MASTER'S DEGREE THESIS

MSc in Applied Mathematics

Title: Mathematical Methods of Signal Processing

Author: Narcís Sayols Baixeras

Advisor: Sebastià Xambó Descamps
Co-Advisor: Ferran Vallverdú Bayés

Department: Departament de matemàtica aplicada II

Academic year: 2011-2012

Facultat de Matemàtiques i Estadística

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Universitat Politècnica de Catalunya
Facultat de Matemàtiques i Estadística

Master's Degree Thesis

Mathematical Methods of Signal Processing

Narcís Sayols Baixeras

Advisor: Sebastià Xambó Descamps
Co-Advisor: Ferran Vallverdú Bayés

Departament de matemàtica aplicada II

Abstract

Key words: Information And Communication, Signal processing, information, Pattern theory, Wavelet analysis

The aim of this project is to present in a systematic way the more relevant mathematical methods of signal processing, and to explore how they are applied to speech and image precessing. After explaining the more common parts of a standard course in signal processing, we put special emphasis in two new tools that have played a significant role in signal processing in the past few years: pattern theory and wavelet theory. Finally, we use all these techniques to implement an algorithm that detects the wallpaper group of a plane mosaic taking an image of it as input and an algorithm that returns the phoneme sequence of a speech signal.

The material in this memory can be grouped in two parts. The first part, consisting of the first six chapters, deals with the theoretical foundation of signal processing. It also includes materials related to plane symmetry groups. The second part, consisting of the last two chapters, is focussed on the applications.

Contents

Introduction

Signal Processing is an area in engineering and applied mathematics that deals with operations on or analysis of signals to extract or modify information. Signals may refer to images, sounds, sensor data, control system signals, telecommunication transmission systems such as radio or television signals. The importance of signal processing stems from the fact that signals usually are distorted by noise, or the information inside them is normally not directly reachable.

Although most natural signals are continuous-time functions, which are called analog signals, modern techniques in signal processing convert these signals from an analog to a digital form due to the advantages of digital hardware to their analog counterparts. These advantages are better performance, high speed, efficiency and error detection and correction in transmission as well as data compression.

In the last few decades the field of the signal processing has grown exponentially due to the advances in digital computer technology and software development and telecommunications. These fast developments in hardware and software technology have resulted in the construction of very powerful, faster, cheaper and smaller digital circuits. This digital hardware is able to perform complex operations in almost no time and is able to perform tasks that cannot be implemented in the analog world.

Another backbone of the present work is pattern theory. Pattern theory tries to describe the behaviour of the real world by detecting regularities that can be affected by random variations. Actually, a pattern is, roughly speaking, a regularity with some probability laws for its possible variations. At first, these models were only applied in signal processing to control the variability of the signals and to characterise the noise by Gaussian models. Nowadays, the usage of pattern theory has been extended to many more areas, and in particular to speech and image recognition. Applications like speech recognisers, a real time translator, or detectors of the patterns that characterise diseases and injuries, are among the applications that many people are working on.

This theory has been studied in the present thesis as a first step in a research plan for the near future. More precisely, we would like to investigate general methods of pattern processing, and in particular, image grammars that can support the description of images into parts and the relations between them.

The aim of this project is to present in a systematic way the more relevant mathematical methods of signal processing, and to explore how they are applied to speech and image precessing. After explaining the more common parts of a normal course in signal processing, we put special emphasis in two new tools that have played a significant role in signal processing in the past few years. These tools are pattern and wavelet theories. Finally, we use all these techniques to implement an algorithm that detects the wallpaper group of a plane mosaic taking an image of it as input and an algorithm that returns the phoneme sequence of a speech signal.

The material in this memory can be grouped in two parts. The first part, consisting of the first six chapters, deals with the theoretical foundation of signal processing. It also includes materials related to plane symmetry groups. The second part, consisting of the last two chapters, is focussed on the applications. In a bit more detail, the contents by chapter are as follows:

In Chapter 1, the focus is on the basics of signal processing: discrete-time signals and systems, linear and invariant time system analysis, correlations, the z-transform and its usage in the analysis of this type of systems.

Chapter 2 is devoted to the Fourier techniques. The different types of Fourier transforms are presented and applied to the analysis of the frequency domain of signals and systems. In the last part, we study the Fast Fourier Transform (FFT).

The preliminaries of digital filters, sampling and signal reconstruction are explained in Chapter 3.

The aim of Chapter 4 is the study of wavelet functions, their proprieties and their uses to compress signals. The wavelet transformations, as a tool for signal and system analysis, are also described.

In Chapter 5, some models of pattern theory are introduced, mainly those that will be used in speech and image processing. In the last part, a speech recogniser is described.

Chapter 6 presents the theory of wallpaper groups needed in the next chapter. We make a description of every wallpaper groups and we summarize the properties that single out one group from the others.

In the last two chapters we explore some applications of the preceding theories. The problems we consider, wich belong to the image processing domain (Chapter 7) and the speech processing domain (Chapter 8), are in particular meant to illustrate the remarkable fact that the same theoretical ideas and results can be applied in both domains. In more detail, in Chapter 7 we present an algorithm, and an implementation in MATLAB, that returns, given an image of a wallpaper pattern, the wallpaper group of that pattern. The chapter ends with some examples to illustrate how the algorithm and the implementation work. Finally, in Chapter 8 we present an application (algorithm and implementation in MATLAB) that parses the phonemes of a given recorded voice. The main theoretical tool used is the wavelet transform.

Chapter 1
Discrete-time signals and systems

In this chapter we will present the backbone of signal processing, which is provided by signals and systems, with emphasis in the characterization of linear and time invariant discrete-time systems. For this purpose, we will define and use convolution and z-transform tools [**17**].

1.1. Discrete-time signals

Definition. A *signal* is a function $x = x(t)$ of an independent variable which represents the time value. If the independent variable is continuous, x is called a *continuous-time signal*.

The next examples show some important continuous-time signals in signal processing.

Example.

(1) The *Dirac delta "function"*, $\delta(t)$, is defined by

$$\delta(t) = \begin{cases} +\infty, & t = 0 \\ 0, & t \neq 0 \end{cases}$$

Another definition is the only "function" which satisfies

$$\int_{\mathbb{R}} f(t)\delta(t)dt = f(0) \quad \forall f$$

In mathematical terms is not a function because $\delta(t) \notin L^2$. However, $\delta(t)$ is the most important continuous-time signal because it operates as a neutral element in the convolution operation and is also used to determinate the response of a continuous-time system.

(2) The exponential function which can be represented by $x(t) = Ae^{ift}$ where A is the amplitude, f is the frequency of the signal and i represents the imaginary unit. This type of functions are used in the frequency-domain analysis.

Definition. A *discrete-time signal* is a function $x(n)$ of an independent variable n which takes only integer values. It is important to note that this function is not defined between two integer values of the independent variable. For a given n, the value $x(n)$ is often called *nth sample* of the signal x.

In the next examples, we show some discrete-time signals which have a lot of importance in digital signal processing.

Example.

(1) The *unit sample sequence*, $\delta(n)$, is defined as

$$\delta(n) = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases}$$

This is the sequence analogue of the Dirac delta function in the discrete-time analysis.

(2) The *unit sample sequence*, $u(n)$, is the sequence defined as

$$u(n) = \begin{cases} 0, & n < 0 \\ 1, & n \geq 0 \end{cases}$$

(3) The *exponential signal* is a sequence of the form $x(n) = a^n$. In Fourier analysis the values of $a = e^{ik}$ are very important, where i is the imaginary unit and k is integer.

1.2. Discrete-time systems

Definition. A *system* is a device or algorithm that transforms a given signal to another one. From a mathematical point of view a system can be modelled as an operator that transforms signals into signals, i.e, $x(t) \xrightarrow{\mathcal{T}} y(t)$ where $x(t)$ is the *input signal* or *excitation* and $y(t)$ the *output signal*. In a picture, we will represent a system as a box.

The difference between a continuous-time system and a discrete-time system is that the first one takes continuous-time signals and transforms them to other continuous-time signals and the second one makes a transformation between discrete-time signals.

At this point, we will give some examples and definitions of systems in a discrete-time system version. In the continuous-time version, they would be the same changing:

- The integer independent variable by a continuous one.
- The summation by a integral among the real numbers.
- The $\delta(n)$ by the $\delta(t)$.

Example.

(1) A system is known as an *identity system* if the output signal is exactly the same as the input for any input signal.

(2) A *k-shifted system*, z^{-k}, is a system that moves the signal k values to the left in the time axis. In other words, $z^{-k}\{x(n)\} = x(n-k)$ for any input signal x.

(3) Another example could be a system whose output is the square of the input, $y(n) = x(n)^2$.

Definition. A system $\mathcal{T}$ is a *time invariant* or *shift invariant system* if and only if

$$x(n) \xrightarrow{\mathcal{T}} y(n) \Rightarrow x(n-k) \xrightarrow{\mathcal{T}} y(n-k)$$

for every input signal $x(n)$ and every shift k. In other words, a system is invariant if and only if commutes with every shift system.

Definition. A system $\mathcal{T}$ is a *linear* if and only if

$$\mathcal{T}\{a_1x_1(n) + a_2x_2(n)\} = a_1\mathcal{T}\{x_1(n)\} + a_2\mathcal{T}\{x_2(n)\}$$

for any input signals $x_1(n)$, $x_2(n)$ and any arbitrary constants a_1 and a_2.

Definition. A system is said to be *causal* if the output $y(n)$ of the system depends only on present and past input values, i.e., $y(n) = F[x(n),\ x(n-1),\ x(n-2), ...]$ for an arbitrary function F.

If a system depends of by a future value of the inputs, it is called *uncausal*. Obviously, in the real world all the systems must be causal because a system cannot depend on future input values.

Definition. A system $\mathcal{T}$ is called *stable* if and only if every bounded input produces a bounded output. In mathematical terms,

$$\text{If } |x(n)| \leq M_x < +\infty\ \forall n \Rightarrow |\mathcal{T}\{x(n)\}| \leq M_{\mathcal{T}x} < +\infty\ \ \forall n$$

for every input signal.

If some bounded input signal produces an unbounded output the system is called *unstable*. Stability is a propriety that must be considered in any practical application because unstable systems has an extreme behaviour and causes overflows in any practical implementations.

Definition. A system is said to be *invertible* if there is a one-to-one correspondence between its inputs and outputs. In other words, a system $\mathcal{T}$ is invertible if there exists another system $\mathcal{T}^{-1}$ called *inverse system* such that $\mathcal{T}^{-1}\{\mathcal{T}\{x(n)\}\} = \mathcal{T}\{\mathcal{T}^{-1}\{x(n)\}\} = x(n)$ for any input signal $x(n)$.

1.3. Convolution and its proprieties

Definition. The *convolution* of the functions f and g is the new function $f*g$ defined by

$$(f*g)(t) = \int_{-\infty}^{+\infty} f(\tau)g(t-\tau)d\tau$$

In the discrete version the definition is

$$(f*g)(n) = \sum_{k=-\infty}^{+\infty} f(k)g(n-k)$$

In the next proprieties and definitions we will use the discrete-time notation. The proprieties are also satisfied in the continuous version and their proof is completely analog.

Properties. The convolution operation.

(1) $f*g = g*f \ \forall f, g$
(2) $f*(g*h) = (f*g)*h \ \forall f, g, h$
(3) $f*(g+h) = (f*g) + (f*h) \ \forall f, g, h$
(4) $(f*\delta)(n) = f(n)$

1.4. Linear and time invariant discrete-time system (LTI) analysis

There are two methods for analysing the response of a linear system to a given input signal. The first one describes the response of the system using the equation of the system which in general has the form

$$y(n) = F[y(n-1), y(n-2), ..., y(n-N), x(n), x(n-1), ..., x(n-M)]$$

The second method is to discompose the input signal in elementary signals and using the linear propriety of the system to calculate the global response of the input signal.

So, we have an input signal $x(n) = \sum_{k=-\infty}^{+\infty} c_k x_k(n)$ And we will take the next linear bases

$$x_k(n) = \begin{cases} 1, & n = k \\ 0, & n \neq k \end{cases}$$

In this case, $x_k(n) \equiv \delta(n-k)$ and c_k are the coefficients of this linear decomposition. So, $c_k \equiv x(k)$. Since we are describing the behaviour of a linear and time invariant system we only have to calculate the response $h(n) = \mathcal{T}\{\delta(n)\}$, called the *impulse response*, of the basic sequence $\delta(n)$

$$y(n) = \mathcal{T}\{x(n)\} = \mathcal{T}\{\sum_{k=-\infty}^{+\infty} x(k)\delta(n-k)\} = \sum_{k=-\infty}^{+\infty} x(k)\mathcal{T}\{\delta(n-k)\} = \sum_{k=-\infty}^{+\infty} x(k)h(n-k) = x(n)*h(n)$$

Observation. To avoid convergence problems in the previous equivalences, we only consider the signals and systems that verify the previous equations.

Hence, we can describe the proprieties of a LTI system applying them to the impulse input signal.

Properties.

(1) A LTI system is causal if and only if the impulse response is zero for negative values of n.
(2) A LTI system is stable if and only if its impulse response is absolute summable.

1.5. Correlation

Definition. A discrete-time signal has *finite energy* if and only if

$$\sum_{n=-\infty}^{+\infty} |x(n)|^2 = E_x < +\infty$$

In this case, E_x is called the *energy* of the signal.

Definition. A discrete-time signal has *finite power* if and only if

$$\lim_{N\to+\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} |x(n)|^2 = P_x < +\infty$$

In this case, P_x is called the *power* of the signal.

Definition. Let $x_1(n), x_2(n)$ be two finite energy signals. The *correlation* of these signals, $r_{x_1x_2}(l)$, is the signal

$$r_{x_1x_2}(n) = \sum_{l=-\infty}^{+\infty} x_1(l)x_2^*(l-n)$$

In the special case that $x_1(n) = x_2(n)$, we obtain the *autocorrelation* of $x_1(n)$.

Observation. The correlation gives some information about the similitude of two signals.

Properties. Let be x, x_1, x_2 finite energy signals and E, E_1, E_2 their energies.

(1) $|r_{x_1x_2}(l)|^2 \leq E_1E_2$
(2) $r_{x_1x_2}(l) = r^*_{x_2x_1}(-l)$. If both are real, $r_{x_1x_2}(n) = r_{x_2x_1}(-n)$
(3) $r_{xx}(0) = E$
(4) $|r_{xx}(l)| \leq E$
(5) $r_{xx}(n) = r^*_{xx}(-n)$. If the signal is real, $r_{xx}(n) = r_{xx}(-n)$
(6) Let be $y(n) = x(n)*h(n)$ where $h(n)$ is the impulse response of a LTI system. Then, $r_{yy} = r_{xx}*r_{hh}$.

1.6. The z-transform and its proprieties

Definition. The *z-transform of a discrete-time signal* $x(n)$, $Z\{x(n)\}$, is the power series

$$X(z) = Z\{x(n)\} = \sum_{n=-\infty}^{+\infty} x(n)z^{-n} \quad \text{where } z \text{ is a complex variable.}$$

Definition. Since the z-transform is a infinite complex power series it only converges for certain values of z. This region is called *region of convergence (ROC)*.

Observation. The ROC of a finite sequence is the whole complex plane except possibly the points $z = 0$ and/or $z = \infty$.

The z-transform and their ROC of the most common signals can be found in the table 1.

Example. In this example we present a signal without ROC and another whose ROC is the outside of a circle.

(1) Let $x(n) = 0.5^n$. Expanding $x(n)$ on the interval $(-\infty, \infty)$ it becomes

$$x(n) = \{..., 0.5^{-3}, 0.5^{-2}, 0.5^{-1}, 1, 0.5, 0.5^2, 0.5^3, ...\} = \{..., 2^3, 2^2, 2, 1, 0.5, 0.5^2, 0.5^3, ...\}$$

Signal $x(n)$	**Z-transform** $X(z)$	**ROC**
$\delta(n)$	1	all z
$\delta(n-n_0)$	z^{-n_0}	$z \neq 0$
$u(n)$	$\frac{1}{1-z^{-1}}$	$\|z\| > 1$
$e^{-\alpha n}u(n)$	$\frac{1}{1-e^{-\alpha}z^{-1}}$	$\|z\| > \|e^{-\alpha}\|$
$-u(-n-1)$	$\frac{1}{1-z^{-1}}$	$\|z\| < 1$
$nu(n)$	$\frac{z^{-1}}{(1-z^{-1})^2}$	$\|z\| > 1$
$n^2u(n)$	$\frac{z^{-1}(1+z^{-1})}{(1-z^{-1})^3}$	$\|z\| > 1$
$a^nu(n)$	$\frac{1}{1-az^{-1}}$	$\|z\| > \|a\|$
$\cos(\omega_0 n)u(n)$	$\frac{1-z^{-1}\cos(\omega_0)}{1-2z^{-1}\cos(\omega_0)+z^{-2}}$	$\|z\| > 1$
$\sin(\omega_0 n)u(n)$	$\frac{z^{-1}\sin(\omega_0)}{1-2z^{-1}\cos(\omega_0)+z^{-2}}$	$\|z\| > 1$
$a^n\cos(\omega_0 n)u(n)$	$\frac{1-az^{-1}\cos(\omega_0)}{1-2az^{-1}\cos(\omega_0)+a^2z^{-2}}$	$\|z\| > \|a\|$
$a^n\sin(\omega_0 n)u(n)$	$\frac{az^{-1}\sin(\omega_0)}{1-2az^{-1}\cos(\omega_0)+a^2z^{-2}}$	$\|z\| > \|a\|$

TABLE 1. Z-transforms of the most common signals

Looking at the sum

$$\sum_{n=-\infty}^{\infty} x(n)z^{-n} \to \infty$$

Therefore, there are no such values of z that satisfy this condition.

(2) Let $x(n) = 0.5^n u(n)$. In this case expanding $x(n)$ in the interval $(-\infty, \infty)$ it becomes

$$x(n) = \{..., 0, 0, 0, 1, 0.5, 0.5^2, 0.5^3, ...\}$$

Looking at the sum

$$\sum_{n=-\infty}^{\infty} x[n]z^{-n} = \sum_{n=0}^{\infty} 0.5^n z^{-n} = \sum_{n=0}^{\infty} \left(\frac{0.5}{z}\right)^n = \frac{1}{1-0.5z^{-1}}$$

The last equality arises from the infinite geometric series and the equality only holds if $\left|0.5z^{-1}\right| < 1$ which can be rewritten in terms of z as $|z| > 0.5$. Thus, the ROC is $|z| > 0.5$. In this case the ROC is the outside of the circle of radius 0.5.

Theorem 1. The ROC of an infinite complex power series satisfies

$$\overset{\circ}{ROC} = \{z \in \mathbb{C} \, | r_2 < |z| < r_1\}$$

where r_2 could be 0 and r_1 could be ∞.

Observation. To know if the boundary are in the *ROC* has to be proved every time.

Notation. In the *ROC* case, we take the convection, $\frac{1}{0} = \infty$ and $\frac{1}{\infty} = 0$.

Theorem 2. The inverse z-transform of a signal $X(z)$ is

$$x(n) = \frac{1}{2\pi i} \oint_C X(z) z^{n-1} dz$$

where C is any contour that encloses the origin.

Properties. Let be the sequence $x(n), x_1(n), x_2(n)$, their z-transform $X(z), X_1(z), X_2(z)$ and their ROCs $ROC_x = \{r_2 < |z| < r_1\}, ROC_{x_1}, ROC_{x_2}$.

(1) *Linearity*: $Z\{a_1 x_1(n) + a_2 x_2(n)\} = a_1 X_1(z) + a_2 X_2(z)$ for any arbitrary constants a_1, a_2. The ROC will be at least the intersection of ROC_{x_1} and ROC_{x_2}.
(2) *Time expansion*: Let be $x_k(n) = \begin{cases} x(r), & n = rk \\ 0, & n \neq rk \end{cases}$ Then, $X_k(z) = X(z^k)$ and the $ROC = ROC_x^{\frac{1}{k}}$
(3) *Time shifting*: $Z\{x(n-k)\} = z^{-k} X(z)$. All the ROC except $z = 0$ if $k > 0$ and $z = \infty$ if $k < 0$.
(4) *Scaling in the z domain*: $Z\{a^n x(n)\} = X(a^{-1} z)$ and the ROC is $\{|a| r_2 < |z| < |a| r_1\}$.
(5) Time reversal: $Z\{x(-n)\} = X(z^{-1})$ with ROC $\{\frac{1}{r_1} < |z| < \frac{1}{r_2}\}$
(6) *Complex conjugation*: $Z\{x^*(n)\} = X^*(z^*)$ with the same ROC.
(7) *Real part*: $Z\{\Re\{x(n)\}\} = \frac{1}{2}[X(z) + X^*(z^*)]$ with the same ROC.
(8) *Imaginary part*: $Z\{\Im\{x(n)\}\} = \frac{1}{2i}[X(z) - X^*(z^*)]$ with the same ROC.
(9) *Differentiation*: $Z\{nx(n)\} = -z\frac{dX}{dz}(z)$ with the same ROC.
(10) *Convolution*: $Z\{x_1(n) * x_2(n)\} = X_1(z) X_2(z)$ with ROC at least the intersection of the two ROCs.
(11) *Correlation*: $Z\{r_{x_1 x_2}(l)\} = R_{x_1 x_2}(z) = X_1(z) X_2^*(z^{*^{-1}})$ with ROC at least the intersection of ROC_{x_1} and the ROC of $X_2^*(z^{*^{-1}})$.

Proposition. Let be the sequence $x_1(n), x_2(n)$ and their z-transform $X_1(z), X_2(z)$.

(1) *Multiplication*:

$$Z\{x_1(n) x_2(n)\} = \frac{1}{2\pi i} \oint_C X_1(v) X_2(z v^{-1}) v^{-1} dv$$

where C is a closed contour that encloses the origin and lies within the region of convergence common to both $X_1(v)$ and $X_2(z v^{-1})$.
(2) *Parseval's relation*:

$$Z\{\sum_{n=-\infty}^{+\infty} x_1(n) x_2^*(n)\} = \frac{1}{2\pi i} \oint_C X_1(v) X_2^*(v^{*^{-1}}) v^{-1} dv$$

Theorem 3.

(1) Initial value theorem: If $x(n)$ is causal, then,

$$x(0) = \lim_{z \to \infty} X(z)$$

(2) Final value theorem: If $(z-1)X(z)$ has all the poles[1] inside the unit circle, then,

$$x(\infty) = \lim_{z\to 1}(z-1)X(z)$$

1.7. Poles ans zeros

Definition. The *zeros* of a complex function $X(z)$ are the values of z for which $X(z) = 0$.

Definition. The *poles* of a complex function $X(z)$ are the values of z for which $X(z) = +\infty$.

In the following paragraphs we only consider rational z-transforms. In practise the majority of the system can be modelled as a rational z-transform system.

If we consider a rational z-transform then it can be written in the following form

$$X(z) = \frac{A(z)}{B(z)} = \frac{\sum_{n=0}^{\infty} a_n z^{-n}}{\sum_{n=0}^{\infty} b_n z^{-n}} = Gz^{N-M}\frac{\prod_{k=1}^{M}(z-z_k)}{\prod_{k=1}^{N}(z-p_k)}$$

where M and N is the number of zeros and poles respectively and G is a constant determinate by the division of the constant terms of the polynomials $A(z)$ and $B(z)$, in other words $G = a_0/b_0$.

Observation. If we consider the poles and zeros of the function in $z = \infty$, we obtain that the number of zeros and poles are exactly the same.

1.8. Unilateral z-transform

Definition. The *unilateral z-transform of a signal* $x(n)$, $X^+(z)$, is the function defined by

$$X^+(z) = Z^+\{x(n)\} = \sum_{n=0}^{+\infty} x(n)z^{-n}$$

Observation. Note that, $X(z) = X^+(z)$ if and only if $x(n)$ is causal.

Properties. Let $x(n)$ be a signal and $X(z), X^+(z)$ its z-transform and unilateral z-transform respectively.

(1) $X^+(z) = X(z)U(z)$ where $U(z)$ is the z-transform of the unit sample sequence.
(2) The ROC of $X^+(z)$ is always the exterior of a circle.
(3) *Shifting propriety*: $Z^+\{x(n-k)\} = z^{-k}[X^+(z) + \sum_{n=1}^{k} x(-n)z^n]$ where $k > 0$.
(4) *Time advance*: $Z^+\{x(n+k)\} = z^k[X^+(z) - \sum_{n=0}^{k-1} x(n)z^{-n}]$ where $k > 0$.

[1] See 1.7 for the definition.

1.9. Z-domain LTI analysis

As it has been shown in 1.4 a LTI system can be described as the convolution of the input signal and the impulse response of the system, i.e., $y(n) = x(n)*h(n)$. So, using the property of the z-transform, one can describe the z-transform of the system as the multiplication of the input z-transform and the impulse response z-transform. In consequence,

$$\mathbb{Z}\{y(n)\} = \mathbb{Z}\{x(n)*h(n)\} = \mathbb{Z}\{x(n)\}Z\{h(n)\} = X(z)H(z)$$

In other words, it is sufficient to characterise the z-transform of the impulse response to determine the properties of the system in the z-domain. It is also possible calculate the output of an LTI system in response to an input with transform $X(z)$ by computing the product $Y(z) = H(z)X(z)$ and then determining the inverse z-transform of $Y(z)$ to obtain the output sequence $y(n)$.

Using this property, it can be possible determinate the causality and stability of a LTI system calculating the ROC of the system as it is shown in the next proposition.

Proposition.

(1) A LTI system is causal if and only if the ROC of the system is the exterior of a circle of radius $r < \infty$ including $z = \infty$.
(2) A LTI system is stable if and only if the ROC of the system includes the unit circle.

Proof.

(1) As defined previously. a causal linear time-invariant system is one whose impulse response $h(n)$ satisfies the condition $h(n) = 0$ for $n < 0$. We have also shown that the ROC of the z-transform of a causal sequence is the exterior of a circle. So, we obtain what we want.
(2) As we have shown a necessary and sufficienr condition for a LTI system to be stable is $|h(n)| < \infty$. In turn, this condition implies that $H(z)$ must contain the unit circle within its ROC. Indeed, since

$$H(z) = \sum_{n=-\infty}^{+\infty} h(n)z^{-n}$$

it follows that

$$|H(z)| \leq \sum_{n=-\infty}^{+\infty} |h(n)z^{-n}| = \sum_{n=-\infty}^{+\infty} |h(n)||z^{-n}|$$

When evaluated on the unit circle (i.e.. $|z| = 1$),

$$|H(z)| \leq \sum_{n=-\infty}^{+\infty} |h(n)|$$

Hence, if the system is stable, the unit circle is contained in the ROC of $H(z)$. The converse is also true.

□

Definition. A LTI system is called *marginally stable* if all its poles are in the unit circle.

Another usage to the z-transform of the impulse response is to cancel poles. A pole cancelation occurs when a z-transform has a pole that is at the same location as a zero, the pole is canceled by the zero and, consequently, the term containing that pole in the inverse z-transform vanishes. Such pole-zero cancellations are very important in the analysis of pole-zero systems.

Pole-zero cancellations can occur either in the system function itself or in the product of the system function with the z-transform of the input signal. In the first case we say that the order of the system is reduced by one. In the latter case we say that the pole of the system is suppressed by the zero in the input signal, or vice versa. Thus, by properly selecting the position of the zeros of the input signal, it is possible to suppress one or more pole factors in the response of the system. Similarly, by proper selection of the zeros of the system function, it is possible to suppress one or more poles of the input signal from the response of the system.

Observation. When the zero is located very near the pole but not exactly at the same location, the term in the response has a very small amplitude. For example, nonexact pole-zero cancellations can occur in practice as a result of insufficiant numerical precision used in representing the coefficients of the system. Consequently, one should not attempt to stabilize an inherently unstable system by placing a zero in the input signal at the location of the pole.

Chapter 2
Fourier techniques

The aim of this chapter is to present and use the Fourier techniques to analyse LTI systems by characterising them in the frequency domain. These techniques basically amount to a decomposition in terms of complex exponential components. In the last part of this chapter we present the Fast Fourier Transform algorithm, which provides the key to the efficient use of all these techniques in the digital signal processing world [**17**].

2.1. Fourier series of a periodic signal

Definition. A signal is *periodic* if $x(n+N) = x(n) \ \ \forall n$, where N is the period of the signal.

Definition. The *Fourier series of a periodic signal* is the representation of a periodic signal $x(n)$ of period N in terms of basic exponential components

$$x(n) = \sum_{k=0}^{N-1} c_k e^{i2\pi kn/N}$$

where $c_k = \frac{1}{N} \sum_{l=0}^{N-1} x(l) e^{-i2\pi kl/N}$

Observation. Since $s_k(n) = e^{i2\pi kn/N}$ has also period N. The coefficients of the series c_k are also periodic, $c_k = c_{k+N}$

Proposition. Parseval's relation: Let be $x(n)$ a periodic signal of period N. Then,

$$\frac{1}{N} \sum_{n=0}^{N-1} |x(n)|^2 = \sum_{k=0}^{N-1} |c_k|^2$$

where c_k are the components of the signal in the periodic composition.

Example. Let $x(n)$ be the periodic function defined as

$$x(n) = \begin{cases} 1, & \text{if } \lfloor \frac{n}{3} \rfloor \bmod 2 = 0 \\ 0, & \text{otherwise} \end{cases}$$

The period of the function is 6, so the Fourier coefficients of the series of $x(n)$ are

$$c_k = \frac{1}{6}\sum_{l=0}^{5} x(l)e^{-i2\pi kl/6} = \frac{1}{6}\sum_{l=0}^{2} e^{-i2\pi kl/6}$$

So,

$$x(n) = \frac{1}{6}\sum_{k=0}^{6}\left(\sum_{l=0}^{2} e^{-i2\pi kl/6}\right)e^{i2\pi kn/6}$$

Observation. In the continuous-time world the previous formulae have the following form:

- The Fourier series of a periodic signal of period T: $x(n) = \sum_{k=-\infty}^{+\infty} c_k e^{i2\pi kt/T}$
- The coefficients: $c_k = \frac{1}{T}\int_T x(t)e^{-i2\pi kt/T}dt$
- Parseval's relation: $\frac{1}{T}\int_T |x(t)|^2 dt = \sum_{k=-\infty}^{+\infty} |c_k|^2$

2.2. The Fourier transform and its proprieties

Definition. The *Fourier transform of a finite energy discrete-signal* $x(n)$, $\mathcal{F}\{x(n)\}$, is defined as

$$X(\omega) = \mathcal{F}\{x(n)\} = \sum_{n=-\infty}^{+\infty} x(n)e^{-i\omega n}$$

In other words, the Fourier transform is the decomposition of $x(n)$ into its frequency components.

Properties. The Fourier transform of a discrete-time signal $x(n)$ has period 2π, i.e., $X(\omega) = X(\omega + 2\pi)$.

In the table 1 there are the Fourier transform of the most common signals.

Proposition. The Fourier transform converges uniformly to $X(\omega)$

Definition. The *inverse Fourier transform of a finite energy discrete-time signal* $X(\omega)$, $\mathcal{F}^{-1}\{X(\omega)\}$, is defined as

$$x(n) = \mathcal{F}^{-1}\{X(\omega)\} = \frac{1}{2\pi}\int_{2\pi} X(\omega)e^{i\omega n}d\omega$$

Properties. Let $x(n)$, $x_1(n)$ and $x_2(n)$ be signals and $X(\omega)$, $X_1(\omega)$ and $X_2(\omega)$ their Fourier transforms.

(1) *Linearity*: $\mathcal{F}\{a_1x_1(n) + a_2x_2(n)\} = a_1X_1(\omega) + a_2X_2(\omega)$ for any arbitrary constants a_1, a_2.
(2) *Time shifting*: $\mathcal{F}\{x(n-k)\} = e^{-i\omega n}X(\omega)$.
(3) *Modulation*: For any real number ω_0, $\mathcal{F}\{e^{in\omega_0}x(n)\} = X(\omega - \omega_0)$.
(4) *Scaling*: For any nonzero real number a, $\mathcal{F}\{x(an)\} = \frac{1}{|a|}X(a^{-1}\omega)$.
(5) *Time reversal*: $\mathcal{F}\{x(-n)\} = X(-\omega)$.
(6) *Complex conjugation*: $\mathcal{F}\{x^*(n)\} = X^*(-\omega)$.
(7) *Differentiation*: $\mathcal{F}\{n^k x(n)\} = i^k\frac{d^kX}{d\omega^k}(\omega)$.

Signal $x(n)$	**Fourier transform** $X(\omega)$
$\delta(n)$	1
$\delta(n-N)$	$e^{-i\omega N}$
$\sum_{m=-\infty}^{\infty}\delta(n-Nm)$	$\sum_{m=-\infty}^{\infty} e^{-i\omega Nm} = \frac{1}{N}\sum_{k=-\infty}^{\infty}\delta\left(\frac{\omega}{2\pi}-\frac{k}{N}\right)$
$u(n)$	$\frac{1}{1-e^{-i\omega}} + \sum_{k=-\infty}^{\infty}\pi\delta(\omega-2\pi k)$
$a^n u(n)\ \ \lvert a<1\rvert$	$\frac{1}{1-ae^{-i\omega}}$
e^{-ian}	$2\pi\delta(\omega+a)$
$\cos(an)$	$\pi\left[\delta(\omega-a)+\delta(\omega+a)\right]$
$\sin(an)$	$\frac{\pi}{i}\left[\delta(\omega-a)-\delta(\omega+a)\right]$

TABLE 1. Fourier transforms of the most common signals

(8) *Convolution*: $\mathcal{F}\{x_1(n)*x_2(n)\} = X_1(\omega)X_2(\omega)$.
(9) *Multiplication*: $\mathcal{F}\{x_1(n)x_2(n)\} = X_1(\omega)*X_2(\omega)$.
(10) *Correlation*: $\mathcal{F}\{r_{x_1x_2}(l)\} = R_{x_1x_2}(\omega) = X_1(\omega)X_2^*(\omega)$.

Observation. The relation between the z-transform and the Fourier Transform is that $\mathcal{F}\{x(n)\} = Z\{x(n)\}|_{z=e^{i\omega}}$. So, if the *ROC* of its z-transform doesn't contains the unit circle, its Fourier transform won't exists.

Example. In this example we calculate the Fourier transform of the signal $x(n) = a^{|n|}$ where $|a| < 1$.

First, we observe that $x(n)$ can be expressed as $x(n) = x_1(n) + x_2(n)$ where

$$x_1(n) = \begin{cases} a^n, & \text{if } n \geq 0 \\ 0, & \text{if } n < 0 \end{cases}$$

and

$$x_2(n) = \begin{cases} a^{-n}, & \text{if } n < 0 \\ 0, & \text{if } n \geq 0 \end{cases}$$

Beginning with the definition of the Fourier transform we have

$$X_1(\omega) = \sum_{n=-\infty}^{+\infty} x_1(n)e^{-i\omega n} = \sum_{n=0}^{\infty} a^n e^{-i\omega n} = \sum_{n=0}^{\infty}(ae^{-i\omega})^n = \frac{1}{1-ae^{-i\omega}}$$

where the last summation converges because

$$|ae^{-i\omega}| = |a| < 1$$

Similarly, the Fourier transform of $x_2(n)$ is

$$X_2(\omega) = \sum_{n=-\infty}^{+\infty} x_2(n)e^{-i\omega n} = \sum_{n=-\infty}^{-1} a^{-n}e^{-i\omega n} = \sum_{n=-\infty}^{-1} (ae^{i\omega})^{-n} = \sum_{k=1}^{\infty} (ae^{i\omega})^{k} = \frac{ae^{i\omega}}{1-ae^{i\omega}}$$

By combining these two transforms. we obtain the Fourier transform of $x(n)$ in the form

$$X(\omega) = X_1(\omega) + X_2(\omega) = \frac{1-a^2}{1-2a\cos(\omega)+a^2}$$

Observation. The continuous-time expressions:

- In the continuous-time version the expression is $X(f) = \int_{\mathbb{R}} x(t)e^{-i2\pi ft}dt$
- In this case, aside from the expression there is a substantial difference which is that the *spectrum* of continuous-time signals (the frequency domain) is $(-\infty, +\infty)$ while in the discrete-time is an interval of measure 2π, normally $(-\pi, \pi)$ or $(0, 2\pi)$.
- The inverse Fourier transform of a continuous-time signal is $x(t) = \int_{\mathbb{R}} X(f)e^{i2\pi ft}df$

2.3. Frequency analysis of LTI systems

As we have seen in 1.4, a LTI system can be determined by the impulse response $h(n) = \mathcal{T}\{\delta(n)\}$, namely $y(n) = \mathcal{T}\{x(n)\} = x(n)*h(n)$. So, using the convolution proprieties the frequency components of the output of a LTI system are the multiplication of the frequency components of the input signal and the impulse response $Y(\omega) = X(\omega)H(\omega)$.

If we express $Y(\omega), H(\omega)$ and $X(\omega)$ in polar form, the magnitude and phase of the output signal can be expressed as

$$|Y(\omega)| = |H(\omega)||X(\omega)|$$

$$\angle Y(\omega) = \angle H(\omega) + \angle X(\omega)$$

where $|H(\omega)|$ and $\angle H(\omega)$ are the magnitude and phase responses of the system.

By its very nature, a finite-energy aperiodic signal contains a continuum of frequency components. The linear time-invariant system, through its frequency response function, attenuates some frequency components of the input signal and amplifies other frequency components. Thus the system acts as a filter to the input signal. Observation of the graph of $|H(\omega)|$ shows which frequency components are amplified and which are attenuated. On the other hand, the angle of $\angle H(w)$ determines the phase shift imparted in the continuum of frequency components of the input signal as a function of frequency. If the input signal spectrum is changed by the system in an undesirable way, we say that the system has caused magnitude and phase distortion.

We also observe that the output of a linear time-invariant system cannot contain frequency components that are not contained in the input signal. It takes either a linear time-variant system or a nonlinear system to create frequency components that are not necessarily contained in the input signal.

2.4. Frequency filters

Filters are usually classified according to their frequency-domain characteristics as *lowpass*, *high-pass*, *bandpass*, *bandstop* and *all-pass* filters. The ideal magnitude response characteristics of these types of filters are illustrated in Figure 1. As shown, these *ideal filters* have a constant-pain (usually taken as unity-gain) *passband* characteristic and zero gain in their *stopband*.

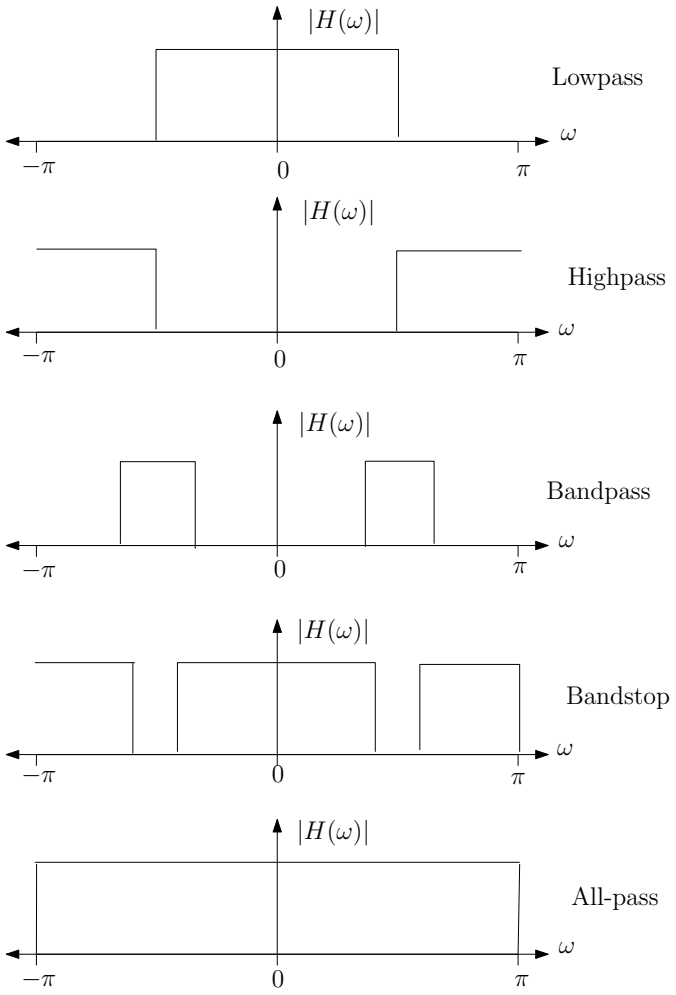


FIG. 1. Types of ideal filters.

Another characteristic of an ideal filter is a linear phase response. To demonstrate this point, let us assume that a signal sequence $x(n)$ with frequency components confined to the frequency range

$\omega_l < \omega < \omega_2$ is passed through a filter with frequency response

$$H(\omega) = \begin{cases} Ce^{-j\omega n_0} & \omega_l < \omega < \omega_2 \\ 0 & \text{otherwise} \end{cases}$$

where C and n_0 are constants. The signal at the output of the filter has a spectrum

$$Y(\omega) = H(\omega)X(\omega) = CX(\omega)e^{-j\omega n_0} \qquad \omega_1 < \omega < \omega_2$$

By applying the scaling and time-shifting properties of the Fourier transform, we obtain the time-domain output

$$Y(n) = Cx(n - n_0)$$

Consequently, the filter output is simply a delayed and amplitude-scaled version of the input signal. A pure delay is usually tolerable and is not considered a distortion of the signal. Neither is amplitude scaling. Therefore, ideal filters have a linear phase characteristic within their passband.

In conclusion, ideal filters have a constant magnitude characteristic and a linear phase characteristic within their passband. In all cases, such filters are not physically realizable but serve as a mathematical idealization of practical filters.

In the following discussion, we treat the design of some simple digital filters by the placement of poles and zeros in the z-plane.

The basic principle underlying the pole-zero placement method is to locate poles near points of the unit circle corresponding to frequencies to be emphasized and to place zeros near the frequencies to be deemphasized. Furthermore, the following constraints must be imposed:

(1) All poles should be placed inside the unit circle in order for the filter to be stable. However. zeros can be placed anywhere in the z-plane.
(2) All complex zeros and poles must occur in complex-conjugate pairs in order for the filter coefficients to be real.

From our previous discussion in 1.7 we recall that for a given pole-zero pattern the system function $H(z)$ can be expressed as

$$H(z) = b_0 \frac{\prod_{k=1}^{M}(1 - z_k z^{-1})}{\prod_{k=1}^{N}(1 - p_k z^{-1})}$$

where b_0 is a constant gain selected to normalize the frequency response at some specified frequency. That is, b_0 is selected such that

$$H(\omega_0) = 1$$

where ω_0 is a frequency in the passband of the filter. Usually, N is selected to equal or exceed M. So that the filter has more nontrivial poles than zeros.

For example, in the design of lowpass digital filters. the poles should be placed near the unit circle at points corresponding to low frequencies (near $\omega = 0$) and zeros should be placed near or on the unit circle at points corresponding to high frequencies (near $\omega = \pi$). The opposite holds true for highpass filters.

2.5. Frequency domain sampling

Let us consider an aperiodic discrete-time signal $x(n)$ with Fourier transform

$$X(\omega) = \sum_{n=-\infty}^{+\infty} x(n)e^{-iwn}$$

Suppose that we sample $X(\omega)$ periodically in frequency at a spacing of $\delta\omega$ radians between successive samples. Since $X(\omega)$ is periodic with period 2π, only samples in the fundamental frequency range are necessary. For convenience. we take N equidistant samples in the interval $[0, 2\pi]$ with spacing $\delta\omega = 2\pi/N$. First, we consider the selection of N, the number of samples in the frequency domain.

If we evaluate the previous equation at $\omega = 2\pi k/N$, we obtain

$$X\left(\frac{2\pi k}{N}\right) = \sum_{n=-\infty}^{+\infty} x(n)e^{-i2\pi kn/N} = \sum_{l=-\infty}^{+\infty} \sum_{n=lN}^{lN+N-1} x(n)e^{-i2\pi kn/N} =$$

$$\sum_{n=0}^{N-1} \left(\sum_{l=-\infty}^{+\infty} x(n - lN) \right) e^{-i2\pi kn/N} \qquad k = 0, ..., N-1 \quad (*)$$

The signal

$$\hat{x}(n) = \sum_{l=-\infty}^{+\infty} x(n - lN)$$

obtained by the periodic repetition of $x(n)$ every N samples is clearly periodic with fundamental period N. Consequently, it can be expanded in a Fourier series as

$$\hat{x}(n) = \sum_{k=0}^{N-1} c_k e^{i2\pi kn/N} \qquad n = 0, ..., N-1$$

with Fourier coefficients

$$c_k = \frac{1}{N} \sum_{n=0}^{N-1} \hat{x}(n)e^{-i2\pi kn/N} \qquad k = 0, ..., N-1$$

Upon comparing this equality with $(*)$ we conclude that

$$\hat{x}(n) = \frac{1}{N} \sum_{k=0}^{N-1} X\left(\frac{2\pi k}{N}\right) e^{i2\pi kn/N} \qquad n = 0, ..., N-1$$

This relationship provides the reconstruction of the periodic signal $\hat{x}(n)$ from the samples of the spectrum $X(\omega)$. However. it does not imply that we can recover $X(\omega)$ or $x(n)$ from the samples. To accomplish this. we need to consider the relationship between $\hat{x}(n)$ and $x(n)$.

Since $\hat{x}(n)$ is the periodic extension of $x(n)$, it is clear that $x(n)$ can be recovered from $\hat{x}(n)$ if there is no aliasing in the time domain, that is, if $x(n)$ is time-limited to less than the period N of $\hat{x}(n)$.

2.6. The discrete Fourier transform (DFT)

Definition. The *discrete Fourier transform (DFT)* of the signal $x(n)$, $\mathcal{DFT}\{x(n)\}$, is defined by

$$X(k) = \mathcal{DFT}\{x(n)\} = \sum_{n=0}^{N-1} x(n)e^{-i2\pi kn/N} \quad k = 0 \ldots N-1$$

An N-DFT point can be expressed as an $N \times N$ multiplication as $X = Wx$ where x is the original input signal and X is the DFT of the input signal. The transformation W can be defined as

$$W = \begin{pmatrix} 1 & 1 & 1 & 1 & \ldots & 1 \\ 1 & e^{-i2\pi/N} & e^{-i2\pi 2/N} & e^{-i2\pi 3/N} & \ldots & e^{-i2\pi(N-1)/N} \\ 1 & e^{-i2\pi 2/N} & e^{-i2\pi 4/N} & e^{-i2\pi 6/N} & \ldots & e^{-i2\pi 2(N-1)/N} \\ 1 & e^{-i2\pi 3/N} & e^{-i2\pi 6/N} & e^{-i2\pi 9/N} & \ldots & e^{-i2\pi 3(N-1)/N} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & e^{-i2\pi(N-1)/N} & e^{-i2\pi 2(N-1)/N} & e^{-i2\pi 3(N-1)/N} & \ldots & e^{-i2\pi(N-1)(N-1)/N} \end{pmatrix}$$

Some Fast Fourier Transform algorithms utilize the symmetries of the matrix to reduce the time of multiplying a vector by this matrix.

Definition. The *inverse discrete Fourier transform (IDFT)* of the discrete transform signal $X(k)$ is the N-periodic signal $\mathcal{DFT}^{-1}\{X(k)\}$ defined by

$$\hat{x}(n) = \mathcal{DFT}^{-1}\{X(k)\} = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{i2\pi kn/N} \quad n = 0 \ldots N-1$$

Observation. If $x(n)$ is a finite sequence of length $L \leq N$, the sequence $\hat{x}(n)$ defined by $\hat{x}(n) = \mathcal{DFT}^{-1}\{\mathcal{DFT}\{x(n)\}\}$ is a N-periodic version of $x(n)$.

Observation. As we have seen in the previous section, the DFT can be seen as a sampling of the Fourier transform in $\omega = \frac{2\pi k}{N}$

Definition. The *circular convolution* of length N of two discrete-time signals $x_1(n)$ and $x_2(n)$ is the new discrete-time signal $x_1(n) \otimes x_2(n)$ defined by

$$x_1(n) \otimes x_2(n) = \sum_{k=0}^{N-1} x_1(k)x_2(n-k) \; for \; n = 0 \ldots N-1$$

and extended periodically with period N.

Notation. In the case that a signal is N-periodic we will write $x(n)_N$ to show this propriety. In this case, the previous definition can be written as

$$x_1(n) \otimes x_2(n) = \sum_{k=0}^{N-1} x_1(k)x_2(n-k)_N \; for \; n = 0 \ldots N-1$$

Properties. Let $x(n)$, $x_1(n)$ and $x_2(n)$ be signals and $X(k)$, $X_1(k)$ and $X_2(k)$ their DFTs.

(1) *Linearity*: $\mathcal{DFT}\{a_1x_1(n) + a_2x_2(n)\} = a_1X_1(k) + a_2X_2(k)$ for any arbitrary constants a_1, a_2.
(2) *Time shifting*: $\mathcal{DFT}\{x(n - n_0)_N\} = e^{-i2\pi kn_0/N}X(k)$.
(3) *Modulation*: For any integer number k_0, $\mathcal{DFT}\{e^{i2\pi nk_0/N}x(n)\} = X(k - k_0)_N$.
(4) *Time reversal*: $\mathcal{DFT}\{x(-n)_N\} = X(-k)_N$.
(5) *Complex conjugation*: $\mathcal{DFT}\{x^*(n)\} = X^*(-k)_N$.
(6) *Duality*: $\mathcal{DFT}\{X(n)\} = Nx(-k)_N$.
(7) *Convolution*: $\mathcal{DFT}\{x_1(n) \otimes x_2(n)\} = X_1(k)X_2(k)$.
(8) *Multiplication*: $\mathcal{DFT}\{x_1(n)x_2(n)\} = \frac{1}{N}X_1(k) \otimes X_2(k)$.

Example. In this example we calculate de DFT of a signal using the definition and sampling the Fourier transform. Let $x(n)$ the finite-duration signal of length L defined as

$$x(n) = \begin{cases} 1, & 0 \leq n \leq L-1 \\ 0, & \text{otherwise} \end{cases}$$

We caclulate the N-point DFT of this signal in two ways: using the definition and sampling the Fourier transform. We take $N > L$.

(1) *Using the definition:* $X(k) = \sum_{n=0}^{N-1} x(n)e^{-i2\pi kn/N} = \sum_{n=0}^{L-1} e^{-i2\pi kn/N} = \frac{1-e-i2\pi kL/N}{1-e^{-i2\pi k/N}} = \frac{\sin(\pi kL/N)}{\sin(\pi k/N)}e^{-i\pi k(L-1)/N}$
(2) *Sampling* $X(\omega)$*:* $X(\omega) = \sum_{n=-\infty}^{+\infty} x(N)e^{-i\omega n} = \sum_{n=0}^{L-1} e^{-i\omega n} = \frac{1-e-i\omega L}{1-e^{-i\omega}} = \frac{\sin(\omega L/2)}{\sin(\omega/2)}e^{-i\omega(L-1)/2}$ The N-point DFT of $x(n)$ is simply $X(\omega)$ evaluated at the set of N equally spaced frequencies $\omega_k = 2\pi k/N$, $k = 0, 1, ..., N-1$. So,
$X(k) = \frac{\sin(\pi kL/N)}{\sin(\pi k/N)}e^{-i\pi k(L-1)/N}$

2.7. Linear filtering methods based on DFT

As we have shown in 2.3, the output sequence of a system can be determined via the inverse frequency transform of $Y(\omega) = H(\omega)X(\omega)$. However, the problem with the frequency domain approach is that $X(\omega)$, $H(\omega)$ and $Y(\omega)$ are functions of the continuous variable ω. In consequence, the computation cannot be done on a digital computer. On the other hand, the DFT does lend itself to computation on a digital computer. In the discussion that follows, we describe how the DFT can be used to perform linear filtering in the frequency domain. We now describe two methods for linear FIR filtering a long sequence on a block-by-block basis using the DFT.

2.7.1. Overlap-save method. Overlap-save is the traditional name for an efficient way to evaluate the discrete convolution between a very long signal $x(n)$ and a finite impulse response filter $h(n)$:

$$y(n) = x(n) * h(n) = \sum_{m=-\infty}^{\infty} h(m)x(n-m) = \sum_{m=0}^{M-1} h(m)x(n-m)$$

where $h(m) = 0$ for m outside the region $[0, M-1]$.

The concept is to compute short segments of $y(n)$ of an arbitrary length L, and concatenate the segments together. Consider a segment that begins at $n = kL + M$, for any integer k, and define:

$$x_k(n) = \begin{cases} x(n+kL) & 1 \le n \le L+M-1 \\ 0 & \text{otherwise.} \end{cases}$$

$$y_k(n) = x_k(n) * h(n)$$

Then, for $kL + M \le n \le kL + L + M - 1$, and equivalently $M \le n - kL \le L + M - 1$, we can write:

$$y(n) = \sum_{m=0}^{M-1} h(m)x_k(n - kL - m) = x_k(n - kL)*h(n) = y_k(n - kL)$$

The task is thereby reduced to computing $y_k(n)$, for $M \le n \le L + M - 1$.

Now note that if we periodically extend $x_k(n)$ with period $N \ge L + M - 1$ according to:

$$x_{k,N}(n) = \sum_{k=-\infty}^{\infty} x_k(n - kN)$$

the convolutions $x_{k,N}*h$ and x_k*h are equivalent in the region $M \le n \le L+M-1$. So it is sufficient to compute the N-point circular (or cyclic) convolution of $x_k(n)$ with $h(n)$ in the region $[1, N]$. The subregion $[M, L + M - 1]$ is appended to the output stream, and the other values are discarded.

The advantage is that the circular convolution can be computed very efficiently according to the circular convolution theorem using a FFT algorithm.

2.7.2. Overlap-add method. The overlap-add method is an efficient way to evaluate the discrete convolution of a very long signal x(n) with a finite impulse response filter h(n). As in the previous mehtod:

$$y(n) = x(n)*h(n) = \sum_{m=1}^{M} h(m)x(n - m)$$

where h(m)=0 for m outside the region $[0, M - 1]$.

The concept is to divide the problem into multiple convolutions of $h(n)$ with short segments of $x(n)$:

$$x_k(n) = \begin{cases} x(n+kL) & n = 1, 2, \ldots, L \\ 0 & \text{otherwise,} \end{cases}$$

where L is an arbitrary segment length. Then:

$$x(n) = \sum_k x_k(n - kL)$$

and y(n) can be written as a sum of short convolutions:

$$y(n) = \left(\sum_k x_k(n - kL) \right) * h(n) = \sum_k (x_k(n - kL) * h(n)) = \sum_k y_k(n - kL)$$

where $y_k(n) = x_k(n) * h(n)$, is zero outside the region $[1, L + M - 1]$. And for any parameter $N \geq L + M - 1$, it is equivalent to the N-point circular convolution of $x_k(n)$ with $h(n)$ in the region $[1, N]$.

The advantage is that the circular convolution can be computed very efficiently according to the circular convolution theorem using the FFT algorithm.

In the figure 2 the algorithm is shown.

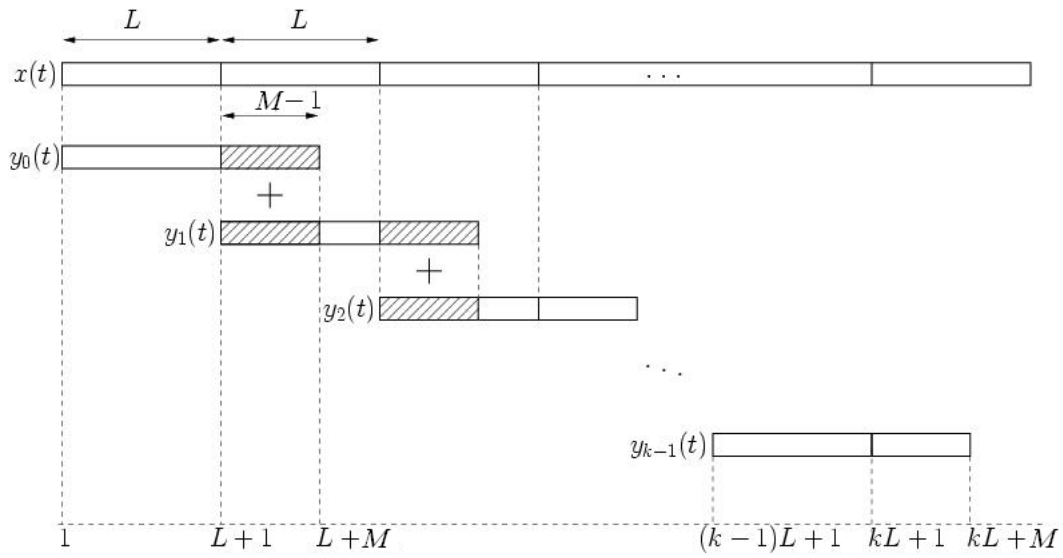


FIG. 2. A diagram of the overlap-add method.

2.8. Frequency analysis using the DFT

To compute the spectrum of either a continuous-time or discrete-time signal, the values of the signal for all time are required. However, in practice, we observe signals for only a finite duration. Consequently, the spectrum of a signal can only be approximated from a finite data record. In this section we examine the implications of just having a finite data record in frequency analysis using the DFT.

Let $x(n)$ denote the sequence to be analyzed. Limiting the duration of the sequence to L samples, in the interval $0 \leq n \leq L - 1$, is equivalent to multiplying $x(n)$ by a rectangular window $w(n)$ of length L. That is,

$$\tilde{x}(n) = x(n)w(n)$$

where

$$w(n) = \begin{cases} 1 & \text{if } 0 \le n \le L-1 \\ 0 & \text{otherwise} \end{cases}$$

Now suppose that the sequence $x(n)$ consists of a single sinusoid, that is,

$$x(n) = cos(\omega_0 n)$$

Then the Fourier transform of the finite-duration sequence $x(n)$ can be expressed as

$$\tilde{X}(\omega) = \frac{1}{2}\left(W(\omega - \omega_0) + W(\omega + \omega_0)\right)$$

where $W(\omega)$ is the Fourier transform of the window sequence, which is (for the rectangular window)

$$W(\omega) = \frac{sin(\omega L/2)}{sin(\omega/2)} e^{-i\omega(L-1)/2}$$

We note that the windowed spectrum $\tilde{X}(\omega)$ is not localized to a single frequency, but instead it is spread out over the whole frequency range. Thus the power of the original signal sequence $x(n)$ that was concentrated at a single frequency has been spread by the window into the entire frequency range. We say that the power has *leaked out* into the entire frequency range. Consequently, this phenomenon, which is a characteristic of windowing the signal, is called *leakage*.

Windowing not only distorts the spectral estimate due to the leakage effects, it also reduces spectral resolution. To illustrate this problem, let us consider a signal sequence consisting of two frequency components,

$$x(n) = cos(\omega_l n) + cos(\omega_2 n)$$

When this sequence is truncated to L samples in the range $0 \le n \le L-1$, the windowed spectrum is

$$\tilde{X}(\omega) = \frac{1}{2}\left(W(\omega - \omega_1) + W(\omega - \omega_2) + W(\omega + \omega_1) + W(\omega + \omega_2)\right)$$

The spectrum $W(\omega)$ of the rectangular window sequence has its first zero crossing at $\omega = 2\pi/L$. Now if $|\omega_1 - \omega_2| < 2\pi/L$, the two window functions $W(\omega - \omega_l)$ and $W(\omega - \omega_2)$ overlap and, as a consequence, the two spectral lines in $x(n)$ are not distinguishable. Thus our ability to resolve spectral lines of different frequencies is limited by the window main lobe width.

To reduce leakage, we can select a data window $w(n)$ that has lower sidelobes in the frequency domain compared with the rectangular window. However, a reduction of the sidelobes in a window $W(\omega)$ is obtained at the expense of an increase in the width of the main lobe of $W(\omega)$ and hence a loss in resolution.

2.9. Fast Fourier transform (FFT)

Definition. A *fast Fourier transform* (FFT) is an efficient algorithm to compute the DFT and its inverse.

An FFT computes the DFT and produces exactly the same result as evaluating the DFT definition directly. The only difference is that an FFT is much faster. Evaluating the DFT definition directly requires $O(N^2)$ operations: there are N outputs $X(k)$, and each output requires a sum of N terms. An FFT is any method to compute the same results in $O(N \log(N))$.

In this chapter we present two algorithms:

2.9.1. The Radix-2 decimation-in-time (DIT) Cooley-Tukey FFT Algorithm. Radix-2 DIT divides a DFT of size N into two interleaved DFTs (hence the name "radix-2") of size $N/2$ with each recursive stage.

Radix-2 DIT first computes the DFTs of the even-indexed inputs $x(2m)$ and of the odd-indexed inputs $x(2m + 1)$, and then combines those two results to produce the DFT of the whole sequence. This idea can then be performed recursively to reduce the overall runtime to $O(N \log N)$. This simplified form assumes that N is a power of two; since the number of sample points N can usually be chosen freely by the application, this is often not an important restriction.

The Radix-2 DIT algorithm rearranges the DFT of the function x_n into two parts: a sum over the even-numbered indices $n = 2m$ and a sum over the odd-numbered indices $n = 2m + 1$:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-i2\pi nk/N} = \sum_{m=0}^{N/2-1} x(2m) e^{-i2\pi 2mk/N} + \sum_{m=0}^{N/2-1} x(2m+1) e^{-i2\pi(2m+1)k/N}$$

One can factor a common multiplier $e^{-i2\pi k/N}$ out of the second sum, as shown in the equation below. It is then clear that the two sums are the DFT of the even-indexed part $x(2m)$ and the DFT of odd-indexed part $x(2m+1)$ of the signal $x(n)$. Denote the DFT of the Even-indexed inputs $x(2m)$ by $E(k)$ and the DFT of the Odd-indexed inputs $x(2m + 1)$ by $O(k)$ and we obtain:

$$X(k) = \underbrace{\sum_{m=0}^{N/2-1} x(2m) e^{-\frac{i2\pi mk}{N/2}}}_{\text{DFTofeven-indexedpartof}\,x(n)} + e^{-\frac{i2\pi k}{N}} \underbrace{\sum_{m=0}^{N/2-1} x(2m+1) e^{-\frac{i2\pi mk}{N/2}}}_{\text{DFTofodd-indexedpartof}\,x(n)} = E(k) + e^{-\frac{i2\pi k}{N}} O(k)$$

However, these smaller DFTs have a length of $N/2$, so we need compute only $N/2$ outputs: thanks to the periodicity properties of the DFT, the outputs for $N/2 \leq k < N$ from a DFT of length $N/2$ are identical to the outputs for $0 \leq k < N/2$. That is, $E(k + N/2) = E(k)$ and $O(k + N/2) = O(k)$. The phase factor $e^{-i2\pi k/N}$ obeys the relation: $e^{-i2\pi(k+N/2)/N} = e^{-i\pi} e^{-i2\pi k/N} = -e^{-i2\pi k/N}$, flipping the sign of the $O(k + N/2)$ terms. Thus, the whole DFT can be calculated as follows:

$$X(k) = \begin{cases} E(k) + e^{-i2\pi k/N} O(k) & \text{if } k < N/2 \\ E(k - N/2) - e^{-i2\pi(k-N/2)/N} O(k - N/2) & \text{if } k \geq N/2 \end{cases}$$

This result, expressing the DFT of length N recursively in terms of two DFTs of size $N/2$, is the core of the radix-2 DIT fast Fourier transform. The algorithm gains its speed by re-using the results of intermediate computations to compute multiple DFT outputs. Note that final outputs are obtained by a $+/-$ combination of $E(k)$ and $O(k)e^{-i2\pi k/N}$, which is simply a size-2 DFT.

2.9.2. Goertzel algorithm. The Goertzel algorithm exploits the periodicity of the phase factors $\{W_N^k = e^{-i2\pi k/N}\}$ and allows us to express the computation of the DFT as a linear filtering operation. Since $W_N^{-kN} = 1$ we can multiply the DFT by this factor. Thus

$$X(k) = W_N^{-kN} \sum_{m=0}^{N-1} x(m) W_N^k m = \sum_{m=0}^{N-1} x(m) W_N^{-k(N-m)}$$

We note that the previous equation is in the form of a convolution. Indeed, if we define the sequence $y_k(n)$ as

$$y_k(n) = \sum_{m=0}^{N-1} x(m) W_N^{-k(N-m)}$$

then it is clear that $y_k(n)$ is the convolution of the finite-duration input sequence $x(n)$ of length N with a filter that has an impulse response

$$h_k(n) = W_N^{-kn} u(n)$$

The output of this filter at $n = N$ yields the value of the DFT at the frequency $\omega_k = 2\pi k/N$. That is, $X(k) = y_k(n)|_{n=N}$

The filter with impulse response $h_k(n)$ has the system function

$$H_k(z) = \frac{1}{1 - W_N^{-k} z^{-1}}$$

This filter has a pole on the unit circle at the frequency $\omega_k = 2\pi k/N$. Thus, the entire DFT can be computed by passing the block of input data into a parallel bank of N single-pole filters (resonators) where each filter has a pole at the corresponding frequency of the DFT.

Instead of performing the computation of the DFT via convolution, we can use the difference equation corresponding to the filter given to compute $y_k(n)$ recursively. Thus we have

$$y_k(n) = W_N^{-k} y_k(n-1) + x(n) \qquad y(-1) = 0$$

The desired output is $X(k) = y_k(N)$, for $k = 0, 1, ..., N-1$. To perform this computation, we can compute once and store the phase factors W_N^{-k}.

The complex multiplications and additions inherent in the difference equation can be avoided by combining the pairs of resonators possessing complex-conjugate poles. This leads to two-pole filters with system functions of the form

$$H_k(z) = \frac{1 - W_N^{-k} z^{-1}}{1 - 2\cos(2\pi k/N) z^{-1} + z^{-2}}$$

The system shown in the figure 3 is a two-poles resonator described by the difference equations

$$v_k(n) = 2\cos(\frac{2\pi k}{N})v_k(n-1) - v_k(n-2) + x(n)$$

$$y_k(n) = v_k(n) - W_N^k v_k(n-1)$$

with initial conditions $v_k(-1) = v_k(-2) = 0$.

The recursive relation in the first equation is iterated for $n = 0, 1, ..., N$, but the second equation is computed only once at time $n = N$. Each iteration requires one real multiplication and two additions. Consequently, for a real input sequence $x(n)$, this algorithm requires $N+1$ real multiplications to yield not only $X(k)$ but also, due to symmetry, the value of $X(N-k)$.

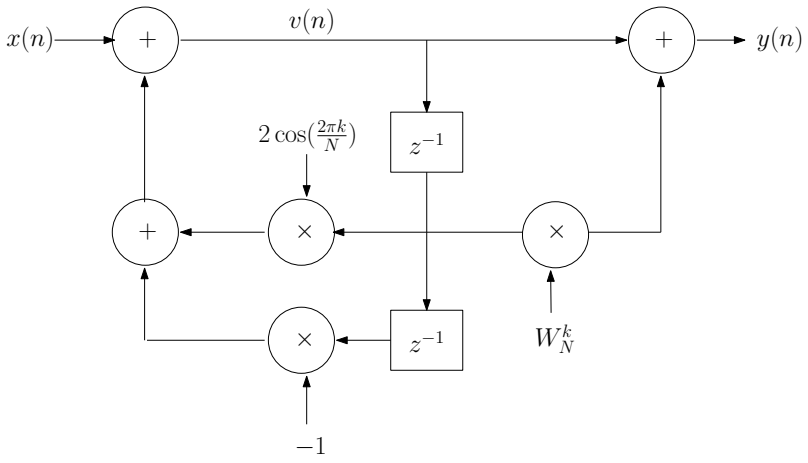


FIG. 3. Direct form II realiztion of two-pole resonator for computing the DFT.

Chapter 3
Digital filters and Sampling

This chapter has two distinct parts. The first part is concerned with digital filters and it includes general considerations to establish that an ideal filter[1] cannot be really implemented. The second part is describes the conversions analog-to-digital and digital-to-analog, which in fact enable to use digital tools to process analog signals [**17**].

3.1. General considerations

Theorem 4. Paley-Wiener Theorem: If $h(n)$ has a finite energy and $h(n) = 0$ for $n < 0$, then

$$\int_{2\pi} |\ln |H(\omega)|| d\omega < +\infty$$

Conversely, if $|H(\omega)|$ is square integrable and the previous integral is finite, then we can associate with $|H(\omega)|$ a phase respond $\Theta(\omega)$, so that the resulting filter with frequency response

$$H(\omega) = |H(\omega)|e^{i\Theta(\omega)}$$

is causal.

Corollary. In any causal filter, $|H(\omega)|$ can be zero at some frequency but it cannot be zero over any finite interval (band of frequencies).

Corollary. Any ideal filter is noncausal.

3.2. FIR filters

Definition. A *Finite impulse response (FIR) filter* of length M with input $x(n)$ and output $y(n)$ is described by the difference equation

$$y(n) = \sum_{k=0}^{M-1} b_k x(n-k)$$

[1] See 2.4 for definition

In terms of its impulse response,

$$H(z) = \sum_{k=0}^{M-1} b_k z^{-k}$$

where the lower and upper limits of the sum reflect the causality and finite response of the system.

Proposition. A FIR filter has linear phase if its impulse response satisfies the condition $h(n) = \pm h(M-1-n)$ for $n = 0 \ldots M-1$.

3.3. IIR filters

Definition. As digital filter are usually described in terms of a differential equation, an *infinite impulse response* filter is a system that can be expressed as

$$\sum_{j=0}^{M} a_j y(n-j) = \sum_{i=0}^{N} b_i x(n-i)$$

We first take the Z-transform of each side of the above equation, where we use the time-shift property to obtain:

$$\sum_{j=0}^{Q} a_j z^{-j} Y(z) = \sum_{i=0}^{P} b_i z^{-i} X(z)$$

Definition. We define the *transfer function* to be:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^{P} b_i z^{-i}}{\sum_{j=0}^{Q} a_j z^{-j}}$$

Considering that in most IIR filter designs coefficient a_0 is 1, the IIR filter transfer function takes the more traditional form:

$$H(z) = \frac{\sum_{i=0}^{P} b_i z^{-i}}{1 + \sum_{j=1}^{Q} a_j z^{-j}}$$

IIR are used in multiple type of different application. In the next observation we explain how a convolutional code works and to make the convolutional encoder an IIR filter is applied.

Observation. In telecommunication, a convolutional code is a type of error-correcting code in which

- each m-bit information symbol (each m-bit string) to be encoded is transformed into an n-bit symbol, where m/n is the code rate ($n \geq m$) and

- the transformation is a function of the last k information symbols, where k is the constraint length of the code.

To convolutionally encode data, start with k memory registers, each holding 1 input bit. Unless otherwise specified, all memory registers start with a value of 0. The encoder has n modulo-2 adders (a modulo-2 adder can be implemented with a single Boolean XOR gate, where the logic is: $0 + 0 = 0$, $0 + 1 = 1$, $1 + 0 = 1$, $1 + 1 = 0$), and n generator polynomials - one for each adder (see figure 1). An input bit m_1 is fed into the leftmost register. Using the generator polynomials and the existing values in the remaining registers, the encoder outputs n bits. Now bit shift all register values to the right (m_1 moves to m_0, m_0 moves to m_{-1}) and wait for the next input bit. If there are no remaining input bits, the encoder continues output until all registers have returned to the zero state.

The figure 1 is a rate $1/3$ (m/n) encoder with constraint length k of 3. Generator polynomials are $G1 = (1, 1, 1)$, $G2 = (0, 1, 1)$, and $G3 = (1, 0, 1)$. Therefore, output bits are calculated (modulo 2) as follows:

- $n_1 = m_1 + m_0 + m_{-1}$
- $n_2 = m_0 + m_{-1}$
- $n_3 = m_1 + m_{-1}$

A convolutional encoder is called so because it performs a convolution of the input stream with the encoder's impulse responses:

$$y^j(i) = \sum_{k=0}^{+\infty} h^j(k)x(i-k)$$

where x, is an input sequence, y^j, is a sequence from output j, and h^j, is an impulse response for output j.

A convolutional encoder is a discrete linear time-invariant system. Every output of an encoder can be described by its own transfer function, which is closely related to a generator polynomial. An impulse response is connected with a transfer function through Z-transform.

The transfer functions for the encoder of the figure 1 are:

- $H_1(z) = 1 + z^{-1} + z^{-2}$
- $H_2(z) = z^{-1} + z^{-2}$
- $H_3(z) = 1 + z^{-2}$

3.4. Frequency transformations

In the practise we only describe low-pass[2] filters, since it is very easy to convert a low-pass filter into a bandpass, bandstop or highpass filters[3] by frequency transformations.

[2]See 2.4 for definition

[3]See 2.4 for definition

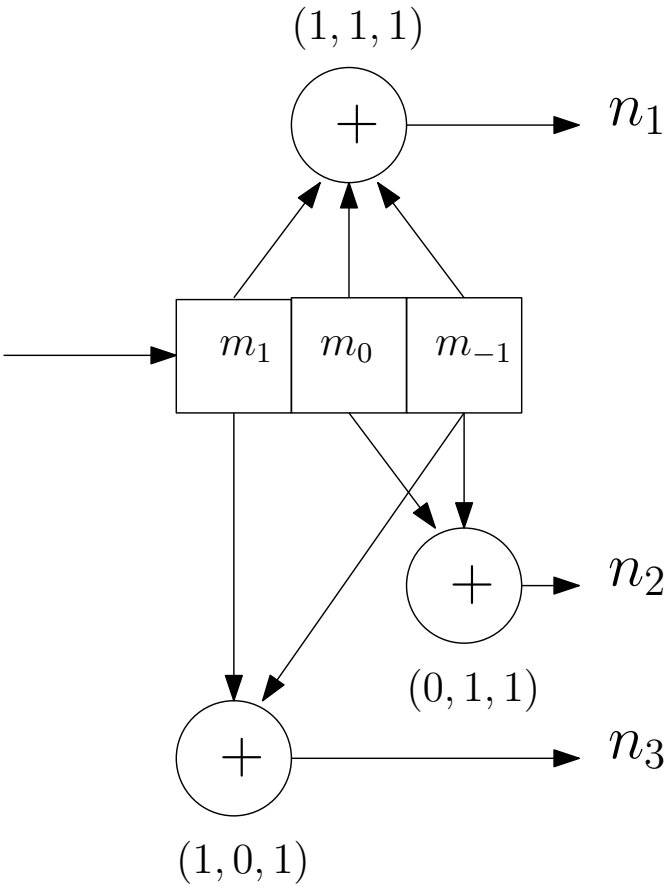


FIG. 1. Rate $1/3$ non-recursive, non-systematic convolutional encoder with constraint length 3

The transformations involve replacing the variable z^{-1} by a rational function $g(z^{-1})$ that must satisfy the following conditions:

(1) The mapping $z \to g(z^{-l})$ must map points inside the unit circle in the z-plane into itself.
(2) The unit circle must also be mapped into itself.

Condition (2) implies that for $r = 1$,

$$e^{-i\omega} = g(e^{-i\omega}) = |g(e^{-i\omega})|e^{-i \arg g(e^{-i\omega})}$$

So we must have $|g(\omega)| = 1$ for all ω. That is, the mapping must be all-pass[4]. Hence it is of the form

[4]See 2.4 for definition

Type of transformation	**Transformation**	**Parameters**
Lowpass	$z^{-1} \rightarrow \frac{z^{-1}-a}{1-az^{-1}}$	$a = \frac{\sin[(\omega_p-\omega'_p)/2]}{\sin[(\omega_p+\omega'_p)/2]}$ ω'_p is the band edge frequency of new filter
Highpass	$z^{-1} \rightarrow -\frac{z^{-1}+a}{1+az^{-1}}$	$a = -\frac{\cos[(\omega_p+\omega'_p)/2]}{\cos[(\omega_p-\omega'_p)/2]}$ ω'_p is the band edge frequency of new filter
Bandpass	$z^{-1} \rightarrow -\frac{z^{-2}-a_1z^{-1}+a_2}{1-a_1z^{-1}+a_2z^{-2}}$	$a_1 = -2\alpha K/(K+1)$ $a_2 = (K-1)/(K+1)$ $\alpha = \frac{\cos[(\omega_u+\omega_l)/2]}{\cos[(\omega_u-\omega_l)/2]}$ $K = \cot\frac{\omega_u-\omega_l}{2}\tan\frac{\omega_p}{2}$ ω_u is the upper band edge frequency ω_l is the lower band edge frequency
Bandstop	$z^{-1} \rightarrow -\frac{z^{-2}-a_1z^{-1}+a_2}{1-a_1z^{-1}+a_2z^{-2}}$	$a_1 = -2\alpha/(K+1)$ $a_2 = (1-K)/(K+1)$ $\alpha = \frac{\cos[(\omega_u+\omega_l)/2]}{\cos[(\omega_u-\omega_l)/2]}$ $K = \tan\frac{\omega_u-\omega_l}{2}\tan\frac{\omega_p}{2}$ ω_u is the upper band edge frequency ω_l is the lower band edge frequency

TABLE 1. Frequency transformation for digital filters (prototype lowpass filter has band edge frequency ω_p)

$$g(z^{-1}) = \pm\prod_{k=1}^{N}\frac{z^{-1}-a_k}{1-a_kz^{-k}}$$

where $|a_k| < 1$ to ensure that a stable filter is transformed into another stable filter to satisfy condition (1).

From the general form showed in the previous equations, we transform to any type of filter using the transformations of the table 1.

3.5. Digital filters using least squares

We now describe several methods for designing digital filters directly. In these techniques, the Padé approximation method and least-squares design methods, the specifications are given in the time domain and the design is carried out in the frequency domain.

3.5.1. Padé approximation method. Suppose that the desired impulse response $h_d(n)$ is specified for $n \geq 0$. The filter to be designed has the system function

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}} = \sum_{k=0}^{\infty} h(z) z^{-k}$$

where $h(k)$ is its unit sample response. The filter has $L = M + N + 1$ parameters, namely, the coefficients $\{a_k\}$ and $\{b_k\}$, which can be selected to minimize some error criterion.

The least-squares error criterion is often used in optimization problems of this type. Suppose that we minimize the sum of the squared errors

$$\varepsilon = \sum_{n=0}^{U} (h_d(n) - h(n))^2$$

with respect to the filter parameters $\{a_k\}$ and $\{b_k\}$, where U is some preselected upper limit in the summation.

In general, $h(n)$ is a nonlinear function of the filter parameters and hence the minimization of ε involves the solution of a set of nonlinear equations. However, if we select the upper limit as $U = L - 1$, it is possible to match $h(n)$ perfectly to the desired response $h_d(n)$ for $0 \leq n \leq M + N$. This can be achieved in the following manner.

The difference equation for the desired filter is

$$y(n) = -a_1 y(n-1) - a_2 y(n-2) - ... - a_N y(n-N) + b_0 x(n) + b_1 x(n-1) + ... + b_M x(n-M)$$

Suppose that the input to the filter is a unit sample [i.e., $x(n) = \delta(n)$]. Then the response of the filter is $y(n) = h(n)$ and hence the previous equations becomes

$$h(n) = \begin{cases} -a_1 h(n-1) - a_2 h(n-2) - ... - a_N h(n-N) + b_n, & 0 \leq n \leq M \\ -a_1 h(n-1) - a_2 h(n-2) - ... - a_N h(n-N), & n > M \end{cases}$$

since $\delta(n-k)$ is equal 0 except for $n = k$.

The set of previous linear equations can be used to solve for the filter parameters $\{a_k\}$ and $\{b_k\}$. We set $h(n) = h_d(n)$ for $0 \leq n \leq M + N$, and use the second linear equations to solve for the filter parameters $\{a_k\}$. Then we use values for the $\{a_k\}$ in first equation and solve for the parameters $\{b_k\}$. Thus we obtain a perfect match between $h(n)$ and the desired response $h_d(n)$ for the first L values of the impulse response. This design technique is usually called the *Padé approximation procedure*.

The degree to which this design technique produces acceptable filter designs depends in part on the number of filter coefficients selected. Since the design method matches $h_d(n)$ only up to the number of filter parameters, the more complex the filter, the better the approximation to $h_d(n)$ for $0 \le n \le M + N$. However, this is also the major limitation with the Padé approximation method, namely, the resulting filter must contain a large number of poles and zeros. For this reason, the Padé approximation method has found limited use in filter designs for practical applications.

Observation. An effective approach in using the Padé approximation is to try different values of M and N until the frequency responses of the resulting filters converge to the desired frequency response within some small, acceptable approximation error. However, in practice, this approach appears to be cumbersome.

3.5.2. Least square design methods. The least-squares method can be used in a pole-zero approximation for $H_d(z)$. If the filter $H(z)$ that approximates $H_d(z)$ has both poles and zeros, its response to the unit impulse $\delta(n)$ is again

$$h(n) = \begin{cases} -a_1 h(n-1) - a_2 h(n-2) - ... - a_N h(n-N) + b_n, & 0 \le n \le M \\ -a_1 h(n-1) - a_2 h(n-2) - ... - a_N h(n-N), & n > M \end{cases}$$

Clearly, if $H_d(z)$ is a pole-zero filter, its response to $\delta(n)$ would satisfy the same equations. In general, however, it does not. Nevertheless, we can use the desired response $h_d(n)$ for $n > M$ to construct an estimate of $h_d(n)$. That is,

$$\hat{h}_d(n) = -\sum_{k=1}^{N} a_k h_d(n-k)$$

Then we can select the filter parameters $\{a_k\}$ to minimize the sum of squared errors between the desired response $h_d(n)$ and the estimate $\hat{h}_d(n)$ for $n > M$. Thus we have

$$\varepsilon = \sum_{n=M+1}^{\infty} \left(h_d(n) - \hat{h}_d(n)\right)^2 = \sum_{n=M+1}^{\infty} \left(h_d(n) + \sum_{k=1}^{N} a_k h_d(n-k)\right)^2$$

The minimization of ε, with respect to the pole parameters $\{a_k\}$, leads to the set of linear equations

$$\sum_{l=1}^{N} a_l r_{hh}(k,l) = -r_{hh}(l,0) \quad k = 1, ..., N$$

where $r_{hh}(k,l)$ is defined as

$$r_{hh}(k,l) = \sum_{n=M+1}^{\infty} h_d(n-l) h_d(n-k)$$

.

The parameters $\{b_k\}$ that determine the zeros of the filter can be obtained simply from the equations of $n \le M$, where $h(n) = h_d(n)$, by substitution of the values $\{\hat{a}_k\}$ obtained by solving the

minimization. Thus

$$\hat{b}_n = h_d(n) + \sum_{k=1}^{N} \hat{a}_k h_d k - n \quad 0 \le n \le M$$

Therefore, the parameters $\{\hat{a}_k\}$ that determine the poles are obtained by the method of least squares while the parameters $\{\hat{b}_k\}$ that determine the zeros are obtained by the the Padé approximation method. The foregoing approach for determining the poles and zeros of $H(z)$ is sometimes called *Prony's method*.

The least-squares method provides good estimates for the pole parameters $\{a_k\}$. However, Prony's method may not be as effective in estimating the parameters $\{b_k\}$ since their computation is not based on the least-squares method.

Observation. An alternative method in which both sets of parameters $\{a_k\}$ and $\{b_k\}$ are determined by application of the least-squares method has been proposed by Shanks and can be found in [**17**].

3.5.3. Wiener filters. In the preceding methods we described the use of the least-squares error criterion in the design of pole-zero filters. In this one we use a similar approach to determine a least-squares FIR inverse filter to a desired filter.

The inverse to a linear time-invariant system with impulse response $h(n)$ and system function $H(z)$ is defined as the system whose impulse response $h_I(n)$ and system function $H_I(z)$, satisfy the respective equations.

$$h(n) * h_I(n) = \delta(n)$$

$$H(z)H_I(z) = 1$$

In general, $H_I(z)$ is IIR, unless $H(z)$ is an all-pole system, in which case $H_I(z)$ is FIR.

In many practical applications, it is desirable to restrict the inverse filter to be FIR. Obviously, one simple method is to truncate $h_I(n)$. In so doing, we incur a total squared approximation error equal to

$$\varepsilon_t = \sum_{n=M+1}^{\infty} h_I^2(n)$$

where $M + 1$ is the length of the truncated filter and ε_t, represents the energy in the tail of the impulse response $h_I(n)$.

Alternatively, we can use the least-squares error criterion to optimize the $M + 1$ coefficients of the FIR filter. First, let $d(n)$ denote the desired output sequence of the FIR filter of length $M + 1$ and let $h(n)$ be the input sequence. Then, if $y(n)$ is the output sequence of the filter, the error sequence between the desired output and the actual output is

$$e(n) = d(n) - \sum_{k=0}^{M} b_k h(n-k)$$

where $\{b_k\}$ are the FIR filter coefficients.

The sum of squares of the error sequence is

$$\varepsilon = \sum_{n=0}^{\infty} \left(d(n) - \sum_{k=0}^{M} b_k h(n-k) \right)^2$$

When ε is minimized with respect to the filter coefficients, we obtain the set of linear equations

$$\sum_{k=0}^{M} b_k r_{hh}(k-l) = r_{dk}(l) \qquad l = 0, ..., M$$

where $r_{hh}(l)$ is the autocorrelation function of $h(n)$, defined as

$$r_{hh}(l) = \sum_{k=0}^{\infty} h(n)h(n-l)$$

and $r_{dh}(l)$ is the crosscorrelation between the desired output $d(n)$ and input sequence $h(n)$, defined as

$$r_{dh}(l) = \sum_{n=0}^{\infty} d(n)h(n-l)$$

The optimum, in the least-squares sense, FIR filter that satisfies the linear equations is called the *Wiener filter*.

Therefore, the coefficients of the least-squares FIR filter are obtained from the solution of the linear equations, which can be expressed in matrix form

$$\begin{pmatrix} r_{hh}(0) & r_{hh}(1) & r_{hh}(2) & ... & r_{hh}(M) \\ r_{hh}(1) & r_{hh}(0) & r_{hh}(1) & ... & r_{hh}(M-1) \\ \vdots & \vdots & \vdots & & \vdots \\ r_{hh}(M) & r_{hh}(M-1) & r_{hh}(M-2) & ... & r_{hh}(0) \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_M \end{pmatrix} = \begin{pmatrix} r_{dh}(0) \\ r_{dh}(1) \\ \vdots \\ r_{dh}(M) \end{pmatrix}$$

We observe that the matrix is not only symmetric but it also has the special property that all the elements along any diagonal are equal. Such a matrix is called a *Toeplitz matrix* and lends itself to efficient inversion by means of the Levinson-Durbin algorithm[**3**].

The minimum value of the least-squares error obtained with the optimum FIR filter is

$$\varepsilon_{\min} = \sum_{n=0}^{\infty} \left(d(n) - \sum_{k=0}^{M} b_k h(n-k) \right) d(n) = \sum_{n=0}^{\infty} d^2(n) - \sum_{k=0}^{M} b_k r_{dh}(k)$$

3.6. Sampling

Most signals of practical interest, such as speech, biological signals, seismic signals, radar signals, sonar signals and various communications signals such as audio and video signals, are continuous-time signals. To process analog signals by digital means, it is first necessary to convert them into digital form. That is, to convert them to a sequence of numbers having finite precision.

Definition. The reduction of a continuous signal to a discrete signal is called *sampling*.

There are many ways to sample a continuous-time slgnal. We limit our discussion to *periodic or uniform sampling*, which is the type of sampling used most often in practice. This is described by the relation

$$x(n) = x_a(nT) \qquad -\infty < n < \infty$$

where $x(n)$ is the discrete-time signal obtained by taking samples of the continuous signal $x_a(t)$ every T seconds.

Definition. The time interval T between successive samples is called the *sampling period* or sample interval and its reciprocal $1/T = F_s$ is called the *sampling rate* (samples per second) or the sampling frequency (hertz).

We can now ask: under what circumstances is it possible to reconstruct the original signal completely and exactly (perfect reconstruction)?

A partial answer is provided by the *Shannon-Nyquist sampling theorem*.

Theorem 5. *Shannon-Nyquist sampling theorem*: Let $x(t)$ represent a continuous-time bandlimited signal and $X(f)$ be the continuous Fourier transform of that signal. Let $\text{Supp}(X) \subseteq [-B, B]$. Then the sufficient condition for exact reconstructability from samples at a uniform sampling rate F_s is that $F_s \geq 2B$.

Definition. The quantity $2B$ is called the *Nyquist rate*.

3.7. Analog-digital conversion

To process analog signals by digital means, it is first necessary to convert them into digital form. This procedure is called *analog-to-digital conversion*, and the corresponding devices are called *analog-to-digital converters* (ADCs).

Conceptually, we view AD conversion as a three-step process. This process is illustrated in Figure 2.

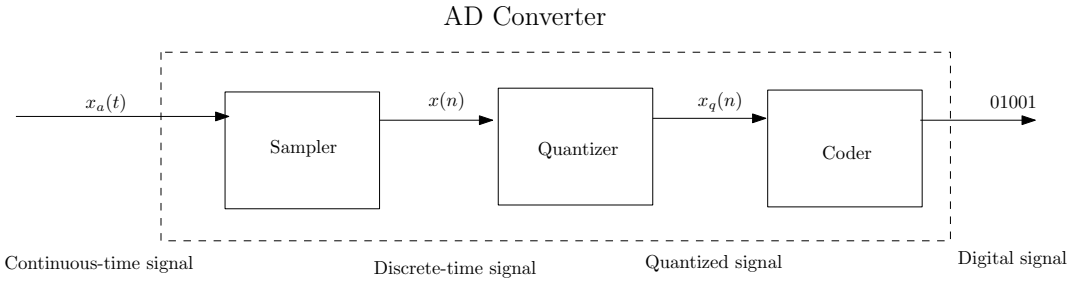


FIG. 2. Block diagram of a ADC

The three steps are:

(1) *Sampling*. This is the conversion of a continuous-time signal into a discrete-time signal obtained by taking samples of the continuous-time signal at discrete-time instants.
(2) *Quantization*. This is the conversion of a discrete-time continuous-valued signal into a discrete-time, discrete-valued (digital) signal. The value of each signal sample is represented by a value selected from a finite set of possible values. The difference between the unquantized sample $x(n)$ and the quantized output $x_q(n)$ is called the *quantization error*.
(3) *Coding*. In the coding process, each discrete value $x_q(n)$ is represented by a bit binary sequence.

3.8. Digital-analog conversion

In many cases of practical interest (e.g. speech processing) it is desirable to convert the processed digital signals into analog form. (Obviously, we cannot listen to the sequence of samples representing a speech signal or see the numbers corresponding to a TV signal.) The process of converting a digital signal into an analog signal is known as *digital-to-analog conversion*. The devices that convert digital code to analog signals are *digital-to-analog converters* (DAC). These devices fill the gaps in a digital signal by performing some kind of interpolation, whose accuracy depends on the quality of the digital-to-analog conversion process.

The simplest form of DAC is called a zero-order hold or a staircase approximation, which simply holds constant the value of one sample until the next one is received. Other approximations are possible, such as linearly connecting a pair of successive samples (linear interpolation), fitting a quadratic through three successive samples (quadratic interpolation) and so on.

For signals having a limited frequency content (finite bandwidth), the sampling theorem introduced in the previous section specifies the optimum form of interpolation, which is using a lowpass filter with cutoff frequency larger than the Nyquist rate after the zero-order holder. The effect of this is that the output voltage is held in time at the current value until the next input number is read. The resulting output is thus a piecewise constant or staircase shaped. This has an effect on the frequency response of the reconstructed signal.

The fact that DACs output a sequence of piecewise constant values or rectangular pulses causes multiple harmonics above the Nyquist frequency. Usually, these are removed with a low pass filter acting as a reconstruction filter in applications that require it.

Chapter 4
Wavelets

Wavelet analysis has begun to play a serious role in a broad range of applications, including signal processing, data and image compression, etc. The aim of this chapter is to present a theoretical approach to this type of functions. The last sections of this chapter deal with some applications. Furthermore, we explain the wavelet transform, which is a generalization of the Fourier transform [**1, 5, 18**]. In our theoretical approach we will follow [**18**].

4.1. Wavelets matrices

Definition. Let $\mathbb{F}$ be a subfield of the field $\mathbb{C}$ of complex numbers. Consider an array $A = (a_k^s)$ consisting of m rows of possibly infinite vectors of the form

$$\begin{pmatrix} \dots & a_{-1}^0 & a_0^0 & a_1^0 & a_2^0 & \dots \\ \dots & a_{-1}^1 & a_0^1 & a_1^1 & a_2^1 & \dots \\ & \vdots & \vdots & \vdots & \vdots & \\ \dots & a_{-1}^{m-1} & a_0^{m-1} & a_1^{m-1} & a_2^{m-1} & \dots \end{pmatrix}$$

where each a_k^s is an element of $\mathbb{F}$ and $m \geq 2$. We will refer to this array A as a *matrix* even though the rows might have infinite length.

We will define the A_l submatrices of A of size $m \times m$ as follows:

$$A_l = (a_{lm+r}^s), \quad r = 0, \dots, m-1, \ s = 0, \dots, m-1$$

for $l \in \mathbb{Z}$.

Definition. We call the *Laurent series of the matrix A* to the formal power series

$$A(z) = \sum_{l=-\infty}^{+\infty} A_l z^l$$

We can represent it as

$$A(z) = \begin{pmatrix} \sum_{k=-\infty}^{+\infty} a_{mk}^0 z^k & \cdots & \sum_{k=-\infty}^{+\infty} a_{mk+m-1}^0 z^k \\ \vdots & \ddots & \vdots \\ & \sum_{k=-\infty}^{+\infty} a_{mk+r}^s z^k & \\ \vdots & & \vdots \\ \sum_{k=-\infty}^{+\infty} a_{mk}^{m-1} z^k & \cdots & \sum_{k=-\infty}^{+\infty} a_{mk+m-1}^{m-1} z^k \end{pmatrix}$$

Definition. Assume that there are a finite number of columns. So,

$$A(z) = \sum_{l=N_1}^{N_2} A_l z^l$$

where we assume that A_{N_1} and A_{N_2} are nonzero matrices. We defined the *genus* of the Laurent series $A(z)$ as $g = N_2 - N_1$.

Definition. The *adjoint* of the Laurent matrix $A(z)$, $\tilde{A}(z)$, is defined by

$$\tilde{A}(z) = A^*(z^{-1}) = \sum_{l=-\infty}^{+\infty} A_l^* z^{-l}$$

Definition. The matrix A is said to be a *wavelet matrix of rank m* if

- $A(z)\tilde{A}(z) = mI$
- $\sum_{k=-\infty}^{+\infty} a_k^s = m\delta^{s,0}$, $0 \le s \le m-1$ where $\delta^{i,j}$ is the Kronecker function.

Proposition. A wavelet matrix with m rows has rank m in the classical sense.

Definition. We denote as $WM(m, g; \mathbb{F})$ the *set of wavelet matrices* of rank m and genus g with coefficients in the field $\mathbb{F}$.

Example.

(1) *Haar matrices of rank* 2*:* The wavelet matrices with genus 1 are called *Haar matrices*. The matrices

$$\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$$

are both wavelet matrices of rank 2, and as one can check, they are the only square wavelet matrices of rank 2 with real coefficients. More generally if we allow complex coefficients, we can easily see that the general complex Haar wavelet matrix of rank 2 has the form for $\theta \in \mathbb{R}$

$$\begin{pmatrix} 1 & 1 \\ -e^{i\theta} & e^{i\theta} \end{pmatrix}$$

(2) *Daubechies Wavelet Matrix of rank* 2 *and genus* 2*:*Let

$$D_2 = \frac{1}{4}\begin{pmatrix} 1+\sqrt{(3)} & 3+\sqrt{(3)} & 3-\sqrt{(3)} & 1-\sqrt{(3)} \\ -1+\sqrt{(3)} & 3-\sqrt{(3)} & -3-\sqrt{(3)} & 1+\sqrt{(3)} \end{pmatrix}$$

This is one of a series of rank 2 wavelet matrices discovered by Daubechies. It leads to a wavelet system which has a scaling function which is continuous with compact support.

(3) *The Discrete Fourier Transform Matrix:* Let $m > 1$ be an integer and $w = e^{i2\pi/N}$ be a primitive N-th root of unity. The Discrete Fourier Transform matrix (DFT) of rank N as defined in 2.6 is

$$W = \begin{pmatrix} 1 & 1 & 1 & 1 & ... & 1 \\ 1 & w & w^2 & w^3 & ... & w^{N-1} \\ 1 & w^2 & w^4 & w^6 & ... & w^{2(N-1)} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & w^{N-1} & w^{2(N-1)} & w^{3(N-1)} & ... & w^{(N-1)(N-1)} \end{pmatrix}$$

W is a Haar matrix over $\mathbb{C}$; if $m = 2$ then W is defined over $\mathbb{R}$, and is identical to the canonical Haar matrix H.

4.2. Wavelet systems

Let $A \in WM(m, g; \mathbb{C})$ be a wavelet matrix and consider the functional difference equation

$$\varphi(x) = \sum_{k=0}^{mg-1} a_k^0 \varphi(mx - k)$$

Definition. This equation is called the *scaling equation* associated with the wavelet matrix A. If φ is a solution of this equation, then φ is called a *calling function* associated with A.

Definition. We defined the *wavelet functions* $\{\psi^1, ..., \psi^{m-1}\}$ associated with the wavelet matrix A and the scaling function φ by the formula

$$\psi^s(x) = \sum_{k=0}^{mg-1} a_k^s \varphi(mx - k)$$

Theorem 6. Let $A \in WM(m, g; \mathbb{C})$ be a wavelet matrix. Then, there exists a unique $\varphi \in L^2(\mathbb{R})$ such that:

(1) φ satisfies $\varphi(x) = \sum_{k=0}^{mg-1} a_k^0 \varphi(mx - k)$
(2) $\int_{\mathbb{R}} \varphi(x)dx = 1$
(3) $\text{Supp}(\varphi) \subset [0, (g-1)(\frac{m}{m-1} + 1)]$

Let A be a wavelet matrix of rank m and $\{\varphi, \psi^1, ..., \psi^{m-1}\}$ its scaling and wavelet functions. For $k, j \in \mathbb{Z}$, we define

$$\varphi_{jk} = m^{\frac{j}{2}} \varphi(m^j x - k)$$

$$\psi^s_{jk}(x) = m^{\frac{j}{2}} \psi^s(m^j x - k)$$

These are the *rescaled* and *translated* scaling and wavelet functions. We introduce the notation

$$\varphi_k(x) = \varphi_{0k}(x)$$

Definition. The *wavelet system* $W[A]$ associated with the wavelet matrix A is the collection of functions

$$W[A] = \{\varphi_k(x), k \in \mathbb{Z}\} \cup \{\psi^s_{jk}(x), j, k \in \mathbb{Z}, j \geq 0, s = 1, ..., m-1\}$$

Theorem 7. Let $A \in WM(m, g; \mathbb{C})$, let $W[A]$ its wavelet system and let $f \in L^2(\mathbb{R})$. Then, there exists an L^2-convergent expansion

$$f(x) = \sum_{k=-\infty}^{+\infty} c_k \varphi_k(x) + \sum_{s=1}^{m-1} \sum_{j=0}^{+\infty} \sum_{k=-\infty}^{+\infty} d^s_{jk} \psi^s_{jk}(x)$$

where the coefficients are given by

$$c_k = \int_{\mathbb{R}} f(x)\varphi_k(x)dx$$

$$d^s_{jk} = \int_{\mathbb{R}} f(x)\psi^s_{jk}(x)dx$$

Definition. Let $\mathcal{H}$ be a Hilbert space with an inner product $\langle \cdot, \cdot \rangle$, and let $\{h_\alpha\}$ a countable set in $\mathcal{H}$. Then $\{h_\alpha\}$ is a *tight frame* if, for some $c \geq 0$,

$$f = c \sum_{\alpha=-\infty}^{+\infty} \langle h_\alpha, f \rangle h_\alpha$$

for each $f \in \mathcal{H}$.

Observation. Clearly, an orthonormal basis for $\mathcal{H}$ has this property with $c = 1$.

Observation. For most wavelet matrix A, the wavelet system $W[A]$ will be a complete orthonormal system and hence an orthonormal basis for $L^2(\mathbb{R})$. However, for some wavelet matrices, the system is not orthonormal and yet the theorem holds. In this case, the system is said to be a *tight frame*.

Example.

(1) *The Haar functions:* If

$$H = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$$

is the canonical Haar wavelet matrix of rank 2, then the scaling and wavelet functions φ and ψ given by Theorem 6 can be determined in an elementary fashion: no iteration scheme is necessary. Their graphs are shown in Figure 1.

(2) *Daubechies Wavelet functions:* Let $D_2 \in WM(2, 2, \mathbb{R})$ be the rank 2 wavelet matrix discovered by Daubechies (Example in Section 4.1), and let φ and ψ be the corresponding scaling and wavelet functions (Figure 2 colour red and green respectively); the common support of *varphi* and the wavelet function ψ is $[0, 3]$.

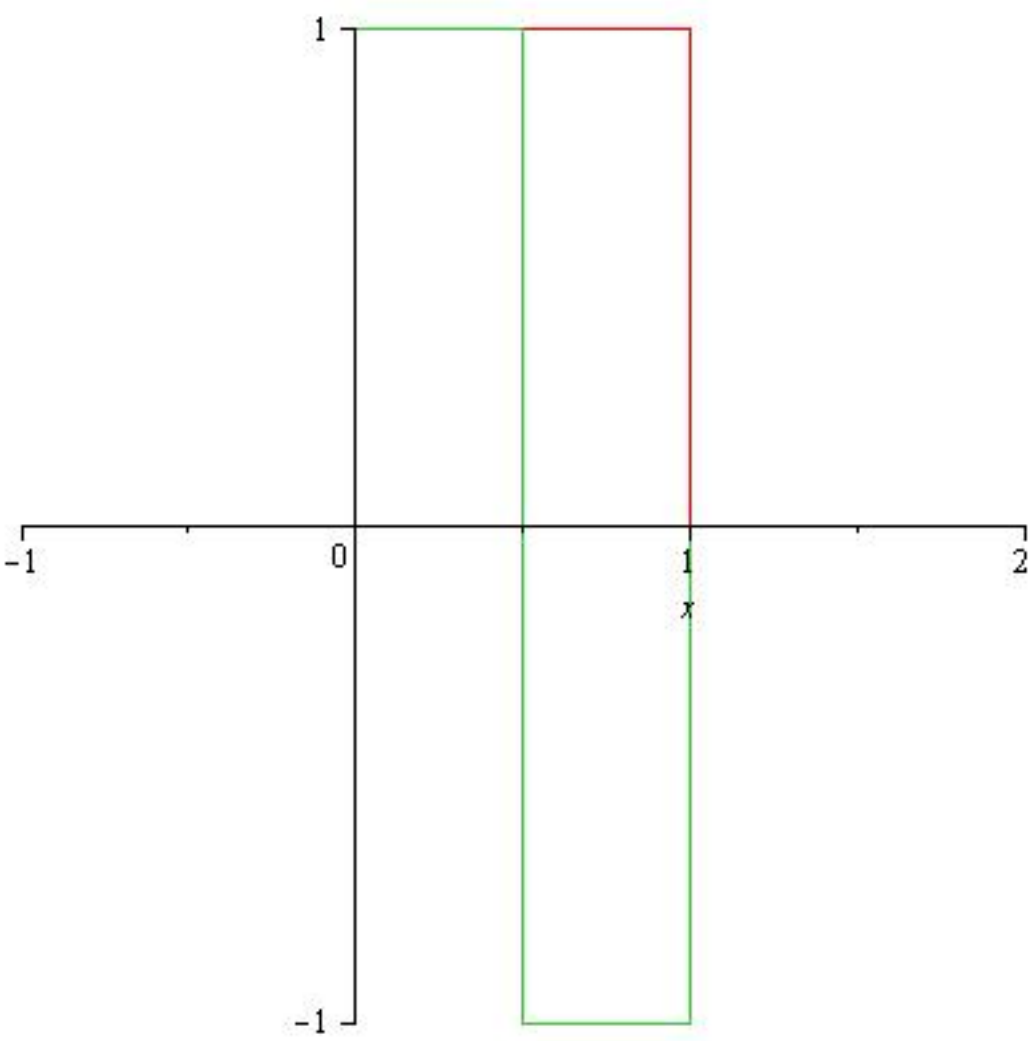


FIG. 1. Haar scaling function (red) and Haar wavelet function (green).

4.3. Biorthogonal wavelet systems

Definition. The $L = (a_i^j)$ and $R = (\tilde{a}_i^j)$ be $m \times mg$ matrices with complex-valued entries are to be a *wavelet matrices pair* of rang m and genus g, (L, R), if

$$L(z)\tilde{R}(z) = mI_m$$

$$\sum_{k=0}^{mg-1} a_k^r = \sum_{k=0}^{mg-1} \tilde{a}_k^r = m\delta^{r,0}$$

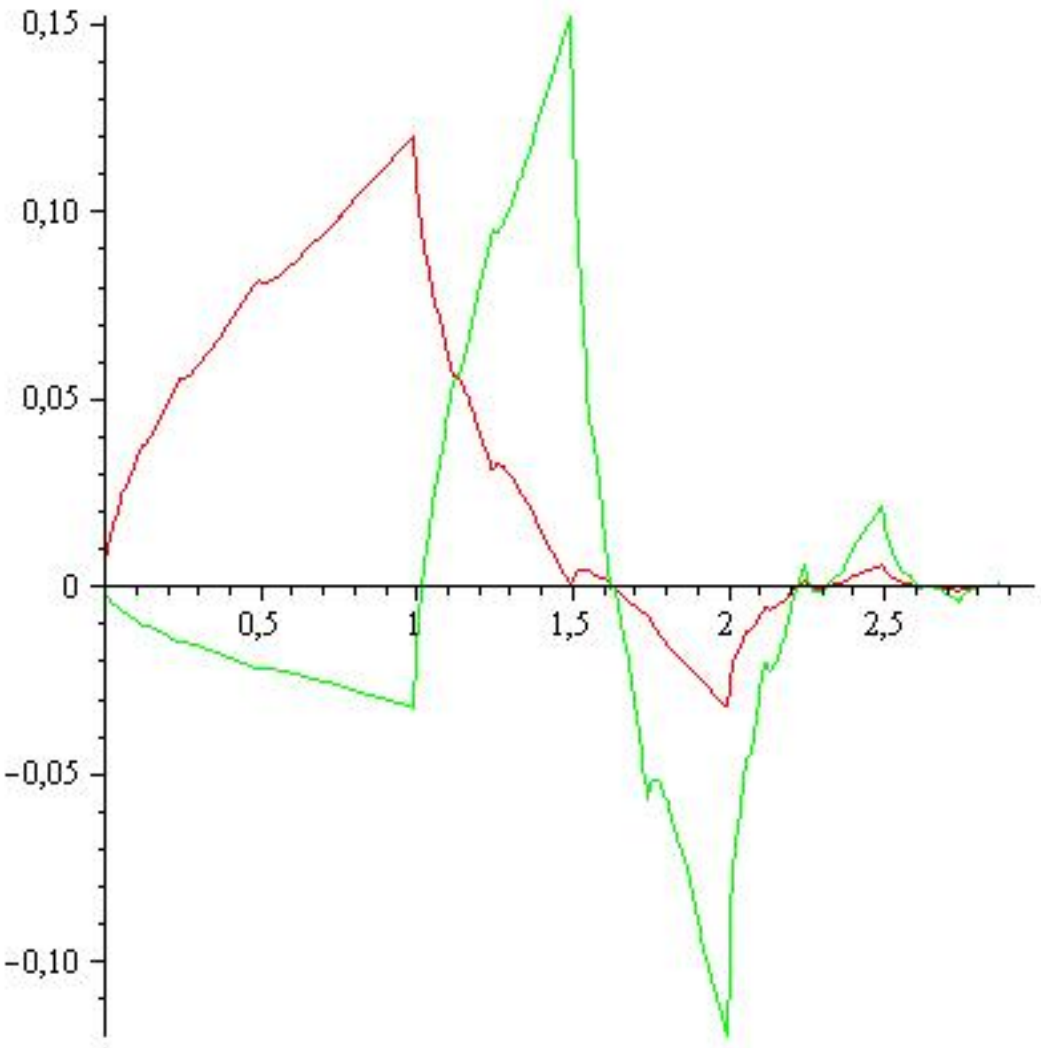


FIG. 2. Daubechies scaling function (red) and Daubechies wavelet function (green).

Let (L, R) be a wavelet matrix pair and $\{\varphi, \tilde{\varphi}, \psi^r, \tilde{\psi}^r, r = 1, ..., m-1\}$ functions satisfying the scaling and wavelt equations

$$\varphi(x) = \sum_{k=0}^{mg-1} a_k^0 \varphi(mx - k)$$

$$\psi^r = \sum_{k=0}^{mg-1} a_k^r \varphi(mx - k)$$

$$\tilde{\varphi}(x) = \sum_{k=0}^{mg-1} \tilde{a}_k^0 \tilde{\varphi}(mx-k)$$
$$\tilde{\psi}^r = \sum_{k=0}^{mg-1} \tilde{a}_k^r \tilde{\varphi}(mx-k)$$

for $r = 1, ..., m-1$. This functions are called *biorthogonal scaling functions* $\{\varphi, \tilde{\varphi}\}$ and *biorthogonal wavelet functions* $\{\psi^r, \tilde{\psi}^r, r = 1, ..., m-1\}$ respectively.

Definition. A *biorthogonal wavelet system*, $W[L, R]$ associated with a wavelet matrix pair (L, R) are the rescaling and translates of its biorthogonal scaling and wavelet functions

$$W[L,R] = \{\varphi_k(x), \tilde{\varphi}_k(x), k \in \mathbb{Z}\} \cup \{\psi^r_{jk}, \tilde{\psi}^r_{jk}, j, k \in \mathbb{Z}, j \geq 0, r = 1, ..., m-1\}$$

Observation. A biorthogonal wavelet system satisfies the following biorthogonality proprieties:

$$\int \varphi_k \tilde{\varphi}_{k'} = \delta_{k,k'}$$
$$\int \varphi_k \tilde{\psi}^r_{jk'} = 0, \quad r = 1, ..., m-1$$
$$\int \psi^r_{jk} \tilde{\varphi}_{k'} = 0, \quad r = 1, ..., m-1$$
$$\int \psi^r_{jk} \tilde{\psi}^{r'}_{j'k'} = \delta^{r,r'} \delta_{j,j'} \delta_{k,k'}$$

Theorem 8. If (L, R) is a wavelet matrix pair, then there exists an associated biorthogonal wavelet system $\{\varphi_k, \psi^r_{jk}, \tilde{\varphi}_k, \tilde{\psi}^r_{jk}\}$ and, moreover, for any $f \in L^2(R)$,

$$f(x) = \sum_{k=-\infty}^{+\infty} \langle f, \varphi_k \rangle \tilde{\varphi}_k(x) + \sum_{r=1}^{m-1} \sum_{j=0}^{+\infty} \sum_{k=-\infty}^{+\infty} \langle f, \psi^r_{jk} \rangle \tilde{\psi}^r_{jk}(x)$$

where the convergence is in the weak L^2 sense.

Example.

(1) *wavelet matrices:* Every wavelet matrix A defines a wavelet matrix pair (A, A). An example could be the Haar matrix of rank 2

$$H_L = H_R = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$$

(2) *Rank* 2 *and genus* 2*:* Here is an example of a wavelet matrix pair of rank $m = 2$ and genus $g = 2$:

$$L = \begin{pmatrix} 1/3 & 2/3 & 2/3 & 1/3 \\ -1 & -2 & 2 & 1 \end{pmatrix}$$
$$R = \begin{pmatrix} -1 & 2 & 2 & -1 \\ 1/3 & -2/3 & 2/3 & -1/3 \end{pmatrix}$$

(3) *Daubechies type, genus 3:* A complex wavelet matrix pair for $m = 2$, $g = 3$.

$$L = \begin{pmatrix} -0.13 + 0.07i & 0.11 + 0.21i & 1.02 + 0.14i & 1.02 - 0.14i & 0.11 - 0.21i & -0.13 - 0.07i \\ -0.13 - 0.07i & -0.11 + 0.21i & 1.02 - 0.14i & -1.02 - 0.14i & 0.11 + 0.21i & 0.13 - 0.07i \end{pmatrix}$$

$$R = \begin{pmatrix} -0.13 - 0.07i & 0.11 - 0.21i & 1.02 - 0.14i & 1.02 + 0.14i & 0.11 + 0.21i & -0.13 + 0.07i \\ -0.13 + 0.07i & -0.11 - 0.21i & 1.02 + 0.14i & -1.02 + 0.14i & 0.11 - 0.21i & 0.13 + 0.07i \end{pmatrix}$$

4.4. The Mallat algorithm

Let

$$A = \begin{pmatrix} a_0 & a_1 & \dots & a_{2g-1} \\ b_0 & b_1 & \dots & b_{2g-1} \end{pmatrix}$$

be a wavelet matrix of rank 2 and genus g, and let $W = \{\varphi_k, \psi_{jk}\}$ be the corresponding orthonormal wavelet system of rank 2 and genus g. Introduce

$$\varphi_{jk} = 2^{j/2}\varphi(2^j x - k)$$

the rescaling and translations of the scaling function $\varphi(x)$. We assume orthonormality for convenience of exposition, although the analysis we present here is valid in general case of tight frames. For $j \in \mathbb{Z}$, define the vector space

$$V_j = \overline{\text{Span}\{\varphi_{jk} \mid k \in \mathbb{Z}\}}^1$$

It can be proved that $L^2(\mathbb{R}) = \cup_j V_j$, so we have the orthogonal projections $P_j : L^2(\mathbb{R}) \to V_j$. Hence, given a function $f \in L^2(\mathbb{R})$, $P_j f$ converges to f in the L^2 norm and its coefficients are given by the formula

$$P_j f(x) = \sum_{k=-\infty}^{+\infty} c_{jk}\varphi_{jk}(x)$$

where

$$c_{jk} = \int_{\mathbb{R}} f(x)\varphi_{jk}(x)dx$$

since the φ_{jk}'s are a tight frame for V_j. If we let W_j denote the orthogonal complement of V_j in V_{j+1}, then for a fixed $J \in \mathbb{Z}^+$

$$V_0 \oplus W_0 \oplus \cdots \oplus W_{J-1} = V_J$$

Moreover, if

$$f(x) = \sum_{k=-\infty}^{+\infty} c_{0k}\varphi_{0k}(x) + \sum_{j=0}^{+\infty}\sum_{k=-\infty}^{+\infty} d_{jk}\psi_{jk}(x)$$

for $f \in L^2(\mathbb{R})$, then we obtain

$$\sum_{k=-\infty}^{+\infty} c_{Jk}\varphi_{Jk}(x) = \sum_{k=-\infty}^{+\infty} c_{0k}\varphi_{0k}(x) + \sum_{j=0}^{J-1}\sum_{k=-\infty}^{+\infty} d_{jk}\psi_{jk}(x)$$

[1]The span of S is defined as the set of all linear combinations of the elements of S not necessary finite

So, one can determine the coefficients on the right-hand side of the previous equation in terms of the coefficients on the left-hand side, and conversely. This is the *Mallet algorithm*.

Here is how the Mallat algorithm works: Suppose we are given an expansion of the form

$$\overline{f}^J = \sum_{k=-\infty}^{+\infty} c_{Jk}\varphi_{Jk}(x),$$

and we want to determine the coefficients of the corresponding lower-order expansion of the type

$$\overline{f}^J = \sum_{k=-\infty}^{+\infty} c_{0k}\varphi_{0k}(x) + \sum_{j=0}^{J-1}\sum_{k=-\infty}^{+\infty} d_{jk}\psi_{jk}(x)$$

We will do this successively in stages. First, we consider the decomposition

$$V_{J-1} \oplus W_{J-1} = V_J,$$

and determine the expansion coefficients $in V_{J-1}$ and W_{J-1} for the coefficients in V_J.

Thus, we can write

$$\sum c_{J-1,k}\varphi_{J-1,k} + \sum d_{J-1,k}\psi_{J-1,k} = \sum c_{J,k}\varphi_{J,k}$$

and try to determine the pair $\{c_{J-1,k}, d_{J-1,k}\}$ in terms of the $\{c_{J,k}\}$ and conversely. First, the last equation by $\varphi_{J-1,l}$ and integrate, obtaining

$$c_{J-1,l} = \sum c_{J,k} \int \varphi_{J,k}(x)\varphi_{J-1,l}(x)dx$$

using the orthonormality of W. Now use the basic scaling equation and wavelet defining equation to write

$$\varphi_{J-1,l}(x) = \frac{1}{\sqrt{2}} \sum_{r=0}^{2g-1} a_r\varphi_{J,r+2l}(x) \quad (*)$$

$$\psi_{J-1,l}(x) = \frac{1}{\sqrt{2}} \sum_{r=0}^{2g-1} b_r\varphi_{J,r+2l}(x) \quad (**)$$

Substituting $(*)$ into $(**)$ and using the orthogonality, we obtain

$$c_{J-1,l} = \frac{1}{\sqrt{2}} \sum_r c_{J,r}a_{r-2l}$$

where we assume, as usual, that a_k and b_k are zero for $k < 0$ and for $k > 2g - 1$. Similarly, we find that

$$d_{J-1,l} = \frac{1}{\sqrt{2}} \sum_r c_{J,r}b_{r-2l}$$

More generally, by expressing

$$V_{J-2} \oplus W_{J-2} = V_{J-1}$$

$$\vdots$$

$$V_{j-1} \oplus W_{j-1} = V_j$$

$$\vdots$$

$$V_0 \oplus W_0 = V_1$$

we obtain inductively defined formulae generating the coefficients $\{c_{0k}, d_{jk}, i \leq j \leq J-1\}$, from the coefficients $\{c_{jk}\}$, given by

$$c_{j-1,l} = \frac{1}{\sqrt{2}} \sum_r c_{j,r} a_{r-2l}$$

$$d_{j-1,l} = \frac{1}{\sqrt{2}} \sum_r c_{j,r} b_{r-2l}$$

for $0 \leq j \leq J$. The inverse of these formulae is derived in the same fashion. We simply consider

$$V_{j-1} \oplus W_{j-1} = V_j$$

and suppose that we know the coefficients of the expression in V_{j-1} and W_{j-1}, and we want to determinate the coefficients in the expression in V_j. Using the same proceed we obtain the inductive formula

$$c_{j,k} = \frac{1}{\sqrt{2}} \sum_k c_{j-1,k} a_{l-2k} + d_{j-1,k} bl - 2k$$

for $j = 1, ..., J$.

4.5. The Mallat algorithm for periodic data

The Mallat algorithm gives a linear mapping

$$V_J \xrightarrow{M} V_0 \oplus W_0 \oplus ... \oplus W_{J-1}$$

and its inverse. These spaces are all infinite dimensional. We want to consider a special case where we replace each space V_j and W_j by its periodic analogue, thereby providing a finite-dimensional version of the Mallat algorithm, which is what is needed in practise. We define the periodic scaling and wavelet spaces with period L as

$$V_j^L = \left\{ \sum c_{j,k} \varphi_{jk} \mid c_{j,k} = c_{j,k+L}, k \in \mathbb{Z} \right\}$$

$$W_j^L = \left\{ \sum d_{j,k} \psi_{jk} \mid d_{j,k} = d_{j,k+L}, k \in \mathbb{Z} \right\}$$

if we assume that L is even, then the Mallat algorithm maps naturally:

$$\begin{array}{ccc} V_j^L & \simeq & \mathbb{R}^L \\ \downarrow M & & \downarrow H \oplus G \\ V_{j-1}^L \oplus W_{j-1}^L & \simeq & \mathbb{R}^{L/2} \oplus \mathbb{R}^{L/2} \end{array}$$

and we see that the linear mapping M can be represented by a pair of matrices H and G, which are illustrated for the case that $g = 2$ and $L = 8$:

$$H = \begin{pmatrix} h_0 & h_1 & h_2 & h_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_0 & h_1 & h_2 & h_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_0 & h_1 & h_2 & h_3 \\ h_2 & h_3 & 0 & 0 & 0 & 0 & h_0 & h_1 \end{pmatrix}$$

and G will be similar with entries $\{g_k\}$ replacing $\{h_k\}$. Here, we assume that the wavelet matrix A has the form

$$A = \begin{pmatrix} a_0 & a_1 & a_2 & a_3 \\ b_0 & b_1 & b_2 & b_3 \end{pmatrix} = \sqrt{2}\begin{pmatrix} h_0 & h_1 & h_2 & h_3 \\ g_0 & g_1 & g_2 & g_3 \end{pmatrix}$$

since the vectors (h_k) and (g_k) are what appear explicitly in the Mallat algorithm. The inverse Mallat algorithm will have a similar matrix representation.

Observation. The implementation of the Mallat algorithm is not carried out by matrix multiplication, but, by a convolution with the vectors $h = (h_0, h_1, h_2, h_3)$ and $g = (g_1, g_2, g_3, g_4)$ followed by a downsampling operator. In this case, we only take the even terms. This is schematically described in Figure 3.

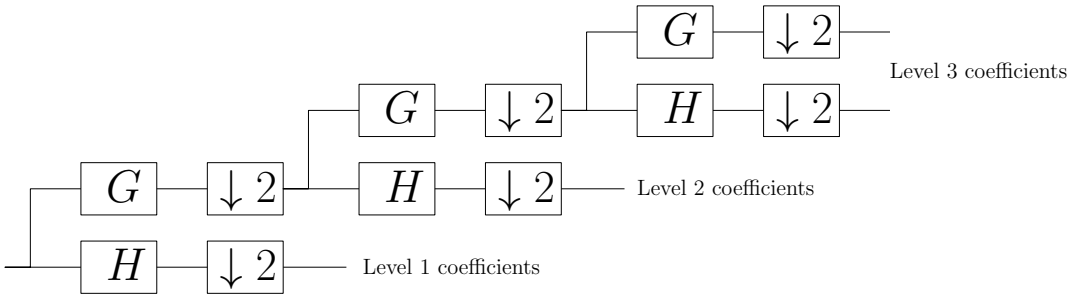


FIG. 3. Discrete wavelet transform scheme

Let $L = 2K$ and let $s_0, ..., s_{2K-1}$ a sequence of numbers, then

$$H(s_0, ..., s_{2K-1}) = (sl_0, ..., sl_{K-1}) \in \mathbb{R}^k \simeq V_{j-1}^L$$

$$G(s_0, ..., s_{2K-1}) = (sh_0, ..., sh_{K-1}) \in \mathbb{R}^k \simeq W_{j-1}^L$$

the low-pass and high-pass outputs at the Mallat mapping M, respectiely, are calculated as follows. Assume $g = 2$, then

$$sl_0 = \sum_{l=0}^{3} s_l h_l \qquad sh_0 = \sum_{l=0}^{3} s_l g_l$$

$$sl_1 = \sum_{l=2}^{5} s_l h_{l-2} \qquad sh_1 = \sum_{l=2}^{5} s_l g_{l-2}$$

$$\vdots \qquad \vdots$$

$$sl_{K-1} = \sum_{l=2K-2}^{2K+1} s_l h_{l-2K-2} \qquad sh_{K-1} = \sum_{l=2K-2}^{2K+1} s_l g_{l-2K-2}$$

Observation. For the last sum, we extend the sequence periodically by setting $s_0 = s_{2K}$ and $s_1 = s_{2K+1}$.

Definition. Let A be a wavelet matrix of rank 2, and let $\mathbb{R}^{2^N}$ be the vector space of rank 2^N over $\mathbb{R}$ of 2^N-tuples $(x_0, ..., x_{2^N-1})$ of real numbers. Let J be an integer $J \leq N$. The *discrete wavelet transform (DWT) of order J* defined by A is a linear mapping

$$\begin{array}{ccccccccccccc}
\mathbb{R}^{2^N} & \overset{N}{\to} & \mathbb{R}^{2^{N-1}} & \oplus & \mathbb{R}^{2^{N-2}} & \oplus\cdots\oplus & \mathbb{R}^{2^{N-J+1}} & \oplus & \mathbb{R}^{2^{N-J}} & \oplus & \mathbb{R}^{2^{N-J}} \\
\wr & & \wr & & \wr & & \wr & & \wr & & \wr \\
V_N^L & \overset{N}{\to} & W_{N-1}^L & \oplus & W_{2^{N-2}}^L & \oplus\cdots\oplus & W_{2^{N-J+1}}^L & \oplus & W_{2^{N-J}}^L & \oplus & V_{2^{N-J}}^L
\end{array}$$

where we identify the vector spaces $\mathbb{R}^{2^j}$ with the periodic scaling and wavelet spaces of periods 2^j as indicated above. The mapping M is given by the Mallat algorithm as given by the recursively of the coefficients. The scheme is described in the figure 3.

Observation. The computational complexity of the discrete wavelet transform is $O(N)$. It is easy to see it since at every level the convolution involves L products and L summands, so the total number of operations at any level is bounded by $2LN$. Moreover at every level, we operate on half the number of points as at the previous level. Thus, we see that $(1 + \frac{1}{2} + ... + \frac{1}{2^J}) < 2$, so that we at most double the constant by going to a finite number J of levels.

4.6. Wavelet image compression

Today, the most successful general image compression method is based on the two dimensional product basis constructed from the Daubechies wavelet matrix D_3 and their biorthogonal variations. Recall that D_3 provides a low-pass representation of polynomials of degree less than 3. From this it follows that the product basis provides a low-pass expansion of polynomial functions of two variables, say x and y, of degree less than 3. These quadric surfaces are good models for light reflected from smooth surfaces such as walls, or the cheeks and forehead of a face. The ability to represent reflected light energy that produces quadric surfaces in a digital image means that this transform is effective in concentrating a large fraction of the image energy in a small number of low-pass ("near dc") transform coefficients. The figure 4 displays the scaling and wavelet function $\varphi(x)$ and $\psi(x)$.

The filters that correspond to these basis functions transform the original image into four components: the low-low, low-high, high-low and high-high parts of the transform.

Each of these outputs contains one quarter as many coefficients as the input image, so the total number of output coefficients is the same as the number of input coefficients. This fact makes it possible to display the transform output as an image also by scaling all outputs to fall in the range

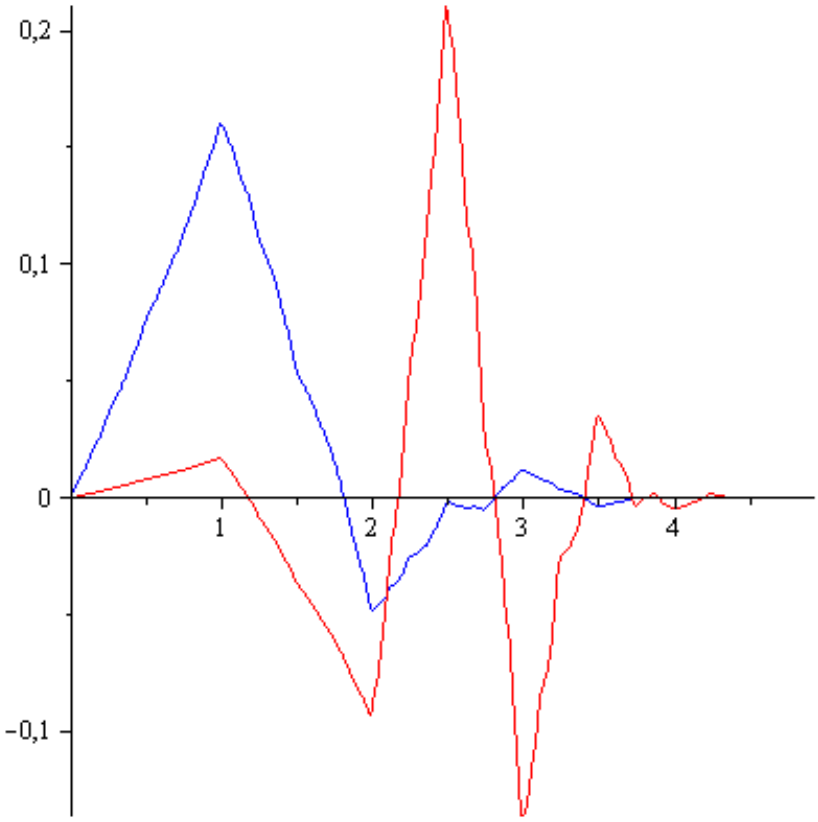


FIG. 4. Daubechies sclaing function φ for D_3 ($m = 2, g = 3$) (blue). Daubechies wavelet function ψ for D_3 ($m = 2, g = 3$) (red).

of the input pixel values - say the range 0 to 255 for an 8 bit gray-scale image.
The four images in the four subsquares of Figure 5 were constructed this way. They illustrate the first stage in applying the wavelet transform for compression, and correspond to the low-low, low-high,high-low, and high-high pass components of the original image (which looks like the low-low quadrant in the upper left).

4.7. Wavelet channel coding

The user of a digital communication system will primarily be concern with the bit rate and the error rate of the communication channel. The communication channel is controlled by allocating power and bandwidth, both of which are precious resources.

Every communication channel experiences noise. Additive white Gaussian noise (AWGN) is often a good first approximation because it results from specific applications. The communication system

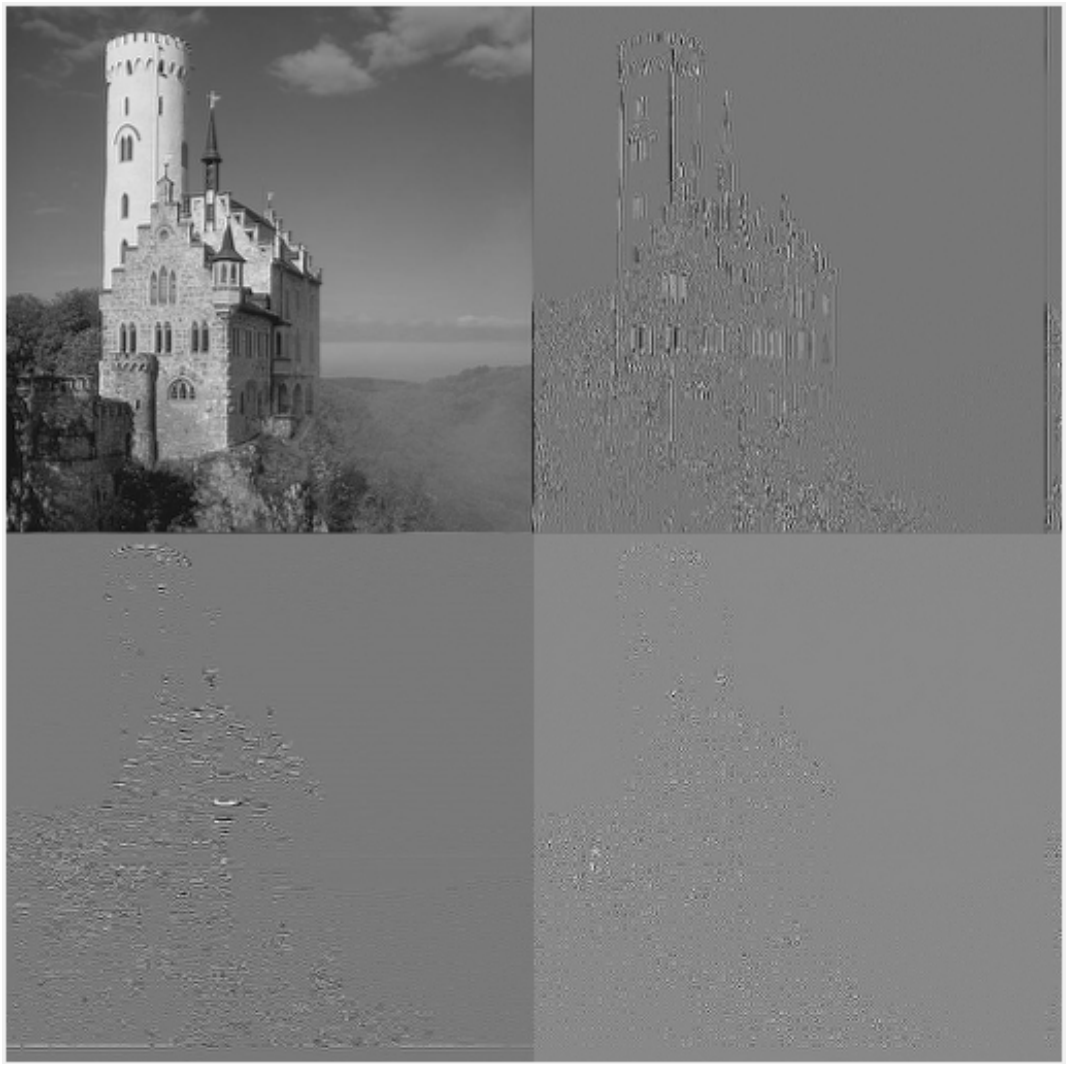


FIG. 5. Four quadrants of a level-1 wavelet decomposition of a Liechestein castle image.

designer would like to achieve the objectives of high bit rate and low error rate with a minimum of power consumption and bandwidth requirements. These objectives are inconsistent; trade-offs must be made that prefer high bit rate to low error rate or low error rate to high bit rate; low power requirements to low bandwidth or low bandwidth to low power requirements. These trade-offs determine the type of modulation and error correction that will be selected.

Wavelet channel coding (WCC) provides a systematic conceptual and design method for making these trade-offs through software selection within a common chipset-based hardware framework.

The advantages of wavelet channel coding stem from the combination of the exact orthogonality of wavelet codewords, the spreading of message symbol information throughout the codeword symbols, and the ability to utilise soft decisions to decode wavelet channel coded symbols.

The main properties of WCC coding are summarised in the following list:

(1) Wavelet Channel Coding is a new class of coding algorithms.
(2) WCC offers a systematic and simple approach to the full range of coding problems, from low rate, power efficient codes to high rate, bandwidth efficient codes.
(3) Wavelet Channel Coding can be applied in a block coding mode or in a trellis coding mode.
(4) WCC codewords are strictly orthogonal.
(5) WCC codewords have small correlations for non-codeword offsets.
(6) WCC coding employs robust soft decision decoding.
(7) WCC codes can be arbitrarily long.
(8) WCC codewords have desirable Hamming distance properties and provide good error correction capability.
(9) WCC codes can be efficiently coded and decoded with simple VLSI circuits.
(10) Synchronisation can be performed by correlation matching.
(11) Analysis and computer simulations show that WCC with arbitrary length wavelets achieves the AWGN performance of coherent BPSK[2]. When compared under a constant channel rate criterion WCC provides coding gains of approximately $10\log_2(mg+1)$ dB. On pulsed interference and flat fading channels WCC outperforms BPSK with gains that are dependent on the wavelet sequence length.

Example. Wavelet Channel Coding for $m = 2$

For the purpose of illustration, let a message

$$x_1, x_2, ..., x_n, ...$$

consist of a sequence of bits represented by the signed units $\{+1, -1\}$. Let

$$A = \begin{pmatrix} a_0 & a_1 & ... & a_{2g-1} \\ b_0 & b_1 & ... & b_{2g-1} \end{pmatrix}$$

be a wavelet matrix of rank 2 and genus g and let a and b denote the scaling and wavelet vectors of this wavelet matrix, respectively (first and second rows of A, or, in engineering language, the low and high pass filters of the 2-band filter bank A). Because the rows of A are mutually orthogonal when translated by steps of two, we can take advantage of the orthogonality of the filters (vectors) a and b to code at a rate of two message bits per $2k$ clock pulses, where $1 < k < \log_2 2g$. Message bits with odd sequence number will be coded using the waveform $(a_0, ..., a_{2g-1})$ and message bits with even sequence number will be coded using the waveform $(b_0, ..., b_{2g-1})$.

The coding procedure is illustrated in Table 1 for a system with maximally overlapped wavelet sequences. This corresponds to the case where the code rate is one information bit per WCC symbol (i.e. $k = 1$above). The sequence of clock pulses is arranged above the upper rule of the

[2]BPSK refers to Binary Phase Shift Keying and it is a modulation technique

1	2	3	4	...	$2g$	$2g+1$	$2g+2$	...
x_1a_0	x_1a_1	x_1a_2	x_1a_3	...	x_1a_{2g-1}	...	...	...
x_2b_0	x_2b_1	x_2b_2	x_2b_3	...	x_2b_{2g-1}	...	...	...
		x_3a_0	x_3a_1	...	x_3a_{2g-3}	x_3a_{2g-2}	x_3a_{2g-1}	...
		x_4b_0	x_4b_1	...	x_4b_{2g-3}	x_4b_{2g-2}	x_4b_{2g-1}	...
				...	$\vdots$	$\vdots$	$\vdots$	$\vdots$
y_1	y_2	y_3	y_4	...	y_{2g}	y_{2g+1}	y_{2g+2}	...

TABLE 1. Wavelet channel coding example.

table, and the sequence of transmitted symbols y_n is shown below the lower rule of the table. The encoding of each message bit x_n is shown in the nth row below the upper rule. Note that for odd n the encoding begins on the nth clock pulse and that for even n the encoding begins on the $(n-1)$th clock pulse. The symbol y_n is equal to the sum of the nth column of encoded message bit values and is therefore not restricted to the values ± 1. The formula for computing the symbol y_n to be transmitted at the nth clock pulse is

$$y_n = \sum_k \{x_{2k+1}a_{n-2k-1} + x_{2k+2}b_{n-2k-1}\}$$

Some observations are in order:

(1) It is evident that code rates as small as $1/g$ can be achieved by varying the overlap of the wavelet sequences; in the limit where the code rate is $1/g$, the wavelet sequences are non overlapped but adjacent.
(2) When flat wavelet matrices are used, the WC encoded symbols that are generated by the WCC algorithm are binomially distributed.

Chapter 5
Pattern theory

In this chapter we present some basic tools of pattern theory. We start with by introducing the concepts of entropy and kurtosis [**15**]. Next we study the Markov chain models [**8**]. In the last part, we use these tools to explain how certain types of speech recognizers work [**6, 7, 8, 13**].

5.1. Entropy

Definition. The *information* gained by knowing that an event A has occurred is defined by

$$I(A) = -\log_2 P(A),$$

where P stands for the underlying probability distribution.

Observation. A dual point of view of the definition is that $I(A)$ is the necessary amount of information needed to specify event A.

Definition. Let X be a discrete random variable which takes finitely many distinct values $\Omega = \{x_1, \ldots, x_N\}$ with probability $p = (p_1, \ldots, p_m)$. The *(Shanon) entropy* of X is the expected amount of information gained when learning the value of X

$$H(X) = -\sum_{i=1}^{N} p_i \log_2 p_i.$$

Observation. The entropy depends on the probability distribution p, but not on the values of X.

Notation. If some of the $p_i = 0$ we take the convention $p_i \log_2 p_i = 0$ (by continuity on $\lim_{x \to 0^+} -x \log_2 x = 0$).

Definition. Let X be a random variable with a probability density function $p(x)$. The *differential entropy* is defined as

$$H(X) = -\int p(x) \log_2 p(x) dx$$

Observation. Unlike the entropy of a discrete random variable, the differential entropy may take negative values. Both entropies are called Shannon entropies.

Definition. Let X, Y be random variables taking values in the sets $\Omega_X = \{x_1, ..., x_N\}$ and $\Omega_Y = \{y_1, ..., y_M\}$.
The *joint entropy* $H(X, Y)$ is defined as

$$H(X, Y) = -\sum_{(x_i, y_j) \in (X,Y)} p_{X,Y}(x_i, y_j) \log_2 p_{X,Y}(x_i, y_j).$$

Definition. The *conditional entropy* $H(X|Y)$ of X, given Y, is defined as

$$H(X|Y) = -\sum_{x_i, y_j} p_{X,Y}(x_i, y_j) \log_2 p_{X|Y}(x_i, y_j).$$

Definition. If X, Y take values on the same set Ω and the marginal probability $p_Y(x) = P(Y = x) > 0 \ \forall x$, then the *relative entropy* $H(X||Y)$ can be defined:

$$H(X||Y) = -\sum_{x \in \Omega} p_X(x) \log_2 \frac{p_X(x)}{p_Y(x)}$$

Definition. The *mutual information* between X and Y is defined as

$$I(X : Y) = \sum_{x,y} p_{X,Y}(x, y) \log_2 \frac{p_{X,Y}(x, y)}{p_X(x) p_Y(y)} = H(X) + H(Y) - H(X, Y)$$

5.2. Markov chains

Definition. Let Ω be a finite set of states. A sequence $\{X_0, X_1, ...\}$ of random variables taking values in Ω is said to be a *Markov chain* if it satisfies the Markov condition:

$$\mathbb{P}(X_n = i_n | X_0 = i_0, ..., X_{n-1} = i_{n-1}) = \mathbb{P}(X_n = i_n | X_{n-1} = i_{n-1})$$

for all $n \geq 1$ and all $i_0, i_1, ..., i_n \in \Omega$.

Definition. The Markov chain is said *homogeneous* if, for all $n \geq 1$ and all $i, j \in \Omega$,

$$\mathbb{P}(X_n = j | X_{n-1} = i) = \mathbb{P}(X_1 = j | X_0 = i)$$

Markov chains will always be assumed homogeneous unless otherwise specified.

Definition. In this case, the *transition matrix* Q of the chain is defined as the $|\Omega| \times |\Omega|$ matrix of all transition probabilities $Q(i, j) = q_{i \to j} = \mathbb{P}(X_1 = j | X_0 = i)$.

Proposition. The matrix Q^n gives the law of X_n for the chain starting at $X_0 = i$

$$\mathbb{P}(X_n = j | X_0 = i) = Q^n(i, j)$$

Definition. We say that a Markov chain is *irreducible* or primitive if for all $i, j \in \Omega$ there exists $n \geq 0$ such that $Q^n(i, j) > 0$.

This definition means that if the chain starts at $X_0 = i$, then for any j there exists an integer n such that $\mathbb{P}(X_n = j | X_0 = i) > 0$. In other words, all the states can be connected through the chain.

Observation. It does not necessarily happen that there is one n such that $Q^n(j,i) > 0$ for all j. For instance, there can be two states a and b such that: $Q(a,b) = Q(b,a) = 1$ and $Q(a,a) = Q(b,b) = 0$. Then the chain just goes back and forth between a and b and in consequence $Q^n(a,a) = Q^n(b,b) = 0$ for all odd n and $Q^n(a,b) = Q^n(b,a) = 0$ for all even n.

Definition. The *period* of a state i is the greatest common divisor of all the n such that $Q^n(i,i) > 0$.

Then, it follows easily that for any irreducible Markov chain, all states have the same period d, and there is a decomposition $\Omega = S_1 \cup ... \cup S_d$ such that the chain Q takes S_k to S_{k+1} with probability 1.

Definition. We say that a Markov chain is *aperiodic* if for all $i \in \Omega$, the greatest common divisor of the $n \geq 1$ such $Q^n(i,i) > 0$ is 1.

Observation. Since Ω is finite, if the Markov chain is irreducible and aperiodic, there exists an n_0 such that $\forall n \geq n_0$ and $\forall i, j \in \Omega$ $Q^n(i,j) > 0$.

Definition. A *probability distribution* Π *on* Ω is an equilibrium (or steady-state) probability distribution of Q iff

$$\forall j \in \Omega \quad \sum_{i \in \Omega} \Pi(i) Q(i,j) = \Pi(j)$$

Theorem 9. Let Q be a finite set of states. If Q is irreducible, then there exists a unique equilibrium probability distribution Π for Q. If, moreover, Q is aperiodic, then

$$\forall i, j \in \Omega, \quad \lim_{n \to +\infty} Q^n(i,j) = \Pi(j)$$

5.3. Signal's boundaries and kurtosis

Definition. If $p(x)$ is a probability density on the real line with mean $\overline{x}$, then the *kurtosis* is its normalised fourth moment

$$k(p) = \frac{\int_{\mathbb{R}} (x - \overline{x})^4 p(x) dx}{(\int_{\mathbb{R}} (x - \overline{x})^2 p(x) dx)^2} = \frac{\mathbb{E}\left((X - \overline{x})^4\right)}{(\mathbb{E}\left((X - \overline{x})^2\right))^2}$$

where X is a random variable with probability density p.

Example.

- *Normal distribution:* Let $p(x)dx$ be the normal distribution of mean $\overline{x}$ and standard deviation σ:

  $$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\overline{x})^2/2\sigma^2}$$

  Then by change of variable $y = (x - \overline{x})/\sigma$, we get

  $$k = \frac{\int_{\mathbb{R}} (x - \overline{x})^4 p(x) dx}{(\int_{\mathbb{R}} (x - \overline{x})^2 p(x) dx)^2} = \frac{2\pi\sigma^2\sigma^5}{\sigma\sqrt{2\pi}\sigma^6} \frac{\int y^4 e^{-y^2/2} dy}{\int y^2 e^{-y^2/2} dy}$$

Now, by integration by parts, we have

$$\int y^4 e^{-y^2/2} dy = 3 \int y^2 e^{-y^2/2} dy = 3\sqrt{2\pi}$$

and so finally,

$$k(\text{any normal distribution}) = 3$$

- *Uniform distribution:* Let $p(x)dx$ be the uniform distribution on $[-\frac{a}{2}, \frac{a}{2}]$ (i.e., $p(x) = \frac{1}{a}$ if $x \in [\frac{a}{2}, \frac{a}{2}]$, and 0 else). Then we find

$$k(\text{any uniform distribution}) = \frac{\int_{\mathbb{R}} (x-\overline{x})^4 p(x)dx}{(\int_{\mathbb{R}} (x-\overline{x})^2 p(x)dx)^2} = \frac{a^2}{a} \frac{\int_{-\frac{a}{2}}^{\frac{a}{2}} x^4 dx}{\left(\int_{-\frac{a}{2}}^{\frac{a}{2}} x^2 dx\right)} = \frac{a\frac{2}{5}(\frac{a}{2})^5}{\left(\frac{2}{3}(\frac{a}{2})^3\right)^2} = \frac{9}{5}$$

- *Double exponential distribution:* Let $p(x)dx$ be the Laplace (or double exponential) distribution

$$p(x) = \frac{1}{Z} e^{-|x|^\alpha}$$

Then changing variables to $y = x^\alpha$, we find

$$\int_{\mathbb{R}} x^{2n} e^{-|x|^\alpha} dx = \frac{2}{\alpha} \int_0^\infty y^{\frac{2n+1}{\alpha}-1} e^{-y} dy = \frac{2}{\alpha} \Gamma\left(\frac{2n+1}{\alpha}\right)$$

This shows that the normalizing constant Z equals $\frac{2}{\alpha}\Gamma\left(\frac{1}{\alpha}\right)$ and that the kurtosis is

$$k(\text{Laplace distributions}) = \frac{\Gamma(5/\alpha)\Gamma(1/\alpha)}{\Gamma(3/\alpha)^2}$$

which is 6 for $\alpha = 1$, $25'2$ for $\alpha = 0'5$, and goes to infinity as α goes to 0.

In probability theory and statistics, the kurtosis is a measure of the peakedness of the probability distribution of a real-valued random variable, although some sources are insistent that heavy tails, and not peakedness, is what is really being measured by kurtosis. Higher kurtosis means more of the variance is the result of infrequent extreme deviations, as opposed to frequent modestly sized deviations. To put it easy, distribution with large tails and peaks at their mean but small shoulders have high kurtosis while distributions with negligible tails, big shoulders, and flat around their mean (or even small at their mean) have small kurtosis.

Kurtosis is used to characterise the discontinuous behaviour due to the following reasoning. If we start with a random variable X that measures the change in some quantity from one measurement to the next, so $X_i = f((i+1)\Delta t) - f(i\Delta t)$ where f is the measure of something. In any measurement, one could measure them more often with a new $\Delta t' = \Delta t/n$. So we have changes over smaller intervals $Y_j = f((j+1)\Delta t') - f(j\Delta t')$ and clearly

$$X_i = Y_{ni} + Y_{ni+1} + ... + Y_{n(i+1)-1}$$

Now if the process is stationary, it is reasonable to assume that all the X_i, Y_j are independent and identically distributed (iid). Hence, we have a decomposition of X_i into the sum of n iid variables Y_j, obtained simply by refining the sampling rate.

Proposition. Let X be a real-valued random variable and assume

$$X = Y_1 + ... + Y_n$$

where Y_i are iid samples of the same random variable Y. Then,

(1) $\overline{Y} = \frac{\overline{X}}{n}$
(2) $\sigma^2(Y) = \frac{\sigma^2(X)}{n}$
(3) $k(Y) - 3 = n(k(X) - 3)$

Observation. We can see here that if the kurtosis of the distribution of the long time interval X_i is larger than 3, then the kurtosis of the short time interval Y_j is even bigger and goes to infinity with n.

Theorem 10. Let X be a random variable with mean 0 and variance σ^2, and assume $b = \mathbb{E}(X^4) - 3\sigma^2 > 0$. Choose μ large enough so that $\mu \geq 8\sigma$ and

$$\mathbb{E}(X^4 I_{(\mu,\infty)}) \leq \frac{b}{64}$$

Then if, for any n, the random variable X can be written as a sum $X = Y_1 + ... + Y_n$ of iid Y_i, the following holds:

$$\mathbb{P}\left(\max_i |Y_i| \geq \sqrt{\frac{b}{2\sigma^2}}\right) \geq \frac{b}{16\mu^4}$$

In other words, this results says that if the kurtosis is larger than 3 and if the changes can be arbitrarily finely subdivided into independent small changes, then in the limit there is a positive probability of jumps in the measured quantity.

So, for example, we can use this theorem as an application to know where the phonemes of an speech signal change.

5.4. Hidden Markov Model

Definition. An *N-state Markov Model* is completely defined by a set of N states forming a finite state machine, and an $N \times N$ stochastic matrix defining transitions between states, whose elements $a_{ij} = P(\text{state } j \text{ at time } t \mid \text{state } i \text{ at time } t-1)$ are the *transition probabilities*.

Definition. With a Hidden Markov Model, each state additionally has associated with it a probability density function $b_j(\theta_t)$ which determines the probability that state j emits a particular observation θ_t at time t (the model is hidden because any state could have emitted the current observation). $b_j(\theta_t)$ is known as the *observation probability*.

Observation. The probability density function can be continuous or discrete. In the speech data b_j is usually a Gaussian distribution with a diagonal covariance matrix.

Such a model can only generate an observation sequence $\theta = \theta_0, \theta_1, ..., \theta_{T-1}$ via a state sequence of length T, as a state only emits one observation at each time t. The set of all such state sequences can be represented as routes through the state-time trellis shown in Figure 1. The $(j,t)-$th node corresponds to the hypothesis that observation O_t was generated by state j. Two nodes $(i, t-1)$ and (j,t) are connected if and only if $a_{ij} > 0$.

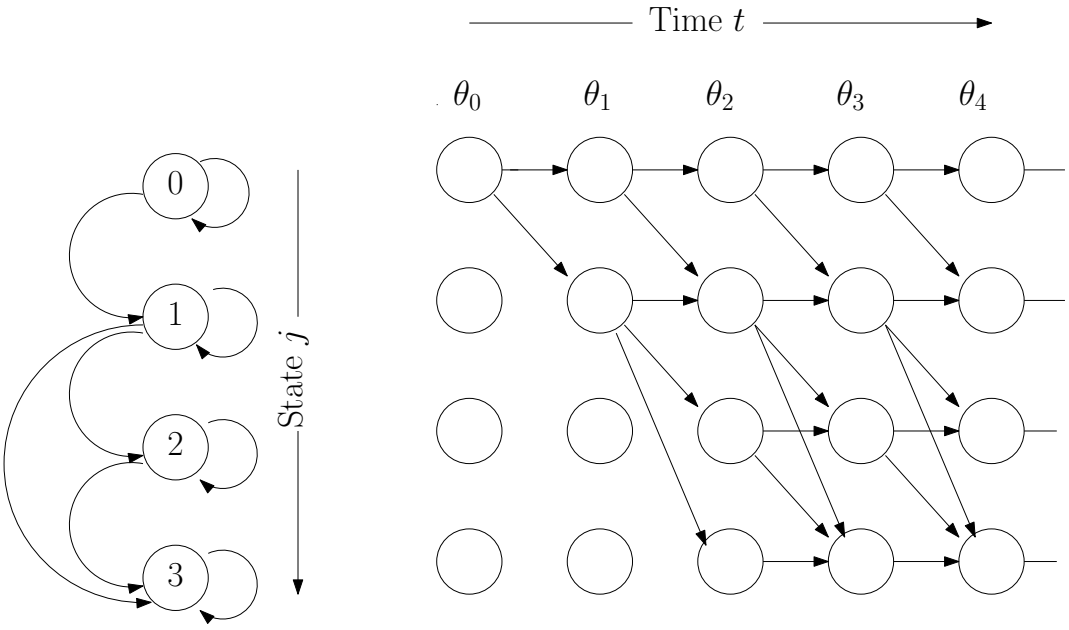


FIG. 1. Hidden Markov Model, showing the finite state machine for the HMM (left), the observation sequence (top), and all the possible routes through the trellis (arrowed lines)

5.5. Speech recognition

A speech recogniser converts an input audio waveform into a sequence of fixed size acoustic vectors $\mathbf{Y_{1:N}} = (\mathbf{y_1}, ..., \mathbf{y_N})$ in a process called feature extraction. The decoder then attempts to find the sequence of words $\mathbf{W}_{1:M} = (w_1, ..., w_M)$ which is most likely to have generated $\mathbf{Y}$. So, the decoder tries to find

$$\hat{\mathbf{W}} = \arg\max\{P(\mathbf{W}|\mathbf{Y})\}$$

Since it is difficult to calculate directly, Bayes's Rules is used to transform to

$$\hat{\mathbf{W}} = \arg\max\{P(\mathbf{Y}|\mathbf{W})P(\mathbf{W})\} \qquad (*)$$

As each spoken word w is decomposed into a sequence of K_w basic sounds called *basic phonemes*, it has to take into account the possibility of multiple pronunciations. So, let $\mathbf{q}^{(w)} = q_1, ..., q_{K_w}$ be a

pronunciation of the word w. Thus, let the likelihood $P(\mathbf{Y}|\mathbf{W})$ as

$$P(\mathbf{Y}|\mathbf{W}) = \sum_{\mathbf{Q}} P(\mathbf{Y}|\mathbf{Q})P(\mathbf{Q}|\mathbf{W})$$

where the summation is over all valid pronunciation sequences for $\mathbf{W}$ and $\mathbf{Q} = \mathbf{q}^{(w_1)}, ..., \mathbf{q}^{(w_M)}$ is a particular sequence of pronunciations. So,

$$P(\mathbf{Q}|\mathbf{W}) = \prod_{l=1}^{M} P(\mathbf{q}^{(w_l)}|w_l)$$

where each $\mathbf{q}^{(w_l)}$ is a valid pronunciation for word w_l.

Each base phone q is represented by a continuous density HMM of the form illustrated before. Given the composite HMM $\mathbf{Q}$ formed by concatenating all of the constituent base phones $\mathbf{q}^{(w_1)}, ..., \mathbf{q}^{(w_M)}$ then the acoustic likelihood is given by

$$P(\mathbf{Y}|\mathbf{Q}) = \sum_{\theta} P(\theta, \mathbf{Y}|\mathbf{Q})$$

where $\theta = \theta_0, ..., \theta_{T+1}$ is a state sequence through the composite model and

$$P(\theta, \mathbf{Y}|\mathbf{Q}) = a_{\theta_0,\theta_1} \prod_{t=1}^{T} b_{\theta_t}(y_t) a_{\theta_t\theta_{t+1}}$$

The acustic model parameters $\lambda = [\{a_{ij}\}, \{b_j\}]$ can be efficiently estimated from a corpus of training utterances using the forward-backward algorithm which is an example of expectation-maximisation (EM)[**15**].

Observation. The major problem with decomposing each vocabulary word into a sequence of context-independent base phones is that it fails to capture the variation that exists in real speech caused by the precedent and following phoneme. A way to mitigate this problem is to use triphonemes which is a unique phone model for every possible trio of phoneme and left and right neighbours.

5.6. N-gram approximation

The prior probability of word sequence $\mathbf{W} = w_1, ..., w_K$ required in $(*)$ is given by

$$P(\mathbf{W}) = \prod_{k=1}^{K} P(w_k|w_1, ..., w_{k-1})$$

For large vocabulary recognition, the previous equation is truncated to $N - 1$ words to form an N-gram language model

$$P(\mathbf{W}) = \prod_{k=1}^{K} P(w_k|w_{k-N+1}, ..., w_{k-1})$$

where N is typically in the range $2 - 4$.

Definition. Given a finite alphabet S (in our case the set of all words), a *language on* S is a set of probability distributions P_n on the set of strings Ω_n of length n for all $n \geq 1$.

To characterise a N-gram model we give three equivalent conditions

Proposition. Let Ω_N be the space of strings $a_1, ..., a_N$ of length N, a_i's being taken from a finite alphabet S. Consider a probability distribution $P : \Omega_N \to \mathbb{R}^+$. Fix an integer $n \geq 1$. The following conditions are equivalent:

(1) *Conditional factorisation*. P has the form

$$P(a_1, ..., a_N) = P_0(a_1, ..., a_{n-1}) \prod_{k=0}^{N-n} P_1(a_{k+n}|a_{k+1}, ..., a_{k+n-1})$$

for suitable P_0 and P_1.

(2) *Exponential form*.

$$P(a_1, ..., a_N) = \frac{1}{Z} \exp\left(- \sum_{\sigma \in \Omega_n} \lambda_\sigma \#\text{occ}(\sigma, a_1, ..., a_N) \right)$$

for suitable λ_σ's, where Ω_n is the set of strings of length n and $\#\text{occ}(\sigma, a_1, ..., a_N)$ denotes the number of occurrences of the string σ in $a_1, ...a_N$. Here we allow $\lambda_\sigma = \infty$ in case no strings of positive probability contain σ.

(3) *Markov property*. For all $I = (k+1, ..., k+n-1)$ of length $n-1$, let $a(I) = a_k + 1, ..., a_{k+n-1}$; then

$$P(a_1, ..., a_N|a(I)) = P_1(a_1, ..., a_k|a(I))P_2(a_k + n, ..., a_N|a(I))$$

which means that a (before I) and a (after I) are conditionally independent given $a(I)$.

Using the third equivalence, we can treat every I as an independent sequence and using Markov chains properties we can calculate the probabilities. The N-gram probabilities are estimated from training text by counting N-gram occurrences to form maximum likelihood (ML) parameter estimates. For example, let $C(w_k, w_{k-1}, w_{k-2})$ represent the number of occurrences of the words w_{k-2}, w_{k-1}, w_k, then

$$P(w_k|w_{k-2}, w_{k-1}) \simeq \frac{C(w_k, w_{k-1}, w_{k-2})}{C(w_{k-1}, w_{k-2})}$$

Chapter 6
Wallpaper groups

In this chapter we present the theory of wallpaper groups that we need in the next chapter. In our theoretical approach we follow [**14**]. First, we review the isometry group of the plane and its principal properties. After that, we define the wallpaper groups and their lattices. Next, we state the fact (theorem 13) that that there are exactly 17 different types of wallpaper groups under isomorphism [**19**]. Finally we explain the properties, summarized in Table 1, that single out each of these groups from the others.

6.1. Euclidean plane isometries

Definition. An *Euclidean plane isometry* is a map $M : \mathbb{R}^2 \to \mathbb{R}^2$ such that

$$d(p, q) = d(Mp, Mq)$$

for any two points $p, q \in \mathbb{R}^2$ where $d(p, q)$ is the Euclidean distance between them.

Surprisingly, there are only four types of isometries of the Euclidean plane, which together form a group under composition known as the *isometry group* of the plane and it is denoted by $\mathrm{Isom}(\mathbb{R}^2)$.

Definition. A *translation* T_v is a direct isometry of the plane that has not got any fixed points, except the identity case. In other words, it is a mapping from $\mathbb{R}^2$ to $\mathbb{R}^2$ defined by $T_v(p) = p + v$ where v is a vector of the plane.

It is elementary to see that if v, w are two vectors of the plane, then $T_v \circ T_w = T_{v+w}$ and $T_v^{-1} = T_{-v}$. Moreover, if 0 is the zero vector, then T_0 is the identity map. These facts show that the set of translations $\mathbb{T}(\mathbb{R}^2)$ forms a subgroup of $\mathrm{Isom}(\mathbb{R}^2)$.

Definition. The *translation subgroup* of $\mathrm{Sym}(W)$ is defined as $\mathbb{T}(W) = \mathrm{Sym}(W) \cap \mathbb{T}(\mathbb{R}^2)$

Lemma. The group $\mathbb{R}^2$ is isomorphic to the subgroup of $\mathrm{Isom}(\mathbb{R}^2)$ consisting of all translations of $\mathbb{R}^2$ via the map $v \mapsto T_v$. Therefore, the translation subgroup of the symmetry group of a figure in $\mathbb{R}^2$ is isomorphic to a subgroup of $\mathbb{R}^2$.

Proof. We show that the map φ given by $\varphi(v) = T_v$ is an isomorphism. We have already pointed out that $T_v \circ T_w = T_{v+w}$ and $T_v^{-1} = T_{-v}$ for any $v, w \in \mathbb{R}^2$. Thus, φ is a homomorphism. It is subjective by the definition of a translation. It is injective, since if $\varphi(v) = Id$, then $T_v(x) = x$ for all $x \in \mathbb{R}^2$. However, since $T_v(x) = x + v$, this forces $v = 0$. Thus, φ is an isomorphism. The final statement is clear, since if T is the translation subgroup of some symmetry group, then $\varphi^{-1}(T)$ is a subgroup of $\mathbb{R}^2$ isomorphic to T. □

Definition. A *rotation*, $R_{c,\theta}$, is a direct isometry with a fixed point c called center. $R_{c,\theta}$ is a transformation that rotates all the points of the plane about the center c and angle θ.

In terms of coordinates, rotations are most easily expressed by breaking them up into two operations. First, a rotation around the origin is given by

$$R(\theta) \equiv R_{0,\theta}(p) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{bmatrix} p_x \\ p_y \end{bmatrix}$$

These matrices are the orthogonal matrices (i.e. each is a square matrix G whose transpose is its inverse, i.e. $GG^T = G^TG = I_2$) with determinant 1. They form the special orthogonal group SO_2.

A rotation around c can be accomplished by first translating c to the origin, then performing the rotation around the origin, and finally translating the origin back to c. That is,

$$R_{c,\theta} = T_c \circ R_{0,\theta} \circ T_{-c}$$

or in other words,

$$R_{c,\theta}(p) = c + R_{0,\theta}(p - c)$$

Alternatively, a rotation around the origin is performed, followed by a translation:

$$R_{c,\theta}(p) = R_{0,\theta}p + v$$

Definition. A *reflection* or mirror isometry F_L, where L is a line of the plane an it is called the reflection axis, is transformation of the plane defined by:

$$F_L(p) = \begin{cases} p & \text{if } p \in L \\ \tilde{p} & \text{if } p \notin L \text{ and } \tilde{p} \text{ verifies that the midpoint of the segment } \overline{p\tilde{p}} \text{ lies in } L \end{cases}$$

A reflection can be defined as the inverse isometries of the plane with fixed points.

If we take the direction of the reflection axis as an element of the base of $\mathbb{R}^2$ and its perpendicular directions as the second vector, the reflection can be written in matrix terms as

$$F_L(p) = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{bmatrix} p_x \\ p_y \end{bmatrix}$$

Definition. A *glide reflection* $G_{L,w}$ is a transformation of the plane that is the composition of a reflection F_L and the translation by a vector w parallel to the reflection axis L. That is,

$$G_{L,w} = T_w \circ F_L$$

or in other words,

$$G_{L,w}(p) = w + F_L(p)$$

(It is also true that $G_{L,w}(p) = F_L(p + w)$. We obtain the same result if we do the translation and the reflection in the opposite order.)

Theorem 11. Every isometry of the plane, other than the identity, is either a translation, a rotation, a reflection, or a glide-reflection.

This theorem is proved in [**21, 2, 20**].

6.2. Wallpaper group

Definition. Let W be a subset of $\mathbb{R}^2$. Then the *symmetry group* of W is the set defined by

$$\mathrm{Sym}(W) = \{\varphi \in \mathrm{Isom}(\mathbb{R}^2) : \varphi(W) = W\}$$

It is clear that $\mathrm{Sym}(W)$ is a subgroup of $\mathrm{Isom}(\mathbb{R}^2)$.

Definition. A *lattice* is a finitely generated subgroup of $\mathbb{R}^n$ for some n.

By the fundamental theorem for finitely generated Abelian groups, every lattice is a free Abelian group, and so is isomorphic to $\mathbb{Z}^r$ for some integer r. Therefore, a lattice has a basis as a $\mathbb{Z}$-module. The dimension of a lattice is the size of a basis. A two-dimensional lattice in $\mathbb{R}^2$ then has the form

$$T = \{nv + mw : n, m \in \mathbb{Z}\}$$

of $\mathbb{Z}$-linear combinations of v and w where v, w are linearly independent. The vectors v, w are called *primitive lattice vectors* of T.

Two simple properties of lattices are

(1) T contains a nonzero vector of minimal length
(2) T contains only finitely many vectors inside any circle.

Definition. The *principal cell* or simply cell of the underlying pair of translations T_u and T_v is the smallest parallelogram whose corners are points of the lattice.

Definition. A subset W of R^2 is a *wallpaper pattern* if the translation subgroup of the symmetry group $\mathrm{Sym}(W)$ is a two-dimensional lattice. The symmetry group of a wallpaper pattern is said to be a *wallpaper group* or a plane crystallographic group.

In other words, a wallpaper group is a type of topologically discrete group of isometries of the Euclidean plane that contains two linearly independent translations.

Because of the multitude of members of wallpaper groups, it is possible for most (except the trivial one) to have a subregion that is smaller than the principal cell that, under repeating application of the isometries of the group, can be used to generate the entire pattern.

Definition. Let Φ be a planar image and let W be the wallpaper group of that image. A *fundamental region* or generating region for a planar image Φ is a convex set $\Gamma \subset \mathbb{R}^2$ such that $\forall y \in \mathbb{R}^2\ \exists x \in \Gamma$ and a composition of isometries $f_1 \circ \cdots \circ f_n$ in W mapping x to y.

Obviously, a primitive cell is a fundamental region of a wallpaper group. And interesting property of some patterns is that same have generating regions that are in fact smaller than their principal cell.

The condition on linearly independent translations means that there exist linearly independent vectors v and w such that the group contains both T_v and T_w.

The purpose of this condition is to distinguish wallpaper groups from frieze groups, which possess a translation but not two linearly independent ones, and from 2-dimensional discrete point groups, which have no translations at all. In other words, wallpaper groups represent patterns that repeat themselves in two distinct directions, in contrast to frieze groups, which only repeat along a single axis.

The discreteness condition means that there is some positive real number ϵ such that, for every translation T_v in the group, the vector v has length at least ϵ (except of course in the case that v is the zero vector).

The purpose of this condition is to ensure that the group has a compact fundamental domain of nonzero finite area. Without this condition, we might have for example a group containing the translation T_x for every rational number x, which would not correspond to any reasonable wallpaper pattern.

There are five lattice or cell types in wallpaper groups. The lattice types has relation with the rotational symmetry of highest order and in the figure 1 there are shown all these types of lattices.

- In the cases of rotational symmetry of order 3 or 6, the cell consists of two equilateral triangles. This cell is called *hexagonal cell*.
- In the cases of rotational symmetry of order 4, the cell is a square (*square cell*).
- In the cases of reflection or glide reflection, but not both, the cell is a rectangle (*rectangular lattice*). Some special cases can be a square.
- In the cases of reflection combined with glide reflection, the cell is a rhombus (*rhombic lattice*); a special case is that it actually is a square.
- In the case of only rotational symmetry of order 2 or with no symmetry other than translations, the cell is in general a parallelogram (*parallelogrammatic lattice*), therefore the diagrams show a parallelogram. Obviously some special cases can be rectangles, rhombus, or squares.

One important and nontrivial consequence of the discreteness condition of the wallpaper groups applied to the lattice is that the group can only contain rotations of order 2, 3, 4, or 6; that is, every rotation in the group must be a rotation by π, $\frac{2\pi}{3}$, $\frac{\pi}{2}$, or $\frac{\pi}{3}$. This fact is known as the crystallographic restriction theorem, and can be generalised to higher-dimensional cases.

Theorem 12. *Crystallographic restriction theorem*: The rotational symmetries of a 2-dimensional lattice with the rotation axis perpendicular to the lattice plane are limited to 2-fold, 3-fold, 4-fold,

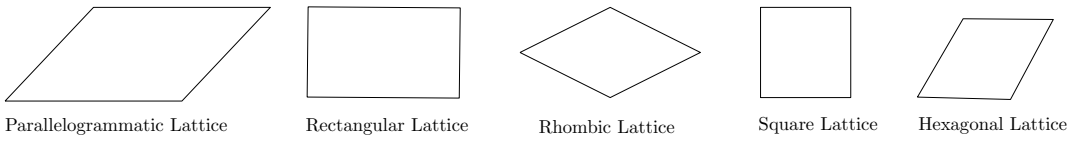


FIG. 1. The 5 lattice of the wallpaper groups.

and 6-fold. This theorem also holds for 3-dimensional lattices. Obviously, there are 1-fold points which are the points with no rotations center except for the identity.

Proof. Assume that a 2-dimensional lattice is invariant under rotation of angle θ. Let u be a primitive lattice vector. Rotate u by angle θ to generate u', and rotate u by angle $-\theta$ to generate u''. Both u' and u'' must be lattice vectors, and so is $u' + u''$, which points in either the same or the opposite direction as u. Since u is primitive (shortest along its direction), $u' + u'' = nu$, where n is an integer (positive, negative, or zero). Taking the dot product of this equation with u yields

$$2|u|^2 \cos\theta = n|u|^2$$

Thus, $\cos\theta = \{0, \pm 1/2, \pm 1\}$. The only possible values of θ are $\{0, \frac{\pi}{3}, \frac{\pi}{2}\}$ and their multiples.

In dimension 3, consider an arbitrary lattice vector v that is not parallel to the rotation axis. Rotate it by θ to generate v'; the difference $v' - v$ is a nonzero lattice vector perpendicular to the rotation axis. It follows that one can always find a line of lattice points perpendicular to the rotation axis; let u be the primitive (shortest) lattice vector for this 1-dimensional lattice. The rest of the proof can now proceed as in the 2-dimensional case. □

Observation. When the dimension of the lattice is four or higher, rotations need no longer be planar and so the 2-dimensional proof is no longer true. However, restrictions still apply, though more symmetries are possible. For example, the hypercubic lattice has an eightfold rotational symmetry, corresponding to an eightfold rotational symmetry of the hypercube.

Corollary. If a wallpaper group contains a 4-centre, then the group contains neither a 3-centre nor a 6-centre.

Proof. A rotation of order 3 and a rotation of order 4 cannot be in the same wallpaper group as the product of the first with the inverse of the second is a rotation of $\frac{\pi}{6}$, which is not possible by the previous theorem. □

6.3. Wallpaper group notations

In this subsection we present the two more standard wallpaper notations.

6.3.1. Crystallographic notation. Crystallography has 230 space groups to distinguish, far more than the 17 wallpaper groups, but many of the symmetries in the groups are the same.

First of all, we choose a basis $\{T_u, T_v\}$ for the translation lattice. We consider the direction of T_u to be the x-axis. The full name consists of four symbols. The first symbol represents the lattice

type; p for primitive and c for centered (or rhombic). The second symbol is the largest order of a rotation. The third symbol is either an m, g, or 1. An m (respectively g) means that there is a reflection line (respectively glide reflection line but not a reflection line) perpendicular to the x-axis while a 1 means there is no line of either type. Finally, the fourth symbol is also either an m, a g, or a 1. In this case an m (resp. g) represents a reflection line (resp. glide reflection line) at an angle α with the x-axis, the angle depending on the largest order of rotation as follows:

- $\alpha = \pi$ for $n = 1, 2$.
- $\alpha = \frac{\pi}{3}$ for $n = 3, 6$.
- $\alpha = \frac{\pi}{4}$ for $n = 4$.

For example, the group name $p3m1$ represents a group with a $\frac{2\pi}{3}$ rotation, a reflection line perpendicular to the x-axis, and no reflection or glide line at an angle of $\frac{\pi}{3}$ with the x-axis. However, in the group $p31m$, we have the same rotation, but no reflection or glide line perpendicular to the x-axis, while there is a reflection line at an angle of $\frac{\pi}{3}$ with the x-axis.

6.3.2. Orbifold notation. Orbifold notation or Conway notation for wallpaper groups, introduced by John Horton Conway, is based not on crystallography, but on topology. The notation takes the symmetries in the orbifold of the tiling. In the present work we do not talk about orbifolds.[1]

A digit, n, indicates a centre of n-fold rotation. By the crystallographic restriction theorem, n must be 2, 3, 4, or 6.

An asterisk, $*$, indicates a reflection. It interacts with the digits as follows:

- Digits before $*$ denote centres of pure rotation (cyclic).
- Digits after $*$ denote centres of rotation with reflection through them.

A cross, $\times$, occurs when a glide reflection is present. Reflections combine with lattice translation to produce glides, but those are already accounted for so we do not notate them.

The "no symmetry" symbol, $\circ$, stands alone, and indicates we have only lattice translations with no other symmetry.

Consider the group denoted in crystallographic notation by *cmm*; in Conway's notation, this will be $2*22$. The 2 before the $*$ says we have a 2-fold rotation centre with no reflection through it. The $*$ itself says we have a reflection. The first 2 after the $*$ says we have a 2-fold rotation centre on a reflection axis. The last 2 says we have an independent second 2-fold rotation centre on a reflection axis, one that is not an image of a center of the first kind by any symmentry.

The group denoted by *pgg* will be $22\times$. We have two pure 2-fold rotation centres, and a glide reflection axis. Contrast this with *pmg*, Conway $22*$, where crystallographic notation mentions a glide that it is implicit in the other symmetries.

In the table 1 there is the correspondence between the different wallpaper groups in both notations.

[1]To know more about orbifold see [**16**].

6.4. The 17 wallpaper groups

This subsection we state the well known theorem that there are exactly 17 different wallpaper group under isomorphism and we describe the distiguishing features of every group.

Theorem 13. There exactly 17 nonisomorphic wallpaper groups.

We do not prove this theorem here, but there is a beautiful and elementary proof, due to Schwarzenberger, that uses only basic trigonometry and group theory in [**19**].

In the following lines we present the properties of each group. At the end, table 1 summarises the 17 wallpaper groups and their properties and figure 2 displays the principal cells and the corresponding symmetries.

6.4.1. p1. This is the simplest symmetry group. It consists only of translations. There are neither reflections, glide-reflections, nor rotations. The two translation axes may be inclined at any angle to each other.

6.4.2. pm. This group contains reflections. The axes of reflection are parallel to one axis of translation and perpendicular to the other axis of translation. The lattice is rectangular. There are neither rotations nor glide reflections.

6.4.3. pg. This group contains glide reflections. The direction of the glide reflection is parallel to one axis of translation and perpendicular to the other axis of translation. There are neither rotations nor reflections.

6.4.4. cm. This group contains reflections and glide reflections with parallel axes. There are no rotations in this group. The translations may be inclined at any angle to each other, but the axes of the reflections bisect the angle formed by the translations, so the lattice is rhombic.

6.4.5. p2. This group differs only from $p1$ in that it contains π rotations, that is, rotations of order 2. As in all symmetry groups there are translations, but there are no reflections nor glide reflections. The two translations axes may be inclined at any angle to each other.

6.4.6. pgg. This group contains no reflections, but it has glide-reflections and π rotations. There are perpendicular axes for the glide reflections, and the rotation centers do not lie on the axes.

6.4.7. pmg. This group contains reflections, and glide reflections which are perpendicular to the reflection axes. It has rotations of order 2 on the glide axes, halfway between the reflection axes.

6.4.8. pmm. This symmetry group contains perpendicular axes of reflection, with π rotations where the axes intersect.

6.4.9. cmm. This group has perpendicular reflection axes, as does the group *pmm*, but it also has rotations of order 2. The centers of the rotations do not lie on the reflection axes.

6.4.10. p3. This is the simplest group that contains a $\frac{2\pi}{3}$-rotation, that is, a rotation of order 3. It has no reflections or glide reflections.

6.4.11. p31m. This group contains reflections (whose axes are inclined at $\frac{\pi}{3}$ to one another) and rotations of order 3. Some of the centers of rotation lie on the reflection axes, and some do not. There are some glide-reflections.

6.4.12. p3m1. This group is similar to the last in that it contains reflections and order-3 rotations. The axes of the reflections are again inclined at $\frac{\pi}{3}$ to one another, but for this group all of the centers of rotation do lie on the reflection axes. There are some glide-reflections.

6.4.13. p4. This group has a $\frac{pi}{2}$ rotation, that is, a rotation of order 4. It also has rotations of order 2. The centers of the order-2 rotations are midway between the centers of the order-4rotations. There are no reflections.

6.4.14. p4g. Like *p*4, this group contains reflections and rotations of orders 2 and 4. There are two perpendicular reflections axis passing through each order 2 rotation. However, the order 4 rotation centers do not lie on any reflection axis. There are four directions of glide reflection.

6.4.15. p4m. This group also has both order 2 and order 4 rotations. This group has four axes of reflection. The axes of reflection are inclined to each other by $\frac{\pi}{4}$ so that four axes of reflection pass through each order 4 rotation center. Every rotation center lies on some reflection axes. There are also two glide reflections passing through each order 2 rotation, with axes at $\frac{\pi}{4}$ to the reflection axes.

6.4.16. p6. This group contains $\frac{\pi}{3}$ rotations, that is, rotations of order 6. It also contains rotations of orders 2 and 3, but no reflections or glide-reflections.

6.4.17. p6m. This complex group has rotations of order 2, 3, and 6 as well as reflections. The axes of reflection meet at all the centers of rotation. At the centers of the order 6 rotations, six reflection axes meet and are inclined at $\frac{\pi}{6}$ to one another. There are some glide reflections.

Crist. notation	**Orbiforl notation**	**Lattice**	**Rotation Order**	**Reflection Order**
$p1$	$\circ$	parallelogram	1	none
pm	$**$	rectangle	1	parallel
pg	$\times\times$	rectangle	1	none
cm	$*\times$	rhombus	1	parallel
$p2$	2222	parallelogram	2	none
pgg	$22\times$	rectangle	2	none
pmg	$22*$	rectangle	2	parallel
pmm	$*2222$	rectangle	2	90°
cmm	$2*22$	rhombus	2	90°
$p3$	333	hexagonal	3	none
$p31m$	$3*3$	hexagonal	$3''$	60°
$p3m1$	$*333$	hexagonal	$3'$	60°
$p4$	442	square	4	none
$p4g$	$4*2$	square	$4''$	90°
$p4m$	$*442$	square	$4'$	45°
$p6$	632	hexagonal	6	none
$p6m$	$*632$	hexagonal	6	30°

TABLE 1. Properties of all 17 wallpaper groups.
$'$ = all rotation centers lie on reflection axes.
$''$ = not all rotation centers on reflection axes.

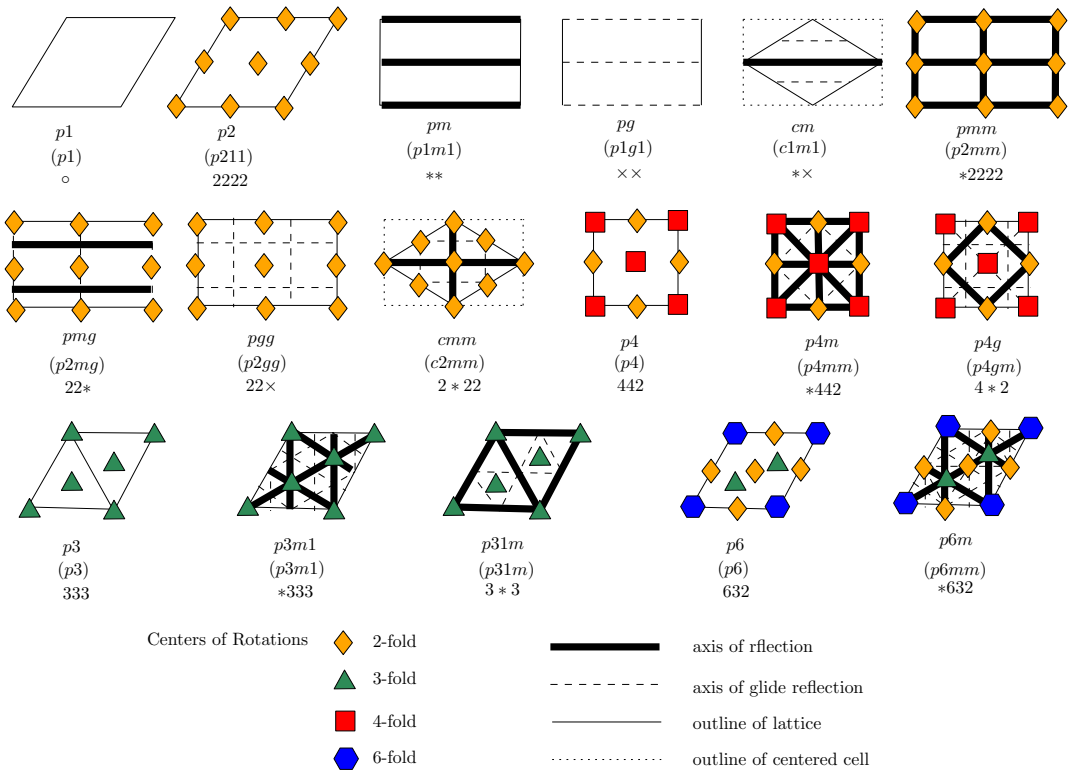


FIG. 2. The 17 wallpaper groups main cell with all their rotational centers, reflections and glide reflections.

Chapter 7
Wallpaper group classification

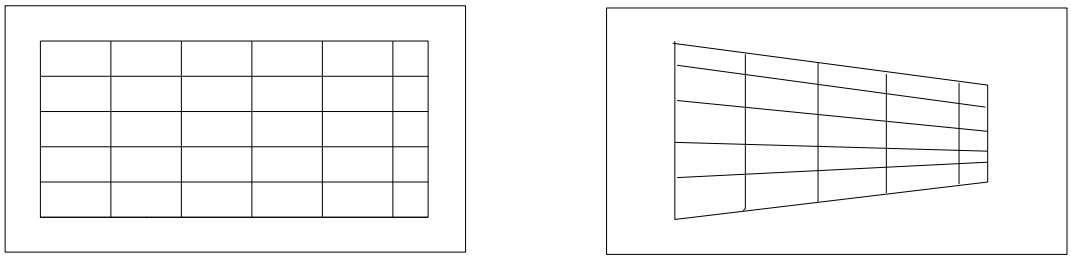
In this chapter we first outline an algorithm that finds the wallpaper group of a plane mosaic taking as input an image of it. Then we describe an implementation of the algorithm in MATLAB. We end with examples that illustrate how the algorithm performs for a sample of mosaic images. Our algorithm is based in [**9, 10, 11**].

7.1. Classification algorithm

The application assumes that a mosaic image is given as a parallel projection in the direction of its perpendicular lines, i.e, from a frontal perspective (see figure 1), and returns the type of wallpaper group of the mosaic.

Our computational algorithm for periodic pattern perception based on wallpaper groups includes two main components:

(1) recovering the underlying translation lattice of the pattern.
(2) classifying the wallpaper group of the image.



Frontal perspection Arbitrary perspective

FIG. 1. Image perspective.

The key issue in this algorithm to detect periodic patterns is the correlation function. We use it as a probability function since we find in every type of isometry the maximum value of the correlation function between the image and the transformation of the image due to the isometry. To make it a probability function we divide it by the maximum value of the autocorrelation of the original image.

Let $I(x, y)$ be an image and let φ be an isometry. To calculate the probability that the image I has this isometry we use the nexts steps.

(1) compute the transformed image $J(x, y) = \varphi(I(x, y))$.
(2) calculate the median of each image.
(3) substract the median to each images and obtain $\hat{I}(x, y)$ and $\hat{J}(x, y)$.
(4) compute the correlation function using the FFT by $r_{\hat{I}\hat{J}}(x, y) = \mathcal{F}^{-1}\{\mathcal{F}^*\{\hat{I}(x, y)\}\mathcal{F}\{\hat{J}(x, y)\}\}$
(5) calculate the maximum of the autocorrelation function of $\hat{I}(x, y)$ as $M_{\hat{I}\hat{I}} = r_{\hat{I}\hat{I}}(x, y) = r_{\hat{I}\hat{I}}(0, 0)$
(6) divide the correlation function by the previous number, i.e, $C_{\hat{I}\hat{J}}(x, y) = \frac{r_{\hat{I}\hat{J}}(x,y)}{M_{\hat{I}\hat{I}}}$
(7) search the maximum of the previous function.

In the next two sections we explain the two parts of our algorithm in more details.

7.2. Translational lattice detection

This part of the algorithm is centered in the processing of the image and the detection of the lattice. Although the most important part is the lattice detection.

The image processing starts with by a transformation of the image into a grey scale image, as it is easier to manipulate than a colour image. In some point of view is a compression technique. After that, the gray image is compressed by a 1-level wavelet transform using Haar wavelets. The advantage of this compression is to make uniform the different regions of the mosaic and emphsize the boundaries.

Once the image has been processed, the algorithm try to find the lattice of the wallpaper group. This search can be divided in three distinguish parts:

(1) Peak finding of the autocorrelation function of the image
(2) Region of dominance of every peak
(3) Calculate the main vectors of the lattice.

7.2.1. Peak Finding. The first step to determinate the lattice of the wallpaper group is to find the peak of the autocorrelation function. To do this in our case we first decompose the autocorrelation image in squares of n (typically 5) pixels length per side. After that, we calculate the maximum value of every square and keep the position of the peak inside the square.

Once the maximum of the square has been found, we look it the value of a square is bigger than the value of the square in its boundary. If the value is higher than the boundary ones, we save the

value and the position of it as a peak candidate. This method is used because in this way we avoid some noise in the autocorrelation function due to the noise in the original image.

7.2.2. Region of dominance. It is a nontrivial task to find a proper set of peaks in an autocorrelation surface of a periodic pattern. After some experimental proves our observation is that the absolute height of a peak is not as important as the size of its region of dominance, defined as the largest circle centered on the candidate peak such that no higher peaks are contained in the circle. A peak with a low height, but located far from any larger neighbours, is much more perceptually important than a high peak that is close to an even higher one.

So, given the peak candidate for the previous part of the algorithm, we ordered them for its region of dominance and keep only a percentage of them. At this form we erase some spurs and it is easier to calculate the lattice vectors.

7.7.2.2.1. *Determinate the lattice vectors.* Having a set of candidate lattice points extracted as dominant peaks in the autocorrelation surface, the next task is to find the shortest linearly independent translation vectors that generate the lattice.

To determinate the lattice vectors, we first find the peak nearer to the center of the autocorrelation function. Since this is always a peak we use it to find the vectors. After that, we calculate the 2 points nearer to that that the vectors are linearly independent. To avoid bad peaks we check that at double size there is another peak.

Once the lattice has been found, we determinate the lattice types calculating the angle between the vectors and the division between their lengths.

7.3. Classifying the wallpaper groups

This part of the algorithm calculates the probabilities of the main cell to have the different types of symmetries. After that, the wallpaper group inside the image is determined using this probabilities.

Once the lattice of the wallpaper group has been found, the first step to calculate the wallpaper group of the image is to apply an affine transform to the image to obtain an square main cell. An square main cell makes easier to calculate the value of correlations inside the image.

Although it would seem that in this form is would be more difficult to calculate symmetries due to the composition of the symmetries with the affine transform, almost every symmetry, except the rotations of order 3 and 6, has diagonal matrix and commutes with the affine matrix transform. So, in these two cases one has to be very careful to compute in the right order the matrix transformations.

The algorithm used to determine the wallpaper group is described in the following lines. The input of the algorithm is the lattice type computed (given by translation vectors T_1 and T_2) by measuring the angles between them, their lengths and an image of the main cell. In the following text we will use the notation T_1-reflection (T_2-reflection) referred to a reflection about an axis of the unit lattice parallel with T_1 (respectively, T_2).

Euclidean Algorithm:

(1) If the lattice is a parallelogram lattice (two possible groups: $p1$; $p2$), test 2-fold rotation. If the answer is yes W is $p2$ otherwise W is $p1$.
(2) If the lattice is a rectangle lattice there are five plus two (seven) possible groups: pm; pg; pmm; pmg; pgg and $p1$; $p2$. First test 2-fold rotation. If 2-fold rotation exists (four possible groups: pmm; pmg; pgg; $p2$). Test T_1-reflection and T_2-reflection. If neither is a glide reflection, W is pmm, if one of them is a glide reflection W is pmg, otherwise W is pgg. If no reflection exists, then W is $p2$. If 2-fold rotation does not exist (three possible groups: pm; pg; $p1$). Check T_1-reflection (or T_2-reflection). If there is a glide-reflection W is pg otherwise W is pm. If no reflection symmetry exists then it is $p1$.
(3) If the lattice is a rhombic lattice, test diagonal reflection symmetry. If it does not exist, go to step 1. Otherwise there are two possible groups: cm;cmm. Test 2-fold rotation. If the symmetry exists, W is a cmm, otherwise, it is a cm.
(4) If the lattice is a square lattice, test 4-fold rotation symmetry. If 4-fold symmetry does not exist, go to step 3. Otherwise there are three possible groups: $p4$; $p4m$; $p4g$. First, test T_1-reflection symmetry. If the symmetry exists W is a $p4m$ group. Otherwise, test diagonal-reflection, if the answer is yes W is a $p4g$. If it has no reflections then W is a $p4$.
(5) If the lattice is an hexagonal lattice, first test 3-fold rotation symmetry. If no 3-fold symmetry exists, go to step 3. Otherwise, there are five possible groups: $p3$; $p3m1$; $p31m$; $p6$; $p6m$, test 2-fold rotation. If 2-fold rotation exists (two possible groups: $p6$; $p6m$). Then test T_1-reflection. If there is a T_1-reflection exists W is $p6m$ otherwise W is a $p6$ group. In the case that there is no 2-fold rotation (three possible groups: $p3$; $p3m1$; $p31m$), test T_1-reflection. If the answer is no W is a $p3$ otherwise check diagonal-reflection. If the symmetry exists W is a $p3m1$ otherwise W is a $p31m$ group.

In short, given a lattice first test the possible existence of special symmetries (3-fold, 4-fold rotations, diagonal reflection); when the answer is yes, the search is limited in a small set of possible groups; when the answer is no, the classification process proceeds to a more general type of lattice.

To compute the different probabilities of the symmetries we use the process explained in 7.1.

7.4. Examples of the algorithm

The examples in this section serve to illustrate the symmetry classification algorithm. We present 3 different examples. The first two examples are computer generated mosaics and hence they have few imperfections and their symmetries are very clear. The last example is an Indian metalwork presented at the Great Exhibition in 1851. Since it is human made, it has some imperfections and those affect significantly the correlation values.

Figure 2 displays the mosaic choosen for the first example. The wallpaper group is a pmg and we will use our program to identify it.

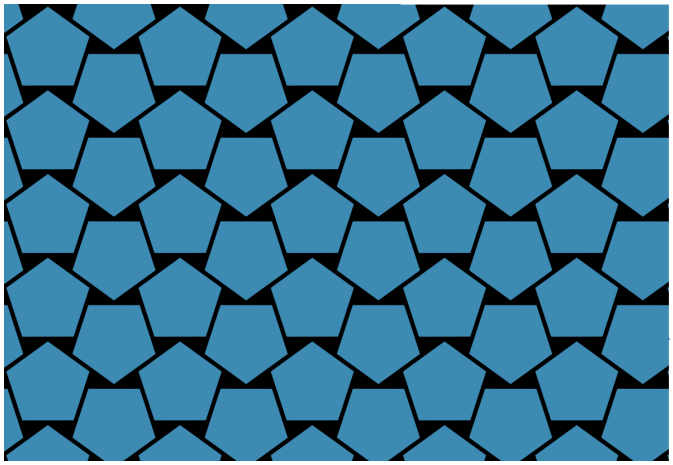


FIG. 2. A *png* wallpaper group made by computer.

First of all, the program transforms the image into grey scale one and compresses it with a 1-level wavelet transform. After that, it computes the autocorrelation function of the image and finds the peaks of the autocorrelation with the algorithm described in 7.2. Figure 3 shows the peaks found by the program.

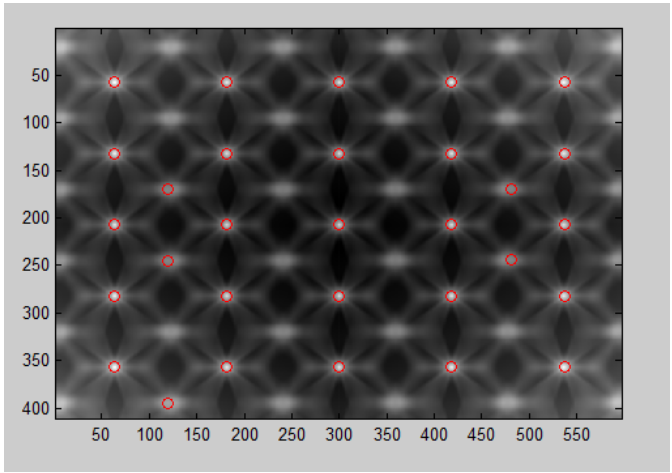


FIG. 3. The autocorrelation function of the figure 2 and the peaks found by the algorithm.

After that, we determine the main cell of the lattice. This can be seen in the figure 4. In the figure 5, the main cell with the 2-fold rotation point in the center is shown.

Once the lattice of the wallpaper group is found, and since the lattice type is a rectangular lattice, we test for a 2 rotation center. The program returns a value of $P(2-fold) = 0.9276$ In Table 1 all

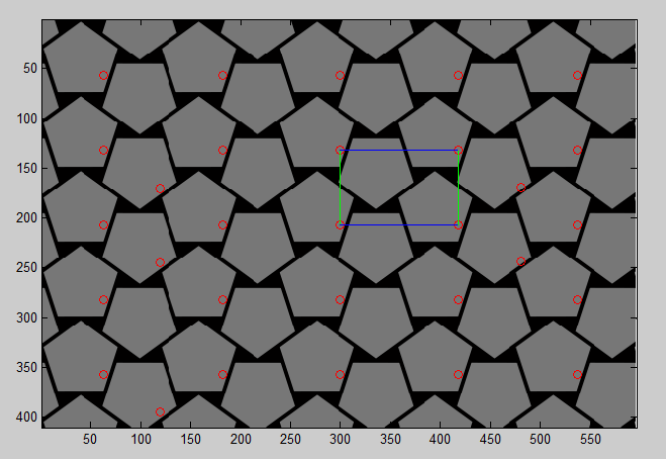


FIG. 4. The lattice of the figure 2 detected by the algorithm.

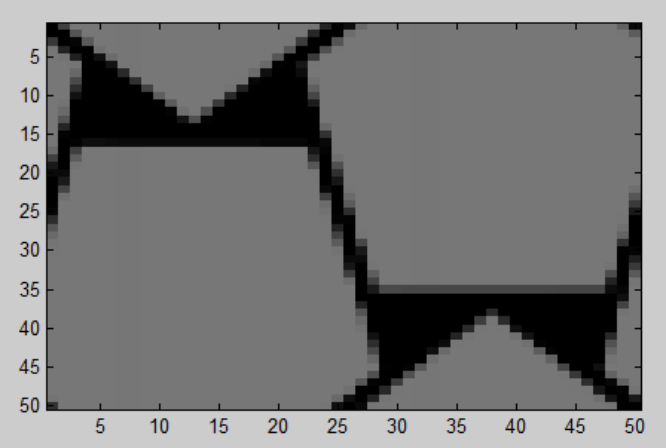


FIG. 5. The main cell of the mosaic in the figure 2 with a 2-fold rotation point in the center.

the symmetries are tested. So, following the algorithm described in 7.3, the only possible groups are $p2$, pmg, pmm and pgg. Now we test for a T_1 and T_2 reflections. To see the result see table 1. So, since there is one reflection and one glide reflection the wallpaper group is the pmg.

Figure 6 shows the mosaic choosen for the second example. The wallpaper group is a $p6m$, which is the wallpaper group with more symmetries and we will use our program to identify it. We notice that the image has some imperfections and due to that the probabilities are lower than the previous example. However, the good values are still higher than 0.5.

Symmetry	**Probability**
2 rotation	0.9276
3 rotation	0.1972
4 rotation	0.2205
6 rotation	0.2011
T_1 reflection	0.9720 with a glide of 25
T_2 reflection	0.8841 with a glide of 0
D_1 reflection	0.2221 with a glide of 32
D_2 reflection	0.2129 with a glide of 25

TABLE 1. Probabilities of the different symmetries of the figure 2. The size of the images after the affine transform is 50×50.

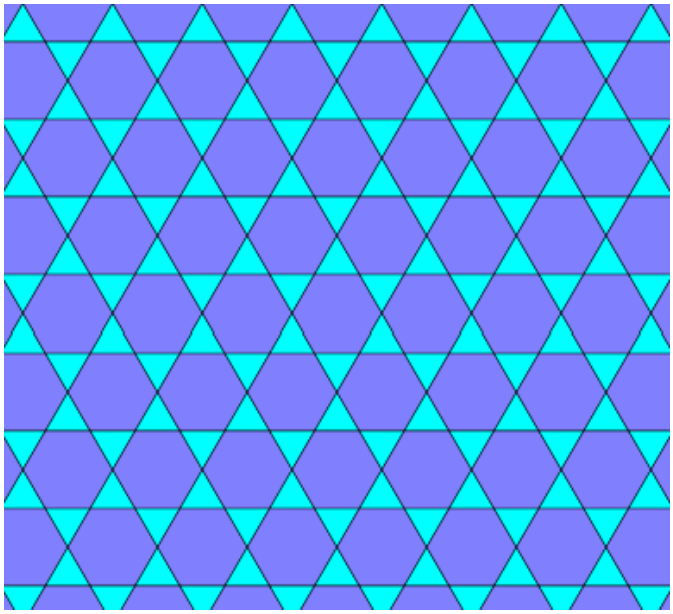


FIG. 6. A *p6m* wallpaper group made by computer.

As in the previous example, the program transforms the image into a grey scale image and compresses the image with a 1-level wavelet transform. After that, it computes the autocorrelation function of the image and finds the peaks of the autocorrelation with the algorithm described in 7.2. In the figure 7 can be seen the peaks found by the program.

After that, we determine the main cell of the lattice. This can be seen in the figure 8. In the figure 9 the main cell after the affine transformation is shown.

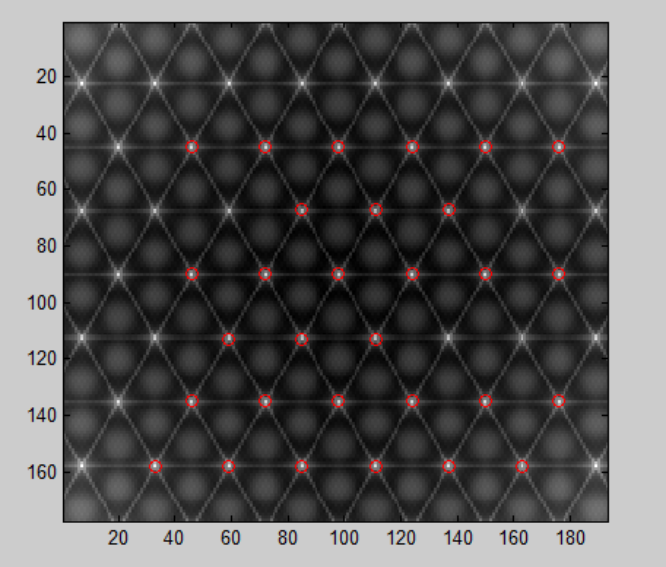


FIG. 7. The autocorrelation function of the figure 6 and the peaks found by the algorithm.

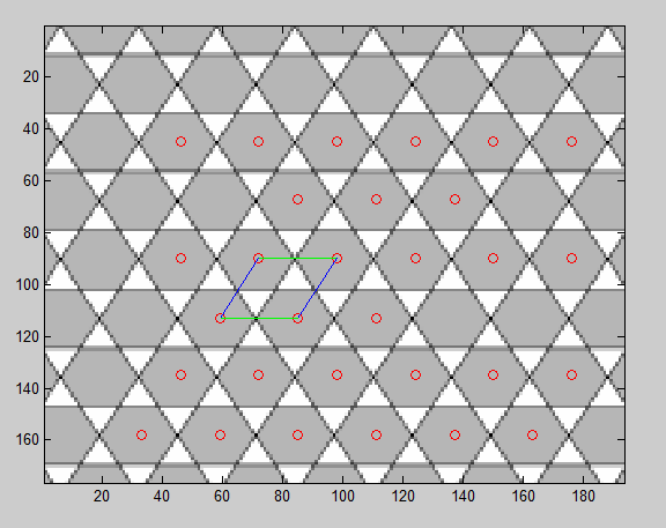


FIG. 8. The lattice of the figure 6 detected by the algorithm.

Once the lattice of the wallpaper group is found and since the lattice type is an hexagonal lattice, we search for 3 rotation centers. As the program fins that the image has 3 rotation center with a probability of 0.7133, we conclude that the image has 3-fold. Then we test the 2 rotation centers and we find one with a probability of 0.7284. So, the wallpaper group can only be $p6$ or $p6m$.

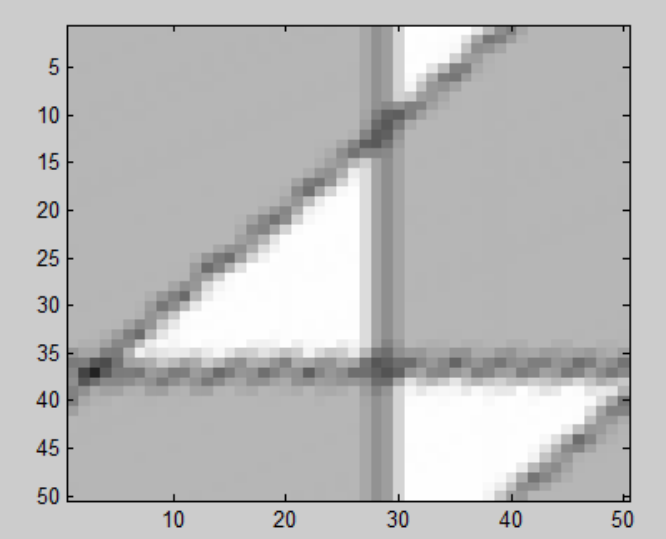


FIG. 9. The main cell of the mosaic in the figure 6 after the affine transformation.

Finally, we test for some symmetries and we find them. Thus, the group is a *p6m*. Table 2 lists all the probabilities.

Symmetry	**Probability**
2 rotation	0.7284
3 rotation	0.7133
4 rotation	0.2749
6 rotation	0.6455
T_1 reflection	0.9123 with a glide of 0
T_2 reflection	0.8970 with a glide of 25
D_1 reflection	0.9495 with a glide of 0
D_2 reflection	0.8867 with a glide of 25

TABLE 2. Probabilities of the different symmetries of the figure 6

Figure 10 shows the mosaic chosen for the last example. This mosaic is human made and due to that fact the probabilities found by the algorithm are lower, although they are higher than 0.5. The wallpaper group is a *cm*.

As in the previous examples, the program transforms the image into a grey scale one and compresses the image with a 1-level wavelet transform. As this image is human made, the compression afects it a little more. Although it is very difficult to see with a human eye, the program obtains

FIG. 10. Indian metalwork at the Great Exhibition in 1851 with a *cm* wallpaper group .

some improved probabilities for the compressed image. After that, it computes the autocorrelation function of the image and finds its peaks. Figure 11 shows the peaks found by the program.

After that, we determine the main cell of the lattice. This can be seen in the figure 12. In the figure 13 the main cell after the affine transformation is shown.

Once the lattice of the wallpaper group is found, and since the lattice type is a rhombic lattice, we search for diagonal rotation centers. As the program returns a value of 0.7518, the algorithm concludes that the wallpaper group can only be *cm* or *cmm*. Finally, the program searches for a 2 rotation center and as the returned value is 0.3495, it infers that the wallpaper group of the image is *cm*. Table 3 shows all the probabilities.

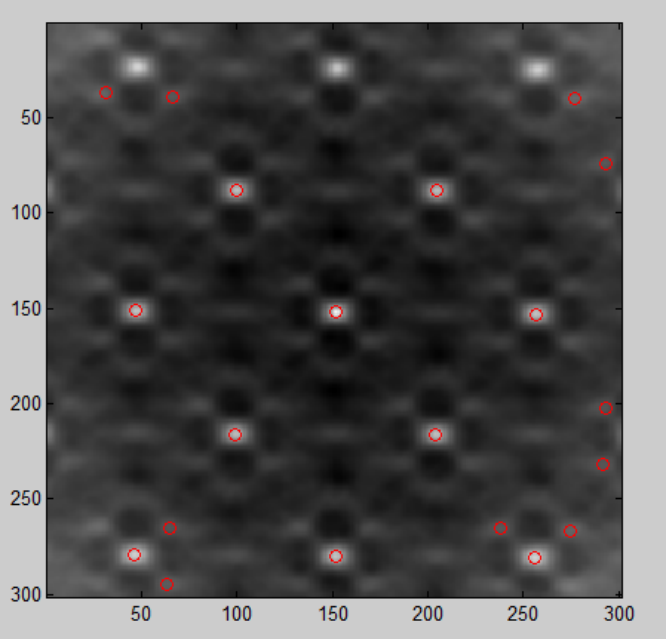


FIG. 11. The autocorrelation function of the figure 10 and the peaks found by the program.

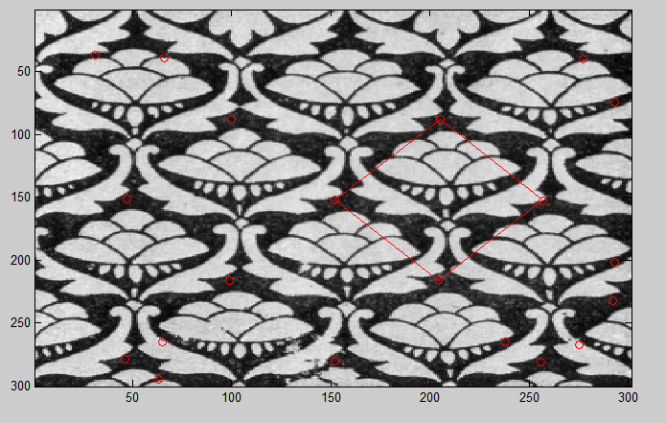


FIG. 12. The lattice of the figure 10 detected by the algorithm.

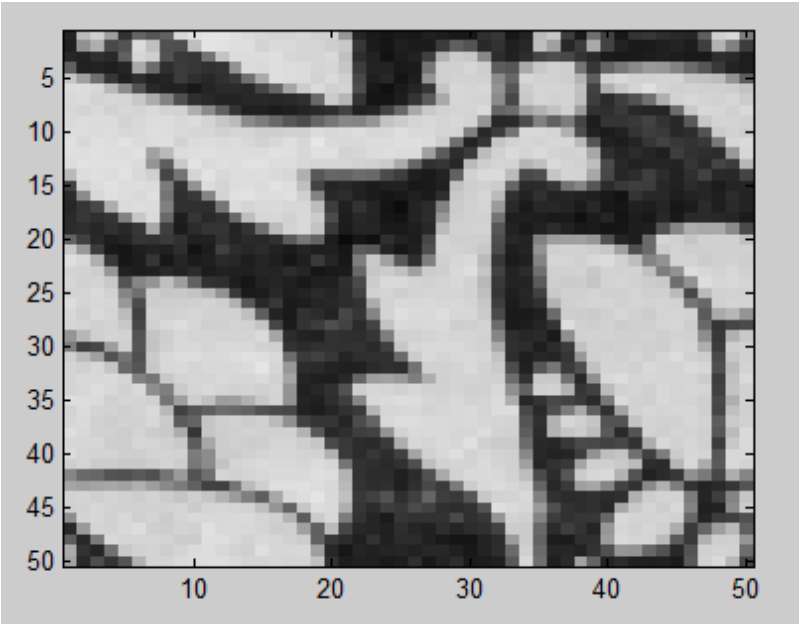


FIG. 13. The main cell of the mosaic in the figure 10 after the affine transformation.

Symmetry	**Probability**
2 rotation	0.3495
3 rotation	0.1865
4 rotation	0.2243
6 rotation	0.2365
T_1 reflection	0.2182 with a glide of 0
T_2 reflection	0.2335 with a glide of 0
D_1 reflection	0.3548 with a glide of 25
D_2 reflection	0.7518 with a glide of 25

TABLE 3. Probabilities of the different symmetries of the figure 10

Chapter 8
Speech classification

As we have indicated in Chapter 5, in this chapter we show a MATLAB application that parses the phonemes of a given recorded voice. In contrast to Chapter 5, we present a phoneme classification based on the wavelet transform. The DWT is interesting in the analysis of speech since it is easy to extract parameters which take into account the properties of the human hearing system. The analysis of the power in different frequency bands has the potential capability to distinguishing the beginning and ending of phonemes because the power in the different subbands is broadly constant over the lifetime of the phoneme [**4, 22, 23**].

The chapter has been divided in three parts. The first part deals with the speech segmentation method used in the algorithm. This process is illustrated with some examples. The second part addresses the problem of phoneme classification by means of a neuronal network. Finally, in the third part some further examples are presented.

8.1. Classification algorithm

The application consists in a program which returns, given a recording of a person saying a word, its sequence of phonemes. In our case, we use a data base of some Catalan words spoken by children.

The algorithm consist in two main components:

(1) *Segmentation*: it divides the spoken word into phonemes.
(2) *Identification*: classifies each phoneme using a neuronal network.

Both components use the properties of the local spectrum of the wavelet transform to determinate the energy of each subband. Segmentation determines a sequence of points that separate the phonemes and Identification uses the energies of the different subbands as features for classification.

In the next two sections we explain the two parts of our algorithm in more detail.

8.2. Speech segmentation

8.2.1. Segmentation algorithm. This part of the algorithm divides the spoken word into phonemes detecting their boundaries using DWT. The DWT is used due of its decomposition in frequency subbands. The analysis of the power in different frequency subbands gives an excellent opportunity to distinguish the beginning and the end of phonemes because many phonemes exhibit rapid changes in particular subbands which can determine their beginnings and end points. The start of a phoneme should be marked by an initially small but rapidly rising power level in one or more of the DWT levels. In other words, we should expect the power to be small and the derivative to be large.

Some experiments as in [**22**] showed that the speech signal should be decomposed into six levels, which cover the frequency band of a human voice. In our case we use a 6-level Symlets DWT.

Let s be the speech signal. First of all we calculate its M-DWT as $DWT(s) = \{\mathbf{d_M}, \mathbf{d_{M-1}}, ..., \mathbf{d_1}, \mathbf{c_1}\}$. As human voice has no power at a very low frequencies, we only take into account the d_j coefficients.

After that as the wavelet spectrum samples in n-level depends on the length N of speech signal in time domain according to $2^{-M+n-1}N$ where $n = 1, ..., M$ the power at each band is calculated in a different way to obtain the same number of samples following the next equation

$$p_n(i) = \sum_{j=1}^{2^{n-1}} d_n(j + 2^{n-1}i) \quad \text{where } i = 0, ..., 2^{-M}N - 1 \quad (*)$$

The DWT subband power shows rapid variations. So, the first order differences in the power are inevitably noisy. Due to that we calculate the envelopes e_{p_n} of each subband power function by choosing the highest values of p_n in a window of given size W to obtain a power envelope. See table 1.

DWT Level	**Number of samples in comparison with level** 1	**Window size**
1	1	3
2	2	3
3	4	3
4	8	5
5	16	5
6	32	5

TABLE 1. Windows size of each DWT level.

To calculate the derivate function of p_n we use a smoothed differencing operator. The subband power p_n is convolved with the mask $[1, 2, -2, -1]$ to obtain smoothed rate-of-change information r_n.

After that, we can detect phoneme boundaries searching for i-points for which the inequality

$$p \geq \left\| \beta \, \|r_n(i)\| - e_{p_n}(i) \right\|$$

holds, where constant p is a value of threshold which accounts for the time scale and sensitivity of the crossing points. In practice we take $p = 0.02$. In the inequality r_n is multiplied by a scaling factor β which is approximately equal to 1.

The algorithm is summarized in the following steps:

(1) Normalise a speech signal by dividing by its maximum value.
(2) Decompose a signal into 6-levels of the DWT.
(3) Calculate the sum of power samples in all frequency subbands to obtain the n-th subband power representation $p_n(i)$ according to $(*)$.
(4) Obtain the envelopes $e_{p_n}(i)$ for functions in each subband by choosing the highest value of p_n in a given window W. (See table 1).
(5) Calculate the rate-of-change function $r_n(i)$ by filtering $p_n(i)$ with $[1, 2, -2, -1]$ mask.
(6) Given a threshold p of the distance between $r_n(i)$ and $e_{p_n}(i)$ and a threshold $p_{\min}$ of minimal e_{p_n}, find indexes for which

$$\left(\left\| \|r_n(i)\| - e_{p_n}(i) \right\| < p\right) \text{AND} \left(\left(\left\| \|r_n(i+1)\| - e_{p_n}(i+1) \right\| > p\right) \text{OR} \left(\left\| \|r_n(i-1)\| - e_{p_n}(i-1) \right\| > p\right)\right)$$

$$\text{AND} \left(e_{p_n}(i) > pmin\right)$$

Write such indexes in one vector.
(7) Find and group indexes where there is no space between neighbouring ones longer than attribute Δ. This attribute is the minimum length of a phoneme which approximately is 0.028 seconds.
(8) Calculate an average index value for each group found in the previous step as points of phoneme change.

8.2.2. Examples. The examples in this section serve to illustrate the segmentation algorithm. We present 3 different examples. The first one the algorithm divides exactly the speech signal while in the second one the program divides the speech signal in more segments than it should. However, the true segmentation points are in the segmentation points. In the third one the segmentation points are a little different that the real ones.

In the figure 1 it can be seen the recorded waveform of a child saying the word *ONA*. In the figures 2 can be seen the power function, the envelope of it and the rate-of-change function. It can be also seen the points of phoneme change at each subband before the average step. Finally, in the figure 3 the recorded waveform is presented with some red lines that indicate the segmentation points found by the algorithm.

The second example can be seen in the figure 4. This example is a speech signal of a child saying the Catalan word *OCI*. The figure 5 shows the segmentation points found by the algorithm. As one can see seeing the figure, there are some segmentation lines that do not represent real segmentation points. However, the real points also appear in the segmentation lines.

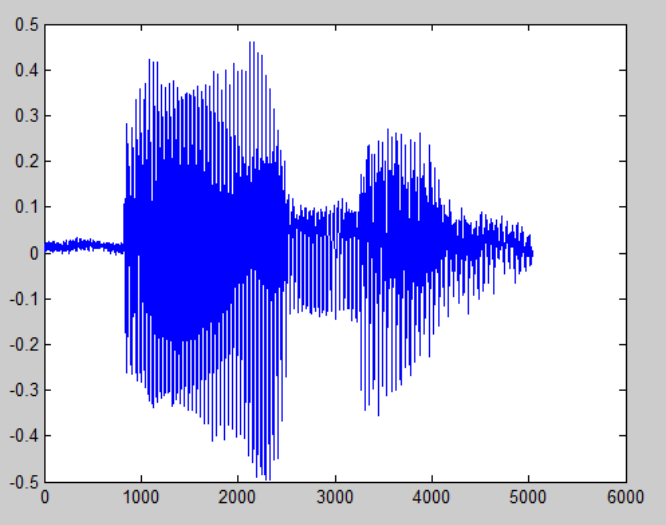


FIG. 1. The speech waveform of a child saying the word *ONA*.

The last example of this section is shown in the figure 6. In this example the word is *ENA*. In the figure 7, it is represented in red the segmentation lines found by the algorithm and in green the true segmentation lines that do not correspond a segmentation points found by the program. However, the real lines that the program do not found are closer to another found by the algorithm.

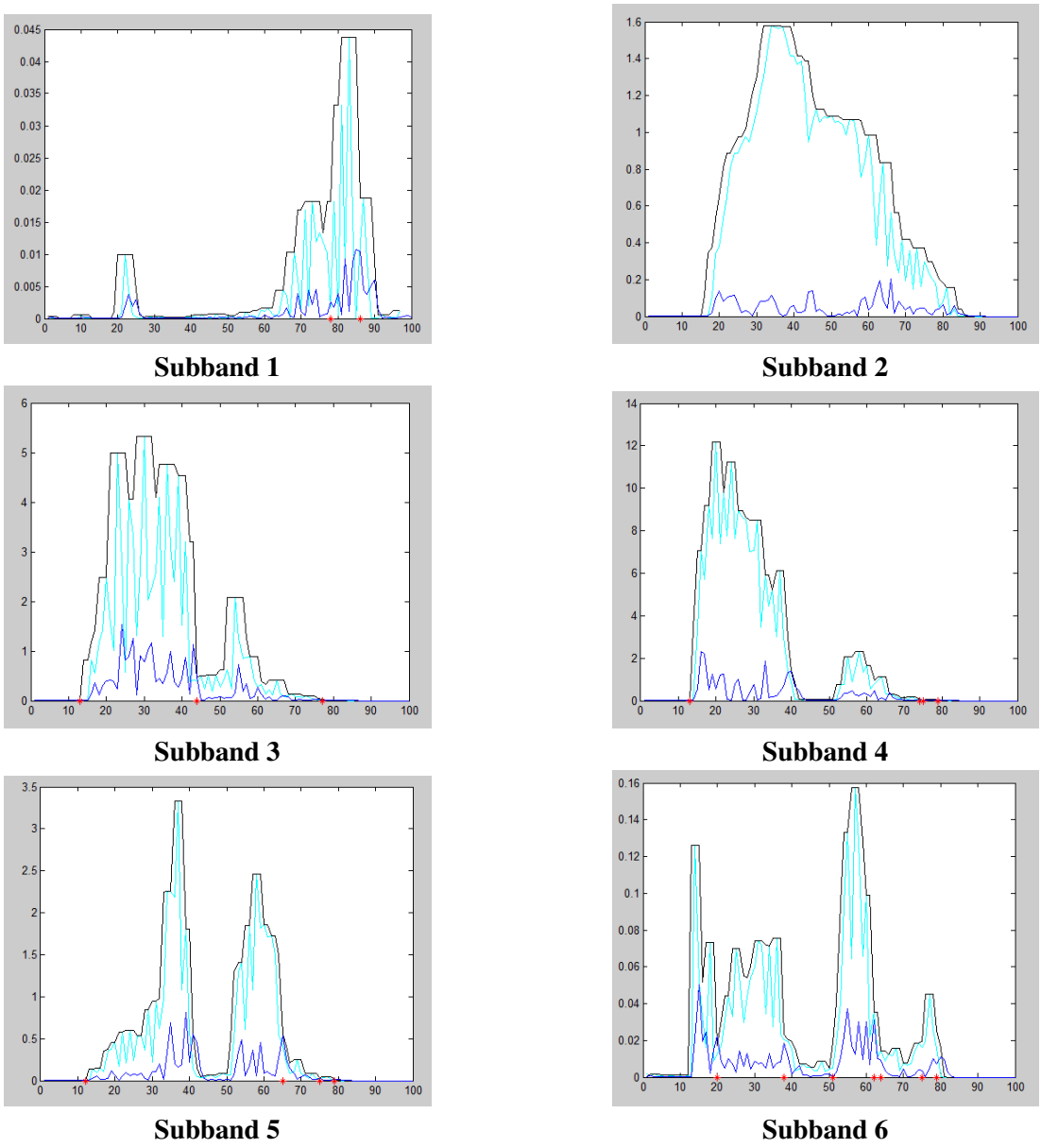


FIG. 2. In cyan the power function p_n of the n-th frequency subband. In black its envelope function e_{p_n} and in blue its rate-of-change function r_n. The asterisks in red are the points where could be a phoneme change.

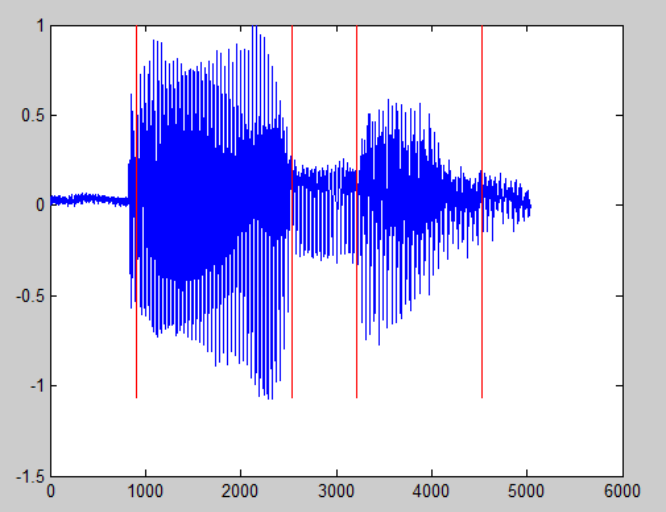


FIG. 3. The speech waveform of a child saying the word *ONA*. In red the segmentation lines found by the program.

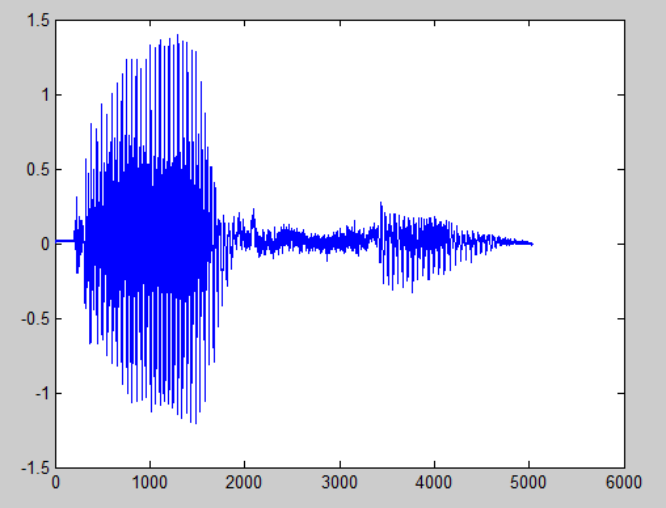


FIG. 4. The speech waveform of a child saying the word *OCI*.

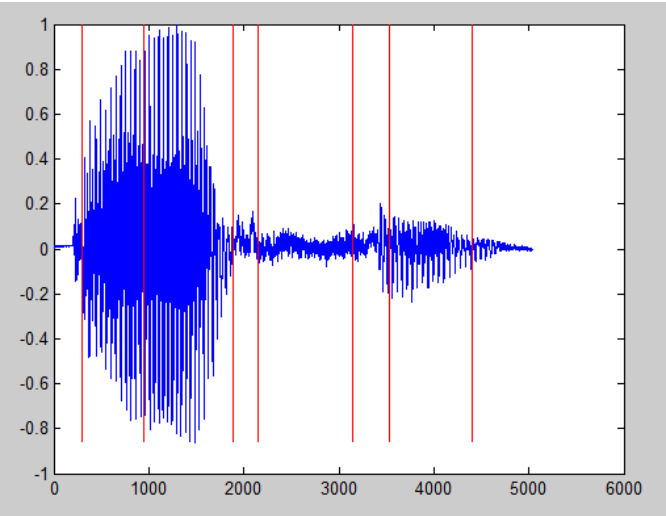


FIG. 5. The speech waveform of a child saying the word *OCI*. In red the segmentation lines found by the program.

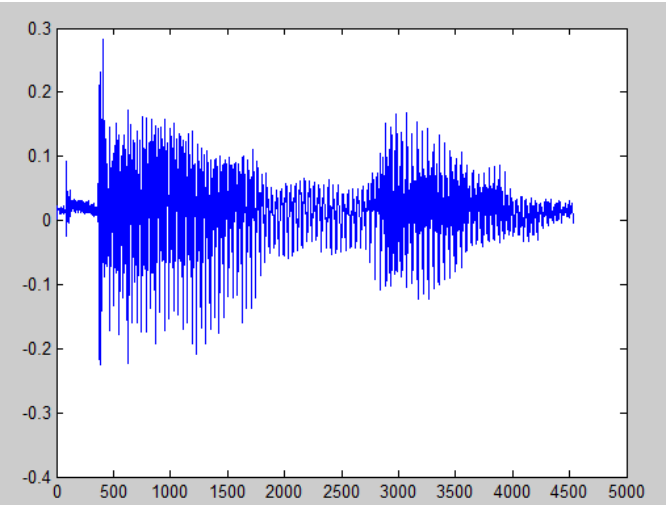


FIG. 6. The speech waveform of a child saying the Catalan word *ENA*.

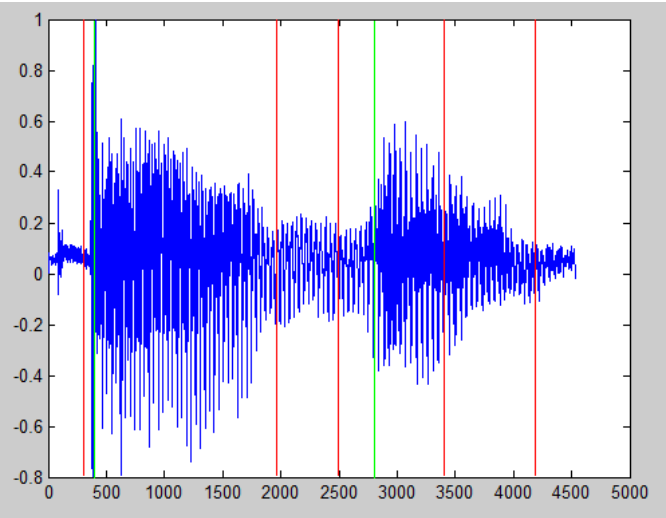


FIG. 7. The speech waveform of a child saying the word *ENA*. In red the segmentation lines found by the program and in green the segmentation point that the program has not found.

8.3. Phoneme recognition

This second part of the algorithm take each phoneme detected by the previous part and decide which phoneme is. To classify the phonemes it uses a Multi-Layer Perceptron (MLP):

To explain how the program works we have divide it in two distinguished parts:

- Determinate the characteristics (features) that represent each phoneme.
- Determinate the classification method.

8.3.1. Feature extraction. Once the speech signal has been segmented, we only keep the corresponding 20 ms samples in the center of the phoneme. This method is used because many time the segmentation program return a segmentation point a little different that the real one. Furthermore, in phoneme boundaries the signal is not stationary and due to that is not valid for classification.

After that, we use a Symlets DWT of order 4. This results in a distribution of the signal over different frequency bands (five bands for a four level decomposition as explained in 4.5). The energy of the signal component in each frequency band is calculated. This energy is normalised by the number of samples in the corresponding band, thereby giving an average energy per sample in each band. The normalisation is essential because each band will have a different number of samples. These average energies per sample for different bands are used as features for classification.

8.3.2. Classification. The performance of the extracted features is tested on a classifier using a MLP. A MLP consists of one or more hidden layers and an output layer. Usually a single hidden layer is used for classification purpose because it has less computational complexity and better stability during the training phase. However, for highly non-linear classification problems more hidden layers may be used to reduce the total number of neurones in the hidden layers. This also helps in reducing the problem of over-fitting that arises in MLP when the number of neurones in the hidden layer is large [**12**]. MLP is trained under supervision and has an ability to form non-linear decision boundary to classify patterns. In our case a MLP with a 12 neurons in its hidden layer has been used.

8.3.3. Experimental results. A database of Catalan children were used to train the MLP. In table 2 are listed the different phonemes used in the present project. A total of 100 samples of each phoneme were used out of which 90 where used for training and 20 for testing the classifier.

Vowels	/ə/ /e/ /i/ /o/
Consonants	/m/ /n/ /s/

TABLE 2. List of the phonemes extracted from the data base.

A simple MLP classifier with one hidden layer with 12 neurons and an output layer is simulated. The number of nodes in the output layer is chosen to be equal to the number of classes (seven).

The recognition performance achieved by using the energy coefficients per sample is found to be quite high as one can see in table 3. The values in the diagonal should be 1 if the classifier worked perfectly. The worst value in the diagonal is 0.53 which is quite high. As the phoneme /s/ is the only one of fricative type it has the higher classifier value.

Proposed feature confusion matrix							
	/ə/	/e/	/i/	/o/	/m/	/n/	/s/
/ə/	0.71	0.04	0	0.04	0.04	0.17	0
/e/	0	0.85	0.05	0.1	0	0	0
/i/	0	0	0.53	0	0.11	0.31	0.05
/o/	0.19	0.05	0	0.71	0	0.05	0
/m/	0	0	0	0	0.8	0.2	0
/n/	0	0	0.12	0.09	0.09	0.61	0.09
/s/	0	0	0	0	0.1	0	0.9

TABLE 3. Confusion matrix obtained with the testing samples.

8.4. Examples

In this section we present two examples of the whole algorithm.

The first one corresponds to the word *OM*. In this case the segmentation process returns the correct points of the phoneme boundaries as one can see in the figure 8. After the simulating step with the MLP the result is the phonetic transcription /o//m/.

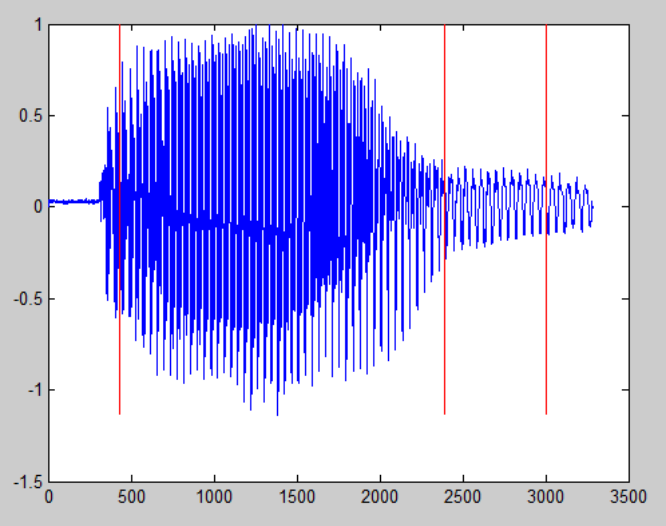


FIG. 8. The segmentation points found by the algorithm in the Catalan word *OM*.

The second one corresponds a child saying the Catalan word *ESA*. In this example the segmentation process divides the signal into three different phonemes but one of them has a bad position (See figure 9. However, once the phonemes are simulated with the MLP the result is /e//s//ə/ due to the fact that the center of each segmented phoneme is correct

The word of the last example is *ÓS*. The output of the first part of the algorithm corresponds to five distinguished phonemes (See figure 10. Even so, the program returns that the phonetic transcription of the speech signal is /o//s//s//s//s/. After that, as there are chains of the same phoneme, the final result of the program is /o//s/.

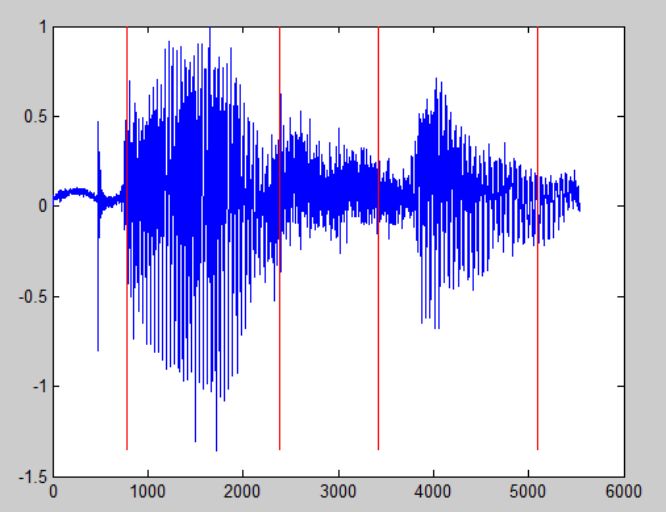


FIG. 9. The segmentation points found by the algorithm in the Catalan word *ESSA*.

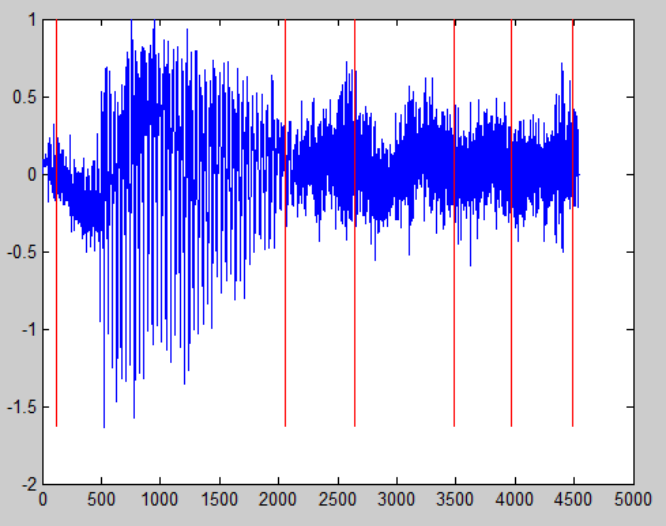


FIG. 10. The segmentation points found by the algorithm in the Catalan word *ÓS*.

Conclusions

In this project we have presented an introduction of the basic analysis tools and techniques in signal processing. We have also trieed to illustrate their potential by implementing them in some applications in the real world.

Firstly, we have introduced the most basic aspects of signal processing in a systematic way. To do it we have tried not to make any difference between analog and digital world, so we ave presented the definitions and properties for both at the same time. We have also tried to explain all concepts in a concise and simple way.

After that we have decided, as signal processing is a huge field, to narrow down on pattern theory and wavelet theory, and how they are used in the real world.

The underlying idea of pattern theory is to find classes of stochastic models that can describe all the patterns we see in nature, so that random samples from these models have the same behaviour as the samples from the world itself. Then the detection of patterns in noisy and ambiguous samples can be achieved by the useing Bayesian inference.

As we have described in the text, wavelet analysis is a relatively new tool that generalises Fourier analysis. The Fourier transform decomposes a signal into a sinusoid basis of different frequencies. As a sine wave has an infinite time duration and a specific frequency, it is perfectly localized in frequency domain but not in time domain. To counter this problem, the wavelet transform decomposes a signal in a base formed by shifted and scaled versions of function limited in both time and frequency.

We have used this local analysis of time signals to extract information from speech signals and parse a recorded voice into phonemes. This is because some phonemes have power variations in a narrow band only. So, it is much easier to detect them analysing DWT signal components than the power of the whole signal. We have also tried to detect the patterns that distinguish one phoneme from the others. In our case, the recognition performance has more than 50% match to each phoneme used.

We also coded an algorithm that finds other patterns, such as the type of wallpaper group that forms a mosaic. For this type of detection we used the autocorrelation of the image with some transformation of it as a standard component.

Future Work

We could pursue this work in several different ways, as we have have become, during the preparation of this memoir, increasingly aware of a variety of applications.

First of all, we would like to improve our speech recogniser explained in the last chapter of the project. It would be very interesting to try it with the whole Catalan phoneme alphabet and with a lot more of samples in order to increase our neuronal network matches. Another application we have in mind is to create fictitious speakers from our training base set. This could be implemented by defining the rules of each phoneme pattern as stochastic processes and creating new samples once the process is defined.

In the pattern theory field, we envision to go deeper into the understanding of image grammars, which are meant to produce a parse graph as the most probable interpretation of the image. This parsed graph amounts to a decomposition of the image into a hirarchy of parts, such as objects, people, background, and so on. At the end, all pixels should be explained and also the mutual relationships between the parts.

References

1. C. Bénéteau and P.J. Van Fleet, *Discrete wavelet transformations and undergraduate education*, Notices of the AMS **58** (2011), 656–666.
2. T. Boswell, *Isometries of the plane*, (2009), `http://www.math.uchicago.edu/~may/VIGRE/VIGRE2009/REUPapers/Boswell.pdf`.
3. P. Castiglioni, *Levinson–Durbin Algorithm*, Encyclopedia of Biostatistics (2005).
4. O. Farooq and S. Datta, *Phoneme recognition using wavelet based features*, Information Sciences **150** (2003), 5–15.
5. M. Frazier, *An introduction to wavelets through linear algebra*, Undergraduate texts in mathematics, Springer Verlag, 1999.
6. M. Gales and S. Young, *The application of hidden Markov models in speech recognition*, Foundations and Trends in Signal Processing **1** (2008), 195–304.
7. S.E. Golowich and D.X. Sun, *A support vector/hidden Markov model approach to phoneme recognition*, ASA Proceedings of the Statistical Computing Section, 1998, pp. 125–130.
8. B.H. Juang and L.R. Rabiner, *Hidden Markov models for speech recognition*, Technometrics (1991), 251–272.
9. H.C. Lin, L.L. Wang, and S.N. Yang, *Extracting periodicity of a regular texture based on autocorrelation functions*, Pattern Recognition Letters **18** (1997), 433–443.
10. Y. Liu and R.T. Collins, *Frieze and wallpaper symmetry groups classification under affine and perspective distortion*, Tech. report, Citeseer, 1998.
11. Y. Liu, R.T. Collins, and Y. Tsin, *A computational model for periodic pattern perception based on frieze and wallpaper groups*, Pattern Analysis and Machine Intelligence, IEEE Transactions on **26** (2004), 354–371.
12. C. G. Looney, *Pattern recognition using neural network: Theory and algorithms for engineers and scientists*, Oxford University Press, 1997.
13. S. Melnikoff, S. Quigley, and M. Russell, *Implementing a hidden markov model speech recognition system in programmable logic*, Field-Programmable Logic and Applications, Springer, 2001, pp. 81–90.
14. Patrick J. Morandi, *The classification of wallpaper patterns: From group cohomology to Escher's tessellations.*
15. D. Mumford and A. Desolneux, *Pattern theory: The stochastic analysis of real-world signals*, AK Peters, 2010.
16. Anke D. Pohl, *Introduction to orbifolds*, (2010), `http://www.math.ethz.ch/~anpohl/ln/Pohl_orbifoldsLN.pdf`.
17. J.G. Proakis and D.G. Manolakis, *Digital signal processing: principles, algorithms, and applications*, vol. 3, Prentice Hall Upper Saddle River, 1996.

18. H.L. Resnikoff and R.O.N. Wells, *Wavelet analysis: the scalable structure of information*, Springer Verlag, 1998.
19. RLE Schwarzenberger, *The 17 plane symmetry groups*, The Mathematical Gazette **58** (1974), 123–131.
20. D.A. Singer, *Isometries of the plane*, The American mathematical monthly **102** (1995), 628–631.
21. S. Xambó Descamps, *Geometría*, vol. 60, Edicions UPC, 2001.
22. B. Zioko, S. Manandhar, and R.C. Wilson, *Phoneme segmentation of speech*, Pattern Recognition **4** (2006), 282–285.
23. M. Ziółko, J. Gałka, and T. Drwiega, *Wavelet transform in speech segmentation*, Progress in Industrial Mathematics at ECMI 2008 (2010), 1073–1078.