UNIVERSITAT DE BARCELONA

The Discrete Fourier Transform and Some of its Applications on Image Processing

Núria Sánchez Font

Adviser: F. Javier Soria de Diego

Master in Advanced Mathematics
University of Barcelona
Barcelona, January 2019

Contents

Chapter 1

Introduction

The digital image processing is a field devoted to the enhancement of a given image in order to make it more suitable for a specific purpose as well as extracting some of its particular attributes. Its applications cover a wide range of different areas: from easing visualisation of a certain body feature (by sharpening, reducing noise, improving contrast, ...) for medical recognition to edge detection so as to localise a predetermined shape and, therefore, identifying a particular element such a car plate. However, the processing may be very time consuming, which is an important drawback. Thus, the main challenge that digital image processing faces is the development of strategies that result in an accurate outcome while speeding up the whole process. One of the tools introduced so as to achieve such purposes is the Discrete Fourier Transform.

The aim of this project is to develop the *Discrete Fourier Transform (DFT)* in order to see some of its applications on image processing. As we are going to discuss in detail throughout the project, the role played by the DFT is the following. The space we are going to deal with is $\ell^2(\mathbb{Z}/M \times \mathbb{Z}/N)$ (defined in Chapter 2) since, as we are going to see in Chapter 5, a *digital image* is regarded as an element of such space. Moreover, we will focus on that type of image processing that consists in applying a *linear* and *translation-invariant* transformation to the given picture. As we will develop in Section 5.2, such processing is equivalent to perform a *convolution* between the image and some other element of $\ell^2(\mathbb{Z}/M \times \mathbb{Z}/N)$ ($filtering$). Since M and N can be very large (even millions), performing convolution appears to be not only too expensive computationally, but even unfeasible if M and N are too large. And here is where the DFT comes to play. One of its key properties is the fact that it can turn a convolution into the *Inverse Discrete Fourier Transform (IDFT)* of a component-wise product between the DFT of two matrices (Proposition 2.1.13). In other words, $f * g = (\hat{f} .* \hat{g})^\vee$, where $f, g \in \ell^2(\mathbb{Z}/M \times \mathbb{Z}/N)$ and $*$ and $.*$ stands for convolution and component-wise product respectively. But, if it is true that a component-wise product is, by far, less time consuming, the computation of the DFT and IDFT requires much more time, since they have to be calculated by means of a change-of-basis matrix (Section 2.2). However, as we will see, such computations can be speeded up by implementing the *Fast Fourier Transform (FFT)* algorithm. Thus,

on the one hand, the introduction of the DFT leads to a faster way of processing an image. In addition, as we will see in Chapter 2, the DFT is a change of basis from the Euclidean to the *Fourier basis*, which as discussed in Section 2.3, allows a *frequency analysis* of the signal. Hence, on the other hand, the DFT splits a signal into its frequency components providing key information of such signal, as illustrated in Sections 5.4 and 5.5. Therefore, the project has two main goals: studying the filtering process and developing some examples of image processing (Chapter 5) as well as analysing the reduction entailed by the implementation of the FFT algorithm (Chapters 3 and 4). In order to do so, it is going to be fundamental the development of the main properties of the DFT (Chapter 2).

Let us now describe in more detail how the project is structured. Chapter 2, which is an extension to two dimensions of [2], gathers all the essential results concerning the DFT. It begins by setting the required tools to define the Discrete Fourier Transform along with its main properties (Section 2.1) whereas Sections 2.2 and 2.3 are more focused on its features as a change of basis. On the one hand, the former provides the change-of-basis matrix (which is going to be fundamental when analysing the computational advantage of the FFT) while, on the other hand, the latter explains the frequency information encoded in the new coefficients. As we will see, the new basis (*Fourier basis*) splits a signal into its *frequency components* in such a way that the coefficients reveal the strength of each component required in making up the signal. This is very useful when recovering a signal, as Section 5.4 will illustrate.

In Section 3.1 we introduce the Fast Fourier Transform algorithm following the approaches set in [2] and [8]. Since this algorithm halves a vector iteratively, the most favourable situation is that in which it is applied to a vector whose size is a power of two. That is the reason why, as also mentioned at the beginning of Chapter 3, in most of the references the FFT is only applied to vectors of such length. However, since we are interested in the computational advantage that results from the use of this algorithm, not only have we studied its implementation in such vectors (Section 3.2) but to any other vector of even length (Section 3.3). In both cases we provide the number of operations required by the algorithm in order to compare it with the ones needed by the change-of-basis matrix. Recall that in Section 2.2 we constructed that matrix and, therefore, we know how many operations are carried out when multiplying the matrix by the vector. By comparing such numbers we are going to obtain the reduction entailed by the use of the FFT. As we will see, such reduction is greater when the vector has length a power of two which leads us to study the *padding* (Section 3.4). In that section we obtain an upper bound for the error caused when padding a vector of any size, not necessarily even, (Proposition 3.4.3) which leads to a uniform bound for normalised vectors (Corollary 3.4.5). Finally, we particularise the study of the padding to vectors of even size but not a power of two in order to stablish under which hypothesis it is worth to pad a vector, that is, under which hypothesis the operations required for computing the FFT is smaller for the padded vector. Moreover, we find out the reduction that ensues from padding.

Keeping in mind the fact that the FFT can only be applied to vectors of even

size, we begin Chapter 4 by stating the result in which this algorithm is based [2, Proposition 4.1.1]. Then, we use this result to extend the FFT to a vector of odd but non-prime size, which we call *extended algorithm* (Section 4.2). In addition, this extension allows us to introduce the *mixed algorithm* (Section 4.3), which improves the FFT algorithm for vectors of even size but not a power of two. In both cases, we compute the number of operations required by the new algorithms and compare it with the change-of-basis matrix in order to analyse the reduction. Next, in Sections 4.4.2 and 4.4.3 we compare the number of operations required by the extended or mixed algorithm with the ones needed by the FFT applied to the padded vector so as to find out the reduction. Moreover, we study, in each case, the conditions under which padding is worth doing. Finally, we focus on vectors of prime size, whose only way of computing its transform is by means of the change-of-basis matrix. In Section 4.4.1, we examine how many operations are needed when padding it and, as we will see, the conclusion is that padding such vectors is always less time-consuming.

Notice that in this two chapters we have distinguished different situations depending on the size of the vector discussing, in each case, the quickest way of computing its Fourier Transform (algorithms/padding). Section 4.5 summarises it and provides the plots depicting the number of operations required by each algorithm (Figure 4.6) and by the FFT applied to the padded vectors (Figure 4.7), both depending on the length of the vectors.

Recall that we mentioned the fact that most of the references in the FFT are only concerned with vectors whose size is a power of two since, otherwise, the vectors are just padded. Therefore, the main purpose of Chapters 3 and 4 is to go deeper on those aspects usually skipped in most of the discussions: the extension of the FFT so as to be applied to a vector of any size, the study of the reduction that ensues from the implementation of the algorithms (which involves counting the number of operations required) and a deeper discussion on the padding issue. The latter includes bounding its error, finding the hypothesis under which it is actually time-saving and, finally, studying the consequent reduction.

Eventually, Chapter 5 focuses on digital image processing. Once it has established the setting and introduced the notation (Section 5.1) it then exposes the basics of filtering, where the results exposed are an extension to two dimensions of the ones appearing in [2]. Moreover, it shows the computational reduction arisen from the use of the DFT instead of the convolution. Finally, this section ends with an introduction to Sections 5.3, 5.4 and 5.5, since some previous assumptions must be made. Such sections provide examples of digital image processing that illustrate the important role played by the Fourier Transform when filtering [3]. In addition, Section 5.4 shows how to take advantage of the information encoded in the Fourier basis in order to eliminate periodic noise from a picture. Finally, Section 5.6 develops an algorithm that can be used so as to reconstruct an image where a uniformly distributed set of its pixels has been removed [1]. Although it is usually used with other types of transformations (such the *Discrete Cosine Transform* as illustrated in [1] and [7]), we are going to implement it with the DFT and see its performance

when a whole square of pixels is missing. Along this chapter we are going to illustrate the approaches set with some particular examples, which were implemented using MATLAB. The codes can be found in the Appendix.

Chapter 2

Discrete Fourier Transform

In this chapter we are going to set a certain Hilbert space over which we will define a linear transformation, the *Discrete Fourier Transform*, which will be regarded as the change of coordinates from the Euclidean basis to the so called *Fourier basis*. By extending to two dimensions the results exposed in [2], we are going to see some of its properties such as the fact that it is invertible or its performance over other transformations like are *translation* or *convolution*. In particular, we are going to pay special attention to the construction of the change-of-basis matrix. Finally, we will see how the Fourier basis allows a frequency analysis of the vectors, fact that is going to be fundamental later on.

2.1 Fundamentals of the Discrete Fourier Transform

In this section we are going to define the so called *Discrete Fourier Transform* over a certain space of functions as well as developing its main properties.

Let us first consider the set

$$\ell^2\left(\mathbb{Z}/(M)\times\mathbb{Z}/(N)\right)=\{f:\mathbb{Z}\times\mathbb{Z}\to\mathbb{C}\text{ such that }f(m,n)=f(m+k_1M,n+k_2N)\\ \text{for all }k_1,k_2\in\mathbb{Z}\},$$

where M and N are positive integers. That is the set of discrete functions of two integer variables taking values in the set of complex numbers such that they have period M with respect to the first variable and period N with respect to the second variable. Notice that it is enough to have these functions defined in the set $\{0,1,\dots,M-1\}\times\{0,1,\dots,N-1\}$ since, then, they can be extended periodically. Thus, any function $f\in\ell^2\left(\mathbb{Z}/(M)\times\mathbb{Z}/(N)\right)$ can be represented by the finite matrix

$$f=\begin{pmatrix} f(0,0) & \dots & f(0,N-1)\\ \vdots & & \vdots\\ f(M-1,0) & \dots & f(M-1,N-1)\end{pmatrix}. \tag{2.1.1}$$

In order to simplify notation, we are going to denote $\mathbb{Z}_M := \mathbb{Z}/(M)$ for any integer M.

The set $\ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ is a vector space over $\mathbb{C}$ with the usual componentwise addition and scalar multiplication. Moreover, with the following inner product

$$\langle f, g\rangle = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)\overline{g(m,n)},\ \ \forall f, g \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$$

it becomes a Hilbert space. Therefore, it is a Banach space with the norm

$$||f|| = \langle f, f\rangle^{1/2} = \left(\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} |f(m,n)|^2\right)^{1/2}.$$

The standard basis for $\ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ is the Euclidean one, defined as $E = \{E_{p,q}\}_{0\leq p\leq M-1, 0\leq q\leq N-1}$, where

$$E_{p,q}(m,n) = \begin{cases} 1, \text{ if } m = p \text{ and } n = q \\ 0, \text{ otherwise} \end{cases},$$

for all $m \in \{0, \ldots, M-1\}$, $n \in \{0, \ldots, N-1\}$ and, then, extended periodically.

Apart from this one, there is another basis that is going to be fundamental for our purposes and is the so called *Fourier basis*. Let us now define it.

Proposition 2.1.1. *Consider the subset $F = \{F_{p,q}\}_{0\leq p\leq M-1, 0\leq q\leq N-1} \subset \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ with*

$$F_{p,q}(m,n) = \frac{1}{\sqrt{MN}} e^{2\pi i p m/M} e^{2\pi i q n/N}$$

for all $m, n \in \mathbb{Z}$. Then, the set F is an orthonormal basis of the space $\ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$. We will refer to the set F as the Fourier basis of the space $\ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$.

Proof. Let us first check that F is certainly contained in $\ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$. If we take $k_1, k_2 \in \mathbb{Z}$, then

$$\begin{aligned} F_{p,q}(m + k_1 M, n + k_2 N) &= \frac{1}{\sqrt{MN}} e^{2\pi i p(m+k_1 M)/M} e^{2\pi i q(n+k_2 N)/N} \\ &= \frac{1}{\sqrt{MN}} e^{2\pi i p m/M} e^{2\pi i p k_1} e^{2\pi i q n/N} e^{2\pi i q k_2} \\ &= \frac{1}{\sqrt{MN}} e^{2\pi i p m/M} e^{2\pi i q n/N} = F_{p,q}(m,n). \end{aligned}$$

So the functions of F are M-periodic with respect to the first variable and N-periodic with respect to the second variable, as we wanted to prove.

Let us now see that the set F is orthonormal. Take $0 \leq p_1, p_2 \leq M-1$ and $0 \leq q_1, q_2 \leq N-1$. Then,

$$\begin{aligned}
\langle F_{p_1,q_1}, F_{p_2,q_2} \rangle &= \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} F_{p_1,q_1}(m,n) \overline{F_{p_2,q_2}(m,n)} \\
&= \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} e^{2\pi i p_1 m/M} e^{2\pi i q_1 n/N} e^{-2\pi i p_2 m/M} e^{-2\pi i q_2 n/N} \\
&= \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} e^{2\pi i (p_1-p_2) m/M} e^{2\pi i (q_1-q_2) n/N} \\
&= \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \left(e^{2\pi i (p_1-p_2)/M}\right)^m \left(e^{2\pi i (q_1-q_2)/N}\right)^n \\
&= \frac{1}{MN} \left[\sum_{m=0}^{M-1} \left(e^{2\pi i (p_1-p_2)/M}\right)^m\right] \left[\sum_{n=0}^{N-1} \left(e^{2\pi i (q_1-q_2)/N}\right)^n\right].
\end{aligned}$$

At this point we will distinguish two cases depending on the values of p_1, p_2, q_1 and q_2:

(i) If $p_1 = p_2$ and $q_1 = q_2$, then

$$\langle F_{p_1,q_1}, F_{p_1,q_1} \rangle = \frac{1}{MN} \left(\sum_{m=0}^{M-1} 1\right) \left(\sum_{n=0}^{N-1} 1\right) = 1.$$

(ii) Suppose now that $p_1 \neq p_2$ or $q_1 \neq q_2$. Without loss of generality, assume $p_1 \neq p_2$. Since $0 \leq p_1, p_2 \leq M-1$, then $-M+1 \leq p_1 - p_2 \leq M-1$ and hence $e^{2\pi i (p_1-p_2)/M} \neq 1$. Therefore, the sum is a geometric series, so

$$\begin{aligned}
\sum_{m=0}^{M-1} \left(e^{2\pi i (p_1-p_2)/M}\right)^m &= \frac{1 - \left(e^{2\pi i (p_1-p_2)/M}\right)^M}{1 - e^{2\pi i (p_1-p_2)/M}} = \frac{1 - e^{2\pi i (p_1-p_2)}}{1 - e^{2\pi i (p_1-p_2)/M}} \\
&= \frac{1-1}{1 - e^{2\pi i (p_1-p_2)/M}} = 0,
\end{aligned}$$

where we have used the fact that $p_1 - p_2 \in \mathbb{Z}$. This implies that $\langle F_{p_1,q_1}, F_{p_2,q_2} \rangle = 0$.

This proves that the set F is orthonormal. In particular, the elements of F are linearly independent. This result together with the fact that the space $\ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ has dimension $M \cdot N$ implies that F generates all the space $\ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$. So, F is an orthonormal basis of $\ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$. □

Remark 2.1.2. An important feature of this basis is the fact that

$$F_{p,q}(m,n) = F_{m,n}(p,q)$$

for all $0 \leq m, p \leq M-1$ and $0 \leq n, q \leq N-1$.

So far we have seen that $\ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ is a Hilbert space with an orthonormal basis called the Fourier basis. So we can now define the Fourier coefficients of the elements of the space.

Definition 2.1.3. We define the *Discrete Fourier Transform (DFT)* as the map

$$\begin{array}{ccc} \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N) & \stackrel{\wedge}{\to} & \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N) \\ f & \mapsto & \hat{f} \end{array}$$

where $\hat{f}(p,q) = \langle f, F_{p,q} \rangle$ for all $p \in \{0, \ldots, M-1\}$, $q \in \{0, \ldots, N-1\}$ and then extended periodically.

Notice that, due to the periodicity of the Fourier basis, this definition holds for all $p, q \in \mathbb{Z}$. Therefore, if $f \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ and $p, q \in \mathbb{Z}$, then

$$\hat{f}(p,q) = \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) e^{-2\pi i p m/M} e^{-2\pi i q n/N}. \tag{2.1.2}$$

Observe that $|\hat{f}(p,q)|$ is the length of the projection of f onto the vector $F_{p,q}$ of the basis.

Remark 2.1.4. In particular, if $f \in \ell^2(\mathbb{Z}_M)$, which means taking $N = 1$, then, for all $p \in \mathbb{Z}$

$$\hat{f}(p) = \frac{1}{\sqrt{M}} \sum_{m=0}^{M-1} f(m) e^{-2\pi i p m/M}.$$

Proposition 2.1.5. *Let $f, g \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$. Then, the following identities hold:*

(i) $\langle f, g \rangle = \langle \hat{f}, \hat{g} \rangle$.

(ii) $||f|| = ||\hat{f}||$. *This is the so called* Parseval's identity.

Proof.

$$\begin{aligned}
(i)\ \langle \hat{f}, \hat{g} \rangle &= \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \hat{f}(p,q) \overline{\hat{g}(p,q)} \\
&= \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \left(\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) \overline{F_{p,q}(m,n)} \right) \left(\overline{\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} g(m,n) \overline{F_{p,q}(m,n)}} \right) \\
&= \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \left(\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) \overline{F_{p,q}(m,n)} \right) \left(\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \overline{g(m,n)} F_{p,q}(m,n) \right) \\
&= \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \left(\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \sum_{r=0}^{M-1} \sum_{s=0}^{N-1} f(m,n) F_{p,q}(m,n) \overline{g(r,s)} F_{p,q}(r,s) \right) \\
&= \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \sum_{r=0}^{M-1} \sum_{s=0}^{N-1} f(m,n) \overline{g(r,s)} F_{m,n}(p,q) F_{r,s}(p,q)
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\sum_{r=0}^{M-1}\sum_{s=0}^{N-1} f(m,n)\overline{g(r,s)}\left(\sum_{p=0}^{M-1}\sum_{q=0}^{N-1} F_{m,n}(p,q)F_{r,s}(p,q)\right)\\
&= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\sum_{r=0}^{M-1}\sum_{s=0}^{N-1} f(m,n)\overline{g(r,s)}\langle F_{m,n}, F_{r,s}\rangle\\
&= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)\overline{g(m,n)} = \langle f, g\rangle,
\end{aligned}
$$

where we have used Remark 2.1.2.

(*ii*) $||f||^2 = \langle f, f\rangle = \langle \hat{f}, \hat{f}\rangle = ||\hat{f}||^2$, where we have used (*i*). □

Notice that, trivially, Parseval's identity implies that the Discrete Fourier Transform $\ell^2(\mathbb{Z}_M \times \mathbb{Z}_N) \xrightarrow{\wedge} \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ is an isometry.

Proposition 2.1.6. *Let $f \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$. Then, f can be expressed in terms of the Fourier basis as follows*

$$
f = \sum_{p=0}^{M-1}\sum_{q=0}^{N-1} \hat{f}(p,q)F_{p,q}. \tag{2.1.3}
$$

Proof. Let $k_1, k_2 \in \mathbb{Z}$. Then,

$$
\begin{aligned}
\sum_{p=0}^{M-1}\sum_{q=0}^{N-1} \hat{f}(p,q)F_{p,q}(k_1,k_2) &= \sum_{p=0}^{M-1}\sum_{q=0}^{N-1}\langle f, F_{p,q}\rangle F_{p,q}(k_1,k_2)\\
&= \sum_{p=0}^{M-1}\sum_{q=0}^{N-1}\left(\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)\overline{F_{p,q}(m,n)}\right)F_{p,q}(k_1,k_2)\\
&= \sum_{p=0}^{M-1}\sum_{q=0}^{N-1}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)\overline{F_{p,q}(m,n)}F_{p,q}(k_1,k_2)\\
&= \sum_{p=0}^{M-1}\sum_{q=0}^{N-1}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)\overline{F_{m,n}(p,q)}F_{k_1,k_2}(p,q)\\
&= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)\left(\sum_{p=0}^{M-1}\sum_{q=0}^{N-1} F_{k_1,k_2}(p,q)\overline{F_{m,n}(p,q)}\right)\\
&= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)\langle F_{k_1,k_2}, F_{m,n}\rangle = f(k_1,k_2).
\end{aligned}
$$

We have used Remarks 2.1.2 and, in the last equality, the fact that the Fourier basis is orthonormal. □

Proposition 2.1.7. *The Discrete Fourier Transform is injective and surjective and, therefore, invertible.*

Proof. (i) Let $f, g \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ such that $\hat{f}(p,q) = \hat{g}(p,q)$ for all $p, q \in \mathbb{Z}$. Then, by equation (2.1.3),

$$f = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \hat{f}(p,q) F_{p,q} = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \hat{g}(p,q) F_{p,q} = g$$

(ii) Let $H \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$. Define $h = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} H(p,q) F_{p,q} \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$. By equation (2.1.3), $h = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \hat{h}(p,q) F_{p,q}$. Since $F = \{F_{p,q}\}_{0 \le p \le M-1, 0 \le q \le N-1}$ is a basis, then $H(p,q) = \hat{h}(p,q)$ for all $p \in \mathbb{Z}_M, q \in \mathbb{Z}_N$. □

So the DFT $\ell^2(\mathbb{Z}_M \times \mathbb{Z}_N) \xrightarrow{\wedge} \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ is invertible. We denote the *Inverse Discrete Fourier Transform (IDFT)* as $\ell^2(\mathbb{Z}_M \times \mathbb{Z}_N) \xrightarrow{\vee} \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$. As we have seen, the inverse is given by equation (2.1.3). That is, if $H \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$, then

$$\check{H} = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} H(p,q) F_{p,q} \tag{2.1.4}$$

or componentwise

$$\check{H}(m,n) = \frac{1}{\sqrt{MN}} \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} H(p,q) e^{2\pi i p m/M} e^{2\pi i q n/N}, \tag{2.1.5}$$

for all $m, n \in \mathbb{Z}$.

Notice that $\check{H}$ is certainly in $\ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$. Let $k_1, k_2 \in \mathbb{Z}$. Then,

$$\begin{aligned} \check{H}(m + k_1 M, n + k_2 N) &= \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} H(p,q) F_{p,q}(m + k_1 M, n + k_2 N) \\ &= \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} H(p,q) F_{p,q}(m,n) = \check{H}(m,n). \end{aligned}$$

So if $f \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$, then $(\hat{f})^\vee = f$. Although the IDFT is given by equation (2.1.5), it can also be expressed in terms of the DFT as the following theorem shows.

Theorem 2.1.8 (Inversion)**.** *Let $f \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$. Then, $\hat{\hat{f}} \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ and it satisfies that, for any $p, q \in \mathbb{Z}$,*

$$\hat{\hat{f}}(p,q) = f(-p,-q).$$

Notice that in order to carry out the IDFT of a function, it is enough to compute its DFT and then change the sign of the components.

Proof. We will begin by checking $\hat{\hat{f}} \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$. Take $k_1, k_2 \in \mathbb{Z}$. Then,

$$\begin{aligned}\hat{\hat{f}}(p+k_1M, q+k_2N) &= \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \hat{f}(m,n) e^{-2\pi i(p+k_1M)m/M} e^{-2\pi i(q+k_2N)n/N} \\ &= \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \hat{f}(m,n) e^{-2\pi ipm/M} e^{-2\pi iqn/N} = \hat{\hat{f}}(p,q).\end{aligned}$$

Let us first take $p \in \{0, -1, -2, ..., -(M-1)\}$ and $q \in \{0, -1, -2, ..., -(N-1)\}$. Then,

$$\begin{aligned}\hat{\hat{f}}(p,q) &= \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \hat{f}(m,n) e^{-2\pi ipm/M} e^{-2\pi iqn/N} \\ &= \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \left[\frac{1}{\sqrt{MN}} \sum_{r=0}^{M-1}\sum_{s=0}^{N-1} f(r,s) e^{\frac{-2\pi imr}{M}} e^{\frac{-2\pi ins}{N}}\right] e^{\frac{-2\pi ipm}{M}} e^{\frac{-2\pi iqn}{N}} \\ &= \frac{1}{\sqrt{MN}} \sum_{r=0}^{M-1}\sum_{s=0}^{N-1} f(r,s) \left[\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} e^{-2\pi imr/M} e^{-2\pi ins/N} e^{-2\pi ipm/M} e^{-2\pi iqn/N}\right] \\ &= \sum_{r=0}^{M-1}\sum_{s=0}^{N-1} f(r,s) \left[\frac{1}{M}\sum_{m=0}^{M-1} e^{-2\pi imr/M} e^{-2\pi ipm/M}\right] \left[\frac{1}{N}\sum_{n=0}^{N-1} e^{-2\pi ins/N} e^{-2\pi iqn/N}\right] \\ &= \sum_{r=0}^{M-1}\sum_{s=0}^{N-1} f(r,s) \langle F_{-p}, F_r\rangle \langle F_{-q}, F_s\rangle = f(-p,-q),\end{aligned}$$

where we have used the fact that

$$\langle F_{-p}, F_r\rangle = \begin{cases} 0, & \text{if } r \neq -p \\ 1, & \text{if } r = -p \end{cases} \qquad \langle F_{-q}, F_s\rangle = \begin{cases} 0, & \text{if } s \neq -q \\ 1, & \text{if } s = -q \end{cases}$$

since $\{F_p\}_{0\leq p\leq M-1}$ is the Fourier basis in $\ell^2(\mathbb{Z}_M)$. Now, if $p, q \in \mathbb{Z}$, then there exists $k_1, k_2 \in \mathbb{Z}$ such that $-(M-1) \leq p + k_1M \leq 0$ and $-(N-1) \leq q + k_2N \leq 0$. Therefore, using the periodicity of f and $\hat{\hat{f}}$, we get

$$\hat{\hat{f}}(p,q) = \hat{\hat{f}}(p+k_1M, q+k_2N) = f(-p-k_1M, -q-k_2N) = f(-p,-q),$$

as wished. □

Let us now define the *translation* of a function and see how the transform behaves over it.

Definition 2.1.9. Let $f \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ and $k_1, k_2 \in \mathbb{Z}$. We define the *translation* of f by k_1 and k_2 as

$$(R_{k_1,k_2} f)(m,n) = f(m-k_1, n-k_2).$$

Notice that $R_{k_1,k_2}f \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ as well.

Proposition 2.1.10. *Let $f \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ and $p, q, k_1, k_2 \in \mathbb{Z}$. Then,*

$$(R_{k_1,k_2}f)^\wedge(p,q) = e^{-2\pi i p k_1/M} e^{-2\pi i q k_2/N} \hat{f}(p,q).$$

Proof. Let $p, q, k_1, k_2 \in \mathbb{Z}$. Then,

$$\begin{aligned}
(R_{k_1,k_2}f)^\wedge(p,q) &= \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} (R_{k_1,k_2}f)(m,n) e^{-2\pi i p m/M} e^{-2\pi i q n/N} \\
&= \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m-k_1, n-k_2) e^{-2\pi i p m/M} e^{-2\pi i q n/N} \\
&= \frac{1}{\sqrt{MN}} \sum_{r=-k_1}^{M-1-k_1} \sum_{s=-k_2}^{N-1-k_2} f(r,s) e^{-2\pi i p (r+k_1)/M} e^{-2\pi i q (s+k_2)/N} \\
&= e^{-2\pi i p k_1/M} e^{-2\pi i q k_2/N} \frac{1}{\sqrt{MN}} \sum_{r=-k_1}^{M-1-k_1} \sum_{s=-k_2}^{N-1-k_2} f(r,s) e^{-2\pi i p r/M} e^{-2\pi i q s/N}.
\end{aligned}$$

Take $a, b \in \mathbb{Z}$ such that $k_1 + aM \in \{0, 1, ..., M-1\}$ and $k_2 + bN \in \{0, 1, ..., N-1\}$. Then, we carry out the changes of variable $u = r - aM$ and $v = s - bM$, so, we get

$$\begin{aligned}
(R_{k_1,k_2}f)^\wedge(p,q) &= e^{\frac{-2\pi i p k_1}{M}} e^{\frac{-2\pi i q k_2}{N}} \frac{1}{\sqrt{MN}} \sum_{u=-k_1-aM}^{M-1-k_1-aM} \sum_{v=-k_2-bN}^{N-1-k_2-bN} \big[f(u+aM, v+bN) \\
&\quad \times e^{-2\pi i p (u+aM)/M} e^{-2\pi i q (v+bN) N} \big] \\
&= e^{-2\pi i p k_1/M} e^{-2\pi i q k_2/N} \frac{1}{\sqrt{MN}} \sum_{u=-k_1-aM}^{M-1-k_1-aM} \sum_{v=-k_2-bN}^{N-1-k_2-bN} \big[f(u,v) \\
&\quad e^{-2\pi i p u/M} e^{-2\pi i q v/N} \big].
\end{aligned}$$

By defining $\ell = k_1 + aM$ and $t = k_2 + bN$ (observe that $\ell \in \{0, 1, ..., M-1\}$ and $t \in \{0, 1, ..., N-1\}$), we can rewrite the previous expression as

$$(R_{k_1,k_2}f)^\wedge(p,q) = e^{-2\pi i p k_1/M} e^{-2\pi i q k_2/N} \frac{1}{\sqrt{MN}} \sum_{u=-\ell}^{M-1-\ell} \underbrace{\sum_{v=-t}^{N-1-t} f(u,v) e^{-2\pi i p u/M} e^{-2\pi i q v/N}}_{A}. \tag{2.1.6}$$

We are now going to compute A. Keeping in mind that u is fixed, we will distinguish two cases.

- If $t = 0$, then

$$A = \sum_{v=0}^{N-1} f(u,v) e^{-2\pi i p u/M} e^{-2\pi i q v/N}.$$

- If $1 \leq t \leq N-1$, then

$$\begin{aligned} A &= \sum_{v=-t}^{-1} f(u, v+N) e^{-2\pi i p u/M} e^{-2\pi i q(v+N)/N} + \sum_{v=0}^{N-1-t} f(u,v) e^{-2\pi i p u/M} e^{-2\pi i q v/N} \\ &= \sum_{c=N-t}^{N-1} f(u,c) e^{-2\pi i p u/M} e^{-2\pi i q c/N} + \sum_{v=0}^{N-t-1} f(u,v) e^{-2\pi i p u/M} e^{-2\pi i q v/N} \\ &= \sum_{v=0}^{N-1} f(u,v) e^{-2\pi i p u/M} e^{-2\pi i q v/N}, \end{aligned}$$

  where in the second equality we have considered, in the first addend, the change of variable $c = v + N$.

Therefore, for all $t \in \{0, 1, ..., N-1\}$,

$$A = \sum_{v=0}^{N-1} f(u,v) e^{-2\pi i p u/M} e^{-2\pi i q v/N}.$$

Hence, equation (2.1.6) can be rewritten as follows

$$(R_{k_1,k_2} f)^{\wedge}(p,q) = e^{-2\pi i p k_1/M} e^{-2\pi i q k_2/N} \frac{1}{\sqrt{MN}} \underbrace{\sum_{u=-\ell}^{M-1-\ell} \sum_{v=0}^{N-1} f(u,v) e^{-2\pi i p u/M} e^{-2\pi i q v/N}}_{B}. \tag{2.1.7}$$

Let us now compute B. Again, we are going to distinguish two cases.

- If $\ell = 0$, then

$$B = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} f(u,v) e^{-2\pi i p u/M} e^{-2\pi i q v/N}.$$

- If $1 \leq \ell \leq M-1$, then

$$\begin{aligned} B &= \sum_{u=-\ell}^{-1} \sum_{v=0}^{N-1} f(u+M, v) e^{\frac{-2\pi i p(u+M)}{M}} e^{\frac{-2\pi i q v}{N}} + \sum_{u=0}^{M-1-\ell} \sum_{v=0}^{N-1} f(u,v) e^{\frac{-2\pi i p u}{M}} e^{\frac{-2\pi i q v}{N}} \\ &= \sum_{c=M-\ell}^{M-1} \sum_{v=0}^{N-1} f(c,v) e^{-2\pi i p c/M} e^{-2\pi i q v/N} + \sum_{u=0}^{M-1-\ell} \sum_{v=0}^{N-1} f(u,v) e^{-2\pi i p u/m} e^{-2\pi i q v/N} \\ &= \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} f(u,v) e^{-2\pi i p u/M} e^{-2\pi i q v/N}, \end{aligned}$$

  where in the second equality we have considered the change of variable $c = u + M$ in the first addend.

Therefore, for all $\ell \in \{0, 1, ..., M-1\}$,

$$B = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} f(u,v) e^{-2\pi i p u/M} e^{-2\pi i q v/N}.$$

Hence, equation (2.1.7) can be rewritten as follows

$$\begin{aligned}(R_{k_1,k_2} f)^\wedge(p,q) &= e^{-2\pi i p k_1/M} e^{-2\pi i q k_2/N} \frac{1}{\sqrt{MN}} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} f(u,v) e^{-2\pi i p u/M} e^{-2\pi i q v/N} \\ &= e^{-2\pi i p k_1/M} e^{-2\pi i q k_2/N} \hat{f}(p,q),\end{aligned}$$

which is the result that we wanted to prove. □

Remark 2.1.11. Notice that when proving

$$\sum_{r=-k_1}^{M-1-k_1} \sum_{s=-k_2}^{N-1-k_2} f(r,s) e^{-2\pi i p r/M} e^{-2\pi i q s/N} = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} f(u,v) e^{-2\pi i p u/M} e^{-2\pi i q v/N},$$

we have just used the fact that, for any $k_1, k_2 \in \mathbb{Z}$,

$$f(u + k_1 M, v + k_2 N) e^{-2\pi i p (u+k_1 M)/M} e^{-2\pi i q (v+k_2 N)/N} = f(u,v) e^{-2\pi i p u/M} e^{-2\pi i q v/N}.$$

Therefore, for all $h \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$, it holds that

$$\sum_{m=-a}^{M-1-a} \sum_{n=-b}^{N-1-b} h(m,n) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} h(m,n),$$

for any $a, b \in \mathbb{Z}$.

As we will see in the following chapters, processing an image consists in carrying out a *convolution* between the image and what is called a *filter*. So let us now define this operation between two elements of $\ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$.

Definition 2.1.12. Let $f, g \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$. Then, we define the *convolution* as

$$(f * g)(m,n) = \frac{1}{\sqrt{MN}} \sum_{j=0}^{M-1} \sum_{k=0}^{N-1} f(m-j, n-k) g(j,k)$$

for all $m, n \in \mathbb{Z}$.

Observe that $f * g \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$. One of the most important features of the convolution is the following property.

Proposition 2.1.13. *Let $f, g \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$. Then,*

$$(f * g)^\wedge(p,q) = \hat{f}(p,q) \hat{g}(p,q)$$

for all $p, q \in \mathbb{Z}$.

Proof.

$$
\begin{aligned}
(f*g)\hat{}(p,q) &= \frac{1}{\sqrt{MN}}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}(f*g)(m,n)e^{-2\pi ipm/M}e^{-2\pi iqn/N}\\
&= \frac{1}{\sqrt{MN}}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left(\frac{1}{\sqrt{MN}}\sum_{j=0}^{M-1}\sum_{k=0}^{N-1}f(m-j,n-k)g(j,k)\right)\\
&\quad\times e^{-2\pi ipm/M}e^{-2\pi iqn/N}\\
&= \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\sum_{j=0}^{M-1}\sum_{k=0}^{N-1}f(m-j,n-k)g(j,k)e^{-2\pi ip(m-j)/M}e^{-2\pi ipj/M}\\
&\quad\times e^{-2\pi iq(n-k)/N}e^{-2\pi iqk/N}\\
&= \frac{1}{MN}\sum_{j=0}^{M-1}\sum_{k=0}^{N-1}g(j,k)e^{-2\pi ipj/M}e^{-2\pi iqk/N}\\
&\quad\times\left(\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}f(m-j,n-k)e^{-2\pi ip(m-j)/M}e^{-2\pi iq(n-k)/N}\right)
\end{aligned}
$$

By doing the changes of variable $r = m - j$ and $s = n - k$ and using the periodicity of the exponentials and the function f, we obtain

$$
\begin{aligned}
&\frac{1}{MN}\sum_{j=0}^{M-1}\sum_{k=0}^{N-1}g(j,k)e^{-2\pi ipj/M}e^{-2\pi iqk/N}\\
&\quad\times\left(\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}f(m-j,n-k)e^{-2\pi ip(m-j)/M}e^{-2\pi iq(n-k)/N}\right)\\
&= \frac{1}{MN}\sum_{j=0}^{M-1}\sum_{k=0}^{N-1}g(j,k)e^{-2\pi ipj/M}e^{-2\pi iqk/N}\left(\sum_{r=-j}^{M-1-j}\sum_{s=-k}^{N-1-k}f(r,s)e^{-2\pi ipr/M}e^{-2\pi iqs/N}\right)\\
&= \frac{1}{MN}\sum_{j=0}^{M-1}\sum_{k=0}^{N-1}g(j,k)e^{-2\pi ipj/M}e^{-2\pi iqk/N}\left(\sum_{r=0}^{M-1}\sum_{s=0}^{N-1}f(r,s)e^{-2\pi ipr/M}e^{-2\pi iqs/N}\right)\\
&= \left(\frac{1}{\sqrt{MN}}\sum_{r=0}^{M-1}\sum_{s=0}^{N-1}f(r,s)e^{-2\pi ipr/M}e^{-2\pi iqs/N}\right)\\
&\quad\times\left(\frac{1}{\sqrt{MN}}\sum_{j=0}^{M-1}\sum_{k=0}^{N-1}g(j,k)e^{-2\pi ipj/M}e^{-2\pi iqk/N}\right) = \hat{f}(p,q)\hat{g}(p,q).
\end{aligned}
$$

□

2.2 The Discrete Fourier Transform as a change of basis

The equation (2.1.3) shows that, given $f \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$, the coordinates of this function with respect to the Fourier basis are the coefficients $\{\hat{f}(p,q)\}_{0\leq p\leq M-1, 0\leq q\leq N-1}$. Thus, we can regard the Discrete Fourier Transform as the change of coordinates from the Euclidean basis to the Fourier basis. It is clear that the Discrete Fourier Transform is a linear map, so it can be represented by a matrix, which is going to be the change of basis matrix.

Let $f \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$. As we have already mentioned, f is completely determined by its values in the interval $\{0,\dots,M-1\}\times\{0,\dots,N-1\}$, hence it is represented by the matrix given in the expression (2.1.1). Consider v and $\hat{v}$ as the vectors obtained by joining the columns of the f and $\hat{f}$ matrices respectively. That is

$$v = \begin{pmatrix} f(0,0) \\ \vdots \\ f(M-1,0) \\ f(0,1) \\ \vdots \\ f(M-1,1) \\ \vdots \\ f(0,N-1) \\ \vdots \\ f(M-1,N-1) \end{pmatrix}_{MN\times 1} \qquad \hat{v} = \begin{pmatrix} \hat{f}(0,0) \\ \vdots \\ \hat{f}(M-1,0) \\ \hat{f}(0,1) \\ \vdots \\ \hat{f}(M-1,1) \\ \vdots \\ \hat{f}(0,N-1) \\ \vdots \\ \hat{f}(M-1,N-1) \end{pmatrix}_{MN\times 1} \tag{2.2.1}$$

So v is the vector of coordinates of f with respect to the Euclidean basis and $\hat{v}$ is the vector of coordinates of f with respect to the Fourier basis.

Let us now compute the change of basis matrix, that is W such that

$$\hat{v} = Wv.$$

By equation (2.1.2),

$$\begin{aligned} \hat{f}(p,q) &= \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n) e^{-2\pi i pm/M} e^{-2\pi i qn/N} \\ &= \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n) \left(e^{-2\pi i/M}\right)^{pm} \left(e^{-2\pi i/N}\right)^{qn}. \end{aligned}$$

Denote, for any integer K,

$$\omega_K := e^{-2\pi i/K}. \tag{2.2.2}$$

Then, we can rewrite the previous expression as

$$\hat{f}(p,q) = \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)\omega_M^{pm}\omega_N^{qn}. \tag{2.2.3}$$

Notice that $f(m,n) = v(nM+m)$ and $\hat{f}(p,q) = \hat{v}(qM+p)$. Hence,

$$\hat{v}(qM+p) = \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} v(nM+m)\omega_M^{pm}\omega_N^{qn}.$$

Therefore,

$$W(qM+p, nM+m) = \frac{\omega_M^{pm}\omega_N^{qn}}{\sqrt{MN}}, \tag{2.2.4}$$

for all $0 \leq m, p \leq M-1$ and $0 \leq n, q \leq N-1$. Notice that W is symmetric

$$W(qM+p, nM+m) = \frac{\omega_M^{pm}\omega_N^{qn}}{\sqrt{MN}} = \frac{\omega_M^{mp}\omega_N^{nq}}{\sqrt{MN}} = W(nM+m, qM+p).$$

This result is formalised in the following proposition.

Proposition 2.2.1. *Let $f \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ and v and $\hat{v}$ be the finite vectors obtained by joining the columns of f and $\hat{f}$ respectively in the way shown in (2.2.1). Then,*

$$\hat{v} = Wv, \tag{2.2.5}$$

where W is the matrix given by equation (2.2.4). That is, the DFT is given by the matrix W.

Similarly, we can find a change of basis matrix for the IDFT as shown in the following proposition.

Proposition 2.2.2. *Let $H \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$. Consider h and $\check{h}$ to be the finite vectors obtained by joining the columns of H and $\check{H}$ respectively in the way shown in (2.2.1). Then,*

$$\check{h} = \overline{W}h, \tag{2.2.6}$$

where W is the matrix given by equation (2.2.4). That is, the IDFT is given by the matrix $\overline{W}$.

Proof. By using the expression (2.2.2) in equation (2.1.5), we obtain

$$\check{H}(m,n) = \frac{1}{\sqrt{MN}} \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} H(p,q)\overline{w}_M^{pm}\overline{w}_N^{qn}.$$

Let h and $\check{h}$ be the vectors obtained by joining the columns of the matrices H and $\check{H}$ respectively. Then, $H(p,q) = h(qM+p)$ and $\check{H}(m,n) = \check{h}(nM+m)$. It allows us to rewrite the previous expression in the following way

$$\check{h}(nM+m) = \frac{1}{\sqrt{MN}} \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} h(qM+p)\overline{w}_M^{pm}\overline{w}_N^{qn}.$$

Therefore, $\check{h} = Ah$, where A is the matrix given by

$$A(nM+m, qM+p) = \frac{\overline{w}_M^{pm}\overline{w}_N^{qn}}{\sqrt{MN}} = \overline{W}(qM+p, nM+m).$$

However, since W is symmetric, $A(nM+m, qM+p) = \overline{W}(nM+m, qM+p)$. □

2.3 Fourier coefficients and frequencies

As seen in (2.1.3), if $f \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$, then it can be expressed in the Fourier basis

$$f = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \hat{f}(p,q) F_{p,q}.$$

Thus the Fourier coefficient $\hat{f}(p,q)$ is the weight of the vector $F_{p,q}$ used in making up f and we are now going to see that it provides a frequency analysis of the function. Recall that for all $m, n \in \mathbb{Z}$

$$F_{p,q}(m,n) = \frac{1}{\sqrt{MN}} e^{2\pi i pm/M} e^{2\pi i qn/N}.$$

We will first analyse the behaviour of the discrete function

$$e^{2\pi i pm/M} = \cos(2\pi pm/M) + i \sin(2\pi pm/M)$$

with $m \in \{0, \ldots, M-1\}$. Let us first consider $\cos(2\pi pm/M)$ as a continuous function of m defined in the interval $[0, M]$ and with p fixed. This function carries out p full cycles of the cosine wave as m goes continuously from 0 to M. That is, the function has frequency p. Therefore, as the value of p increases, the same does the frequency of the cosine function. However, if m is defined in the discrete set $\{0, 1, ..., M-1\}$ (as is the case in which we are interested) instead of the interval $[0, M]$, then p is not necessarily the frequency. Let us see this fact.

Take $p < M/2$ a non-negative integer and define $k = M - p$. Observe that k is the symmetric element of p with respect to $M/2$ in the interval $[0, M]$. Then,

$$\begin{aligned} \cos(2\pi km/M) &= \cos(2\pi(M-p)m/M) = \cos(2\pi m - 2\pi pm/M) \\ &= \cos(-2\pi pm/M) = \cos(2\pi pm/M). \end{aligned}$$

This implies that although $\cos(2\pi km/M)$ has higher frequency with respect to $\cos(2\pi pm/M)$ when $m \in [0, M]$, the samples that we obtain when taking $m \in \{0, 1, ..., M-1\}$ are the same. In other words, the discrete functions $\cos(2\pi km/M)$ and $\cos(2\pi pm/M)$ taking $m \in \{0, 1, ..., M-1\}$ are the same and, in particular, their frequencies are the same, p.

As an example we can consider $M = 17$, $p = 4$ and $k = M - p = 13$ as shown in Figure 2.1. In this case it is clear that, although $\cos(2\pi 13m/17)$ has higher frequency than $\cos(2\pi 4m/17)$ when m takes real values, they are the same function when m takes integer values. Consequently, both functions are regarded as having period 4.

This behaviour is summarized by the fact that as p increases from 0 to the closest integer to $M/2$, the discrete function $\cos(2\pi pm/M)$ oscillates more and more rapidly (p is the frequency). However, as p increases from $[M/2] + 1$ to $M - 1$, the discrete function $\cos(2\pi pm/M)$ oscillates more and more slowly ($M - p$ is the frequency). Therefore, higher frequencies are obtained for values of p near $M/2$ and lower frequencies for values of p near 0 and $M - 1$.

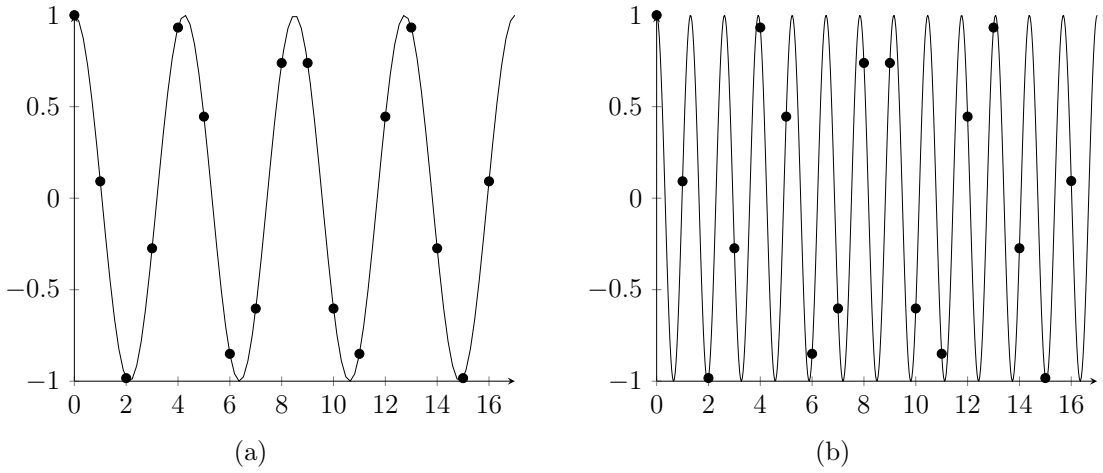


Figure 2.1 (a) Overlapping of the continuous and discrete functions $\cos(2\pi 4m/17)$ with $m \in$ [0,17] and $m \in \{0, 1, ..., 16\}$. **(b)** Overlapping of the continuous and discrete functions $\cos(2\pi 13m/17)$ with $m \in$ [0,17] and $m \in \{0, 1, ..., 16\}$.

Let us now see that the behaviour of the sinus wave is quite the same. Take the function $\sin(2\pi pm/M)$, $m \in \{0, 1, ..., M-1\}$ with $p < M/2$ a non-negative integer and $k = M - p$. Then,

$$\begin{aligned}\sin(2\pi km/M) &= \sin(2\pi(M-p)m/M) = \sin(2\pi m - 2\pi pm/M) \\ &= \sin(-2\pi pm/M) = -\sin(2\pi pm/M).\end{aligned}$$

This means that the discrete functions $\sin(2\pi pm/M)$ and $\sin(2\pi km/M)$ have the same frequency. So, again, if $0 \leq p \leq [M/2]$, then p is the frequency and if $[M/2] < p \leq M-1$, then the frequency is $M - p$.

Therefore, the discrete function $e^{2\pi ipm/M}$ carries out p full cycles as m goes from 0 to $M-1$ if $0 \leq p \leq [M/2]$. On the other hand, it carries out $M - p$ full cycles as m goes from 0 to $M-1$ if $[M/2] < p \leq M-1$. Thus we regard $e^{2\pi ipm/M}$ as a high frequency vector for p near $M/2$ and as a low frequency vector for p near 0 or $M-1$. This implies that the matrix $F_{p,q}$ of the Fourier basis is considered as a high frequency matrix if p is near $M/2$ and q is near $N/2$. On the other hand, it is considered a low frequency matrix if p is near 0 or $M-1$ and q is near 0 or $N-1$. Therefore, by the previous formula

$$f = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \hat{f}(p,q) F_{p,q},$$

since $|\hat{f}(p,q)|$ is the length of the projection of f onto $F_{p,q}$, then it can be regarded as the strength of the frequency component $F_{p,q}$ needed in making up f. That is, if the values of $|\hat{f}(p,q)|$ are high for the higher frequency components, then f has strong high-frequency components. On the other hand, if the values of $|\hat{f}(p,q)|$ are high for the lower frequency components, then f has strong low-frequency components.

Observe that if we consider the expression

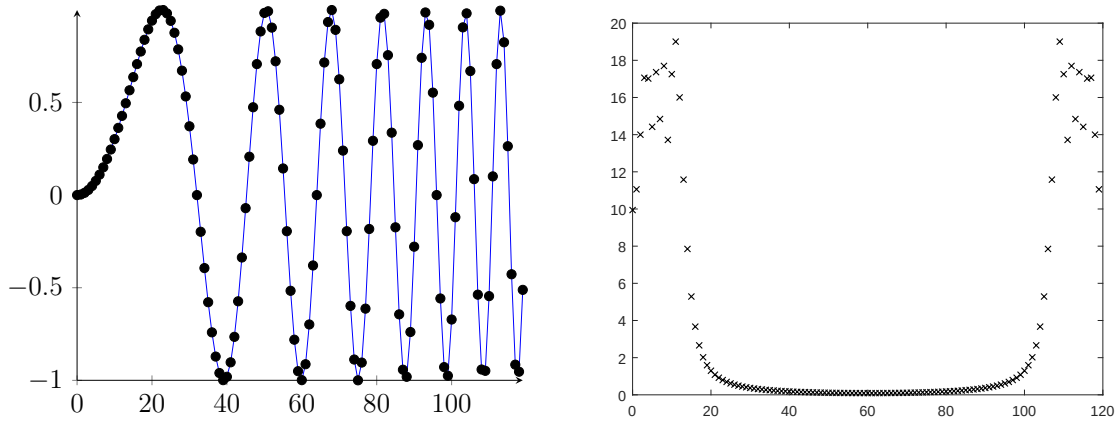
$$\hat{f} = \begin{pmatrix} \hat{f}(0,0) & \dots & \hat{f}(0,N-1) \\ \vdots & & \vdots \\ \hat{f}(M-1,0) & \dots & \hat{f}(M-1,N-1) \end{pmatrix},$$

the coefficients located in the center of the matrix will correspond to higher frequencies, whereas the ones located near the corners will correspond to lower frequencies.

Notice that $|\hat{f}(p,q)|$ provides information about the global behaviour of f in the sense that it says whether f is made up by lower and/or higher frequency components but it does not localise them.

So the modulus of the Fourier coefficients provide a frequency analysis of the function. On the contrary, the information encoded in its phase is not so easy to interpret.

Let us see an example of the behaviour of the modulus of the Fourier coefficients. In order to simplify visualisation we will take $N = 1$, so the function will be one-dimensional. Consider $f(m) = \sin(\pi m^2/1024)$ with $m \in \{0, ..., 119\}$. As Figure 2.2 (a) shows, this function seems to have strong low-frequency components (so as to see this fact clearer we have drawn a blue line joining the points). Therefore, Figure 2.2 (b) shows that the magnitude of the higher frequency Fourier coefficients (m near 120/2) is low whereas the magnitude of the lower frequency Fourier coefficients (m near 0 and 119) is high. These coefficients were computed using MATLAB.



(a) (b)

Figure 2.2 (a) $f(m) = \sin(\pi m^2/1024)$ with $m \in \{0, ..., 119\}$. **(b)** $|\hat{f}(m)|$ with $m \in \{0, ..., 119\}$.

Chapter 3

Fast Fourier Transform

As discussed in Section 2.2, in order to compute the Discrete Fourier Transform of a function $f \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ it is enough to multiply it by the change of basis matrix. That is

$$\hat{v} = Wv,$$

where v is the vector obtained by joining the columns of f and W is the change of basis matrix given in the expression (2.2.4). Keeping in mind that v is an $MN \times 1$ vector and W is an $MN \times MN$ matrix, let us analyse the computational cost of this change of basis. In order to do so, we are just going to take into account multiplications since, computationally, they require much more time than sums.

When calculating the k-th component of the vector $\hat{v}$, $\hat{v}(k) = \sum_{n=0}^{MN-1} W(k,n)v(n)$, MN multiplications are required, which implies $(MN)^2$ for the whole vector $\hat{v}_{MN\times 1}$. Observe that these multiplications are of complex numbers and each one involves three different real multiplications since

$$\begin{aligned}(a+bi)(c+di) &= ac + adi + bci - bd = (ac - bd + ad - ad) + (ad + bc + bd - bd)i \\ &= [(a-b)d + (c-d)a] + [(a-b)d + (c+d)b]i.\end{aligned}$$

So in order to compute $\hat{v}$ we need $(MN)^2$ complex multiplications or, equivalently, $3(MN)^2$ real ones.

The Discrete Fourier Transform is one of the fundamental tools in the area of signal processing and, in particular, in image processing. As we will see in the following chapters, an image can be thought of as a matrix where each component represents a pixel. That is, an image is regarded as an element of the set $\ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$. It is common to deal with images that have thousands or even millions of pixels, so computing the Discrete Fourier Transform of an image by means of the change-of-basis matrix appears to be a task that can take hours. For that reason, an algorithm called the *Fast Fourier Transform (FFT)* is introduced.

At the beginning of this chapter we are going to develop the steps of this well-known algorithm of the FFT following the approaches set in [2] and [8], which, like most of the references on the topic, are only focused on vectors of length a power of two. However, in contrast with them, we are going to distinguish its performance over such vectors and those of even length but not a power of two. In both cases,

we will detail the number of operations required as well as analysing the consequent reduction. Moreover, since the last will appear to be greater when the length is a power of two, we will then study under which conditions *padding* the vector speeds up the whole process, which is a question usually skipped in most of the discussions on the FFT. In doing so, it is going to be essential keeping the error under control. That is the reason why we will pay special attention on finding an upper bound for it. Therefore, the main goal of this chapter is to go deeper on those issues not usually covered in the references on the FFT such as its performance on arbitrary vectors of even length, the precise number of operations required or the setting where *padding* improves the computation as well as bounding its error.

3.1 The Fast Fourier Transform algorithm

Let $f \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ and take $v \in \ell^2(\mathbb{Z}_{MN})$ the vector obtained by joining the columns of f. Let us now develop the FFT algorithm in the computation of $\hat{v}$.

By definition of the Discrete Fourier Transform (see Remark 2.1.4)

$$\hat{v}(k) = \frac{1}{\sqrt{MN}} \sum_{m=0}^{MN-1} v(m) e^{-2\pi i k \frac{m}{MN}}$$

for all $0 \leq k \leq MN-1$. Assume that MN is an integer multiple of 2. Then, we can split the previous expression as follows

$$\begin{aligned}
\hat{v}(k) &= \frac{1}{\sqrt{MN}} \sum_{m=0}^{\frac{MN}{2}-1} v(2m) e^{-2\pi i k \frac{2m}{MN}} + \frac{1}{\sqrt{MN}} \sum_{m=0}^{\frac{MN}{2}-1} v(2m+1) e^{-2\pi i k \frac{2m+1}{MN}} \\
&= \frac{1}{\sqrt{MN}} \sum_{m=0}^{\frac{MN}{2}-1} v(2m) e^{-2\pi i k \frac{2m}{MN}} + e^{-2\pi i \frac{k}{MN}} \frac{1}{\sqrt{MN}} \sum_{m=0}^{\frac{MN}{2}-1} v(2m+1) e^{-2\pi i k \frac{2m}{MN}} \\
&= \frac{1}{\sqrt{MN}} \sum_{m=0}^{\frac{MN}{2}-1} v(2m) e^{-2\pi i k \frac{m}{\frac{MN}{2}}} + e^{-2\pi i \frac{k}{MN}} \frac{1}{\sqrt{MN}} \sum_{m=0}^{\frac{MN}{2}-1} v(2m+1) e^{-2\pi i k \frac{m}{\frac{MN}{2}}}.
\end{aligned}$$

Denoting

$$\begin{aligned}
\hat{v}_0(k) &= \frac{1}{\sqrt{\frac{MN}{2}}} \sum_{m=0}^{\frac{MN}{2}-1} v(2m) e^{-2\pi i k \frac{m}{\frac{MN}{2}}} \\
\hat{v}_1(k) &= \frac{1}{\sqrt{\frac{MN}{2}}} \sum_{m=0}^{\frac{MN}{2}-1} v(2m+1) e^{-2\pi i k \frac{m}{\frac{MN}{2}}},
\end{aligned}$$

we can rewrite

$$\hat{v}(k) = \frac{1}{\sqrt{2}} \hat{v}_0(k) + e^{-2\pi i \frac{k}{MN}} \frac{1}{\sqrt{2}} \hat{v}_1(k).$$

Notice that $\hat{v}_0(k)$ and $\hat{v}_1(k)$ are the k-th Fourier coefficients of the $\ell^2\left(\mathbb{Z}_{\frac{MN}{2}}\right)$ vectors $(v(0), v(2), ..., v(MN-2))$ and $(v(1), v(3), ..., v(MN-1))$ respectively. Moreover, taking $0 \leq k \leq \frac{MN}{2} - 1$, observe that

$$\begin{aligned}\hat{v}\left(k + \frac{MN}{2}\right) &= \frac{1}{\sqrt{2}}\hat{v}_0\left(k + \frac{MN}{2}\right) + e^{-2\pi i \frac{k+\frac{MN}{2}}{MN}} + \frac{1}{\sqrt{2}}\hat{v}_1\left(k + \frac{MN}{2}\right) \\ &= \frac{1}{\sqrt{2}}\hat{v}_0(k) + e^{-2\pi i \frac{k}{MN}} e^{-\pi i} + \frac{1}{\sqrt{2}}\hat{v}_1(k) \\ &= \frac{1}{\sqrt{2}}\hat{v}_0(k) - e^{-2\pi i \frac{k}{MN}} \frac{1}{\sqrt{2}}\hat{v}_1(k),\end{aligned}$$

where we have used the fact that $\hat{v}_0$ and $\hat{v}_1$ have period $\frac{MN}{2}$.

Therefore, for $0 \leq k \leq \frac{MN}{2} - 1$,

$$\begin{aligned}\hat{v}(k) &= \frac{1}{\sqrt{2}}\left[\hat{v}_0(k) + e^{-2\pi i \frac{k}{MN}}\hat{v}_1(k)\right] \\ \hat{v}\left(\frac{MN}{2} + k\right) &= \frac{1}{\sqrt{2}}\left[\hat{v}_0(k) - e^{-2\pi i \frac{k}{MN}}\hat{v}_1(k)\right].\end{aligned}$$

This result is summarized in the following proposition.

Proposition 3.1.1. *Let $v \in \ell^2(\mathbb{Z}_M)$ and $M \in 2\mathbb{Z}$. Split the vector v in the following vectors of $\ell^2\left(\mathbb{Z}_{\frac{M}{2}}\right)$*

$$\begin{aligned}v_0 &= (v(0), v(2), v(4), ..., v(M-2)) \\ v_1 &= (v(1), v(3), v(5), ..., v(M-1)).\end{aligned}$$

Then, for all $0 \leq k \leq \frac{M}{2} - 1$, the Discrete Fourier Transform of v is

$$\begin{aligned}\hat{v}(k) &= \frac{1}{\sqrt{2}}\left[\hat{v}_0(k) + e^{-2\pi i \frac{k}{M}}\hat{v}_1(k)\right] \\ \hat{v}\left(\frac{M}{2} + k\right) &= \frac{1}{\sqrt{2}}\left[\hat{v}_0(k) - e^{-2\pi i \frac{k}{M}}\hat{v}_1(k)\right],\end{aligned} \tag{3.1.1}$$

where $\hat{v}_0$ and $\hat{v}_1$ are the Discrete Fourier Transforms of v_0 and v_1 respectively.

Let us now see, following the notation just introduced in Proposition 3.1.1, the computational advantage of this approach. To begin with, we are not going to take into account the division by $\sqrt{2}$ since it is not as time consuming as complex products. Then, the change-of-basis matrix needed in computing the DFT of the vectors v_0 and v_1 is of size $M/2 \times M/2$. Therefore, the computation of $\hat{v}_0$ and $\hat{v}_1$ requires $(M/2)^2$ complex multiplications each. Moreover, there are $M/2 - 1$ more due to the multiplication by the exponentials (without considering $k = 0$). Thus it all requires

$$2\left(\frac{M}{2}\right)^2 + \frac{M}{2} - 1 = \frac{M^2}{2} + \frac{M}{2} - 1$$

complex multiplications, which can be thought as $\frac{M^2}{2}$ for M large enough. Recall that we needed M^2 of these operations in order to compute $\hat{v}$ by means of the change-of-basis matrix, so this approach cuts the computation time nearly in half. The main point of this procedure is the fact that the computation of the DFT of an M-vector is reduced to that of two $M/2$-vectors.

Then, returning to our initial function $f \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ and its corresponding vector v, this method will just perform $\frac{(MN)^2}{2} + \frac{MN}{2} - 1$ complex multiplications when computing $\hat{v}$, as long as $MN \in 2\mathbb{Z}$.

In Proposition 3.1.1, we are just assuming $M \in 2\mathbb{Z}$ but notice that if $M \in 4\mathbb{Z}$ as well, then we can also apply the already mentioned proposition to the vectors $v_0, v_1 \in \ell^2\left(\mathbb{Z}_{\frac{M}{2}}\right)$, which results on a further reduction. That is, we can divide v_0 and v_1 obtaining the following elements of $\ell^2\left(\mathbb{Z}_{\frac{M}{4}}\right)$

$$v_0 \begin{cases} v_{00} = (v(0), v(4), v(8), ..., v(M-4)) \\ v_{01} = (v(2), v(6), v(10), ..., v(M-2)) \end{cases}$$

$$v_1 \begin{cases} v_{10} = (v(1), v(5), v(9), ..., v(M-3)) \\ v_{11} = (v(3), v(7), v(11), ..., v(M-1)). \end{cases}$$

Then, for $0 \leq k \leq \frac{M}{4} - 1$, we get

$$\begin{aligned} \hat{v}_0(k) &= \frac{1}{\sqrt{2}}\left[\hat{v}_{00}(k) + e^{-2\pi ik/M}\hat{v}_{01}(k)\right], \ \hat{v}_0\left(\frac{M}{4}+k\right) = \frac{1}{\sqrt{2}}\left[\hat{v}_{00}(k) - e^{-2\pi ik/M}\hat{v}_{01}(k)\right] \\ \hat{v}_1(k) &= \frac{1}{\sqrt{2}}\left[\hat{v}_{10}(k) + e^{-2\pi ik/M}\hat{v}_{11}(k)\right], \ \hat{v}_1\left(\frac{M}{4}+k\right) = \frac{1}{\sqrt{2}}\left[\hat{v}_{10}(k) - e^{-2\pi ik/M}\hat{v}_{11}(k)\right] \end{aligned} \tag{3.1.2}$$

Therefore, the number of complex multiplications required this time is $4(M/4)^2$ (due to the four DFT) plus $M/4-1+M/4-1$ (regarding the exponentials in (3.1.2) without considering $k=0$) plus another $M/2-1$ (regarding the exponentials in (3.1.1)). That is

$$\frac{M^2}{4} + 2\left(\frac{M}{4} - 1\right) + \frac{M}{2} - 1 \approx \frac{M^2}{4} \quad \text{for } M \text{ large enough,}$$

which is four times faster than using the change-of-basis matrix directly.

So the Fast Fourier Transform algorithm consists in applying Proposition 3.1.1 repeatedly to the vectors that appear at each stage until it encounters odd sized vectors. That is, if $M = 2^m p$ for some $m, p \in \mathbb{N}$ and p odd, then the algorithm will entail m stages. Notice that, in the last stage, there will be 2^m vectors belonging to $\ell^2(\mathbb{Z}_p)$. Then, the Fourier Transforms of these vectors will be calculated by means of the change-of-basis-matrix. In the following sections, we are going to distinguish how the Fast Fourier Transform applies when $M = 2^m$ and $M = 2^m p$ for some $p \in \mathbb{N}_{\geq 3}$ odd as well as discussing the ensuing reduction in the number of operations.

3.2 Vector of size a power of 2

The algorithm of the Fast Fourier Transform that we have just developed leads us to think that the most favourable case is when $M = 2^m$ for some $m \in \mathbb{N}$, following the notation introduced in Proposition 3.1.1. In this situation, the vector $v \in \ell^2(\mathbb{Z}_M)$ can be subdivided $m = \log_2 M$ times. Notice that at the last stage of division there are M vectors all belonging to $\ell^2(\mathbb{Z}_1)$, so their Discrete Fourier Transforms are themselves. This implies that there is, actually, no need of computing any Fourier Transform, thus the only complex multiplications involved are the ones regarding the exponentials. Observe that, at the first stage of division (given by equation (3.1.1)), there are $\frac{M}{2} - 1$ multiplications of this type and, at the second one (given by equation (3.1.2)), there are $2\left(\frac{M}{4} - 1\right) = \frac{M}{2} - 2$. Then, at the third stage there would be $4\left(\frac{M}{8} - 1\right) = \frac{M}{2} - 4$. Hence, intuitively, in the k-th stage there would be $2^{k-1}\left(\frac{M}{2^k} - 1\right) = \frac{M}{2} - 2^{k-1}$. Since there are $\log_2 M$ stages, it seems that there are

$$\frac{M}{2}\log_2 M - \sum_{n=0}^{\log_2 M - 1} 2^n$$

multiplications regarding exponentials along the algorithm. The following proposition formalises this idea giving us the number of operations required.

Proposition 3.2.1. *Let $M = 2^m$ for some $m \in \mathbb{N}$. Denote as $\#_M^{FFT}$ the maximum number of complex multiplications required to compute the Discrete Fourier Transform of a vector of length M by means of the Fast Fourier Transform algorithm. Then,*

$$\#_M^{FFT} = \frac{1}{2}M\log_2 M - \sum_{n=0}^{\log_2 M - 1} 2^n = M\left(\frac{\log_2 M}{2} - 1\right) + 1.$$

Proof. We begin by proving the second equality.

$$\begin{aligned}\frac{1}{2}M\log_2 M - \sum_{n=0}^{\log_2 M - 1} 2^n &= \frac{M}{2}\log_2 M - \frac{1 - 2^{\log_2 M}}{1-2} = \frac{M}{2}\log_2 M + 1 - 2^{\log_2 M} \\ &= \frac{M}{2}\log_2 M + 1 - M = M\left(\frac{\log_2 M}{2} - 1\right) + 1.\end{aligned}$$

We are now going to prove $\#_M^{\mathrm{FFT}} = M\left(\frac{\log_2 M}{2} - 1\right) + 1$ using induction on m.

Let $v \in \ell^2(\mathbb{Z}_M)$. If $m = 1$, notice that $v = (v_0, v_1)$. Then, $\hat{v} = \frac{1}{\sqrt{2}}(v_0 + v_1, v_0 - v_1)$. Since we don't need any complex multiplication, $\#_2^{\mathrm{FFT}} = 0$. On the other hand,

$$M\left(\frac{\log_2 M}{2} - 1\right) + 1 = 2\left(\frac{1}{2} - 1\right) + 1 = 0.$$

Therefore, the equality holds when $m = 1$.

Assume the result true for $m = k-1$. Let us now see that it also holds for $m = k$. By Proposition 3.1.1, it is clear that

$$\#_M^{\mathrm{FFT}} = 2\#_{\frac{M}{2}}^{\mathrm{FFT}} + \frac{M}{2} - 1 = 2\#_{2^{k-1}}^{\mathrm{FFT}} + 2^{k-1} - 1.$$

Take $M = 2^k$. Then,

$$\begin{aligned}\#_M^{\mathrm{FFT}} = \#_{2^k}^{\mathrm{FFT}} &= 2\#_{2^{k-1}}^{\mathrm{FFT}} + 2^{k-1} - 1 = 2\left(2^{k-1}\left(\frac{k-1}{2} - 1\right) + 1\right) + 2^{k-1} - 1 \\ &= 2^k\left(\frac{k-1}{2} - 1\right) + 2 + 2^{k-1} - 1 = 2^k\frac{k}{2} - 2^{k-1} - 2^k + 2^{k-1} + 1 \\ &= 2^k\frac{k}{2} - 2^k + 1 = 2^k\left(\frac{k}{2} - 1\right) + 1 = M\left(\frac{\log_2 M}{2} - 1\right) + 1,\end{aligned}$$

where in the third equality we have used the hypothesis of induction. □

Let us now see the reduction that ensues from the use of the Fast Fourier Transform instead of the change-of-basis matrix. Keeping in mind that the latter requires M^2 operations, then

$$\begin{aligned}\frac{M^2}{\#_M^{\mathrm{FFT}}} &= \frac{M^2}{M\left(\frac{\log_2 M}{2} - 1\right) + 1} = \frac{M}{\frac{\log_2 M}{2} - 1 + \frac{1}{M}} = \frac{2^m}{\frac{m}{2} - 1 + \frac{1}{2^m}} = \frac{2^{m+1}}{m - 2 + \frac{1}{2^{m-1}}} \\ &\approx \frac{2^{m+1}}{m-2} \quad \text{for } M \text{ big enough.}\end{aligned}$$

This means that the computation of $\hat{v} \in \ell^2(\mathbb{Z}_{2^m})$ by means of the FFT is $\frac{2^{m+1}}{m-2}$ times faster than using the change-of-basis matrix. The behaviour of this reduction is depicted in Figure 3.1 (a). As an example, take a vector of size 2^{18}. Then, the Fast Fourier Transform will need $2^{18} \cdot 8 + 1 = 2\,097\,153$ complex multiplications in contrast with the $(2^{18})^2 = 68\,719\,476\,736$ required by using the change-of-basis matrix. Therefore, the computation becomes more than 32 000 times faster. In Figure 3.1 (b), we can see a comparison of the number of operations required by both methods.

3.3 Vector of even size but not a power of 2

Let us now discuss what happens if $M = 2^m p$ for some $m \in \mathbb{N}$ and $p \in \mathbb{N}_{\geq 3}$ odd, again following the notation introduced in Proposition 3.1.1. In this case, the Fast Fourier Transform algorithm consists on m stages where, at the last one, there are 2^m vectors belonging to $\ell^2(\mathbb{Z}_p)$ whose Fourier Transform must be calculated by means of the change-of-basis matrix. So let us now count, intuitively, the number of operations involved. Recall that we have already discussed that, at the k-th stage of the algorithm, there are $\frac{M}{2} - 2^{k-1}$ multiplications regarding exponentials.

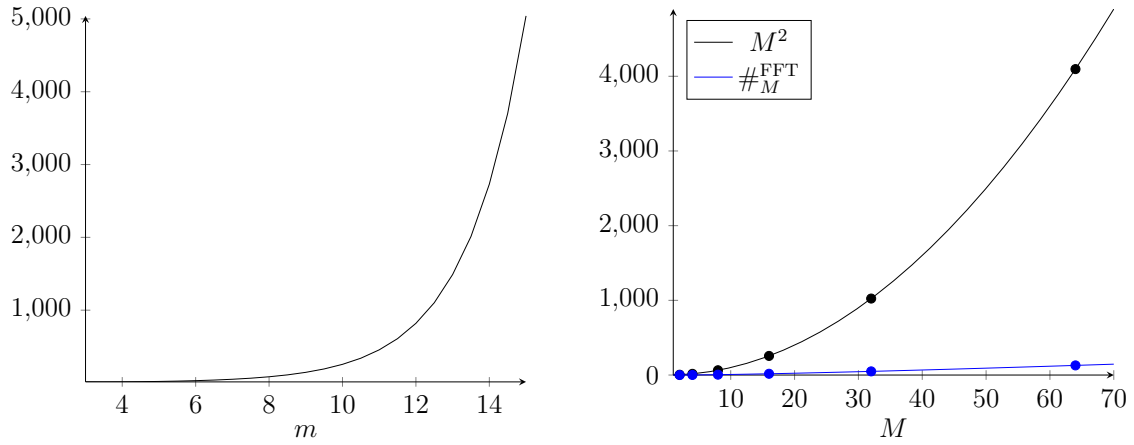


Figure 3.1 (a) Plot of the function $\frac{2^{m+1}}{m-2}$, that is the reduction due to the use of the Fast Fourier Transform instead of the change-of-basis matrix. **(b)** Comparison of the number of operations required by the Fast Fourier Transform and the change-of-basis matrix when $M = 2^m$.

Therefore, it seems that the FFT requires, on the one hand, $m\frac{M}{2} - \sum_{n=0}^{m-1} 2^n$ multiplications due to exponentials and, on the other hand, $2^m p^2$ operations regarding the Fourier transforms of the last stage vectors. This idea is formalised in the following proposition.

Proposition 3.3.1. *Let $v \in \ell^2(\mathbb{Z}_M)$ with $M = 2^m p$ for some $m \in \mathbb{N}$ and $p \in \mathbb{N}_{\geq 3}$ odd. Let $\#_M^{FFT}$ be defined as in Proposition 3.2.1. Then, when computing $\hat{v}$ by means of the Fast Fourier Transform algorithm, we have*

$$\#_M^{FFT} = m\frac{M}{2} - \sum_{n=0}^{m-1} 2^n + 2^m p^2 = \frac{M^2}{2^m} + M\left(\frac{m}{2} - \frac{1}{p}\right) + 1.$$

Proof. Let us begin by proving the second equality.

$$\begin{aligned} m\frac{M}{2} - \sum_{n=0}^{m-1} 2^n + 2^m p^2 &= m\frac{M}{2} - \frac{1-2^m}{1-2} + 2^m p^2 = m\frac{M}{2} + 1 - 2^m + 2^m p^2 \\ &= m\frac{M}{2} + 1 - \frac{M}{p} + 2^m\left(\frac{M}{2^m}\right)^2 = \frac{M^2}{2^m} + M\left(\frac{m}{2} - \frac{1}{p}\right) + 1. \end{aligned}$$

We are now going to prove $\#_M^{\text{FFT}} = \frac{M^2}{2^m} + M\left(\frac{m}{2} - \frac{1}{p}\right) + 1$ by induction on m.

If $m = 1$, then the vector v can be divided just once. Therefore, by Proposition 3.1.1, there are $\frac{M}{2} - 1$ multiplications regarding exponentials plus another $2\left(\frac{M}{2}\right)^2 = 2p^2$ due to the computation of $\hat{v}_0$ and $\hat{v}_1$. That is, $\#_M^{\text{FFT}} = \frac{M}{2} - 1 + 2p^2$. On the other hand,

$$\frac{M^2}{2^m} + M\left(\frac{m}{2} - \frac{1}{p}\right) + 1 = \frac{4p^2}{2} + 2p\left(\frac{1}{2} - \frac{1}{p}\right) + 1 = 2p^2 + p - 2 + 1 = 2p^2 + \frac{M}{2} - 1.$$

Therefore, the equality holds for $m = 1$.

Assume the result true for $m-1$. Let us see that it also holds for m. Let $M = 2^m p$. By Proposition 3.1.1, it is clear that

$$\#_M^{\mathrm{FFT}} = 2\#_{\frac{M}{2}}^{\mathrm{FFT}} + \frac{M}{2} - 1.$$

Then,

$$\begin{aligned}
\#_M^{\mathrm{FFT}} &= 2\#_{\frac{M}{2}}^{\mathrm{FFT}} + \frac{M}{2} - 1 = 2\#_{2^{m-1}p}^{\mathrm{FFT}} + \frac{M}{2} - 1 \\
&= 2\left(\frac{2^{m-1}p \cdot 2^{m-1}p}{2^{m-1}} + 2^{m-1}p\left(\frac{m-1}{2} - \frac{1}{p}\right) + 1\right) + \frac{2^m p}{2} - 1 \\
&= \frac{2^m p \cdot 2^{m-1}p}{2^{m-1}} + 2^m p\left(\frac{m-1}{2} - \frac{1}{p}\right) + 2 + \frac{2^m p}{2} - 1 \\
&= \frac{2^m p \cdot 2^m p}{2^m} + 2^m p\frac{m}{2} - \frac{2^m p}{2} - \frac{2^m p}{p} + \frac{2^m p}{2} + 1 \\
&= \frac{M^2}{2^m} + 2^m p\left(\frac{m}{2} - \frac{1}{p}\right) + 1 = \frac{M^2}{2^m} + M\left(\frac{m}{2} - \frac{1}{p}\right) + 1,
\end{aligned}$$

where in the third equality we have used the hypothesis of induction. □

To get an idea of the power of this algorithm, let us now see the reduction in the number of operations and, therefore, in computation time that the FFT entails (with respect to the change-of-basis matrix) in this case.

$$\begin{aligned}
\frac{M^2}{\#_M^{\mathrm{FFT}}} &= \frac{M^2}{\frac{M^2}{2^m} + M\left(\frac{m}{2} - \frac{1}{p}\right) + 1} = \frac{M^2}{M\left(p + \frac{m}{2} - \frac{1}{p}\right) + 1} = \frac{M}{p + \frac{m}{2} - \frac{1}{p} + \frac{1}{M}} \\
&= \frac{2^m p}{p + \frac{m}{2} - \frac{1}{p} + \frac{1}{2^m p}} = \frac{2^m}{1 + \frac{m}{2p} - \frac{1}{p^2} + \frac{1}{2^m p^2}} \approx \frac{2^m}{1 + \frac{m}{2p} - \frac{1}{p^2}} \approx \frac{2^m}{1 + \frac{m}{2p}},
\end{aligned}$$

where in the first approximation we are assuming M big enough and in the second one we are using the fact that $p \geq 3$, so $1 - \frac{1}{p^2} \approx 1$.

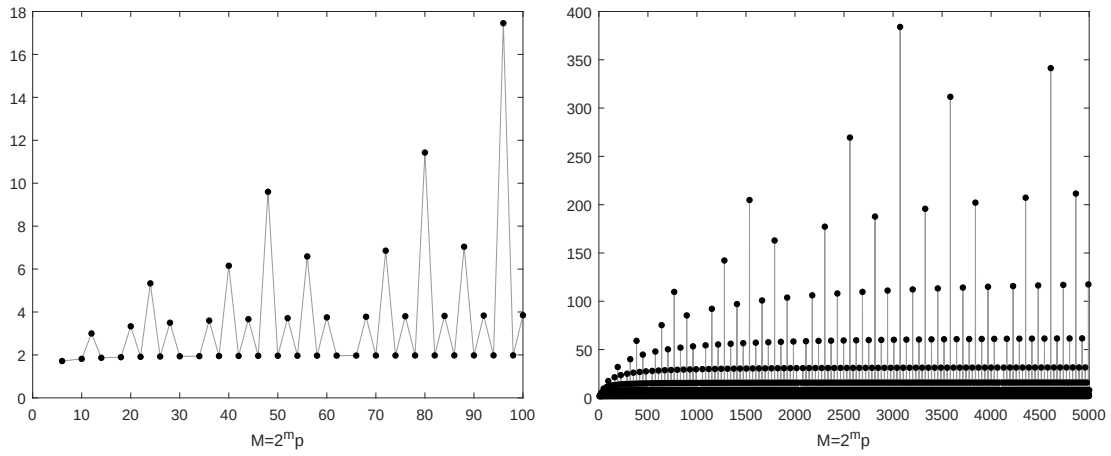
This means that the FFT is $\frac{2^m}{1+\frac{m}{2p}}$ times faster than using the change-of-basis matrix. Notice that, if m is fixed, then

$$\lim_{p\to\infty} \frac{2^m}{1 + \frac{m}{2p}} = 2^m,$$

which causes the plot of the function to display a layer-like structure. This behaviour is depicted in Figure 3.2 (a) and, mostly, (b). Moreover, observe that, as opposed to the situation stated in Section 3.2, the reduction, which is not as high, is not an increasing function.

So if p is big enough, the FFT algorithm becomes 2^m times faster. In Figure 3.3, we can see a comparison of the number of operations required by both methods. Notice that we can write

$$\#_M^{\mathrm{FFT}} = 2^m p^2 + 2^m p\left(\frac{m}{2} - \frac{1}{p}\right) + 1,$$



(a) (b)

Figure 3.2 (a) and **(b)** show the plots of the function $\frac{2^m}{1+\frac{m}{2p}}$ on two different domains.

therefore, if m is fixed, then its plot is quadratic in p, as can be seen in the image. However, observe that, since the domain is M and not p, the smaller is m, the more increasing is $\#_M^{\text{FFT}}$.

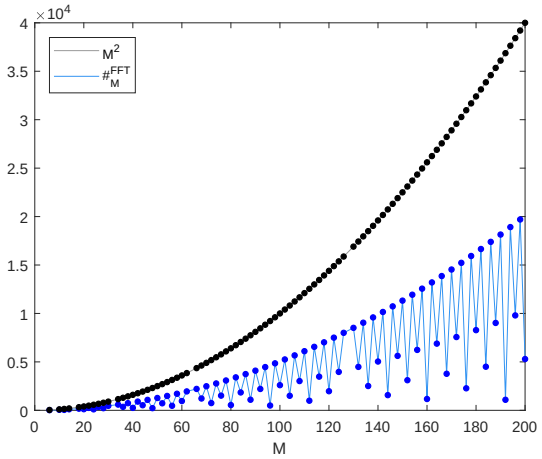


Figure 3.3 Comparison of the number of operations required by the Fast Fourier Transform and the change-of-basis matrix when $M = 2^m p$.

So far, we have analysed the computational difference between the Fast Fourier Transform algorithm and the calculation of the Fourier Transform through the change-of-basis matrix distinguishing whether $M = 2^m$ or $M = 2^m p$ for some $m \in \mathbb{N}$ and $p \in \mathbb{N}_{\geq 3}$ odd. In particular, we have seen that the latter method is always more time-consuming. Putting together the results seen in Proposition 3.2.1 and Proposition 3.3.1, we can plot $\#_M^{\text{FFT}}$ for all $M \in \mathbb{N}_{\geq 2}$ even, as Figure 3.4 shows. The blue values correspond to points of the type $M = 2^m p$ whereas the red ones correspond

to points of the type $M = 2^m$. Since the latter appear to be lower, it makes sense to set out under which circumstances it could be useful to *pad* the vector. This is what we are going to discuss next.

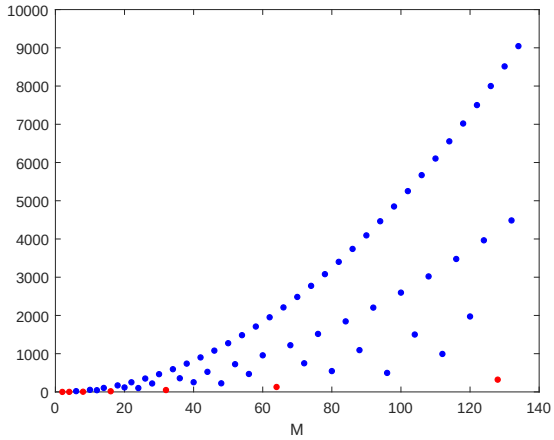


Figure 3.4 Plot of $\#_M^{\mathrm{FFT}}$ where the blue dots are for $M = 2^m p$ and the red ones for $M = 2^m$.

3.4 Padding of an even sized vector

Let us begin by defining what it means to *pad* a vector.

Definition 3.4.1. Let $v \in \ell^2(\mathbb{Z}_M)$ with $M \in \mathbb{N}$ such that $M \neq 2^m$ for all $m \in \mathbb{N}$. Let $k \in \mathbb{N}$ be such that $2^{k-1} < M < 2^k$. Denote as $M_{PAD} = 2^k$. Let $v_{PAD} \in \ell^2(\mathbb{Z}_{M_{PAD}})$ be defined as

$$v_{PAD}(n) = \begin{cases} v(n), & \text{if } 0 \leq n < M \\ 0, & \text{if } M \leq n < M_{PAD} \end{cases}.$$

Then, we will say that the vector v_{PAD} is the vector v *padded*.

In other words, padding a vector consists in filling it with 0's at the end until its length becomes a power of 2.

Remark 3.4.2. Following the notation just introduced in Definition 3.4.1, let us now find k in terms of M.

$$2^{k-1} < M < 2^k \Leftrightarrow k - 1 < \log_2 M < k,$$

hence $k = [\log_2 M] + 1$, where $[\log_2 p]$ means the integer part of $\log_2 p$. Therefore,

$$M_{PAD} = 2^{[\log_2 M]+1}.$$

Proposition 3.4.3. *Let $M \in \mathbb{N}$ such that $M \neq 2^m$ for all $m \in \mathbb{N}$. Suppose $v \in \ell^2(\mathbb{Z}_M)$. Define $\widehat{v_{PAD}}|_M$ as the first M components of the Fourier Transform of the padded vector. Then,*

$$||\widehat{v_{PAD}}|_M - \hat{v}||_\infty \leq ||v||_2 \sup_{1\leq n\leq M-1} \left[\frac{M}{M_{PAD}} + 1 + \frac{1}{\sqrt{M \cdot M_{PAD}}} \right.$$
$$\left. \times \sin\left(M 2\pi n \left(\frac{1}{M} - \frac{1}{M_{PAD}}\right)\right) \frac{\sin\left(2\pi n\left(\frac{1}{M} - \frac{1}{M_{PAD}}\right)\right)}{1 - \cos\left(2\pi n\left(\frac{1}{M} - \frac{1}{M_{PAD}}\right)\right)}\right]^{\frac{1}{2}}.$$

Proof. By definition of Discrete Fourier Transform and of v_{PAD},

$$\begin{aligned}
||\widehat{v_{PAD}}|_M - \hat{v}||_\infty &= \sup_{0\leq n\leq M-1} |\widehat{v_{PAD}}(n) - \hat{v}(n)| \\
&= \sup_{0\leq n\leq M-1} \left| \frac{1}{\sqrt{M_{PAD}}} \sum_{k=0}^{M_{PAD}-1} v_{PAD}(k) e^{\frac{-2\pi i n k}{M_{PAD}}} - \frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} v(k) e^{\frac{-2\pi i n k}{M}} \right| \\
&= \sup_{0\leq n\leq M-1} \left| \frac{1}{\sqrt{M_{PAD}}} \sum_{k=0}^{M-1} v(k) e^{\frac{-2\pi i n k}{M_{PAD}}} - \frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} v(k) e^{\frac{-2\pi i n k}{M}} \right| \\
&= \sup_{0\leq n\leq M-1} \left| \sum_{k=0}^{M-1} v(k) \left(\frac{1}{\sqrt{M_{PAD}}} e^{\frac{-2\pi i n k}{M_{PAD}}} - \frac{1}{\sqrt{M}} e^{\frac{-2\pi i n k}{M}} \right) \right| \\
&\leq \sup_{0\leq n\leq M-1} \sum_{k=0}^{M-1} \left| v(k) \left(\frac{1}{\sqrt{M_{PAD}}} e^{\frac{-2\pi i n k}{M_{PAD}}} - \frac{1}{\sqrt{M}} e^{\frac{-2\pi i n k}{M}} \right) \right| \\
&\leq \sup_{0\leq n\leq M-1} ||v||_2 \left(\sum_{k=0}^{M-1} \left| \frac{1}{\sqrt{M_{PAD}}} e^{\frac{-2\pi i n k}{M_{PAD}}} - \frac{1}{\sqrt{M}} e^{\frac{-2\pi i n k}{M}} \right|^2 \right)^{\frac{1}{2}},
\end{aligned}$$

where in the last step we have used Hölder's inequality. Recall that it states that $|w_1 w_2|_1 \leq |w_1|_p |w_2|_{p'}$ for all $w_1, w_2 \in \ell^1(\mathbb{Z}_k)$ and p, p' satisfying $\frac{1}{p} + \frac{1}{p'} = 1$. Let us now compute the modulus.

$$\begin{aligned}
&\left| \frac{1}{\sqrt{M_{PAD}}} e^{\frac{-2\pi i n k}{M_{PAD}}} - \frac{1}{\sqrt{M}} e^{\frac{-2\pi i n k}{M}} \right|^2 = \left[\frac{1}{\sqrt{M_{PAD}}} \cos\left(\frac{2\pi n k}{M_{PAD}}\right) - \frac{1}{\sqrt{M}} \cos\left(\frac{2\pi n k}{M}\right) \right]^2 \\
&\quad + \left[\frac{1}{\sqrt{M}} \sin\left(\frac{2\pi n k}{M}\right) - \frac{1}{\sqrt{M_{PAD}}} \sin\left(\frac{2\pi n k}{M_{PAD}}\right) \right]^2 \\
&= \frac{1}{M_{PAD}} \cos^2\left(\frac{2\pi n k}{M_{PAD}}\right) + \frac{1}{M} \cos^2\left(\frac{2\pi n k}{M}\right) - 2\frac{1}{\sqrt{M_{PAD} M}} \cos\left(\frac{2\pi n k}{M_{PAD}}\right) \\
&\quad \times \cos\left(\frac{2\pi n k}{M}\right) + \frac{1}{M_{PAD}} \sin^2\left(\frac{2\pi n k}{M_{PAD}}\right) + \frac{1}{M} \sin^2\left(\frac{2\pi n k}{M}\right) - 2\frac{1}{\sqrt{M_{PAD} M}} \\
&\quad \times \sin\left(\frac{2\pi n k}{M_{PAD}}\right) \sin\left(\frac{2\pi n k}{M}\right)
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{M_{PAD}} + \frac{1}{M} - 2\frac{1}{\sqrt{M_{PAD}M}}\left[\cos\left(\frac{2\pi nk}{M_{PAD}}\right)\cos\left(\frac{2\pi nk}{M}\right)\right. \\
&\quad \left. + \sin\left(\frac{2\pi nk}{M_{PAD}}\right)\sin\left(\frac{2\pi nk}{M}\right)\right] \\
&= \frac{1}{M_{PAD}} + \frac{1}{M} - \frac{2}{\sqrt{M_{PAD}M}}\cos\left(\frac{2\pi nk}{M_{PAD}} - \frac{2\pi nk}{M}\right),
\end{aligned}
$$

where in the last equality we have used the following trigonometric identity

$$
\cos(\alpha - \beta) = \cos(\alpha)\cos(\beta) + \sin(\alpha)\sin(\beta). \tag{3.4.1}
$$

Then,

$$
\begin{aligned}
&||\widehat{v_{PAD}}|_M - \hat{v}||_\infty \leq \\
&\leq ||v||_2 \sup_{0\leq n\leq M-1}\left[\sum_{k=0}^{M-1}\left(\frac{1}{M_{PAD}} + \frac{1}{M} - \frac{2}{\sqrt{M_{PAD}M}}\cos\left(\frac{2\pi nk}{M_{PAD}} - \frac{2\pi nk}{M}\right)\right)\right]^{\frac{1}{2}} \\
&\leq ||v||_2 \sup_{0\leq n\leq M-1}\left[\frac{M}{M_{PAD}} + 1 - \frac{2}{\sqrt{M_{PAD}M}}\sum_{k=0}^{M-1}\cos\left(\frac{2\pi nk}{M_{PAD}} - \frac{2\pi nk}{M}\right)\right]^{\frac{1}{2}} \\
&= ||v||_2 \sup_{1\leq n\leq M-1}\left[\frac{M}{M_{PAD}} + 1 - \frac{2}{\sqrt{M_{PAD}M}}\sum_{k=0}^{M-1}\cos\left(\frac{2\pi nk}{M_{PAD}} - \frac{2\pi nk}{M}\right)\right]^{\frac{1}{2}},
\end{aligned} \tag{3.4.2}
$$

where the last equality is due to the fact that the $\cos\left(\frac{2\pi nk}{M_{PAD}} - \frac{2\pi nk}{M}\right)$ reaches its maximum value when $n = 0$. Let us now compute the finite sum. Since $\cos(x) = \frac{e^{ix}+e^{-ix}}{2}$, then

$$
\begin{aligned}
\sum_{k=0}^{M-1}\cos\left(\frac{2\pi nk}{M_{PAD}} - \frac{2\pi nk}{M}\right) &= \sum_{k=0}^{M-1}\cos\left(\frac{2\pi nk}{M} - \frac{2\pi nk}{M_{PAD}}\right) \\
&= \frac{1}{2}\sum_{k=0}^{M-1}\left[e^{i2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)}\right]^k + \frac{1}{2}\sum_{k=0}^{M-1}\left[e^{-i2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)}\right]^k
\end{aligned} \tag{3.4.3}
$$

Keeping in mind that $1 \leq n \leq M-1$, observe that

$$
n\left(\frac{1}{M} - \frac{1}{M_{PAD}}\right) = \frac{n(M_{PAD} - M)}{M \cdot M_{PAD}} < 1,
$$

since $\frac{n}{M} < 1$ and $\frac{M_{PAD}-M}{M_{PAD}} < 1$. Therefore, $0 < 2\pi n\left(\frac{1}{M} - \frac{1}{M_{PAD}}\right) < 2\pi$, which means that the ratios of the geometric series appearing on equation (3.4.3) are different

from 1. Then, this equation can be expressed as follows

$$
\begin{aligned}
&\sum_{k=0}^{M-1}\cos\left(\frac{2\pi nk}{M_{PAD}}-\frac{2\pi nk}{M}\right)=\frac{1}{2}\cdot\frac{1-e^{i2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)M}}{1-e^{i2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)}}+\frac{1}{2}\cdot\frac{1-e^{-i2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)M}}{1-e^{-i2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)}}\\
&=\frac{1}{2}\cdot\frac{1-e^{-i2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)}-e^{i2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)M}+e^{i2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)(M-1)}}{2-e^{i2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)}-e^{-i2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)}}\\
&\quad+\frac{1}{2}\cdot\frac{1-e^{i2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)}-e^{-i2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)M}+e^{-i2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)(M-1)}}{2-e^{i2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)}-e^{-i2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)}}\\
&=\frac{1}{2}\cdot\frac{2-2\cos\left(2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)-2\cos\left(2\pi nM\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)}{2-2\cos\left(2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)}\\
&\quad+\frac{1}{2}\cdot\frac{2\cos\left(2\pi n(M-1)\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)}{2-2\cos\left(2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)}\\
&=\frac{1}{2}\cdot\left(1-\frac{\cos\left(2\pi n-\frac{2\pi nM}{M_{PAD}}\right)-\cos\left(-\frac{2\pi nM}{M_{PAD}}+\frac{2\pi n}{M_{PAD}}+2\pi n-\frac{2\pi n}{M}\right)}{1-\cos\left(2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)}\right)\\
&=\frac{1}{2}\left(1-\frac{\cos\left(\frac{2\pi nM}{M_{PAD}}\right)-\cos\left(2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)+\frac{2\pi nM}{M_{PAD}}\right)}{1-\cos\left(2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)}\right)\\
&=\frac{1}{2}\left(1-\frac{\cos\left(\frac{2\pi nM}{M_{PAD}}\right)-\cos\left(2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)\cos\left(\frac{2\pi nM}{M_{PAD}}\right)}{1-\cos\left(2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)}\right.\\
&\quad\left.-\frac{\sin\left(2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)\sin\left(\frac{2\pi nM}{M_{PAD}}\right)}{1-\cos\left(2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)}\right)\\
&=\frac{1}{2}\left(1-\cos\left(\frac{2\pi nM}{M_{PAD}}\right)+\frac{\sin\left(2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)\sin\left(\frac{2\pi nM}{M_{PAD}}\right)}{1-\cos\left(2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)}\right)\\
&=\frac{1}{2}\left(1-\cos\left(\frac{2\pi nM}{M_{PAD}}\right)+\sin\left(\frac{2\pi nM}{M_{PAD}}-\frac{2\pi nM}{M}\right)\frac{\sin\left(2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)}{1-\cos\left(2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)}\right)\\
&=\frac{1}{2}\left(1-\cos\left(\frac{2\pi nM}{M_{PAD}}\right)-\sin\left(M2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)\right.\\
&\quad\left.\times\frac{\sin\left(2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)}{1-\cos\left(2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)}\right),
\end{aligned}
\tag{3.4.4}
$$

where we have used the trigonometric identity

$$\cos(\alpha+\beta)=\cos(\alpha)\cos(\beta)-\sin(\alpha)\sin(\beta).$$

Then, we can rewrite equation (3.4.2) as follows

$$\begin{aligned}
||\widehat{v_{PAD}}-\hat{v}||_\infty &\leq ||v||_2 \sup_{1\leq n\leq M-1}\left[\frac{M}{M_{PAD}}+1-\frac{1}{\sqrt{M_{PAD}M}}+\frac{1}{\sqrt{M_{PAD}M}}\cos\left(\frac{2\pi nM}{M_{PAD}}\right)\right.\\
&\quad+\frac{1}{\sqrt{M_{PAD}M}}\sin\left(M2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)\\
&\quad\left.\times\frac{\sin\left(2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)}{1-\cos\left(2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)}\right]^{\frac{1}{2}}\\
&\leq ||v||_2 \sup_{1\leq n\leq M-1}\left[\frac{M}{M_{PAD}}+1-\frac{1}{\sqrt{M_{PAD}M}}+\frac{1}{\sqrt{M_{PAD}M}}+\frac{1}{\sqrt{M_{PAD}M}}\right.\\
&\quad\left.\times\sin\left(M2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)\frac{\sin\left(2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)}{1-\cos\left(2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)}\right]^{\frac{1}{2}}\\
&= ||v||_2 \sup_{1\leq n\leq M-1}\left[\frac{M}{M_{PAD}}+1+\frac{1}{\sqrt{M_{PAD}M}}\right.\\
&\quad\left.\times\sin\left(M2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)\frac{\sin\left(2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)}{1-\cos\left(2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)}\right]^{\frac{1}{2}},
\end{aligned}\tag{3.4.5}$$

as we wanted to see. □

Remark 3.4.4. Notice that, in expression (3.4.5), we could have finished the computations after the first inequality, obtaining

$$\begin{aligned}
||\widehat{v_{PAD}}-\hat{v}||_\infty &\leq ||v||_2 \sup_{1\leq n\leq M-1}\left[\frac{M}{M_{PAD}}+1-\frac{1}{\sqrt{M_{PAD}M}}+\frac{1}{\sqrt{M_{PAD}M}}\cos\left(\frac{2\pi nM}{M_{PAD}}\right)\right.\\
&\quad+\frac{1}{\sqrt{M_{PAD}M}}\sin\left(M2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)\\
&\quad\left.\times\frac{\sin\left(2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)}{1-\cos\left(2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)}\right]^{\frac{1}{2}},
\end{aligned}$$

which is a more accurate bound than the one given in the statement. However, observe that both have the same behaviour for large values of M. Therefore, it is preferable to work with the latter since it is more manageable.

Corollary 3.4.5. *Let $v \in \ell^2(\mathbb{Z}_M)$ with $M \in \mathbb{N}$ such that $M \neq 2^m$ for all $m \in \mathbb{N}$. Let $\widehat{v_{PAD}}|_M$ be as defined in Proposition 3.4.3. Then,*

$$||\widehat{v_{PAD}}|_M - \hat{v}||_\infty \leq \sqrt{6}||v||_2.$$

Proof. From equation (3.4.4) it follows that

$$-2\sum_{k=0}^{M-1}\cos\left(\frac{2\pi nk}{M_{PAD}} - \frac{2\pi nk}{M}\right) = -1 + \cos\left(\frac{2\pi nM}{M_{PAD}}\right) + \sin\left(M2\pi n\left(\frac{1}{M} - \frac{1}{M_{PAD}}\right)\right) \times \frac{\sin\left(2\pi n\left(\frac{1}{M} - \frac{1}{M_{PAD}}\right)\right)}{1 - \cos\left(2\pi n\left(\frac{1}{M} - \frac{1}{M_{PAD}}\right)\right)}. \tag{3.4.6}$$

Since the left term is upper bounded from above by $2M$ and

$$-2 \leq -1 + \cos\left(\frac{2\pi nM}{M_{PAD}}\right) \leq 0,$$

then

$$\sin\left(M2\pi n\left(\frac{1}{M} - \frac{1}{M_{PAD}}\right)\right)\frac{\sin\left(2\pi n\left(\frac{1}{M} - \frac{1}{M_{PAD}}\right)\right)}{1 - \cos\left(2\pi n\left(\frac{1}{M} - \frac{1}{M_{PAD}}\right)\right)} \leq 2M + 2.$$

Therefore, using the result seen in Proposition 3.4.3,

$$\begin{aligned}
||\widehat{v_{PAD}}|_M - \hat{v}||_\infty &\leq ||v||_2 \sup_{1\leq n\leq M-1}\left[\frac{M}{M_{PAD}} + 1 + \frac{1}{\sqrt{M \cdot M_{PAD}}}\right. \\
&\qquad \left.\times \sin\left(M2\pi n\left(\frac{1}{M} - \frac{1}{M_{PAD}}\right)\right)\frac{\sin\left(2\pi n\left(\frac{1}{M} - \frac{1}{M_{PAD}}\right)\right)}{1 - \cos\left(2\pi n\left(\frac{1}{M} - \frac{1}{M_{PAD}}\right)\right)}\right]^{\frac{1}{2}} \\
&\leq ||v||_2 \sup_{1\leq n\leq M-1}\left[\frac{M}{M_{PAD}} + 1 + \frac{2M+2}{\sqrt{M_{PAD}M}}\right]^{\frac{1}{2}} \\
&= ||v||_2\left[\frac{M}{M_{PAD}} + 1 + 2\sqrt{\frac{M}{M_{PAD}}} + \frac{2}{\sqrt{M_{PAD}M}}\right]^{\frac{1}{2}} \leq \sqrt{6}||v||_2.
\end{aligned}$$

□

Remark 3.4.6. Notice that applying the argument used in the proof of the Corollary 3.4.5, in particular equation (3.4.6), to the bound mentioned in Remark 3.4.4, we get

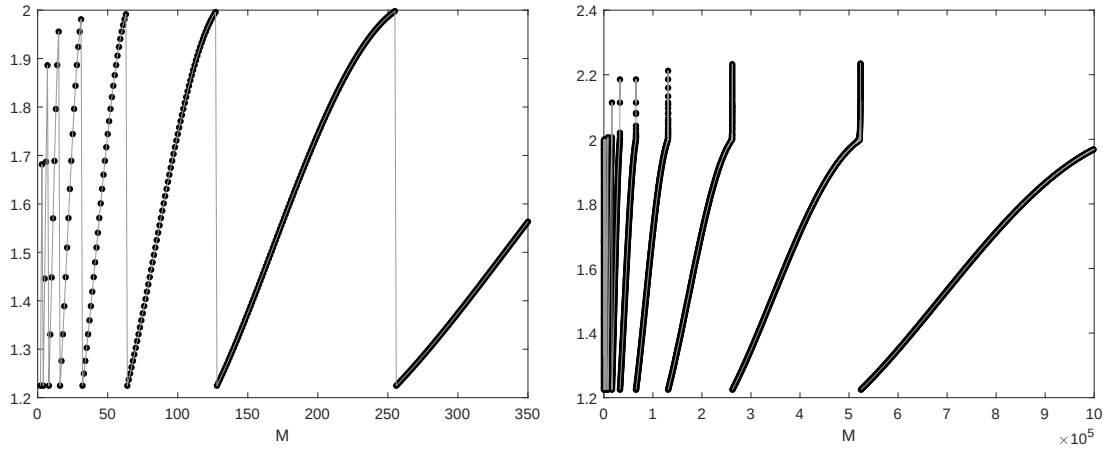
$$||\widehat{v_{PAD}}|_M - \hat{v}||_\infty \leq ||v||_2\left[\frac{M}{M_{PAD}} + 1 + \frac{2M}{\sqrt{M_{PAD}M}}\right]^{\frac{1}{2}} \leq 2||v||_2,$$

which is a bit more accurate.

Figure 3.5 shows a plot of the function

$$\sup_{1\leq n\leq M-1}\left[\frac{M}{M_{PAD}}+1+\frac{1}{\sqrt{M\cdot M_{PAD}}}\sin\left(M2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)\right.$$
$$\left.\times\frac{\sin\left(2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)}{1-\cos\left(2\pi n\left(\frac{1}{M}-\frac{1}{M_{PAD}}\right)\right)}\right]^{\frac{1}{2}} \tag{3.4.7}$$

for the values of M on the interval $0 < M \leq 1\,000\,000$. There, we can see that it actually remains below $\sqrt{6}\sim 2.45$.



(a) (b)

Figure 3.5 Plot of the function (3.4.7) in two different domains

So, provided that $||\widehat{v_{PAD}}|M-\hat{v}||_\infty$ is small enough, the discussion of the Sections 3.2 and 3.3 brings up the question of when it could be useful to compute $\hat{v}$ through $\widehat{v_{PAD}}$. That is, instead of applying the FFT to v, applying it to v_{PAD} and then just keep the first M components of the latter. One situation in which it is preferable to compute $\widehat{v_{PAD}}$ instead of $\hat{v}$ is when the number of multiplications required in computing $\hat{v}$ is higher in comparison to that of $\widehat{v_{PAD}}$. Let us now see when such situation occurs.

Let $v\in\ell^2(\mathbb{Z}_M)$ with $M=2^m p$ for some $m\in\mathbb{N}$ and $p\in\mathbb{N}_{\geq 3}$ odd. Then, by Propositions 3.2.1 and 3.3.1 and Remark 3.4.2, we know that

$$\begin{aligned}\#^{\text{FFT}}_{M_{PAD}} &= M_{PAD}\left(\frac{\log_2 M_{PAD}}{2}-1\right)+1=\frac{M_{PAD}}{2}\log_2 M_{PAD}-M_{PAD}+1\\ &=\frac{2^m\cdot 2^{1+[\log_2 p]}}{2}(m+1+[\log_2 p])-2^m\cdot 2^{1+[\log_2 p]}+1\\ \#^{\text{FFT}}_{M} &= \frac{M^2}{2^m}+M\left(\frac{m}{2}-\frac{1}{p}\right)+1=\frac{2^m\cdot 2^m p^2}{2^m}+2^m p\left(\frac{m}{2}-\frac{1}{p}\right)+1\\ &=2^m p^2+\frac{2^m pm}{2}-2^m+1.\end{aligned}$$

We want to see when $\#_M^{\text{FFT}} > \#_{M_{PAD}}^{\text{FFT}}$. That is,

$$\begin{aligned} 2^m p^2 + \frac{2^m pm}{2} - 2^m + 1 &> \frac{2^m \cdot 2^{1+[\log_2 p]}}{2}(m + 1 + [\log_2 p]) - 2^m \cdot 2^{1+[\log_2 p]} + 1 \\ &\Updownarrow \\ p^2 + \frac{pm}{2} - 1 &> 2^{[\log_2 p]}(m + 1 + [\log_2 p]) - 2^{1+[\log_2 p]} \\ &\Updownarrow \\ \frac{pm}{2} - 2^{[\log_2 p]} m &> 2^{[\log_2 p]}(1 + [\log_2 p]) - 2^{1+[\log_2 p]} + 1 - p^2. \end{aligned} \tag{3.4.8}$$

Observe that $\frac{p}{2} - 2^{[\log_2 p]} < 0$ since

$$\frac{p}{2} - 2^{[\log_2 p]} < 0 \Leftrightarrow \frac{p}{2} < 2^{[\log_2 p]} \Leftrightarrow p < 2^{[\log_2 p]+1}, \tag{3.4.9}$$

which is true for all $p \in \mathbb{N}$. Therefore, $\#_M^{\text{FFT}} > \#_{M_{PAD}}^{\text{FFT}}$ if and only if

$$\begin{aligned} m &< \frac{2^{[\log_2 p]}(1 + [\log_2 p]) - 2^{1+[\log_2 p]} + 1 - p^2}{\frac{p}{2} - 2^{[\log_2 p]}} = \frac{2^{[\log_2 p]}([\log_2 p] - 1) + 1 - p^2}{\frac{p}{2} - 2^{[\log_2 p]}} \\ &= \frac{2^{[\log_2 p]+1}([\log_2 p] - 1) + 2 - 2p^2}{p - 2^{[\log_2 p]+1}}. \end{aligned}$$

In conclusion, $\#_M^{\text{FFT}} > \#_{M_{PAD}}^{\text{FFT}}$ if and only if

$$m < \frac{2^{[\log_2 p]+1}([\log_2 p] - 1) + 2 - 2p^2}{p - 2^{[\log_2 p]+1}}. \tag{3.4.10}$$

So in case m satisfies inequality (3.4.10), it is faster to compute $\widehat{v_{PAD}}$ and then keeping the first M components than computing $\hat{v}$. Let us now analyse the behaviour of this function.

By looking at Figure 3.6 (a) and (b), we get an idea of the performance of the expression (3.4.10) as a function of p. However, by looking at (c), we can see that the larger p is, the wider the range of values of m satisfying inequality (3.4.10) is. However, we also observe that there is a certain set of points where the values of the function

$$\frac{2^{[\log_2 p]+1}([\log_2 p] - 1) + 2 - 2p^2}{p - 2^{[\log_2 p]+1}}$$

blow up. Notice that these are the points where the denominator takes lower values, that is, when p approaches $2^{1+[\log_2 p]}$ or, equivalently, when $\log_2 p - [\log_2 p]$ approaches 1. In other words, these are the values of p near the jumps of the function $[\log_2 p]$, which can be seen in Figure 3.6 (d).

Going back to the inequality (3.4.10), if, for instance, $M = 2^{18} \cdot 3$, then

$$18 = m \not< \frac{2^{[\log_2 p]+1}([\log_2 p] - 1) + 2 - 2p^2}{p - 2^{[\log_2 p]+1}} = 16.$$

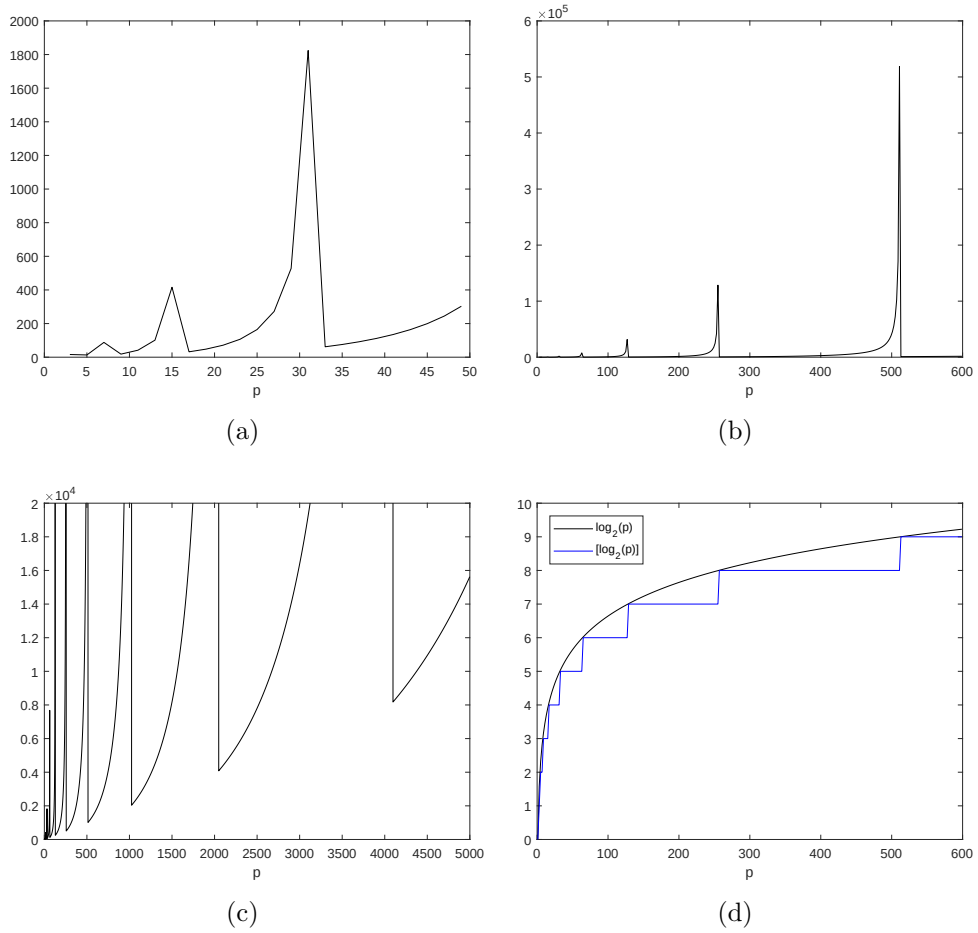


Figure 3.6 (a) and **(b)** show the plots of the function $\frac{2^{[\log_2 p]+1}([\log_2 p]-1)+2-2p^2}{p-2^{[\log_2 p]+1}}$ on two different domains. **(c)** Shows the plot of the same function than (a) and (b) although in a much larger domain and with its image limited to the interval [0,20000]. **(d)** Is a comparative of the functions $\log_2 p$ and $[\log_2 p]$.

Thus, in this case, we would compute $\hat{v}$ directly using the FFT rather than padding the vector and computing its transform. On the contrary, if $M = 2^{18} \cdot 93$, then

$$18 = m < \frac{2^{[\log_2 p]+1}\left([\log_2 p]-1\right)+2-2p^2}{p-2^{[\log_2 p]+1}} \approx 476.$$

Therefore, in this situation, it would take less time computing the transform of the padded vector, as long as $||\widehat{v_{PAD}}|M - \hat{v}||_\infty$ is small enough.

Observe that

$$\frac{\#_M^{\text{FFT}}}{\#_{M_{PAD}}^{\text{FFT}}} = \frac{\frac{M^2}{2^m} + M\left(\frac{m}{2}-\frac{1}{p}\right)+1}{\frac{M_{PAD}}{2}\log_2 M_{PAD} - M_{PAD}+1}$$

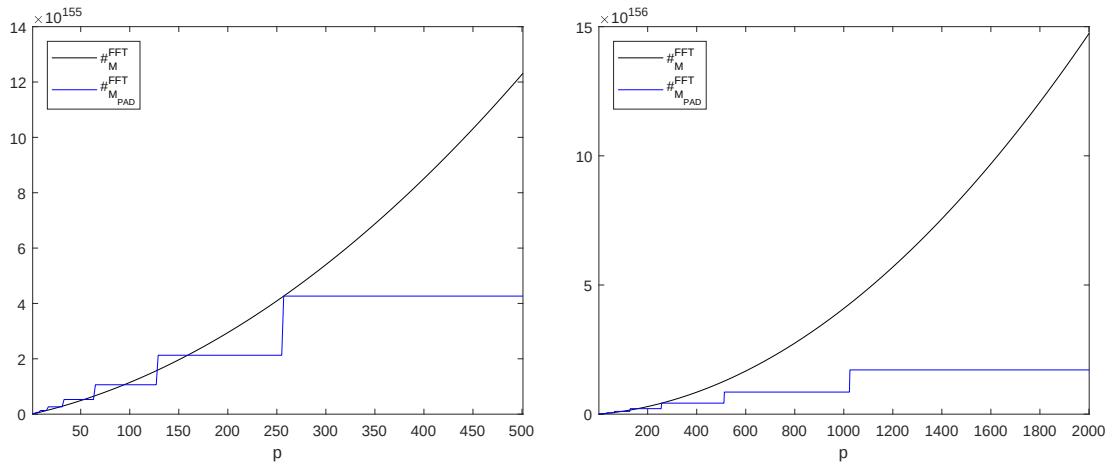
$$= \frac{2^m p^2 + 2^m p\left(\frac{m}{2}-\frac{1}{p}\right)+1}{2^{m+[\log_2 p]}\left(1+m+[\log_2 p]\right)-2^{m+1+[\log_2 p]}+1} \approx \frac{2^m p^2 + 2^m p\left(\frac{m}{2}-\frac{1}{p}\right)}{2^{m+[\log_2 p]}\left(-1+m+[\log_2 p]\right)+1}.$$

Then, cancelling terms and using the fact that $2^{[\log_2 p]} < p$,

$$\frac{\#_M^{\text{FFT}}}{\#_{M_{PAD}}^{\text{FFT}}} \approx \frac{p^2 + p\left(\frac{m}{2} - \frac{1}{p}\right)}{2^{[\log_2 p]}(-1 + m + [\log_2 p]) + \frac{1}{2^m}} > \frac{p^2 + p\left(\frac{m}{2} - \frac{1}{p}\right)}{p(-1 + m + [\log_2 p]) + \frac{1}{2^m}}$$
$$= \frac{p + \left(\frac{m}{2} - \frac{1}{p}\right)}{m - 1 + [\log_2 p] + \frac{1}{2^m p}} \xrightarrow[p\to\infty]{} \infty.$$

So if m satisfies inequality (3.4.10), then the larger p is, the faster the computation of $\widehat{v_{PAD}}$ is in comparison to $\hat{v}$ (by means of the Fast Fourier Transform). This behaviour can be seen in Figure 3.7, where we have compared $\#_M^{\text{FFT}}$ and $\#_{M_{PAD}}^{\text{FFT}}$ for $m = 500$. In (a) we can see that if p is, approximately, less than 160, then $\#_{M_{PAD}}^{\text{FFT}}$ is not necessarily less than $\#_M^{\text{FFT}}$, so the vector should not be padded. However, (b) shows that as p increases, $\#_{M_{PAD}}^{\text{FFT}}$ becomes much lower than $\#_M^{\text{FFT}}$, so padding the vector would save a lot of time.

Putting these results together, assume that $v \in \ell^2(\mathbb{Z}_{2^m p})$ for some $m \in \mathbb{N}$ and $p \in \mathbb{N}_{\geq 3}$ odd and we want to compute $\hat{v}$. Then, if m satisfies inequality (3.4.10) and $||\widehat{v_{PAD}}|M - \hat{v}||$ is small enough, then it is faster to compute $\widehat{v_{PAD}}$ (by means of the FFT) and then keeping the first M components of it. On the contrary, if one of these two conditions fails to be true, then we have to apply the Fast Fourier Transform algorithm to the vector v itself.



(a) (b)

Figure 3.7 (a) Shows a comparative of $\#_M^{\text{FFT}}$ and $\#_{M_{PAD}}^{\text{FFT}}$ for $M = 2^{500}p$ on a small domain. **(b)** Shows the same as (a) but on a larger domain.

Chapter 4

Extension of the Fast Fourier Transform

In this chapter we are going to state the result on which the FFT algorithm is based, Proposition 4.1.1, which can be found in [2]. Following this idea, we are going to extend this algorithm to another one that applies to vectors of any size, not just even. Hence, the first concern is going to be its performance over vectors of odd length. The results found will lead to the *mixed algorithm*, an improvement of the FFT for the vectors of even length but not a power of two which will arise, again, the padding discussion. In each of these two cases mentioned, we are going to precise the number of operations required by the corresponding approach as well as analysing the reduction that follows. Since we will end up with a wide range of algorithms, Section 4.5 provides a summary of the different situations that we can encounter and gives the algorithm that should be used in each of them.

4.1 Extension of the Fast Fourier Transform algorithm

Let us begin by formulating the main result behind the FFT that, as mentioned above, will allow its extension.

Proposition 4.1.1. *Let $v \in \ell^2(\mathbb{Z}_M)$ and $M = pq$ for some $p, q \in \mathbb{N}_{>1}$. Define the vectors $x_0, x_1, ..., x_{p-1} \in \ell^2(\mathbb{Z}_q)$ as*

$$x_\ell(k) = v(kp + \ell) \quad \text{for } k = 0, 1, ..., q-1.$$

Then, for $a = 0, 1, ..., p-1$ and $b = 0, 1, ..., q-1$

$$\widehat{v}(aq + b) = \frac{1}{\sqrt{p}} \sum_{k=0}^{p-1} e^{-2\pi i k\left(\frac{b}{M} + \frac{a}{p}\right)} \widehat{x}_k(b). \tag{4.1.1}$$

Notice that every $m = 0, 1, ..., M-1$ can be expressed as $m = aq + b$ for some $a \in \{0, 1, ..., p-1\}$ and $b \in \{0, 1, ..., q-1\}$, so equation (4.1.1) gives the full DFT of v.

Proof. Let $0 \leq m \leq M-1$ and $a \in \{0,1,...,p-1\}$, $b \in \{0,1,...,q-1\}$ be such that $m = aq+b$. Then,

$$\widehat{v}(m) = \widehat{v}(aq+b) = \frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} v(k) e^{-2\pi i(aq+b)k/M}.$$

Notice that there exist $r \in \{0,1,...,q-1\}$ and $s \in \{0,1,...,p-1\}$ such that $k = rp+s$. Hence,

$$\widehat{v}(aq+b) = \frac{1}{\sqrt{M}} \sum_{s=0}^{p-1} \sum_{r=0}^{q-1} v(rp+s) e^{-2\pi i(aq+b)(rp+s)/(pq)}.$$

Since,

$$e^{-2\pi i(aq+b)(rp+s)/(pq)} = e^{-2\pi iar} e^{-2\pi ias/p} e^{-2\pi ibr/q} e^{-2\pi ibs/(pq)},$$

then

$$\begin{aligned}
\widehat{v}(aq+b) &= \frac{1}{\sqrt{M}} \sum_{s=0}^{p-1} e^{-2\pi ias/p} e^{-2\pi ibs/M} \sum_{r=0}^{q-1} v(rp+s) e^{-2\pi ibr/q} \\
&= \frac{1}{\sqrt{p}} \sum_{s=0}^{p-1} e^{-2\pi ias/p} e^{-2\pi ibs/M} \left(\frac{1}{\sqrt{q}} \sum_{r=0}^{q-1} x_s(r) e^{-2\pi ibr/q} \right) \\
&= \frac{1}{\sqrt{p}} \sum_{s=0}^{p-1} e^{-2\pi ias/p} e^{-2\pi ibs/M} \widehat{x}_s(b) = \frac{1}{\sqrt{p}} \sum_{s=0}^{p-1} e^{-2\pi is\left(\frac{a}{p}+\frac{b}{M}\right)} \widehat{x}_s(b).
\end{aligned}$$

□

The reduction that this approach entails is based on the fact that, by equation (4.1.1), for each value of a, we need to compute $\widehat{x}_k$ for all k. So, the algorithm that we are now going to develop recognises this fact and calculates $\widehat{x}_k$ just once.

Notice that, if q is not a prime number, then we can apply again Proposition 4.1.1 so as to compute $\widehat{x}_\ell$ for all $0 \leq \ell \leq p-1$. This iteration of the procedure allows the development of the following algorithm. We are going to follow the notation just introduced. Let $M = p_1 p_2 \cdots p_r$ for some $p_1, p_2, ..., p_r$ prime numbers, not necessarily different, strictly greater than 1. For reasons that we are going to see afterwards, in Remark 4.2.2, let $p_1 \geq p_2 \geq p_3 \geq ... \geq p_r$. Now, take $p = p_1$ and $q = p_2 p_3 \cdots p_r$ and apply Proposition 4.1.1. Then, for $a = 0,1,...,p-1$ and $b = 0,1,...,q-1$

$$\widehat{v}(aq+b) = \frac{1}{\sqrt{p}} \sum_{k=0}^{p-1} e^{-2\pi ik\left(\frac{b}{M}+\frac{a}{p}\right)} \widehat{x}_k(b).$$

In order to compute $\widehat{x}_\ell$, take $p = p_2$ and $q = p_3 p_4 \cdots p_r$ and, again, apply the previous result to the vector $x \in \ell^2(\mathbb{Z}_{p_2 p_3 \cdots p_r})$. Then, for $c = 0,1,...,p-1$ and $d = 0,1,...,q-1$

$$\widehat{v}(cq+d) = \frac{1}{\sqrt{p}} \sum_{k=0}^{p-1} e^{-2\pi ik\left(\frac{d}{p_2 p_3 \cdots p_r}+\frac{c}{p}\right)} \widehat{y}_k(d),$$

where

$$y_\ell(k) = x(kp + l) \quad \text{for } k = 0, 1, ..., q - 1.$$

One more time, we implement the result to y_ℓ taking $p = p_3$ and $q = p_4 p_5 \cdots p_r$ so as to obtain $\widehat{y}_\ell$. And we keep applying the proposition in order to compute the Fourier Transforms until a set of vectors of size p_r arises. At this point we perform the last step of the algorithm, which consists in calculating its transforms by means of the change-of-basis matrix, since p_r is prime.

Notice that, apart from the algorithm that we have just set out, Proposition 4.1.1 gives rise to many others. For instance, we could take, in the first step, $p = p_1 p_2$ and $q = p_3 p_4 ... p_r$. So, a wide range of different algorithms could be developed as long as they are computationally faster than using the change-of-basis matrix, which is the main goal.

Remark 4.1.2. In the particular case of M even, Proposition 4.1.1 becomes Proposition 3.1.1 and, therefore, the preceding algorithm almost becomes the Fast Fourier Transform. That is the reason why, from now on, we are going to refer to it as the *extended algorithm*. Let us now see this fact. Taking $p = 2$, we get the vectors

$$x_0 = (v(0), v(2), v(4), ..., v(2q - 2)) = (v(0), v(2), v(4), ..., v(M - 2))$$
$$x_1 = (v(1), v(3), v(5), ..., v(2q - 1)) = (v(1), v(3), v(5), ..., v(M - 1)).$$

Hence, for all $0 \leq b \leq \frac{M}{2} - 1$,

$$\widehat{v}(b) = \frac{1}{\sqrt{2}} \sum_{k=0}^{1} e^{-2\pi i k b/M} \widehat{x}_k(b) = \frac{1}{\sqrt{2}} \left[\widehat{x}_0(b) + e^{-2\pi i b/M} \widehat{x}_1(b)\right]$$

$$\widehat{v}\left(\frac{M}{2} + b\right) = \frac{1}{\sqrt{2}} \sum_{k=0}^{1} e^{-2\pi i k \left(\frac{b}{M} + \frac{1}{2}\right)} \widehat{x}_k(b) = \frac{1}{\sqrt{2}} \left[\widehat{x}_0(b) + e^{-2\pi i \left(\frac{b}{M} + \frac{1}{2}\right)} \widehat{x}_1(b)\right]$$
$$= \frac{1}{\sqrt{2}} \left[\widehat{x}_0(b) - e^{-2\pi i b/M} \widehat{x}_1(b)\right].$$

Notice that, actually, the extended algorithm does not recognise the fact that for $a = 0$ and $a = 1$ the products by the exponentials are the same. In other words, equation (4.1.1) shows that the algorithm carries out $p - 1$ multiplications for each value of a. This implies that, in the particular case when M is even, it is less time consuming applying the Fast Fourier Transform rather than the extended algorithm. So, we will only implement the latter in the case M is odd and non-prime. Observe that, in this case, it is not possible to obtain a reduction in the number of products since such a reduction can only occur if

$$e^{-2\pi i k \left(\frac{b}{M} + \frac{a}{p}\right)} \widehat{x}_k(b) = -e^{-2\pi i k b/M} \widehat{x}_k(b)$$

for some values of k and a, as happened when M was even and we took $p = 2$. If M is odd, so must be p, which implies that $2ka/p$ could never be an odd integer for any values of k and a. Therefore,

$$e^{-2\pi i k \left(\frac{b}{M} + \frac{a}{p}\right)} \widehat{x}_k(b) \neq -e^{-2\pi i k b/M} \widehat{x}_k(b)$$

for all possible values of k and a. So, as opposed to the case when M is even, in this case we cannot reduce the number of multiplications by the exponential.

Since, as already mentioned, the FFT is preferable over the extended algorithm when M is even, let us now focus on the case when M is odd.

4.2 Vector of odd size but not prime (extended algorithm)

In this section we are going to centre attention on how the extended algorithm performs when computing the Fourier Transform of a vector of length M with $M = p_1 p_2 \cdots p_r$ for some $p_1, p_2, ..., p_r$ prime numbers strictly greater than one. Moreover, we are going to assume $r \geq 2$, since if M is prime, the algorithm does not apply.

Let us begin by studying the number of complex multiplications required in carrying out the algorithm. First, let us count the complex products needed in each of its steps or, equivalently, the ones appearing in Proposition 4.1.1. If we define, similarly as we have done in Chapter 3, $\#_M^{\text{ext}}$ as the number of these type of operations that arise in computing the DFT of a vector of length M using the extended algorithm, then we need, on the one hand, $\widehat{x}_\ell$ for all $0 \leq \ell \leq p-1$, which is $p\#_q^{\text{ext}}$. As we have already discussed at the end of Section 4.1, there are, on the other hand, $p-1$ more products due to the multiplications by the exponentials for each value of a. Hence, the number of operations required in Proposition 4.1.1 is

$$\#_M^{\text{ext}} = p\,\#_q^{\text{ext}} + p(p-1). \tag{4.2.1}$$

Recall that we are not taking into account the multiplication by $\frac{1}{\sqrt{p}}$ since this is not as time consuming as a complex product. Keeping in mind the number of products required at each step, we can now state the total number of them needed along the algorithm.

Proposition 4.2.1. *Let $v \in \ell^2(\mathbb{Z}_M)$ and M an odd and non-prime number. That is, $M = p_1 p_2 \cdots p_r$ for some $p_j \in \mathbb{Z}_{\geq 3}$ prime for all $1 \leq j \leq r$. Then, the number of complex multiplications required along the extended algorithm is*

$$\#_M^{ext} = M\left(p_r + \sum_{k=1}^{r-1} \frac{p_{r-k}-1}{\prod_{j=0}^{k-1} p_{r-j}}\right) = M\left(p_r + \frac{p_{r-1}-1}{p_r} + \frac{p_{r-2}-1}{p_{r-1}p_r} + \cdots + \frac{p_1-1}{p_2 p_3 ... p_r}\right).$$

Proof. We are going to prove it by induction on r. Let $r = 2$, that is $M = p_1 p_2$. In this case we just have to apply Proposition 4.1.1 once. Then, by equation (4.2.1),

$$\#_M^{\text{ext}} = p_1\,\#_{p_2}^{\text{ext}} + p_1(p_1-1) = p_1 p_2^2 + p_1(p_1-1) = M\left(p_2 + \frac{p_1-1}{p_2}\right).$$

Assume, now, the result true for r. Let us now see that it also holds for $r+1$, so we are taking $M = p_1 p_2 \cdots p_{r+1}$. By equation (4.2.1),

$$\begin{aligned}
\#_M^{\text{ext}} &= p_1 \, \#_{p_2 p_3 \cdots p_{r+1}}^{\text{ext}} + p_1(p_1 - 1) \\
&= p_1 \left[p_2 p_3 \cdots p_{r+1} \left(p_{r+1} + \sum_{k=1}^{r-1} \frac{p_{(r+1)-k} - 1}{\prod_{j=0}^{k-1} p_{(r+1)-j}} \right) \right] + p_1(p_1 - 1) \\
&= M \left(p_{r+1} + \sum_{k=1}^{r-1} \frac{p_{(r+1)-k} - 1}{\prod_{j=0}^{k-1} p_{(r+1)-j}} \right) + p_1(p_1 - 1) \\
&= M \left(p_{r+1} + \sum_{k=1}^{r-1} \frac{p_{(r+1)-k} - 1}{\prod_{j=0}^{k-1} p_{(r+1)-j}} + \frac{p_1 - 1}{p_2 p_3 \cdots p_r} \right) \\
&= M \left(p_{r+1} + \sum_{k=1}^{r} \frac{p_{(r+1)-k} - 1}{\prod_{j=0}^{k-1} p_{(r+1)-j}} \right)
\end{aligned}$$

where in the second equality we have used the hypothesis of induction. □

Remark 4.2.2. Notice that, when $r = 2$, the function $p_2 + \frac{p_1 - 1}{p_2}$ satisfies the following.

- If $p_1 \approx p_2$, then $p_2 + \frac{p_1 - 1}{p_2} \approx p_1 + \frac{p_2 - 1}{p_1}$.
- If $p_1 < p_2$, then $p_2 < p_2 + \frac{p_1 - 1}{p_2} < p_2 + 1$. Hence, $p_2 + \frac{p_1 - 1}{p_2} \approx p_2$.
- If $p_1 >> p_2$, then $p_2 + \frac{p_1 - 1}{p_2} < p_2 + \frac{p_1}{p_2} << p_1$.

Therefore, taking $p_1 \geq p_2 \geq \ldots \geq p_r$ seems to be a good choice for reducing the number of operations. That is the reason why, when the developing the algorithm, we have sorted the prime numbers as a decreasing sequence.

In order to analyse the computational advantage that this approach entails, we are going to compare the two values $\#_M^{\text{ext}}$ and M^2. We will use the fact that $p_1 \geq p_j \geq 3$ for all possible values of j.

$$\begin{aligned}
\frac{M^2}{\#_M^{\text{ext}}} &= \frac{p_1 \cdots p_r}{p_r + \frac{p_{r-1} - 1}{p_r} + \frac{p_{r-2} - 1}{p_{r-1} p_r} + \cdots + \frac{p_1 - 1}{p_2 \cdots p_r}} \geq \frac{p_1 \cdots p_r}{p_r + \frac{p_{r-1}}{p_r} + \frac{p_{r-2}}{p_{r-1} p_r} + \cdots + \frac{p_1}{p_2 \cdots p_r}} \\
&\geq \frac{p_1 \cdots p_r}{p_1 + \frac{p_1}{p_r} + \frac{p_1}{p_{r-1} p_r} + \cdots + \frac{p_1}{p_2 \cdots p_r}} = \frac{1}{\frac{1}{p_2 \cdots p_r} + \frac{1}{p_2 \cdots p_{r-1} p_r^2} + \frac{1}{p_2 \cdots p_{r-2} p_{r-1}^2 p_r^2} + \cdots + \frac{1}{p_2^2 \cdots p_r^2}} \\
&\geq \frac{1}{r \frac{1}{p_2 \cdots p_r}} = \frac{p_2 \cdots p_r}{r} \geq \frac{3^{r-1}}{r} \geq \frac{3}{2} > 1.
\end{aligned}$$

The second-to-last step is due to the fact that $r \geq 2$ and the function $\frac{3^{r-1}}{r}$ is increasing in this set of values. So, since, for all values of M, $M^2 > \#_M^{\text{ext}}$, the extended algorithm is always faster. Moreover, notice that, on the one hand,

$$\frac{p_2 \cdots p_r}{r} \xrightarrow[\frac{p}{p_1} \to \infty]{} \infty$$

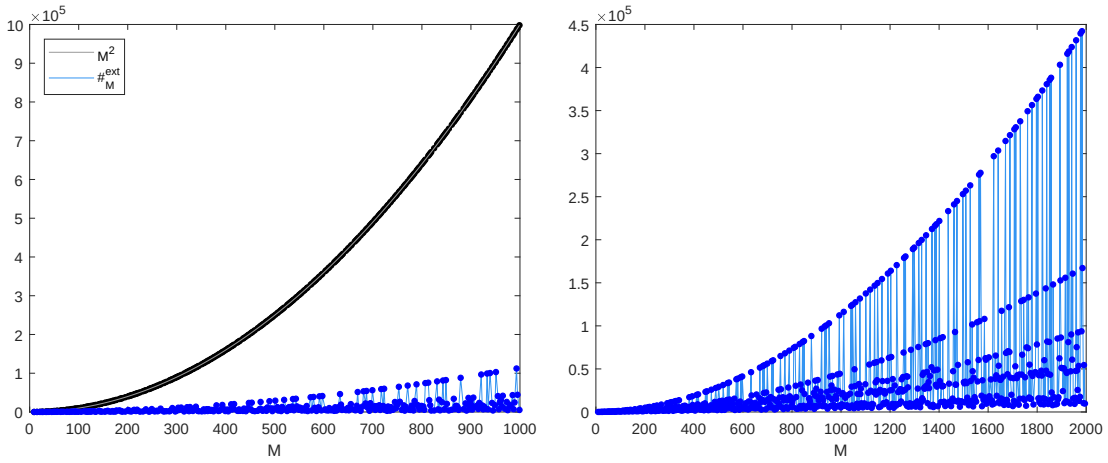
and, on the other hand,

$$\frac{p_2 \cdots p_r}{r} \geq \frac{3^{r-1}}{r} \xrightarrow[r\to\infty]{} \infty,$$

which implies that the reduction in the number of operations increases as p or r grows, or equivalently, as M grows, as can be seen in Figure 4.1 (a). Moreover, notice that the tendency of $\#_M^{\text{ext}}$ is

$$\#_M^{\text{ext}} \approx p_1 \cdots p_{r-1} p_r^2 + p_1 \cdots p_{r-2} p_{r-1}^2 + \cdots + p_1 p_2^2 + p_1^2$$

if the prime numbers are big enough. Therefore, fixing $p_2, p_3, ..., p_r$, it is quadratic on p_1, although this behaviour can only be observed when p_1 is, by far, greater than the other primes. This fact is depicted in Figure 4.1 (b).



(a) (b)

Figure 4.1 (a) Comparison of the number of operations required by the extended algorithm and the change-of-basis matrix when M is odd and non-prime. **(b)** Plot of the function $\#_M^{\text{ext}}$ when M is odd and non-prime.

Recall that we said that, for even values of M, it was faster to use the FFT than the extended algorithm. However, since, as we have seen, the latter is faster than the matrix, it seems that we could obtain an even faster algorithm for when M is even if we mixed the FFT and the extended one. That is what we are going to study next.

4.3 Vector of even size but not a power of 2 (mixed algorithm)

Let us begin by recalling how the Fast Fourier Transform performed over vectors of size $M = 2^m p$ for some $m \in \mathbb{N}$ and $p \in \mathbb{N}_{\geq 3}$ odd.

In Section 3.3 we saw that, for such an M, the FFT consisted on m stages where, at the last one, there were 2^m vectors belonging to $\ell^2(\mathbb{Z}_p)$ whose Fourier Transform

must be calculated by means of the change-of-basis matrix. Then, Proposition 3.3.1 showed that the number of operations required was

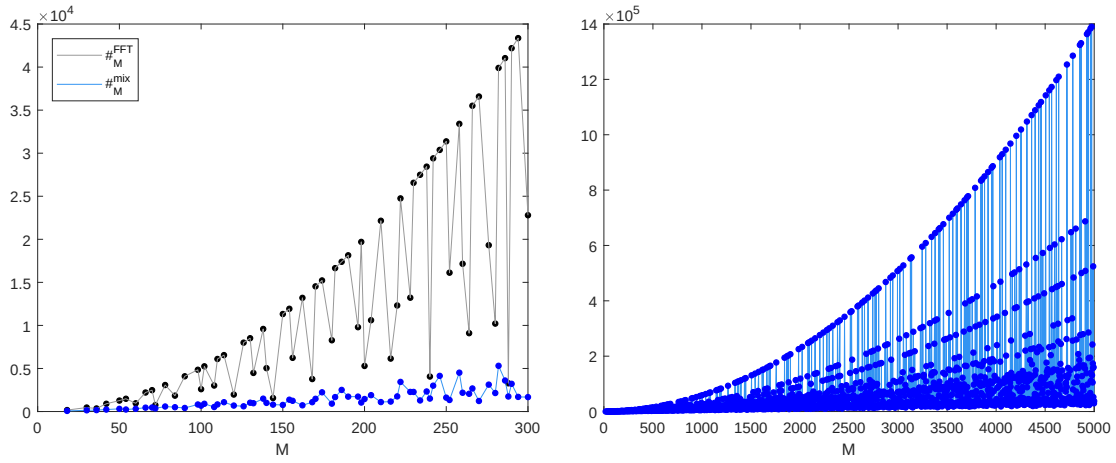
$$\#_M^{\text{FFT}} = 2^m p^2 + M\left(\frac{m}{2} - \frac{1}{p}\right) + 1,$$

where $M\left(\frac{m}{2} - \frac{1}{p}\right) + 1$ was due to the product by the exponentials whereas $2^m p^2$ resulted from the computation of the 2^m Fourier Transforms of length p. However, notice that, if p is not a prime number, then the transforms can be computed by means of the extended algorithm, which, as we have seen in Section 4.2, is faster. Therefore, if we denote as $\#_M^{\text{mix}}$ as the number of complex products required when applying this *mixed algorithm*, that is FFT until we get 2^m vectors of length p and then the extended algorithm for computing its transforms, then

$$\#_M^{\text{mix}} = 2^m \#_p^{\text{ext}} + M\left(\frac{m}{2} - \frac{1}{p}\right) + 1 = 2^m p\left(p_r + \sum_{k=1}^{r-1} \frac{p_{r-k} - 1}{\prod_{j=0}^{k-1} p_{r-j}}\right) + M\left(\frac{m}{2} - \frac{1}{p}\right) + 1, \tag{4.3.1}$$

where we are assuming $p = p_1, p_2 \cdots, p_r$ for $p_1 \geq p_2 \geq \cdots \geq p_r$ are prime numbers strictly greater than 1. Notice that if p is prime, then $\#_M^{\text{mix}} = \#_M^{\text{FFT}}$ since $p = p_r$ and the sum will have no terms.

Figure 4.2 (a) shows a comparative of the functions $\#_M^{\text{FFT}}$ and $\#_M^{\text{mix}}$ for that values of $M = 2^m p$ where p is not a prime number. In it, we can see the significant reduction that the mixed algorithm entails. Moreover, notice that, if we fix m, the behaviour of $\#_M^{\text{mix}}$ is going to be determined by that of $\#_p^{\text{ext}}$, which we have already discussed in Section 4.2. However, in comparison to the latter, the quadratic behaviour of the former requires greater values of M to be appreciated as Figure 4.2 (b) shows.



(a) (b)

Figure 4.2 (a) Plot of the functions $\#_M^{\text{FFT}}$ and $\#_M^{\text{mix}}$ at the points $M = 2^m p$ where p is not a prime number. **(b)** Plot of the function $\#_M^{\text{mix}}$ for the same points as (a).

4.4 Padding

So far, we have developed three different algorithms depending on the size of the vector: FFT (if it is a power of two), extended (if it is odd and non-prime) and mixed (if it is even but not a power of two). Since the first one appears to entail the greatest reduction, we are now going to analyse whether it is worth to pad the vector so as to apply the FFT. That is on what we are going to focus now.

As discussed in Section 3.4, padding is going to be useful as long as the error that follows from performing the FFT of the padded vector is small enough and the number of operations is reduced. Since Proposition 3.4.3 already gave an upper bound for the former, let us now fix our attention on the latter.

From now on, M is going to denote the length of the vector.

4.4.1 Vector of prime size

Notice that none of the algorithms developed so far apply to vectors of prime length, which means that its Fourier Transform can only be computed by means of the change-of-basis matrix, involving M^2 operations. Let us now analyse under which conditions $\#^{\text{FFT}}_{M_{PAD}} < M^2$. Observe that we can assume $M \geq 3$ since if $M = 1$, the Fourier Transform is the vector itself and if $M = 2$, $M = M_{PAD}$. Then, by Remark 3.4.2 and Proposition 3.2.1,

$$\begin{aligned}\#^{\text{FFT}}_{M_{PAD}} &= 2^{[\log_2 M]+1}\left(\frac{\log_2 2^{[\log_2 M]+1}}{2} - 1\right) + 1 = 2^{[\log_2 M]+1}\left(\frac{[\log_2 M]+1}{2} - 1\right) + 1 \\ &= 2^{[\log_2 M]}\left([\log_2 M] - 1\right) + 1 < 2^{[\log_2 M]}[\log_2 M] < M^2,\end{aligned}$$

where the first inequality is due to the fact that $2^{[\log_2 M]} > 1$ for $M \geq 3$.

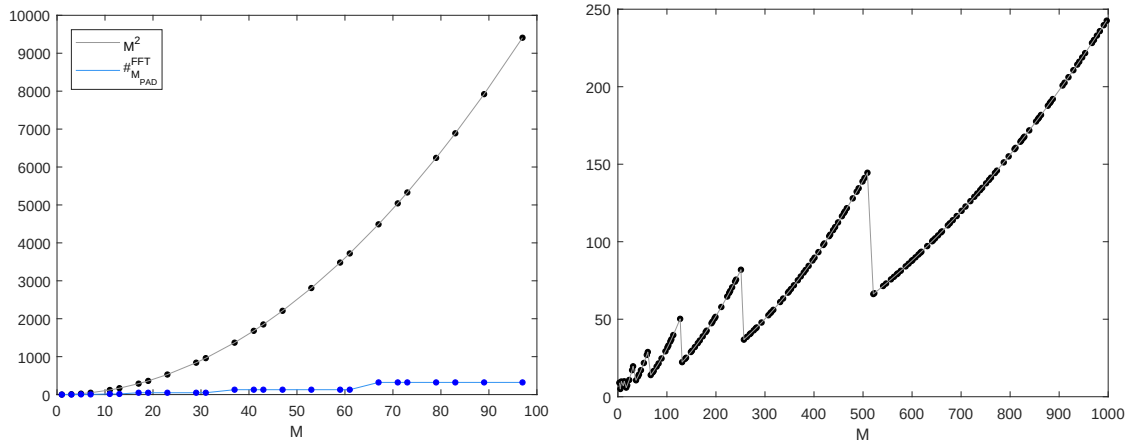
Therefore, if M is prime, it is always faster to compute the FFT of the padded vector and then keep the first M components, as Figure 4.3 (a) shows (provided that the error is small enough). Moreover, as can be seen in Figure 4.3 (b), the reduction that the padding entails is greater as M grows, although it is not an increasing function.

4.4.2 Vector of odd size but not prime

Following the notation introduced in Section 4.2, let $M = p_1 p_2 ... p_r$. In this case, the vector should be padded if $\#^{\text{FFT}}_{M_{PAD}} < \#^{\text{ext}}_{M}$, that is, using the computations done above,

$$2^{[\log_2 M]}\left([\log_2 M] - 1\right) + 1 < M\left(p_r + \sum_{k=1}^{r-1} \frac{p_{r-k} - 1}{\prod_{j=0}^{k-1} p_{r-j}}\right). \tag{4.4.1}$$

Unlike the other cases where we have studied the implementation of the padding, we are not going to manipulate this expression in order to isolate one of its variables since, when doing so, we get a less manageable inequality. Figure 4.4 (a) shows a comparison of these two functions. There, we can see that there are some points



(a) (b)

Figure 4.3 (a) Comparison of the functions M^2 and $\#^{\text{FFT}}_{M_{PAD}}$ over prime values of M. **(b)** Plot of the function $\frac{M^2}{\#^{\text{FFT}}_{M_{PAD}}}$ for M prime.

where padding is more time consuming, such as $M = 273$ or $M = 297$. Let us now see the reduction that follows from padding. Since $M \leq 2^{[\log_2 M]}$ for $M \geq 2$, then

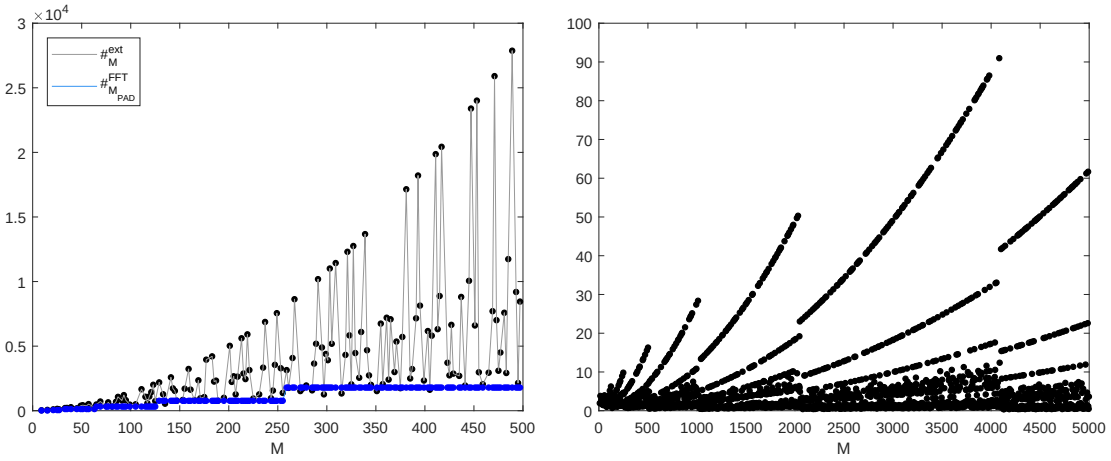
$$\begin{aligned}\frac{\#^{\text{ext}}_M}{\#^{\text{FFT}}_{M_{PAD}}} &= \frac{M\left(p_r + \sum_{k=1}^{r-1} \frac{p_{r-k}-1}{\prod_{j=0}^{k-1} p_{r-j}}\right)}{2^{[\log_2 M]}\left([\log_2 M]-1\right)+1} \geq \frac{p_r + \sum_{k=1}^{r-1} \frac{p_{r-k}-1}{\prod_{j=0}^{k-1} p_{r-j}}}{[\log_2 M]-1+\frac{1}{M}} \\ &= \frac{p_r + \frac{p_{r-1}-1}{p_r} + \frac{p_{r-2}-1}{p_{r-1}p_r} + \cdots + \frac{p_1-1}{p_2 p_3 \ldots p_r}}{[\log_2 M]-1+\frac{1}{M}} \xrightarrow[p_1\to\infty]{} \infty.\end{aligned}$$

Therefore, as p_1 increases so does the reduction. However, notice that this growth becomes significant when $p_1 >> p_2, p_3, ..., p_r$. This behaviour is depicted in Figure 4.4 (b).

4.4.3 Vector of even size but not a power of two

Following the notation introduced so far, recall that, for $M = 2^m p$, we get

$$\begin{aligned} M_{PAD} &= 2^{m+1+[\log_2 p]} \\ \#^{\text{FFT}}_{M_{PAD}} &= 2^{m+[\log_2 p]}(m+1+[\log_2 p]) - 2^{m+1+[\log_2 p]} + 1 \\ \#^{\text{mix}}_M &= 2^m p\left(p_r + \sum_{k=1}^{r-1} \frac{p_{r-k}-1}{\prod_{j=0}^{k-1} p_{r-j}}\right) + 2^m p\left(\frac{m}{2} - \frac{1}{p}\right) + 1. \end{aligned}$$



(a) (b)

Figure 4.4 (a) Comparison of the functions $\#_M^{\text{ext}}$ and $\#_{M_{PAD}}^{\text{FFT}}$ over odd and non-prime values of M. **(b)** Plot of the function $\frac{\#_M^{\text{ext}}}{\#_{M_{PAD}}^{\text{FFT}}}$ for M odd and non-prime.

After simplifying some terms, we see that $\#_M^{\text{mix}} > \#_{M_{PAD}}^{\text{FFT}}$ if and only if

$$p\left(p_r + \sum_{k=1}^{r-1} \frac{p_{r-k}-1}{\prod_{j=0}^{k-1} p_{r-j}}\right) + \frac{pm}{2} - 1 \; > 2^{[\log_2 p]}(m+1+[\log_2 p]) - 2^{1+[\log_2 p]}$$

$$\Updownarrow$$

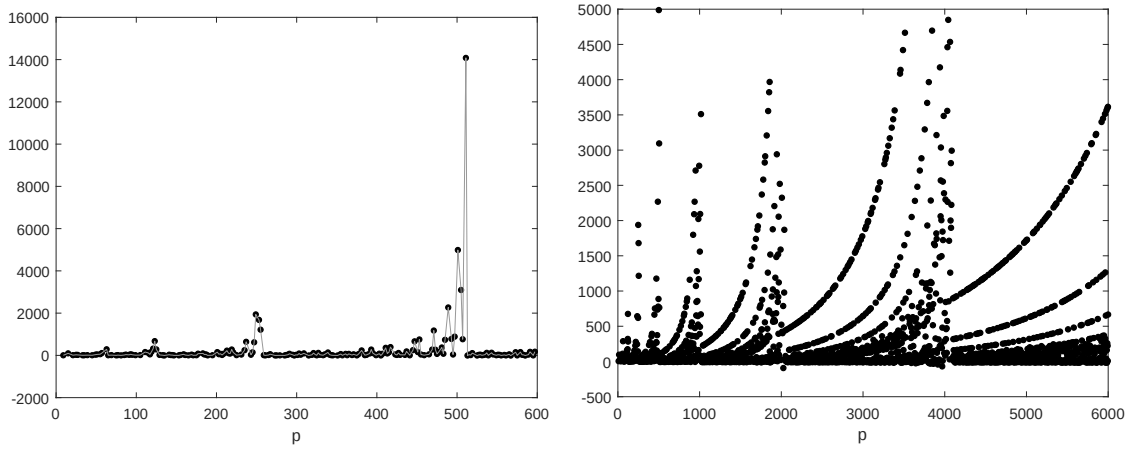
$$\frac{pm}{2} - 2^{[\log_2 p]}m > 2^{[\log_2 p]}(1+[\log_2 p]) - 2^{1+[\log_2 p]} - p\left(p_r + \sum_{k=1}^{r-1} \frac{p_{r-k}-1}{\prod_{j=0}^{k-1} p_{r-j}}\right) + 1.$$

As we observed in expression (3.4.9), $\frac{p}{2} - 2^{[\log_2 p]} < 0$ for all $p \in \mathbb{N}$, which implies that $\#_M^{\text{mix}} > \#_{M_{PAD}}^{\text{FFT}}$ if and only if

$$\begin{aligned} m &< \frac{2^{[\log_2 p]}(1+[\log_2 p]) - 2^{1+[\log_2 p]} - p\left(p_r + \sum_{k=1}^{r-1} \frac{p_{r-k}-1}{\prod_{j=0}^{k-1} p_{r-j}}\right) + 1}{\frac{p}{2} - 2^{[\log_2 p]}} \\ &= \frac{2^{[\log_2 p]+1}(1+[\log_2 p]) - 2^{2+[\log_2 p]} - 2p\left(p_r + \sum_{k=1}^{r-1} \frac{p_{r-k}-1}{\prod_{j=0}^{k-1} p_{r-j}}\right) + 2}{p - 2^{[\log_2 p]+1}} \\ &= \frac{2^{[\log_2 p]+1}([\log_2 p]-1) - 2p\left(p_r + \sum_{k=1}^{r-1} \frac{p_{r-k}-1}{\prod_{j=0}^{k-1} p_{r-j}}\right) + 2}{p - 2^{[\log_2 p]+1}}. \end{aligned} \tag{4.4.2}$$

Observe that this inequality shares some of the features of (3.4.10) as, for example, the fact that both functions blow up at those values of p where $[\log_2 p]$ is discontinuous. However, by looking at Figure 4.5, the function seems to carry out different

behaviours. For instance, notice that there are some values of p for which it is negative, which means that none of the vectors of length $M = 2^m p$ should be padded for such values of p.



(a) (b)

Figure 4.5 (a) Plot of the function given in (4.4.2) for values of p even and non-prime. **(b)** Plot of the same function than (a) but on a wider domain and limiting the y axis.

Observe that

$$\begin{aligned}
\frac{\#_M^{\text{mix}}}{\#_{M_{PAD}}^{\text{FFT}}} &= \frac{2^m p\left(p_r + \sum_{k=1}^{r-1} \frac{p_{r-k}-1}{\prod_{j=0}^{k-1} p_{r-j}}\right) + 2^m p\left(\frac{m}{2} - \frac{1}{p}\right) + 1}{2^{m+[\log_2 p]}(m+1+[\log_2 p]) - 2^{m+1+[\log_2 p]} + 1} \\
&\approx \frac{2^m p\left(p_r + \sum_{k=1}^{r-1} \frac{p_{r-k}-1}{\prod_{j=0}^{k-1} p_{r-j}}\right) + 2^m p\left(\frac{m}{2} - \frac{1}{p}\right)}{2^{m+[\log_2 p]}(m+1+[\log_2 p]) - 2^{m+1+[\log_2 p]} + 1} \\
&= \frac{p\left(p_r + \sum_{k=1}^{r-1} \frac{p_{r-k}-1}{\prod_{j=0}^{k-1} p_{r-j}}\right) + p\left(\frac{m}{2} - \frac{1}{p}\right)}{2^{[\log_2 p]}(m+1+[\log_2 p]) - 2^{1+[\log_2 p]} + \frac{1}{2^m}} \\
&= \frac{p\left(p_r + \sum_{k=1}^{r-1} \frac{p_{r-k}-1}{\prod_{j=0}^{k-1} p_{r-j}}\right) + p\left(\frac{m}{2} - \frac{1}{p}\right)}{2^{[\log_2 p]}(m-1+[\log_2 p]) + \frac{1}{2^m}}.
\end{aligned}$$

Now, using the fact that $2^{[\log_2 p]} < p$, we get

$$\begin{aligned}\frac{\#_M^{\text{mix}}}{\#_{M_{PAD}}^{\text{FFT}}} &> \frac{p\left(p_r + \sum_{k=1}^{r-1} \frac{p_{r-k}-1}{\prod_{j=0}^{k-1} p_{r-j}}\right) + p\left(\frac{m}{2} - \frac{1}{p}\right)}{p(m-1+[\log_2 p]) + \frac{1}{2^m}} \\ &= \frac{p_r + \sum_{k=1}^{r-1} \frac{p_{r-k}-1}{\prod_{j=0}^{k-1} p_{r-j}} + \frac{m}{2} - \frac{1}{p}}{m-1+[\log_2 p] + \frac{1}{2^m p}} \\ &= \frac{p_r + \frac{p_{r-1}-1}{p_r} + \frac{p_{r-2}-1}{p_{r-1}p_r} + \cdots + \frac{p_1-1}{p_2 p_3 \ldots p_r} + \frac{m}{2} - \frac{1}{p}}{m-1+[\log_2 p] + \frac{1}{2^m p}} \xrightarrow[p_1 \to \infty]{} \infty.\end{aligned}$$

Therefore, if we fix $p_2, p_3, ..., p_r$, then the larger p_1 is, the faster the computation of $\widehat{v_{PAD}}$ is in comparison to $\hat{v}$ (by means of the Fast Fourier Transform), provided that m satisfies inequality (4.4.2).

Summarising the results seen so far, assume that $v \in \ell^2(\mathbb{Z}_{2^m p})$ for some $m \in \mathbb{N}$ and $p \in \mathbb{N}_{\geq 3}$ odd and non-prime. Then, if m satisfies inequality (4.4.2) and $||\widehat{v_{PAD}}|M - \hat{v}||$ is small enough, it is faster to compute $\widehat{v_{PAD}}$ (by means of the FFT) and then keeping the first M components of it. On the contrary, if one of these two conditions fails to be true, we have to apply the mixed algorithm to the vector v itself.

4.5 General procedure to compute the Fourier Transform (summary)

Let us now put together all the results seen so far. In order to compute the Fourier Transform of a vector of length M, we have different approaches available depending on M and on the vector itself. We need to distinguish four different situations:

- If M is a power of two, then we apply the FFT, which, by Proposition 3.2.1, will give rise to
$$\#_M^{\text{FFT}} = M\left(\frac{\log_2 M}{2} - 1\right) + 1.$$
- Assume that M is not a power of two but is even, that is, $M = 2^m p$ for some p odd. If p is prime and m satisfies inequality (3.4.10) or p is non-prime and m satisfies inequality (4.4.2), then we apply the FFT to the padded vector, as long as $||\widehat{v_{PAD}}|_M - \hat{v}||_\infty$ is small enough (see Proposition 3.4.3). Otherwise, we apply the mixed algorithm which, as shown in expression (4.3.1), will require
$$\#_M^{\text{mix}} = 2^m p\left(p_r + \sum_{k=1}^{r-1} \frac{p_{r-k}-1}{\prod_{j=0}^{k-1} p_{r-j}}\right) + M\left(\frac{m}{2} - \frac{1}{p}\right) + 1.$$
- Following the notation introduced in Section 4.2, take M odd and non-prime. If inequality 4.4.1 is satisfied and, again, $||\widehat{v_{PAD}}|_M - \hat{v}||_\infty$ is small enough,

then we apply the FFT to the padded vector. Otherwise, we apply the extension of the algorithm that we have developed in this chapter which, by Proposition 4.2.1, will perform

$$\#_M^{\text{ext}} = M\left(p_r + \sum_{k=1}^{r-1} \frac{p_{r-k}-1}{\prod_{j=0}^{k-1} p_{r-j}}\right).$$

- Finally, if M is prime, we compute the transform of the padded vector. However, if the ensuing error is huge, we can only use the change-of-basis matrix, which implies M^2 products.

Notice that this allows us to define the following functions

$$\#_M = \begin{cases} \#_M^{\text{FFT}}, & \text{if } M \text{ is a power of two} \\ \#_M^{\text{mix}}, & \text{if } M \text{ is even but not a power of two} \\ \#_M^{\text{ext}}, & \text{if } M \text{ is odd and non-prime} \\ M^2, & \text{if } M \text{ is prime} \end{cases}$$

$$\#_M^{\text{PAD}} = \begin{cases} \#_M^{\text{FFT}}, & \text{if } M \text{ is a power of two} \\ \#_{M_{PAD}}^{\text{FFT}}, & \text{otherwise} \end{cases}$$

that gives the number of operations for each M depending on whether the padding can be implemented.

Figure 4.6 shows the plot of the function $\#_M$ for $1 \leq M \leq 1000$. By looking at M's of the same type we can see that, although the function is not increasing over them, its global tendency is. In addition, it shows that this tendency to increase is greater when M is prime. Then, it gets lower for M even and even lower for M odd. And, finally, it is the lowest when M is a power of 2.

In Figure 4.7 we can see the plot of the function $\#_M^{\text{PAD}}$ for $1 \leq M \leq 1000$. As expected, it is an increasing function that remains constant along each dyadic partition. Moreover, its growth is moderate and much lower than the one observed in $\#_M$, which shows its importance and justifies its implementation.

By looking at both figures, we realise that $\#_M$ may take very distinct values on two consecutive points. However, this variability is not so pronounced in $\#_M^{\text{PAD}}$ since that difference only occurs for that values of M near the discontinuity points and, besides, the length of the jump is rather moderate. Let us now illustrate this fact with an example.

Take $M = 512 = 2^9$. Since it is a power of two, its Fourier Transform is computed using the FFT, which involves

$$\#_{512} = \#_{512}^{\text{PAD}} = \#_{512}^{\text{FFT}} = 512\left(\frac{\log_2 512}{2} - 1\right) + 1 = 1793.$$

Now, consider $M = 510 = 2 \cdot 255$. Since $p = 255$ is non-prime ($255 = 3 \cdot 5 \cdot 17$) and $m = 1$ satisfies inequality (4.4.2), then we apply the FFT to the padded vector, as long as the error is small enough. Observe that $M_{PAD} = 512$, hence

$$\#_{510}^{\text{PAD}} = \#_{512}^{\text{FFT}} = 1793.$$

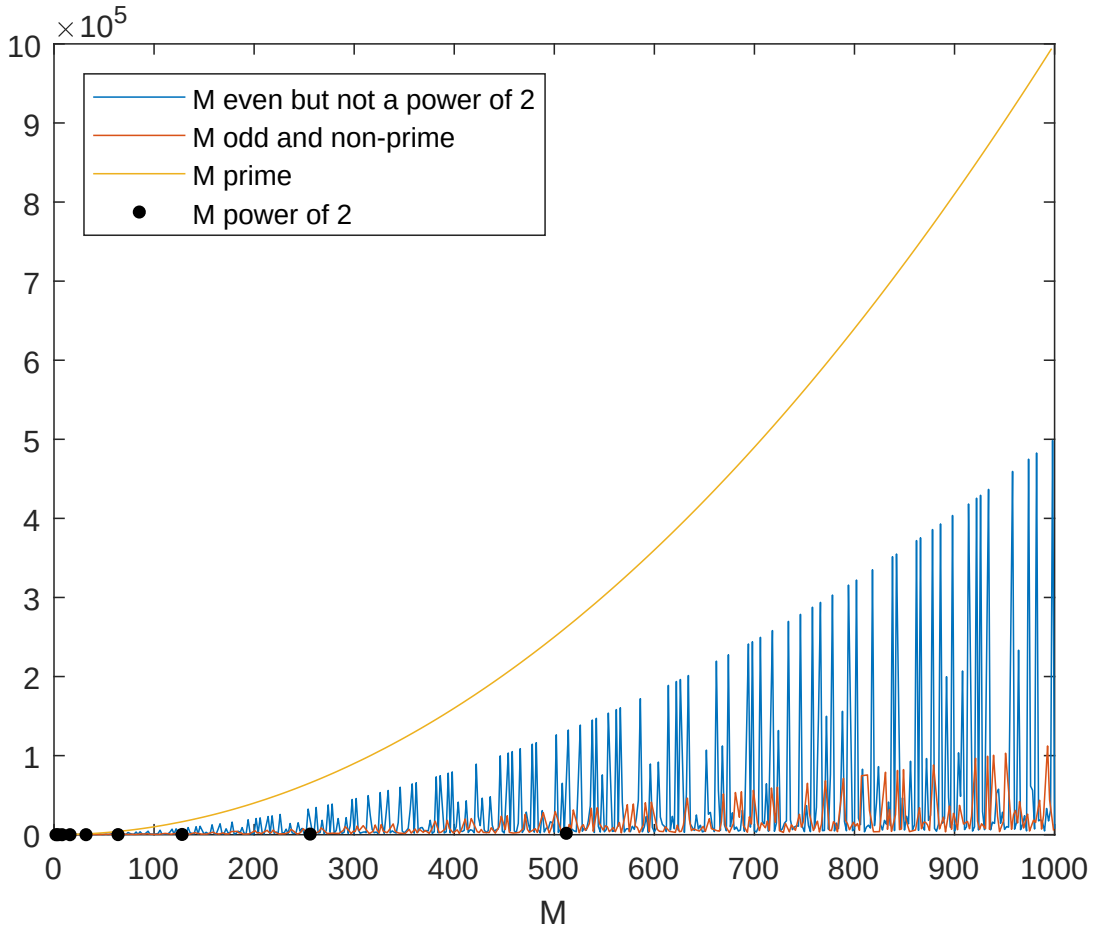


Figure 4.6 Plot of the function $\#_M$.

However, if the error is too large for our purposes, padding cannot be implemented, which implies that the computations have to be done by means of the mixed algorithm. That means

$$\#_{510} = \#_{510}^{\text{mix}} = 510\left(3 + \frac{5-1}{3} + \frac{17-1}{3\cdot 5}\right) + 510\left(\frac{1}{2} - \frac{1}{255}\right) + 1 = 3008,$$

which is greater but not that much. Notice that 510 and 512 belong to the same dyadic partition, thus $\#_{510}^{\text{PAD}} = \#_{512}^{\text{PAD}}$. However, if we increase the value of M until it reaches the next partition, that is $M = 514$, we expect $\#_{514}^{\text{PAD}}$ to be greater. Since $514 = 2 \cdot 257$ satisfies inequality (3.4.10) (notice that now $p = 257$ is prime), we can apply the FFT to the padded vector (now $M_{PAD} = 1024$) giving rise to

$$\#_{514}^{\text{PAD}} = \#_{1024}^{\text{FFT}} = 1024\left(\frac{\log_2 1024}{2} - 1\right) + 1 = 4097.$$

However, if $||\widehat{v_{PAD}}|_{514} - \hat{v}||_\infty$ is not negligible, the mixed algorithm must be used.

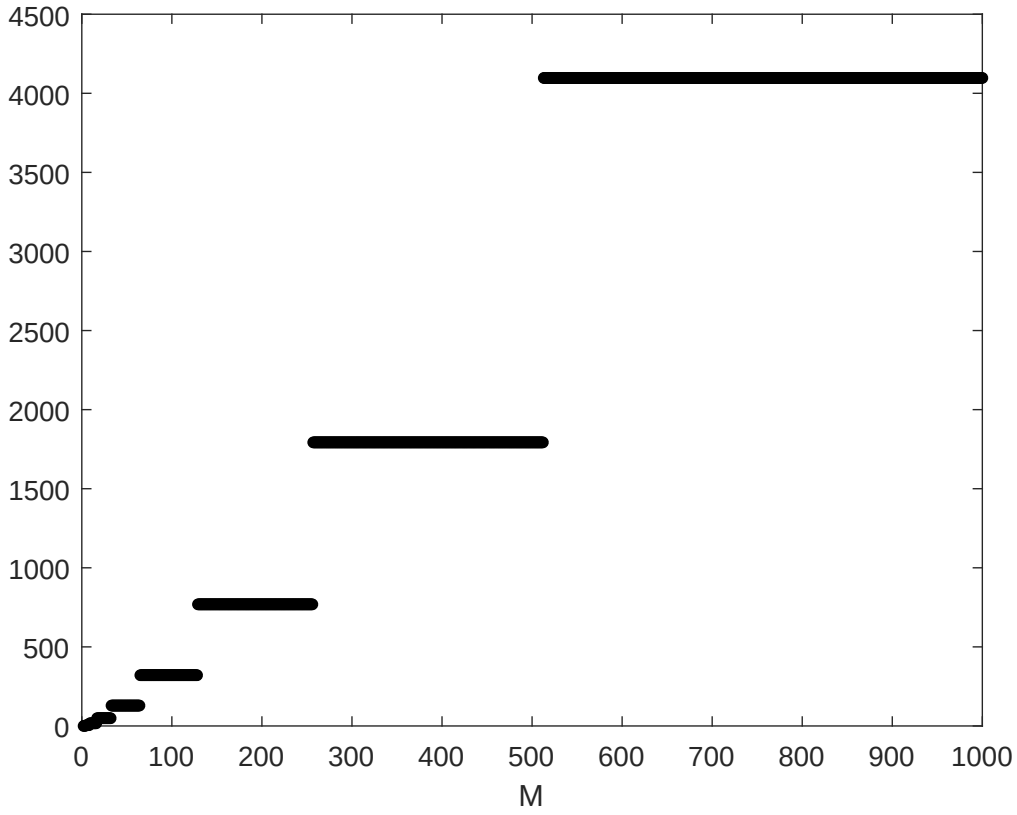


Figure 4.7 Plot of the function $\#_M^{\text{PAD}}$.

Then,

$$\#_{514} = \#_{514}^{\text{mix}} = 514 \cdot 257 + 514 \left(\frac{1}{2} - \frac{1}{257} \right) + 1 = 132\,354,$$

which becomes more than 32 times slower. Recall that if p is prime, $\#_{2^m p}^{\text{mix}} = \#_{2^m p}^{\text{FFT}}$. Take now $M = 519 = 3 \cdot 173$, which is odd and non-prime. Since it satisfies inequality (4.4.1), then, provided that the error is not significant, we apply the FFT to the padded vector, so

$$\#_{519}^{\text{PAD}} = \#_{1024}^{\text{FFT}} = 4097.$$

Otherwise, we compute the Fourier Transform through the extended algorithm, which performs

$$\#_{519} = \#_{519}^{\text{ext}} = 519 \left(3 + \frac{173 - 1}{3} \right) = 31\,313.$$

Finally, let $M = 521$, which is prime. Recall that in Section 4.4.1 we saw that padding such values of M always speeds up the computations. In this particular case it becomes more than 66 times faster

$$\begin{aligned} \#_{521}^{\text{PAD}} &= \#_{1024}^{\text{FFT}} = 4097 \\ \#_{521} &= 521^2 = 271\,441. \end{aligned}$$

Chapter 5

Digital image processing

In this chapter we are going to set the basis of digital image processing as well as developing the role played by the Discrete Fourier Transform. We will begin by giving the relationship between filtering, convolution and the DFT, which is going to be the key point [2]. After that, in Section 5.3, 5.4, 5.5 and 5.6 we will develop different examples of image processing that illustrates different applications of the DFT [1, 3, 7]. Such sections as well as the notation followed in them are introduced at the end of Section 5.2.

Throughout the chapter we are going to discuss different strategies to process a digital image which will be illustrated with particular examples. The codes used in such examples are shown in the Appendix.

5.1 Introduction to image processing

Let us begin by setting out what is going to be the object of study in this chapter, the images. An *image* can be regarded as a positive and finite two-dimensional function, $f(x, y)$, where the variables x and y, usually referred to as *spatial coordinates*, are continuously defined on a certain *spatial domain*. That is, $(x, y) \in [0, x_0] \times [0, y_0]$ for some $x_0, y_0 \in \mathbb{R}$. The value of the function at each point is called *intensity* or, since the images that we are going to deal with are grey-scaled, *grey-level*. However, when processing an image, this must be *digital*. This means that the spatial domain and the intensity values of f have to be discrete quantities. Therefore, the image has to undergo a previous step consisting of *sampling* (digitizing the spatial domain) and *quantization* (digitizing the intensity values). Although we are not going to go deeper in this previous step (in [3] is discussed in detail), it is important to mention that the discrete points obtained in the spatial domain are vertically and horizontally equidistant and that, for reasons concerning storage and quantizing hardware, the number of intensity levels is usually a power of two, being $2^8 = 256$ the most common.

Notice that, since the spatial coordinates become equidistant discrete quantities, we can assume without loss of generality that $x \in \{0, 1, 2, ..., M-1\}$ and that $y \in \{0, 1, 2, ..., N-1\}$ for some $M, N \in \mathbb{N}$. Therefore, a digital image given by a

function f can be regarded as an $M \times N$ matrix. That is,

$$f = \begin{pmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \vdots & & & \vdots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{pmatrix},$$

where each of its elements is called a *pixel*. For reasons that will become clear later on, we can extend them periodically, which implies that $f \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$.

From now on, we are going to deal with digital images, simply referred to as *images*, whose intensity levels take values in a partition of the interval $[0,1]$ consisting in 256 equidistant points. The first element of the partition is 0, which stands for black, and the last one is 1, which stands for white. As mentioned above, we are always considering $\mathbb{Z}_M \times \mathbb{Z}_N$ as the spatial domain and, in order to be consistent with the notation used so far, we are going to denote the spatial coordinates by (m,n) instead of (x,y).

Among all the different types of image processing, we are just going to consider those consisting in applying a *linear* and *translation-invariant* transformation, T, to a given image, f, so as to obtain another, g,

$$f \xrightarrow{T} g.$$

For instance, f could be a noisy picture and T an appropriate transformation for reducing such noise.

Notice that since T is linear, then the effect of the transformation on two images together is the sum of its effects on each image. Moreover, if we multiply the input picture by some amount, the output picture will be multiplied by that amount as well. Let us now define what a *translation-invariant* transformation is according to the notation introduced in Definition 2.1.9.

Definition 5.1.1. A transformation $T : \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N) \longrightarrow \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ is said to be *translation-invariant* if, for all $k_1, k_2 \in \mathbb{Z}$ and all $f \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$,

$$T(R_{k_1,k_2} f) = R_{k_1,k_2} T(f).$$

Thus, it means that the value of the response at one point only depends on the value of f at that point, not on its position. Therefore, the assumptions made on the transformation T seem reasonable.

5.2 Filtering

In Section 5.1 we have seen that processing an image consists in performing a linear and translation-invariant transformation that results in an image more suitable for a particular purpose. However, finding or carrying out such transformations is not always an easy task, and here is where convolution and the Fourier Transform come to play. Let us begin by introducing the key elements needed for achieving our target.

Definition 5.2.1. Given $g \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$, let $T_g : \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N) \longrightarrow \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ be defined as

$$T_g(f) = g * f,$$

for all $f \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$. Any linear transformation of this form is called a *convolution operator*.

Theorem 5.2.2. *If $T : \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N) \longrightarrow \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ is a linear and translation-invariant transformation, then each element of the Fourier basis is an eigenvector of T.*

Proof. Let $F_{p,q}$ be an element of the Fourier basis, which was defined in Proposition 2.1.1. Notice that, since F is a basis of $\ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$, then there exist complex coefficients $\{a_{r,s}\}_{0 \le r \le M-1, 0 \le s \le N-1}$ such that

$$T(F_{p,q})(m,n) = \sum_{r=0}^{M-1}\sum_{s=0}^{N-1} a_{r,s}F_{r,s}(m,n) = \frac{1}{\sqrt{MN}}\sum_{r=0}^{M-1}\sum_{s=0}^{N-1} a_{r,s}e^{2\pi irm/M}e^{2\pi isn/N} \tag{5.2.1}$$

for all $m, n \in \mathbb{Z}$. Now, observe that

$$\begin{aligned} R_{1,1}F_{p,q}(m,n) &= F_{p,q}(m-1,n-1) = \frac{1}{\sqrt{MN}}e^{2\pi ip(m-1)/M}e^{2\pi iq(n-1)/N} \\ &= e^{-2\pi ip/M}e^{-2\pi iq/N}F_{p,q}(m,n). \end{aligned}$$

Applying T to the previous expression, it becomes

$$\begin{aligned} T(R_{1,1}F_{p,q})(m,n) &= e^{-2\pi ip/M}e^{-2\pi iq/N}T(F_{p,q})(m,n) \\ &= e^{-2\pi ip/M}e^{-2\pi iq/N}\sum_{r=0}^{M-1}\sum_{s=0}^{N-1} a_{r,s}F_{r,s}(m,n) \\ &= \sum_{r=0}^{M-1}\sum_{s=0}^{N-1} e^{-2\pi ip/M}e^{-2\pi iq/N}a_{r,s}F_{r,s}(m,n), \end{aligned}$$

due to equation (5.2.1) and the linearity of T. On the other hand, using again equation (5.2.1),

$$\begin{aligned} (R_{1,1}T(F_{p,q}))(m,n) &= T(F_{p,q})(m-1,n-1) \\ &= \frac{1}{\sqrt{MN}}\sum_{r=0}^{M-1}\sum_{s=0}^{N-1} a_{r,s}e^{2\pi ir(m-1)/M}e^{2\pi is(n-1)/N} \\ &= \sum_{r=0}^{M-1}\sum_{s=0}^{N-1} a_{r,s}e^{-2\pi ir/M}e^{-2\pi is/N}F_{r,s}(m,n). \end{aligned}$$

Since T is translation-invariant,

$$\sum_{r=0}^{M-1}\sum_{s=0}^{N-1} e^{-2\pi ip/M}e^{-2\pi iq/N}a_{r,s}F_{r,s}(m,n) = \sum_{r=0}^{M-1}\sum_{s=0}^{N-1} e^{-2\pi ir/M}e^{-2\pi is/N}a_{r,s}F_{r,s}(m,n).$$

Using the uniqueness of the coefficients of an expression in terms of a basis, we get

$$e^{-2\pi ip/M}e^{-2\pi iq/N}a_{r,s} = e^{-2\pi ir/M}e^{-2\pi is/N}a_{r,s}.$$

Since $0 \leq p, r \leq M-1$ and $0 \leq q, s \leq N-1$, then $a_{r,s} = 0$ when $r \neq p$ and $s \neq q$. Therefore, equation (5.2.1) becomes

$$T(F_{p,q})(m,n) = a_{p,q}F_{p,q}(m,n),$$

that is

$$T(F_{p,q}) = a_{p,q}F_{p,q}.$$

Thus, any element of the Fourier basis is an eigenvector of T.

□

Let us now give the main result that relates convolution with image processing.

Theorem 5.2.3. *Let $T : \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N) \longrightarrow \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ be a linear transformation. Then, the following statements are equivalent:*

(i) T is translation invariant.

(ii) T is a convolution operator.

Proof. Let us begin by proving that $(ii) \Rightarrow (i)$. Let $g \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ be such that $T(f) = T_g(f) = g * f$. Take $f \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ and $k_1, k_2 \in \mathbb{Z}$. Then, for any $m, n \in \mathbb{Z}$,

$$\begin{aligned} T_g(R_{k_1,k_2}f)(m,n) = (g * R_{k_1,k_2}f)(m,n) &= \frac{1}{\sqrt{MN}}\sum_{r=0}^{M-1}\sum_{s=0}^{N-1} g(m-r,n-s)R_{k_1,k_2}f(r,s) \\ &= \frac{1}{\sqrt{MN}}\sum_{r=0}^{M-1}\sum_{s=0}^{N-1} g(m-r,n-s)f(r-k_1,s-k_2). \end{aligned}$$

We now make the change of variables $\ell_1 = r - k_1$ and $\ell_2 = s - k_2$. Then, by Remark 2.1.11,

$$\begin{aligned} T_g(R_{k_1,k_2}f)(m,n) &= \frac{1}{\sqrt{MN}}\sum_{\ell_1=-k_1}^{M-1-k_1}\sum_{\ell_2=-k_2}^{N-1-k_2} g(m-\ell_1-k_1,n-\ell_2-k_2)f(\ell_1,\ell_2) \\ &= \frac{1}{\sqrt{MN}}\sum_{\ell_1=0}^{M-1}\sum_{\ell_2=0}^{N-1} g(m-\ell_1-k_1,n-\ell_2-k_2)f(\ell_1,\ell_2) \\ &= (g*f)(m-k_1,n-k_2) = R_{k_1,k_2}(g*f)(m,n) \\ &= (R_{k_1,k_2}T_g(f))(m,n). \end{aligned}$$

Thus, T_g is translation invariant.

Let us now see that $(i) \Rightarrow (ii)$. Let $f \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$. Then, by Proposition 2.1.6,

$$f = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \hat{f}(p,q) F_{p,q}.$$

By Theorem 5.2.2, for each $F_{p,q} \in F$ there exists $\lambda_{p,q} \in \mathbb{C}$ such that $T(F_{p,q}) = \lambda_{p,q} F_{p,q}$. Then, since T is a linear transformation,

$$T(f) = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \hat{f}(p,q) T(F_{p,q}) = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \hat{f}(p,q) \lambda_{p,q} F_{p,q}.$$

Let us now define the matrix $A \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ such that $A(p,q) = \lambda_{p,q} \hat{f}_{p,q}$. Then, using equation (2.1.4),

$$T(f) = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} A(p,q) F_{p,q} = \overset{\vee}{A}. \tag{5.2.2}$$

Notice that, if we define the matrix $H \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ as $H(p,q) = \lambda_{p,q}$ and let $.*$ denote the component-wise product of matrices, then

$$A = \hat{f}. * H,$$

which means $A(m,n) = \hat{f}(m,n) H(m,n)$. Thus, by Proposition 2.1.13, equation (5.2.2) becomes

$$T(f) = (H. * \hat{f})^{\vee} = \left(\overset{\wedge}{\overset{\vee}{H}}. * \hat{f} \right)^{\vee} = \left(\left(\overset{\vee}{H} * f \right)^{\wedge} \right)^{\vee} = \overset{\vee}{H} * f = T_{\overset{\vee}{H}}(f).$$

Therefore, T is a convolution operator. □

This theorem states that, for any linear translation-invariant transformation T, there exists a suitable element $g \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$, called *filter*, such that applying T to an image is equivalent to performing the convolution of g with that image. However, this approach has two main drawbacks. Firstly, it is not always an easy task to find an appropriate filter and, what is more, convolution is a very time-consuming operation due to the large amount of multiplications that it requires. In view of this fact, we can take its Fourier Transform which, along with the result seen in Proposition 2.1.13, allows us to rewrite it as follows

$$(T(f))^{\wedge} = (g * f)^{\wedge} = \hat{g}. * \hat{f}.$$

Or, equivalently,

$$T(f) = (\hat{g}. * \hat{f})^{\vee}. \tag{5.2.3}$$

As a result, the computation of the convolution is almost reduced to that of a component-wise product of matrices since, as we have already discussed, the calculation of the DFT and the IDFT can be speeded up by means of the FFT. Let us now analyse such reduction.

On the one hand, it is clear that $f * g$ requires $(MN)^2$ operations (without taking into account the product by $1/\sqrt{MN}$). Observe that if g takes real values, all the products involved in the convolution are real, since f always takes real values in the interval $[0, 1]$. This means that, at least, the convolution performs $(MN)^2$ real multiplications.

On the other hand, we are going to assume that $MN = 2^m$ for some $m \in \mathbb{N}$ since, as we already discussed in Section 3, it is the most favourable case so as to apply the FFT. This assumption is indeed reasonable due to the fact that most of the images are of size $M \times N$ with both M and N a power of two. Let us now compute the number of complex products required in carrying out equation (5.2.3). First of all, notice that in the same way that we developed the FFT in Section 3.1, we could have developed the *Inverse Fast Fourier Transform (IFFT)* by working with the expression

$$\check{v}(k) = \frac{1}{\sqrt{MN}} \sum_{m=0}^{MN-1} v(m) e^{2\pi i k \frac{m}{MN}}$$

instead of

$$\hat{v}(k) = \frac{1}{\sqrt{MN}} \sum_{m=0}^{MN-1} v(m) e^{-2\pi i k \frac{m}{MN}},$$

which would have given rise to the same number of operations. Therefore, by Proposition 3.2.1, the number of complex multiplications required by equation (5.2.3) are

$$3\#_{MN}^{\text{FFT}} + MN = 3MN\left(\frac{\log_2(MN)}{2} - 1\right) + 3 + MN = MN\left(\frac{3\log_2(MN)}{2} - 2\right) + 3,$$

where $3\#_{MN}^{\text{FFT}}$ and MN are due to the Fourier transforms and the component-wise product of matrices respectively. As we showed at the beginning of Chapter 3, each complex product requires three real multiplications, which means that equation (5.2.3) needs

$$P_{real}^{FFT}(MN) = 3MN\left(\frac{3\log_2(MN)}{2} - 2\right) + 9$$

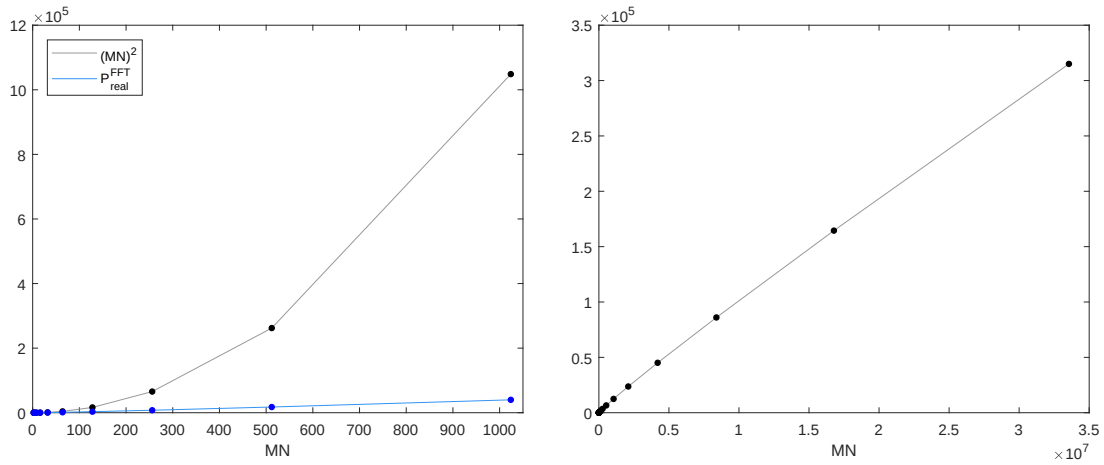
real products.

Figure 5.1 (a) shows a comparison of the number of operations required by both approaches whereas (b) depicts the reduction ensued from the use of the FFT, that is

$$\frac{(MN)^2}{P_{real}^{FFT}(MN)}.$$

For instance, if we take a picture of size 512×512, as we will do in Section 5.4, the computations by means of the FFT become almost 3 500 times faster.

In brief, if given a transformation T it is easy to find an appropriate filter g just by inspection of the spatial domain, then $T(f)$ can be quickly computed, for all $f \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$, using equation (5.2.3). Notice that, when doing so, $\hat{g}$ takes action



(a) (b)

Figure 5.1 (a) Comparison of the number of real products required by the convolution (black) and equation (5.2.3) (blue). **(b)** Quotient of the two expressions shown in (a).

on the Fourier Transform of the image, whose domain is called *frequency domain*. Thus, the filtering process is carried out in the frequency domain. This approach is based on inferring the filter by analysing the spatial domain. However, the choice of the filter depends on the purpose of the processing and there are some situations in which the information encoded in the spatial domain is not so easy to understand. If this is the case, an easier approach could be to interpret the frequency domain so as to obtain the filter, although this new filtering process will not follow at all the procedure shown in equation (5.2.3) but

$$T(f) = (G. * \hat{f})^{\vee},$$

where G is the filter inferred by inspection of the frequency domain and, recall, $.*$ is the component-wise product of matrices. Notice that, on the one hand, the DFT is used for speeding up the whole process and, on the other hand, for deducing a filter using the information encoded in the frequency domain.

Let us now stablish the setting for the following sections, which are examples of image processing based on filtering. In Section 5.3, we are going to infer the filter by inspection of the spatial domain whereas in Section 5.4 we are going to use the information encoded in the frequency domain so as to deduce a filter. The filtering process will be

$$f_{filt} = (\hat{g}. * \hat{f})^{\vee} \tag{5.2.4}$$

$$f_{filt} = (G. * \hat{f})^{\vee}, \tag{5.2.5}$$

respectively. Similarly to the latter, Section 5.5 will show how the modelling of a certain degradation function can lead to the definition of an appropriate filter. Finally,

in Section 5.6 we are going to develop an algorithm that allows the reconstruction of a picture with missing pixels by means of the DFT.

The notation that we are going to follow from now on, and that we have already used in equations (5.2.4) and (5.2.5), is the following: f is the input image, g and G are the filters in the spatial and frequency domains respectively and f_{filt} is the output or *filtered* image. Recall that we regard f, G and f_{filt} as elements of $\ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ and that $.*$ stands for component-wise product of matrices. In addition, we are going to regard M and N as even integers since it is the first step in the development of the techniques that we will consider next. Incidentally, [3] extends it to a general case.

5.3 Edge detection

In this section we are going to work on a kind of image processing where the output is a certain attribute of the input image: its *edges*. Given a picture, we define an *edge* as a set of consecutive pixels where the intensity changes abruptly within a small neighbourhood. In order to detect such features, we are going to determine a filter in the spatial domain so as to apply equation (5.2.4). The introduction of a discrete version of the Laplacian is going to be fundamental.

Take $f \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$. Then, we define

$$\frac{\partial^2 f(m,n)}{\partial m^2} = f(m+1,n) + f(m-1,n) - 2f(m,n)$$
$$\frac{\partial^2 f(m,n)}{\partial n^2} = f(m,n+1) + f(m,n-1) - 2f(m,n).$$

Therefore, the Laplacian is

$$\Delta^2 f(m,n) = f(m+1,n) + f(m-1,n) + f(m,n+1) + f(m,n-1) - 4f(m,n).$$

This operation has associated a so-called *mask*, which is a 3×3 matrix defined and denoted as follows

$$Mk = \begin{array}{|c|c|c|} \hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array} \ .$$

Computing the Laplacian of f at (m,n) is equivalent to centre the mask at (m,n) and then add the nine products resulting from multiplying each element of the mask with the corresponding component of f. In other words,

$$\begin{aligned} \Delta^2 f(m,n) = {} & Mk(0,0)f(m-1,n-1) + Mk(0,1)f(m-1,n) \\ & + Mk(0,2)f(m-1,n+1) + Mk(1,0)f(m,n-1) + Mk(1,1)f(m,n) \\ & + Mk(1,2)f(m,n+1) + Mk(2,0)f(m+1,n-1) + Mk(2,1)f(m+1,n) \\ & + Mk(2,2)f(m+1,n+1). \end{aligned}$$

Thus, it is enough to slide the mask along f to compute its Laplacian. Notice that this is equivalent to perform the convolution between f and the $\ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ filter

$$g = \sqrt{MN} \begin{pmatrix} \boxed{0} & 0 & \dots & 0 & \boxed{0} & \boxed{1} \\ 0 & & & & 0 & \\ \vdots & & & & \vdots & \\ 0 & & & & 0 & \\ \boxed{0} & 0 & \dots & 0 & \boxed{0} & \boxed{1} \\ \boxed{1} & 0 & \dots & 0 & \boxed{1} & \boxed{-4} \end{pmatrix} \tag{5.3.1}$$

where all the missing components are 0. The framed elements in the top left, top right, bottom left and bottom right of the matrix are the elements located at bottom right, bottom left, top right and top left, respectively, in the mask. In fact, this is the procedure followed when obtaining a filter from a given mask.

What makes the Laplacian so suitable for detecting fine detail is the following behaviour.

- It is zero in areas of constant intensity.
- It is zero in areas where the intensity increment in the horizontal and vertical directions is constant. If, for instance, the intensity values are integers in the interval $[1, 8]$, a region of this type could be

*	*	*	*	*	1	2	3	4
*	*	*	*	1	2	3	4	5
*	*	*	1	2	3	4	5	6
*	*	1	2	3	4	5	6	7
*	1	2	3	4	5	6	7	8
*	2	3	4	5	6	7	8	*
2	3	4	5	6	7	8	*	*

  This type of regions are called *ramps*.

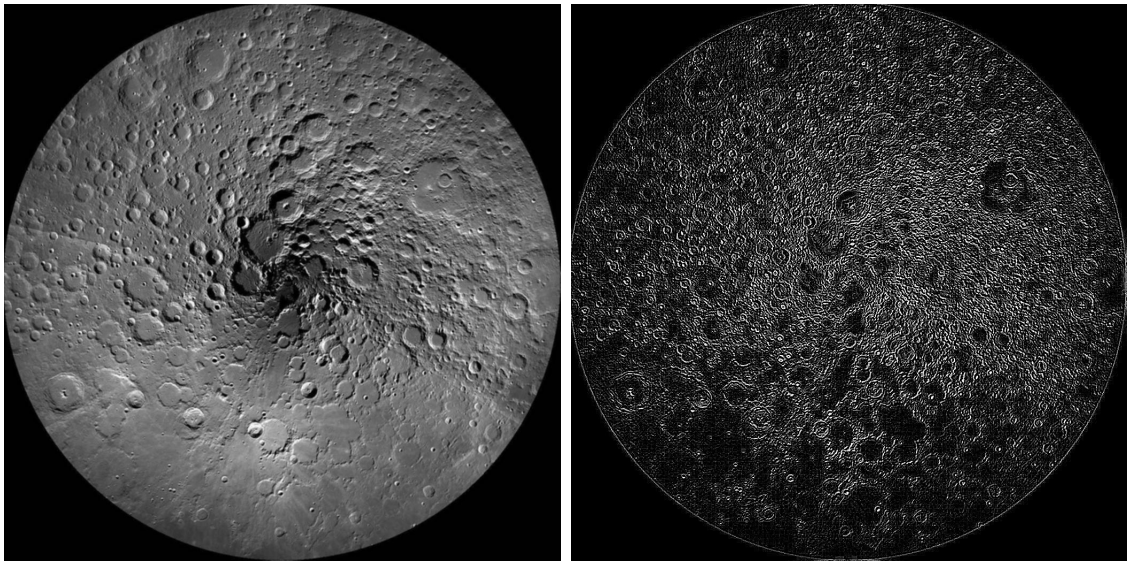
- It is non-zero at the onset and end of a ramp and step discontinuity (abrupt change).

Therefore, since edges are mainly characterised by relatively strong transitions in the intensity levels, the convolution of a picture with the filter given in equation (5.3.1) results in a black image showing the edges. As widely discussed in Section 5.2, convolution is implemented by means of the Fourier Transform (in order to speed computations) as shown in equation (5.2.4). Figure 5.2 and Figure 5.3 are two examples of the performance of this procedure. Notice the weird intensity levels of the pixels located in the right and bottom border due to the periodic extension of the image when filtering.

Notice that the difference of an image and its Laplacian results in a sharpened version of the first one, as shown in Figure 5.4.

(a) (b)

Figure 5.2 (a) Original picture. **(b)** Laplacian of (a).



(a) (b)

Figure 5.3 (a) Picture of the north pole of the moon taken from NASA's web page. **(b)** Laplacian of (a).

(a) (b)

Figure 5.4 (a) Result of performing the difference between the images shown in Figure 5.2. **(b)** Result of performing the difference between the images shown in Figure 5.3.

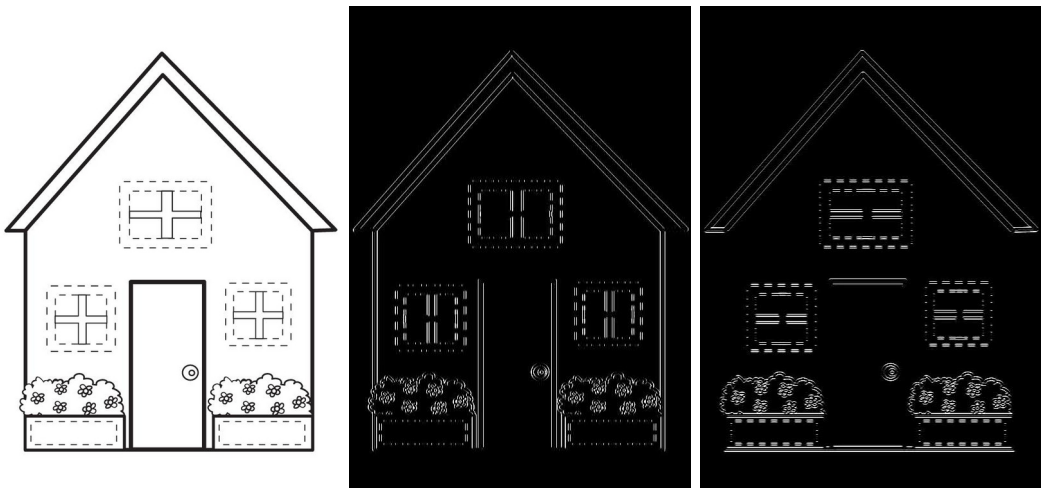
Notice that the Laplacian identifies all edges in an image. However, there are other masks that can detect them in a particular direction. For instance,

$$\begin{array}{|c|c|c|}\hline -1 & 2 & -1 \\ \hline -1 & 2 & -1 \\ \hline -1 & 2 & -1 \\ \hline \end{array} \quad \begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline 2 & 2 & 2 \\ \hline -1 & -1 & -1 \\ \hline \end{array} \tag{5.3.2}$$

recognise vertical and horizontal lines respectively. This is due, on the one hand, to their behaviour in areas where the intensity is constant or increases constantly and at the onset and end of a step discontinuity or ramp and, on the other hand, at the weight of its coefficients. Figure 5.5 shows an example of the performance of such masks. Recall that we first compute the filter using the method explained above and then we perform the convolution, which is actually computed using Discrete Fourier Transforms.

5.4 Periodic noise reduction

One of the most common degradation phenomenon is *periodic noise*, which is caused by electrical and electromechanical interference during the image capturing process. This type of alteration can be approximated by a finite sum of certain sinusoids as follows.



(a) (b) (c)

Figure 5.5 (a) Original picture. **(b)** and **(c)** are the results of applying the masks shown in expression (5.3.2) to (a).

Definition 5.4.1. Let $k \in \mathbb{N}$ and take $\{A_j\}_{1\leq j\leq k}$,$\{\varphi_j\}_{1\leq j\leq k} \subset \mathbb{R}$ sequences of real numbers. Assuming M and N even integers, take $\{u_j\}_{1\leq j\leq k} \subset \{0, 1, 2, ..., M/2 - 1\}$ and $\{v_j\}_{1\leq j\leq k} \subset \{0, 1, 2, ..., N/2 - 1\}$. Then, we say that $\eta \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ is a *periodic noise* if it can be expressed as follows

$$\eta(m, n) = \sum_{j=1}^{k} A_j \sin\left(2\pi u_j \frac{m}{M} + 2\pi v_j \frac{n}{N} + \varphi_j\right).$$

Remark 5.4.2. In Section 2.3, we discussed in detail the fact that

$$\sin\left(2\pi u_j \frac{m}{M} + 2\pi v_j \frac{n}{N} + \varphi_j\right) = \sin\left(2\pi(M - 1 - u_j)\frac{m}{M} + 2\pi(N - 1 - v_j)\frac{n}{N} + \varphi_j\right),$$

for $0 \leq u_j \leq M/2$ and $0 \leq v_j \leq N/2$. Therefore, there is no loss of generality when regarding the frequencies on the finite sets $\{u_j\}_{1\leq j\leq k} \subset \{0, 1, 2, ..., M/2 - 1\}$ and $\{v_j\}_{1\leq j\leq k} \subset \{0, 1, 2, ..., N/2 - 1\}$.

Notice that this type of noise is spatial dependant. Following the notation introduced at the end of Section 5.2, the input image f will be of the form

$$f = f_{orig} + \eta,$$

where f_{orig} is the *original picture* without the noise. Notice that f will look as the original image with some repeating pattern superimposed, as shown in Figure 5.6. Our aim is to obtain a filtered image that is as close as possible to the original one.

In order to characterise periodic noise the information encoded in the spatial domain is not enough whereas, as we will now see, the one given by the frequency

Figure 5.6 Corrupted image due to periodic noise.

domain is. Therefore, the strategy will consist in finding a suitable filter G so as to apply equation (5.2.5). In order to achieve so, we are going to take advantage of the following feature of the Fourier Transform of a sinusoid.

Lemma 5.4.3. *Let $\eta \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ be a sinusoid*

$$\eta(m,n) = A\sin\left(2\pi u\frac{m}{M} + 2\pi v\frac{n}{N} + \varphi\right),$$

where $A \in \mathbb{R}_+$, $0 \leq u \leq \frac{M}{2} - 1$, $0 \leq v \leq \frac{N}{2} - 1$ and $\varphi \in \mathbb{R}$. Then, its Fourier spectrum satisfies

$$|\hat{\eta}(p,q)| = \begin{cases} \frac{A\sqrt{MN}}{2}, & \text{if } p = u \text{ and } q = v \\ \frac{A\sqrt{MN}}{2}, & \text{if } p = M-u \text{ and } q = N-v \\ 0, & \text{otherwise} \end{cases}.$$

Proof. Take $0 \leq p \leq M-1$ and $0 \leq q \leq N-1$. Then,

$$\begin{aligned}
\hat{\eta}(p,q) &= \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} A\sin\left(2\pi u\frac{m}{M} + 2\pi v\frac{n}{N} + \varphi\right) e^{-2\pi ipm/M} e^{-2\pi iqn/N} \\
&= \frac{A}{2i\sqrt{MN}} \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \left(e^{2\pi i\left(u\frac{m}{M}+v\frac{n}{N}+\varphi\right)} - e^{-2\pi i\left(u\frac{m}{M}+v\frac{n}{N}+\varphi\right)}\right) e^{-2\pi ipm/M} e^{-2\pi iqn/N} \\
&= \frac{A}{2i\sqrt{MN}} e^{2\pi i\varphi} \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} e^{2\pi ium/M} e^{2\pi ivn/N} e^{-2\pi ipm/M} e^{-2\pi iqn/N} \\
&\quad - \frac{A}{2i\sqrt{MN}} e^{-2\pi i\varphi} \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} e^{-2\pi ium/M} e^{-2\pi ivn/N} e^{-2\pi ipm/M} e^{-2\pi iqn/N}
\end{aligned}$$

$$
\begin{aligned}
&= \frac{Ae^{2\pi i\varphi}\sqrt{MN}}{2i}\langle F_{u,v}, F_{p,q}\rangle - \frac{Ae^{-2\pi i\varphi}\sqrt{MN}}{2i}\langle F_{-u,-v}, F_{p,q}\rangle \\
&= \frac{iAe^{-2\pi i\varphi}\sqrt{MN}}{2}\langle F_{-u,-v}, F_{p,q}\rangle - \frac{iAe^{2\pi i\varphi}\sqrt{MN}}{2}\langle F_{u,v}, F_{p,q}\rangle \\
&= \begin{cases} \frac{iAe^{2\pi i\varphi}\sqrt{MN}}{2}, & \text{if } p = u \text{ and } q = v \\ \frac{iAe^{-2\pi i\varphi}\sqrt{MN}}{2}, & \text{if } p = -u \text{ and } q = -v \\ 0, & \text{otherwise} \end{cases}.
\end{aligned}
$$

Therefore,

$$
|\hat{\eta}(p,q)| = \begin{cases} \frac{A\sqrt{MN}}{2}, & \text{if } p = u \text{ and } q = v \\ \frac{A\sqrt{MN}}{2}, & \text{if } p = -u \text{ and } q = -v \\ 0, & \text{otherwise} \end{cases}.
$$

Finally, using the periodicity of $\hat{\eta}$ we get the desired result. □

This result implies that, if the periodic noise is as shown in Definition 5.4.1 and the amplitudes are big enough, then the spectrum of the Fourier Transform of the input image ($|\hat{f}|$) will show a collection of $2k$ intensity bursts, where each sinusoid contributes the pair located at $(p,q) = (u_j, v_j)$ and $(p,q) = (M - u_j, N - v_j)$. Since $f \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$, we can shift the matrix representation of $|\hat{f}|$ so as to be centred at the origin. That is, the point $(p,q) = (0,0)$ is located at the centre of the matrix. Observe that in such matrices, from now on referred to as $|\hat{f}|_{shift}$, the points of each burst pair are located in the conjugate positions $(M/2 - 1 + u_j, N/2 - 1 + v_j)$ and $(M/2 - 1 - u_j, N/2 - 1 - v_j)$. The main point of $\hat{f}_{shift}$ is the fact that it eases visualisation facilitating the choice of the filter. This fact is illustrated in Figure 5.7, which is the centred Fourier spectrum of the picture shown in Figure 5.6. In it, we can see two pairs of intensity spikes located on top of the vertical and horizontal light stripes that intersect in the center of the image. In order to ease visualisation, we have added four red circles surrounding them.

Let us now see how the periodic noise reduction is carried out. As mentioned above, such noise is identified by intensity spikes, which leads us to think that the filter should act as a patch that cancels them. Such filters are called *notches* and are constructed as follows.

Let f be an image that has been altered by a periodic noise whose parameters satisfy the hypothesis stated in Definition 5.4.1. That is, $f = f_{filt} + \eta$, where

$$
\eta(m,n) = \sum_{j=1}^{k} A_j \sin\left(2\pi u_j \frac{m}{M} + 2\pi v_j \frac{n}{N} + \varphi_j\right).
$$

Then, the notch is defined as

$$
G(p,q) = \prod_{j=1}^{k} H_j(p,q) H_{-j}(p,q), \tag{5.4.1}
$$

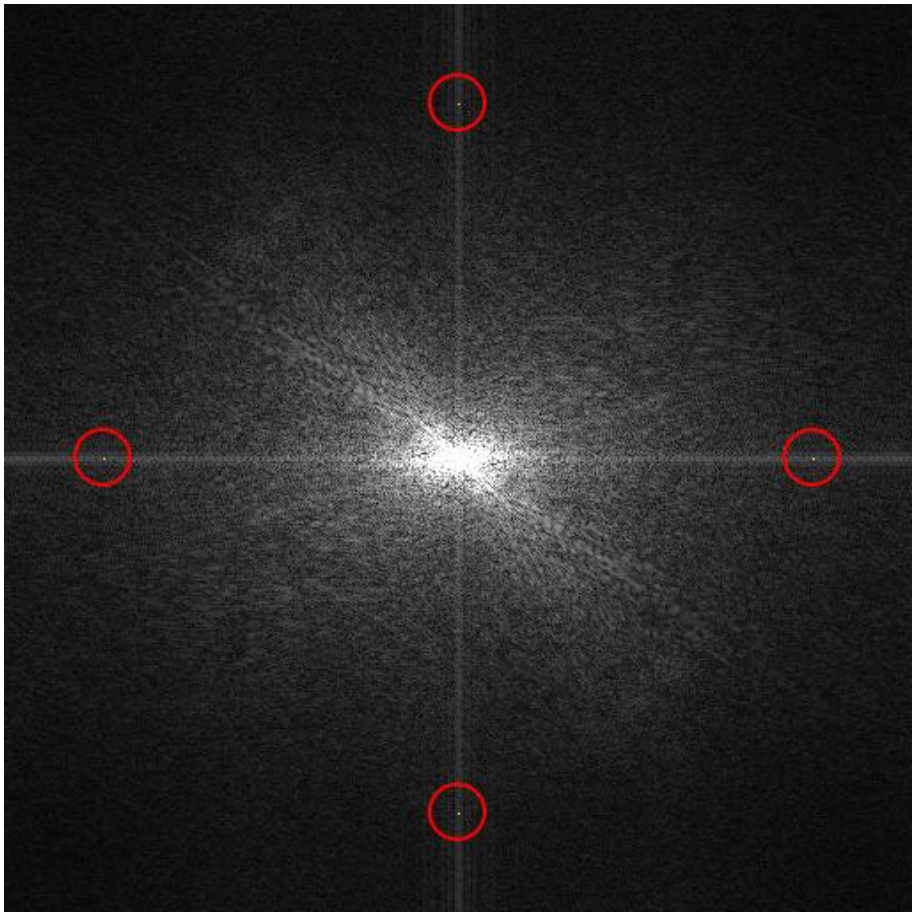


Figure 5.7 Centered Fourier spectrum of the picture shown in Figure 5.6. The red circles were added to highlight the intensity spikes caused by the periodic noise.

where H_j and H_{-j} are the *patch functions*. The main goal of each patch function is to eliminate one of the bursts without changing the other values of the frequency domain. This fact justifies the notation used for such functions: H_j and H_{-j} are supposed to remove the pair of bursts corresponding to the sinusoid $A_j \sin\left(2\pi u_j \frac{m}{M} + 2\pi v_j \frac{n}{N} + \varphi_j\right)$. As widely discussed in [3], there are different choices for the patch functions but, since the Gaussian is one of the most intuitive, this is the one that we are going to implement. Let us now define a Gaussian patch function.

Definition 5.4.4. We say that $H \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ is a *Gaussian patch function* centred at coordinates (a, b) if it is of the form

$$H(p, q) = 1 - e^{\frac{-D(p,q)^2}{2D_0^2}},$$

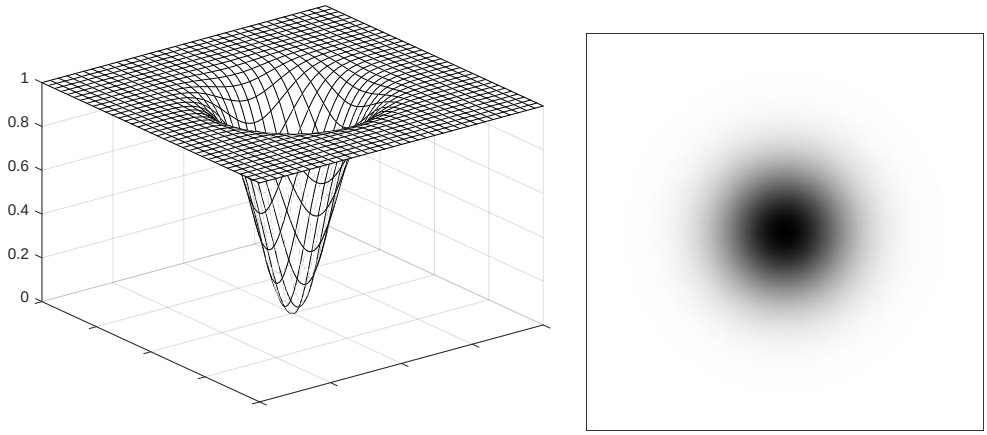
where $D_0 \in \mathbb{R}\backslash\{0\}$ is the *cut-off frequency* and $D(p, q)$ is the distance between the location (p, q) and (a, b). That is,

$$D(p, q) = \left[(p - a)^2 + (q - b)^2\right]^{\frac{1}{2}}.$$

Usually, Gaussian patches are centred at the center of the matrix, which is the point $(M/2-1, N/2-1)$. Then,

$$D(p,q) = \left[\left(p - \frac{M}{2} + 1\right)^2 + \left(q - \frac{N}{2} + 1\right)^2\right]^{\frac{1}{2}}.$$

Figure 5.8 shows a plot of a Gaussian patch function along with its image representation. The latter has a black border in order to distinguish the picture from the background. Therefore, when multiplying component-wise a Gaussian patch with a Fourier spectrum, the frequency where the patch is centred together with some neighbourhood determined by D_0 become essentially zero whereas the other values remain almost unchanged.



(a) (b)

Figure 5.8 (a) Plot of a Gaussian patch function. **(b)** Image representation of (a). The black borders were added for clarity.

Now, it is clear that, in order to eliminate the intensity spikes, the patch functions have to be Gaussians centred at these points. In other words, expression (5.4.1) can be rewritten as follows

$$G(p,q) = \prod_{j=1}^{k} \left(1 - e^{\frac{-D_j(p,q)^2}{2D_0^2}}\right)\left(1 - e^{\frac{-D_{-j}(p,q)^2}{2D_0^2}}\right),$$

where $D_0 \in \mathbb{R}\backslash\{0\}$ and $D_j(p,q)$ is the distance between the location (p,q) and the point $(M/2-1+u_j, N/2-1+v_j)$, that is,

$$\begin{cases} D_j(p,q) = \left[\left(p - \frac{M}{2} + 1 - u_j\right) + \left(q - \frac{N}{2} + 1 - v_j\right)^2\right]^{\frac{1}{2}} \\ D_{-j}(p,q) = \left[\left(p - \frac{M}{2} + 1 + u_j\right) + \left(q - \frac{N}{2} + 1 + v_j\right)^2\right]^{\frac{1}{2}} \end{cases}. \tag{5.4.2}$$

Unless we had some information about the periodic noise, the parameters D_0, u_j and v_j are usually chosen by trial and error. Notice that the Gaussians are centred at $(M/2-1+u_j, N/2-1+v_j)$ and $(M/2-1-u_j, N/2-1-v_j)$, which are the coordinates of the bursts in $|\hat{f}|_{shift}$. Thus, the processing is

$$(f_{filt}) = ((G. * \hat{f}_{shift})^{\star})^{\vee}, \tag{5.4.3}$$

where * stands for reverting the shifting. Notice that this procedure is slightly different from the one shown in equation (5.2.5). Coming back to the previous example, Figure 5.7 leads us to take the notch depicted in Figure 5.9 (a), which was obtained by taking $D_0 = 5$, $(u_1, v_1) = (200, 0)$ and $(u_2, v_2) = (0, 200)$. The black border is drawn for clarity. Finally, by performing the right hand-side of equation (5.4.3), we obtained the picture shown in Figure 5.9 (b). Notice that we have actually eliminated the periodic noise, which revealed as a repeating pattern.

(a) (b)

Figure 5.9 (a) Notch used for reducing the periodic noise present in Figure 5.6. **(b)** Figure 5.6 after filtering it using the notch in (a).

5.5 Blur reduction

In Section 5.4 we have modelled a particular case of degradation function as is periodic noise. The main point of such approach was to construct a filter G by inspection of the frequency domain so as to implement the formula shown in equation (5.2.5). In this section, however, the strategy is going to be slightly different. We are going to see how the modelling of a certain type of degradation functions can result, straightforwardly, in a useful filter. To some extent, it can be thought as another way of deducing equation (5.2.5), although the underlying idea is the same.

Throughout this section we are going to consider linear and translation-invariant degradation functions T. Notice that the periodic noise considered in Section 5.4 does not satisfy these conditions. Following the notation introduced at the beginning of Section 5.4, the input image can be expressed as

$$f = T(f_{orig}) + \eta,$$

where f_{orig} is the original image that we pretend to restore and $\eta \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ is some noise. By Theorem 5.2.3, there exists an element $h \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ such that

$$f = h * f_{orig} + \eta$$

and, by Proposition 2.1.13,

$$\hat{f} = \hat{h}. * \hat{f}_{orig} + \hat{\eta}.$$

Let us denote $H = \hat{h}$ and assume that $H(p,q) \neq 0$ for all p and q. Since the aim is the recovery of f_{orig}, then

$$f_{orig} = \left(\frac{\hat{f} - \hat{\eta}}{H}\right)^{\vee} = \left(\frac{\hat{f}}{H}\right)^{\vee} - \left(\frac{\hat{\eta}}{H}\right)^{\vee}, \tag{5.5.1}$$

where $\frac{\hat{f}-\hat{\eta}}{H}$ has to be understood as the component-wise division. As we said at the introduction of the chapter, we assume that we have some knowledge about the degradation function. That is, we have enough information about T so as to model the functions h or H but we don not know anything about the noise. Therefore, by equation (5.5.1), the filtered image must be

$$f_{filt} = \left(\frac{\hat{f}}{H}\right)^{\vee}, \tag{5.5.2}$$

which, by taking $G \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ as $G(m,n) = 1/H(m,n)$, can be expressed as

$$f_{filt} = (G. * \hat{f})^{\vee}.$$

Notice that, proceeding this way, not only have we obtained equation (5.2.5), but also the particular filter G. Thus, as long as H has non-zero components, equation (5.5.1) can be rewritten as

$$f_{orig} = f_{filt} - \left(\frac{\hat{\eta}}{H}\right)^{\vee},$$

where f_{filt} is given by equation (5.5.2). If we now take their Fourier Transforms, we get

$$\hat{f}_{filt} = \hat{f}_{orig} + \frac{\hat{\eta}}{H}.$$

Observe that the larger are the values of H, the closer is f_{filt} to f_{orig}. Therefore, we need some technique to prevent $\hat{\eta}/H$ from blowing up.

Let us begin by noticing an important feature of the Discrete Fourier Transform. Let $t \in \ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$ be such that it takes values on $\mathbb{R}_{\geq 0}$. Then,

$$\begin{aligned}\left|\hat{t}(p,q)\right| &= \frac{1}{\sqrt{MN}}\left|\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} t(m,n)e^{-2\pi ipm/}e^{-2\pi iqn/N}\right| \\ &\leq \frac{1}{\sqrt{MN}}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left|t(m,n)e^{-2\pi ipm/}e^{-2\pi iqn/N}\right| \\ &= \frac{1}{\sqrt{MN}}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} t(m,n) = \hat{t}(0,0).\end{aligned}$$

This implies that, under such hypothesis, the highest intensity is attained at the location $(0,0)$. This leads us to think that, as long as H takes non-negative values, we could perform the division by H just at some neighbourhood of $(0,0)$ (since, apparently, it reduces the probability of encountering zero-valued intensities) and, then, setting to zero the other values of the resulting matrix. This strategy makes sense since, as discussed in Section 2.3, a high value of the modulus of a particular Fourier coefficient implies a strong contribution of that frequency component to the image. Therefore, by doing so, we do not lose relevant information, which allows the properly recovery of the image. The filtering of the neighbourhood can be easily achieved by means of a patch function in the following way

$$P. * (\hat{f}_{filt})_{shift} = P. * \frac{\hat{f}_{shift}}{H_{shift}},$$

where P is some patch function that behaves as an identity matrix over frequencies near $(0,0)$ (which are now located at the centre of the matrix) while lowering the others. Notice that we want the performance of P to be the opposite of that of a Gaussian patch centred at the centre of the matrix. Therefore, we can define

$$P(p,q) = 1 - \left(1 - e^{\frac{-D(p,q)^2}{2D_0^2}}\right) = e^{\frac{-D(p,q)^2}{2D_0^2}}, \tag{5.5.3}$$

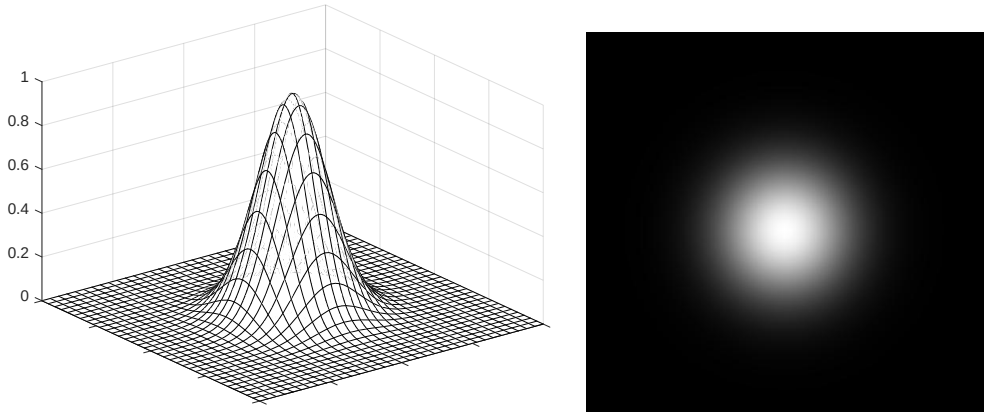
where the distance is given by

$$D(p,q) = \left[\left(p - \frac{M}{2} + 1\right)^2 + \left(q - \frac{N}{2} + 1\right)^2\right]^{\frac{1}{2}}.$$

Figure 5.10 shows a plot of such P along with its image representation. Thus, the filtered image is obtained as follows

$$f_{filt} = \left(\left(P. * \frac{\hat{f}_{shift}}{H_{shift}}\right)^{\star}\right)^{\vee}, \tag{5.5.4}$$

where P is as given above and, recall, $\star$ stands for reverting the shifting.



(a) (b)

Figure 5.10 (a) Plot of the patch function given in equation (5.5.3). **(b)** Image representation of (a).

Notice that, although we need the assumption that H takes strictly positive values, it is not enough to guarantee an accurate result. The reason is that the construction of the patch is based on the fact that the probability of encountering small values near the frequency $(0, 0)$ is low. Thus, if H is one of these non-probable matrices with small intensities near $(0, 0)$, then the method cannot be applied. This means that, even though H takes positive values, the procedure may not work properly. Therefore, this approach does not give a general way of proceeding but a well-suited approach for most of the cases encountered.

Let us finally see some examples of the performance of this strategy. We are going to consider the degradation caused by atmospheric turbulence, which is a linear and translation-invariant transformation. The model we are going to work with is the one proposed in [5], which is defined as

$$H(p, q) = e^{-k\left(p^2+q^2\right)^{5/6}},$$

where k depends on the environmental conditions. This phenomenon is quite common in images taken by satellites and produces a blurred image, as can be seen in Figure 5.11. In these cases, $k = 0.0025$ and $k = 0.0015$ respectively. Notice that the centred filter is given by

$$H_{shift}(p, q) = e^{-k\left(\left(p-\frac{M}{2}+1\right)^2+\left(q-\frac{N}{2}+1\right)^2\right)^{5/6}}, \tag{5.5.5}$$

which satisfies the condition of taking strictly positive values, although it becomes really small at the locations that are farthest from the centre of the matrix. These are the values that we want to get rid of.

In order to obtain the filtered image, given by expression (5.5.4), we need to determine the patch P. In other words, we have to find an appropriate value for the parameter D_0. As we have already discussed, this parameter is chosen so as to

(a) (b)

Figure 5.11 Images blurred due to atmospherical turbulence.

eliminate the smallest values of the centred spectrum H_{shift}. Therefore, D_0 could be approximated by inspection of H_{shift}, although, as mentioned in Section 5.4, such parameters are ultimately chosen by trial and error.

On the one hand, when filtering the image shown in Figure 5.11 (a), the patch with $D_0 = 125$ proves to give the best result. In Figure 5.12 (a) we can see the filtered image obtained by performing equation (5.5.4) with such D_0.

On the other hand, when filtering the image shown in Figure 5.11 (b), we realise (after trial-and-error) that no patch is actually required. In other words, the patch function is a constant matrix whose coefficients are all equal to 1. This means that, in this case, any value of the following component-wise division

$$\frac{\hat{f}_{shift}}{H_{shift}}$$

blow up. Therefore, there is no need of using a patch. Figure 5.12 (b) shows the filtered image obtained.

5.6 Image reconstruction

In this section we are going to see the role played by the Fourier transform when reconstructing an image from a picture where some of its pixels have been removed. We are going to assume that the missing pixels are uniformly distributed. Figure 5.13 (a) and the first column of Figure 5.14 show some pictures with a few number of available pixels where the missing ones are depicted in white to ease visualisation. This type of degradation usually arises when the signal sampling is not

(a) (b)

Figure 5.12 (a) Result of filtering the image shown in Figure 5.11 (a) with $D_0 = 125$. **(b)** Result of filtering the image shown in Figure 5.11 (b) without using a patch. In both pictures the blurring effect has disappeared.

carried out properly or due to the lose of some parts of the signal during transmission.

Let $X_{M\times N}$ be a signal where only the pixels located at $\Omega_a \subset M \times N$ are available. Then, we define $Y_{M\times N}$ as

$$Y(m,n) = \begin{cases} X(m,n) & \text{if } (m,n) \in \Omega_a \\ 0 & \text{if } (m,n) \in \Omega_m \end{cases},$$

for all $(m,n) \in M \times N$ where $\Omega_m = M \times N \setminus \Omega_a$. As discussed in Section 2.3, the module of the (p,q) coefficient of the Fourier transform of a signal is a measure of the strength of that frequency component in making up the signal. Since, in general, an image has a small number of strong frequency components, most of its Fourier coefficients are going to be almost zero. Therefore, a signal $Z_{M\times N}$ that minimises

$$||\hat{Z}||_1$$

can be regarded as a good approximation of X as long as $Z(m,n) = X(m,n)$ for all $(m,n) \in \Omega_a$. Notice that in order to reconstruct X properly, Z must keep the same frequency information. In other words, the spikes in the frequency domain must be the same although the smaller coefficients may vary. Thus, the more pixels X preserves, the more accurate is going to be the reconstruction, as can be seen comparing the first row of Figure 5.13 and the second one in Figure 5.14. Hence, the strategy to recover X consists in developing an algorithm that at each iteration

changes the values of the pixels of Y located at Ω_m in such a way that

$$||\hat{Y}_\ell||_1 > ||\hat{Y}_{\ell+1}||_1,$$

where Y_ℓ is the ℓ-th iteration.

Let us now develop the algorithm. Since we want to minimise the function $||\mathcal{F}(\cdot)||_1$, where $\mathcal{F}$ stands for the Fourier Transform, we are going to use the discrete version of the well-known gradient descent algorithm. It states that if G is a differentiable function, then the sequence

$$a_{k+1} = a_k - \mu_k \nabla G$$

converges to a minimum of G. Therefore, the sequence $\{Y_k\}_k$ obtained as

$$Y_{k+1}(m,n) = \begin{cases} Y_k(m,n) & \text{if } (m,n) \in \Omega_a \\ Y_k(m,n) - \mu_k \frac{||\mathcal{F}(Y_k + H^k_{(m,n)})||_1 - ||\mathcal{F}(Y_k - H^k_{(m,n)})||_1}{2h_k} & \text{if } (m,n) \in \Omega_m \end{cases},$$

where $H^k_{(m,n)}$ is a null matrix with value h_k at the component (m,n), converges component-wise to Z and, consequently, to X. Following the notation introduced so far, the algorithm that allows the reconstruction of the signal X is the following.

Require: Y, Ω_a, Ω_m, h_0, μ_0, it, α and β.
  Set $Y_0 = Y$
  for $t \in \{1, 2, ..., \text{it}\}$ **do**
    for $(m,n) \in \Omega_a \cup \Omega_m$ **do**
      if $(m,n) \in \Omega_m$ **then**
        $\text{Grad}^k(m,n) = \frac{||\mathcal{F}(Y_k + H^k_{(m,n)})||_1 - ||\mathcal{F}(Y_k - H^k_{(m,n)})||_1}{2h_k}$
      else
        $\text{Grad}^k(m,n) = 0$
      end if
    end for
    $Y_{k+1} = Y_k - \mu_k \text{Grad}^k$
    $h_{k+1} = h_k/\alpha$
    $\mu_{k+1} = \mu_k/\beta$
    $k \leftarrow k + 1$
  end for
  return Y_k

As discussed in previous sections, the way of determining the most appropriate values for the parameters is trial and error. However, there are some techniques ([1],[6]) to stablish the number of iterations required so as to bound the error caused, although we are not going to discuss them deeper. Figure 5.13 shows some examples obtained by varying the parameter values. In it we can see that modifying a bit the value of one parameter can result in a less accurate recovering. However, this fact can sometimes be resolved by increasing the number of iterations, as happens in (c)

(a) (b)

(c) (d)

Figure 5.13 (a) Shows an image with 30% of its pixels removed. **(b)**, **(c)** and **(d)** are the reconstructed images taking 5 iterations, $h_0 = 1$ and $\mu_0 = 0.5, 5, 0.5$, $\alpha = 2, 2, 0.2$ and $\beta = 2, 2, 5$ respectively.

if we consider 20 iterations. Picture (d), on the other hand, does not improve after 500 iterations.

Figure 5.14 shows two pictures where 70% and 90% of its pixels are missing along with the best recovery obtained after trial and error. Notice that although many pixels are missing, both pictures can be restored quite well. In Figure 5.15 we can see two more examples of image reconstruction. Notice that, in Figure 5.15 (a) and (b), the restoration procedure does a great job recovering the background, although the silhouette of the man and the camera does not appear to be sharp. One of the reasons for the latter is the fact that it is displayed in a larger size than the original picture, which was 256×256. On the other hand, the recovered image shown in (d) is very accurate even though (c) has only 15% of its pixels available.

So far, we have developed a method for reconstructing an image where the pixels removed were uniformly distributed. Let us now see if this approach works as well when a whole square of pixels is missing. Figure 5.16 shows some examples. First of all, the pictures, which were of size 256×256, are displayed in a larger size so as to ease visualisation, which causes some blurring. In the images we can see that the method performs well in regions of constant intensity or similar texture, as shown in (d), (e) and (k). Moreover, in picture (j) we have recovered the outline of the hat, which was removed. Finally, observe that the method is not well suited for restoring sharp or detailed pieces, as illustrated in (f) and (l).

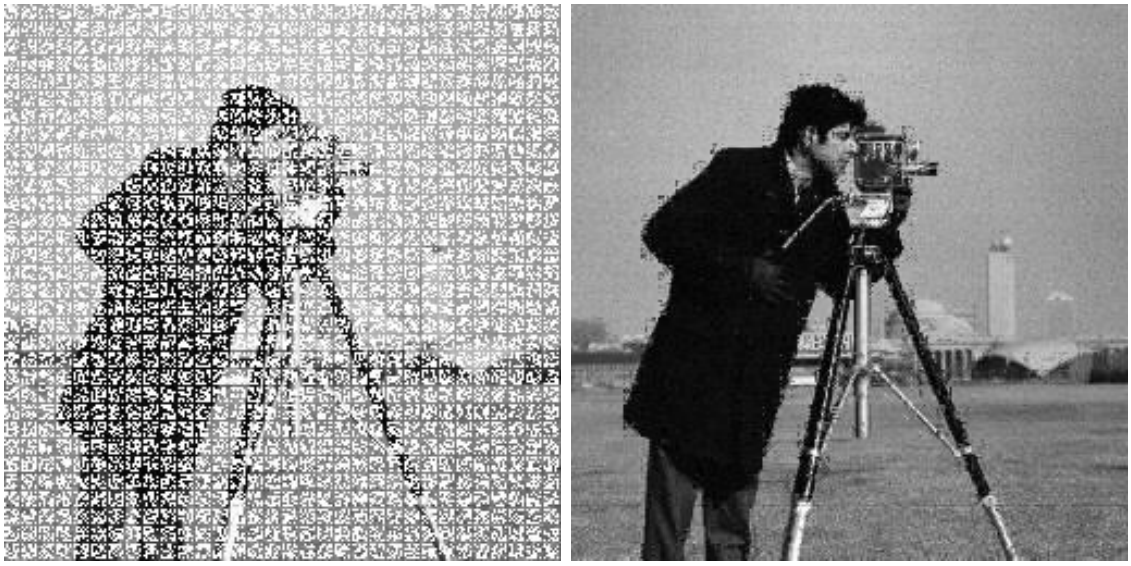
In summary, the method developed above is appropriate for reconstructing images where the removed pixels were uniformly distributed. Moreover, it yields good results when recovering missing squares located in regions of constant or similar texture as well as restoring outlines in areas with very few detail. In the Appendix we can find the code of three MATLAB functions that implement the algorithm for reconstructing an image (with both missing squares or uniformly distributed pixels). As mentioned there, the procedure is not carried out in the whole picture but in each of the 8×8 blocks that form the image. The reason is the fact that this procedure not only works for the Fourier Transform but for any other transformation for which most of the coefficients in the new basis are almost zero. For instance, the *Discrete Cosine Transform (DCT)* is a transformation very suitable for reconstructing purposes since it strongly satisfies this condition. Although we are not going to introduce the DCT here, we mention the fact that it is applied to a matrix by performing the transformation on each of its 8×8 blocks. Therefore, programming the function this way allows the implementation of the algorithm for both the DFT and the DCT. Moreover, it is less expensive computationally.

Due to this fact, the pixels removed in the examples were uniformly distributed within each block of 8×8. That is the reason why the degraded pictures may appear to be split into blocks.

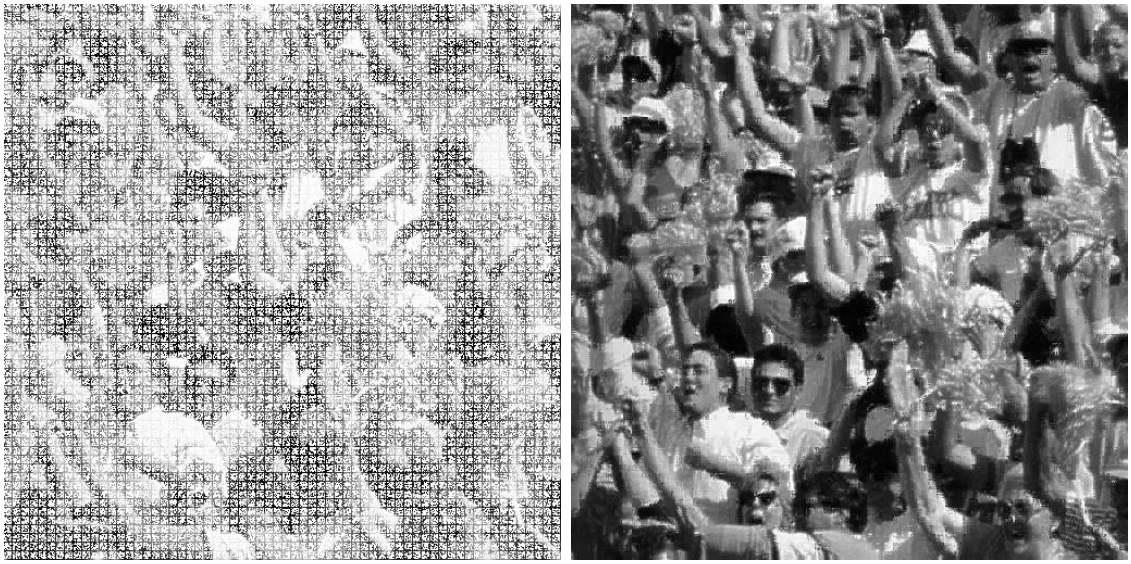
(a) (b)

(c) (d)

Figure 5.14 (a) and **(c)** show two pictures with 70% and 90% of its pixels removed. **(b)** and **(d)** are the reconstructed images taking 10 iterations, $h_0 = 1$, $\mu_0 = 1$, $\alpha = 1.5$ and $\beta = 1.5$.



(a) (b)



(c) (d)

Figure 5.15 (a) and **(c)** show two pictures with 60% and 85% of its pixels removed. **(b)** and **(d)** are the reconstructed images taking $h_0 = 0.5$, $\mu_0 = 1.5$, $\alpha = 1.2$ and $\beta = 1.2$ and 20 and 50 iterations respectively.

Figure 5.16 Pictures **(d)**, **(e)**, **(f)**, **(j)**, **(k)** and **(l)** are the reconstructions of **(a)**, **(b)**, **(c)**, **(g)**, **(h)** and **(i)** respectively. The parameters used are 100 iterations, $h_0 = 0.5$, $\mu_0 = 1$, $\alpha = \beta = 1.2$ and the squares are of size 10×10 pixels.

Appendix

Along Chapter 5 we have developed different instances of digital image processing and, in each of them, the approaches set were illustrated by means of different examples. In the appendix we present the codes of the functions developed so as to implement such strategies. These functions, which are programmed in MATLAB, were used to carry out all the examples appearing in the already mentioned chapter.

The aim of Section 5.3 was to obtain the edges of a given image. In order to achieve so we created a function, *EdgeDetect*, such that given an input image and a mask computes the filter associated to the latter and implements equation (5.2.4). The code is as follows.

```
function EdgeDetect(Im,Mk)
Im=imread(Im);
Im=im2double(Im);
if size(Im,3)~=1
    Im=rgb2gray(Im);
end
M=size(Im,1);
N=size(Im,2);

ImExt=[Im Im Im;Im Im Im;Im Im Im];
filter=zeros(3*M,3*N);
a=(size(Mk,1)-1)/2;
b=(size(Mk,2)-1)/2;
filter(3*M-a:3*M,3*N-b:3*N)=Mk(1:a+1,1:b+1); %bottom right
filter(3*M-a:3*M,1:b)=Mk(1:a+1,b+2:2*b+1); %bottom left
filter(1:a,1:b)=Mk(a+2:2*a+1,b+2:2*b+1); %top left
filter(1:a,3*N-b:3*N)=Mk(a+2:2*a+1,1:b+1); %top right
filter=sqrt(M*N)*filter;

FourTransImExt=fft2(ImExt)/sqrt(M*N);
FourTransFilter=fft2(filter)/sqrt(M*N);
Conv=sqrt(M*N)*ifft2(FourTransImExt.*FourTransFilter);
FiltIm=Conv(M+1:2*M,N+1:2*N);
imshow(FiltIm)
end
```

Notice that we regard images as elements of $\ell^2(\mathbb{Z}_M \times \mathbb{Z}_N)$, so they should be extended periodically (as shown in line 10). Therefore, the computations are done with finite matrices of sizes $3M \times 3N$. However, the result that we are looking for is not the filtered $3M \times 3N$ matrix but the middle submatrix of size $M \times N$ (line 23).

Moreover, this function works for any mask of length $(2a+1) \times (2b+1)$ for $a, b \in \mathbb{Z}$, although all the masks considered in Section 5.3 where of size 3×3, that is, $a = b = 1$.

In order to ease visualisation in the pictures shown in Figure 5.2 and 5.3, we added the command

```
FiltIm=5.5*log(1+FiltIm);
```

and

```
FiltIm=7.5*log(1+FiltIm);
```

before the *imshow* respectively. Its purpose was to extend a narrow range of intensity levels located near 1 since, otherwise, the edges were too light.

Observe that this function is very general because it computes the convolution of an image with the filter associated to the input mask. Since we were just interested in edge detection, we executed it for the particular masks associated to that task.

In order to eliminate periodic noise (Section 5.4), we generated a function, *PerNoise*, such that, given the noisy picture and the notch, it performs the filtering and shows the result.

```
function PerNoise(NoisyIm,Notch)
if ischar(NoisyIm)==1
    NoisyIm=imread(NoisyIm);
end
NoisyIm=im2double(NoisyIm);
if size(NoisyIm,3)~=1
    NoisyIm=rgb2gray(NoisyIm);
end
M=size(NoisyIm,1);
N=size(NoisyIm,2);

FourTrans=fft2(NoisyIm)/sqrt(M*N);
Recovered=sqrt(M*N)*real(ifft2(ifftshift(Notch.*fftshift(FourTrans))));
imshow(Recovered)
end
```

In our case, the degraded picture seen in Figure 5.6 was created by adding periodic noise to the original one. The procedure was

```
eta1=zeros(M,N);
eta2=zeros(M,N);
for m=1:1:M
    for n=1:N
        eta1(m,n)=0.5*sin(2*pi*200*(m-1)/M);
        eta2(m,n)=0.5*sin(2*pi*200*(n-1)/N);
    end
end
eta=eta1+eta2;
NoisyIm=Im+eta;
```

where Im is the original grey-scaled image expressed as a matrix with values in $[0, 1]$. Finally, the notch, which is shown in Figure 5.9 (a), was constructed as follows

```
CtOf=5;  % cut-off frequency
Notch=zeros(M,N);
for p=1:M
    for q=1:N
        A=norm([p-M/2-200 q-N/2],2)^2;
        B=norm([p-M/2+200 q-N/2],2)^2;
        C=norm([p-M/2 q-N/2-200],2)^2;
        D=norm([p-M/2 q-N/2+200],2)^2;
        Notch(p,q)=(1-exp(-A/(2*CtOf^2)))*(1-exp(-B/(2*CtOf^2)))*(1-exp
            (-C/(2*CtOf^2)))*(1-exp(-D/(2*CtOf^2)));
    end
end
```

Notice that the distances considered are slightly different than the ones seen in equation (5.4.2) due to the fact that MATLAB indexes matrices beginning at 1, not at 0.

In Section 5.5 we saw how to filter an image that had been degraded by a linear and translation-invariant transformation whose corresponding filter H (following the notation introduced then) can be modelled. In order to perform such filtering, we designed a function, *BlurRed*, that follows the strategy shown in equation (5.5.4). The inputs are the noisy picture, the shifted model H and the cut-off value for the patch function.

```
function BlurRed(NoisyIm,HShift,CtOf)
if ischar(NoisyIm)==1
    NoisyIm=imread(NoisyIm);
end
NoisyIm=im2double(NoisyIm);
if size(NoisyIm,3)~=1
    NoisyIm=rgb2gray(NoisyIm);
end
M=size(NoisyIm,1);
N=size(NoisyIm,2);

Patch=zeros(M,N);
for p=1:M
    for q=1:N
        Patch(p,q)=exp(-norm([p-M/2 q-N/2],2)^2/(2*CtOf^2));
    end
end

FourSpec=fft2(NoisyIm)/sqrt(M*N);
Recovered=sqrt(M*N)*real(ifft2(ifftshift(Patch.*fftshift(FourSpec)./
    HShift)));
imshow(Recovered)

end
```

The examples that we worked on consisted in images degraded due to atmospheric turbulence. Therefore, according to equation (5.5.5), the second input of the function had to be

```
k=0.0025;
```

```
HShift=zeros(M,N);
for p=1:M
    for q=1:N
        HShift(p,q)=exp(-k*((p-M/2)^2+(q-N/2)^2)^(5/6));
    end
end
```

In fact, the noisy image was obtained by applying H to the Fourier Transform of the original image and then computing its Inverse Fourier Transform. That is

```
FourIm=fft2(Im)/sqrt(M*N);
NoisyIm=sqrt(M*N)*real(ifft2(ifftshift(fftshift(FourIm).*HShift)));
```

where Im is the original image and $HShift$ is the filter defined above. Finally, as discussed in the section itself, the cut-off chosen for Figure 5.11 (a) was 125 whereas for Figure 5.11 (b) the patch was a constant matrix with all its components equal to 1. Therefore, in order to filter the latter we can execute the function taking a large enough cut-off or we can modify it by suppressing the $Patch$.

Finally, in Section 5.6 we gave an algorithm for reconstructing an image with missing pixels. In order to implement it, we designed a function, *Reconstruction*, that removes s pixels (uniformly distributed) of a given image and then reconstructs it. Since, as already mentioned in Section 5.6, the recovery is carried out in each of the 8×8 blocks that composes the picture, we defined another function which is called by the first one, *BlckRecons*, that adds noise and restores each of the blocks. The notation for the parameters is the one introduced in the already mentioned section.

```
function Reconstruction(Im,s,iter,h_0,mu_0,alpha,beta)
Im=imread(Im);
if size(Im,3)~=1
    Im=rgb2gray(Im);
end
Im=im2double(Im);
M=size(Im,1);
N=size(Im,2);

if floor(M/8)==M/8
    auxM=M/8;
else
    auxM=floor(M/8)+1;
end
if floor(N/8)==N/8
    auxN=N/8;
else
    auxN=floor(N/8)+1;
end

ImExt=[Im Im; Im Im];
Blocks=cell(auxM,auxN);
BlocksRec=cell(auxM,auxN);
BlocksNoisy=cell(auxM,auxN);
for i=1:auxM
```

```
    for j=1:auxN
        Blocks{i,j}=ImExt(8*(i-1)+1:8*(i-1)+1+7,8*(j-1)+1:8*(j-1)+1+7);
        [BlocksNoisy{i,j},BlocksRec{i,j}]=BlckRecons(Blocks{i,j},s,iter
            ,h_0,mu_0,alpha,beta);
    end
end

NoisyIm=cell2mat(BlocksNoisy);
RecIm=cell2mat(BlocksRec);
NoisyIm=NoisyIm(1:M,1:N);
RecIm=RecIm(1:M,1:N);

subplot(1,2,1);imshow(NoisyIm),title('Degraded')
subplot(1,2,2);imshow(RecIm),title('Reconstructed')
end
```

where

```
function [BlVis,y]=BlckRecons(Block,s,iter,h_0,mu_0,alpha,beta)
M=8;
N=8;
Rows=floor(1+(M-1)*rand(1,s));
Colms=floor(1+(N-1)*rand(1,s));
NoisyBl=Block;
BlVis=Block;
for i=1:s
    NoisyBl(Rows(i),Colms(i))=0;
    BlVis(Rows(i),Colms(i))=1;
end

%Gradient descent algorithm
y=NoisyBl;
mu=mu_0;
for t=1:iter
    Grad=zeros(M,N);
        for i=1:s
            a=Rows(i);
            b=Colms(i);
            Aux=zeros(M,N);
            Aux(a,b)=h_0;
            X_pos=fft2(y+Aux)/sqrt(M*N);
            X_neg=fft2(y-Aux)/sqrt(M*N);
            Grad(a,b)=1/(2*h_0)*(sum(sum(abs(X_pos)))-sum(sum(abs(X_neg
                )))));
        end
        y=y-mu*Grad;

    h_0=h_0/alpha;
    mu=mu/beta;
end
end
```

Notice that the given image is extended periodically. Finally, we implemented a function, *QuadRec*, that adds a square of size $quad \times quad$ to a given image in a

random location and uses the gradient descent algorithm to restore it.

```
function QuadRec(Im,iter,h_0,mu_0,alpha,beta,quad)
Im=imread(Im);
if size(Im,3)~=1
    Im=rgb2gray(Im);
end
Im=im2double(Im);
M=size(Im,1);
N=size(Im,2);

Rows=floor(1+(M-quad+1)*rand);
Colms=floor(1+(N-quad+1)*rand);
NoisyIm=Im;
ImVis=Im;
NoisyIm(Rows:Rows+quad,Colms:Colms+quad)=0;
ImVis(Rows:Rows+quad,Colms:Colms+quad)=1;

%Gradient descent algorithm
y=NoisyIm;
mu=mu_0;

for t=1:iter
    Grad=zeros(M,N);
        for i=Rows:Rows+quad
            for j=Colms:Colms+quad
            a=i;
            b=j;
            Aux=zeros(M,N);
            Aux(a,b)=h_0;
            X_pos=fft2(y+Aux)/sqrt(M*N);
            X_neg=fft2(y-Aux)/sqrt(M*N);
            Grad(a,b)=1/(2*h_0)*(sum(sum(abs(X_pos)))-sum(sum(abs(X_neg
                ))));
            end
        end
    y=y-mu*Grad;
    h_0=h_0/alpha;
    mu=mu/beta;
end
subplot(1,2,1);imshow(ImVis),title('Degraded')
subplot(1,2,2);imshow(y),title('Reconstructed')
end
```

Bibliography

[1] A. Draganić, I. Orović and S. Stanković, *On some common compressive sensing recovery algorithms and applications,* Facta Universitatis, Series: Electronics and Energetics **30** (2017), no. 4, 477–510.

[2] M. W. Frazier, *An Introduction to Wavelets Through Linear Algebra*, Springer, 1999.

[3] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Pearson Prentice Hall, 2008.

[4] R. C. Gonzalez, R. E. Woods and S. L. Eddins, *Digital Image Processing USING MATLAB*, Gatesmark, 2009.

[5] R. E. Hufnagel and N. R. Stanley, *Modulation Transfer Function Associated with Image Transmission Through Turbulent Media,* J. Opt. Soc. Amer. **54** (1964), 52–61.

[6] I. Orović, V. Papić, C. Ioana, X. Li, and S. Stanković, *Compressive Sensing in Signal Processing: Algorithms and Transform Domain Formulations,* Mathematical Problems in Engineering **2016** (2016), 1–16.

[7] I. Stanković, *Recovery of Images with Missing Pixels using a Gradient Compressive Sensing Algorithm,* (2014), https://arxiv.org/abs/1407.3695.

[8] J. S. Walker, *Fast Fourier Transforms*, CRC Press, 1996.