 4/3 * x(2*(0:(N0-1)) + 2);
y = czt(y, alpha);
```

Correction (exercise 55) The Matlab 50 procedure performs the fractional Fourier transform $G(f,\, \alpha)$, quite simply by using the `czt` algorithm (44). We can then apply it to the rows and then the columns of a matrix to calculate a 2D transform, as the 51 procedure does.

Program 50 Procedure `frft`

```
function y = frft (x, alpha)
N = length(x);
w = exp(-2i * pi * alpha/N);
y = czt(x, w);
```

Program 51 Procedure `frft2d`

```
function y = frft2d (x, alpha)
n = length(x);
for i = 1:n
  y(i,:)  = frft (x(i,:), alpha);
end
for j = 1:n
  y(:,j) = frft (y(:,j), alpha);
end
```

A.6 Correction of the exercises of chapter 6

Correction (exercise 56) The procedure Maple 52 calculates the first integer such that Φ_n has a coefficient equal to $\pm k$. Be careful, it is very slow.

Program 52 Procedure `CycloCoef`

```
CycloCoef := proc  (k)
  local i, j, P, s:
  for i from 0 to 10000 do
    P := cyclotomic (i, X): s := degree (P):
    for j from 0 to s do
      if abs(coeff(P, X, j)) = k then return (i):  end if :
    end do :
  end do :
end proc :
```

Correction (exercise 57)

1. A primary root ζ on $\mathbb{F}_p$ is a primitive root. The group generated by ζ is cardinal n, it is a subgroup of $\mathbb{F}_p^*$, so $n|p-1$.

2. We have $\mathrm{GCD}(\zeta,\, p) = 1$, so $\mathrm{GCD}(\zeta,\, p^r) = 1$, so ζ is invertible in $\mathbb{Z}/p^r\mathbb{Z}$. As $\Phi(p^r) = p^{r-1}(p-1)$, we have, with Euler's theorem, $\zeta^{\Phi(p^r)} = \zeta_0^{p-1} = 1$.

3. In $\mathbb{F}_p$, we have $\zeta^p = \zeta$, so $\zeta^s = (\zeta^p)^s = \cdots = (\zeta^{p^{r-1}})^s$. So, like $s < n \leq p$, $\zeta_0^s - 1$ is invertible in $\mathbb{F}_p$. This means that $\mathrm{GCD}(\zeta_0^s - 1,\, p) = 1$, so we have $\mathrm{GCD}(\zeta_0^s - 1,\, p^r) = 1$, and $\zeta_0^s - 1$ is also invertible in $\mathbb{Z}/p^r\mathbb{Z}$.

4. With the Chinese theorem, we have $\mathbb{Z}/m\mathbb{Z} \simeq \prod \mathbb{Z}/p_i^{k_i}\mathbb{Z}$. In each $\mathbb{Z}/p_i^{k_i}\mathbb{Z}$, we choose a root $n^{\text{ième}}$ principal ζ_i. We then check that $(\zeta_1, \ldots, \zeta_r) \in \prod \mathbb{Z}/p_i^{k_i}\mathbb{Z}$ is a main $n^{\text{ième}}$ root.

Correction (exercise 58)

1. If ζ is a mn^{th} root of the unit, then $\alpha = \zeta^m$ is a $n^{\text{th root}}$ of the unit. Moreover, $\alpha^i - 1 = \zeta^{mi} - 1$ is not a divisor of zero, since $0 < mi \leq mn$. Ditto for $\beta = \zeta^n$.
   Conversely, if ζ^m is a principal root $n^{\text{ième}}$, then $\zeta^{mn} = (\zeta^m)^n = 1$ so ζ is a mn^{th} root of the unit. Let $0 < i < mn$, and a such that $a\zeta^i = a$. Let us show that $a = 0$. By Euclidean division, we write

$i = nq + r$, with $0 \leq r < n$, and $q < m$. If $r = 0$, then, since $\zeta^{nq} - 1$ is not a divisor of zero (since ζ^m is principal root n^{th}), we have finished. If $r > 0$, then we have, by iterating $a\zeta^i = a$, the relation $a(\zeta^i)^k = a$. By taking $k = m$, we get $a\zeta^{mnq+mr} = a\zeta^{mr} = a$, which implies $a = 0$ because ζ^m is principal root $n^{\text{ième}}$.

2. The square roots of the unit are roots of the polynomial $(X-1)(X+1)$ therefore are -1 or $+1$. To obtain a principal root, it is therefore necessary that $\zeta = -1$ and that $\zeta - 1 = -2$ is not a divisor of zero. So if 2 is not a divisor of zero, -1 is the only principal square root.

3. The property was demonstrated in the previous question for $k = 1$. We assume the property demonstrated up to the range $k-1 > 0$. With question 1, ζ is a $(2^k)^{\text{ith}}$ principal root if and only if $(2^{k-1})^{\text{ième}}$ is a principal square root (thus equal to -1) and if ζ^2 is a root $(2^{k-1})^{\text{th}}$. By applying the induction hypothesis to ζ^2, we have $(\zeta^2)^{2^{k-2}} = \zeta^{2^{k-1}} = -1$.

Correction (exercise 59)

1. We denote by $v_k^0 = 1 + v_k$ and $v_k^1 = v_k$. We have decomposition
$$\tilde{f}(v_0, \ldots, v_{n-1}) = \sum_{(i_0, \ldots, i_{n-1}) \in \{0,1\}^n} \tilde{f}(i_0, \ldots, i_{n-1}) \prod_{k=0}^{n-1} v_k^{i_k}.$$
By developing the products, we find a polynomial of the requested form. Since there are 2^n such polynomials, and 2^n boolean functions, the decomposition found is unique.

2. We have
$$\mathcal{W}(f)(k) = \sum_{u \in (\mathbb{F}_2)^n} (-1)^{\langle u, k\rangle + \tilde{f}(u)},$$
so $\mathcal{W}(f)(k)$ is equal to the number of 0 minus the number of 1 in the vector $\{\tilde{f}(u) + \langle u, k\rangle\}_{u \in (\mathbb{F}_2)^n}$. This means that
$$\mathcal{W}(f)(k) = 2^n - 2d(f, f_{k,0}).$$
As we also have
$$d(f, 1 + f_{k,0}) = d(f, f_{k,1}) = 2^n - d(f, f_{k,0}),$$
he comes
$$\min\left(d(f, f_{k,0}), d(f, f_{k,1})\right) = \frac{1}{2}\left(2^n - |\mathcal{W}(f)(k)|\right).$$
Hence the result, passing to min on the set of $f_{k,b}$.

3. If f matches $|\mathcal{W}(f)(k)| = 2^{n/2}$, then $N(f) = 2^{n-1} - 2^{n/2-1}$. Let g be a function such that $\exists k,\ |\mathcal{W}(g)(k)| \neq 2^{n/2}$. With the Plancherel formula, proposition 15, we have
$$\sum_{s=0}^{2^n-1} |\mathcal{W}(g)(s)|^2 = 2^{2n}.$$
Thus, since there are 2^n terms in the sum, $\exists k$ such that $|\mathcal{W}(g)(k)| > 2^{n/2}$. So we have $N(g) < 2^{n-1} - 2^{n/2-1}$.

4. We have
$$\begin{aligned}\mathcal{W}(h)(w) &= \sum_{t \in (\mathbb{F}_2)^{n+m}} (-1)^{\langle w, t\rangle + \tilde{h}(t)} \\ &= \sum_{r \in (\mathbb{F}_2)^n} \sum_{s \in (\mathbb{F}_2)^m} (-1)^{\langle u, r\rangle + \tilde{f}(r)} (-1)^{\langle v, s\rangle + \tilde{g}(s)} = \mathcal{W}(f)(u)\mathcal{W}(g)(v).\end{aligned}$$

If $\tilde{f}$ and $\tilde{g}$ are bents, $|\mathcal{W}(f)(u)| = 2^{n/2}$ and $|\mathcal{W}(g)(v)| = 2^{m/2}$ and therefore we have $|\mathcal{W}(h)(w)| = 2^{(m+n)/2}$. Conversely, if for example $\tilde{f}$ is not bent, we have already seen in question 3 that $\exists u_0,\ |\mathcal{W}(f)(u_0)| > 2^{n/2}$. If we assume that $\tilde{h}$ is bent, then for $w = (u_0,\, v)$,

$$2^{(m+n)/2} = |\mathcal{W}(h)(w)| = |\mathcal{W}(f)(u_0)||\mathcal{W}(g)(v)| \Longrightarrow \forall v,\ |\mathcal{W}(g)(v)| < 2^{m/2},$$

which is impossible because $N(g) \leq 2^{m-1} - 2^{m/2-1}$.
We check that $\mathcal{W}(f_0) = \{2,\, 2,\, 2,\, -2\}$, so f_0 is bent. Thus the function

$$f(u_0, \ldots,\, u_{n-1}) = u_0 u_1 + \cdots + u_{n-2} u_{n-1}$$

is bent.

5. The dimension of $R(1,\, n)$ is $n+1$, and its minimum distance is 2^{n-1}. This is because the functions $\tilde{f}_{a,\,b}$, for $a \neq 0$, take 2^{n-1} times the value 0, and 2^{n-1} times the value 1.
We have $f_{a,\,b}(u) = (-1)^b \mathcal{W}(\delta_a)(u)$, which is calculated quickly using the FWT algorithm. The Matlab 53 procedure performs this encoding. It takes as parameter a as an integer $0 \leq a \leq 2^n - 1$.

Program 53 Procedure `encode_rm`

```
function y = encode_rm(a, b, n)
y = zeros(2^n, 1); y(a+1) = 1;
y = fwt(y); y = (1-y)/2;
if  (b==1) y = 1-y; end ;
```

With question 2, we see that $d(F_{a,\,b},\, F)$ is minimal when $|\mathcal{W}(f)(a)|$ is maximum. Then, we have $b = 0$ if $\mathcal{W}(f)(a) > 0$, and $b = 1$ otherwise. The Matlab 54 procedure performs this decoding, and takes as input a vector `x` of size 2^n representing F.

Program 54 Procedure `decode_rm`

```
function y = decode_rm(x)
N = length(x);
f = fwt((-1).^x);
[v, a] = max(abs(f));
y = encode_rm(a-1, f(a) <0, log2(N));
```

Correction (exercise 60)

1. We have, using Plancherel's formula, proposition 15,

$$E((fh)^2) = \sum_{\alpha \neq \beta} a_\alpha^2 = 1 - a_\beta^2.$$

2. We denote by $I(x)$ the function which is worth 1 if $f(x) \neq h_0(x)$, and 0 otherwise. We have

$$\mathbb{P}(f(x) \neq h_0(x)) = \frac{1}{2^n} \sum_x I(x).$$

We must therefore show that $I(x) \leq (f(x) - h(x))^2$. If $f(x) \neq h_0(x)$, then $I(x) = 0$ and the inequality is true. Otherwise, then necessarily $|f(x) - h(x)| \geq 1$, and we have $I(x) = 1 \leq (f(x) - h(x))^2$.

3. By applying the Chernoff-Hoeffding bound to $X_i = f(x_i)$, which are iid and such that $E(X_i) = E(f)$, $X_i \in [-1,\, 1]$, we get

$$\mathbb{P}(|\tilde{c_\beta} - c_\beta| \geq \lambda) \leq 2e^{-\lambda^2 m/2} \leq \delta.$$

With a probability less than δ, we therefore have

$$\mathbb{P}(f(x) \neq \varphi_0(x)) \leq E((f - \tilde{a_\beta}\chi_\beta)^2) \leq \sum_{\alpha \neq \beta} a_\alpha^2 + (a_\beta - \tilde{a_\beta})^2 \leq 1 - a_\beta^2 + \lambda^2.$$

4. We note $S_d \overset{\text{def.}}{=} \{s : w(s) < d\}$. Using the Chernoff-Hoeffding bound, we have $\mathbb{P}(|a_s - \tilde{a_s}| \geq \lambda) \leq 2e^{-\lambda^2 m/2}$. Moreover, under the condition $|a_s - \tilde{a_s}| \leq \lambda$, we have

$$E((f-\varphi)^2) \leq \alpha + \sum_{s \in S_d} (a_s - \tilde{a_s})^2 \leq \alpha + n^d \lambda^2,$$

since $\mathrm{Card}(S_d) \leq n^d$. To have $\mathbb{P}(f(x) \neq \varphi(x)) \leq \alpha + \epsilon$, we must therefore impose $\lambda \leq \sqrt{\epsilon/n^d}$, and so that this takes place with a probability $1-\delta$, we must have $2e^{-\lambda^2 m/2} n^d \leq \delta$, since

$$\mathbb{P}\left(\forall s \in S_d,\ |a_s - \tilde{a_s}| \geq \lambda\right) \leq \sum_{s \in S_d} \mathbb{P}(|a_s - \tilde{a_s}| \geq \lambda) \leq 2e^{-\lambda^2 m/2} n^d.$$

Correction (exercise 61)

1. We denote by G and H generator and control matrices. We have

$$y \in \mathcal{C}^\perp \Leftrightarrow \forall x \in (\mathbb{F}_p)^n,\ \langle Gx,\, y\rangle = 0 \Leftrightarrow \forall x \in (\mathbb{F}_p)^n,\ \langle x,\, G^\top y\rangle = 0 \Leftrightarrow G^\top y = 0.$$

So a control matrix of $\mathcal{C}^\perp$ is $G^\top$, and a generator matrix is $H^\top$.

2. Such a form simplifies the encoding (as well as the decoding, as we will see on the form of H). We can choose $H = (A|\,\mathrm{Id}_{nm})$, and we verify that $HG = 0$, with $\mathrm{rang}(G) = mn$.

3. Two codes are equivalent if we can pass from a G_1 matrix of the first code to a G_2 matrix of the second by
   - elementary operations on the columns (which do not modify the space generated by the columns, and therefore the code). This corresponds to the operations $C_i \leftarrow \lambda C_i$ as well as $C_i \leftarrow C_i + \lambda C_j$, for $C_i \neq C_j$ of columns and $\lambda \neq 0$.
   - permutation of lines (which corresponds to a permutation of symbols).

   By Gaussian pivoting (see [16]), we can reduce ourselves, by these operations, to a systematic form.

Correction (exercise 62)

1. The minimum distance of a code is the minimum number of linearly dependent columns. As two columns are distinct here, their modulo 2 sum is never zero, and they are therefore independent. On the other hand, this sum is necessarily equal to a third column, so we can find three related columns. The minimum distance is therefore 3.
   Since H has k lines, the dimension of $\mathcal{C}$ is $2^k - 1 - k$. If $v' = v + e$ is the word received, with $w(e) = 1$ and $v \in \mathcal{C}$, then $s = Hv' = He$ is the syndrome. So, s is a column of H. For example, if we have taken as i^{th} column of H the decomposition of i into binary, then the position of the error in v' is simply the integer of which s is the binary decomposition. The Matlab 55 function performs the decoding. It uses the procedure `writing_binary` 56, which decomposes an integer into binary writing.

Program 55 Procedure `decode_hamming`

```
function y = decode_hamming (x)
n = length(x); k = log2(n+1);
H = zeros(k, n); y = x;
for (j = 1:n) H(:,j) = write_binary(j, k); end ;
s = mod(H * x, 2);
e = dot (s, 2.∧(0:(k-1)));
if  (e ∼ = 0) y(e) = 1-y(e); end ;
```

Program 56 Procedure `write_binary`

```
function y = write_binary(x, k)
y = zeros(k, 1);
for (i = 1:k) q = floor(x/2); y(i) = x-2*q; x = q; end ;
```

2. In base $\{\alpha, \dots, \alpha^k\}$ of $\mathbb{F}_{2^k}$ as vector space on $\mathbb{F}_2$, α s' written, in vector form, $(1, 0, \dots, 0)$, α^2 is written $(0, 1, , 0, \dots, 0)$, etc. The α^i, for i between 1 and n, describe all binary representations of numbers between 1 and n. A word of $\mathcal{C}_\alpha$, the BCH code generated by α (of assigned distance n) is represented by a polynomial P and $P \in \mathbb{C}_\alpha$ if and only if $P(\alpha) = \cdots = P(\alpha^n) = 0$. By writing P in the form of a binary vector $\tilde{P}$ of size n, the preceding equalities are written $H\tilde{P} = 0$. This code is therefore the Hamming code of size n.

3. There are $n + 1 = 2^k$ words in a ball of radius 1. Since the minimum code distance is 3, the balls centered in the code words are disjoint. This set of balls therefore contains $2^m \times 2^k = 2^{2^k-1} = 2^n$ words, that is to say all the space $(\mathbb{F}_2)^n$. In the case of $n = 7$, we can take
$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix},$$
and we check that $HG = 0$, where G is the matrix of the example 12.

4. The generator matrix of $\mathcal{C}^\perp$ is $G_0 = H^\top$. For H, we have chosen for $i^{\text{ième}}$ column $(1 \le i \le 2^k - 1)$ the decomposition of i in binary writing. We precede G_0 with a line of 0 (which does not change the weight of the words). We then see that the first column is an alternation of 0 and 1, that the second is an alternation of 00 and 11, and so on. All the columns of G_0 thus have for weight 2^{k-1}. We can easily see that an elementary operation on the rows of the type $L_i \leftarrow L_i + \sum_{j\in J} L_j$ is content to perform a permutation on the columns of the matrix, therefore the weight of the columns remainder 2^{k-1}. By such operations, we obtain all the non-zero words of $\mathcal{C}^\perp$, which therefore have as weight 2^{k-1}. As the distance between two words is the weight of the word difference, this code is very simple.

5. On $\mathbb{F}_q$, we choose for the columns of H representatives of vector lines, i.e. of the projective space $\mathbb{P}(\mathbb{F}_q^n)$. As $\mathrm{Card}(\mathbb{P}(\mathbb{F}_q^n)) = \frac{q^k-1}{q-1}$ (each line has $q-1$ non-zero elements), we get the size and the desired dimension. The proof of the minimum distance is unchanged. The code is still perfect, a ball of radius 1 containing $n(q-1)+1$ words.

Correction (exercise 63)

1. The binary representation of integers allows to establish a bijection between the set $\{1, \dots, 2^k - 1\}$ and $\mathbb{F}_2^k \backslash \{0\}$. For $k = 3$, we have
$$W_{\mathcal{C}_7}(X, Y) = X^7 + 7X^3Y^4 + 7X^4Y^3 + X^7$$

2. See exercise 62, question 4.

3. We have $W_{\mathcal{C}^\perp}(X, Y) = Y^n + 2^{k-1}X^{n-2^{k-1}}Y^{2^{k-1}}$, hence the result with the theorem 9.

4. We denote by $P(Y) = W_{\mathcal{C}}(1, Y)$. We have $A_s = \frac{1}{s!}\frac{\mathrm{d}^s P}{\mathrm{d}Y^s}(0)$. The script Maple 57 performs the calculation, and we find $A_1 = A_2 = 0$, which is logical, since the minimum distance is 3. Here are some values of A_k

k	1	2	3	4	5	6
A_k	0	0	$\frac{n(n-1)}{6}$	$\frac{n(n-1)(n-3)}{24}$	$\frac{n(n-1)(n-3)(n-7)}{120}$	$\frac{n(n-1)(n-3)(n-5)(n-7)}{720}$

Program 57 Calculation of A_k for a Hamming code

```
P := Y -> 1/(n+1) * ((1 + Y)^n + n * (1-Y)^(( n+1)/2) * (1+Y)^((n-1)/2));
A := s -> factor (1/(s!)  * eval(diff(P (Y), Y$s), Y=0));
```

5. To show symmetry, we have to show that $P(Y) = Y^n P(1/Y)$, which is easily verified.

Correction (exercise 64) We check that $W_{\mathcal{C}}(X, Y) = X^8 + 14X^4Y^4 + Y^8$, and that this polynomial is invariant by the change of variable $(X, Y) \mapsto \frac{1}{\sqrt{2}}(X + Y, XY)$ (this can easily be checked using Maple).

Correction (exercise 65)

1. $\mathcal{C}$ is a code of dimension 1 and minimum distance n. Its enumerator polynomial is $Y^n + (q-1)X^n$.
2. We have $x \in \mathcal{C}^\perp \Leftrightarrow \sum x_i = 0$ in $\mathbb{F}_q$. The dual code corresponds to the code of the parity bit (example 9 which we extend to $\mathbb{F}_q$). It has dimension $n-1$ and minimum distance 1. We can easily see that $\mathcal{C} = \mathcal{C}^\perp$ if and only if $n = 1$ (trivial code) or $n = 2$.

Correction (exercise 66)

1. In the case of the construction by Walsh matrix ($n = 2^k$), the words of $\mathcal{A}_n$ are the words of $R(1, k)$, the Reed-Muller code (see exercise 59, question 5), to which the first symbol has been removed. Equivalently, $R(1, k)$ corresponds to the code $\mathcal{A}_n$ with an additional parity bit (which is always 1). For this construction, $\mathcal{A}_n$, just like $\mathcal{B}_n$, is therefore linear. For $n = 12$, with the construction of Paley, we obtain the following code $\mathcal{B}_{12}$, which is non-linear.

$\mathcal{A}_{12}$	$\mathcal{B}_{12} \backslash \mathcal{A}_{12}$
(00000000000) (00010110111)	(11111111111) (11101001000)
(11011100010) (10001011011)	(00100011101) (01110100100)
(01101110001) (11000101101)	(10010001110) (00111010010)
(10110111000) (11100010110)	(01001000111) (00011101001)
(01011011100) (01110001011)	(10100100011) (10001110100)
(00101101110) (10111000101)	(11010010001) (01000111010)

2. The orthogonality of the lines of H_n means exactly that two lines have as many equal entries as there are different entries. Two words of $\mathcal{A}_n$ are therefore distant by $n/2$, which is the minimum distance. If we denote by $u, v \in \mathcal{A}_n$ and $\overline{u}, \overline{v}$ their complements, we have $d(u, v) = d(\overline{u}, \overline{v}) = n/2$, and $d(u, \overline{v}) = n/2 - 1$. So the minimum distance of $\mathcal{B}_n$ is $n/2 - 1$. The code $\mathcal{A}_n$ is of size $n-1$ with n elements, and $\mathcal{B}_n$ is of size $n-1$ with $2n$ elements.

Correction (exercise 67)

1. By developing the equality of MacWilliams, we get

$$|\mathcal{C}| W_{\mathcal{C}^\perp}(1, Y) = \sum_{k=0}^{n} A_k \left(\sum_{j=0}^{nk} C_{nk}^j Y^j \right) \left(\sum_{j=0}^{k} (-1)^j C_k^j Y^j \right).$$

By equaling the coefficients of these two polynomials, we find the required equalities.

2. We derive k times the equality $2^{nm}\mathcal{W}_{\mathcal{C}}(X,\,1) = \mathcal{W}_{\mathcal{C}^\perp}(X+1,\,X-1)$. By using Leibniz's rule to derive a product, we get
$$2^{nm}\sum_{i=0}^{nk} A_i k! C_{ni}^k X^i = \sum_{i=0}^{k} A'_i \sum_{s=0}^{k} C_k^s \frac{\mathrm{d}^{ks}}{\mathrm{d}X^{kz}}(X+1)^{ni}\frac{\mathrm{d}^s}{\mathrm{d}X^s}(X-1)^i.$$
By making $X = 1$ in this equality, only the term corresponding to $s = i$ remains in the second sum from the right. We obtain the desired equalities.

3. There is no word of weight $i \in \{1, \dots, d-1 = nm\}$ in $\mathcal{C}$. For $\mathcal{C}^\perp$, we must show that its minimum distance is $m+1$. Indeed, d is equal to the smallest number of linearly dependent columns in H (a control matrix of $\mathcal{C}$). So the greatest number of linearly independent columns in H is equal to $d-1 = nm$. Now $H^\top$ is the generator matrix of $\mathcal{C}^\perp$, so as soon as a word of $\mathcal{C}^\perp$ has less than nk non-zero coordinates, this word is zero. As a result, the minimum distance of $\mathcal{C}^\perp$ is at least $n-(nk)+1 = k+1$. There is in fact equality by applying the Singleton bound to $\mathcal{C}^\perp$. By restricting the sums to indices i such as $A_i \neq 0$ and $A'_i \neq 0$, using $C_n^{nk} = C_n^k$, and $A'_0 = 1$, we obtain the requested m equations.

4. We have a system of m equations with m unknowns (the A_i, for $i = n-m+1, \dots, n$). This system is in fact triangular, and it is easily resolved by ascent. Thus, the A_i are uniquely determined. For example, for $i = m-1$, we have $A_d = C_n^{k-1}$.

Correction (exercise 68)

1. With the proposition 62, we know that $B'_i \geq 0$. With the expression of A'_i found in exercise 67, question 1, which extends to the distribution B'_i (thanks to the theorem 10), we get the desired inequalities.

2. If $\mathcal{C}$ has minimum distance d, then $B_1 = \cdots = B_d = 0$, and the cardinality of $\mathcal{C}$ is $\sum_{i=0}^n B_i$. The distance distribution of $\mathcal{C}$ therefore belongs to E_n^d, and we find the announced bound.

A.7 Correction of the exercises of chapter 7

Correction (exercise 69) The line $\mathrm{Span}(e_1)$ is stable, so the representation is reducible. On the other hand, if the representation were decomposable, we would have the sum of sub-representations $K^2 = \mathrm{Span}(e_1) \oplus \mathrm{Span}(e_2 + \lambda e_1)$, where $\lambda \in K$. It is easy to see that $\mathrm{Span}(e_2 + \lambda e_1)$ cannot be stable.

Correction (exercise 70) We denote by $\varphi = A\delta_e$, where e is the neutral element of G. We have
$$Af = \sum_{g\in G} f(g)A\delta_g = \sum_{g\in G} f(g)\tau_g(A\delta_e) = \sum_{g\in G} f(g)(\tau_g f) = f * \varphi.$$

Correction (exercise 71) We can see $\chi_W = \chi_U\chi_V$ as the character of the representation of morphisms on $W = \mathcal{L}(U,\,V^*)$, or as the tensor product $W = U \otimes V$. As U is non- trivial, χ_W is distinct from χ_U and χ_V, so we have constructed a new representation. As $|\chi_U| = 1$, we have
$$\sum_{g\in G} |\chi_W(g)|^2 = \sum_{g\in G} |\chi_V(g)|^2 = |G|,$$
therefore W is indeed irreducible.

Correction (exercise 72) We denote by $\rho_k : G \to GL(V_k)$, $k = 1, \dots, p$, the irreducible representations of G, and $\sigma_l : H \mapsto GL(W_l)$, $l = 1, \dots, q$, those of H. We denote by $m_k = \dim(U_k)$ and $n_l = \dim(V_l)$. For $(k,\,l) \in \{1, \dots, p\} \times \{1, \dots, q\}$, we define
$$\tau_{k,\,l}(g,\,h) = \rho_k(g) \otimes \sigma_l(h) \in GL(V_k \otimes W_l),$$

which is a representation of $G \times H$ over $V_k \otimes W_l$. We have $\chi_{\tau_{k,\,l}} = \chi_{\rho_k}\chi_{\sigma_l}$, in particular, $\|\chi_{\tau_{k,\,l}}\|_2 = \|\chi_{\rho_k}\|_2\|\chi_{\sigma_l}\|_2 = 1$, so $\tau_{k,\,l}$ is irreducible. In addition, we have

$$\sum_{k,\,l} \dim(V_k \otimes W_l)^2 = \sum_{k,\,l} m_k^2 n_l^2 = |G| \times |H| = |G \times H|,$$

therefore we have found all the irreducible representations.

Correction (exercise 73)

1. For $\sigma \in \mathfrak{S}_n$, we consider the matrix $M_\sigma : e_i \mapsto e_{\sigma(i)}$. The group $G = \mathfrak{S}_n$ is isomorphic to the matrix group $H = \{M_\sigma\}_{\sigma\in G}$, and the action of $\mathfrak{S}_n$ by permutation of the indeterminates corresponds to the linear action of H on $K[X_1, \dots, X_n]$.
   A classical result asserts that the ring of invariants is generated by the polynomials
   $$\sigma_k(X_1, \dots, X_n) \overset{\text{def.}}{=} \sum_{i_1<\cdots<i_k} X_{i_1}X_{i_2}\cdots X_{i_k}.$$
2. We have $K[X, Y]^{V_4} = K[X^2, Y^2]$. Writing an element of $K[X, Y]^{V_4}$ as a function of X^2 and Y^2 is unique.
   We have $K[X, Y]^{C_2} = K[X^2, Y^2, XY]$. The decomposition is not unique, since we have the relation $(XY)^2 = X^2Y^2$.
3. If $f = \sum c_\alpha X^\alpha \in K[X_1, \dots, X_n]^G$, then $f = \sum c_\alpha R_G(X^\alpha)$. Thus, if we prove that $R_G(X^\alpha)$ is written as a polynomial in $R_G(X^\beta)$ for $|\beta| \le |G|$, we can write f as a function of these $R_G(X^\beta)$.
4. We have $(U_A)^k = \sum_{|\alpha|=k} a_\alpha (A \cdot X)^\alpha u^\alpha$. By adding these equalities for $A \in G$, we find the expression of the desired $S_k = S_k(U_A\ ;\ A \in G)$. Any symmetric polynomial in U_A is written as a function of $|G|$ Newton sums $S_1, \dots, S_{|G|}$. Since S_k is a symmetric polynomial, there exists $F \in K[Y_1, \dots, Y_{|G|}]$ such that $S_k = F(S_1, \dots, S_{|G|})$, which gives the requested polynomial equality. By equaling the coefficients of u^α of the two members of the equality, we see that $|G|a_\alpha R_G(X^\alpha)$ is equal to a polynomial in $R_G(X^\beta)$, for $|\beta| \le |G|$. Since K has characteristic 0, we can divide by $|G|a_\alpha$.
5. We can calculate the $R_{C_4}(X^\beta)$, for $|\beta| \le 4$:

X^iY^i	$R_{C_4}(X^iY^i)$
X	0
Y	0
X^2	$(X^2+Y^2)/2$
XY	0
Y^2	$(X^2+Y^2)/2$
X^3	0
X^2Y	0

X^iY^i	$R_{C_4}(X^iY^i)$
XY^2	0
Y^3	0
X^4	$(X^4+Y^4)/2$
X^3Y	$(X^3Y - XY^3)/2$
X^2Y^2	X^2Y^2
XY^3	$-(X^3Y - XY^3)/2$
Y^4	$(X^4+Y^4)/2$

   The ring of invariants is therefore generated by
   $$P_1 = X^2 + Y^2, \quad P_2 = X^4 + Y^4, \quad P_3 = X^3Y - XY^3, \quad P_4 = X^2Y^2.$$
   However, we note that $P_2 = P_1^2 - 2P_4$, so we can remove P_2 from this list.

Correction (exercise 74)

1. The action of G being linear, it preserves the degree of the homogeneous components. Since two polynomials are equal if and only if their homogeneous components are equal, we deduce the desired result.

2. The theorem 13, tells us that $\dim_{\mathbb{C}}(V_s^G) = \mathrm{Tr}(R_G)$, where R_G is the Reynolds operator for the ρ_s representation. As $\mathrm{Tr}(R_G) = \frac{1}{|G|}\sum_{g\in G}\mathrm{Tr}(A^{[s]})$, we obtain the requested formula.

3. Let $A \in G$, with eigenvalues $\omega_1, \ldots, \omega_n$. With a change of base, we can write

$$A = A^{[1]} = \mathrm{diag}(\omega_1, \ldots, \omega_n), \quad A^{[s]} = \mathrm{diag}(\omega_1^s, \ldots, \omega_n^s, \omega_1^{s-1}\omega_2, \ldots).$$

So we have

$$\mathrm{Tr}\left(A^{[s]}\right) = \sum_{i_1 \leq \cdots \leq i_s} \omega_{i_1} \cdots \omega_{i_s},$$

which corresponds well to the power of λ^s in the expansion of

$$\det(\mathrm{Id} - \lambda A)^{-1} = \prod_{i=1}^{n} (1 - \lambda\omega_i)^{-1}.$$

4. The group G_1 has the Molien series $[(1-\lambda)(1-\lambda^2)]^{-1}$. The group G_2 has the Molien series $(1-\lambda^2)^{-2}$. The number of linearly independent polynomials limits the number of algebraically independent polynomials, which makes it possible to limit the search.

Correction (exercise 75)

1. We have

$$\langle \chi_1, \chi_\pi \rangle = \frac{1}{|G|}\sum_{g\in G} |\chi_\pi(g)| = \frac{1}{|G|}\sum_{g\in G} |X_g|.$$

2. We denote by $G_x = \{g \in G \,:\, g \cdot x = x\}$ the stabilizer of X, and $Gx = \{g \cdot x \,:\, g \in G\}$ the orbit of x. We have $|G| = |G_x||Gx|$ (see for example [52]). We denote by $x_1, \ldots, x_t$ the representatives of t distinct orbits of the action of G on X. We denote by $T = \{(g, x) \in G \times X \,:\, g \cdot x = x\}$. By counting "in both directions" the elements of T, we have

$$\sum_{g\in G} |X_g| = |T| = \sum_{x\in X} |G_x| = \sum_{i=1}^{t} \sum_{x \in Gx_i} |G_x| = \sum_{i=1}^{t} |G_{x_i}||Gx_i| = t|G|.$$

3. The cardinality of X (the "virtual" collars) is $2^6 = 64$. The number of different "real" collars is t, the number of orbits of the action of D_6 on X, applying an isometry $\sigma \in D_6$ to the regular hexagon whose the affixes are the $e^{\imath k\pi/3}$ (which is likened to a necklace!). We denote by $\{c_0, \ldots, c_5\} \in X$ a virtual necklace. We calculate $|X_\sigma|$ by making the following distinctions.
   - If $\sigma = \mathrm{Id}$, then $|X_{\mathrm{Id}}| = 64$.
   - If σ is the angle rotation $\pm\pi/6$, then if c is stable under σ, it must check $c_{i+1} = c_i$. So the necklace is one-color, hence $|X_\sigma| = 2$.
   - If σ is the angle rotation $\pm\pi/3$, then if c is stable under σ, it must check $c_0 = c_2 = c_4$ and $c_1 = c_3 = c_5$. We have 2 color choices to make, hence $|X_\sigma| = 4$.
   - Let σ be a symmetry whose axis forms an angle of $\pi/3$ with the x-coordinates (same reasoning with 0 and $2\pi/3$). So if c is stable under σ, it must check $c_0 = c_2$ and $c_3 = c_5$. There are 4 color choices to make, so $|X_\sigma| = 16$.
   - Let σ be a symmetry whose axis forms an angle of $\pi/6$ with the abscissa (same reasoning with $\pi/2$ and $5\pi/6$). So if c is stable under σ, it must check $c_0 = c_3$, $c_1 = c_2$ and $c_4 = c_5$. There are 3 color choices to make, so $|X_\sigma| = 8$.

   In the end, we therefore have

$$t = \frac{1}{12}(64 + 2 \times 2 + 2 \times 4 + 3 \times 16 + 3 \times 8) = 13\text{different necklaces.}$$

4. We write f in matrix form $f = \{\varphi(s,\, t)\}_{(s,\, t)\in X^2}$. We can easily see that the fact that f is an interleaving operator is equivalent to $\varphi(g\cdot s,\, g\cdot t) = \varphi(s,\, t)$, so φ is constant on each of the orbits of $X\times X$ under the action of G defined by $g\cdot(s,\, t) = (g\cdot s,\, g\cdot t)$. Now the double transitivity of G is equivalent to the fact that $X\times X$ has exactly 2 orbits

$$O_1 = \{(s,\, s)\ :\ s\in X\} \quad \text{et} \quad O_2 = \{(s,\, t)\ :\ s\neq t\in X\}\,.$$

If we identify f with φ, the space $\mathrm{Hom}_G(V)$ admits for base $\{1_{O_1},\, 1_{O_2}\}$, the indicator functions of O_1 and O_2.
We have already seen, during the proof of the theorem 14, the fact that $\dim(\mathrm{Hom}_G(V)) = \dim(\mathcal{L}(V,\, V)^G)$, where we have considered on $\mathcal{L}(V,\, V)$ the representation of morphisms associated with G . We have also seen that $\dim(\mathcal{L}(V,\, V)^G)$ is equal to $\mathrm{Tr}(R_G) = \langle\chi_\pi,\, \chi_\pi\rangle$, where R_G is the Reynolds operator associated with the representation of morphisms. In the end, we therefore have $\langle\chi_\pi,\, \chi_\pi\rangle = 2$.
It is obvious that the space $U = 1\mathbb{C}$ generated by the constant vector equal to 1 is invariant under G. It therefore admits a stable additional W. We have $\chi_\pi = \chi_1 + \chi_W$, where χ_1 is the character of the trivial representation. With question 2, we have $\langle\chi_\pi,\, \chi_1\rangle = 1$, since G acts transitively on X. So we have

$$\langle\chi_W,\, \chi_W\rangle = \langle\chi_\pi,\, \chi_\pi\rangle - 2\langle\chi_\pi,\, \chi_1\rangle + \langle\chi_1,\, \chi_1\rangle = 2 - 2\times 1 + 1 = 1.$$

So W is indeed irreducible.
The group $\mathfrak{S}_n$ acts doubly transitively on $X = \{1,\ldots,\, n\}$. In the previous construction, W corresponds to the standard representation, which is thus irreducible.

Correction (exercise 76)

1. We must show that f is an interleaving operator:

$$\rho(h)\circ f\circ\rho(h^{-1}) = \sum_{g\in K}\rho(hgh^{-1}) = f.$$

We have $\mathrm{Tr}(f) = d_\rho r(\rho,\, K) = \sum_{g\in K}\mathrm{Tr}(\rho(g)) = \mathrm{Card}(K)\chi_\rho(K)$.

2. We have

$$\chi(K)\chi(K^{-1}) = \frac{d_\rho}{\mathrm{Card}(K)} r(\rho,\, K)\chi(K^{-1}).$$

As ρ is irreducible, we have $\sum_{g\in G}\chi(g)\chi(g^{-1}) = |G|$, hence

$$|G| = \sum_K\sum_{g\in K}\chi(g)\chi(g^{-1}) = d_\rho\sum_K r(\rho,\, K)\chi(K^{-1}).$$

3. If $K''\nsubseteq K\cdot K'$, then $a(K,\, K',\, K'') = 0$. Let $K''\subseteq K\cdot K'$. If $hh' = h_1h_1'\in K''$, then $uxu^{-1} = uhu^{-1}\cdot uh'u^{-1} = uh_1u^{-1}\cdot uh_1'u^{-1}$, so $a(K,\, K',\, hh') = a(K,\, K',\, h_1h_1')$.

4. We have

$$\begin{aligned} r(\rho,\, K)r(\rho,\, K')\,\mathrm{Id}_V &= \left(\sum_{k\in K}\rho(g)\right)\left(\sum_{k\in K'}\rho(g)\right) \\ &= \sum_{(g,\, h)\in K\times K'}\rho(gh) = \sum_{K''} a(K,\, K',\, K'')\sum_{u\in K''}\rho(u), \end{aligned}$$

hence the requested formula.

5. The previous formula shows that the product of two generators of A is still an element of A. So A is indeed a ring. It admits a finite number of generators, so it is of finite type on $\mathbb{Z}$. This is one of the equivalent definitions of algebraic integers (see [58]).
   We have $\frac{|G|}{d_\rho} = \sum_K r(\rho, K)\chi(K^{-1})$. The $r(\rho, K)$ are algebraic integers. Moreover, the $\chi(K^{-1})$ are roots of unity, therefore algebraic integers. So, $|G|/d_\rho$ is both a rational number and an algebraic integer, so it's an integer.

Correction (exercise 77)

1. We denote by A the matrix $\rho(X)$. We have
$$A_{g,\,h} = \langle A\delta_h,\, \delta_g\rangle = \sum_{a\in G} X_a \langle \delta_{ah},\, \delta_g\rangle = X_{gh^{-1}}.$$

2. The matrix of $\rho_{U\oplus V}$ is written as block diagonal of $\rho_U(X)$ and $\rho_V(X)$, hence the formula by taking the determinant.

3. As the V_i are irreducible, the morphisms of algebras ρ_i are surjective (otherwise, there would be a non-trivial under-representation). Suppose we have a relation of the type $\sum c_{jk}\lambda_{jk}(X) = 0$. By the surjectivity of ρ_i, we can find a value x_0 of X in $\mathbb{C}[G]$ such that $\rho_i(x_0) = E_{j_0k_0}$ (the matrix with a 1 in $(i_0,\, j_0)$, and 0s everywhere else). We get $\sum c_{jk}\lambda_{jk}(x_0) = c_{j_0k_0} = 0$, hence the independence.

4. We denote by $D_n(Y_1, \ldots,\, Y_{n^2})$ the generic determinant in n^2 variables. By expanding according to the first line, we obtain the relation
$$D_n(Y_1, \ldots,\, Y_{n^2}) = D_{n-1}(Y_2, \ldots)Y_1 + B(Y_2, \ldots,\, Y_{n^2}).$$
   Thus, D_n is written as a polynomial of degree 1 in $A[Y_1]$, with $A = \mathbb{C}[Y_2, \ldots,\, Y_{n^2}]$ which is factorial. By induction, if we have assumed D_{n-1} irreducible, D_n is still irreducible.

5. According to question 3, we can complete the n_i^2 linear forms λ_{jk} in a basis of linear forms, denoted $\{Y_1, \ldots,\, Y_{|G|}\}$. In this new basis, we have the equality $\Theta_\rho(G)(X) = D_{n_i}(Y_1, \ldots,\, Y_{n_i^2})$, which is irreducible as a polynomial in Y_i. By inverse change of coordinates, we see that $\Theta_\rho(G)(X)$ is still irreducible as a polynomial in X_g.

6. As $\rho_i(1) = \mathrm{Id}_{n_i}$, X_1 only appears on the diagonal of $\rho_i(X)$. By writing the expansion of the determinant, we obtain, by writing only the degree terms n_i and $n_i - 1$ in X_1,
$$\begin{aligned}\Theta_{\rho_i}(X) &= \prod_{j=1}^{n_i} \lambda_{jj}(X) + \cdots = \prod_{h\in G}\sum_{g\in G} \langle \rho_i(g)\delta_h,\, \delta_h\rangle X_g + \cdots \\ &= X_1^{n_i} + \sum_{g\neq 1} X_1^{n_i-1}\left(\sum_{h\in G} \langle \rho_i(g)\delta_h,\, \delta_h\rangle\right) X_g,\end{aligned}$$
   hence the requested expression. The coefficients of the terms in $X_g X_1^{n_i-1}$ therefore determine χ_i, and therefore ρ_i. If Θ_{ρ_i} and Θ_{ρ_j} are proportional, they are equal (the dominant term in X_1 is equal to 1), so $\rho_i = \rho_j$.

7. The decomposition of the regular representation gives, with question 2, the requested factorization. As the $\Theta_{\rho_i}(G)$ are two by two non-proportional and irreducible, it is indeed the factorization of $\Theta(G)$ into irreducible factors.

Correction (exercise 78)

1. The product on G_p is defined by $(b,\, a)\cdot(b',\, a') = (b+ab', aa')$. The converse is given by $(b,\, a)^{-1} = (-a^{-1}b,\, a^{-1})$. The neutral element is $(0,\, 1)$. We can see G_p as $(\mathbb{F}_p,\, +) \rtimes_\varphi (\mathbb{F}_p^*,\, \cdot)$, where $\varphi : \mathbb{F}_p^* \to \mathrm{Aut}(\mathbb{F}_p)$ is defined by $\varphi(a) : b \mapsto ab$ (see [52] for the definition of the semi-direct product).

2. We have a group action of G_p on $\mathbb{F}_p$ via $(b,\, a)\cdot x = a(x+b)$. π is the translational action induced on $\mathbb{C}[\mathbb{F}_p]$. The fact that it is unitary is immediate, since $x \mapsto (b,\, a)\cdot x$ is a permutation of $\mathbb{F}_p$.

3. $f \in E$ is equivalent to $\langle f,\, 1\rangle = 0$, where we have denoted 1 the constant function equal to 1. Since π is unit, we have
$$\langle f_{(b,\, a)},\, 1\rangle = \langle f_{(b,\, a)},\, 1_{(b,\, a)}\rangle = \langle f,\, 1\rangle = 0,$$
so $f_{(b,\, a)} \in E$. We denote by $\omega_p = e^{2\imath\pi/p}$ and $e_k : x \mapsto \omega_p^{xk}$. The e_k are additive characters of $\mathbb{F}_p$, so they form an orthogonal basis of $\mathbb{C}[\mathbb{F}_p]$. As we have the decomposition $\mathbb{C}[\mathbb{F}_p] = E \oplus \mathrm{Span}(e_0)$ (orthogonal sum), the $\{e_k\}_{k=1}^{p-1}$ form a basis orthogonal of E. We have $\pi(b,\, a)(e_k) = \omega_p^{a^{-1}b} e_{ka^{-1}}$. We denote by χ the character associated with the restriction from π to E. According to the previous calculation, if $a \neq 1$, $e_{ka^{-1}} \neq e_k$ and therefore $\chi(b,\, a) = 0$. If $a = 1$, we have
$$\chi(b,\, a) = \sum_{k=0}^{p-1} \omega_p^{bk} - 1 = \begin{cases} -1 & \text{si} \quad b \neq 0, \\ p-1 & \text{si} \quad b = 0. \end{cases}$$
We denote by $\langle \cdot,\, \cdot\rangle$ the scalar product normalized on $\mathbb{C}[G_p]$. So we have
$$|G_p|\langle \chi,\, \chi\rangle = \sum_{b\in\mathbb{F}_p} \chi(b,\, 1)^2 = (p-1)^2 + p - 1 = |G_p|,$$
thus, according to the theorem 7, π restricted to E is irreducible.

Correction (exercise 79)

1. We have, using Plancherel's formula,
$$\mathcal{W}(f)(b,\, a) = \frac{1}{p}\sum_{n=0}^{p-1} \hat{f}(n)\omega_p^{bn}\overline{\hat{\psi}(an)}.$$

2. We denote by $\Phi(x)$ the right-hand side of the equality. We recall that, as f and ψ are in E, we have $\hat{f}(0) = \hat{\psi}(0) = 0$. We have
$$\begin{aligned} \hat{\Phi}(k) &= \sum_{(b,\, a)} \frac{1}{p}\sum_{n=0}^{p-1} \hat{f}(n)\omega_p^{bn}\overline{\hat{\psi}(an)}\omega_p^{-kb}\hat{\psi}(ak) \\ &= \frac{1}{p}\sum_{n=0}^{p-1} \hat{f}(n)\left(\sum_{b=0}^{p-1}\omega_p^{b(nk)}\right)\left(\sum_{a=1}^{p-1}\overline{\hat{\psi}(an)}\hat{\psi}(ak)\right). \end{aligned}$$
We then use the fact that $\sum_{b=0}^{p-1}\omega_p^{b(nk)} = p\delta_n^k$ as well as
$$\sum_{a=1}^{p-1}|\hat{\psi}(ak)|^2 = \sum_{a=0}^{p-1}|\hat{\psi}(ak)|^2 = \begin{cases} 0 & \text{si} k = 0, \\ p^2\langle \psi,\, \psi\rangle & \text{if not.} \end{cases}$$
In the end we get
$$\hat{\Phi}(k) = p^2\langle \psi,\, \psi\rangle \hat{f}(k).$$

3. Resuming the previous demonstration, this time we have

$$\sum_{a=1}^{p-1} |\hat{\psi}(ak)|^2 = \left\{ \begin{array}{ll} (p-1)|\hat{\psi}(0)|^2 & \text{si} k = 0, \\ \sum_{a=1}^{p-1} |\hat{\psi}(a)|^2 & \text{otherwise.} \end{array} \right.$$

We therefore have $\hat{\Phi} = d_\psi \hat{f}$, which gives the inversion formula.
We write $\mathcal{W}^* : \mathbb{C}[G_p] \to \mathbb{C}[\mathbb{F}_p]$ the application defined by

$$\mathcal{W}^* \varphi = \sum_{(b,\, a) \in G_p} \varphi(b,\, a) \psi_{b,\, a}.$$

It is easy to see that $\mathcal{W}^*$ is the adjunct of $\mathcal{W}$ for the usual dot products on $\mathbb{C}[\mathbb{F}_p]$ and $\mathbb{C}[G_p]$, that is, say that

$$\langle \mathcal{W} f,\, \varphi \rangle_{\mathbb{C}[G_p]} = \langle f,\, \mathcal{W}^* \varphi \rangle_{\mathbb{C}[\mathbb{F}_p]}, \quad \text{for} f \in \mathbb{C}[\mathbb{F}_p] \,\text{and} \varphi \in \mathbb{C}[G_p].$$

The inversion formula is thus written $\mathcal{W}^* \circ \mathcal{W} = d_\psi \,\mathrm{Id}$, so $\frac{1}{\sqrt{d_\psi}} \mathcal{W}$ is an isometry(although sure not bijective, the wavelet transform being very "redundant").
It is amusing to notice that the proof of the inversion formula on a finite field is in every way similar to that of the continuous wavelet transform (see [47]).

4. The Matlab 58 procedure calculates the transform into wavelet. This is a slow procedure ($O(p^2)$ operations), there is no dichotomous algorithm, as there is for the "classical" wavelet transforms (see the Haar transform , exercise 15). The procedure 59 computes the inverse transform (assuming the admissibility condition $f \in E$ and $\psi \in E$). These two procedures use the function `invmod`, program 48.

Program 58 Procedure `transfo_ondelettes`

```
function y = transformer_ondelettes (x, psi)
p = length(psi); y = zeros(p-1, p);
for (a = 1:p-1) for (b = 0:p-1)
  order = mod(invmod(a, p) * ((0:p-1) -b), p) +1;
  y(a, b+1) = dot (x, psi (order));
end ; end ;
```

Program 59 Procedure `reconstruct_ondelets`

```
function y = reconstruct_ondelets (x, psi)
p = length(psi); c = p * dot (psi, psi); y = zeros(p, 1);
for (a = 1:p-1) for (b = 0:p-1)
  order = mod(invmod(a, p) * ((0:p-1) -b), p) +1;
  y = y + x(a, b+1) * psi (order);
end ; end ;
y = y/c;
```

A.8 Correction of the exercises of chapter 8

Correction (exercise 80) We have

$$(\Phi K \Phi^*)_{i,\, j} = \sum_{s=1}^{p} k_s \chi_i(C_s) \chi_j(C_s) = \sum_{g \in G} \chi_i(g) \chi_j(g) = |G| \delta_i^j,$$

using the property of character orthogonality. Since the columns of a unit matrix are orthogonal, the desired property of orthogonality is obtained.

Correction (exercise 81) The line (Oz) is stable, so the representation is not irreducible. We calculate the character χ which checks $\chi(r^k) = 1 + 2\cos(2k\pi/n)$ as well as $\chi(sr^k) = -1$. For example, we consider the case where n is even. We obtain the decomposition coefficients

$$\langle \chi, \chi_{\psi_1} \rangle = \langle \chi, \chi_{\psi_3} \rangle = \langle \chi, \chi_{\psi_4} \rangle = 1, \quad \langle \chi, \chi_{\psi_2} \rangle = 1,$$

as well as $\langle \chi, \chi_1 \rangle = 1$ and $\langle \chi, \chi_k \rangle = 0$ for $k \neq 1$.

Correction (exercise 82) The transformation matrices are written

$$\rho((12)) = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad \rho((123)) = \frac{1}{2}\begin{pmatrix} -1 & -\sqrt{3} \\ \sqrt{3} & -1 \end{pmatrix}.$$

By adding the trivial character and the alternate character, we obtain the following table:

	1 Id	3 (12)	2 (123)
χ_1	1	1	1
χ_ϵ	1	-1	1
χ_ρ	2	0	-1

Correction (exercise 83) We denote by χ_{so} the character of the representation by permutation of vertices, and χ_{ar} that of the permutation of edges. We have

	1 Id	6 (12)	8 (123)	6 (1234)	3 (12)(34)
χ_{so}	8	0	2	0	0
χ_{ar}	12	2	0	0	0

We therefore obtain the following multiplicities:

	χ_1	χ_ϵ	χ_s	χ_W	$\chi_{W'}$
χ_{so}	1	1	1	1	0
χ_{ar}	1	0	2	1	1

Correction (exercise 84)

1. $\rho_{W'}((12)(34))$ is an involution (therefore is diagonalizable) of trace 2. It is necessarily identity.

2. If ρ is trivial over H, then $H \subset \text{Ker}(\rho)$, so ρ goes to the quotient by H which is distinguished. Conversely, if ρ goes to the quotient, then $\rho = \tilde{\rho} \circ \pi$ which is trivial over H since π is.

3. We have $H = \{\text{Id}, (12)(34), (13)(24), (14)(23)\}$, which is distinguished. The action of $\mathfrak{S}_4$ on the cube gives rise to a permutation of the pairs of opposite faces, i.e. to an application $\varphi : \mathfrak{S}_4 \to \mathfrak{S}_3$ (after suitable numbering of the faces). It is easy to see that the permutations which leave the pairs of opposite faces stable are the elements of H. We therefore have $\text{Ker}(\varphi) = H$ (which shows that H is distinguished), and by passing to the quotient, an isomorphism between $\mathfrak{S}_4/H$ and $\mathfrak{S}_3$.

4. According to question 1, $\rho_{W'}$ is trivial over H, the group generated by $(12)(34)$. So with question 2, $\rho_{W'}$ identifies with an element of $\widehat{\mathfrak{S}_4/H}$, that is, of $\hat{\mathfrak{S}}_3$. As this representation is irreducible, by consideration of dimension, it is necessarily the standard representation.

Correction (exercise 85)

1. We keep the notations of the correctness of the exercise 75. We denote by $Gx_1, \ldots, Gx_t$ the orbits, with $x_1 = e$ the neutral element of G. The orbits forming a partition of X, we have the class equation $1 + \sum_{i=2}^{t} |Gx_i| = |X| = |G|^p = 0 \mod p$, since $p||G|$. But since the $|Gx_i|$ divide p, we have $|Gx_i| \in \{1, p\}$, and the previous equality requires that at least one $x_i = (g, \ldots, g)$ matches $|Gx_i| = 1$. This means exactly that g is of order p.

2. With the result of 76, we have that 2 divides $|G|$, so with the previous question, G has an element t of order 2.

3. $\varphi \overset{\text{def.}}{=} \det \circ \rho : G \mapsto \mathbb{C}^*$ is a representation of degree 1 whose kernel is a simple group not reduced to the neutral element of G (because if $\mathrm{Ker}(\varphi) = \{1\}$, then φ is injective and G is commutative) . As G is simple, we have $\mathrm{Ker}(\varphi) = G$, and $\det(\rho(g)) = 1$, so ρ has a value in $SL_2(\mathbb{C})$. Since $X^2 - 1$ is the minimal polynomial of $\rho(t)$, the latter is diagonalizable. Like $\det(\rho(t)) = 1$, and that $\rho(t) \neq \mathrm{Id}$ (because ρ is injective since $\mathrm{Ker}(\rho)$ is a subgroup distinguished from G), its two eigenvalues are equal to -1. We can therefore find $P \in GL_2(\mathbb{C})$ such that $P\rho(t)P^{-1} = -\mathrm{Id}_2$, and therefore $\rho(t) = -\mathrm{Id}_2$. Moreover, for all $g \in G$, we have $\rho(gtg^{-1}) = \rho(g)(-\mathrm{Id}_2)\rho(g)^{-1} = \rho(t)$. But ρ is injective, so $gtg^{-1} = t$, and $t \in Z(G)$, the center of G. As $Z(G)$ is distinguished, we have $Z(G) = \{1\}$, which is a contradiction, because $t \neq 1$.

Correction (exercise 86)

1. We have to show that φ is an algebra morphism. Linearity is obvious. It remains to check the relations on the generators, for example $\varphi(jk) = \varphi(i)$.

2. It suffices to note that the application $\varphi(q) \mapsto \psi(q)$ is an algebra isomorphism. This is obvious, since $a + \imath b \mapsto \left(\begin{smallmatrix} a & -b \\ b & a \end{smallmatrix}\right)$ is an algebra isomorphism of $\mathbb{C}$ in $\mathrm{Sim}(\mathbb{R}^2)$ (the similarities of $\mathbb{R}^2$). This also shows that the representation obtained is unitary. Moreover, we check that $\|\chi_\psi\|_2 = 1$, so this representation is irreducible.

3. We have the trivial representation ρ_1 as well as the following representations:

$$\begin{aligned} \rho_2(\pm 1) = \rho_2(\pm i) = 1, \quad & \rho_2(\pm j) = \rho_2(\pm k) = -1, \\ \rho_3(\pm 1) = \rho_3(\pm j) = 1, \quad & \rho_3(\pm i) = \rho_3(\pm k) = -1, \\ \rho_4(\pm 1) = \rho_4(\pm k) = 1, \quad & \rho_4(\pm i) = \rho_4(\pm j) = -1. \end{aligned}$$

   We check that these representations are indeed irreducible, and if we denote by ρ_5 the representation of the previous question, and n_i the dimensions of the representations, we have $\sum n_i^2 = 4 \times 1^2 + 2^2 = |H_8|$, so we have all the irreducible representations. We fix an order among the elements of H_8, and we denote the entries of the matrices H_8 in the form of vectors of size 8, which gives

$$\begin{aligned} v_0 &= (1,\, 1,\, 1,\, 1,\, 1,\, 1,\, 1,\, 1), & v_4 &= \sqrt{2}(1,\, -1,\, 0,\, 0,\, 0,\, 0,\, \imath,\, -\imath), \\ v_1 &= (1,\, 1,\, 1,\, 1,\, -1,\, -1,\, -1,\, -1), & v_5 &= \sqrt{2}(0,\, 0,\, -1,\, 1,\, -\imath,\, \imath,\, 0,\, 0), \\ v_2 &= (1,\, 1,\, -1,\, -1,\, 1,\, 1,\, -1,\, -1), & v_6 &= \sqrt{2}(0,\, 0,\, 1,\, -1,\, -\imath,\, \imath,\, 0,\, 0), \\ v_3 &= (1,\, 1,\, -1,\, -1,\, -1,\, -1,\, 1,\, 1), & v_7 &= \sqrt{2}(1,\, -1,\, 0,\, 0,\, 0,\, 0,\, -\imath,\, \imath), \end{aligned}$$

   and forms an orthogonal basis of $\mathbb{C}^8$.

4. The exercise 18 explains how we can construct, by tensor product, an orthogonal basis of $\mathbb{C}^{8n}$ from a basis of $\mathbb{C}^8$. The Matlab 10 procedure allows to calculate, for $f \in \mathbb{C}^{8n}$, $(A^{\otimes n})f$, the coefficients of f in the constructed orthonormal basis. The lines of A are the vectors v_i.

Correction (exercise 87)

1. The elements of G are rotations, they keep the distance to the origin, so the polynomial $X^2 + Y^2 + Z^2$.

2. We denote by $V(XYZ) = \{(x,\, y,\, z) \in \mathbb{R}^3 \,:\, xyz = 0\}$. As the elements of G con ser vent the set of faces of the cubes, they ser vent the union of the three coordinate planes, i.e. $V(XYZ)$.
   We denote by $I(V(XYZ)) = \{P \in K[X,\, Y,\, Z] \,:\, \forall (x,\, y,\, z) \in V(XYZ),\ P(x,\, y,\, z) = 0\}$. Let then be $P \in I(V(XYZ))$. By doing the Euclidean division of P by X as a polynomial in X (which is possible because the dominant coefficient of X is invertible in $K[Y,\, Z]$), we write $P(X,\, Y,\, Z) = XQ(X,\, Y,\, Z) + R(Y,\, Z)$. Since $P(0,\, Y,\, Z) = 0$, we have $R = 0$. Continuing with the variables Y and Z, we find $P = \lambda XYZ$ with $\lambda \in \mathbb{R}$.
   Let $A \in G$. Since $V(XYZ)$ is stable by A, we have, for $(x,\, y,\, z) \in V(XYZ)$, the equality $f(A \cdot (x,\, y,\, z) = 0$, i.e. $f(A \cdot (X,\, Y,\, Z)) \in I(V(XYZ))$. so $f(A \cdot (X,\, Y,\, Z)) = \lambda f$. As $A^n = \mathrm{Id}$ for a certain n, we necessarily have $\lambda = \pm\, 1$.

3. This time, $V(f)$ is the union of the 4 planes orthogonal to the three large diagonals. As these diagonals are stable by G, we deduce that $V(f)$ is stable by G. We can do the same reasoning as in the previous question, this time starting a Euclian division by $X + Y + Z$.
   Similarly, $V(g)$ is the union of the 6 planes orthogonal to the 6 pairs of opposite diagonals inscribed in the faces of the cubes. Once again, $V(g)$ is stable by G.

Correction (exercise 88)

1. n must be even and we have $k = n/2$.

2. As the polynomial $W_{\mathcal{C}}$ is homogeneous of degree n, the identity of MacWilliams is rewritten $\mathcal{W}_{\mathcal{C}}(A \cdot (X,\, Y)) = \mathcal{W}_{\mathcal{C}}(X,\, Y)$.

3. We have $A^2 = \mathrm{Id}_2$ so $G_1 = \{A,\, \mathrm{Id}_2\}$. By applying the Reynolds operator, we find
$$R_G(X) = \frac{\sqrt{2}+1}{2\sqrt{2}}(X + (\sqrt{2}-1)Y), \quad R_G(X^2) = \frac{1}{2}(X^2 + (X+Y)^2/2).$$
   We can remove the multiplicative constant in front of $R_G(X)$ and subtract $3/4 R_G(X)^2$ from $R_G(X^2)$ to obtain the two invariants of $K[X,Y]^{G_1}$ announced. To show that these are the only ones, we can use the Molien series calculated in the exercise 74, question 5, or else calculate $R_G(XY)$, $R_G(Y)$ and $R_G(Y^2)$ to see that they are written according to the invariants already found.

4. If v is a code word, we have $\langle v,\, v\rangle = \langle v,\, 1\rangle = 0$, so $\sum v_i = 0 \mod 2$. Thus, $\mathcal{C}$ only contains words of even weight, and $\mathcal{W}_{\mathcal{C}}(-X,\, -Y) = \mathcal{W}_{\mathcal{C}}(X,\, Y)$. So $\mathcal{W}_{\mathcal{C}}$ is invariant under the action of A and $-\mathrm{Id}$.

5. The program 60 allows to calculate generators of the ring of invariants by trying all $R_G(X^k)$, for $|k| \leq |G|$. Calculation times become very important for a large group. The combinatorial explosion is twofold, both at the level of the cardinality of the group and of the set of X^k for $k \leq |G|$.

Program 60 File `invariant-polynomials.msw`

Applique une matrice a un polynome :

```
> action_matrice := proc (A,p)
>    local  g,v,m;
>    v := array([[x],[y]]);
>    m := evalm(A&*v);
>    g := subs ({x=m[1,1],y=m[2,1]},p);
>    return (g)
> end  proc :
```

Un exemple invariant :

```
> A:=1/sqrt(2)*matrix(2,2,[[1,1],[1,-1]]);
> expand( action_matrice(A,x^2+y^2) );
```

$$A := \frac{1}{2}\sqrt{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$x^2 + y^2$$

Calcule l'opérateur de Reynolds :

```
> operateur_reynolds := proc (G,p)
>    local  i,r;
>    r := (1/nops(G))*sum('action_matrice(G[k],p)','k'=1..nops(G));
>    return (r)
> end  proc :
```

Calcule des générateurs de l'anneau des invariants :

```
> polynomes_invariants := proc (G)
>    local  i,j,r;
>    r := [];
>    for  i from  1 to  nops(G) do
>    for  j from  0 to  i do
>       r := [op(r),expand(operateur_reynolds(G,x^j*y^(i-j)))];
>    end  do :  end  do :
> end  proc :
```

Un exemple en rapport aux codes auto-duaux, pour le cas où 2 divise les poids des mots du code :

```
> B:=matrix(2,2,[[-1,0],[0,-1]]):
> G:=[A,-A,B,-B]:
> polynomes_invariants( G, 4 );
```

$$[0,\, 0,\, \frac{1}{4}x^2 - \frac{1}{2}x\,y + \frac{3}{4}y^2,\, \frac{1}{4}x^2 - \frac{1}{4}y^2 + \frac{1}{2}x\,y,\, \frac{3}{4}x^2 + \frac{1}{2}x\,y + \frac{1}{4}y^2,\, 0,\, 0,\, 0,\, 0,$$
$$\frac{1}{8}x^4 - \frac{1}{2}x^3\,y + \frac{3}{4}x^2\,y^2 - \frac{1}{2}x\,y^3 + \frac{5}{8}y^4,\, \frac{1}{8}x^4 - \frac{1}{4}x^3\,y + \frac{3}{4}x\,y^3 - \frac{1}{8}y^4,$$
$$\frac{1}{8}x^4 + \frac{1}{4}x^2\,y^2 + \frac{1}{8}y^4,\, \frac{1}{8}x^4 + \frac{3}{4}x^3\,y - \frac{1}{4}x\,y^3 - \frac{1}{8}y^4,$$
$$\frac{5}{8}x^4 + \frac{1}{2}x^3\,y + \frac{3}{4}x^2\,y^2 + \frac{1}{2}x\,y^3 + \frac{1}{8}y^4]$$

Appendix B

Matlab Programs

Here is the set of Matlab programs mentioned in the previous chapters. Each program constitutes a separate file. Most of them are *procedures*, this means that these programs must be copied into a file with the same name. For example, the procedure `fht` is written in the file `fht.m`.

B.1 FWT Algorithm

The program `fwt` is a Matlab implementation of the *fast Walsh transform* algorithm presented in Paragraph 2.2.2. It is recursive, but does not use additional memory, so is relatively efficient. Note that up to a factor of $1/N$, the Walsh transform is its own inverse, so the routine does not include a parameter to calculate the inverse transform (just divide the result by N) .

Program 61 Procedure `fwt`

```
function y = fwt (x)
N = length (x); % N must be a power of 2
if  (N == 1) y = x; return ; end ;
P = N / 2;
x = [fwt (x (1:  P)); fwt (x ((P + 1):  N))];
y = zeros (N, 1);
y (1:  P) = x (1:  P) + x ((P + 1):  N);
y ((P + 1):  N) = x (1:  P) - x ((P + 1):  N);
```

B.2 FHT Algorithm

The paragraph 5.1.2, explains the operation of the *fast Hartley transform* algorithm. Here is a Matlab implementation of this algorithm.

– Procedure `fht` (program 62): the FHT algorithm itself. To calculate the inverse Hartley transform, just use the `fht` routine and divide the result by N, the length of the sample.
– Procedure `operator_chi` (program 63): used to calculate the operator χ_N^x.
– Procedure `fht_convol` (program 64): used to calculate the convolution of two real signals using the FHT algorithm.

B.3 FFT Algorithm

Here are the different implementations of the FFT algorithm presented in detail in paragraphs 3.2.1, 3.2.5 and 3.2.6.

Program 62 Procedure `fht`

```
function y = fht (f)
% N must be a power of 2
N = length (f); N1 = N / 2;
if  (N == 1) y = f; return ; end ;
y = zeros (size (f));
% construction of the two sub-vectors
f_p = f(1:  2:  N); f_i = f(2:  2:  N);
% recursive calls
f_p = fht (f_p); f_i = fht (f_i);
% application of the chi operator
f_i = operator_chi (f_i, 0.5);
% mix of the two results
y (1:  N1) = f_p + f_i; y ((N1 + 1):  N) = f_p - f_i;
```

Program 63 Procedure `operator_chi`

```
function y = operator_chi (a, x)
N = length (a); a_inv = [a (1); a (N: -1:  2)];
y = a.  * cos (2 * pi * x * (0:  N-1) '/ N) + a_inv.  * sin (2 * pi * x * (0:  N-1)' / NOT );
```

– Procedure `fft_rec` (program 65): naive and recursive version of the algorithm (temporal decimation).
– Procedure `fft_dit` (program 66): efficient implementation (both in memory use and in speed) of the algorithm (non-recursive and temporal decimation).
– Procedure `fft_dif` (program 67): frequency decimation version of the algorithm.
– Procedure `operator_s` (program 68): implementation of the operator $\mathcal{S}$.
– Procedure `rev_bits` (program 69): classifies the vector according to the reverse direction of the index bits.
– Procedure `rev_index` (program 70): calculate the integer obtained by inverting the bits of another integer.

It should be borne in mind that these Matlab programs are primarily intended for teaching. The implementation is far from being as efficient as those that can be achieved in a fast language (eg in C). There is a lot of software available on the internet which is very powerful, for example *FFTW* [32].

B.4 Multiplication of large integers by FFT

The following Matlab programs allow you to multiply large integers represented by their decomposition in a given b base.

– Procedure `mult_entiers` (program 71): allows to multiply two integers represented as vectors. We must of course provide the base used.
– Procedure `number2vector` (program 72): practical function which allows to pass from the representation in the form of an integer to the representation in the form of a vector (of limited interest however , because Matlab does not handle integers of arbitrary size).
– Procedure `vector2number` (program 73): inverse function of the previous one.
– File `test_mult_entiers.m` (program 74): small test program.

B.5 Solving the Poisson equation

Here are the different files to implement the resolution of the Poisson equation described in Paragraph 4.4.3.

– File `poisson.m` (program 75): main file, builds the different matrices, calculates the FFTs in 2D and solves the convolution equation.
– Procedure `f` (program 76): the right side of the equation. This is the function $f(x, y) = (x^2 + y^2)e^{xy}$. It is calculated so that the solution is known in advance.

Program 64 Procedure `fht_convol`

```
function y = fht_convol (x, y)
% N must be a power of 2
N = length (x); y = zeros (size (x));
a = fht (x); b = fht (y);
a_inv = [a (1); a (N: -1:  2)];
b_inv = [b (1); b (N: -1:  2)];
y = 0.5 * (a.  * b - a_inv.  * b_inv + a.  * b_inv + a_inv.  * b);
y = fht (y) / N;
```

Program 65 Procedure `fft_rec`

```
function y = fft_rec (f, dir)
% N must be a power of 2
N = length (f); N1 = N / 2;
if  (N == 1) y = f; return  end ;
y = zeros (size (f));
% construction of the two sub-vectors
f_p = f(1:  2:  N); f_i = f(2:  2:  N);
% recursive calls
f_p = fft_rec (f_p, dir);
f_i = fft_rec (f_i, dir);
% application of the S operator
f_i = operator_s (f_i, dir * 0.5);
% mix of the two results
y (1:  N1) = f_p + f_i; y ((N1 + 1):  N) = f_p - f_i;
```

Program 66 Procedure `fft_dit`

```
function y = fft_dit (f, dir)
% N must be a power of 2
N = length (f); ldn = floor (log2 (N));
f = rev_bits (f);
for ldm = 1:  ldn
  m = 2∧ ldm; m1 = m / 2;
  for j = 0:  m1-1
    e = exp (-dir * 2.0i * pi * j / m);
    for r = 0:  m:  Nm
      u = f(r + j + 1); v = f(r + j + m1 + 1) * e;
      f(r + j + 1) = u + v; f(r + j + m1 + 1) = u - v;
    end
  end
end
y = f;
```

Program 67 Procedure `fft_dif`

```
function y = fft_dif(f, dir)
% N must be a power of 2
N = length (f); ldn = floor (log2 (N));
for ldm = ldn:  -1:  1
  m = 2∧ ldm; m1 = m / 2;
  for j = 0:  m1-1
    e = exp (-dir * 2.0i * pi * j / m);
    for r = 0:  m:  N-1
      u = f(r + j + 1); v = f(r + j + m1 + 1);
      f(r + j + 1) = u + v; f(r + j + m1 + 1) = (uv) * e;
    end
  end
end
% put the transformed vector back in the correct order
y = rev_bits (f);
```

Program 68 Procedure `operator_s`

```
function y = operator_s (a, x)
N = length (a);
y = a.  * exp (-2.0i * x * (0:  N-1) '* pi / N);
```

Program 69 Procedure `rev_bits`

```
function y = rev_bits (x)
n = length (x); t = floor (log2 (n)); y = zeros (n, 1);
for i = 0:  n-1
  j = rev_index (t, i); y (j + 1) = x (i + 1);
end
```

Program 70 Procedure `rev_index`

```
function y = rev_index (t, index)
y = 0; tmp = index;
for i = 0:  t-1
  bit = mod (tmp, 2);
  tmp = floor (tmp / 2);
  y = y * 2 + bit;
end
```

Program 71 Procedure `mult_entiers`

```
function r = mult_integer (x, y, b)
N = length (x);
% add zeros for acyclic convolution
x = [x; zeros (N, 1)]; y = [y; zeros (N, 1)];
% calculate the convolution
r = round (real (ifft (fft (x).  * fft (y))));
for i = 1:  2 * N-1 % remove the retainers
  q = floor (r (i) / b);
  r (i) = r (i) -q * b; r (i + 1) = r (i + 1) + q;
end
```

Program 72 Procedure `number2vector`

```
function y = number2vector (x, b)
N = floor (log (x) / log (b)) +1; y = zeros (N, 1);
for i = 1:  N
  q = floor (x / b); y (i) = x - q * b; x = q;
end
```

Program 73 Procedure `vector2number`

```
function y = vector2number (v, b)
N = length (v); y = sum (v.  * (b.∧ (0:  N-1) '));
```

Program 74 File `test_mult_entiers.m`

```
x = 262122154512154212; y = 314134464653513212;
b = 20; % the base
xx = number2vector (x, b); yy = number2vector (y, b);
zz = mult_integers (xx, yy, b); z = vector2number (zz, b);
z - x * y % the result must be zero
```

– Procedure `sol` (program 77): the exact solution of the equation (we cheat a bit, we use an equation for which we already know the solution!). We took $s(x, y) = e^{xy}$. Its Laplacian is therefore the function f of the file `fm`.
– Procedure `u_0y`, `u_1y.m`, `u_x0.m` and `u_x1.m` (program type: 78): value of the solution u on each of the edges $x = 0$, $x = 1$, $y = 0$ and $y = 1$. We only reported the program of the `u_0y.m` function, the others being written in the same way.

The choice of the solution function is perfectly arbitrary; we can make tests with other functions (taking care to update the value of the Laplacian in the `fm` file). One will be able to make tests with conditions on the arbitrary edge (but one will not have any more exact solution of reference ...). Likewise, it is easy to change the precision of the resolution to observe the convergence of the error made by the finite difference method.

Program 75 File `poisson.m`

```
% some constants
N = 30; h = 1 / N; nb_iter = 30;
M = zeros (N + 1, N + 1); f_val = zeros (N-1, N-1);
% we start with x = h (only the center points)
for i = 1:  N-1 % calculation of the right hand side
  for j = 1:  N-1
    x = i * h; y = j * h;
    f_val (i, j) = f(x, y);
  end
end
for i = 1:  N-1 % addition of boundary terms
  x = i * h;
  f_val (i, 1) = f_val (i, 1) - 1 / h∧ 2 * f_0y (x);
  f_val (i, N-1) = f_val (i, N-1) - 1 / h∧ 2 * f_1y (x);
  f_val (1, i) = f_val (1, i) - 1 / h∧ 2 * f_x0 (x);
  f_val (N-1, i) = f_val (N-1, i) - 1 / h∧ 2 * f_x1 (x);
end
% we make the matrix odd
ff = [zeros (N-1,1), f_val, zeros (N-1,1), - f_val (:, N-1:  -1:  1)];
ff = [zeros (1,2 * N); ff; zeros (1.2 * N); -ff(N-1:  -1:  1,:)];
ff = fft2 (ff); % we calculate the FFT
d = -4 / h∧ 2 * sin ((0:  2 * N-1) * pi / (2 * N)).∧ 2;
for i = 1:  2 * N % system resolution d * u + u * d = ff
  for j = 1:  2 * N
    s = d (i) + d (j);
    if  (s == 0) s = 1; end ; % avoid division by 0
    ff(i, j) = ff(i, j) / s;
  end
end
ff = real (ifft2 (ff)); % we calculate the inverse transform
% we extract the solution
u = zeros (N + 1, N + 1); u (2:  N, 2:  N) = ff(2:  N, 2:  N);
for i = 1:  N + 1 % we put the boundary terms
  x = (i-1) * h;
  u (i, 1) = f_0y (x); u (i, N + 1) = f_1y (x);
  u (1, i) = f_x0 (x); u (N + 1, i) = f_x1 (x);
end
surf(u); title ('Resolution by FFT');
```

Program 76 Procedure `f`

```
function r = f(x, y)
r = (x∧ 2 + y∧ 2) * exp (x * y);
```

Program 77 Procedure `sol`

```
function r = sol (x, y)
r = exp (x * y);
```

Program 78 Procedure `u_0y`

```
function r = f_0y (y)
r = sol (0, y);
```

B.6 Solving the heat equation

The programs which follow allow to solve the heat equation by the method described in Paragraph 4.4.2.

– File `heat.m` (program 79): calculate the solution of the equation for different time values by calling the program `solve_eq.  m`, then draw the evolution of the solution.

– Procedure `solve_eq` (program 80): solve the heat equation for a given time by calculating the Fourier coefficients by FFT.

– Procedure `f` (program 81): the initial distribution of heat at time $t = 0$. We have taken here a step function (therefore discontinuous).

Of course, it is easy to modify these programs, especially to experiment with different initial conditions, as well as with other values of the time parameter.

Program 79 File `heat.m`

```
% number of interpolation points for the calculation of the integral
M = 2∧ 8; h = 1 / M;
% value of f at interpolation points
f_val = zeros (M, 1);
for  (i = 1:  M) f_val (i) = f((i-1) * h); end ;
% calculation of the fft
dft_val = fft (f_val);
% calculation of the fourier coefficients
fcoef = zeros (M, 1);
for n = 1:  M
  i = 1 + mod (- (n-1), M); % we must reverse the clues
  fcoef(n) = h * dft_val (i);
end
% draw an evolution of the solution
for t = [0.01,0.02,0.03,0.04,0.05,0.06]
  xx = [xx, real (solve_eq (0, fcoef))];
end
plot (xx);
```

Program 80 Procedure `solve_eq`

```
function u = solve_eq (t, fcoef_val)
prec = 300; h = 1 / prec; % plot precision
u = zeros (prec + 1, 1);
M = length (fcoef_val); % size of solution
v = [0:  M / 2, -M / 2 + 1:  -1] '; % coefficient frequencies
% calculate the solution
for i = 0:  prec
  x = i * h;
  w = exp (-2.0 * pi * pi * t * v.  * v + 2.0i * pi * x * v).  * fcoef_val;
  u (i + 1) = sum (w);
end
```

Program 81 Procedure `f`

```
function y = f(x)
if  (x <0.3) y = 0;
elseif  (x <0.7) y = 1;
else y = 0;
end
```

Appendix C

Programs Maple

This chapter brings together all the Maple programs in the book. Each program is a separate `.dsw` file. They have often been cut into several pieces for the sake of clarity.

C.1 Transform over a finite field

The file `fft-finite-body.msw` successively realizes

1. a search for irreducible factors of $X^n - 1$ over a finite field $\mathbb{F}_p$ (we have taken $p = 2$). With the command `alias`, we name α a primitive root of the unit.
2. a naive implementation of the Fourier transform on the cyclotomic field $\mathbb{F}_{p^r}$. In the case where n is of the form 2^s, it is possible to implement a recursive version of the algorithm. This is done for the Fourier transform on a ring, appendix C.2.
3. a test on a randomly drawn $f \in \mathbb{F}_p^n$ vector. We can see that $\hat{f} \notin \mathbb{F}_p^n$, since we have to do the calculations in a cyclotomic extension of $\mathbb{F}_p$.

C.2 Transform on a ring

The Maple `fft-ring.msw` program calculates a transform of size n to value in a ring $\mathbb{Z}/m\mathbb{Z}$ for a judiciously chosen integer m (in accordance with the explanations given in Paragraph 6.2). We chose n of the form 2^s, which allows to implement a recursive algorithm of type FFT. We use an intermediate function, `FFT _rec`, which allows to update the main root of the unit with each call.

C.3 Multiplication of large integers

The Maple `mult-large-integers.mws` program allows you to calculate the product of two integers represented by their decomposition in a given base b. This program uses the constants n and m as well as the function `xFFT` which is in the file `fft-ring.msw`, 83. Here are the different things that can be found in this program.

1. We first calculate an optimal value of b, so as to satisfy $n(b-1)^2 < m$.
2. Then several very useful functions are defined (to pass from the representation in the form of number to that in the form of vector).
3. The `prod _entiers` function calculates the convolution product of the two vectors, then propagates the carry.

Program 82 File `fft-finite-body.msw`

Les paramètres pour faire un TFD de taille n fixé sur $\mathbb{F}_p$:

```
> with (numtheory): n := 16: p := 3:
```

Liste des facteurs de $X^n - 1$. Choix d'un facteur irréductible de degré r et de la racine primive associée : on constate que r est bien l'ordre de p dans $(\mathbb{Z}/n\mathbb{Z})^*$.

```
> liste_div := op(Factor( cyclotomic(n,X) ) mod p );
> P := liste_div[1];
> alias( alpha = RootOf(P) ):
```

$$liste\_div := X^4 + 2\,X^2 + 2,\ X^4 + X^2 + 2$$

$$P := X^4 + 2\,X^2 + 2$$

Transformée de Fourier, version $O(n^2)$:

```
> TFD := proc (f, signe)
>    local  res, formule;
>    # pour plus de lisibilité, on écrit à part la formule de TFD :
>    formule := 'f[l+1]*alpha^(-signe*(k-1)*l)';
>    res := [ seq( sum( formule, 'l'=0..n-1 ) mod p , k=1..n)  ];
>    if  signe=-1 then  res := 1/n*res mod p end  if ;
>    return (Normal(res) mod p);
> end  proc :
```

Test simple :

```
> hasard := rand(0..(p-1)):
> x := [seq( hasard(), i=1..n )];
> y := TFD(x,1);  # TFD(x) n'est plus à coefficients dans F_2.
> evalb( x = TFD(y,-1) ); # Mais on retombe bien sur nos pattes.
```

$$x := [0,\ 2,\ 0,\ 2,\ 1,\ 2,\ 2,\ 2,\ 1,\ 1,\ 1,\ 0,\ 0,\ 2,\ 2,\ 1]$$

$$y := [1,\ 2\,\alpha^3 + 2\,\alpha + \alpha^2 + 2,\ 1,\ 2\,\alpha^3 + \alpha + 2\,\alpha^2,\ \alpha^2 + 1,\ \alpha^3 + \alpha,\ 1,\ 2\,\alpha^3 + \alpha,\ 1,$$
$$\alpha^3 + \alpha + \alpha^2 + 2,\ 2,\ \alpha^3 + 2\,\alpha + 2\,\alpha^2,\ 2\,\alpha^2 + 2,\ 2\,\alpha^3 + 2\,\alpha,\ 2,\ \alpha^3 + 2\,\alpha]$$

true

Program 83 File `fft-ring.msw`

Définition des paramètres de la transformée

```
> s := 4:  n := 2^s:  m := 2^(2^(s-1)) + 1:
```

Sous-procédure récursive :

```
> FFT_rec := proc (f, signe, zeta)
>    local  nn, n1, s, t, r;
>    nn := nops(f); n1 := nn/2; # taille du vecteur
>    if  nn=1 then  return (f) end  if ; # fin de l'algorithme
>    # construction des deux sous-vecteurs de taille n1
>    s := [ seq(f[2*k+1], k=0..n1-1) ];
>    t := [ seq(f[2*k], k=1..n1) ];
>    # calcul des deux sous-FFT :
>    s := FFT_rec(s, signe, zeta^2 mod m);
>    t := FFT_rec(t, signe, zeta^2 mod m);
>    # mixage des deux résultats
>    a := seq( s[k]+zeta^(-signe*(k-1))*t[k] mod m, k=1..n1 );
>    b := seq( s[k]-zeta^(-signe*(k-1))*t[k] mod m, k=1..n1 );
>    r := [a,b];
>    return (r);
> end  proc :
```

Procédure principale (attention, le nom FFT est protégé en Maple ...) :

```
> xFFT := proc (f, signe)
>    local  r;
>    r := FFT_rec(f,signe,2);
>    if  signe=-1 then  r := 1/n*r mod m;
>    else  r; end  if
> end  proc :
```

Un test :

```
> hasard := rand(0..m-1):
> x := [seq( hasard(), i=1..n )];
> y := xFFT(x,+1);
> evalb( x = xFFT(y,-1) ); # On retombe bien sur nos pattes.
```

$$x := [179, 220, 230, 218, 49, 253, 197, 218, 67, 177, 136, 127, 190, 106, 210, 255]$$

$$y := [5, 250, 28, 179, 190, 157, 195, 216, 198, 11, 13, 43, 5, 59, 49, 238]$$

$$\mathit{true}$$

4. Finally, a test is performed. Of course, the usefulness of these functions is to multiply whole numbers that Maple cannot handle (because they are too large), which is not the case in this test (because we make Maplecheck that the product is right).

C.4 Decoding BCH codes

This Maple program uses the `FFT` function defined in the C.1 program. This procedure should therefore be copied at the start of the program. The program has been split into three parts:

– Part 1 (program 85): search for the irreducible factors of $X^n - 1$ on $\mathbb{F}_2$, and construction of the generator polynomial of the BCH code.

– Part 2 (program 86): definition of routines to manipulate code words both in the form of vectors and polynomials, to generate words at random.

– Part 3 (program 87): the first part of the decoding algorithm, we calculate the values of $\sigma_1, \ldots, \sigma_t$.

– Part 4 (program 88): the second part of the decoding algorithm, we calculate the values of $\hat{\epsilon_0}$, $\epsilon_{2\hat{t}+1}, \ldots, \epsilon_{\hat{n}-1}$.

Program 84 File `mult-large-integers.msw`

b désigne la base de calcul. Il faut que $n(b-1)^2 < m$.

```
> b := floor( evalf(sqrt(m/n))+1 ):
```

Calcule le produit point à point :

```
> cw_mult := proc (a,b)
>    [seq( a[i]*b[i], i=1..n )]:
> end  proc :
```

Transforme un entier en vecteur :

```
> number2vector := proc (x)
>    local  N, res, i, r, q, xx:
>    N := floor( log(x)/log(b) )+1;
>    res := []:  xx := x:
>    for  i from  1 to  N do
>       xx := iquo(xx,b,'r'):
>       res := [op(res), r]:
>    end :
>    return (res):
> end  proc :
```

Transforme un vecteur en entier :

```
> vector2number := proc (v)
>    add(v[k]*b^(k-1), k=1..nops(v));
> end  proc :
```

Calcule le produit de convolution :

```
> convol := proc (f,g)
>    xFFT( cw_mult(xFFT(f,1),xFFT(g,1)), -1):
> end  proc :
```

Calcule le produit de deux entiers représentés sous forme de vecteurs de taille n. Attention, les $n/2$ dernières entrées des vecteurs doivent être nulles.

```
> prod_entiers := proc (x,y)
>    local  res, i:
>    res := convol(x,y):
>    for  i from  1 to  n-1 do
>       res[i] := irem(res[i],b,'q'):
>       res[i+1] := res[i+1]+q;
>    end :
>    return (res):
> end  proc :
```

Un test :

```
> hasard := rand(0..b-1):
> xx := [seq( hasard(), i=1..n/2 ), seq(0, i=1..n/2)];
> yy := [seq( hasard(), i=1..n/2 ), seq(0, i=1..n/2)];
> x := vector2number(xx):  y := vector2number(yy);
> zz := prod_entiers(xx,yy);
> evalb( vector2number(zz) = x*y ); # il doit y avoir égalité ...
```

$$xx := [4, 0, 0, 3, 3, 1, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0]$$

$$yy := [3, 0, 4, 1, 4, 2, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0]$$

$$y := 55853$$

$$zz := [2, 2, 1, 1, 3, 3, 2, 2, 1, 0, 3, 3, 2, 4, 2, 0]$$

$$\mathit{true}$$

Program 85 File `decoding-bch.msw` part 1

r : degrés des facteurs irréductibles de $X^n - 1$ sur $\mathbb{F}_2$; t : capacité de correction.

```
> with (numtheory):  with (linalg):
> n := 15:  t := 3:  delta:=2*t+1:
```

Liste des facteurs de $X^n - 1$choix d'un facteur irréductible de degré r et de la racine primive associée : on constate que r est bien l'ordre de p dans $(\mathbb{Z}/n\mathbb{Z})^*$.

```
> liste_div := op(Factor( X^n-1 ) mod 2 ):
> P := liste_div[2];
> alias( alpha = RootOf(P) ):
```

$$P := X^4 + X^3 + 1$$

Calcule le polynôme générateur du code de distance prescrite $2\,t + 1$, le PPCM des polynômes minimaux des α^i, pour $i = 1, \ldots, 2t$

```
> calc_gen := proc ()
>    local  result, Q, i, liste_pol_rest:
>    result := P: # on sait déjà que P est dans le PPCM
>    liste_pol_rest := {liste_div} minus {P}:
>    # alpha^2 est racine de P, donc on peut le sauter
>    for  i from  3 to 2*t do
>    for  Q in  liste_pol_rest do
>       if  Eval(Q, X=alpha^i) mod 2 = 0 then
>       result := result*Q:
>       liste_pol_rest:=liste_pol_rest minus {Q}:  break:
>    end  if :  end  do :  end  do :
>    result := Expand(result) mod 2
> end  proc :
```

Polynôme générateur et dimension du code :

```
> G := sort( calc_gen() ); d := n - degree(G);
```

$$G := X^{10} + X^9 + X^8 + X^6 + X^5 + X^2 + 1$$
$$d := 5$$

Program 86 File `decoding-bch.msw` part 2

[...] Suite du script précédent

Calcule le mot de taille n (liste de 0/1) correspondant à un polynôme de degré $n-1$

```
> Mot := proc (Q)
>    [seq( coeff(Q, X,it), it=0..n-1 )]
> end  proc :
```

Calcule le polynôme de degré $n-1$ correspondant à un mot de taille n

```
> Pol := proc (mot)
>    sum(mot[it]*X^(it-1), it=1..n);
> end  proc :
```

Calcule le syndrôme d'indice i, i.e. $\mathrm{P}(\alpha^i)$:

```
> Syndi := proc (pol, i)
>    Eval(pol, X = alpha^i) mod 2;
> end  proc :
```

Calcule un vecteur aléatoire avec `nb_erreurs` erreurs

```
> Aleat := proc (nb_erreurs)
>    local  hasard:
>    hasard := rand(1..(n-1)):
>    Mot( add(X^hasard(), i=1..nb_erreurs) mod 2 );
> end  proc :
```

Calcule un mot du code au hasard

```
> MotCode := proc ()
>    local  Q;
>    Q := Randpoly(d-1, X) mod 2;
>    Q := Expand( Q*G ) mod 2;
>    Mot(Q);
> end  proc :
```

On simule une transmission avec erreur :

```
> mot_code := MotCode();
> mot_transmis := mot_code + Aleat(3) mod 2;
> p_recu := Pol(mot_transmis);
```

$$mot\_code := [0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1]$$

$$mot\_transmis := [0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]$$

$$p\_recu := X + X^2 + X^3 + X^4 + X^8$$

Program 87 File `decoding-bch.msw` part 3

[...] Suite du script précédent

1ère partie : Résolution des équations pour $i = n - t$ ... $n - 1$ pour trouver σ[1] ... σ[t]

Calcule de l'équation polynomiale à résoudre (attention, on note la ε *transformée de Fourier* de l'erreur) :

```
>  eqn := (1+add(sigma[i]*Z^i,i=1..t))*
>            (add(epsilon[n-i]*Z^i,i=1..n)):
>  eqn := rem(eqn,Z^n-1,Z,'q');  # l'équation est modulo Z^n-1
```

$$\begin{aligned}
eqn &:= (\sigma_1\,\varepsilon_2 + \sigma_3\,\varepsilon_4 + \varepsilon_1 + \sigma_2\,\varepsilon_3)\,Z^{14} + (\sigma_3\,\varepsilon_5 + \varepsilon_2 + \sigma_1\,\varepsilon_3 + \sigma_2\,\varepsilon_4)\,Z^{13}\\
&+ (\varepsilon_3 + \sigma_2\,\varepsilon_5 + \sigma_1\,\varepsilon_4 + \sigma_3\,\varepsilon_6)\,Z^{12} + (\sigma_3\,\varepsilon_7 + \sigma_2\,\varepsilon_6 + \sigma_1\,\varepsilon_5 + \varepsilon_4)\,Z^{11}\\
&+ (\sigma_1\,\varepsilon_6 + \varepsilon_5 + \sigma_3\,\varepsilon_8 + \sigma_2\,\varepsilon_7)\,Z^{10} + (\sigma_1\,\varepsilon_7 + \sigma_3\,\varepsilon_9 + \varepsilon_6 + \sigma_2\,\varepsilon_8)\,Z^{9}\\
&+ (\sigma_3\,\varepsilon_{10} + \sigma_2\,\varepsilon_9 + \sigma_1\,\varepsilon_8 + \varepsilon_7)\,Z^{8} + (\sigma_1\,\varepsilon_9 + \varepsilon_8 + \sigma_2\,\varepsilon_{10} + \sigma_3\,\varepsilon_{11})\,Z^{7}\\
&+ (\varepsilon_9 + \sigma_1\,\varepsilon_{10} + \sigma_2\,\varepsilon_{11} + \sigma_3\,\varepsilon_{12})\,Z^{6} + (\sigma_2\,\varepsilon_{12} + \varepsilon_{10} + \sigma_1\,\varepsilon_{11} + \sigma_3\,\varepsilon_{13})\,Z^{5}\\
&+ (\sigma_1\,\varepsilon_{12} + \varepsilon_{11} + \sigma_3\,\varepsilon_{14} + \sigma_2\,\varepsilon_{13})\,Z^{4} + (\sigma_1\,\varepsilon_{13} + \sigma_2\,\varepsilon_{14} + \varepsilon_{12} + \sigma_3\,\varepsilon_0)\,Z^{3}\\
&+ (\sigma_1\,\varepsilon_{14} + \varepsilon_{13} + \sigma_3\,\varepsilon_1 + \sigma_2\,\varepsilon_0)\,Z^{2} + (\varepsilon_{14} + \sigma_1\,\varepsilon_0 + \sigma_3\,\varepsilon_2 + \sigma_2\,\varepsilon_1)\,Z + \sigma_3\,\varepsilon_3 + \varepsilon_0\\
&+ \sigma_2\,\varepsilon_2 + \sigma_1\,\varepsilon_1
\end{aligned}$$

Calcule les équations à résoudre, liste les valeurs de ε connues, pour $i = 1$ ... $2\,t$, puis évalue les équations :

```
>  list_eqn1 := {seq( coeff(eqn,Z,i), i=n-t..n-1 )}:
>  epsilon_connu := {seq( epsilon[i] = Syndi(p_recu,i), i=1..2*t )};
>  eqn_eval1 := eval(list_eqn1, epsilon_connu);
```

$$\begin{aligned}
&epsilon\_connu :=\\
&\{\varepsilon_2 = \alpha^3 + \alpha^2,\ \varepsilon_1 = \alpha^3,\ \varepsilon_6 = \alpha^3 + 1,\ \varepsilon_5 = 1,\ \varepsilon_4 = \alpha^3 + \alpha^2 + \alpha + 1,\ \varepsilon_3 = \alpha^3 + \alpha + 1\}
\end{aligned}$$

$$\begin{aligned}
eqn\_eval1 &:= \{\sigma_1\,(\alpha^3 + \alpha^2 + \alpha + 1) + \alpha^3 + \alpha + 1 + \sigma_3\,(\alpha^3 + 1) + \sigma_2,\\
&\sigma_2\,(\alpha^3 + \alpha^2 + \alpha + 1) + \alpha^3 + \alpha^2 + \sigma_3 + \sigma_1\,(\alpha^3 + \alpha + 1),\\
&\alpha^3 + \sigma_3\,(\alpha^3 + \alpha^2 + \alpha + 1) + \sigma_2\,(\alpha^3 + \alpha + 1) + \sigma_1\,(\alpha^3 + \alpha^2)\}
\end{aligned}$$

Met sous forme matricielle les équations :

```
>  m1 := matrix(t,t):
>  b1 := vector(t):
>  i := 1:
>  for  eq in  eqn_eval1 do
>     for  j from  1 to  t do
>        m1[i,j] := coeff(eq,sigma[j],1);
>     end  do :
>     b1[i] := eval( eq, [seq(sigma[k]=0,k=1..t)] );
>     i := i+1:
>  end  do :
```

Calcule les valeurs de σ en résolvant le système :

```
>  sigma_val := Linsolve(m1,b1) mod 2:
>  sigma_connu := { seq(sigma[i]=sigma_val[i], i = 1..t) };
```

$$sigma\_connu := \{\sigma_1 = \alpha^3 + 1,\ \sigma_2 = \alpha^3 + \alpha^2 + \alpha,\ \sigma_3 = \alpha^2 + 1\}$$

Program 88 File `decoding-bch.msw` part 4

2e partie : Résolution des équations pour $i = 0 \ldots n-2t-1$ pour trouver ε[0], ε[2t+1] ... ε[n-1]

Calcule les équations pour $i = 0 \ldots n-2t-1$, puis les évalue :

```
> list_eqn2 := {seq( coeff(eqn,Z,i), i=0..n-2*t-1 )}:
> eqn_eval2 := eval(list_eqn2, epsilon_connu):
> eqn_eval2 := eval(eqn_eval2, sigma_connu);
```

$$\begin{aligned}
&\mathit{eqn\_eval2} := \{\%1\,\varepsilon_{14} + \varepsilon_{12} + (\alpha^3+1)\,\varepsilon_{13} + (\alpha^2+1)\,\varepsilon_0,\\
&\varepsilon_{14} + (\alpha^3+1)\,\varepsilon_0 + (\alpha^2+1)\,(\alpha^3+\alpha^2) + \%1\,\alpha^3,\\
&\varepsilon_{13} + (\alpha^3+1)\,\varepsilon_{14} + \%1\,\varepsilon_0 + (\alpha^2+1)\,\alpha^3,\\
&\varepsilon_0 + \%1\,(\alpha^3+\alpha^2) + (\alpha^3+\alpha+1)\,(\alpha^2+1) + (\alpha^3+1)\,\alpha^3,\\
&\varepsilon_{11} + (\alpha^2+1)\,\varepsilon_{14} + \%1\,\varepsilon_{13} + (\alpha^3+1)\,\varepsilon_{12},\ \varepsilon_{10} + (\alpha^3+1)\,\varepsilon_{11} + (\alpha^2+1)\,\varepsilon_{13} + \%1\,\varepsilon_{12},\\
&\varepsilon_8 + (\alpha^3+1)\,\varepsilon_9 + \%1\,\varepsilon_{10} + (\alpha^2+1)\,\varepsilon_{11},\ (\alpha^2+1)\,\varepsilon_{12} + \%1\,\varepsilon_{11} + (\alpha^3+1)\,\varepsilon_{10} + \varepsilon_9,\\
&(\alpha^2+1)\,\varepsilon_{10} + \varepsilon_7 + \%1\,\varepsilon_9 + (\alpha^3+1)\,\varepsilon_8\}\\
&\%1 := \alpha^3+\alpha^2+\alpha
\end{aligned}$$

Met sous forme matricielle les équations :

```
> # les indices de epsilon a calculer
> epsilon_indices := [0,seq(i, i=2*t+1..n-1)]:
> m2 := matrix(n-2*t,n-2*t):
> b2 := vector(n-2*t):
> i := 1:
> for  eq in  eqn_eval2 do
>    j:= 1:
>    for  index in  epsilon_indices do
>       m2[i,j] := coeff(eq,epsilon[index],1):
>       j := j+1;
>    end  do :
>    b2[i]:=eval(eq,[epsilon[0]=0,seq(epsilon[k]=0,k=2*t+1..n-1)]);
>    i := i+1:
> end  do :
```

Calcule les valeurs de ε[0], ε[2t+1] ... ε[n-1], puis regroupe toutes les valeurs :

```
> epsilon_val := Linsolve(m2,b2) mod 2:
> epsilon_val := [epsilon_val[1], seq(Syndi(p_recu,it),it=1..2*t),
> seq(epsilon_val[it],it=2..n-2*t)];
```

$$\begin{aligned}
&\mathit{epsilon\_val} := [1,\ \alpha^3,\ \alpha^3+\alpha^2,\ \alpha^3+\alpha+1,\ \alpha^3+\alpha^2+\alpha+1,\ 1,\ \alpha^3+1,\ \alpha^3+\alpha+1,\ \alpha^3+\alpha,\\
&\alpha^3+\alpha^2+\alpha,\ 1,\ \alpha^3+\alpha^2+\alpha,\ \alpha^3+\alpha^2+1,\ \alpha^3+\alpha^2+1,\ \alpha^3+1]
\end{aligned}$$

On peut maintenant déterminer l'erreur par transformée de Fourier inverse :

```
> erreurs := Normal( TFD(epsilon_val,-1) ) mod 2;
> mot_corrige := mot_transmis - erreurs mod 2:
> evalb( mot_corrige = mot_code );
```

$$\mathit{erreurs} := [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1]$$

$$\mathit{true}$$

Bibliography

[1] Jonathan L. Alperin and Rowen B. Bell. *Groups and representations.* Springer Verlag, 1995.

[2] Jörg Arndt. *Algorithms for programmers.* 2002.

[3] Michael Artin. *Algebra.* Prentice Hall, 1991.

[4] Laszlo Babai. The Fourier transform and equations over finite abelian groups. *Technical Report, TR-2001-01*, 1989.

[5] David H. Bailey. The computation of π to 29.360.000 decimal digits using Borweins' quartically convergent algorithm. *Mathematics of Computation*, 50(181):283–296, 1987.

[6] David H. Bailey and Paul N. Swarztrauber. The fractional Fourier transform and applications. *RNR Technical Report*, 1995.

[7] Riccardo Bernardini and Jelena Kovacevic. Designing local orthogonal bases on finite groups I: abelian case. *Journal of Fourier Analysis and Applications*, 6(1):1–23, 2000.

[8] Dan Boney. Learning using group representations. *Proc. COLT 1995*, pages 418–426, 1995.

[9] John P. Boyd. *Chebyshev and Fourier Spectral Methods.* Dover Publications, 2000.

[10] Ronald Bracewell. *The Hartley transform.* Oxford University Press, 1986.

[11] E. Oran Brigham. *Fast Fourier Transform and its Applications.* Prentice Hall, 1988.

[12] Charles S. Burrus. Notes on the FFT. 1997.

[13] Cagatay Candan, M. Alper Kutay, and Haldun M. Ozaktas. The discrete fractional Fourier transform. *IEEE Trans. on Signal Processing*, 48:1329–1337, 2000.

[14] Gianfranco Cariolaro, Tomaso Erseghe, Peter Kraniauskas, and Nicola Laurenti. Multiplicity of fractional Fourier transforms and their relationships. *IEEE Transactions on Signal Processing*, 48(1):227–241, 2000.

[15] Henri Cartan. *Théorie élémentaire des fonctions d'une variable complexe.* Hermann, 1961.

[16] Philippe G. Ciarlet. *Introduction à lanalyse numérique et à l'optimisation.* Dunod, 1990.

[17] Jon Claerbout. *Fundamentals of Geophysical Data Processing.* McGraw Hill, 1976.

[18] Henri Cohen. *A course in computational algebraic number theory.* Springer Verlag, 1991.

[19] Michael J. Collins. *Representations and characters of finite groups.* Cambridge University Press, 1990.

[20] Thomas Cormen, Charles Leiserson, and Ronald Rivest. *Introduction à l'algorithmique.* Dunod, 1992.

[21] David Cox, John Little, and Donald O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Algebraic Geometry and Commutative Algebra, 2nd ed.* Springer Verlag, 1996.

[22] Jean Pierre Demailly. *Analyse numérique et équations différentielles.* EDP, 1996.

[23] Michel Demazure. *Cours d'algèbre. Primalité, divisibilité, codes.* Cassini, 1997.

[24] Gilbert Demengel. *Transformées de Fourier généralisées.* Ellipses, 1999.

[25] Persi Diaconis. *Group Representations in Probability and Statistics.* IMS Lecture Series 11, Institute of Mathematical Statistics, Hayward, California, 1988.

[26] Bradley W. Dickinson and Kenneth Steiglitz. Eigenvectors and functions of the discrete Fourier transform. *IEEE Transaction on Acoustics, Speech, and Signal Processing*, 30(1):25–31, 1982.

[27] David L. Donoho and Philip B. Stark. Uncertainty principles and signal recovery. *SIAM Journal of Applied Mathematics*, 49:906–931, 1989.

[28] Harry Dym and Henry Mc Keam. *Fourier series and integrals.* Academic press, 1972.

[29] Noam D. Elkies. Lattices, linear codes, and invariants, part I. *Notices of the AMS*, 47(10):1238–1245, 2002.

[30] Kristin Flornes, Alex Grossman, Matthias Holschneider, and Bruno Torrésani. Wavelets on discrete fields. *Applied and Computational Harmonic Analysis*, 1:137–147, 1994.

[31] Lemmermeyer Franz. *Reciprocity Laws : From Euler to Eisenstein.* Springer Verlag, 2000.

[32] Matteo Frigo and Steven G. Johnson. FFTW: An adaptive software architecture for the FFT. *ICASSP conference proceedings*, 3:1381–1384, 1998.

[33] Christine Froidevaux, Marie-Claude Gaudel, and Michèle Soria. *Types de données et algorithmes.* Ediscience international, 1990.

[34] William Fulton and Joe Harris. *Representation theory : a first course.* Springer Verlag, 1991.

[35] Roe Goodman and Nolan R. Wallach. *Representations and invariants of the classical groups.* Cambridge University Press, 1999.

[36] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics.* Addison-Wesley, 1994.

[37] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the Fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, 1991.

[38] Tsit Yuen Lam. Representations of finite groups: A hundred years, part I. *Notices of the AMS*, 45(3):368–372, 1998.

[39] Tsit Yuen Lam. Representations of finite groups: A hundred years, part II. *Notices of the AMS*, 45(4):465–474, 1998.

[40] Serge Lang. *Algebra.* Addison-Wesley, 1965.

[41] Philippe Langevin. *Les sommes de caractères et la formule de Poisson dans la théorie des codes, des séquences et des fonctions booléennes.* Université de Toulon, 1999.

[42] Reinhard C. Laubenbacher. Eisenstein misunderstood geometric proof of the quadratique reciprocity theorem. *College Mathematics Journal*, 25:29–34, 1994.

[43] J.P. Lewis. Fast normalized cross-correlation. *Vison Interface*, pages 120–123, 1995.

[44] Rudolf Lidl and Harald Niederreiter. *Finite fields.* Cambridge University Press, 1983.

[45] Larry S. Liebovitch, Yi Tao, Angelo T. Todorov, and Leo Levine. Is there an error correcting code in the base sequence in DNA ? *Biopphysical Journal*, 71:1539-1544, 1996.

[46] Jessie MacWilliams and Neil Sloane. *The theory of error-correcting codes, Part I.* North-Holland, 1977.

[47] Stéphane Mallat. *Une exploration des signaux en ondelettes.* Editions de l'école Polytechnique, 2000.

[48] David K. Malsen and Daniel N. Rockmore. Generalized FFTs - a survey of some recent results. *Proc. 1995 DIMACS Workshop in Groups and Computation*, 28:183–238, 1995.

[49] Jean-Yves Ouvrard. *Probabilité 2.* Cassini, 2000.

[50] Odile Papini and Jacques Wolfman. *Algèbre discrète et codes correcteurs.* Springer Verlag, 1995.

[51] Enes Passalic and Thomas Johansson. Further results on the relation between nonlinearity and resiliency for boolean functions. *Conference on Cryptography and Coding*, 1994.

[52] Daniel Perrin. *Cours d'algèbre.* Ellipses, 1996.

[53] William Press. *Numerical Recipes in C : the art of computer programming.* Cambridge University Press, 1988.

[54] Ramis, Deschamps, and Odoux. *Tome 1 : algèbre.* Masson, 1979.

[55] Daniel N. Rockemore. The FFT - an algorithm the whole family can use. *Computing in Science and Engineering*, 2(1):60–64, 2000.

[56] Oscar S. Rothaus. On bent functions. *Journal of combinatorial theory*, 20A:300–305, 1976.

[57] Walter Rudin. *Analyse réelle et complexe.* Dunod, 1987.

[58] Pierre Samuel. *Théorie des nombres.* Hermann, 1967.

[59] Jean-Pierre Serre. *Representations lineaires des groupes finis.* Hermann, 1966.

[60] Jean-Pierre Serre. *Cours d'arithmétique.* PUF, 1970.

[61] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing.* California Technical Publishing, 1997.

[62] Gilbert Strang. The discrete cosine transform. *SIAM Review*, 41(1):135–147, 1999.

[63] Paul N. Swarztrauber and Roland A. Sweet. The Fourier and cyclic reduction for solving Poisson's equation. *Handbook of Fluid Fynamics and Fluid Machinery*, 1996.

[64] Paul N. Swarztrauber, Roland A. Sweet, William L. Briggs, Van-Emden Henson, and James Otto. Bluestein's FFT for arbitrary N on the hypercube. *Parallel Computing*, Vol.17:607–617, 1991.

[65] Audrey Terras. *Fourier analysis on finite groups and applications.* London Mathematical Society, 1999.

[66] Ronald F. Ullmann. An algorithm for the fast Hartley transform. *Stanford Exploration Project*, SEP-38, 1984.

[67] Michael Unser, Akram Aldroubi, and Murray Eden. Fast B-spline transforms for continous image representation and interpolation. *IEE transaction on pattern analysis and machine intelligence*, 13(3):277–285, 1991.

[68] Martin Vetterli and Pierre Duhamel. Split-radix algorithms for length-p^m DFT's. *IEEE Transactions on Accoustic, Speech, Signal Processing*, 37(1):57–64, 1989.

[69] André Warusfel. *Structures algébriques finies.* Hachette, 1971.

[70] Robert Wich. *Z transform, theory and applications.* D.Reidel Publishing Compaghy, 1987.

[71] Herbert S. Wilf. *Generatingfunctionology.* Academic Press, 1990.

[72] Zuily and Queffelec. *Eléments d'analyse pour l'agrégation.* Masson, 1995.