Mathematical Foundations of Data Sciences

Gabriel Peyré
CNRS & DMA
École Normale Supérieure
gabriel.peyre@ens.fr
www.gpeyre.com
www.numerical-tours.com

October 7, 2017

Presentation

This book draft presents an overview of important mathematical and numerical foundations for modern data sciences. It covers in particulars the basics of signal and image processing (Fourier, Wavelets, and their applications to denoising and compression), imaging sciences (inverse problems, sparsity, compressed sensing) and machine learning (linear regression, logistic classification, deep learning). The focus is on the mathematically-sounded exposition of the methodological tools (in particular linear operators, non-linear approximation, convex optimization, optimal transport) and how they can be mapped to efficient computational algorithms. These course notes are also intended to be the theortical companion for the Numerical Tours[1] web site, which presents Matlab/Python/Julia/R detailed implementations of all the concepts covered here.

[1] www.numerical-tours.com

Contents

Chapter 1

Shannon Theory

The main reference is [29].

1.1 Analog vs. Discrete Signals

To develop numerical tools and analyze their performances, the mathematical modeling is usually done over a continuous setting. An analog signal is a 1D function $f_0 \in \mathrm{L}^2([0,1])$ where $[0,1]$ denotes the domain of acquisition, which might for instance be time. An analog image is a 2D function $f_0 \in \mathrm{L}^2([0,1]^2)$ where the unit square $[0,1]^2$ is the image domain.

Although these notes are focussed on the processing of sounds and natural images, most of the methods extend to multi-dimensional datasets, which are higher dimensional mappings

$$f_0 : [0,1]^d \to [0,1]^s$$

where d is the dimensionality of the input space ($d = 1$ for sound and $d = 2$ for images) whereas s is the dimensionality of the feature space. For instance, gray scale images corresponds to $(d = 2, s = 1)$, videos to $(d = 3, s = 1)$, color images to $(d = 2, s = 3)$ where one has three channels (R, G, B). One can even consider multi-spectral images where $(d = 2, s \gg 3)$ that is made of a large number of channels for different light wavelengths. Figures 1.1 and 1.2 show examples of such data.

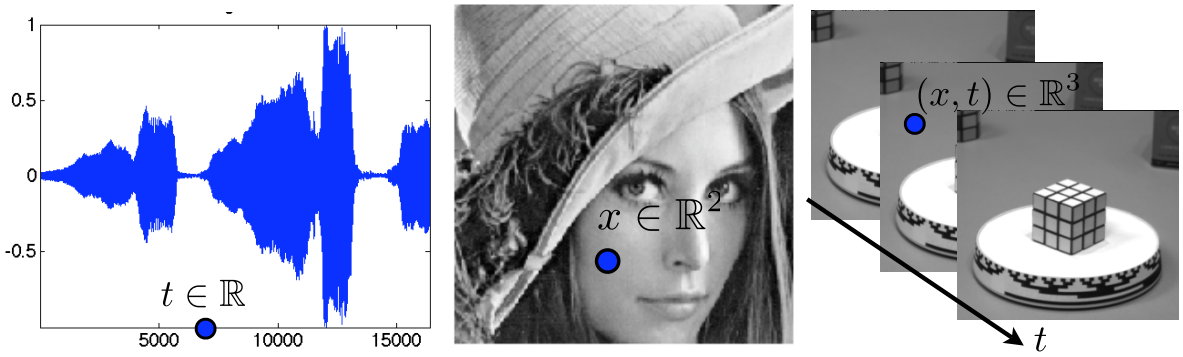


Figure 1.1: Examples of sounds ($d = 1$), image ($d = 2$) and videos ($d = 3$).

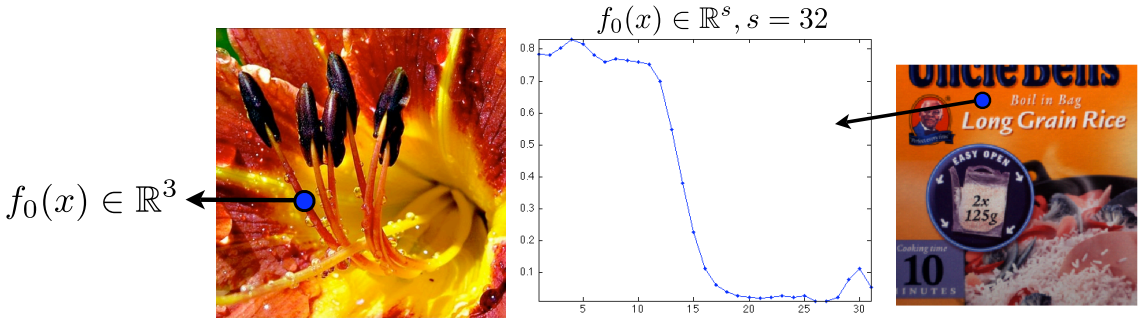


Figure 1.2: Example of color image $s = 3$ and multispectral image ($s = 32$).

1.1.1 Acquisition and Sampling

Signal acquisition is a low dimensional projection of the continuous signal performed by some hardware device. This is for instance the case for a microphone that acquires 1D samples or a digital camera that acquires 2D pixel samples. The sampling operation thus corresponds to mapping from the set of continuous functions to a discrete finite dimensional vector with N entries.

$$f_0 \in \mathrm{L}^2([0,1]^d) \mapsto f \in \mathbb{C}^N$$

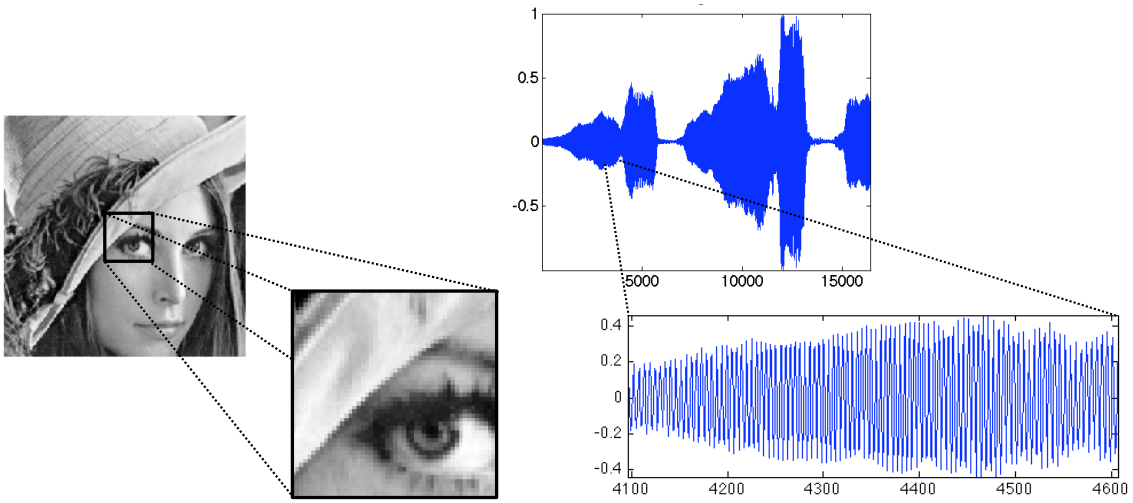


Figure 1.3: Image and sound discretization.

Figure 1.3 shows examples of discretized signals.

1.1.2 Linear Translation Invariant Sampler

A translation invariant sampler performs the acquisition as an inner product between the continuous signal and a constant impulse response h translated at the sample location

$$f[n] = \int_{-S/2}^{S/2} f_0(x) h(n/N - x) \mathrm{d}x = f_0 \star h(n/N). \tag{1.1}$$

The precise shape of $h(x)$ depends on the sampling device, and is usually a smooth low pass function that is maximal around $x = 0$. The size S of the sampler determines the precision of the sampling device, and is usually of the order of $1/N$ to avoid blurring (if S is too large) or aliasing (if S is too small).

Section **??** details how to reverse the sampling operation in the case where the function is smooth.

1.2 Shannon Sampling Theorem

Reminders about Fourier transform. For $f \in L^1(\mathbb{R})$, its Fourier transform is defined as

$$\forall\, \omega \in \mathbb{R}, \quad \hat{f}(\omega) \stackrel{\text{def.}}{=} \int_{\mathbb{R}} f(x) e^{-\mathrm{i}x\omega} \mathrm{d}x. \tag{1.2}$$

One has $\|\hat{f}\|^2 = (2\pi)^{-1}\|f\|^2$, so that $f \mapsto \hat{f}$ can be extended by continuity to $L^2(\mathbb{R})$, which corresponds to computing $\hat{f}$ as a limit when $T \to +\infty$ of $\int_{-T}^{T} f(x)e^{-\mathrm{i}x\omega}\mathrm{d}x$. When $\hat{f} \in L^1(\mathbb{R})$, one can invert the Fourier transform so that

$$f(x) = \int_{\mathbb{R}} \hat{f}(\omega) e^{\mathrm{i}x\omega} \mathrm{d}\omega, \tag{1.3}$$

which shows in particular that f is continuous with vanishing limits at $\pm\infty$.

The Fourier transform $\mathcal{F} : f \mapsto \hat{f}$ exchanges regularity and decay. For instance, if $f \in C^p(\mathbb{R})$ with an integrable Fourier transform, then $\mathcal{F}(f^{(p)})(\omega) = (\mathrm{i}\omega)^{-p}\hat{f}(\omega)$ so that $|\hat{f}(\omega)| = O(1/|\omega|^p)$. Conversely,

$$\int_{\mathbb{R}} (1 + |\omega|)^{-p} |\hat{f}(\omega)| \mathrm{d}\omega < +\infty \quad \Longrightarrow \quad f \in C^p(\mathbb{R}). \tag{1.4}$$

Reminders about Fourier series. We denote $\mathbb{T} = \mathbb{R}/2\pi\mathbb{Z}$ the torus. A function $f \in L^2(\mathbb{T})$ is 2π-periodic, and can be viewed as a function $f \in L^2([0, 1])$ (beware that this means that the boundary points are glued together), and its Fourier coefficients are

$$\forall\, n \in \mathbb{Z}, \quad \hat{f}_n \stackrel{\text{def.}}{=} \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-\mathrm{i}xn} \mathrm{d}x.$$

This formula is equivalent to the computation of an inner-product $\hat{f}_n = \langle f,\, e_n \rangle$ for the inner-product $\langle f,\, g \rangle \stackrel{\text{def.}}{=} \frac{1}{2\pi}\int_{\mathbb{T}} f(x)\bar{g}(x)\mathrm{d}x$. For this inner product, $(e_n)_n$ is orthonormal and is actually an Hilbert basis, meaning that one reconstruct with the following converging series

$$f = \sum_{n \in \mathbb{Z}} \langle f,\, e_n \rangle e_n \tag{1.5}$$

which means $\|f - \sum_{n=-N}^{N} \langle f,\, e_n \rangle e_n\|_{L^2(\mathbb{T})} \to 0$ for $N \to +\infty$. The pointwise convergence of (1.5), and is ensured (and there is normal convergence) when for instance $f \in C^3(\mathbb{T})$.

Poisson formula. The poisson formula connects the Fourier transform and the Fourier series to sampling and periodization operators. For some function $\hat{f}(\omega)$ defined on $\mathbb{R}$, its periodization reads

$$\hat{f}_P(\omega) \stackrel{\text{def.}}{=} \sum_n f(\omega - 2\pi n). \tag{1.6}$$

This formula makes sense if $\hat{f} \in L^1(\mathbb{R})$, and in this case $\|\hat{f}_P\|_{L^1(\mathbb{T})} \leqslant \|\hat{f}\|_1$ The Poisson formula, state in Proposition 1 bellow, corresponds to proving that the following diagram

$$\begin{array}{ccccc} & f(x) & \stackrel{\mathcal{F}}{\longrightarrow} & \hat{f}(\omega) & \\ \text{sampling} & \downarrow & & \downarrow & \text{periodization} \\ & (f(n))_n & \stackrel{\text{Fourier serie}}{\longrightarrow} & \sum_n f(n) e^{-\mathrm{i}\omega n} & \end{array}$$

is actually commutative.

Proposition 1 (Poisson formula). *Assume that $\hat{f}$ has compact support and that $|f(x)| \leqslant C(1+|x|)^{-3}$ for some C. Then one has*

$$\forall\, \omega \in \mathbb{R}, \quad \sum_n f(n) e^{-\mathrm{i}\omega n} = \hat{f}_P(\omega). \tag{1.7}$$

Proof. Since $\hat{f}$ is compactly supported, $\hat{f}_P$ is well defined (it involves only a finite sum) and since f has fast decay, using (1.4), $\hat{f}_P$ is C^1. It is thus the sum of its Fourier transform

$$\hat{f}_P(\omega) = \sum_k c_k e^{\mathrm{i}k\omega}, \tag{1.8}$$

where

$$c_k = \frac{1}{2\pi}\int_0^{2\pi} \hat{f}_P(\omega) e^{-\mathrm{i}k\omega}\mathrm{d}\omega = \frac{1}{2\pi}\int_0^{2\pi} \sum_n f(x-2\pi n) e^{-\mathrm{i}k\omega}\mathrm{d}\omega.$$

One has

$$\int_0^{2\pi} \sum_n |f(x-2\pi n) e^{-\mathrm{i}k\omega}|\mathrm{d}\omega = \int_{\mathbb{R}} |f|$$

which is bounded because $\hat{f} \in L^1(\mathbb{R})$ (it has a compact support and is C^1), so one can exchange the sum and integral

$$c_k = \sum_n \frac{1}{2\pi}\int_0^{2\pi} f(x-2\pi n) e^{-\mathrm{i}k\omega}\mathrm{d}\omega = \frac{1}{2\pi}\int_{\mathbb{R}} f(x) e^{-\mathrm{i}k\omega}\mathrm{d}\omega = f(-k)$$

where we used the inverse Fourier transform formula (1.3), which is legit because $\hat{f} \in L^1(\mathbb{R})$. □

Shannon theorem. Shannon sampling theorem state a sufficient condition ensuring that the sampling operator $f \mapsto (f(ns))_n$ is invertible for some sampling step size $s > 0$. It require that $\mathrm{supp}(\hat{f}) \subset [-\pi/s, \pi/s]$, which, thanks to formula (1.3), implies that $\hat{f}$ is C^∞ (in fact it is even analytic).

Theorem 1. *If $|f(x)| \leqslant C(1+|x|)^{-3}$ for some C and $\mathrm{supp}(\hat{f}) \subset [-\pi/s, \pi/s]$, then one has*

$$\forall\, x \in \mathbb{R}, \quad f(x) = \sum_n f(ns)\,\mathrm{sinc}(x/s - n) \quad \textit{where} \quad \mathrm{sinc}(u) = \frac{\sin(\pi u)}{\pi u} \tag{1.9}$$

with uniform convergence.

Proof. The change of variable $g = f(s\cdot)$ results in $\hat{g} = s\hat{f}(s\cdot)$ so that we can restrict our attention to $s = 1$. The compact support hypothesis implies $\hat{f}(\omega) = 1_{[-\pi,\pi]}(\omega)\hat{f}_P(\omega)$. Combining the inversion formula (1.3) with Poisson formula (1.8)

$$f(x) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \hat{f}_P(\omega) e^{\mathrm{i}\omega x}\mathrm{d}\omega = \frac{1}{2\pi}\int_{-\pi}^{\pi} \sum_n f(n) e^{\mathrm{i}\omega(x-n)}\mathrm{d}\omega.$$

Since f has fast decay, $\int_{-\pi}^{\pi} \sum_n |f(n)e^{\mathrm{i}\omega(x-n)}|\mathrm{d}\omega = \sum_n |f(n)| < +\infty$, so that one can exchange summation and integration and obtain

$$f(x) = \sum_n f(n) \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{\mathrm{i}\omega(x-n)}\mathrm{d}\omega = \sum_n f(n)\,\mathrm{sinc}(x-n).$$

□

1.3 Shannon Source Coding Theorem

We consider an alphabet $(x_1, \dots, x_K)$ of K symbols, and assume at our disposal some probability distribution over this alphabet, which is just an histogram $p = (p_1, \dots, p_K) \in \mathbb{R}_+^K$ in the simplex, i.e. $\sum_k p_k = 1$.

The entropy of such an histogram is

$$H(p) \stackrel{\text{def.}}{=} -\sum_k p_k \log_2(p_k)$$

with the convention $O \log_2(0) = 0$.

Lemma 1. *One has*

$$0 \leqslant H(p) \leqslant \log_2(K).$$

Proof. We consider the following constrained optimization problem

$$\min_p \left\{ f(p) \; ; \; g(p) = \sum_k p_k = 1 \right\}$$

where $f = -H$. According to the linked extrema theorem, at an optimum $p^\star$, $\nabla f(p^\star) = \lambda \nabla g(p^\star)$ for some $\lambda \in \mathbb{R}$, so that here $\log(p_k^\star) + 1 = \lambda$, i.e. $p_k^\star = c$ is constant, and since $\sum_k p_k^\star = 1$, one has $p_k^* = 1/K$ and thus $H(p) = \log_2(K)$. □

A code $c_k = c(x_k)$ associate to each symbol x_k a code word $c_k \in \{0,1\}^{\mathbb{N}}$ with a varying length $|c_k| \in \mathbb{N}^*$. We denote the average length associated to this code as

$$L(c) \stackrel{\text{def.}}{=} \sum_k p_k |c_k|.$$

A prefix code $c_k = c(x_k)$ is such that no word c_k is the beginning of another word c_k'. This is equivalent to be able to embed the $(c_k)_k$ as leaves of a binary tree T, with the code being output of a traversal from root to leaves (with a convention that going to a left (resp. right) child output a 0 (resp. a 1). We denote $c = \text{Leaves}(T)$ such prefix property. The following fundamental lemma describes the set of prefix code using an inequality.

Lemma 2 (Kraft inequality)**.** *(i) For a code c, if there exists a tree T such that $c = Leaves(T)$ then*

$$\sum_k 2^{-|c_k|} \leqslant 1. \tag{1.10}$$

(ii) Conversely, if $(\ell_k)_k$ are such that

$$\sum_k 2^{-\ell_k} \leqslant 1 \tag{1.11}$$

then there exists a code $c = Leaves(T)$ such that $|c_k| = \ell_k$.

Proof. $\Rightarrow$ We suppose $c = \text{Leaves}(T)$. We denote $m = \max_k |c_k|$ and consider the full binary tree. Bellow each c_k, one has a sub-tree of height $m - |c_k|$. This sub-tree has $2^{m-|c_k|}$ leaves. Since all these sub-trees do not overlap, the total number of leaf do not exceed the total number of leaves 2^m of the full binary tree, hence

$$\sum_k 2^{m-|c_k|} \leqslant 2^m,$$

hence (1.10).

$\Leftarrow$ Conversely, we assume (1.10) holds. Without loss of generality, we assume that $|c_1| \leqslant \ldots \leqslant |c_K|$. We start by putting a sub-tree of height $2^{m-|c_1|}$. Since the second tree is smaller, one can put it immediately aside, and continue this way. Since $\sum_k 2^{m-|c_k|} \leqslant 2^m$, this ensure that we can stack side-by-side all these sub-tree, and this defines a proper sub-tree of the full binary tree. □

We now are ready to state and prove Shannon theory for entropic coding.

Theorem 2. *(i) If $c = \mathit{Leaves}(T)$ for some tree T, then*

$$L(c) \geqslant H(p).$$

(ii) Conversely, there exists a code c with $c = \mathit{Leaves}(T)$ such that

$$L(c) \leqslant H(p) + 1.$$

Proof. First, we consider the following optimization problem

$$\min_{\ell=(\ell_k)_k} \left\{ f(\ell) \stackrel{\text{def.}}{=} \sum_k \ell_k p_k \;;\; g(\ell) \stackrel{\text{def.}}{=} \sum_k 2^{-\ell_k} \leqslant 1 \right\}. \tag{1.12}$$

We fist show that at an optimal $\ell^\star$, the constraint is saturated, i.g. $g(\ell^\star) = 1$. Indeed, if $g(\ell^\star) = 2^{-u} < 1$, with $u > 0$, we define $\ell'_k \stackrel{\text{def.}}{=} \ell^\star_k - u$, which satisfies $g(\ell') = 1$ and also $f(\ell') = \sum_k (\ell_k - u)p_k < f(\ell^\star)$, which is a contradiction. So we can restrict in (1.12) the constraint to $g(\ell) = 1$ and apply the linked extra theorem, which shows that necessarily, there exists $\lambda \in \mathbb{R}$ with $\nabla f(\ell^\star) = \nabla g(\ell^\star)$, i.e. $(p_k)_k = -\lambda \ln(2)(2^{-\ell^\star_k})_k$. Since $\sum_k p_k = \sum_k 2^{-\ell^\star_k} = 1$, we deduce that $\ell^\star_k = -\log(p_k)$.

(i) If $c = \mathrm{Leave}(T)$, the by Kraft inequality (1.10), necessarily $\ell_k = |c_k|$ satisfy the constraints of (1.12), and thus $H(p) = f(\ell^\star) \leqslant f(\ell) = L(\ell)$.

(ii) We define $\ell_k \stackrel{\text{def.}}{=} \lceil -\log_2(p_k) \rceil \in \mathbb{N}^*$. Then $\sum_k 2^{-\ell_k} \leqslant \sum_k 2^{\log_2(p_k)} = 1$, so that these lengths satisfy (1.11). Thanks to Proposition 2 (ii), there thus exists a prefix code c with $|c_k| = \lceil -\log_2(p_k) \rceil$. Furthermore

$$L(c) = \sum_k p_k \lceil -\log_2(p_k) \rceil \leqslant \sum_k p_k (-\log_2(p_k) + 1) = H(p) + 1.$$

□

Chapter 2

Fourier Transforms

The main references for this chapter is [29]. The Fourier transforms offers a perfect blend of analysis (solution of PDEs, approximation of functions), algebra (characters of groups, representation theory) and computer science (the FFT). This chapter offers a glimpse of all these different facets.

2.1 Hilbert spaces and Fourier Transforms

2.1.1 Hilbertian bases.

An Hilbert space $(\mathcal{H}, \langle \cdot, \cdot \rangle)$ is complete. If it is separable, it can be equipped with an Hilbertian orthogonal basis $(\varphi_n)_{n \in \mathbb{N}}$, which means that one can expand any $f \in \mathcal{H}$ as

$$f = \sum_n \langle f, \varphi_n \rangle \varphi_n$$

where the convergence is in the sense of $\|f\|^2 \overset{\text{def.}}{=} \langle f, f \rangle$, i.e. $\|f - \sum_{n=0}^N \langle f, \varphi_n \rangle \varphi_n\| \to 0$ as $N \to +\infty$. One also have the conservation of energy

$$\|f\|^2 = \sum_n \langle f, \varphi_n \rangle^2.$$

A way to construct such an ortho-basis is using Gram-Schmidt orthogonalization procedure. On $L^2([0,1])$ equipped with the usual inner product, orthogonalization of monomials defines the Legendre polynomials. On $L^2(\mathbb{R})$ equipped with a Gaussian measure, this leads to functions of the form $P_n(x)e^{-x^2}$ where P_n are Laguerre polynomials. Intuitively, orthogonality forces φ_n to have n "oscillations", e.g. orthogonal polynomials have exactly n zeros.

Figure 2.1, left, shows examples of the real part of Fourier atoms.

2.1.2 Fourier basis on $\mathbb{R}/2\pi\mathbb{Z}$.

On $L^2(\mathbb{T})$ where $\mathbb{T} = \mathbb{R}/2\pi\mathbb{Z}$, equipped with $\langle f, g \rangle \overset{\text{def.}}{=} \frac{1}{2\pi} \int_{\mathbb{T}} f(x)\bar{g}(x)\mathrm{d}x$, one can use the Fourier basis

$$\varphi_n(x) \overset{\text{def.}}{=} e^{\mathrm{i}nx} \quad \text{for} \quad n \in \mathbb{Z}. \tag{2.1}$$

One thus has

$$f = \sum_n \hat{f}_n e^{\mathrm{i}n\cdot} \quad \text{where} \quad \hat{f}_n \overset{\text{def.}}{=} \frac{1}{2\pi} \int_0^{2\pi} f(x)e^{-\mathrm{i}nx}\mathrm{d}x, \tag{2.2}$$

in $L^2(\mathbb{T})$ sense. Pointwise convergence is delicate, see Section 1.2.

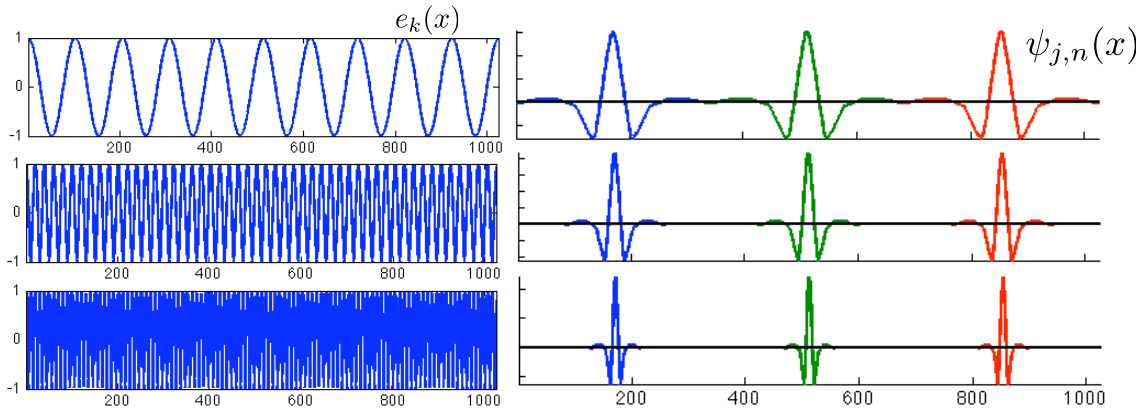


Figure 2.1: Left: 1D Fourier (real part), right: wavelet bases.

We recall that for $f \in L^1(\mathbb{R})$, its Fourier transform is defined as

$$\forall \omega \in \mathbb{R}, \quad \hat{f}(\omega) \stackrel{\text{def.}}{=} \int_{\mathbb{R}} f(x) e^{-\mathrm{i}x\omega} \mathrm{d}x.$$

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

The connexion between the Fourier transform on $\mathbb{R}$ and the Fourier coefficients on $\mathbb{T}$ is given by the following diagram

$$\begin{array}{ccc} f(x) & \stackrel{\mathcal{F}}{\longrightarrow} & \hat{f}(\omega) \\ \text{sampling} \downarrow & & \downarrow \text{periodization} \\ (f(n))_n & \stackrel{\text{Fourier serie}}{\longrightarrow} & \sum_n f(n) e^{-\mathrm{i}\omega n} \end{array}.$$

Its commutativity sates

$$\sum_n f(n) e^{-\mathrm{i}\omega n} = \sum_n \hat{f}(\omega - 2\pi n) \tag{2.3}$$

and this is in fact the celebrated Poisson formula (Proposition 1).

2.2 Convolution on $\mathbb{R}$ and $\mathbb{T}$

2.2.1 Convolution

On $\mathbb{X} = \mathbb{R}$ or $\mathbb{T}$, one defines

$$f \star g(x) = \int_{\mathbb{X}} f(t) g(x-t) \mathrm{d}t. \tag{2.4}$$

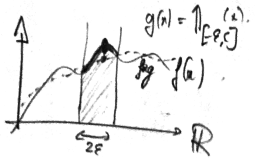


Figure 2.3: Convolution on $\mathbb{R}$.

Young's inequality shows that this quantity is well defined if $(f, g) \in L^p(\mathbb{X}) \times L^q(\mathbb{X})$

$$\frac{1}{p} + \frac{1}{q} = 1 + \frac{1}{r} \quad \Longrightarrow \quad f \star g \in L^r(\mathbb{X}) \quad \text{and} \quad \|f \star g\|_{L^r(\mathbb{X})} \leqslant \|f\|_{L^p(\mathbb{X})} \|g\|_{L^q(\mathbb{X})}. \tag{2.5}$$

This shows that if $f \in L^1(\mathbb{X})$, then one has the map $g \in L^p(\mathbb{X}) \mapsto f \star g \in L^p(\mathbb{X})$ is a continuous map on $L^p(\mathbb{X})$. Furthermore, when $r = \infty$, $f \star g \in \mathcal{C}^0(\mathbb{X})$ is a

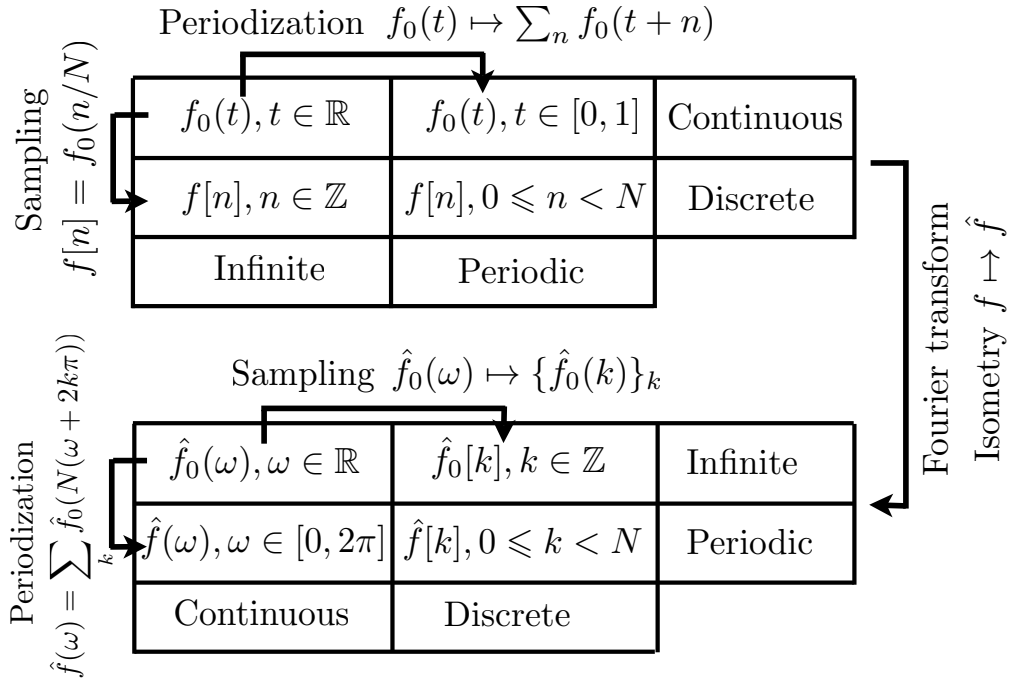


Figure 2.2: The four different settings for Fourier analysis, and the sampling-periodization relationship.

continuous function (which shows some regularizing effect). Note that for $\mathbb{X} = \mathbb{T}$, $p < q \quad \Longrightarrow \quad L^q(\mathbb{T}) \subset L^p(\mathbb{T})$, so that $L^\infty(\mathbb{X})$ is the smallest space.

Convolution is mostly used in order to regularize functions. For instance, if $f \in L^1(\mathbb{X})$ and $g \in \mathcal{C}^1(\mathbb{X})$ is bounded, then $f \star g$ is differentiable and $(f \star g)' = f \star g'$. This is used to produce smooth approximate identity $(\rho_\varepsilon = \frac{1}{\varepsilon}\rho(\cdot/\varepsilon))_\varepsilon$ with convergence $f \star \rho_\varepsilon \to f$ in $L^p(\mathbb{X})$ for $1 \leqslant p < +\infty$ of smooth approximations (and convergence in $L^\infty(\mathbb{X})$ if f is uniformly continuous). This is also used for denoising applications in signal and image processing.

For $(f, g) \in L^1(\mathbb{X})^2$ (so on $\mathbb{X} = \mathbb{T}$, also in any $L^p(\mathbb{X})$), one has

$$\mathcal{F}(f \star g) = \hat{f} \odot \hat{g} \tag{2.6}$$

which means that $\mathcal{F}$ is a morphism of algebra. For instance if $\mathbb{X} = \mathbb{R}$, its range is included in the algebra of continuous functions with vanishing limits in $\pm\infty$.

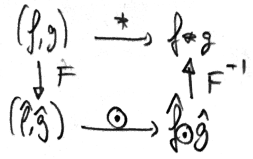


Figure 2.6: Commutative diagram of convolution-Fourier.

2.2.2 Translation Invariant Operators

Translation invariant operators (which commutes with translation) are fundamental because in most situations, input (signal, image, etc) should be processed without spatial preference. The following propositions shows that any translation invariant[1] (i.e. which commutes with translations) operator is actually a "convolution" against a distribution with bounded Fourier transform. The proof and conclusion (regularity of the convolution kernel) vary depending on the topology on the input and output spaces. We first study the case of convolution mapping to continuous functions.

Proposition 2. *We define $T_\tau f = f(\cdot - \tau)$. A bounded linear operator $H : (L^2(\mathbb{X}), \|\cdot\|_2) \to (C^0(\mathbb{X}), \|\cdot\|_\infty)$*

[1] One should rather actually say "translation equivariant".

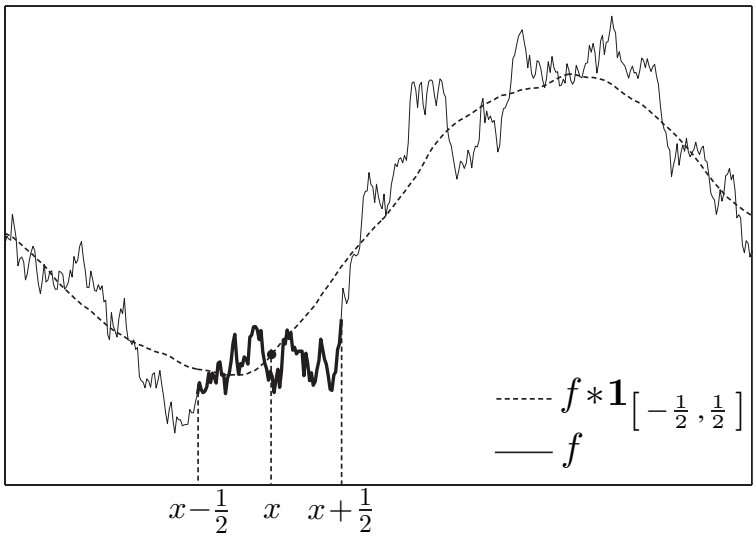


Figure 2.4: Signal filtering with a box filter (running average).

is such that for all τ, $H \circ T_\tau = T_\tau \circ H$ *if and only if*

$$\forall\, f \in L^2(\mathbb{T}), \quad H(f) = f \star g$$

with $g \in L^2(\mathbb{X})$.

Proof. Thanks to (2.5) (and the remark in the case $r = \infty$), $T : f \mapsto f \star g$ with $g \in L^2(\mathbb{X})$ is indeed a continuous operator from $L^2(\mathbb{X})$ to $C^0(\mathbb{X})$. Furthermore

$$(H \circ T_\tau)(f) = \int_{\mathbb{X}} f((\cdot - \tau) - y) g(y) \mathrm{d}\tau = (f \star g)(\cdot - \tau) = T_\tau(Hf),$$

so that such an H is translation-invariant.

Conversely, we define $\ell : f \mapsto H(f)(0) \in \mathbb{R}$, which is legit since $H(f) \in \mathcal{C}^0(\mathbb{X})$. Since H is continuous, there exists C such that $\|Hf\|_\infty \leqslant C\|f\|_2$, and hence $|\ell(f)| \leqslant C\|f\|_2$, so that f is a continuous linear form on the Hilbert space $L^2(\mathbb{X})$. Hence, according to Fréchet-Riesz theorem, there exists $h \in L^2(\mathbb{X})$ such that $\ell(f) = \langle f, h \rangle$. Hence, $\forall\, x \in \mathbb{X}$,

$$H(f)(x) = T_{-x}(Hf)(0) = H(T_{-x}f)(0) = \ell(T_{-x}f) = \langle T_{-x}f, h \rangle = \int_{\mathbb{X}} f(y+x)h(y)\mathrm{d}y = f \star \bar{h}(x).$$

where $g \overset{\text{def.}}{=} \bar{h} = h(-\cdot) \in L^2(\mathbb{X})$. □

We now study, on $\mathbb{T}$, the case of convolution which can output non-continuous functions. In this case, the kernel can be a "distribution", so the convolution is defined over the Fourier domain.

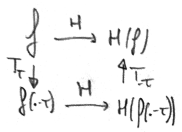


Figure 2.7: Commutative diagram for translation invariance.

Proposition 3. *A bounded linear operator* $H : L^2(\mathbb{T}) \to L^2(\mathbb{T})$ *is such that for all* τ, $H \circ T_\tau = T_\tau \circ H$ *if and only if*

$$\forall\, f \in L^2(\mathbb{T}), \quad \mathcal{F}(H(f)) = \hat{f} \odot c$$

where $c \in \ell^\infty(\mathbb{Z})$ *(a bounded sequence).*

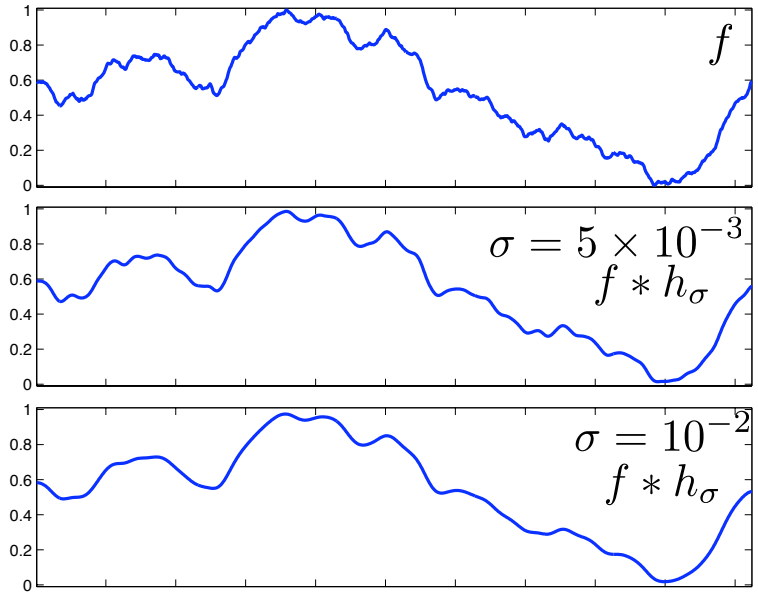


Figure 2.5: Filtering an irregular signal with a Gaussian filter of increasing filter size σ.

Proof. We denote $\varphi_n \overset{\text{def.}}{=} e^{\mathrm{i}n\cdot}$. One has

$$H(\varphi_n) = e^{\mathrm{i}n\tau} H(T_\tau(\varphi_n)) = e^{\mathrm{i}n\tau} T_\tau(H(\varphi_n)).$$

Thus, for all n,

$$\langle H(\varphi_n),\, \varphi_m\rangle = e^{\mathrm{i}n\tau}\langle T_\tau \circ H(\varphi_n),\, \varphi_m\rangle = e^{\mathrm{i}n\tau}\langle H(\varphi_n),\, T_{-\tau}(\varphi_m)\rangle = e^{\mathrm{i}(n-m)\tau}\langle H(\varphi_n),\, \varphi_m\rangle$$

So for $n \neq m$, $\langle H(\varphi_n),\, \varphi_m\rangle = 0$, and we define $c_n \overset{\text{def.}}{=} \langle H(\varphi_n),\, \varphi_n\rangle$. Since H is continuous, $\|Hf\|_{L^2(\mathbb{T})} \leqslant C\|f\|_{L^2(\mathbb{T})}$ for some constant C, and thus by Cauchy-Schwartz

$$|c_n| = |\langle H(\varphi_n),\, \varphi_n\rangle| \leqslant \|H(\varphi_n)\|\|\varphi_n\| \leqslant C$$

because $\|\varphi_n\| = 1$, so that $c \in \ell^\infty(\mathbb{Z})$. By continuity, recalling that by definition $\hat{f}_n \overset{\text{def.}}{=} \langle f,\, \varphi_n\rangle$,

$$H(f) = \lim_N H(\sum_{n=-N}^{N} \hat{f}_n\varphi_n) = \lim_N \sum_{n=-N}^{N} \hat{f}_n H(\varphi_n) = \lim_N \sum_{n=-N}^{N} c_n\hat{f}_n\varphi_n = \sum_{n\in\mathbb{Z}} c_n\hat{f}_n\varphi_n$$

so that in particular one has the desired result. □

This theorem thus states that translation invariant operators are those which are "diagonal" in the Fourier ortho-basis.

2.2.3 Revisiting Poisson formula using distributions.

Informally, the Fourier series

$$\sum_n f(n)e^{-\mathrm{i}\omega n}$$

can be thought as the Fourier transform $\mathcal{F}(\Pi_1 \odot f)$ of the discrete distribution

$$\Pi_1 \odot f = \sum_n f(n)\delta_n \quad \text{where} \quad \Pi_s = \sum_n \delta_{sn}$$

for $s \in \mathbb{R}$, where δ_a is the Dirac mass at location $a \in \mathbb{R}$, i.e. the distribution such that $\int f \mathrm{d}(\mathrm{d}_a) = f(a)$ for any continuous f. Indeed, one can multiply a distribution by a continuous function, and the definition of the Fourier transform of a distribution μ is a distributions $\mathcal{F}(\mu)$ such that that

$$\forall\, g \in \mathcal{S}(\mathbb{R}), \quad \int_{\mathbb{R}} g(x)\mathrm{d}\mathcal{F}(\mu) = \int_{\mathbb{R}} \mathcal{F}^*(g)\mathrm{d}\mu, \quad \text{where} \quad \mathcal{F}^*(g) \overset{\text{def.}}{=} \int_{\mathbb{R}} g(x)e^{\mathrm{i}x\cdot}\mathrm{d}x,$$

where $\mathcal{S}(\mathbb{R})$ are smooth and rapidly decaying (Schwartz class) functions.

The Poisson formula (2.3) can thus be interpreted as

$$\mathcal{F}(\Pi_1 \odot f) = \sum_n \hat{f}(\cdot - 2\pi n) = \int_{\mathbb{R}} \hat{f}(\cdot - \omega)\mathrm{d}\Pi_{2\pi}(\omega) = \hat{f} \star \Pi_{2\pi}$$

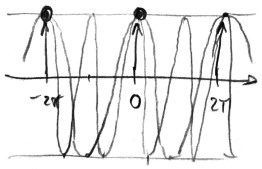


Figure 2.8: Sine wave being summed in the Poisson formula.

Since $\mathcal{F}^{-1} = \frac{1}{2\pi}\mathcal{S} \circ \mathcal{F}$ where $\mathcal{S}(f) = f(-\cdot)$, one has, applying this operator on both sides

$$\Pi_1 \odot f = \frac{1}{2\pi}\, \mathcal{S}\circ\mathcal{F}(\hat{f}\star\Pi_{2\pi}) = \mathcal{S}(\frac{1}{2\pi}\mathcal{F}(\hat{f})\odot\hat{\Pi}_{2\pi}) = \mathcal{S}(\frac{1}{2\pi}\mathcal{F}(\hat{f}))\odot\mathcal{S}(\hat{\Pi}_{2\pi}) = \hat{\Pi}_{2\pi}\odot f.$$

This can be interpreted as the relation

$$\hat{\Pi}_{2\pi} = \Pi_1 \quad \Longrightarrow \quad \hat{\Pi}_1 = 2\pi\Pi_{2\pi}.$$

To intuitively understand this relation, one can compute a finite Fourier series

$$\sum_{n=-N}^{N} e^{-\mathrm{i}n\omega} = \frac{\sin((N+1/2)x)}{\sin(x/2)}$$

which is a smooth function which grows unbounded with $N \to +\infty$ as $N \to +\infty$.

2.3 Finite Fourier Transform and Convolution

2.3.1 Discrete Ortho-bases

Discrete signals are finite dimensional vector $f \in \mathbb{C}^N$ where N is the number of samples and where each $f[n]$ is the value of the signal at a 1D or 2D location. For a 2-D images $f \in \mathbb{C}^N \simeq \mathbb{C}^{N_0 \times N_0}$, $N = N_0 \times N_0$, where N_0 is the number of pixels along each direction.

Discrete signals and images are processed using a discrete inner product that mimics the continuous L^2 inner product

$$\langle f,\, g\rangle = \sum_{n=0}^{N-1} f[n]\bar{g}[n].$$

One thus defines a distance between discretized vectors as

$$\|f - g\|^2 = \sum_{n=0}^{N-1} |f[n] - g[n]|^2.$$

Exactly as in the continuous case, a discrete orthogonal basis $\{\psi_m\}_{0\leqslant m<N}$ of $\mathbb{C}^N$, satisfies

$$\langle \psi_m,\, \psi_{m'}\rangle = \delta[m - m']. \tag{2.7}$$

The decomposition of a signal in such an ortho-basis is written

$$f = \sum_{m=0}^{N-1} \langle f,\, \psi_m\rangle \psi_m.$$

It satisfies a conservation of energy

$$\|f\|^2 = \sum_{n=0}^{N-1} |f[n]|^2 = \sum_{m=0}^{N-1} |\langle f,\, \psi_m\rangle|^2$$

Computing the set of all inner product $\{\langle f,\, \psi_m\rangle\}_{0\leqslant m<N}$ is done in a brute force way in $O(N^2)$ operations. This is not feasible for large datasets where N is of the order of millions. When designing an ortho-basis, one should keep this limitation in mind and enforce some structure in the basis elements so that the decomposition can be computed with fast algorithm. This is the case for the Fourier and wavelet bases, that enjoy respectively $O(N\log(N))$ and $O(N)$ algorithms.

2.3.2 Discrete Fourier transform

We denote $f = (f_n)_{n=0}^{N-1} \in \mathbb{R}^N$, but we insists that such vector should really be understood as being indexed by $n \in \mathbb{Z}/N\mathbb{Z}$, which is a finite commutative group for the addition. This corresponds to using periodic boundary conditions.

The discrete Fourier transform is defined as

$$\forall\, k = 0, \ldots, N-1, \quad \hat{f}_k \overset{\text{def.}}{=} \sum_{n=0}^{N-1} f_n e^{-\frac{2\mathrm{i}\pi}{N}kn} = \langle f,\, \varphi_k\rangle \quad \text{where} \quad \varphi_k \overset{\text{def.}}{=} (e^{\frac{2\mathrm{i}\pi}{N}kn})_{n=0}^{N-1} \in \mathbb{C}^N \tag{2.8}$$

where the canonical inner product on $\mathbb{C}^N$ is $\langle u,\, v\rangle = \sum_{n=1}^N u_n \bar{v}_n$ for $(u, v) \in (\mathbb{C}^N)^2$. This definition can intuitively be motivated by sampling the Fourier basis $x \mapsto e^{\mathrm{i}kx}$ on $\mathbb{R}/2\pi\mathbb{Z}$ at equi-spaced points $(\frac{2\pi}{N}n)_{n=0}^{N-1}$. The following proposition shows that this corresponds to a decomposition in an ortho-basis.

Proposition 4. *One has*

$$\langle \varphi_k,\, \varphi_\ell\rangle = \begin{cases} N & \text{if} \quad k = \ell, \\ 0 & \text{otherwise.} \end{cases}$$

In particular, this implies

$$\forall\, n = 0, \ldots, N-1, \quad f_n = \frac{1}{N}\sum_k \hat{f}_k e^{\frac{2\mathrm{i}\pi}{N}kn}. \tag{2.9}$$

Proof. One has, if $k \neq \ell$

$$\langle \varphi_k,\, \varphi_\ell\rangle = \sum_n e^{\frac{2\mathrm{i}\pi}{N}(k-\ell)n} = \frac{1 - e^{\frac{2\mathrm{i}\pi}{N}(k-\ell)N}}{1 - e^{\frac{2\mathrm{i}\pi}{N}(k-\ell)}} = 0$$

which is the sum of a geometric serie (equivalently, sum of equi-spaced points on a circle). The inversion formula is simply $f = \sum_k \langle f,\, \varphi_k\rangle \frac{\varphi_k}{\|\varphi_k\|^2}$. □

2.3.3 Fast Fourier transform

Assuming $N = 2N'$, one has

$$\hat{f}_{2k} = \sum_{n=0}^{N'-1} (f_n + f_{n+N/2}) e^{-\frac{2\mathrm{i}\pi}{N'}kn}$$

$$\hat{f}_{2k+1} = \sum_{n=0}^{N'-1} e^{-\frac{2\mathrm{i}\pi}{N'}n} (f_n - f_{n+N/2}) e^{-\frac{2\mathrm{i}\pi}{N'}kn}.$$

For the second line, we used the computation

$$e^{-\frac{2\mathrm{i}\pi}{N}(2k+1)(n+N/2)} = e^{-\frac{2\mathrm{i}\pi}{N}(2kn+kN+n+N/2)} = -e^{-\frac{2\mathrm{i}\pi}{N'}n} e^{-\frac{2\mathrm{i}\pi}{N'}kn}.$$

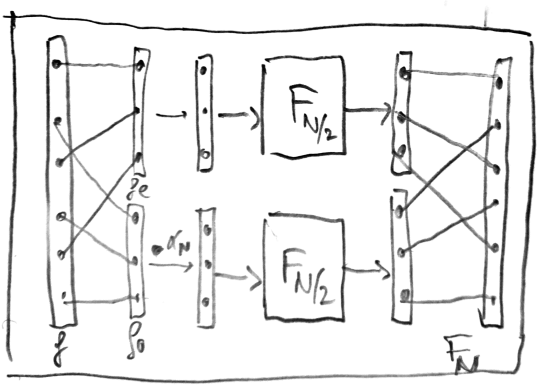


Figure 2.9: Diagram of one step of the FFT.

Denoting $\mathcal{F}_N(f) = \hat{f}$ the discrete Fourier transform on $\mathbb{R}^N$, and introducing the notation $f_e = (f_n + f_{n+N/2})_n \in \mathbb{R}^{N'}$, $f_o = (f_n - f_{n+N/2})_n \in \mathbb{R}^{N'}$ and $\alpha_N = (e^{-\frac{2\mathrm{i}\pi}{N'}n})_n \in \mathbb{R}^{N'}$, one has the following recursion formula

$$\mathcal{F}_N(f) = \mathcal{I}_N(\mathcal{F}_{N/2}(f_e), \mathcal{F}_{N/2}(f_o \odot \alpha_N))$$

where $\mathcal{I}_N$ is the "interleaving" operator, defined by $\mathcal{I}_N(a,b) \overset{\text{def.}}{=} (a_1, b_1, a_2, b_2, \ldots, a_{N'}, b_{N'})$. There iterations define the so-called Fast Fourier Transform algorithm, which works here when N is a power of 2. These iterations can be extended to arbitrary number N, but a workaround is to simply pad with 0 (or use more complicated extensions) to have vector with size that are power of 2.

Denoting $C(N)$ the numerical complexity (number of elementary operations) associated to the computation of $\hat{f}$, one thus has

$$C(N) = 2C(N/2) + NK \tag{2.10}$$

where K is the complexity of N complex additions and $N/2$ multiplications. Making the change of variable

$$\ell \overset{\text{def.}}{=} \log_2(N) \quad \text{and} \quad T(\ell) \overset{\text{def.}}{=} \frac{C(N)}{N}$$

i.e. $C(N) = 2^\ell T(\ell)$, the relation (2.10) becomes

$$2^\ell T(\ell) = 2 \times 2^{\ell-1} T(\ell-1) + 2^\ell K \quad \Longrightarrow \quad T(\ell) = T(\ell-1) + K \quad \Longrightarrow \quad T(\ell) = T(0) + K\ell$$

and using the fact that $T(0) = C(1)/1 = 0$, one obtains

$$C(N) = KN \log_2(N).$$

This complexity should be contrasted with the complexity $O(N^2)$ of directly computing the N coefficients (2.8), each involving a sum of size N.

2.3.4 Finite convolution

For $(f,g) \in (\mathbb{R}^N)^2$, one defines $f \star g \in \mathbb{R}^N$ as

$$\forall\, n = 0, \ldots, N-1, \quad (f \star g)_n \overset{\text{def.}}{=} \sum_{k=0}^{N-1} f_k g_{n-k} = \sum_{k+\ell=n} f_k g_\ell \tag{2.11}$$

where one should be careful that here $+$ and $-$ should be understood modulo N (vectors should be seen as being defined on the group $\mathbb{Z}/N\mathbb{Z}$, or equivalently, one uses periodic boundary conditions). This defines an

commutative algebra structure $(\mathbb{R}^N, +, \star)$, with neutral element the "Dirac" $\delta_0 \stackrel{\text{def.}}{=} (1, 0, \ldots, 0)^\top \in \mathbb{R}^N$. The following proposition shows that $\mathcal{F} : f \mapsto \hat{f}$ is an algebra bijective isometry (up to a scaling by $\sqrt{N}$ of the norm) mapping to $(\mathbb{R}^N, +, \odot)$ with neutral element $\mathbb{1}_N = (1, \ldots, 1) \in \mathbb{R}^N$.

Proposition 5. *One has* $\mathcal{F}(f \star g) = \hat{f} \odot \hat{g}$.

Proof. We denote $T : g \mapsto f \star g$. One has

$$(T\varphi_\ell)_n = \sum_k f_k e^{\frac{2\mathrm{i}\pi}{N}\ell(n-k)} = e^{\frac{2\mathrm{i}\pi}{N}\ell n}\hat{f}_\ell.$$

This shows that $(\varphi_\ell)_\ell$ are the N eigenvectors of T with associated eigenvalues $\hat{f}_\ell$. So T is diagonalizable in this basis. Denoting $F = (e^{-\frac{2\mathrm{i}\pi}{N}kn})_{k,n}$ the matrix of the Fourier transform, the Fourier inversion formula (2.9) reads $F^{-1} = \frac{1}{N}F^*$ where $F^* = \bar{F}^\top$ is the adjoint matrix (trans-conjugate). The diagonalization of T now reads

$$T = F^{-1}\operatorname{diag}(\hat{f})F = \quad \Longrightarrow \quad \mathcal{F}(Tg) = \operatorname{diag}(\hat{f})Fg \quad \Longrightarrow \quad \mathcal{F}(f \star g) = \operatorname{diag}(\hat{f})\hat{g}.$$

□

This proposition shows that one can compute in $O(N \log(N))$ operation via the formula

$$f \star g = \mathbb{F}^{-1}(\hat{f} \odot \hat{g}).$$

This is very advantageous with respect to the naive implementation of formula (2.11), in the case where f and g have large support. In case where $|\operatorname{Supp}(g)| = P$ is small, then direct implementation is $O(PN)$ which might be advantageous. An example is $g = [1, 1, 0, \ldots, 0, 1]/3$, the moving average, where

$$(f \star g)_n = \frac{f_{n-1} + f_n + f_{n+1}}{3}$$

needs $3N$ operations.

An example of application of the FFT is the multiplication of large polynomial, and thus of large integers (viewing the expansion in a certain basis as a polynomial). Indeed

$$(\sum_{i=0}^{A} a_i X_i)(\sum_{j=0}^{B} b_i X^j) = \sum_{k=0}^{A+B}(\sum_{i+j=k} a_i b_j)X^k$$

One can write $\sum_{i+j=k} a_i b_j = (\bar{a} \star \bar{b})_k$ when one defines $\bar{a}, \bar{b} \in \mathbb{R}^{A+B}$ by zero padding.

2.4 Discretisation Issues

Beside computing convolutions, another major application of the FFT is to approximate the Fourier transform and its inverse, thus leading to a computationally efficient spectral interpolation method.

2.4.1 Fourier approximation via spatial zero padding.

It is possible to view the discrete finite Fourier transform (2.8) as a first order approximation to compute Fourier coefficients, or rather actually samples from the Fourier transform (1.2). Supposing that f is a smooth enough function is supported on $[0, 1]$, we consider the discrete Fourier transform of the vector $f^Q \stackrel{\text{def.}}{=} (f(n/N))_{n=0^{Q-1}} \in \mathbb{R}^Q$ where $Q \geqslant N$ induced a padding by 0 (since $f(n/N) = 0$ for $n > N$)

$$\forall\, k \in [\![-\frac{Q}{2}, \frac{Q}{2}]\!], \quad \frac{1}{N}\hat{f}^Q_k = \frac{1}{N}\sum_{n=0}^{N-1} f\left(\frac{n}{N}\right) e^{-\frac{2\mathrm{i}\pi}{Q}nk} \approx \int_0^1 f(x)e^{-\frac{2k\mathrm{i}\pi}{T}x}\mathrm{d}x = \hat{f}\left(\frac{2k\pi}{T}\right) \quad \text{where} \quad T \stackrel{\text{def.}}{=} \frac{Q}{N}.$$

The approximation is first order accurate, i.e. $O(1/N)$ for a C^1 function f. Increasing the amount Q of zero padding is a way to compute larger frequencies. Increasing the discretization precision N is on contrary a way to increase the precision of the Fourier sampling (using smaller step size $2\pi/T$).

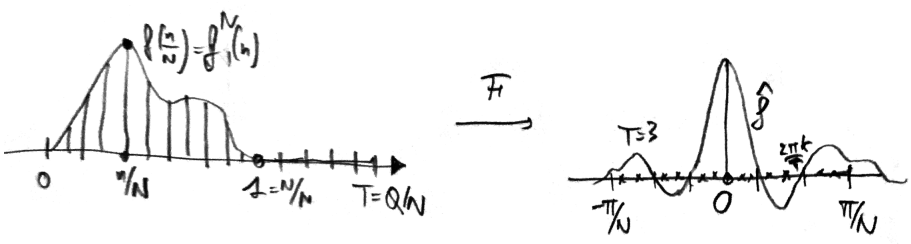


Figure 2.10: Fourier transform approximation by zero-padding in the spatial domain.

2.4.2 Fourier approximation via spatial zero padding.

If one has at its disposal N uniform discrete samples $f^N = (f^N_n)_{n=0}^{N-1}$, one can compute its discrete Fourier transform $\mathcal{F}(f^N) = \hat{f}^N$ (in $O(N\log(N))$ operations with the FFT),

$$\hat{f}^N_k \overset{\text{def.}}{=} \sum_{n=0}^{N-1} f^N_n e^{-\frac{2\mathrm{i}\pi}{N}nk},$$

and then zero-pad it to obtain a vector of length Q. For simplicity, we assume $N = 2N'+1$ is odd, and this computation can be also done (but is more involved) with even size. Indexing the frequencies as $-N' \leqslant k \leqslant N'$ The padding vector is of the form,

$$\tilde{f}^Q \overset{\text{def.}}{=} (0,\dots,0,\hat{f}^N,0,\dots,0) \in \mathbb{R}^Q$$

One can then compute the (with a normalization constant Q/N) inverse discrete Fourier transform of size Q (in $O(Q\log(Q))$ operations with the FFT) to obtain

$$\begin{aligned}
\frac{Q}{N}\mathcal{F}^{-1}(\tilde{f}^Q)_\ell &= \frac{Q}{N}\times\frac{1}{N}\sum_{k=-N'}^{N'}\hat{f}^N_k e^{\frac{2\mathrm{i}\pi}{Q}\ell k} = \frac{1}{N}\sum_{k=-N'}^{N'}\sum_{n=0}^{N-1} f^N_n e^{\frac{2\mathrm{i}\pi}{N}nk}e^{\frac{2\mathrm{i}\pi}{Q}\ell k}\\
&= \sum_{n=0}^{N-1} f^N_n \frac{1}{N}\sum_{k=-N'}^{N'} e^{2\mathrm{i}\pi(-\frac{n}{N}+\frac{\ell}{Q})k} = \sum_{n=0}^{N-1} f^N_n \frac{\sin\left[\pi N\left(\frac{\ell}{Q}-\frac{n}{N}\right)\right]}{N\sin\left[\pi\left(\frac{\ell}{Q}-\frac{n}{N}\right)\right]}\\
&= \sum_{n=0}^{N-1} f^N_n \,\mathrm{sinc}_N\left(\frac{\ell}{T}-n\right) \quad \text{where} \quad T \overset{\text{def.}}{=} \frac{Q}{N} \quad \text{and} \quad \mathrm{sinc}_N(u) \overset{\text{def.}}{=} \frac{\sin(\pi u)}{N\sin(\pi u/N)}.
\end{aligned}$$

Here we use the following summation rule for geometric series for $\rho = e^{\mathrm{i}\omega}$, $a=-b$, $\omega = 2\pi\left(-\frac{n}{N}+\frac{\ell}{Q}\right)$,

$$\sum_{i=a}^{b}\rho^i = \frac{\rho^{a-\frac{1}{2}}-\rho^{b+\frac{1}{2}}}{\rho^{-\frac{1}{2}}-\rho^{\frac{1}{2}}} = \frac{\sin((b+\frac{1}{2})\omega)}{\sin(\omega/2)}.$$

This zero-padding method leads to a discrete version of the Shannon interpolation formula (1.9), which allows to comput the interpolation on a grid of size Q are cost $O(Q\log(Q))$. Increasing N increases the accuracy of the formula, since $\mathrm{sinc}_N \to \mathrm{sinc}$ as $N \to +\infty$.

2.5 Fourier in Multiple Dimensions

The Fourier transform is extended from 1-D to arbitrary finite dimension $d > 1$ by tensor product.

2.5.1 On Continuous Domains

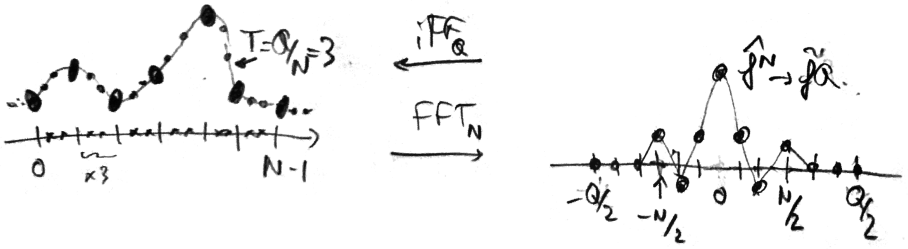


Figure 2.11: Interpolation by zero-padding in the frequency domain.

On $\mathbb{R}^d$. The crux of the power of Fourier transform in arbitrary dimension is that a product of elementary 1-D sine waves is still a sine wave

$$\prod_{\ell=1}^{d} e^{\mathrm{i}x_\ell\omega_\ell} = e^{\mathrm{i}\langle x,\,\omega\rangle}$$

moving orthogonally to the wave vector $\omega = (\omega_\ell)_{\ell=1}^d \in \mathbb{R}^d$. Here $\langle x,\,\omega\rangle = \sum_\ell x_\ell\omega_\ell$ is the canonical inner product on $\mathbb{R}^d$.

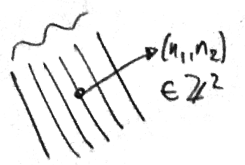


Figure 2.12: 2-D sine wave.

The definition of the Fourier transform and its inverse are

$$\forall\,\omega\in\mathbb{R}^d,\quad \hat{f}(\omega) \stackrel{\text{def.}}{=} \int_{\mathbb{R}^d} f(x)e^{-\mathrm{i}\langle x,\,\omega\rangle}\mathrm{d}x,$$
$$\forall\,x\in\mathbb{R}^d,\quad f(x) = \frac{1}{(2\pi)^d}\int_{\mathbb{R}^d} f(x)e^{\mathrm{i}\langle x,\,\omega\rangle}\mathrm{d}\omega,$$

under hypotheses of integrability matching exactly those in 1-D.

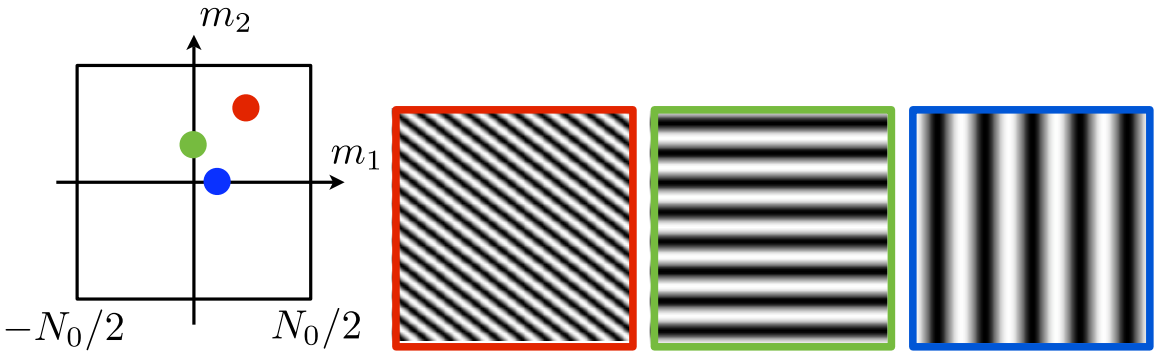


Figure 2.13: 2D Fourier orthogonal bases.

On $(\mathbb{R}/2\pi\mathbb{Z})^d$. Given an Hilbertian basis $(\varphi_{n_1})_{n_1\in\mathbb{N}}$ of $L^2(\mathbb{X})$, one construct an Hilbertian basis of $L^2(\mathbb{X}^d)$ by tensorization

$$\forall\, n=(n_1,\ldots,n_d)\in\mathbb{N}^d,\quad \forall\, x\in\mathbb{X}^d,\quad \varphi_n(x)=\varphi_{n_1}(x_1)\ldots\varphi_{n_d}(x_d). \tag{2.12}$$

Orthogonality is simple to check, and one can also prove convergence for sum of the form $\sum_{\|n\|_\infty\leqslant N}\langle f,\,\varphi_n\rangle\varphi_n \to f$ in $L^2(\mathbb{X}^d)$.

For the multi-dimensional torus $(\mathbb{R}/2\pi\mathbb{Z})^d$, using the Fourier basis (2.1), this leads to consider the basis

$$\forall\, n \in \mathbb{R}^d, \quad \varphi_n(x) = e^{\mathrm{i}\langle x,\, n\rangle}$$

which is indeed an Hilbertian orthonormal basis for the inner product $\langle f,\, g\rangle \overset{\text{def.}}{=} \frac{1}{(2\pi)^d}\int_{\mathbb{T}^d} f(x)\bar{g}(x)\mathrm{d}x$. This defines the Fourier transform and the reconstruction formula on $L^2(\mathbb{T}^d)$

$$\hat{f}_n \overset{\text{def.}}{=} \frac{1}{(2\pi)^d}\int_{\mathbb{T}^d} f(x)e^{-\mathrm{i}\langle x,\, n\rangle}\mathrm{d}x \quad \text{and} \quad f = \sum_{n\in\mathbb{Z}^d} \hat{f}_n e^{\mathrm{i}\langle x,\, n\rangle}.$$

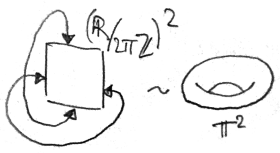


Figure 2.14: The 2-dimensional torus $\mathbb{T}^2 = (\mathbb{R}/2\pi\mathbb{Z})^2$

2.5.2 On Discrete Domains

Discrete Fourier Transform. On d-dimensional discrete domain of the form

$$n = (n_1, \ldots, n_d) \in \mathbb{Y}_d \overset{\text{def.}}{=} [\![1, N_1]\!] \times \ldots \times [\![1, N_d]\!]$$

(we denote $[\![a, b]\!] \overset{\text{def.}}{=} \{i \in \mathbb{Z}\,;\, a \leqslant i \leqslant b\}$) of $N = N_1 \ldots N_d$ points, with periodic boundary conditions, one defines an orthogonal basis $(\varphi_k)_k$ by the same tensor product formula as (2.12) but using the 1-D discrete Fourier basis (2.8)

$$\forall\,(k, n) \in \mathbb{Y}_d^2, \quad \varphi_k(n) = \varphi_{k_1}(n_1)\ldots\varphi_{k_d}(n_d) = \prod_{\ell=1}^{d} e^{\frac{2\mathrm{i}\pi}{N_\ell}k_\ell n_\ell} = e^{2\mathrm{i}\pi\langle k,\, n\rangle_{\mathbb{Y}_d}} \tag{2.13}$$

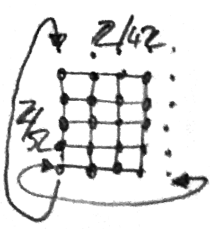


Figure 2.15: Discrete 2-D torus.

where we used the (rescaled) inner product

$$\langle k,\, n\rangle_{\mathbb{Y}_d} \overset{\text{def.}}{=} \sum_{\ell=1}^{d} \frac{k_\ell n_\ell}{N_\ell}. \tag{2.14}$$

The basis $(\varphi_k)_k$ is indeed orthonormal for this inner product. The Fourier transform gathers inner products in this basis, and (similarly to the 1-D case) the convention is to not normalize them with $(N_\ell)_\ell$, so that

$$\forall\, k \in \mathbb{Y}_d, \quad \hat{f}_k \overset{\text{def.}}{=} \sum_{n\in\mathbb{Y}_d} f_n e^{-\mathrm{i}\langle k,\, n\rangle_{\mathbb{Y}_d}},$$
$$\forall\, n \in \mathbb{Y}_d, \quad f_n = \frac{1}{N}\sum_{k\in\mathbb{Y}_d} \hat{f}_k e^{\mathrm{i}\langle k,\, n\rangle_{\mathbb{Y}_d}}.$$

Fast Fourier Transform. We detail the algorithm in dimension $d = 2$ for notation simplicity, but this extends similarly in arbitrary dimension. The general idea is that if a fast algorithm is available to compute ortho-decompositions on two 1-D bases $(\varphi^1_{k_1})_{k_1=1}^{N_1}$, $(\varphi^2_{k_2})_{k_2=1}^{N_2}$, is extended to compute decomposition on the tensor product basis $(\varphi^1_{k_1}\otimes\varphi^2_{k_2})_{k_1,k_2}$ by apply succesively the algorithm on the "rows" and then "columns" (the order does not matters) of the matrix $(f_n)_{n=(n_1,n_2)} \in \mathbb{R}^{N_1\times N_2}$. Indeed

$$\forall\, k = (k_1, k_2), \quad \langle f,\, \varphi^1_{k_1}\otimes\varphi^2_{k_2}\rangle = \sum_{n=(n_1,n_2)} f_n\varphi^1_{k_1}(n_1)\varphi^1_{k_1}(n_2) = \sum_{n_1}\left(\sum_{n_2} f_{n_1,n_2}\varphi^1_{k_1}(n_2)\right)\varphi^1_{k_1}(n_1).$$

Denoting $C(N_1)$ the complexity of the 1-D algorithm on $\mathbb{R}^{N_1}$, the complexity of the resulting 2-D decomposition is $N_2C(N_1) + N_1C(N_2)$, and hence for the FFT, it is $O(N_1N_2\log(N_1N_2)) = O(N\log(N))$ for $N = N_1N_2$.

If we represent $f \in \mathbb{R}^{N_1 \times N_2}$ as a matrix, and denote $F_N = (e^{-\frac{2\mathrm{i}\pi}{N}kn})_{k,n}$ the Fourier transform matrix (or the matrix where rows are the φ_k^*), then one can compute the 2-D Fourier transform as matrix-matrix products

$$\hat{f} = F_{N_1} \times f \times F_{N_2}^* \in \mathbb{R}^{N_1 \times N_2}.$$

But of course these multiplications are not computed explicitly (one uses the FFT).

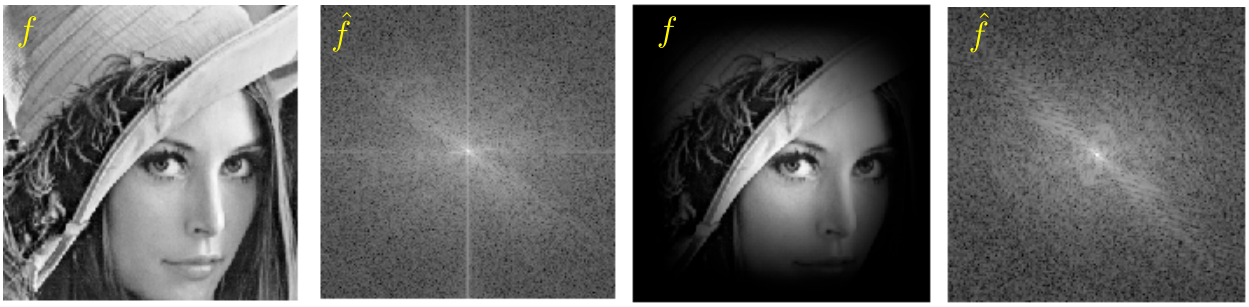


Figure 2.16: 2D Fourier analysis of a image (left), and attenuation of the periodicity artifact using masking (right).

2.5.3 Shannon sampling theorem.

The sampling Theorem 1 extends easily to $\mathbb{R}^d$ by tensorization, assuming that the sampling is on a uniform Cartesian grid. In 2-D for instance, if $\mathrm{supp}(\hat{f}) \subset [-\pi/s_1, \pi/s_1] \times [-\pi/s_2, \pi/s_2]$ and f is decaying fast enough,

$$\forall\, x \in \mathbb{R}^2, \quad f(x) = \sum_{n \in \mathbb{Z}^2} f(n_1 s_1, n_2 s_2)\, \mathrm{sinc}(x_1/s_1 - n_1)\, \mathrm{sinc}(x_2/s_2 - n_2) \quad \text{where} \quad \mathrm{sinc}(u) = \frac{\sin(\pi u)}{\pi u}.$$

2.5.4 Convolution in higher dimension.

Convolution on $\mathbb{X}^d$ with $\mathbb{X} = \mathbb{R}$ or $\mathbb{X} = \mathbb{R}/2\pi\mathbb{Z}$ are defined in the very same way as in 1-D (2.4) as

$$f \star g(x) = \int_{\mathbb{X}^d} f(t) g(x-t) \mathrm{d}t.$$

Similarly, finite discrete convolution of vectors $f \in \mathbb{R}^{N_1 \times N_2}$ extend formula (2.11) as

$$\forall\, n \in [\![0, N_1 - 1]\!] \times [\![0, N_2 - 1]\!], \quad (f \star g)_n \overset{\text{def.}}{=} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} f_k g_{n-k}$$

where additions and subtractions of vectors are performed modulo (N_1, N_2).

The Fourier-convolution theorem is still valid in all this cases, namely $\mathcal{F}(f \star g) = \hat{f} \odot \hat{g}$. In the finite case, this offers a fast $O(N \log(N))$ method to compute convolutions even if f and g do not have small support.

2.6 Application to ODEs and PDEs

2.6.1 On Continuous Domains

We here give only the intuition without formal proofs.

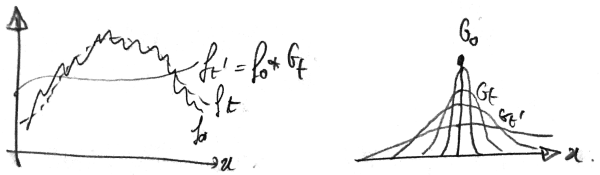


Figure 2.17: Heat diffusion as a convolution.

One $\mathbb{X} = \mathbb{R}$ or $\mathbb{T}$, one has

$$\mathcal{F}(f^{(k)})(\omega) = (\mathrm{i}\omega)^k \hat{f}(\omega).$$

Intuitively, $f^{(k)} = f \star \delta^{(k)}$ where $\delta^{(k)}$ is a distribution with Fourier transform $\hat{\delta}^{(k)}(\omega) = (\mathrm{i}\omega)^k$. Similarly on $\mathbb{X} = \mathbb{R}^d$ (see Section 2.5 for the definition of the Fourier transform in dimension d), one has

$$\mathcal{F}(\Delta f)(\omega) = -\|\omega\|^2 \hat{f}(\omega) \tag{2.15}$$

(and similarly on $\mathbb{T}$ replacing ω by $n \in \mathbb{Z}^d$). The Fourier transform (or Fourier coefficients) are thus powerful to study linear differential equations with constant coefficients, because they are turned into algebraic equations.

As a typical example, we consider the heat equation

$$\frac{\partial f_t}{\partial t} = \Delta f_t \quad \Longrightarrow \quad \forall\, \omega, \quad \frac{\partial \hat{f}_t(\omega)}{\partial t} = -\|\omega\|^2 \hat{f}(\omega).$$

This shows that $\hat{f}_t(\omega) = \hat{f}_0(\omega) e^{-\|\omega\|^2 t}$ and by inverse Fourier transform and the convolution theorem

$$f_t = G_t \star f_0 \quad \text{where} \quad G_t = \frac{1}{(4\pi t)^{d/2}} e^{-\frac{\|x\|^2}{4t}}$$

which is a Gaussian of standard deviation $\sqrt{2t}$.

2.6.2 Finite Domain and Discretization

On $\mathbb{R}/N\mathbb{Z}$ (i.e. discrete domains with periodic boundary conditions), one typically considers forward finite differences (first and second order)

$$D_1 f \overset{\text{def.}}{=} N(f_{n+1} - f_n)_n = f \star d_1 \quad \text{where} \quad d_1 = [-1, 0, \ldots, 0, 1]^\top \in \mathbb{R}^N, \tag{2.16}$$

$$D_2 f = D_1^\top D_1 f \overset{\text{def.}}{=} N^2(f_{n+1} + f_{n-1} - 2f_n)_n = f \star d_2 \quad \text{where} \quad d_2 = d_1 \star \bar{d}_1 = [2, -1, 0, \ldots, 0, -1]^\top \in \mathbb{R}^N. \tag{2.17}$$

Thanks to Proposition 5, one can alternatively computes

$$\mathcal{F}(D_2 f) = \hat{d}_2 \odot \hat{f} \quad \text{where} \quad (\hat{d}_2)_k = N^2(e^{\frac{2\mathrm{i}\pi}{N}} + e^{-\frac{2\mathrm{i}\pi}{N}} - 2) = -4N^2 \sin\left(\frac{\pi k}{N}\right)^2. \tag{2.18}$$

For $N \gg k$, one thus has $(\hat{d}_2)_k \sim -(2\pi k)^2$ which matches the scaling of (2.15).

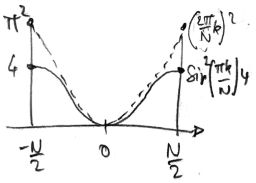


Figure 2.18: Comparison of the spectrum of Δ and D_2.

2.7 A Bit of Group Theory

The reference for this section is [33].

2.7.1 Characters

For $(G,+)$ a commutative group, a character is a group morphism $\chi : (G,+) \to (\mathbb{C}^*,\cdot)$, i.e. is satisfies

$$\forall\,(n,m)\in G,\quad \chi(n+m)=\chi(n)\chi(m).$$

The set of characters is the so-called dual $(\hat{G},\odot)$ and is a group for the pointwise multiplication $(\chi_1\odot\chi_2)(n)\overset{\text{def.}}{=}\chi_1(n)\chi_2(n)$. Indeed, the inverse of a character χ is $\chi^{-1}(n)=\chi(-n)$.

Note that for a finite group G with $|G|=N$, then since $N\times n=0$ for any $n\in G$, then $\chi(n)^N=\chi(Nn)=\chi(0)=1$, so that characters assume values in the unit circle, and more precisely

$$\chi(n)\in\left\{e^{\frac{2\mathrm{i}\pi}{N}k}\;;\;0\leqslant k\leqslant N-1\right\}. \tag{2.19}$$

So in particular $\hat{G}$ is a finite group (since there is a finite number of application between two finite sets) and $\chi^{-1}=\bar{\chi}$. In the case of a cyclic group, the dual is actually simple to describe.

Proposition 6. *For $G=\mathbb{Z}/N\mathbb{Z}$, then $\hat{G}=(\varphi_k)_{k=0}^{N-1}$ where $\varphi_k=(e^{\frac{2\mathrm{i}\pi}{N}nk})_n$ and $k\mapsto\varphi_k$ defines a (non-canonical) isomorphism $G\sim\hat{G}$.*

Proof. The φ_k are indeed characters.

Conversely, for any $\chi\in\hat{G}$, according to (2.19), $\chi(1)=e^{\frac{2\mathrm{i}\pi}{N}k}$ for some k. Then

$$\chi(n)=\chi(1)^n=^{\frac{2\mathrm{i}\pi}{N}kn}=\varphi_k(n).$$

Note that all these applications are different (because $\varphi_k(1)$ are all distincts) which shows that $|G|=|\hat{G}|$ so that they are isomorphic. □

This proposition thus shows that characters of cyclic groups are exactly the discrete Fourier orthonormal basis defined in (2.8).

Commutative groups. For more general commutative groups with a finite number of generators, according to the celebrated structure theorem, one can "decompose" them as a product of cyclic group (which are in some sense the basic building blocks), i.e. there is the following isomorphism of groups

$$G\sim(\mathbb{Z}/N_1\mathbb{Z})\times\ldots\times(\mathbb{Z}/N_d\mathbb{Z})\times\mathbb{Z}^Q. \tag{2.20}$$

If G is finite, then $Q=0$ and $N=N_1\times N_d$. In this case, G is simply a discrete d-dimensional "rectangle" with periodic boundary conditions.

For two finite groups (G_1,G_2) one has

$$\widehat{G_1\times G_2}=\hat{G}_1\otimes\hat{G}_2=\left\{\chi_1\otimes\chi_2\;;\;(\chi_1,\chi_2)\in\hat{G}_1\times\hat{G}_2\right\}. \tag{2.21}$$

Here $\otimes$ is the tensor product of two functions

$$\forall\,(n_1,n_2)\in G_1\times G_2,\quad(\chi_1\otimes\chi_2)(n_1,n_2)\overset{\text{def.}}{=}\chi_1(n_1)\chi_2(n_2).$$

Indeed, one verifies that $\chi_1\otimes\chi_2$ is a morphism, and in fact one has the factorization $\chi=\chi(\cdot,0)\otimes\chi(0,\cdot)$ because one decomposes $(n_1,n_2)=(n_1,0)+(0,n_2)$.

This construction, thanks to the structure theorem, leads to a constructive proof of the isomorphism theorem.

Proposition 7. *If G is commutative and finite then $\hat{G}\sim G$.*

Proof. The structure theorem (2.20) for $Q = 0$ and the dual of a product (2.21) shows that

$$\hat{G} \sim \hat{G}_1 \otimes \ldots \otimes \hat{G}_d$$

where we denoted $G_\ell \overset{\text{def.}}{=} \mathbb{Z}/N_\ell\mathbb{Z}$. One then remark that $\hat{G}_1 \otimes \hat{G}_2 \sim G_1 \times \hat{G}_2$. One conclude thanks to Proposition 6, since one has $\hat{G}_k \sim G_k$. □

Note that the isomorphism $\hat{G} \sim G$ is not "canonical" since it depends on the indexing of the roots of unity on the circle. Similarly to the case of duality of vector space, the isomorphism $\hat{\hat{G}} \sim G$ can be made canonical by considering the evaluation map

$$g \in G \longmapsto e_g \in \hat{\hat{G}} \quad \text{where} \quad \Big(e_g : \chi \in \hat{G} \mapsto \chi(g) \in \mathbb{C}^*.\Big)$$

Discrete Fourier transform from character's point of view. One can be even more constructive by remarking that characters in $\hat{G}_\ell$ are the discrete Fourier atoms (2.8), i.e. are of the form

$$(e^{\frac{2\mathrm{i}\pi}{N_\ell} k_\ell n_\ell})_{n_\ell=0}^{N_\ell-1} \quad \text{for some} \quad 0 \leqslant k_\ell < N_\ell.$$

Identifying G and $G_1 \times \ldots \times G_d$, by tensorizing these functions together, one thus obtains that the characters composing $\hat{G}$ are exactly the orthogonal multi-dimensional discrete Fourier basis (2.13).

2.7.2 More General cases

Infinite groups. For an infinite group with a finite number of generator, one has $Q > 0$, and the definition of $\hat{G}$ should impose the continuity of the characters (and also use an invariant measure on G to define inner products). In the case $G = \mathbb{Z}$, the dual are indexed by a continuous parameter,

$$\hat{\mathbb{Z}} = \big\{\varphi_\omega : n \mapsto e^{\mathrm{i}n\omega} \in L^2(\mathbb{R}/2\pi\mathbb{Z}) \;;\; \omega \in \mathbb{R}/2\pi\mathbb{Z}\big\}$$

so that $\hat{\mathbb{Z}} \sim \mathbb{R}/2\pi\mathbb{Z}$. The case $G = \mathbb{Z}^Q$ follows by tensorization. The $(\varphi_\omega)_\omega$ are "orthogonal" in the sense that $\langle \varphi_\omega, \varphi_{\omega'} \rangle_{\mathbb{Z}} = \delta(\omega - \omega')$ can be understood as a Dirac kernel (this is similar to the Poisson formula), where $\langle u, v \rangle_{\mathbb{Z}} \overset{\text{def.}}{=} \sum_n u_n \bar{v}_n$. The "decomposition" of a sequence $(c_n)_{n\in\mathbb{Z}}$ on the set of characters is equivalent to forming a Fourier series $\sum_n c_n e^{-\mathrm{i}n\omega}$.

Similarly, for $G = \mathbb{R}/2\pi\mathbb{Z}$, one has $\hat{G} = \mathbb{Z}$, with orthonormal characters $\varphi_n = e^{\mathrm{i}\cdot n}$, so that the decomposition of functions in $L^2(G)$ is the computation of Fourier coefficients.

Non-commutative groups. For non-commutative group, one also observe that G is not isometric to $\hat{G}$. A typical example is the symmetric group Σ_N of N elements, where one can show that $\hat{G} = \{\mathrm{Id}, \varepsilon\}$ where $\varepsilon(\sigma) = (-1)^q$ is the signature, where q is the number of permutations involved in a decomposition of $\sigma \in \Sigma_N$.

In order to study non-commutative groups, one has to replace morphisms $\chi : G \to \mathbb{C}^*$ by morphisms $\rho : G \to \mathrm{GL}(\mathbb{C}^d)$ for some d, which are called "representations" of the group G. For $(g, g') \in G$ (denoting now multiplicatively the operation on G), one should thus have $\rho(gg') = \rho(g) \circ \rho(g')$. When $d = 1$, identifying $\mathrm{GL}(\mathbb{C}) \sim \mathbb{C}^*$, one retrieve the definition of characters. Note that if ρ is a representation, then $\chi(g) \overset{\text{def.}}{=} \mathrm{tr}(\rho(g))$, where tr is the trace, defines a character. In order to limit the set of such representations, one is only interested in "elementary" ones, which does not have invariant sub-spaces, and are called "irreducible" (otherwise one could create arbitrary large representation by stacking others in a block diagonal way). One can show that the dimensions of these irreducible representations sum to N and that the entries of the matrices involved in these representation define an orthogonal basis of the space of functions $f : G \to \mathbb{C}$, thus leading to a theory extending the one on commutative groups. In particular, this Fourier transform in some sense also diagonalizes convolutions over G

$$f \star h(a) \overset{\text{def.}}{=} \sum_{bc=a} f(b)h(c).$$

For the symmetric group, there is an explicit description of the set of irreducible representations.

2.8 A Bit of Spectral Theory

In order to define Fourier methods on general domains $\mathbb{X}$, one can use the aforementioned group-theoretic approach if $\mathbb{X} = G$ is a group, or also if a group acts transitively on $\mathbb{X}$. An alternative way is to describe the equivalent of Fourier basis functions as diagonalizing a specific differential operator (as we have seen in Section 2.6 that it is in some sense a way to characterise the Fourier basis). Of particular interest is the Laplacian, since it is the lowest order rotation-invariant differential operator, and that there exists natural generalization on domains such as surfaces or graphs.

2.8.1 On a Surface or a Manifold

The presentation here is very informal. One can define the Laplacian of a smooth function $f : \mathbb{X} \to \mathbb{C}$ defined on a "surface" $\mathbb{X}$ as

$$\forall\, x \in \mathbb{X}, \quad (\Delta f)(x) \overset{\text{def.}}{=} \lim_{\varepsilon \to 0} \frac{1}{\mathrm{Vol}(B_\varepsilon(x))} \int_{B_\varepsilon(x)} f(x) \mathrm{d}\mu(x) - f(x).$$

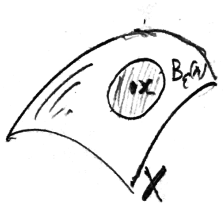


Figure 2.19: Computing Laplacian on a surface

Here $\mu(x)$ is the area measure on $\mathbb{X}$, $\mathrm{Vol}(B) \overset{\text{def.}}{=} \int_B \mathrm{d}\mu(x)$, and $B_\varepsilon(x) = \{y \,;\, d_{\mathbb{X}}(x,y) \leqslant \varepsilon\}$ is the geodesic ball of radius ε at x, where $d_{\mathbb{X}}$ is the geodesic distance on $\mathbb{X}$ (length of the shortest path).

If the surface $\mathbb{X}$ is smooth, compact and connected, then it is possible to show that Δ is itself a compact operator with a negative spectrum $0 > \lambda_1 > \lambda_2 > \ldots$ and an orthogonal set of eigenvectors $(\varphi_n)_{n \geqslant 0}$ where $\varphi_1 = 1$. Here the inner product is $\langle f, g \rangle_{\mathbb{X}} \overset{\text{def.}}{=} \int_{\mathbb{X}} f(x) g(x) \mathrm{d}\mu(x)$ on $L^2(\mathbb{X})$. In the case of a flat torus $\mathbb{X} = (\mathbb{R}/\mathbb{Z})^d$, then writing $x = (x_1, \ldots, x_d)$,

$$\Delta f = \sum_{s=1}^{d} \frac{\partial^2 f}{\partial^2 x_s}.$$

Similarly to (2.15) (which was for an unbounded domain), then one can chose for this eigen-functions φ_n the Fourier basis (2.1) and $\lambda_n = -\|n\|^2$

2.8.2 Spherical Harmonics

Of particular interest is the special case of the previous construction on the $(d-1)$-dimensional sphere $\mathbb{S}^{d-1} = \left\{x \in \mathbb{R}^d \,;\, \|x\|_{\mathbb{R}^d} = 1\right\}$. In this case, there exists a closed form expression for the eigenvectors of the Laplacian. In the 3-D case $d = 3$, they are indexed by $n = (\ell, m)$

$$\forall\, \ell \in \mathbb{N}, \quad \forall\, m = -\ell, \ldots, \ell, \quad \varphi_{\ell,m}(\theta, \varphi) = e^{\mathrm{i}m\varphi} P_\ell^m(\cos(\theta))$$

Figure 2.20: Spherical coordinates.

and then the eigenvalue of the Laplacian is $\lambda_{\ell,m} = -\ell(\ell+1)$. Here P_ℓ^m are associated Legendre polynomials, and we used spherical coordinates $x = (\cos(\varphi), \sin(\varphi)\sin(\theta), \sin(\varphi)\cos(\theta)) \in \mathbb{S}^3$ for $(\theta, \varphi) \in [0, \pi] \times [0, 2\pi]$. The index ℓ is analogous to the amplitude of Fourier frequencies in 2-D. For a fixed ℓ, the space $V_\ell = \mathrm{span}(\varphi_{\ell,m})$ is an eigenspace of Δ, and is also invariant under rotation.

2.8.3 On a Graph

We assume $\mathbb{X}$ is a graph of N vertices, simply indexed $\{1, \ldots, N\}$. Its "geometry" is described by a connectivity matrix of weights $W = (w_{i,j})_{i \sim j}$ where we denote $i \sim j$ to indicate that (i,j) is an edge of the graph for $(i,j) \in \mathbb{X}^2$. We assume that this weight matrix and the connectity is symmetric, $w_{i,j} = w_{j,i}$.

The graph Laplacian $\Delta : \mathbb{R}^N \to \mathbb{R}^N$ is computing the difference between the average of values around a point and the value at this point

$$\forall\, f \in \mathbb{R}^N, \quad (\Delta f)_i \overset{\text{def.}}{=} \sum_{j\sim i} w_{i,j} f_j - (\sum_{j\sim i} w_{i,j}) f_i \quad \Longrightarrow \quad \Delta = W - D$$

where $D \overset{\text{def.}}{=} \mathrm{diag}_i(\sum_{j\sim i} w_{i,j})$. In particular, note $\Delta \mathbb{1} = 0$

For instance, if $\mathbb{X} = \mathbb{Z}/N\mathbb{Z}$ with the graph $i \sim i-1$ and $i \sim i+1$ (modulo N), then Δ is the finite difference Laplacian operator $\Delta = D_2$ defined in (2.17). This extends to any dimension by tensorization.

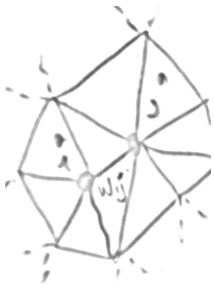


Figure 2.21: Weighted graph.

Proposition 8. *Denoting $G : f \in \mathbb{R}^N \mapsto (\sqrt{w_{i,j}}(f_i - f_j))_{i<j}$ the graph-gradient operator, one verifies that*

$$-\Delta = G^\top G \quad \Longrightarrow \quad \forall\, f \in \mathbb{R}^N, \quad \langle \Delta f,\, f\rangle_{\mathbb{R}^N} = -\langle Gf,\, Gf\rangle_{\mathbb{R}^P}.$$

where P is the number of (ordered) edges $E = \{(i,j)\ ;\ i \sim j, i < j\}$.

Proof. One has

$$\begin{aligned}
\|Gf\|^2 &= \sum_{(i,j)\in E} w_{i,j} |f_i - f_j|^2 = \sum_{i<j} w_{i,j} f_i^2 + \sum_{i<j} w_{i,j} f_j^2 - 2\sum_{i<j} w_{i,j} f_i f_j \\
&= \sum_{i<j} w_{i,j} f_i^2 + \sum_{i>j} w_{i,j} f_i^2 - \sum_{i,j} w_{i,j} f_i f_j = \sum_j f_i^2 \sum_{i,j} w_{i,j} - \sum_i f_i \sum_j w_{i,j} f_j \\
&= \langle Df,\, f\rangle - \langle Lf,\, f\rangle = -\langle Lf,\, f\rangle.
\end{aligned}$$

□

This proposition shows that Δ is a negative semi-definite operator, which thus diagonalizes in an orthobasis $(\varphi_n)_{n=1}^N$, with $\varphi_1 = 1$, with eigenvalues $0 \geqslant \lambda_1 \geqslant \lambda_N$. If $\mathbb{X}$ is connected, one can show that $\lambda_1 < 0$. In the case of a regular graph associated to a uniform grid, one retrieves the discrete Fourier basis (2.8).

More details and application of Laplacians on graphs can be found in Chapter 3, see in particular Section 3.2.5.

Chapter 3

Linear Mesh Processing

This chapter exposes the basics of surface approximation with 3D meshes and the way to process such meshes with linear operators. In particular, it studies filtering on 3D meshes and explains how a Fourier theory can be built to analyze these filters.

3.1 Surface Discretization with Triangulated Mesh

3.1.1 Continuous Geometry of Surfaces

In this course, in order to simplify the mathematical description of surfaces, we consider only globally parameterized surfaces. We begin by considering surfaces embedded in euclidean space $\mathcal{M} \subset \mathbb{R}^k$.

Definition 1 (Parameterized surface)**.** *A parameterized surface is a mapping*

$$u \in \mathcal{D} \subset \mathbb{R}^2 \mapsto \varphi(u) \in \mathcal{M}.$$

Of course, most surfaces do not benefit from such a simple parameterization. For instance, a sphere should be split into two parts in order to be mapped on two disks $\mathcal{D}_1, \mathcal{D}_2$. These topological difficulties require the machinery of manifolds in order to incorporate a set of charts $\mathcal{D} = \{\mathcal{D}_i\}_i$ that overlap in a smooth manner. All the explanations of this course extend seamlessly to this multi-charts setting.

A curve is defined in parameter domain as a 1D mapping $t \in [0,1] \mapsto \gamma(t) \in \mathcal{D}$. This curve can be traced over the surface and its geometric realization is $\bar{\gamma}(t) \stackrel{\text{def.}}{=} \varphi(\gamma(t)) \in \mathcal{M}$. The computation of the length of γ in ambient k-dimensional space $\mathbb{R}^k$ follows the usual definition, but to do the computation over the parametric domain, one needs to use a local metric defined as follow.

Definition 2 (First fundamental form)**.** *For an embedded manifold $\mathcal{M} \subset \mathbb{R}^k$, the first fundamental form is*

$$I_\varphi = \left(\langle \frac{\partial \varphi}{\partial u_i}, \frac{\partial \varphi}{\partial u_j} \rangle \right)_{i,j=1,2}.$$

This local metric I_φ defines at each point the infinitesimal length of a curve as

$$L(\gamma) \stackrel{\text{def.}}{=} \int_0^1 \|\bar{\gamma}'(t)\| \mathrm{d}t = \int_0^1 \sqrt{\gamma'(t)^{\mathrm{T}} I_\varphi(\gamma(t)) \gamma'(t)} \mathrm{d}t.$$

This fundamental form is an intrinsic invariant that does not depends on how the surfaces is isometrically embedded in space (since the length depends only on this tensor field I_φ). In contrast, higher order differential quantities such as curvature might depend on the bending of the surface and are thus usually not intrinsic (with the notable exception of invariants such as the gaussian curvature). In this course, we restrict ourselves to first order quantities since we are mostly interested in lengths and the intrinsic study of surfaces.

Example 1 (Isometry and conformality). A surface $\mathcal{M}$ is locally isometric to the plane if $I_\varphi = \mathrm{Id}_2$. This is for instance the case for a cylinder. The mapping φ is said to be conformal if $I_\varphi(u) = \lambda(u)\mathrm{Id}_2$. It means that the length of a curve over the plane is only locally scaled when mapped to the surface. In particular, the angle of two interescting curves is the same over the parametric domain and over the surface. This is for instance the case for the stereographic mapping between the plane and a sphere.

3.1.2 Discretization of Surfaces with Triangulations

Mesh Data Structure A triangulated mesh is a discrete structure that can be used to approximate a surface embedded in Euclidean space $\mathbb{R}^k$. It is composed of a topological part $M = (V, E, F)$ and a geometrical realization $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$. It is important to make the distinction between these two parts since many algorithms rely only on geometry (point clouds processings such as dimension reduction) or on topology (such as compression).

The topology M of the mesh is composed of

– *Vertices* (0D): this is an abstract set of indices $V \simeq \{1, \ldots, n\}$.

– *Edges* (1D): this is a set of pair of vertices $E \subset V \times V$. This set is assumed to be symmetric

$$(i,j) \in E \quad \Longleftrightarrow \quad i \sim j \Leftrightarrow (j,i) \in E.$$

– *Faces* (2D): this is a collection of 3-tuples of vertices $F \subset V \times V \times V$, with the additional compatibility condition

$$(i,j,k) \in F \quad \Longrightarrow \quad (i,j), (j,k), (k,i) \in E.$$

We further assumer that there is no isolated edges

$$\forall\, (i,j) \in E, \quad \exists\, k,\ (i,j,k) \in F.$$

The set of edges can be stored in a symmetric matrix $A \in \mathbb{R}^{n \times n}$ such that $A_{ij} = 1$ if $(i,j) \in E$ and $A_{ij} = 0$ otherwise. This matrix is often stored as a sparse matrix since the number of edges is usually much smaller than n^2. The set of vertices and edges form a non-oriented graph $\mathcal{G} = (V, E)$. Faces are often stored as a matrix $A_F \in \{1, \ldots, n\}^{3 \times m}$ where m is the number of faces and a column $((A_F)_{i,1}, (A_F)_{i,2}, (A_F)_{i,3})$ stores the indices of a face. In a triangulation, the face matrix A_F allows to recover the edge incidence matrix A. The face data structure allows to really capture the 2D geometry of surfaces, which is not possible with graphs alone.

The geometric realization $\mathcal{M}$ is defined through a spacial localization of the vertices (for instance in 3D space)

$$\mathcal{V} \stackrel{\text{def.}}{=} \{x_i\ ;\ i \in V\} \subset \mathbb{R}^3.$$

This allows to define a piecewise linear mesh

$$\mathcal{F} \stackrel{\text{def.}}{=} \bigcup_{(i,j,k) \in F} \mathrm{Conv}(x_i, x_j, x_k) \subset \mathbb{R}^3,$$

where the convex envelop $\mathrm{Conv}(x, y, z)$ of three points is the Euclidean triangle generated by (x, y, z).

This piecewise linear realization $\mathcal{M}$ can be displayed as a 3D surface on a computer screen. This is performed through a perspective projection of the points and a linear interpolation of color and light inside the triangle. Figure 3.1 shows an example of 3D display, with a zoom on the faces of the mesh.

Adjacency Relationships From the basis topological information given by $M = (V, E, F)$, one can deduce several adjacency data-structures that are important to navigate over the triangulation.

Definition 3 (Vertex 1-ring)**.** *The vertex 1-ring of a vertex $i \in V$ is*

$$V_i \stackrel{\text{def.}}{=} \{j \in V\ ;\ (i,j) \in E\} \subset V.\text{`} \tag{3.1}$$

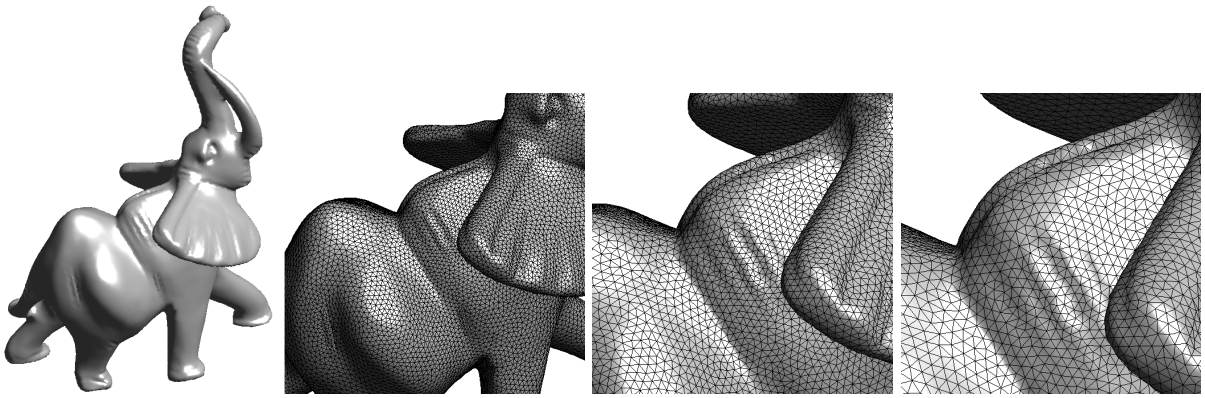


Figure 3.1: Example of display of a 3D mesh.

The s-ring is defined by induction as

$$\forall\, s > 1, \quad V_i^{(s)} = \left\{ j \in V \;;\; (k,j) \in E \quad \textit{and} \quad k \in V_i^{(s-1)} \right\}. \tag{3.2}$$

Definition 4 (Face 1-ring)**.** *The face 1-ring of a vertex $i \in V$ is*

$$F_i \stackrel{\text{def.}}{=} \{(i,j,k) \in F \;;\; i,j \in V\} \subset F.$$

The geometrical realization of a vertex 1-ring is

$$\mathcal{V}_i = \bigcup_{(i,j,k) \in V_i} \mathrm{Conv}(x_i, x_j, x_k).$$

A triangulated mesh is a manifold mesh if all the rings $\mathcal{V}_i$ for $i \in V$ are homeomorphic to either a disk (for interior vertices) or to a half disk (for boundary vertices). This ensures that the geometrical mesh really has the topology of a 2D surface embedded in $\mathbb{R}^3$ (possibly with boundaries). In particular, it implies that there is at most two faces connected to each edge

$$\forall\, (i,j) \in E, \quad \#\{k \;;\; (i,j,k) \in F\} \leqslant 2.$$

As an application of these local rings, one can compute a normal at each point using a simple rule

$$\forall\, f = (i,j,k) \in F, \quad \overrightarrow{n_f} \stackrel{\text{def.}}{=} \frac{(x_j - x_i) \wedge (x_k - x_i)}{\|(x_j - x_i) \wedge (x_k - x_i)\|}.$$

and where

$$\forall\, i \in V, \quad \overrightarrow{n_i} \stackrel{\text{def.}}{=} \frac{\sum_{f \in F_i} \overrightarrow{n_f}}{\| \sum_{f \in F_i} \overrightarrow{n_f} \|}.$$

These normals are used to define for instance a light intensity $I(i) = \max(\langle n_i, \ell(i), , \rangle 0)$, where $\ell(i)$ is the incident light. In practice one uses a infinite light source $\ell(i) = \ell$ =constant or a local spot located at position $s \in \mathbb{R}^3$ through $\ell(i) = (v_i - s)/\|v_i - s\|$. This light intensity is interpolated on the whole mesh during display.

3.2 Linear Mesh Processing

The light intensity I is a particular example of a function defined at each vertex of the mesh. Mesh processing is intended to process such functions and we thus define carefully vector spaces and operators on meshes.

3.2.1 Functions on a Mesh

In this course, a function is a discrete set of values defined at each vertex location.

Definition 5 (Linear space on a mesh). *A function on a mesh is a mapping $f \in \ell^2(\mathcal{V}) \simeq \ell^2(V) \simeq \mathbb{R}^n$ and can be viewed equivalently as*

$$f : \left\{ \begin{array}{ccc} \mathcal{V} & \longrightarrow & \mathbb{R} \\ x_i & \longmapsto & f(x_i) \end{array} \right. \iff f : \left\{ \begin{array}{ccc} V & \longrightarrow & \mathbb{R} \\ i & \longmapsto & f_i \end{array} \right. \iff f = (f_i)_{i \in V} \in \mathbb{R}^n.$$

The linear space of the functions on a mesh is equipped with an Hilbert space structure that allows to quantify approximation error and compute projections of functions.

Definition 6 (Inner product and norm). *One defines the following inner product and norm for vector $f, g \in \mathbb{R}^n$*

$$\langle f, g \rangle \stackrel{\text{def.}}{=} \sum_{i \in V} f_i g_i \quad \textit{and} \quad \|f\|^2 = \langle f, f \rangle.$$

In order to modify (process) functions on a mesh (such as a light intensity I), this course considers only linear operations that are defined through a large matrix.

Definition 7 (Linear operator A). *A linear operator A is defined as*

$$A : \ell^2(V) \to \ell^2(V) \iff A = (a_{ij})_{i,j \in V} \in \mathbb{R}^{n \times n} \textit{ (matrix).}$$

and operate on a function f as follow

$$(Af)(x_i) = \sum_{j \in V} a_{ij} f(x_j) \iff (Af)_i = \sum_{j \in V} a_{ij} f_j.$$

Example 2. If the coordinates of the point of a mesh are written $x_i = (x_i^1, x_i^2, x_i^3) \in \mathbb{R}^3$, then the X-coordinate defines a function $f : i \in V \mapsto x_i^1 \in \mathbb{R}$. A geometric mesh $\mathcal{M}$ is thus 3 functions defined on M.

Mesh processing is the task of modifying functions $f \in \ell^2(V)$. For instance, one can denoise a mesh $\mathcal{M}$ as 3 functions on M. The usual strategy applies a linear operator $f \mapsto Af$. Sometimes, A can computed from M only (for instance for compression) but most of the times it requires both M and $\mathcal{M}$.

3.2.2 Local Operators

In most applications, one can not store and manipulate a full matrix $A \in \mathbb{R}^{n \times n}$. Furthermore, one is usually interested in exploiting the local redundancies that exist in most usual functions $f \in \mathbb{R}^n$ defined on a mesh. This is why we restrict our attention to local operators that can be conveniently stored as sparse matrices (the zeros are not kept in memory).

Definition 8 (Local operator). *A local operator $W \in \mathbb{R}^{n \times n}$ satisfies $w_{ij} = 0$ if $(i,j) \notin E$.*

$$(Wf)_i = \sum_{(i,j) \in E} w_{ij} f_j.$$

A particularly important class of local operators are local smoothings (also called filterings) that perform a local weighted sum around each vertex of the mesh. For this averaging to be consistent, we define a normalized operator $\tilde{W}$ whose set of weights sum to one.

Definition 9 (Local averaging operator). *A local normalized averaging is $\tilde{W} = (\tilde{w}_{ij})_{i,j \in V} \geqslant 0$ where*

$$\forall\, (i,j) \in E, \quad \tilde{w}_{ij} = \frac{w_{ij}}{\sum_{(i,j) \in E} w_{ij}}.$$

It can be equivalently expressed in matrix form as

$$\tilde{W} = D^{-1} W \quad \textit{with} \quad D = \mathrm{diag}_i(d_i) \quad \textit{where} \quad d_i = \sum_{(i,j) \in E} w_{ij}.$$

The smoothing property corresponds to $\tilde{W}1 = 1$ which means that the unit vector is an eigenvector of W with eigenvalue 1.

Example 3. In practice, we use three popular kinds of averaging operators.

– *Combinatorial weights:* they depends only on the topology (V, E) of the vertex graph

$$\forall\,(i,j) \in E, \quad w_{ij} = 1.$$

– *Distance weights:* they depends both on the geometry and the topology of the mesh, but do not require faces information,

$$\forall\,(i,j) \in E, \quad w_{ij} = \frac{1}{\|x_j - x_i\|^2}.$$

– *Conformal weights:* they depends on the full geometrical realization of the 3D mesh since they require the face information

$$\forall\,(i,j) \in E, \quad w_{ij} = \cot(\alpha_{ij}) + \cot(\beta_{ij}). \tag{3.3}$$

Figure 3.2 shows the geometrical meaning of the angles α_{ij} and β_{ij}

$$\alpha_{ij} = \angle(x_i, x_j, x_{k_1}) \qquad \text{and} \qquad \beta_{ij} = \angle(x_i, x_j, x_{k_2}),$$

where $(i,j,k_1) \in F$ and $(i,j,k_2) \in F$ are the two faces adjacent to edge $(i,j) \in E$. We will see in the next section the explanation of these celebrated cotangent weights.

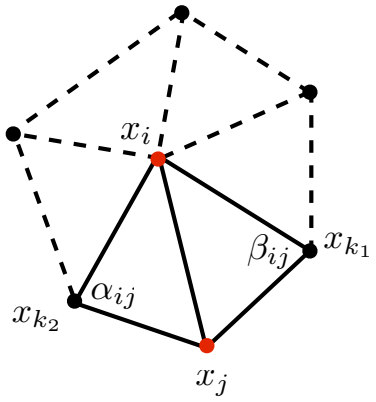


Figure 3.2: One ring around a vertex i, together with the geometrical angles α_{ij} and β_{ij} used to compute the conformal weights.

One can use iteratively a smoothing in order to further filter a function on a mesh. The resulting vectors $\tilde{W}f, \tilde{W}^2, \ldots, \tilde{W}^k f$ are increasingly smoothed version of f. Figure 3.3 shows an example of such iterations applied to the three coordinates of mesh. The sharp features of the mesh tend to disappear during iterations. We will make this statement more precise in the following, by studying the convergence of these iterations.

3.2.3 Approximating Integrals on a Mesh

Before investigating algebraically the properties of smoothing operators, one should be careful about what are these discrete operators really approximating. In order for the derivation to be simple, we make computation for a planar triangulation M of a mesh $\mathcal{M} \subset \mathbb{R}^2$.

In the continuous domain, filtering is defined through integration of functions over the mesh. In order to descretize integrals, one needs to define a partition of the plane into small cells centered around a vertex or an edge.

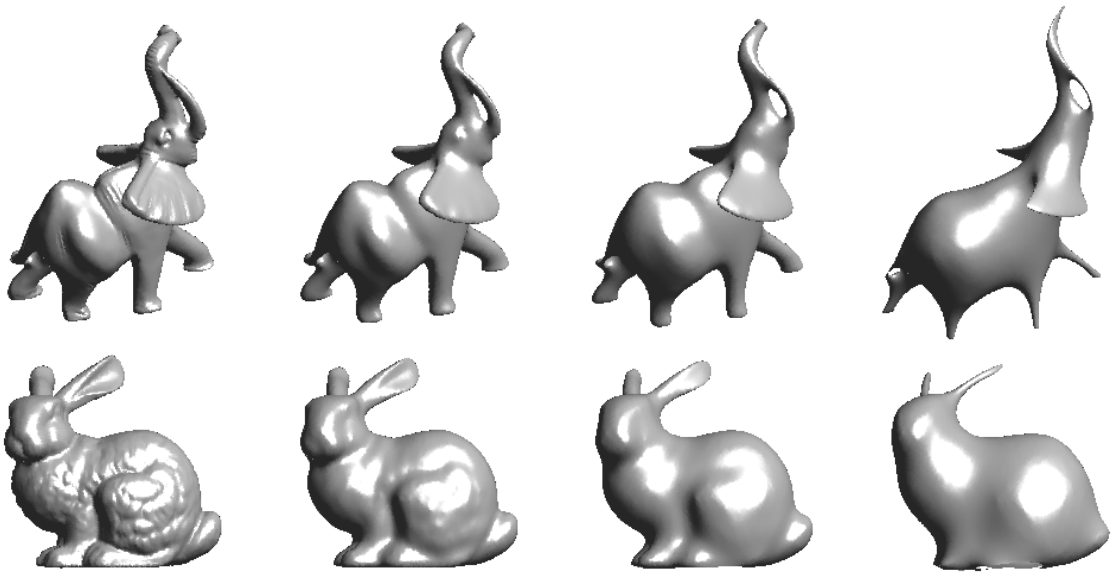


Figure 3.3: Examples of iterative smoothing of a 3D mesh.

Definition 10 (Vertices Voronoi)**.** *The Voronoi diagram associated to the vertices is*

$$\forall\, i \in V, \quad E_i = \{x \in \mathcal{M}\ ;\ \forall\, j \neq i, \|x - x_i\| \leqslant \|x - x_j\|\}$$

Definition 11 (Edges Voronoi)**.** *The Voronoi diagram associated to the edges is*

$$\forall\, e = (i,j) \in E, \quad E_e = \{x \in \mathcal{M}\ ;\ \forall\, e' \neq e, d(x,e) \leqslant d(x,e')\}$$

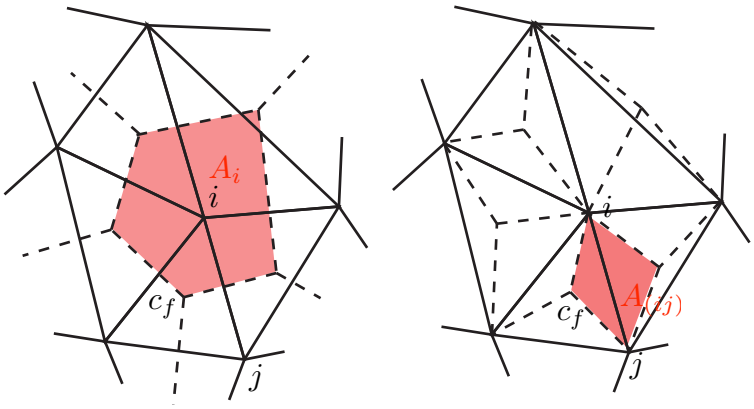


Figure 3.4: Left: vertex Voronoi cell, right: delaunay Voronoi cell. The point c_f is the orthocenter of a face $f = (i,j,k)$.

These Voronoi cells indeed form a partition of the mesh

$$\mathcal{M} = \bigcup_{i \in V} E_i = \bigcup_{e \in E} E_e.$$

The following theorem gives the formula for the area of these cells.

Theorem 3 (Voronoi area formulas). *For all $e = (i, j) \in E$, $\forall\, i \in V$, one has*

$$A_e = Area(E_e) = \frac{1}{2}\|x_i - x_j\|^2 \left(\cot(\alpha_{ij}) + \cot(\beta_{ij})\right)$$

$$A_i = Area(E_i) = \frac{1}{2}\sum_{j \in N_i} A_{(ij)}.$$

With these areas, one can approximate integrals on vertices and edges using

$$\int_{\mathcal{M}} f(x)\mathrm{d}x \approx \sum_{i \in V} A_i\, f(x_i) \approx \sum_{e=(i,j)\in E} A_e\, f([x_i, x_j]).$$

Of particular interest is the approximation of the so-called Dirichelet energy $\int_{\mathcal{M}} \|\nabla_x f\|^2 \mathrm{d}x$. In order to compute it on a triangular mesh, one can use a finite difference approximation of the gradient of a function at the point $x_{ij} = (x_i + x_j)/2$ along an edge (i, j)

$$\langle \nabla_{x_{ij}} f,\, \frac{x_i - x_j}{\|x_i - x_j\|} \rangle \approx \frac{f(x_i) - f(x_j)}{\|x_i - x_j\|}.$$

This leads to the following approximation of the Dirichlet energy

$$\int_{\mathcal{M}} \|\nabla_x f\|^2 \mathrm{d}x \approx \sum_{(i,j)\in E} A_{(i,j)} \langle \nabla_{x_{ij}} f,\, \frac{x_i - x_j}{\|x_i - x_j\|} \rangle^2 \approx \sum_{(i,j)\in E} A_{(i,j)}\, \frac{|f(x_j) - f(x_i)|^2}{\|x_j - x_i\|^2} \tag{3.4}$$

$$= \sum_{(i,j)\in E} w_{ij} |f(x_j) - f(x_i)|^2 \qquad \text{where} \qquad w_{ij} = \cot(\alpha_{ij}) + \cot(\beta_{ij}). \tag{3.5}$$

This discrete formulation shows that the correct weights to approximate the Dirichlet energy are the cotangent one, already introduced in equation (3.3).

3.2.4 Example on a Regular Grid

A regular grid is an uniform discretization with n points of $[0, 1)$ (in 1D) or $[0, 1)^2$ (in 2D). One usually assumes periodic boundary conditions, which means that each side of the square is associated with its opposite.

Since the geometry of a regular grid is invariant under translation, local averaging operators can be computed as convolution on $D = (\mathbb{Z}/p\mathbb{Z})^d$ where $n = p^d$ for d the dimension of the domain ($d = 1$ or $d = 2$)

$$\forall\, i \in D, \quad \tilde{W} f(i) = \sum_{k \in D} f(k)\tilde{w}(i - k),$$

where the operation $+$ and $-$ should be computed modulo p and $\tilde{w}(k) = \tilde{W}(0, k)$ is the convolution kernel.

Example 4 (Averaging). The uniform averaging filter is defined as

$$\tilde{W} f(i) = \frac{1}{|N|} \sum_{k \in N} f(i + k),$$

where N is the set of neighbors of the point 0 and $|N| = 2^d$. In this case, in dimension 1, $\tilde{w} = (1, 0, 1)/2$, where this notation assumes that $\tilde{w}$ is centered at the point 0.

In order to study translation invariant operators like local filtering, one needs to use the discrete Fourier transform that diagonalizes these operators.

Definition 12 (Discrete Fourier transform). *The 1D discrete Fourier transform $\Phi(f) \in \mathbb{C}^n$ of the vector $f \in \mathbb{C}^n$*

$$\Phi(f)(\omega) = \hat{f}(\omega) \stackrel{\text{def.}}{=} \frac{1}{n} \sum_k f_k e^{\frac{2\mathrm{i}\pi}{n} k\omega}.$$

A similar definition can be given for the 2D discrete Fourier transform. The main property of the Fourier transform is the following diagonalization result.

Theorem 4 (Convolution and Fourier). *For any vector f, one has*

$$\Phi(\tilde{W}^k f) = \Phi(\tilde{w} * \ldots * \tilde{w} * f) \quad \Longrightarrow \quad \Phi(\tilde{W}^k f)(\omega) = \widehat{\tilde{w}}(\omega)^k \, \hat{f}(\omega).$$

The main interest of this tools is that $\Phi(f)$ can be computed in $O(n \log(n))$ operations with the FFT algorithm. Using the following theorem, it gives an alternative expression of a local filtering. This expression in the Fourier domain can be used to speed up the computation of $\tilde{w} * f$ if $\tilde{w}$ has a lot of non zero entries (which is not the case in our setting of local operators). It is also useful to analyze theoretically the behavior of iterated filterings.

Theorem 5 (Convergence). *For any function f defined on a regular grid in 1D or 2D, one has*

$$\tilde{W}^k f \stackrel{k \to +\infty}{\longrightarrow} \frac{1}{|V|} \sum_{i \in V} f_i$$

This Fourier theory can only be developed for domains that have a group structure that enables translation invariant filtering. In particular, it does not carry over easily to an arbitrary surface. In the remaining, we define a corresponding theory for graphs and triangulated surfaces using the eigenvector of Laplacian operators. This Fourier transform on meshes enables the analysis of the convergence of many filtering schemes.

3.2.5 Gradients and Laplacians on Meshes

Gradient operator A gradient operator defines directional derivatives on a triangulation. It maps functions defined on vertices to functions defined on the set of oriented edges

$$\bar{E} \stackrel{\text{def.}}{=} \{(i,j) \in E \,;\, i > j\}.$$

Definition 13 (Gradient). *Given a local averaging W, the gradient operator G is defined as*

$$\forall\, (i,j) \in E,\ i < j, \quad (Gf)_{(i,j)} \stackrel{\text{def.}}{=} \sqrt{w_{ij}}(f_j - f_i) \in \mathbb{R}.$$

This mapping can be viewed equivalently as

$$\begin{array}{lll} G : \ell^2(V) \longrightarrow \ell^2(E), & \text{or} & G : \mathbb{R}^n \longrightarrow \mathbb{R}^p \quad \text{where} \quad p = |E|, \\ & \text{or} & G \in \mathbb{R}^{n \times p} \quad \text{(a matrix)}. \end{array}$$

The value of $(Gf)_e$ for an edge $e = (i,j)$ can be thought as a derivative along direction $\overrightarrow{x_i x_j}$.

Example 5. For the local averaging based on square distances, one has

$$w_{ij} = \|x_i - x_j\|^{-2}, \quad (Gf)_{(i,j)} = \frac{f(x_j) - f(x_i)}{\|x_i - x_j\|}.$$

which is exactly the finite difference discretization of a directional derivative.

One a regular grid, one can note that

– Gf discretizes $\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)^{\mathrm{T}}$.

– $G^{\mathrm{T}} v$ discretizes $\div(v) = \frac{\partial v_1}{\partial x} + \frac{\partial v_2}{\partial y}$.

Laplacian Operator A Laplacian operator is a discrete version of a second order derivative operator.

Definition 14 (Laplacian)**.** *Given a local averaging W, the Laplacian operator D is defined as*

$$L \overset{\text{def.}}{=} D - W, \quad \textit{where} \quad D = \operatorname{diag}_i(d_i), \quad \textit{with} \quad d_i = \sum_j w_{ij}.$$

In the remaining, we also make use of normalized operators, which have an unit diagonal.

Definition 15 (Normalized Laplacian)**.** *The normalized Laplacian is defined as*

$$\tilde{L} \overset{\text{def.}}{=} D^{-1/2} L D^{-1/2} = \mathrm{Id}_n - D^{-1/2} W D^{1/2} = \mathrm{Id}_n - D^{1/2} \tilde{W} D^{-1/2}.$$

This normalized Laplacian correspond to the weighted graph Laplacian used in graph theory, see for instance [13].

Remark 1. One can note that
– Laplacians are symmetric operators $L, \tilde{L} \in \mathbb{R}^{n \times n}$.
– L acts like a (second order) derivative since $L1 = 0$.
– in contrast, the normalized Laplacian is not a real derivative since $\tilde{L}1 \neq 0$ in general.

The main interest of the gradient operator is that it factorizes the Laplacian as follow.

Theorem 6 (Laplacian factorization)**.** *One has*

$$L = G^{\mathrm{T}} G \quad \textit{and} \quad \tilde{L} = (G D^{-1/2})^{\mathrm{T}} (G D^{-1/2}).$$

This theorem proves in particular that L and $\tilde{L}$ are symmetric positive definite operators. The inner product defined by the Laplacian can be expressed as an energy summed over all the edges of the mesh

$$\langle Lf,\, f \rangle = \|Gf\|^2 = \sum_{(i,j) \in E} w_{ij} \|f_i - f_j\|^2.$$

In the particular case of the cotangent weights introduced in equation (3.3), one can see that the Laplacian norm $\langle Lf,\, f \rangle$ is exactly the finite differences approximation of the continuous Dirichlet energy $\int_{\mathcal{M}} |\nabla_x f| \mathrm{d}x$ derived in equation (3.5). This is why these cotangent weights are the best choice to compute a Laplacian that truly approximates the continuous Laplace Beltrami operator (see definition 16).

A similar expression is derived for the normalized laplacian

$$\langle \tilde{L} f,\, f \rangle = \|G D^{-1/2} f\|^2 = \sum_{(i,j) \in E} w_{ij} \left\| \frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right\|^2.$$

Of particular interest for the study of filtering on meshes is the behavior of the spectrum of the Laplacian. We can first study its kernel.

Theorem 7 (Kernel of the Laplacian)**.** *If M is connected, then*

$$\textit{ker}(L) = \textit{span}(1) \quad \textit{and} \quad \textit{ker}(L) = \textit{span}(D^{1/2}).$$

3.2.6 Examples in 1D and 2D

In 1D, all local weights are equivalent since the points are equi-spaced. The corresponding Laplacian is a convolution that can be written as

$$(Lf)_i = \frac{1}{h^2} (2 f_i - f_{i+1} - f_{i-1}) = \frac{1}{h^2} f * (-1, 2, 1),$$

where it is important to remember that the notation $(-1,2,1)$ means that the vector is centered around 0.

This discrete 1D Laplacian is the finite difference approximation of the continuous Laplacian on the torus $\mathcal{T}$ of the segment $[0,1)$ modulo 1. Up to a minus sign, this Laplacian is just the second order derivative

$$L \overset{h\to 0}{\longrightarrow} -\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}(x_i)$$

One should be careful with our notation that consider positive semi-definite Laplacian, that have the opposite sign with respect to second order derivative operators (which are definite negative).

The gradient operator corresponds to a discretization of the first order derivative $f \mapsto f'$ (which is anti symmetric). The continuous counterpart of the factorization $L = G^{\mathrm{T}}G$ is the integration by part formula on the torus

$$\int_{\mathcal{T}} f''(x)g(x)\mathrm{d}x = -\int_{\mathcal{T}} -f(x)g'(x)\mathrm{d}x \quad \Longrightarrow \quad \int_{\mathcal{T}} f''(x)f(x)\mathrm{d}x = -\int_{\mathcal{T}} |f'(x)|^2 \leqslant 0.$$

The discrete Laplacian on a 2D grid can also be written as a 2D convolution

$$(Lf)_i = \frac{1}{h^2}\left(4f_i - f_{j_1} - f_{j_2} - f_{j_3} - f_{j_4}\right) = \frac{1}{h^2} f * \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

where $\{j_k\}_k$ are the four neighbors of the point i. This operator is the finite difference approximation to the continuous 2D Laplacian

$$L \overset{h\to 0}{\longrightarrow} -\frac{\partial^2 f}{\partial x^2}(x_i) - \frac{\partial^2 f}{\partial y^2}(x_i) = -\Delta f(x_i).$$

The factorization $Lf = G^{\mathrm{T}}Gf$ corresponds to the decomposition $\Delta f = \div(\nabla f)$.

3.2.7 Example of a Parametric Surface

We recall that a parameterized surface is a mapping $u \in \mathcal{D} \subset \mathbb{R}^2 \mapsto \varphi(u) \in \mathcal{M}$. Whereas the continuous Laplacian is simple to define on the plane using partial derivatives, its definition on a surface requires the intervention of an arbitrary parameterization φ which makes its expression cumbersome.

Definition 16 (Laplace-Beltrami)**.** *The Laplace-Beltrami operator on a parametric surface $\mathcal{M}$ is defined as*

$$\sqrt{g}\Delta_{\mathcal{M}} \overset{\text{def.}}{=} \frac{\partial}{\partial u_1}\left(\frac{g_{22}}{\sqrt{g}}\frac{\partial}{\partial u_1} - \frac{g_{12}}{\sqrt{g}}\frac{\partial}{\partial u_2}\right) + \frac{\partial}{\partial u_2}\left(\frac{g_{11}}{\sqrt{g}}\frac{\partial}{\partial u_2} - \frac{g_{12}}{\sqrt{g}}\frac{\partial}{\partial u_1}\right)$$

where $g = \det(I_\varphi)$ and $I_\varphi = (g_{ij})_{i,j=1,2}$.

The Laplacian is however an intrinsic operator that does not depends on the chosen parameterization, as shown by the following approximation theorem.

Remark 2 (Laplacian using averaging)*.*

$$\Delta_{\mathcal{M}} f(x) = \lim_{h\to 0} \frac{1}{|B_h(x)|}\int_{y\in\mathcal{M}} f(y)\mathrm{d}y \quad \text{where} \quad B_h(x) = \{y \,;\, d_{\mathcal{M}}(x,y) \leqslant h\}$$

where $d_{\mathcal{M}}$ is the geodesic distance on $\mathcal{M}$ and $h = \max_{(i,j)\in E} \|x_i - x_j\|$ is the discretization precision.

3.3 Diffusion and Regularization on Surfaces

3.3.1 Heat Diffusion

The main linear PDE for regularization of functions is the heat equation that governs the isotropic diffusion of the values of a function in time.

Definition 17 (Heat diffusion). $\forall t > 0$, *one defines* $F_t : M \to \mathbb{R}$ *solving*

$$\frac{\partial F_t}{\partial t} = -D^{-1}LF_t = -(\mathrm{Id}_n - \tilde{W})F_t \qquad \text{and} \qquad \forall i \in V,\ F_0(i) = f(i)$$

In order to compute numerically the solution of this PDE, one can fix a time step $\delta > 0$ and use an explicit discretization in time $\bar{F}_k$ as $F_0 = f$ and

$$\frac{1}{\delta}\left(\bar{F}_{k+1} - \bar{F}_k\right) = -D^{-1}L\bar{F}_k \quad \Longrightarrow \quad \bar{F}_{k+1} = \bar{F}_k - \delta D^{-1}L\bar{F}_k = (\mathrm{Id} - \delta)\bar{F}_k + \delta\tilde{W}\bar{F}_k. \tag{3.6}$$

If δ is small enough, one hopes that the discrete solution $\bar{F}_k$ is close to the continuous time solution F_t for $t = \delta k$. This is indeed the case as proven later in these notes.

Remark 3. In order for this scheme to be stable, one needs $\delta < 1$. This is be proven later using the extension of Fourier theory to meshes.

Remark 4. If $\delta = 1$, then the discretization of the Heat equation corresponds to iterative smoothing since $\bar{F}_k = \tilde{W}^k f$. In this case stability is not guaranteed but only pathological meshes give unstable filtering (see theorem 15).

Instead of using the explicit discretization in time (3.6), one can use an implicit scheme which compute an approximate solution $\tilde{F}_k$ at step k by solving

$$\frac{1}{\delta}\left(\tilde{F}_{k+1} - \tilde{F}_k\right) = -D^{-1}L\tilde{F}_{k+1} \quad \Longrightarrow \quad ((\delta+1)\mathrm{Id}_n - \delta\tilde{W})\tilde{F}_{k+1} = \tilde{F}_k. \tag{3.7}$$

Computing $\tilde{F}_k$ requires the solution of a sparse linear system at each step k. The implicit scheme (3.7) is thus computationally more involved than the explicit scheme (3.6). We will however see later that the implicit scheme is always stable for any value of $\delta \leqslant 1$.

Example 6 (Mesh smoothing). In order to smooth a mesh whose points are $x_i = (x_i^1, x_i^2, x_i^3)$, one can perform a heat diffusion for each component $f_i = (x_i^k), k = 1, 2, 3$. Figure 3.5 shows an example of such a smoothing.

In practice, mesh smoothing is used to denoise a function $f = f_0 + \sigma g$ where $g \in \mathbb{R}^n$ is a realization of a gaussian white noise (each entry $g(i)$ are independent and follow a gaussian law with unit variance). The difficult task it to find an optimal stopping time t to minimize $\|F_t - f_0\|$, which is not available since one does not know f_0. For uniformly smooth surfaces, the theory predicts that a linear filtering such as the heat equation requires a stopping time proportional to the noise level σ. This is however false for more complex surfaces such as the one used in computer graphics. In these case, alternate non linear diffusions such as non-linear PDE or wavelet thresholding usually perform better, see [28] for an overview of these methods in image processing.

Other differential equations. One can solve other partial differential equations involving the Laplacian over a 3D mesh $M = (V, E, F)$. For instance, one can consider the wave equation, which defines, for all $t > 0$, a vector $F_t \in \ell^2(V)$ as the solution of

$$\frac{\partial^2 F_t}{\partial t^2} = -D^{-1}LF_t \qquad \text{and} \qquad \begin{cases} F_0 = f \in \mathbb{R}^n, \\ \frac{\mathrm{d}}{\mathrm{d}t}F_0 = g \in \mathbb{R}^n, \end{cases} \tag{3.8}$$

In order to compute numerically the solution of this PDE, one can fix a time step $\delta > 0$ and use an explicit discretization in time $\bar{F}_k$ as $F_0 = f$, $F_1 = F_0 + \delta g$ and for $k > 1$

$$\frac{1}{\delta^2}\left(\bar{F}_{k+1} + \bar{F}_{k-1} - 2\bar{F}_k\right) = -D^{-1}L\bar{F}_k \quad \Longrightarrow \quad \bar{F}_{k+1} = 2\bar{F}_k - \bar{F}_{k-1} - \delta^2 D^{-1}L\bar{F}_k.$$

Figure 3.6 shows examples of the resolution of the wave equation on 3D meshes.

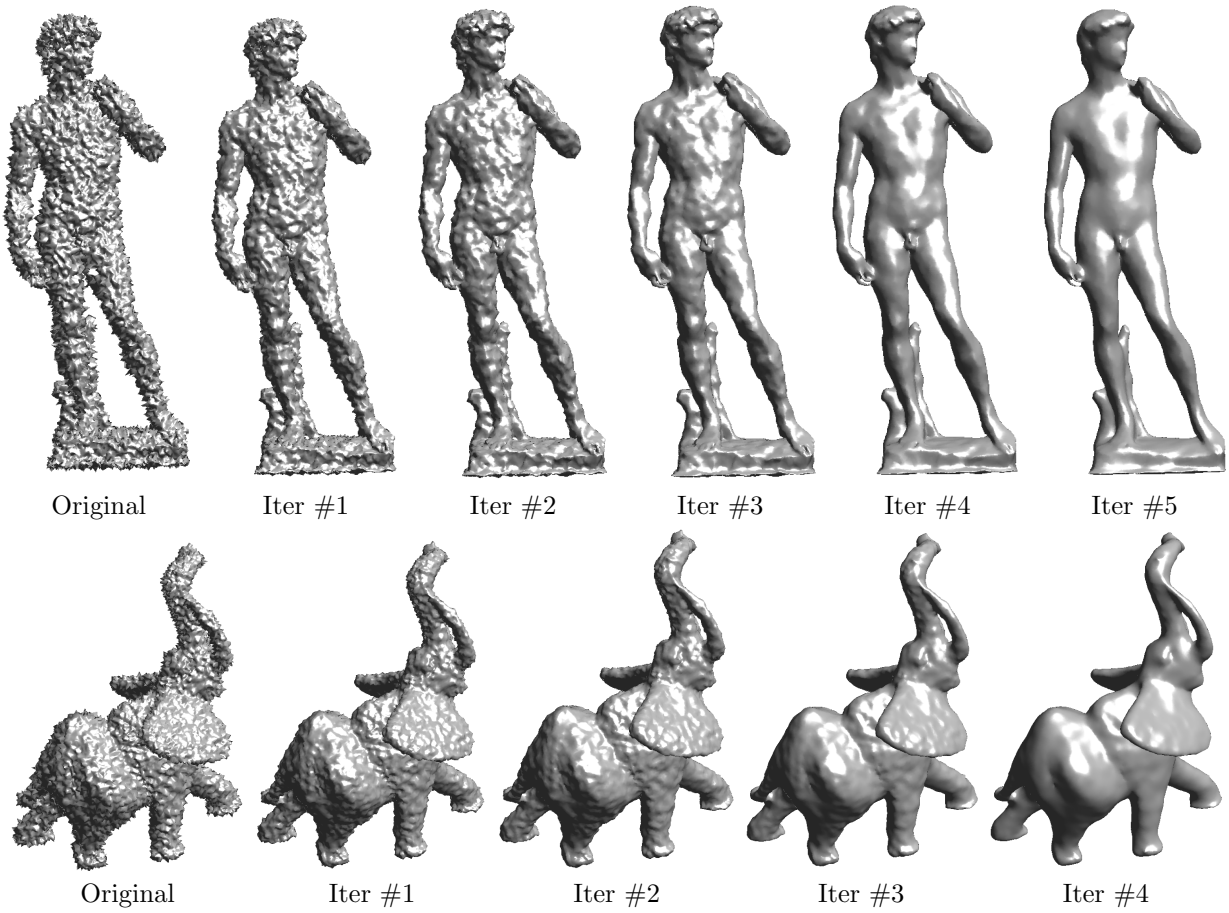


Figure 3.5: Examples of mesh denoising with the heat equation.

3.3.2 Spectral Decomposition

In order to better understand the behavior of linear smoothing on meshes, one needs to study the spectral content of Laplacian operators. This leads to the definition of a Fourier theory for meshes. The decomposition $\tilde{L} = (GD^{-1/2})^{\mathrm{T}}(GD^{-1/2})$ of the Laplacian implies that it is a positive semi-definite operator. One can thus introduce the following orthogonal factorization.

Theorem 8 (Eigen-decomposition of the Laplacian)**.** *It exists a matrix* U, $\quad U^{\mathrm{T}}U = \mathrm{Id}_n$ *such that*

$$\tilde{L} = U\Lambda U^{\mathrm{T}} \qquad \textit{where} \qquad \Lambda = \mathrm{diag}_\omega(\lambda_\omega), \qquad \lambda_1 \leqslant \ldots \leqslant \lambda_n.$$

The eigenvalues λ_ω correspond to a frequency index that ranks the eigenvectors u_ω of $U = (u_\omega)_\omega$. One can first state some bounds on these eigenvalues.

Theorem 9 (Spectral bounds)**.** $\forall\, i,\ \lambda_i \in [0, 2]$ *and*
– *If* M *is connected then* $0 = \lambda_1 < \lambda_2$.
– $\lambda_n = 2$ *if and only if* M *is 2-colorable.*

We recall the definition of a colorable graph next.

Definition 18 (Colorable graph)**.** *A graph* (V, E) *is* k*-colorable if it exist a mapping* $f : V \to \{1, \ldots, k\}$ *such that*

$$\forall\, (i, j) \in E, \quad f(i) \neq f(j).$$

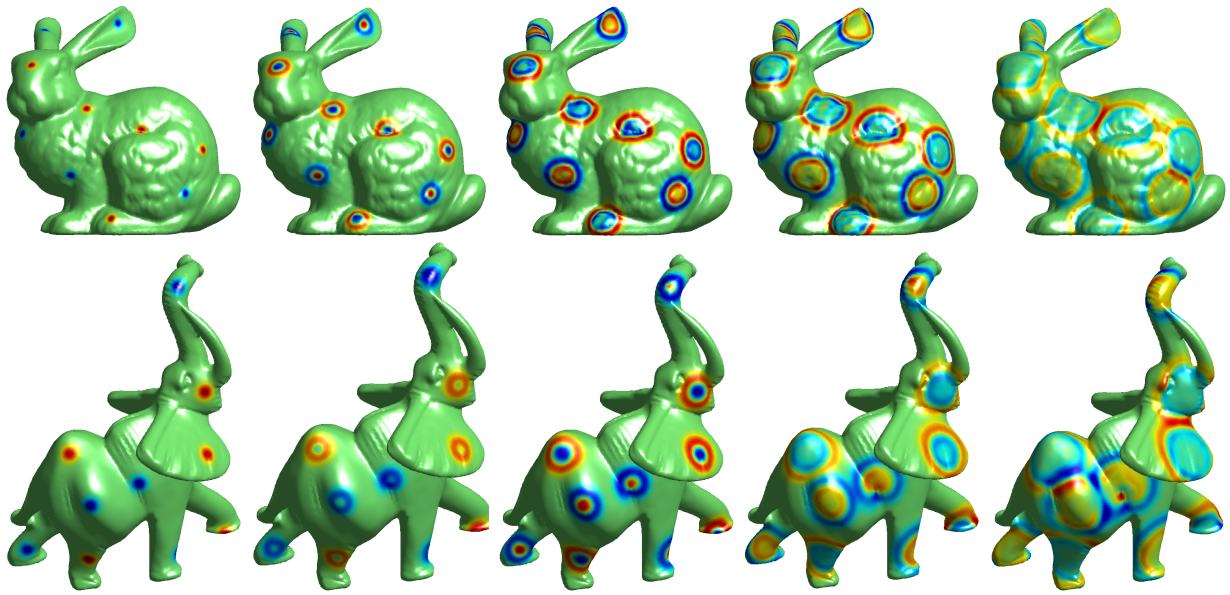


Figure 3.6: Example of evolution of the wave equation on 3D mesh. The initial condition f is a superposition of small positive and negative gaussians.

A 2-colorable graph is also called bi-partite. A 2-colorable mesh is pathological for filtering since one can split the set of vertices into two parts without inner connexions. The filtering process can oscillate by exchanging values between these sets, thus never converging.

The orthogonal eigen-basis $U = (u_\omega)_\omega$ is an orthogonal basis of the space $\mathbb{R}^n \simeq \ell^2(V)$, which can be written as

$$u_\omega : \left\{ \begin{array}{ccc} V & \longrightarrow & \mathbb{R} \\ i & \longmapsto & u_\omega(x_i) \end{array} \right.$$

The orthogonality means that $\langle u_\omega, u_{\omega'} \rangle = \delta_\omega^{\omega'}$. This basis allows to compute an orthogonal decomposition of any functions f

$$\forall\, f \in \ell^2(V), \quad f = \sum_\omega \langle f, u_\omega \rangle u_\omega.$$

Having such a tool allows to split a function f in elementary contributions $\langle f, u_\omega \rangle$ with a control in the energy because of orthogonality

$$\|f\|^2 = \sum_\omega |\langle f, u_\omega \rangle|^2.$$

Figure 3.7 shows some examples of eigenfunctions depicted using color ranging from blue (negative values of the eigenfunction) to red (positive values). One can see that these functions are oscillating, in a way similar to the traditional Fourier basis. In some sense (made more precise latter), this basis is the extension of the Fourier basis to meshes. A function u_ω corresponding to a large spectral value λ_ω is highly oscillating and corresponds thus intuitively to a high frequency atom.

Extracting numerically eigenvectors from a large matrix is a difficult problem. If the matrix is sparse, a method of choice consists in using iterative powers of a shifted version of the laplacian. One starts from a random initial vector v_0 and iterates

$$v_{k+1} = \frac{w_{k+1}}{\|w_{k+1}\|} \qquad \text{where} \qquad w_{k+1} = (\tilde{L} - \lambda \mathrm{Id}_n)^{-1} v_k. \tag{3.9}$$

These iterates converges to the eigenvectors corresponding to the eigenvalue the closest to λ, as staten in the following theorem.

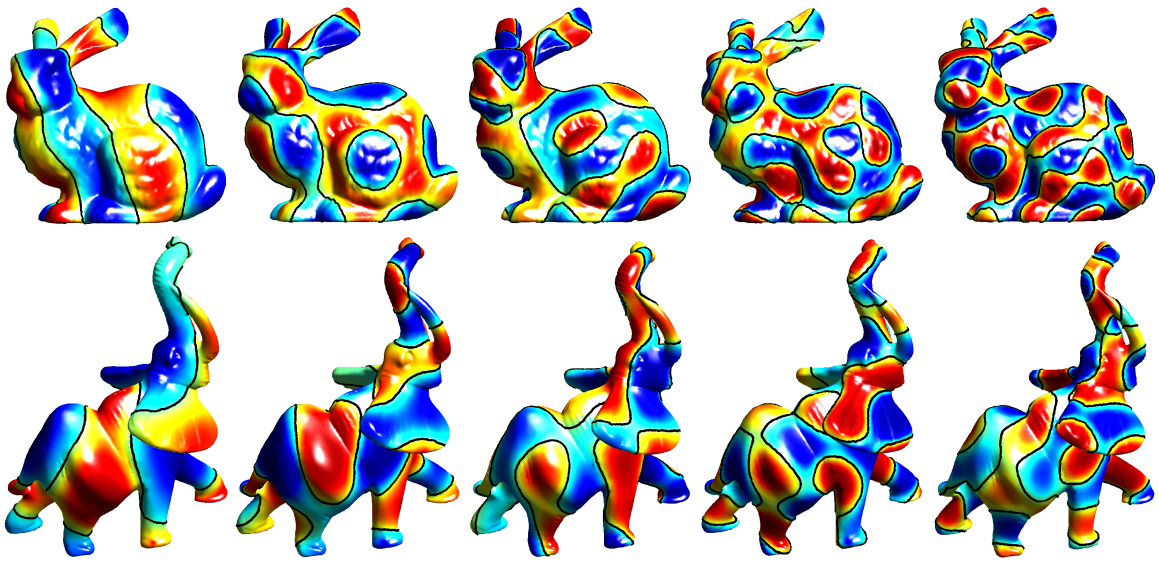


Figure 3.7: Examples of eigenvectors u_ω of the Laplacian $\tilde{L}$. The blue colors indicated negative values, red colors positive ones. The black curve is the 0 level set of the eigenvector.

Theorem 10 (Inverse iterations). *For a given shift λ, lets denote*

$$\omega^\star = \operatorname*{argmin}_\omega |\lambda - \lambda_\omega| \qquad \textit{and} \qquad \omega^+ = \operatorname*{argmin}_{\omega \neq \omega^\star} |\lambda - \lambda_\omega|$$

If $|\lambda - \lambda_{\omega^\star}| < |\lambda - \lambda_{\omega^+}|$, then

$$v_k \overset{k\to+\infty}{\longrightarrow} u_{\omega^\star} \quad \textit{and} \quad \langle L v_k, v_k \rangle \overset{k\to+\infty}{\longrightarrow} \lambda_{\omega^\star}.$$

The speed of convergence of these inverse iterations is governed by the conditioning of $(\tilde{L} - \lambda \mathrm{Id}_n)^{-1}$ since

$$\|v_k - u_{\omega^\star}\| \leqslant C \rho(\lambda)^k \qquad \text{where} \qquad \rho(\lambda) \overset{\text{def.}}{=} \frac{|\lambda - \lambda_{\omega^\star}|}{|\lambda - \lambda_{\omega^+}|} < 1.$$

The smallest $\rho(\lambda)$ is, the faster the method converges.

In order to compute an iteration (3.9) of the method, one needs to solve a sparse linear system $A w_{k+1} = v_k$ whith $A = \tilde{L} - \lambda \mathrm{Id}_n$. In order to do so, one can use a direct method such as LU factorization. The advantage of such an approach is that the factorization is computed once for all and can be re-used to solve very quickly at each step k. These factorization are however quite slow to compute especially for large matrices. For large problems, one can solve this linear system using an iterative algorithm such as conjugate gradient. These iterative method are attractive for sparse matrices, but a fast convergence requires $1/\rho(\lambda)$, the conditioning of $\tilde{L} - \lambda \mathrm{Id}_n$ to be not large, with is contradictory with the constraint for iterations 3.9 to converge fast.

3.3.3 Spectral Theory on a Regular Grid

In the particular case of a 1D or 2D lattice, the eigenfunctions defined earlier correspond exactly to the Fourier basis used in the discrete Fourier transform.

Theorem 11 (Spectrum in 1D). *For a 1D regular lattice,*

$$u_\omega(k) = \frac{1}{\sqrt{n}} \exp\left(\frac{2\mathrm{i}\pi}{n} k\omega\right) \qquad \textit{and} \qquad \lambda_\omega = 4 \sin^2\left(\frac{2\pi}{n}\omega\right).$$

Theorem 12 (Spectrum in 2D). *For a 2D regular lattice, $n = n_1 n_2$, $\omega = (\omega_1, \omega_2)$*

$$u_\omega(k) = \frac{1}{\sqrt{n}} \exp\left(\frac{2\mathrm{i}\pi}{n}\langle k, \omega\rangle\right) \qquad \text{and} \qquad \lambda_\omega = 4\left(\sin^2\left(\frac{2\pi}{n_1}\omega_1\right) + \sin^2\left(\frac{2\pi}{n_2}\omega_2\right)\right).$$

As already mentioned, on a mesh, the eigenvectors of $\tilde{L}$ correspond to a extension of the Fourier basis to meshes. The definition of the Fourier transform on meshes requires a little care since a diagonal normalization by D is used as defined next.

Definition 19 (Manifold-Fourier transform). *For $f \in \ell^2(V)$,*

$$\Phi(f)(\omega) = \hat{f}(\omega) \overset{\text{def.}}{=} \langle D^{1/2} f, u_\omega\rangle \quad \Longleftrightarrow \quad \Phi(f) = \hat{f} = U^{\mathrm{T}} D^{1/2}.$$

where $(u_\omega)_\omega$ are the eigenvectors of $\tilde{L}$.

One can note that there is still a degree of freedom in designing this Fourier transform since one can use any local weighting (for instance combinatorial, distance or conformal). Depending on the application, one might need to use weights depending only on the topology of the mesh (combinatorial for mesh compression).

A major theoretical interest of this Fourier transform is that it diagonalizes local averaging operators.

Theorem 13 (Spectral smoothing). *One has $\Phi\tilde{W}\Phi^{-1} = \mathrm{Id}_n - \Lambda$ and thus for any function f*

$$\widehat{\tilde{W} f}(\omega) = (1 - \lambda_\omega)\hat{f}(\omega)$$

This diagonalization allows to prove the convergence of iterative smoothing.

Theorem 14 (Convergence of iterated smoothing). *If $\lambda_n < 2$ (i.e. M is not 2-colorable), then for any function f*

$$\tilde{W}^k f \overset{k\to+\infty}{\longrightarrow} \frac{1}{n}\sum_{i\in V} f_i.$$

3.3.4 Spectral Resolution of the Heat Diffusion

Recall that the heat diffusion is defined as

$$\forall\, t > 0, \quad \frac{\partial F_t}{\partial t} = -D^{-1} L F_t = -(\mathrm{Id}_n - \tilde{W}) F_t$$

Using the manifold Fourier expansion $\hat{F}_t \overset{\text{def.}}{=} U^{\mathrm{T}} D^{1/2} F_t$, this differential equation can be re-written as

$$\frac{\partial \hat{F}_t(\omega)}{\partial t} = -\lambda_\omega \hat{F}_t(\omega) \quad \Longrightarrow \quad \hat{F}_t(\omega) = \exp(-\lambda_\omega t)\hat{f}(\omega). \tag{3.10}$$

This allows to study the convergence of the continuous heat equation.

Theorem 15 (Convergence of heat equation). *If $\mathcal{M}$ is connected,*

$$F_t \overset{t\to+\infty}{\longrightarrow} \frac{1}{n}\sum_{i\in V} f_i.$$

Recall that the heat equation is discretized using the following explicit and implicit schemes, equations (3.6) and (3.7)

$$\begin{cases} \bar{F}_k = (1-\delta)\bar{F}_k + \delta\tilde{W}\bar{F}_k, \\ ((1+\delta)\mathrm{Id}_n - \delta\tilde{W})\tilde{F}_{k+1} = \tilde{F}_k. \end{cases}$$

These filtering iterations can be re-written over the Fourier domain as

$$\begin{cases} \widehat{\bar{F}_{k+1}}(\omega) = (1 - \delta\lambda_\omega)\widehat{\bar{F}_k}(\omega), \\ \widehat{\tilde{F}_{k+1}}(\omega) = \frac{1}{(1+\delta\lambda_\omega)}\widehat{\bar{F}_k}(\omega). \end{cases}$$

This allows to state the stability and convergence of the finite difference discretization.

Theorem 16 (Convergence of discretization). *The explicit scheme is stable if $\delta < 1$. The implicit scheme is always stable. One has*

$$\begin{cases} \bar{F}_{t/\delta} \xrightarrow{\delta\to 0} F_t, \\ \tilde{F}_{t/\delta} \xrightarrow{\delta\to 0} F_t. \end{cases}$$

with the restriction that for the explicit scheme, the mesh must not be 2-colorable.

Other Differential Equations. The manifold Fourier transform can also be used to solve the wave equation (3.8) since

$$\frac{\partial^2 \hat{F}_t(\omega)}{\partial t^2} = -\lambda_\omega \hat{F}_t(\omega) \quad \Longrightarrow \quad \hat{F}_t(\omega) = \cos(\sqrt{\lambda_\omega} t)\hat{f}(\omega) + \frac{1}{\sqrt{\lambda_\omega}} \sin(\sqrt{\lambda_\omega} t)\hat{g}(\omega).$$

3.3.5 Quadratic Regularization

Instead of using a PDE for regularization, one can try to find a new function that is both close to the original one f and that is smooth in a certain sense. This leads to the notion of quadratic regularization, where one uses a Laplacian as a smoothness prior on the recovered function.

Definition 20 (Quadratic regularizer). *For $t > 0$, one defines*

$$F_t^q = \operatorname*{argmin}_{g\in\mathbb{R}^n} \|f - g\|^2 + t\|\tilde{G}g\|^2 \qquad \textit{where} \qquad \tilde{G} = GD^{-1/2}.$$

This optimization replaces $f \in \ell^2(V)$ by $F_t^{\mathrm{q}} \in \ell^2(V)$ with small gradients. This optimization can be found in closed form by inverting a sparse linear system.

Theorem 17 (Solution of quadratic regularization). *F_t^q is unique and*

$$F_t^q = (\mathrm{Id}_n + t\tilde{L})^{-1} f.$$

Over the Fourier domain, this inversion reads

$$\hat{F}_t^{\mathrm{q}}(\omega) = \frac{1}{1 + t\lambda_\omega} \hat{f}(\omega).$$

This corresponds to an attenuation of the high frequency content of f, in a way very similar to equation (3.10).

Once again, similarly to the heat equation, the spectral expression of the quadratic regularizer allows to study its convergence for large t.

Theorem 18 (Convergence of quadratic regularization). *If $\mathcal{M}$ is connected,*

$$F_t^q \xrightarrow{t\to+\infty} \frac{1}{n}\sum_{i\in V} f_i.$$

3.3.6 Application to Mesh Compression

We have shown how the Fourier basis on meshes can be used to compute in a diagonal fashion filtering, heat diffusion and quadratic regularization. This Fourier transform is however of little interest in practice, since the original filterings (or finite difference approximation of the heat equation) are usually faster to compute directly than over the Fourier domain. The Fourier transform is thus mainly of theoretical interest in these cases since it allows to prove convergence results.

Another class of applications makes use of an orthogonal expansion such as the Fourier one to perform mesh compression. This section shows how to compute a linear M-term approximation in this Fourier basis and to do mesh compression. We refer to the survey [1] for more advanced non-linear mesh compression methods.

The orthogonal basis $U = (u_\omega)_\omega$ of $\ell^2(V) \simeq \mathbb{R}^n$, where $\tilde{L} = U\Lambda U^{\mathrm{T}}$ allows to define a linear approximation as followed.

Definition 21 (Linear M-term approximation). *For any $M > 0$, the linear M-term approximation of f is*

$$f = \sum_{\omega=1}^{n} \langle f,\, u_\omega \rangle u_\omega \quad \overset{M\text{-term approx.}}{\Longrightarrow} \quad f_M \overset{\text{def.}}{=} \sum_{\omega=1}^{M} \langle f,\, u_\omega \rangle u_\omega.$$

The quality of the approximation is measured using the error decay, which can in turn be estimated using the removed coefficients

$$E(M) \overset{\text{def.}}{=} \|f - f_M\|^2 = \sum_{\omega > M} |\langle f,\, u_\omega \rangle|^2.$$

A good orthogonal basis U is a basis for which $E(M)$ decays fast on the signals of interest. Equivalently, a fast decay of E with M corresponds to a fast decay of $|\langle f,\, u_\omega \rangle|$ for large ω. Figure 3.8 shows the decay of the Fourier spectrum for two different functions defined on a 3D mesh. The smooth function (left in the figure) exhibits a fast decay of its spectrum, meaning that it can be well approximated with only a few Fourier coefficients.

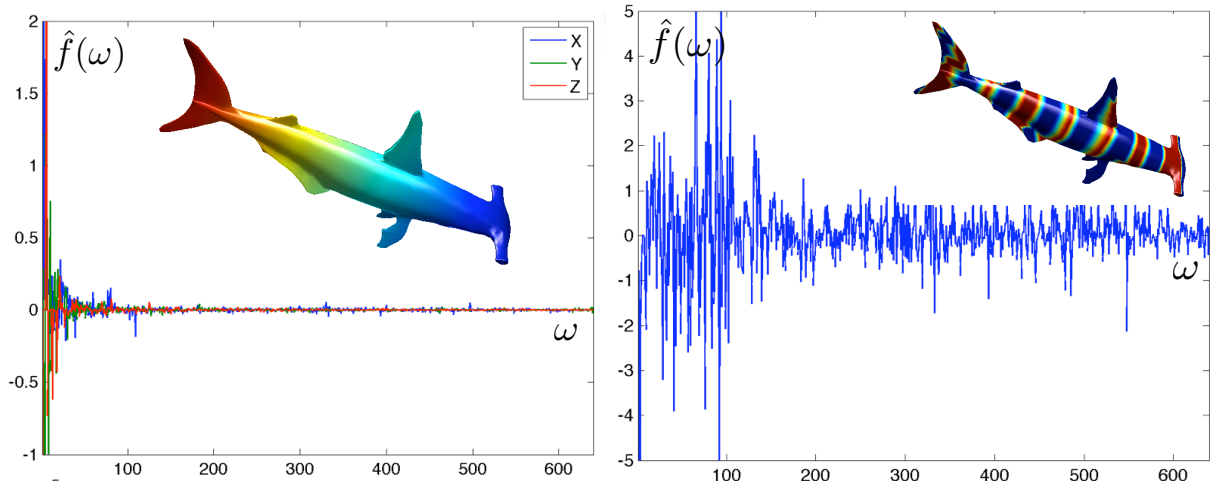


Figure 3.8: Examples of Fourier spectrum for a smooth and a non-smooth function.

We recall that the Fourier atoms

$$\forall\, \omega \in \mathbb{Z}, \quad u_\omega(x) = \frac{1}{\sqrt{2\pi}} e^{\mathrm{i}\omega x}$$

are the eigenvectors of the compact, symmetric, semi-definite negative operator $f \mapsto f''$ (that should be defined on the Hilbert space of twice Sobolev derivable functions). This set of function is also an Hilbert basis of the space $\mathrm{L}^2(\mathbb{R}/(2\pi\mathbb{Z}))$ of 2π-periodic square integrable functions and a Fourier coefficient is $\hat{f}(\omega) \overset{\text{def.}}{=} \langle f,\, u_\omega \rangle$.

Approximation theory studies this linear error decay for classical functional spaces. One can for instance study the Fourier expansion over euclidean spaces.

Theorem 19 (Fourier in 1D). *If f is C^α regular on $\mathbb{R}/(2\pi\mathbb{Z})$,*

$$\left|\hat{f}(\omega)\right| \leqslant \|f^{(\alpha)}\|_\infty |\omega|^{-\alpha}.$$

This result can be proven with a simple integration by parts. A slightly more difficult result shows that the linear approximation error decays like $M^{-\alpha}$.

Theorem 20 (Fourier approximation). *If f is C^α on $\mathbb{R}/(2\pi\mathbb{Z})$, then it exist $C > 0$ such that*

$$\sum_\omega |\omega|^{2\alpha} |\langle f, u_\omega\rangle|^2 < +\infty \quad \Longrightarrow \quad E(M) \leqslant CM^{-\alpha}.$$

This kind of results can be extended to continuous surfaces thanks to the continuous Laplacian. We suppose that $\mathcal{M}$ is a surface parameterized by φ, and a function $f = \varphi \circ \bar{f}$ is defined on it. By definition, this function f is C^α if $\bar{f}$ is C^α in euclidean space. For a compact surface $\mathcal{M}$, the Laplace-Beltrami operator $\Delta_\mathcal{M}$ is symmetric (for the inner product on the surface), is negative semi-definite and has a discrete spectrum $\Delta_\mathcal{M} u_\omega = -\lambda_\omega u_\omega$ for $\omega \in \mathbb{N}$. The functions $\{u_\omega\}_\omega$ are an orthogonal basis for function of finite energy on the surface $\mathrm{L}^2(\mathcal{M})$. The inner product of an arbitrary smooth function $f \in \mathrm{C}^\alpha(\mathcal{M})$ can be bounded using integration by parts

$$\langle f, u_\omega\rangle = \frac{1}{\lambda_\omega^k} \langle \Delta_\mathcal{M}^k f, u_\omega\rangle \quad \Longrightarrow \quad |\langle f, u_\omega\rangle| \leqslant \frac{\|f\|_{\mathrm{C}^\alpha}}{\lambda_\omega^{\alpha/2}}.$$

This proves the efficiency of the Fourier basis on surfaces to approximate smooth functions.

When computing the M-term approximation f_M of f one removes the small amplitude Fourier coefficients of the orthogonal expansion of f. Figure 3.9 shows some examples of mesh approximation where one retains an increasing number of Fourier coefficients. Mesh compression is only a step further, since one also need to code the remaining coefficients. This requires first quantifying the coefficients up to some finite precision and then binary code these coefficients into a file.

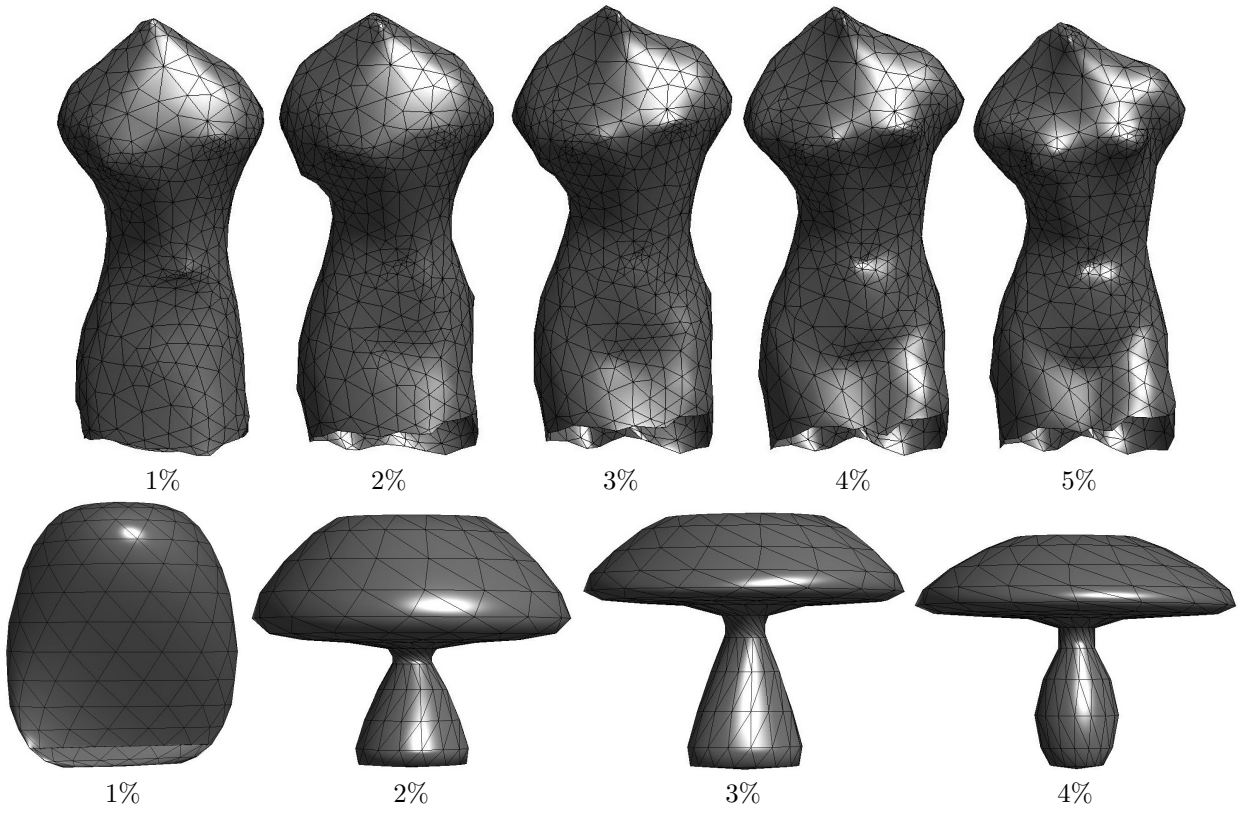


Figure 3.9: Examples of spectral mesh compression.

3.3.7 Application to Mesh Parameterization

This section is restricted to the study of meshes that can be globally parameterized on a plane. It means that they are topologically equivalent to a 2D disk. More complex meshes should be first segmented in cells that are equivalent to a disk.

A parameterization of a continuous surface $\mathcal{M}$ is a bijection

$$\psi : \mathcal{M} \longrightarrow \mathcal{D} \subset \mathbb{R}^2.$$

A similar definition applies to a discrete mesh where one computes a 2D position $\psi(i)$ for all the vertices $i \in V$ and then interpolates linearly the mapping to the whole piecewise linear geometric mesh. This section explains the basics of linear methods for mesh parameterization. We refer to various surveys [22, 41] for more details on mesh parameterization.

Usually, a 2D mesh is computed from range scanning or artistic modeling, so it does not come with such a parameterization. In order to perform texture mapping or more general mesh deformations, it is however important to use such a parameterization. Since many bijections are possible to layout the mesh in 2D, the mapping ψ has to satisfy additional smoothness assumptions. Classically, one requires that each coordinate of ψ has a vanishing Laplacian (it is thus harmonic) outside a set of constrained vertices that enforce boundary conditions.

More precisely, $\psi = (\psi_1, \psi_2)$ is the solution of

$$\begin{cases} \forall\, i \notin \partial\mathcal{M}, & (L\psi_1)(i) = (L\psi_2)(i) = 0 \\ \forall\, i \in \partial\mathcal{M}, & \psi(i) = \psi^0(i) \in \partial\mathcal{D}, \end{cases}$$

where $\partial\mathcal{M}$ is the boundary of the mesh, which consists in vertices whose face ring is not homeomorphic to a disk but rather to a half disk. This formulation requires the solution of two sparse linear systems (one for each coordinate of ψ).

The boundary condition $\psi^0(i)$ for $i \in \partial\mathcal{M}$ describes a 1D piecewise linear curve in the plane, that is fixed by the user. In the following, we will see that this curve should be convex for the parameterization to be bijective.

Remark 5. For such an harmonic parameterization, each point is the average of its neighbors since

$$\forall\, i, \quad \psi(i) = \frac{1}{\sum_j w_{ij}} \sum_{(i,j)\in E} w_{i,j}\psi(j).$$

The powerful feature of this linear parameterization method is that it can be proven to produce a valid (bijective) parameterization as long as the constrained position (boundary values of ψ) are along a convex curve.

Theorem 21 (Tutte theorem)**.** *If $\forall\,(i,j) \in E,\ w_{ij} > 0$, and if $\partial\mathcal{D}$ is a convex curve, then ψ is a bijection.*

Figure 3.10 shows several examples of parameterizations. One is free to use any laplacian (combinatorial, distance or conformal) as long as it produces positive weights. There is a issue with the conformal weights, which can be negative if the mesh contains obtuse triangles. In practice however it leads to the best results. The efficiency of a parameterization can be measured by some amount of distortion induced by the planar mapping. Linear methods cannot hope to cope with large isoperimetric distortions (for instance large extrusions in the mesh) since harmonicity leads to clustering of vertices.

3.3.8 Application to Mesh Flattening

One of the difficulty with linear parameterization methods is that they require to set up the positions of the vertices along the boundary of the mesh. In order to let the boundary free to evolve and find some optimal shape, one can replace the fixed point constraint by a global constraints of unit variance as follow

$$\min_{\psi_1,\psi_2\in\mathbb{R}^n} \|\tilde{G}\psi_1\|^2 + \|\tilde{G}\psi_2\|^2 \qquad \text{with} \qquad \begin{cases} \|\psi_i\| = 1, \\ \langle \psi_1,\, \psi_2 \rangle = 0, \\ \langle \psi_i,\, 1 \rangle = 0. \end{cases}$$

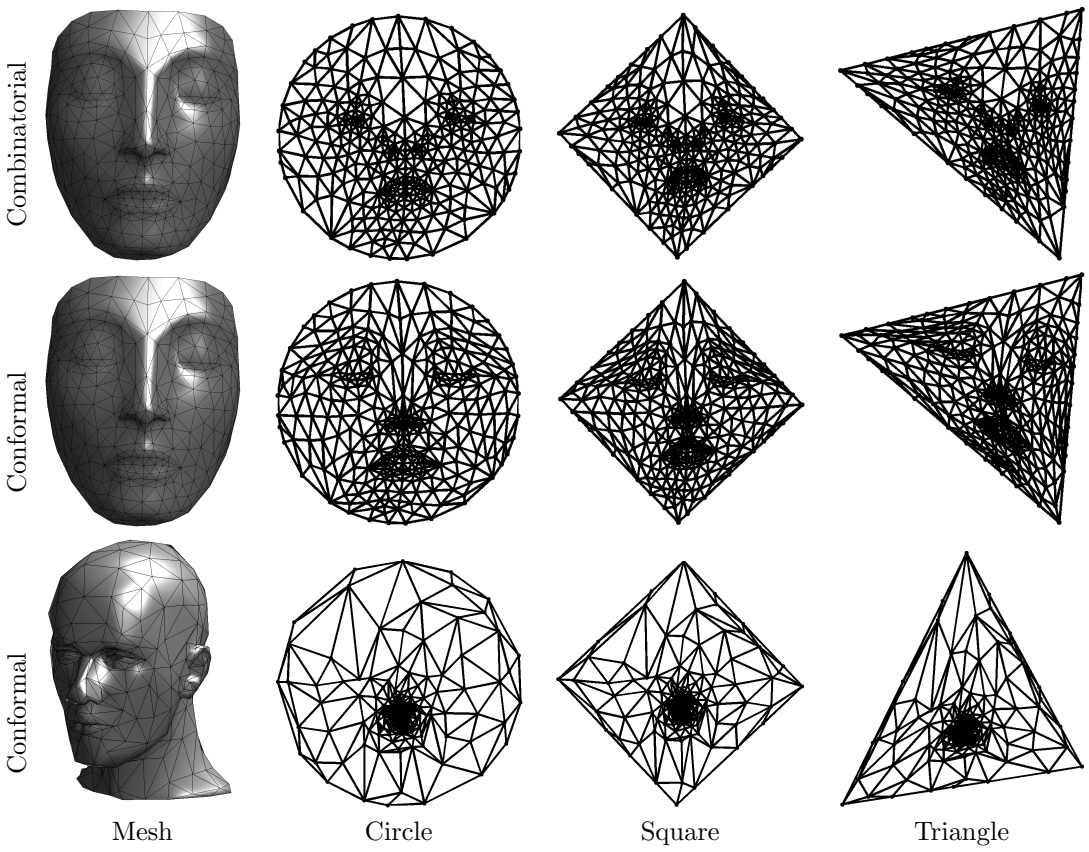


Figure 3.10: Examples of mesh parameterizations.

This optimization problem also has a simple global solution using eigenvectors of the Laplacian.

Theorem 22 (Mesh flattening solution)**.** *The mesh flattening solution is given by*

$$Span(\psi_1, \psi 2) = Span(u_1, u_2) \quad where \quad \tilde{L} = U \Lambda U^{\mathrm{T}}.$$

In order to compute this flattening, one thus needs to extract 2 eigenvectors from a sparse matrix. Note however that, in contrast to linear parameterization schemes, this flattening is not ensured to be bijective. Figure 3.11 shows that for meshes with large distortion, this flattening indeed leads to wrong parameterizations.

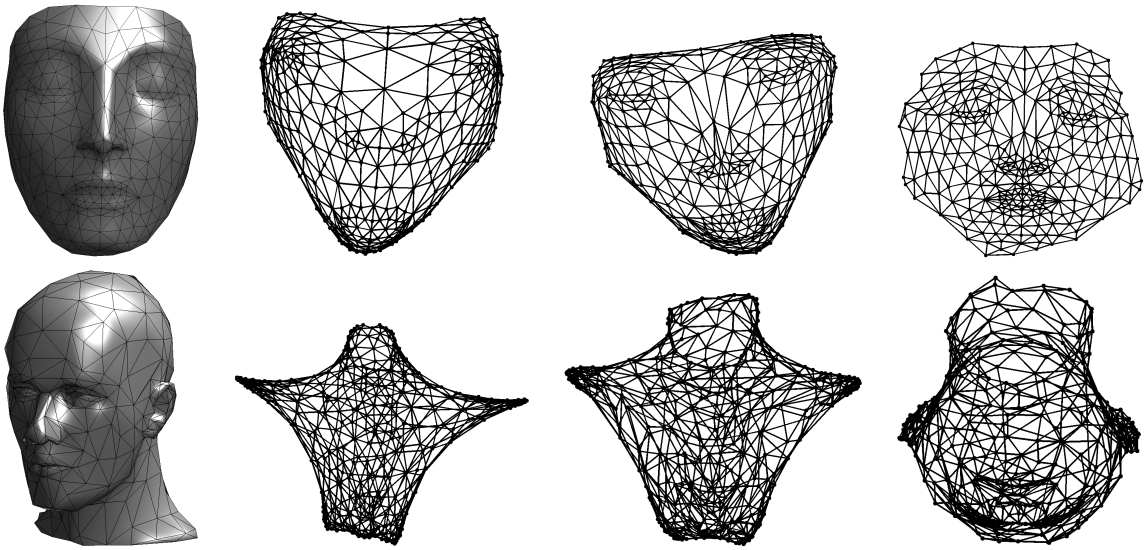


Figure 3.11: Examples of mesh flattening.

Chapter 4

Wavelets

The reference for this chapter is [29].

4.1 Multi-resolution Approximation Spaces

A multiresolution approximation of $L^2(\mathbb{R})$ is a set of nested closed subspaces $(V_j)_j$

$$L^2(\mathbb{R}) \supset \ldots \supset V^{j-1} \supset V_j \supset V_{j+1} \supset \ldots \supset \{0\} \tag{4.1}$$

which must be related one from each other by dyadic scaling and must also be stable by dyadic translation

$$f \in V_j \iff f(\cdot/2) \in V_{j+1} \quad \text{and} \quad f \in V_j \iff \forall\, n \in \mathbb{Z},\ f(\cdot + t2^j) \in V_j$$

So large j corresponds to coarse approximation spaces, and 2^j is often call the "scale".

The limit on the left of (4.1) means that $\cup_j V_j$ is dense in $L^2(\mathbb{R})$, or equivalently that $P_{V_j}(f) \to f$ as $j \to -\infty$ where P_V is the orthogonal projector on V

$$P_V(f) = \operatorname*{argmin}_{f' \in V} \|f - f'\|.$$

The limit on the right of (4.1) means that $\cap_j V_j = \{0\}$, or equivalently that $P_{V_j}(f) \to 0$ as $j \to +\infty$.

Scaling functions. We also require that there exists a scaling function $\varphi \in L^2(\mathbb{R})$ so that

$$\{\varphi(\cdot - n)\ \}_n \text{ is an Hilertian orthonormal basis of } V_0.$$

By the dilation property, this implies that

$$\{\varphi_{j,n}\ \}_n \text{ is an Hilertian orthonormal basis of } V_j \quad \text{where} \quad \varphi_{j,n} \overset{\text{def.}}{=} \frac{1}{2^{j/2}} \varphi\left(\frac{\cdot - 2^j n}{2^j}\right).$$

Note that one then has

$$P_{V_j}(f) = \sum_n \langle f,\, \varphi_{j,n} \rangle \varphi_{j,n}.$$

Figure 4.1 illustrates the translation and scaling effect.

A typical example is approximation by piecewise constant signals

$$V_j \overset{\text{def.}}{=} \left\{ f \in L^2(\mathbb{R})\ ;\ \forall n, f \text{ is constant on } [2^j n, 2^j(n+1)[\right\}, \tag{4.2}$$

in which case one can use $\varphi = 1_{[0,1[}$ and $\varphi_{j,n} = 2^{-j/2} 1_{[2^j n, 2^j(n+1)[}$.

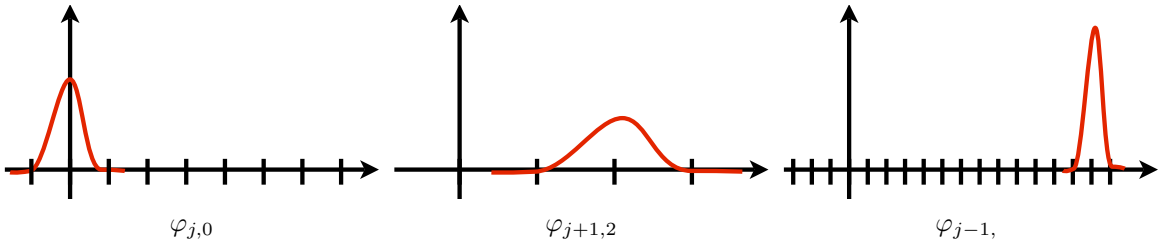


Figure 4.1: Translation and scaling to generate approximation spaces.

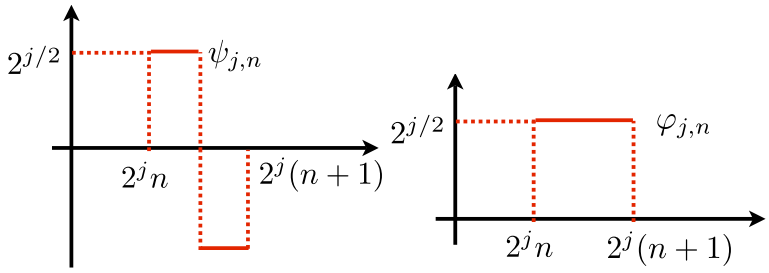


Figure 4.2: Haar scaling (left) and wavelet (right) functions.

Spectral orthogonalization. In many case of practical interest, the space V_j is describe by a translation-invariant basis which is not-orthogonal, $V_j = \mathrm{Span}(\theta(\cdot - n))_{n\in\mathbb{Z}}$. The following proposition shows how to orthogonalize it using the Fourier transform.

Proposition 9. *For $\theta \in L^2(\mathbb{R})$ (assumed regular and with fast enough decay), $\{\theta(\cdot - n)\}_{n\in\mathbb{Z}}$ is orthonormal if and only if*

$$\forall\, \omega, \quad A(\omega) \overset{\text{def.}}{=} \sum_k |\hat{\theta}(\omega - 2\pi k)|^2 = 1.$$

If there exists $0 < a \leqslant b < +\infty$ such that $a \leqslant A(\omega) \leqslant b$, then φ defined by

$$\hat{\varphi}(\omega) = \frac{\hat{\theta}(\omega)}{\sqrt{A(\omega)}}$$

is such that $\{\varphi(\cdot - n)\}_{n\in\mathbb{Z}}$ is an Hilbertian basis of $\mathrm{Span}\{\theta(\cdot - n)\}_{n\in\mathbb{Z}}$.

Proof. One has that $\{\theta(\cdot - n)\}_{n\in\mathbb{Z}}$ is orthonormal if and only if

$$\langle \theta, \theta(\cdot - n)\rangle = (\theta \star \bar{\theta})(n) = \delta_{0,n} \overset{\text{def.}}{=} \begin{cases} 0 & \text{if} \quad n = 0, \\ 1 & \text{otherwise.} \end{cases}$$

where $\bar{\theta} = \theta(?\cdot)$ and δ_0 is the discrete Dirac vector. The Poisson summation formula (1.8) then reads

$$\sum_n \mathcal{F}(\theta \star \bar{\theta})(\omega - 2\pi n) = \sum_n \delta_{0,n} e^{-\mathrm{i}\omega} = 1.$$

We conclude with the Fourier-convolution formula (2.6) shows that $\mathcal{F}(\theta \star \bar{\theta})(\omega) = \hat{\theta}(\omega)\hat{\theta}(\omega)^* = |\hat{\theta}(\omega)|^2$ which leads to the desired formula, and it is if and only if. Normalizing by $1/\sqrt{A(\omega)}$, which is a bounded function, shows that $\hat{\varphi}$ satsifies $\sum_k |\hat{\varphi}(\omega - 2\pi k)|^2 = 1$. □

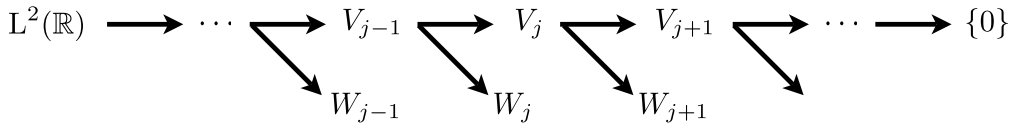
A typical example of application is spline (e.g. cubic ones) interpolations, which are generated by the box-spline function θ which is a piecewise polynomial with a compact support.

4.2 Multi-resolution Details Spaces

The details spaces are defined as orthogonal complement of $V_j \subset V_{j-1}$, which is legit because these are closed subspaces

$$\forall\, j, \quad W_j \text{ is such that } V_{j-1} = V_j \oplus^{\perp} W_j.$$

This leads to the following sequence of embedded spaces



Once again, we suppose that W_0 has an Hilbertian ortho-basis of the form $\{\psi(\cdot - n)\}_n$, so that

$$\{\psi_{j,n}\}_n \text{ is an Hilertian orthonormal basis of } V_j \quad \text{where} \quad \psi_{j,n} \stackrel{\text{def.}}{=} \frac{1}{2^{j/2}}\psi\left(\frac{\cdot - 2^j n}{2^j}\right).$$

Due to the orthogonal complementarity property, one has

$$L^2(\mathbb{R}) = \bigoplus_{j=-\infty}^{j=+\infty} W_j = V_{j_0} \bigoplus_{j \leqslant j_0}^{j} V_j.$$

This means that for all $f \in L^2(\mathbb{R})$, one has the following convergence in $L^2(\mathbb{R})$,

$$f = \lim_{(j_-,j_+)\to(-\infty,+\infty)} \sum_{j=j_-}^{j_+} P_{V_j} f = \lim_{j_+\to+\infty} P_{j_0} f + \sum_{j=j_0}^{j_+} P_{V_j} f.$$

This decomposition shows that

$$\left\{\psi_{j,n} \;;\; (j,n) \in \mathbb{Z}^2\right\}$$

is an Hilbertian orthogonal basis of $L^2(\mathbb{R})$, which is called a wavelet basis. One also have a "truncated" ortho-basis

$$\{\psi_{j,n} \;;\; j \leqslant j_0, n \in \mathbb{Z}\} \cup \{\varphi_{j_0,n} \;;\; n \in \mathbb{Z}\}.$$

A (forward) Wavelet transform corresponds to the computation of all the inner products of some function f with the elements of these basis.

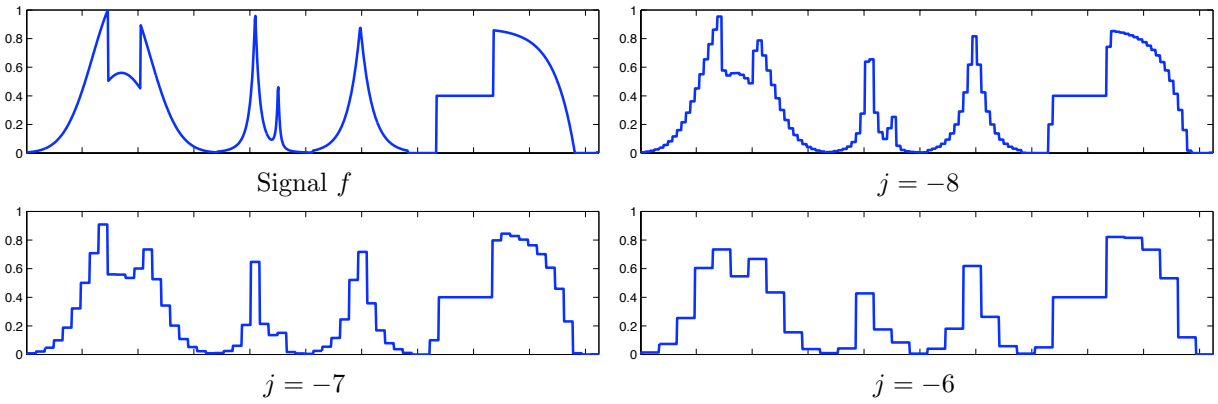


Figure 4.3: 1-D Haar multiresolution projection $P_{V_j} f$ of a function f.

Haar wavelets. For the Haar multiresolution (4.2), one has

$$W_j = \left\{ f \;;\; \forall n \in \mathbb{Z},\; f \text{ constant on } [2^{j+1}n, 2^{j+1}(n+1)) \quad \text{and} \quad \int_{n2^j}^{(n+1)2^j} f = 0 \right\}. \tag{4.3}$$

A possible choice for a mother wavelet function is

$$\psi(t) = \frac{1}{\sqrt{2}} \begin{cases} 1 & \text{for} \quad 0 \leqslant t < 1/2, \\ -1 & \text{for} \quad 1/2 \leqslant t < 1, \\ 0 & \text{otherwise,} \end{cases}$$

as shown on Figure 4.2, right.

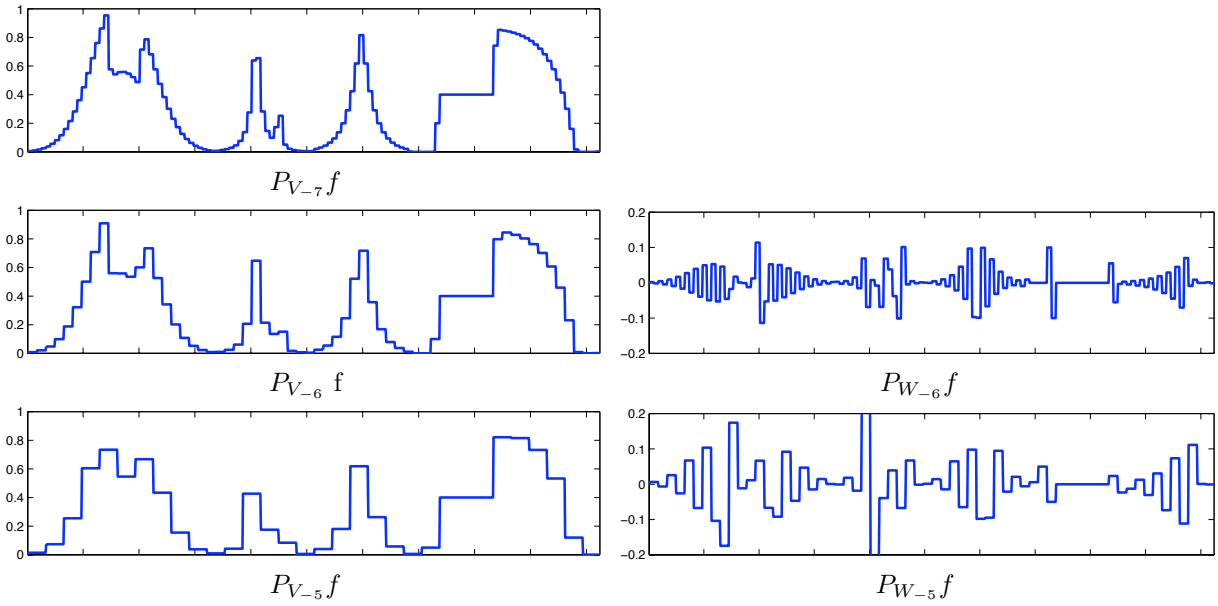


Figure 4.4: Projection on Haar approximation spaces (left) and detail spaces (right).

Figure 4.4 shows examples of projections on details spaces, and how they can be derived from projection on approximation spaces.

4.3 On Bounded Domains

On a periodic bounded domain $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ (note that we use here 1-periodicity, in contrast to the convention we used for Fourier series of 2π-periodicity), one obtains an orthogonal wavelet basis of $L^2(\mathbb{T})$ by periodizing the original wavelets, and also restricting the translation points $2^j n$ to be in $[0,1]$, i.e. $0 \leqslant n < 2^{-j}$. Similarly to (1.6), the periodization of a function $f \in L^1(\mathbb{R})$ is the function

$$f^P = \sum_{n \in \mathbb{Z}} f(\cdot - n) \in L^1(\mathbb{T}).$$

The wavelet basis is thus define as

$$\left\{ \psi^P_{j,n} \;;\; j \leqslant j_0, 0 \leqslant n < 2^{-j} \right\} \cup \left\{ \varphi^P_{j_0,n} \;;\; 0 \leqslant n < 2^{-j_0} \right\}.$$

and one verifies that it defines an Hilbertian ortho-basis of $L^2(\mathbb{T})$. It is possible to define wavelet basis using Neumann (mirror) boundary conditions, but this is more involved.

4.4 Fast Wavelet Transform

4.4.1 Discretization

We now work over $\mathbb{R}/\mathbb{Z}$. The modelling hypothesis is that one has acess to a discrete signal $a_J \in \mathbb{R}^N$ with $N = 2^{-J}$ at some fixed scale 2^J, and that this signal exactly matches the inner-product with the scaling functions, i.e.

$$\forall\, n, \in \{0, \ldots, N-1\}, \quad a_{J,n} = \langle f, \varphi^P_{J,n} \rangle \tag{4.4}$$

for some function of interest f we are sampling. This is equivalent to saying that the discretization process have exactly access to $P_{V_J} f$. This hypothesis is questionnable, and similar to the Shannon bandlimit assumption. In practice, the scaling functions $\varphi_{j,n}$ are often quite close to the point-spread function of the acquisition device, so it is acceptable. One can however improves this by correcting the acquired values by the device to be closer to asumption (4.4).

The discrete wavelet transform then computes, from this input a_J, all the coefficients

$$\forall\, j \in \{J+1, J+2, \ldots, 0\}, \quad \forall\, n \in [\![0, 2^{-j}-1]\!], \quad a_{j,n} \overset{\text{def.}}{=} \langle f, \varphi^P_{j,n} \rangle, \quad \text{and} \quad d_{j,n} \overset{\text{def.}}{=} \langle f, \psi^P_{j,n} \rangle$$

in this order (increasing values of j). During the algorithm, the previously computed vector a_j can be discarded, and only the d_j are kept.

The forward discrete wavelet transform on a bounded domain is thus the orthogonal finite dimensional map

$$a_J \in \mathbb{R}^N \longmapsto \left\{ d_{j,n} \;;\; 0 \leqslant j < J, 0 \leqslant n < 2^{-j} \right\} \cup \{a_0 \in \mathbb{R}\}.$$

The inverse transform, which is thus the adjoint due to orthogonality, is the inverse map.

Figure 4.5 shows examples of wavelet coefficients. For each scale 2^j, there are 2^{-j} coefficients.

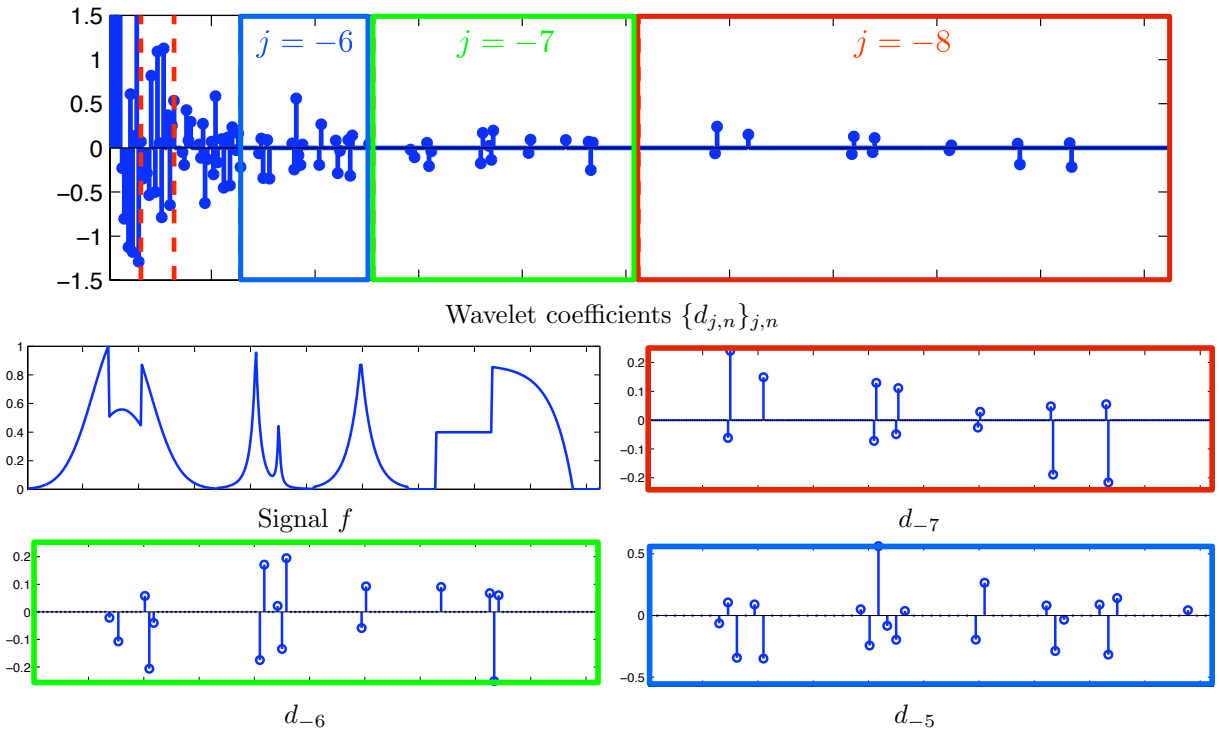


Figure 4.5: Wavelet coefficients. Top row: all the coefficients. Bottoms rows: zoom on the different scales

4.4.2 Forward Fast Wavelet Transform (FWT)

The algorithm proceeds by computing a series of simple operators

$$\forall\, j = J+1, \ldots, 0, \quad (a_j, d_j) = \mathcal{W}_j(a_{j-1}) \quad \text{where} \quad \mathcal{W}_j : \mathbb{R}^{2^{-j+1}} \to \mathbb{R}^{2^{-j}} \times \mathbb{R}^{2^{-j}} \tag{4.5}$$

The number of such steps is thus $|J| = \log_2(N)$. Each $\mathcal{W}_j$ is orthogonal since it corresponds to linear maps between coefficients in orthogonal bases.

In order to describe the algorihm that computes $\mathcal{W}_j$, we introduce the filters "filter" coefficients $f, g \in \mathbb{R}^{\mathbb{Z}}$

$$h_n \overset{\text{def.}}{=} \frac{1}{\sqrt{2}}\langle \varphi(\cdot/2),\, \varphi(\cdot - n)\rangle \quad \text{and} \quad g_n \overset{\text{def.}}{=} \frac{1}{\sqrt{2}}\langle \psi(\cdot/2),\, \varphi(\cdot - n)\rangle. \tag{4.6}$$

We denote as $\downarrow_2 \colon \mathbb{R}^K \to \mathbb{R}^{K/2}$ the subsampling operator by a factor of 2, i.e.

$$u \downarrow_2 \overset{\text{def.}}{=} (u_0, u_2, u_4, \ldots, u_{K-2}, u_K).$$

In the following, we assume that these filters are decaying fast enough.

Proposition 10. *One has*

$$\mathcal{W}_j(a_j) = ((\bar{h} \star a_{j-1}) \downarrow_2, (\bar{g} \star a_{j-1}) \downarrow_2), \tag{4.7}$$

where $\star$ denotes periodic convolutions on $\mathbb{R}^{-j+1}$.

Proof. We note that $\varphi(\cdot/2)$ and $\psi(\cdot/2)$ are in $\mathcal{W}_j$, so one has the decompositions

$$\frac{1}{\sqrt{2}}\varphi(t/2) = \sum_n h_n \varphi(t-n) \quad \text{and} \quad \frac{1}{\sqrt{2}}\psi(t/2) = \sum_n g_n \varphi(t-n) \tag{4.8}$$

Doing the change of variable $t \mapsto \frac{t-2^j p}{2^{j-1}}$ in (4.8), one obtains

$$\frac{1}{\sqrt{2}}\varphi\left(\frac{t-2^j p}{2^j}\right) = \sum_n h_n \varphi\left(\frac{t}{2^{j-1}} - (n+2p)\right)$$

(similarly for ψ) and then doing the change $n \mapsto n - 2p$, one obtains

$$\varphi_{j,p} = \sum_{n\in\mathbb{Z}} h_{n-2p}\varphi_{j-1,n} \quad \text{and} \quad \psi_{j,p} = \sum_{n\in\mathbb{Z}} g_{n-2p}\psi_{j-1,n}.$$

When working with periodized function $(\varphi^P_{j,n}, \psi^P_{j,p})$, this formula is still valid, but the summation over $n \in \mathbb{Z}$ should be done modulo 2^{-j+1}. Taking inner product of both size with respect to f (which is legit if h, g are decaying fast enough), one obtains the fundamental recursion fromula

$$a_{j,p} = \sum_{n\in\mathbb{Z}} h_{n-2p} a_{j-1,n} = (\bar{h} \star a_{j-1})_{2p} \quad \text{and} \quad d_{j,p} = \sum_{n\in\mathbb{Z}} g_{n-2p} a_{j-1,n} = (\bar{g} \star a_{j-1})_{2p} \tag{4.9}$$

where $\bar{u}_n \overset{\text{def.}}{=} u_{-n}$. One can show that this formula is still valid when working over a bounded interval $\mathbb{T} = \mathbb{R}/\mathbb{Z}$, but then $\star$ denotes the periodic convolution over $\mathbb{Z}/2^{-j+1}\mathbb{Z}$. □

Figure 4.6 shows two steps of application of these refinement relationships.

The FWT thus operates as follow:

– **Input:** signal $f \in \mathbb{C}^N$.

– **Initialization:** $a_J = f$.

– **For** $j = J, \ldots, j_0 - 1$.

$$a_{j+1} = (a_j \star \tilde{h}) \downarrow 2 \quad \text{and} \quad d_{j+1} = (a_j \star \tilde{g}) \downarrow 2$$

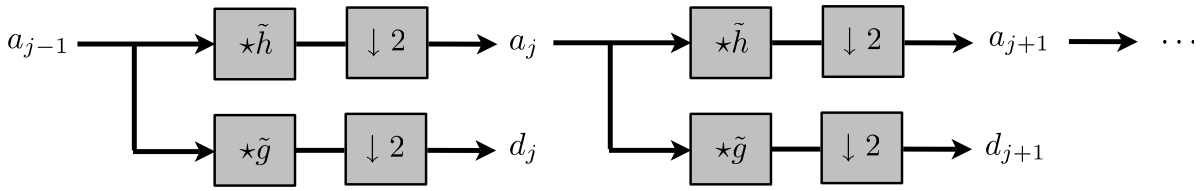


Figure 4.6: Forward filter bank decomposition.

– Output: the coefficients $\{d_j\}_{j_0 \leqslant j < J} \cup \{a_{j_0}\}$.

If $|h|, |g| \leqslant C$ so that both filter are compactly supported, then computing each $\mathcal{W}_j$ is $(2C)2^{-j}$ operation, so that the complexity of the whole wavelet transform is

$$\sum_{j=J}^{1} (2C)2^{-j} = (2C)2^{-J} = 2CN.$$

This shows that the fast wavelet transform is a linear time algorithm. Figure 4.7 shows the process of extracting iteratively the wavelet coefficients. Figure 4.8 shows an example of computation, where at each iteration, the coefficients of a_j and d_j are added to the left of the output vector.

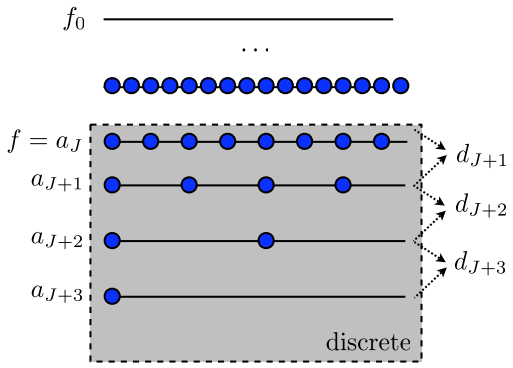


Figure 4.7: Pyramid computation of the coefficients.

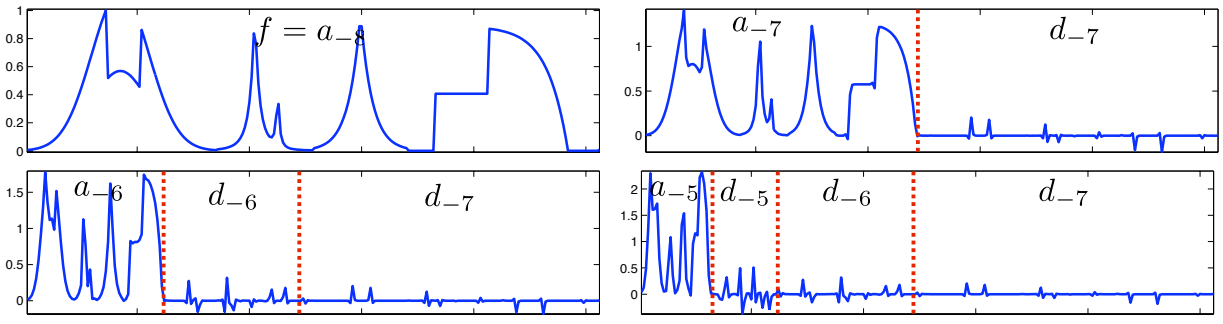


Figure 4.8: Wavelet decomposition algorithm.

Fast Haar transform. For the Haar wavelets, one has

$$\varphi_{j,n} = \frac{1}{\sqrt{2}}(\varphi_{j-1,2n} + \varphi_{j-1,2n+1}),$$

$$\psi_{j,n} = \frac{1}{\sqrt{2}}(\varphi_{j-1,2n} - \varphi_{j-1,2n+1}).$$

This corresponds to the filters

$$h = [\ldots,\ 0,\ h[0] = \frac{1}{\sqrt{2}},\ \frac{1}{\sqrt{2}},\ 0, \ldots],$$

$$g = [\ldots,\ 0,\ h[0] = \frac{1}{\sqrt{2}},\ -\frac{1}{\sqrt{2}},\ 0, \ldots].$$

The Haar wavelet transform algorithm thus processes by iterating averaging and differences:
– Input: signal $f \in \mathbb{C}^N$.
– Initialization: $a_J = f$.
– For $j = J, \ldots, j_0 - 1$.

$$a_{j+1,n} = \frac{1}{\sqrt{2}}(a_{j-1,2n} + a_{j-1,2n+1}) \quad \text{and} \quad d_{j+1,n} = \frac{1}{\sqrt{2}}(a_{j-1,2n} - a_{j-1,2n+1}).$$

– Output: the coefficients $\{d_j\}_{j_0 \leqslant j < J} \cup \{a_{j_0}\}$.

4.4.3 Inverse Fast Transform (iFWT)

The inverse algorithm proceeds by inverting each step (4.10)

$$\forall\, j = 0, \ldots, J+1, \quad a_{j-1} = \mathcal{W}_j^{-1}(a_j, d_j), \tag{4.10}$$

where $\mathcal{W}_j^*$ is the adjoint for the canonical inner product on $\mathbb{R}^{2^{-j+1}}$, i.e. when viewed as a matrix, the transpose.

We denote $\uparrow_2 \colon \mathbb{R}^{K/2} \to \mathbb{R}^K$ the up-sampling operator

$$a \uparrow_2 = (a_0, 0, a_1, 0, \ldots, 0, a_{K/2}, 0) \in \mathbb{R}^K.$$

Proposition 11. *One has*

$$\mathcal{W}_j^{-1}(a_j, d_j) = (a_j \uparrow_2) \star h + (d_j \uparrow_2) \star g.$$

Proof. Since $\mathcal{W}_j$ is orthogonal, $\mathcal{W}_j^{-1} = \mathcal{W}_j^*$. We write the whole transform as

$$\mathcal{W}_j = S_2 \circ C_{\bar{h},\bar{g}} \circ \mathcal{D} \quad \text{where} \quad \begin{cases} \mathcal{D}(a) = (a, a), \\ C_{\bar{h},\bar{g}}(a,b) = (\bar{h} \star a, \bar{g} \star a), \\ S_2(a,b) = (a \downarrow_2, b \downarrow_2). \end{cases}$$

One has the following adjoint operator

$$\mathcal{D}^*(a,b) = a + b, \quad C^*_{\bar{h},\bar{g}}(a,b) = (h \star a, g \star b), \quad \text{and} \quad S_2(a,b) = (a \uparrow_2, b \uparrow_2).$$

Indeed, let us check this for the convolution, assuming involved sequences are in ℓ_1 (they are actually finite sums when considering periodic signals),

$$\langle f \star \bar{h},\, g \rangle = \sum_n (f \star \bar{h})_n g_n = \sum_n \sum_k f_k h_{k-n} g_n = \sum_k f_k \sum_n h_{k-n} g_n = \langle f,\, h \star g \rangle,$$

for the copying

$$\langle \mathcal{D}(a),\, (u,v) \rangle = \langle a,\, u \rangle + \langle b,\, u \rangle = \langle a,\, u + v \rangle = \langle a,\, \mathcal{D}^*(u,v) \rangle,$$

and for the down-sampling

$$\langle f \downarrow_2, g \rangle = \sum_n f_{2n} g_n = \sum_n (f_{2n} g_n + f_{2n+1} 0) = \langle f, g \uparrow_2 \rangle.$$

This is shown using matrix notations in Figure 4.9. Putting everything together gives the desired formula. □

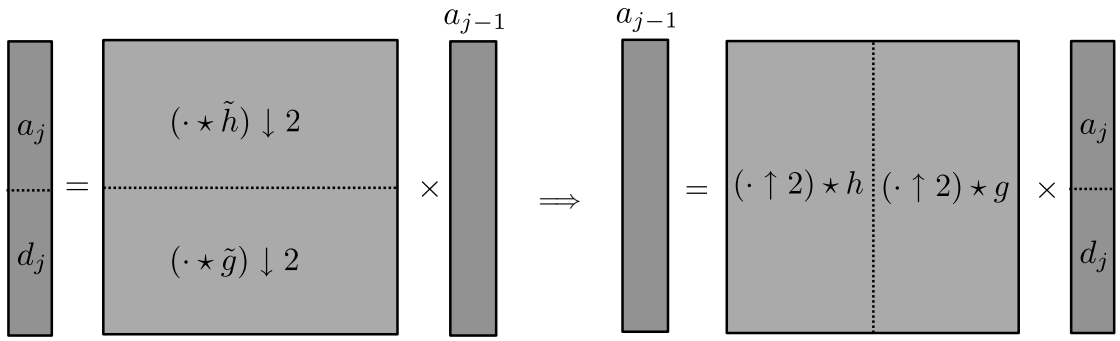


Figure 4.9: Wavelet inversion in matrix format.

The inverse Fast wavelet transform iteratively applies this elementary step

– **Input:** $\{d_j\}_{j_0 \leqslant j < J} \cup \{a_{j_0}\}$.

– **For** $j = j_0, \ldots, J+1$.

$$a_{j-1} = (a_j \uparrow 2) \star h + (d_j \uparrow 2) \star g.$$

– **Output:** $f = a_J$.

This process is shown using a block diagram in Figure 4.10, which is the inverse of the block diagram 4.6.

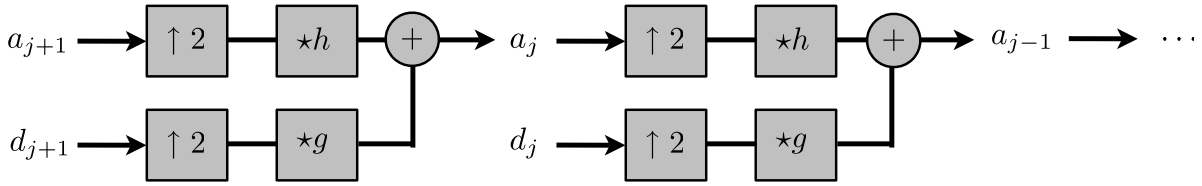


Figure 4.10: Backward filterbank recomposition algorithm.

4.5 2-D Wavelets

4.5.1 Anisotropic Wavelets

2-D anisotropic wavelets are defined using tensor product of Wavelet basis functions (2.12). The basis over $\mathbb{T}^2$ is thus of the form

$$\left\{ \psi_{(j_1,j_2),(n_1,n_2)} \; ; \; (j_1,j_2) \in \mathbb{Z}^2, 0 \leqslant n_1 < 2^{-j_1}, 0 \leqslant n_2 < 2^{-j_2} \right\}$$

$$\text{where} \quad \psi_{(j_1,j_2),(n_1,n_2)}(x_1,x_2) = \psi_{j_1,n_1}(x_1)\psi_{j_2,n_2}(x_2). \tag{4.11}$$

The computation of the fast anisotropic Wavelet transform in 2-D is similar to the 2-D FFT detailed in Section 2.5.2. Viewing the input image $a_J \in \mathbb{R}^{2^{-J} \times 2^{-J}}$ as a matrix, one first apply the 1-D FWT to each row, and then to each column, resulting in a linear time $O(N)$ algorithm, where $N = 2^{-2J}$.

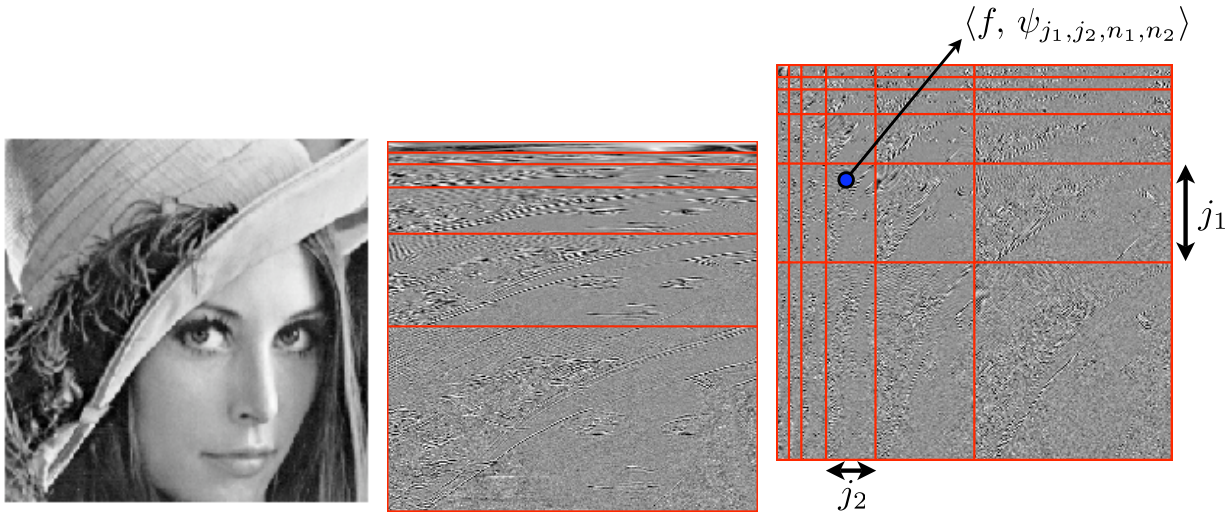


Image f Row transform Column transform.

Figure 4.11: Steps of the anisotropic wavelet transform.

4.5.2 Isotropic Wavelets

A major issue with these anisotropic wavelet (4.11) is that a function $\psi_{(j_1,j_2),(n_1,n_2)}$ is scaled independently in each direction, leads to functions concentrated along an axis-oriented rectangle of size $2^{-j_1} \times 2^{j_2}$. This is not a very good features (since natural images usually do not exhibit such an anisotropy) and typically leads to visually unpleasant artifacts when used for processing (denoising or compression).

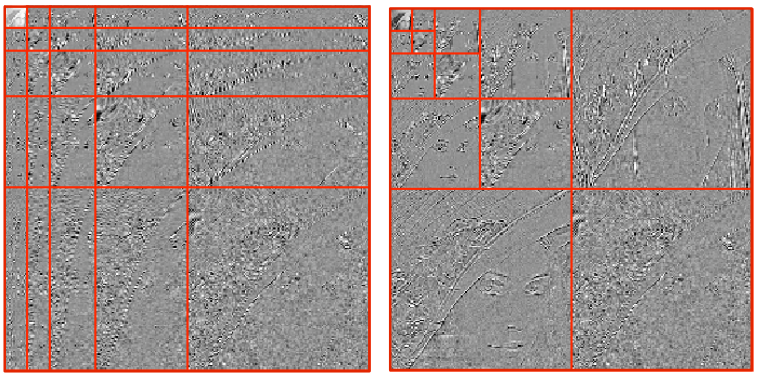


Figure 4.12: Anisotropic (left) versus isotropic (right) wavelet coefficients.

One rather use isotropic wavelet obtained by considering 2-D multi-resolution, obtained by tensor products of the 1-D approximation spaces

$$L^2(\mathbb{R}^2) \supset \ldots \supset V^{j-1} \otimes V^{j-1} \supset V_j \otimes V_j \supset V_{j+1} \otimes V_{j+1} \supset \ldots \supset \{0\}.$$

In the following, we denote

$$V_j^O \stackrel{\text{def.}}{=} V_j \otimes V_j$$

this isotropic 2-D multiresolution space.

Recall that the tensor product of two space $(V_1, V_2) \in L^2(\mathbb{R})^2$ is

$$V_1 \otimes V_2 = \text{Closure}\left(\text{Span}\left\{f_1(x_1)f_2(x_2) \in L^2(\mathbb{R}^2)\ ;\ f_1 \in V_1, f_2 \in V_2\right\}\right).$$

If $(\varphi_k^s)_k$ are Hilbertian bases for V_s, then one can show that $(\varphi_k^1(x_1)\varphi_k^2(x_2))_k$ is an Hilbertian basis for $V_1 \otimes V_2$.

One easily verify that one has the distributivity

$$(V_j \oplus^\perp W_J) \otimes (V_j \oplus^\perp W_J) = V_j^O \oplus^\perp W_j^V \oplus^\perp W_j^V \oplus^\perp W_j^V \quad \text{where} \quad \left\{ \begin{array}{l} W_j^V \stackrel{\text{def.}}{=} (V_j \otimes W_j), \\ W_j^H \stackrel{\text{def.}}{=} (W_j \otimes V_j), \\ W_j^D \stackrel{\text{def.}}{=} (W_j \otimes W_j). \end{array} \right.$$

Here the letters $\{V, H, D\}$ stands for *Vertical, Horizontal, Diagonal* detail spaces. This leads to the following diagram of embedded spaces

$$\mathrm{L}^2(\mathbb{R}^2) \longrightarrow \cdots \searrow V_{j-1} \otimes V_{j-1} \searrow V_j \otimes V_j \searrow V_{j+1} \otimes V_{j+1} \searrow \cdots \longrightarrow \{0\}$$
$$\qquad\qquad\qquad\quad \downarrow \qquad\qquad\quad \downarrow \qquad\qquad\quad \downarrow$$
$$\qquad\qquad\qquad\quad W_{j-1}^{(2)} \qquad\quad W_j^{(2)} \qquad\quad W_{j+1}^{(2)}$$

For $j \in \mathbb{Z}$, each of the three wavelet spaces is spanned with a wavelet, where basis elements are indexed by $n = (n_1, n_2) \in \mathbb{Z}$ (or in $\{0, \ldots, 2^{-j} - 1\}^2$ on the interval $\mathbb{T}$),

$$\forall\, \omega \in \{V, H, D\}, \quad W_j^\omega = \text{Span}\{\psi_{j,n_1,n_2}^\omega\}_{n_1,n_2}$$

where

$$\forall\, \omega \in \{V, H, D\}, \quad \psi_{j,n_1,n_2}^\omega(x) = \frac{1}{2^j}\psi^\omega\left(\frac{x_1 - 2^j n_1}{2^j}, \frac{x_2 - 2^j n_2}{2^j}\right)$$

and where the three mother wavelets are

$$\psi^H(x) = \psi(x_1)\varphi(x_2), \quad \psi^V(x) = \varphi(x_1)\psi(x_2), \quad \text{and} \quad \psi^D(x) = \psi(x_1)\psi(x_2).$$

Figure 4.13 displays an examples of these wavelets.

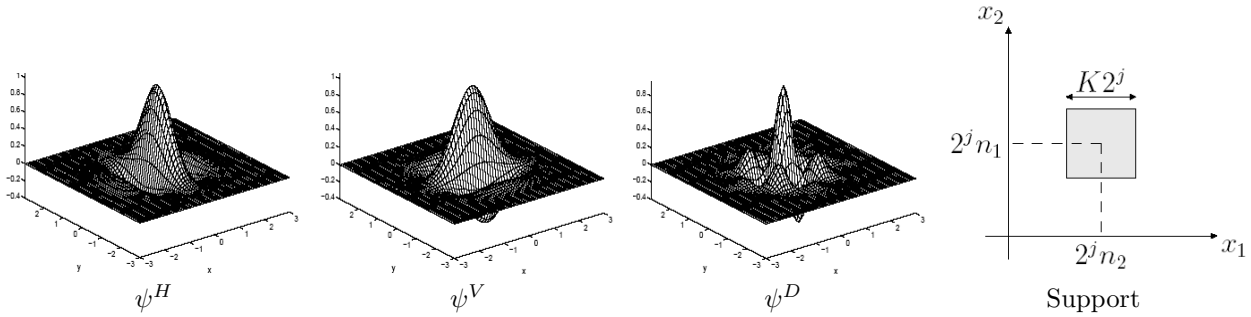


Figure 4.13: 2-D wavelets and their approximative support (right).

Haar 2-D multiresolution. For the Haar multiresolution, one obtains 2-D piecewise-constant Haar approximation. A function of $V_j \otimes V_j$ is constant on squares of size $2^j \times 2^j$. Figure 4.14 shows an example of projection of an image onto these 2-D Haar approximation spaces.

Figure 4.14: 2-D Haar approximation $P_{V_j^O} f$ for increasing j.

Discrete 2-D wavelet coefficients. Similarly to (4.4), we suppose that the sampling mechanism gives us access to inner product of the analog (continuous) signal f with the scaling function at scale $N = 2^{-J}$

$$\forall\, n, \in \{0, \ldots, N-1\}^2, \quad a_{J,n} = \langle f, \varphi^P_{J,n} \rangle$$

Discrete wavelet coefficients are defined as

$$\forall\, \omega \in \{V, H, D\},\ \forall\, J < j \leqslant 0,\ \forall\, 0 \leqslant n_1, n_2 < 2^{-j}, \quad d^\omega_{j,n} = \langle f, \psi^\omega_{j,n} \rangle.$$

(we use here periodized wavelets). Approximation coefficients are defined as

$$a_{j,n} = \langle f_0, \varphi^O_{j,n} \rangle.$$

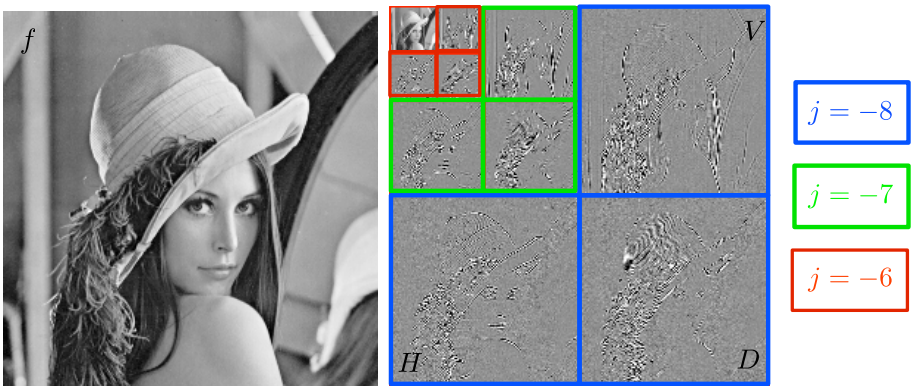


Figure 4.15: 2-D wavelet coefficients.

Figure 4.15 shows examples of wavelet coefficients, that are packed in an image of N pixels. Figure 4.16 shows other examples of wavelet decompositions.

Forward 2-D wavelet transform basic step. A basic step of the computation of the 2-D wavelet transform computes detail coefficients and a low pass residual from the fine scale coefficients

$$a_{j-1} \longmapsto (a_j, d^H_j, d^V_j, d^D_j).$$

Similarly to the 1-D setting, this mapping is orthogonal, and is computed using the 1-D filtering and subsampling formula (4.7).

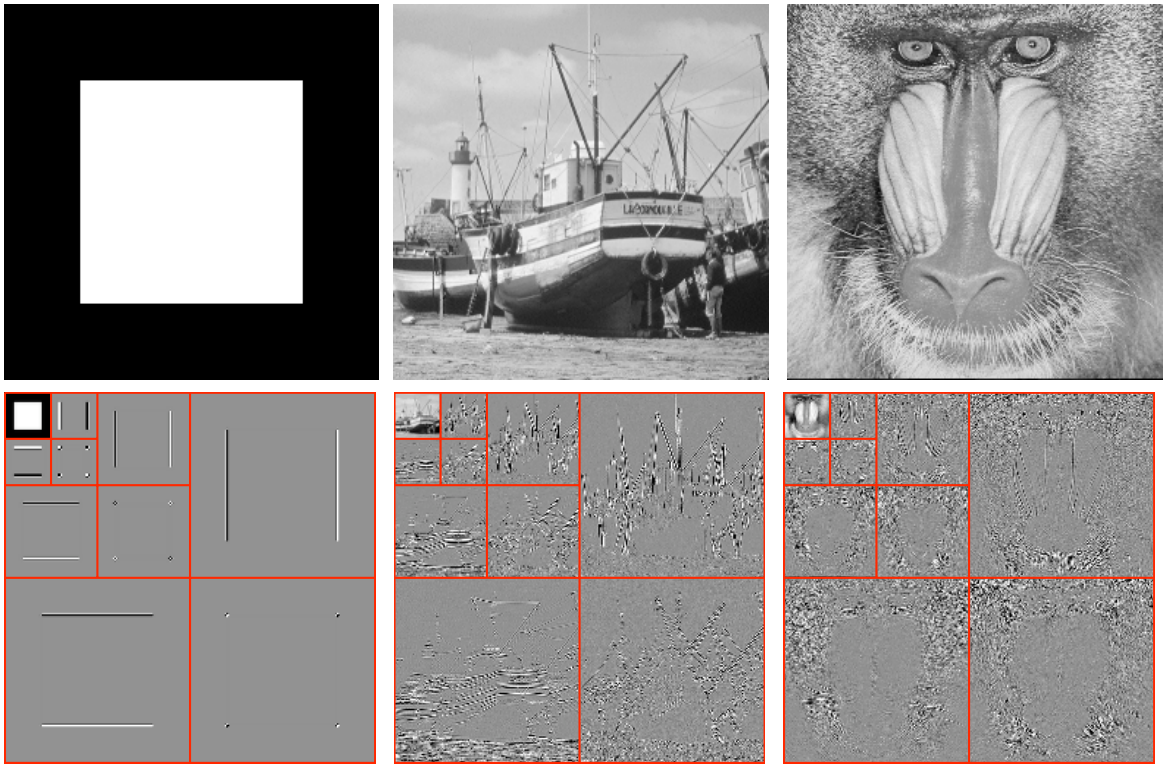


Figure 4.16: Examples of images (top row) and the corresponding wavelet coefficients (bottom row) .

One first applies 1-D horizontal filtering and sub-sampling

$$\tilde{a}_j = (a_{j-1} \star^H \tilde{h}) \downarrow^H 2$$
$$\tilde{d}_j = (a_{j-1} \star^H \tilde{h}) \downarrow^H 2,$$

where $\star^H$ is the horizontal convolution, that applies the 1-D convolution to each column of a matrix

$$a \star^H b_{n_1,n_2} = \sum_{m_1=0}^{P-1} a_{n_1-m_1,n_2} b_{m_1}$$

where $a \in \mathbb{C}^{P \times P}$ and $b \in \mathbb{C}^P$ are matrix and vectors. The notation $\downarrow^H 2$ accounts for sub-sampling in the horizontal direction

$$(a \downarrow^H 2)_{n_1,n_2} = a_{2n_1,n_2}.$$

One then applies 1-D vertical filtering and sub-sampling to $\tilde{a}_j$ and $\tilde{d}_j$ to obtain

$$\begin{aligned} a_j &= (\tilde{a}_j \star^V \tilde{h}) \downarrow^V 2, & d_j^H &= (\tilde{d}_j \star^V \tilde{h}) \downarrow^V 2, \\ d_j^V &= (\tilde{a}_j \star^V \tilde{g}) \downarrow^V 2, & d_j^D &= (\tilde{d}_j \star^V \tilde{g}) \downarrow^V 2, \end{aligned}$$

where the vertical operators are defined similarly to horizontal operators but operating on rows.

These two forward steps are shown in block diagram in Figure 4.17. These steps can be applied in place, so that the coefficients are stored in an image of N pixels, as shown in Figure 4.18. This gives the traditional display of wavelet coefficients used in Figure 4.16.

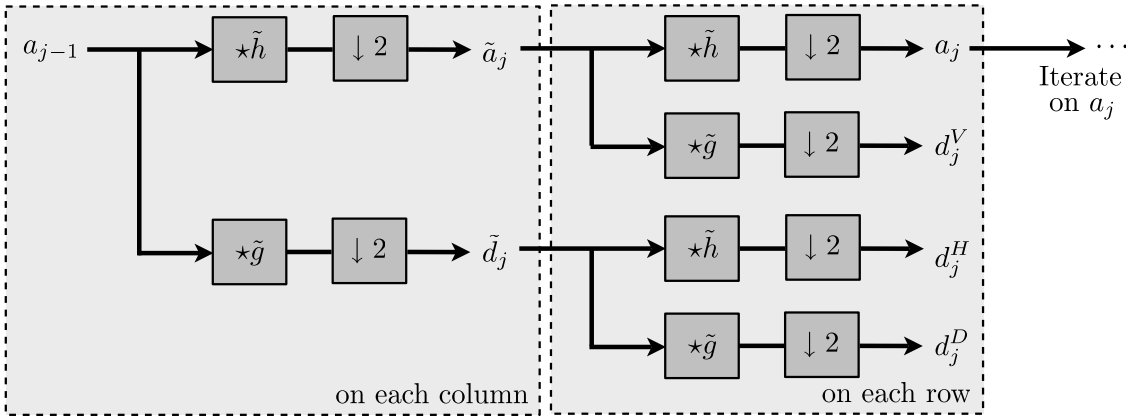


Figure 4.17: Forward 2-D filterbank step.

Coefficients a_j Transform on rows Transform on columns

Figure 4.18: One step of the 2-D wavelet transform algorithm.

Fast 2-D wavelet transform. The 2-D FWT algorithm iterates these steps through the scales:
– **Input:** signal $f \in \mathbb{C}^N$.
– **Initialization:** $a_J = f$.
– **For** $j = J, \ldots, j_0 - 1$.

$$\begin{aligned} \tilde{a}_j &= (a_{j-1} \star^H \tilde{h}) \downarrow^H 2, & d_j^V &= (\tilde{a}_j \star^V \tilde{g}) \downarrow^V 2, \\ \tilde{d}_j &= (a_{j-1} \star^H \tilde{h}) \downarrow^H 2, & d_j^H &= (\tilde{d}_j \star^V \tilde{h}) \downarrow^V 2, \\ a_j &= (\tilde{a}_j \star^V \tilde{h}) \downarrow^V 2, & d_j^D &= (\tilde{d}_j \star^V \tilde{g}) \downarrow^V 2. \end{aligned}$$

– **Output:** the coefficients $\{d_j^\omega\}_{j_0 \leqslant j < J, \omega} \cup \{a_{j_0}\}$.

Fast 2-D inverse wavelet transform. The inverse transform undo the horizontal and vertical filtering steps. The first step computes

$$\begin{aligned} \tilde{a}_j &= (a_j \star^V h) \uparrow^V 2 + (d_j^V \star^V g) \uparrow^V 2, \\ \tilde{d}_j &= (d_j^H \star^V h) \uparrow^V 2 + (d_j^D \star^V g) \uparrow^V 2, \end{aligned}$$

where the vertical up-sampling is

$$(a \uparrow^V 2)_{n_1,n_2} = \begin{cases} a_{k,n_2} & \text{if} \quad n_1 = 2k, \\ 0 & \text{if} \quad n = 2k+1. \end{cases}$$

The second inverse step computes

$$a_{j-1} = (\tilde{a}_j \star^H h) \uparrow^H 2 + (\tilde{d}_j \star^H g) \uparrow^H 2.$$

Figure 4.19 shows in block diagram this inverse filter banks, that is the inverse of the diagram 4.17.

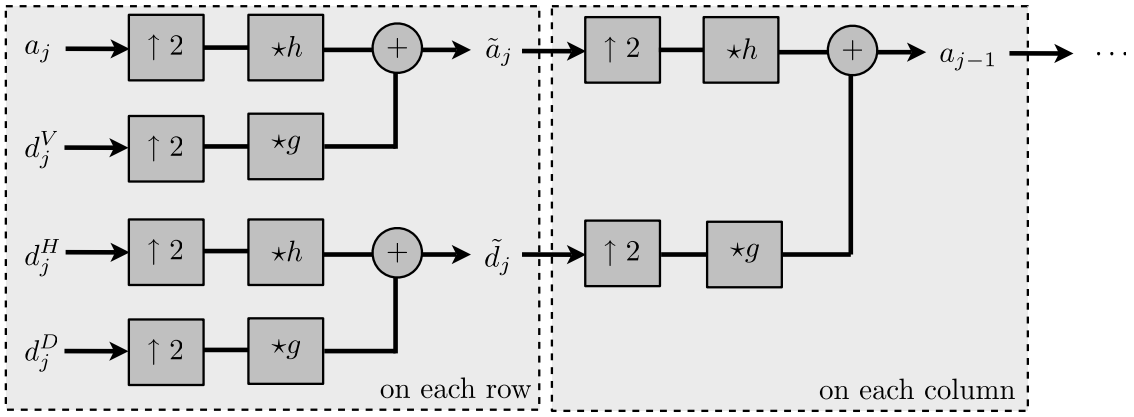


Figure 4.19: Backward 2-D filterbank step.

The inverse Fast wavelet transform iteratively applies these elementary steps

– Input: $\{d_j^\omega\}_{j_0 \leqslant j < J, \omega} \cup \{a_{j_0}\}$.

– For $j = j_0, \ldots, J+1$.

$$\begin{aligned}
\tilde{a}_j &= (a_j \star^V h) \uparrow^V 2 + (d_j^V \star^V g) \uparrow^V 2, \\
\tilde{d}_j &= (d_j^H \star^V h) \uparrow^V 2 + (d_j^D \star^V g) \uparrow^V 2, \\
a_{j-1} &= (\tilde{a}_j \star^H h) \uparrow^V 2 + (\tilde{d}_j \star^H g) \uparrow^V 2.
\end{aligned}$$

– Output: $f = a_J$.

4.6 Wavelet Design

To be able to compute the wavelet coefficients using the FWT algorithm, it remains to know how to compute the scaling and wavelet functions. The FWT only makes use of the filters h and g, so instead of explicitly knowing the functions φ and ψ, one can only know these filters. Indeed, most of the known wavelets do not have explicit formula, and are implicitly defined through the cascade of the FWT algorithm.

This section shows what are the constraints h and g should satisfy, and gives practical examples. Furthermore, it shows that the knowledge of h determines g under the constraint of having quadrature filters, which is the most usual choice for wavelet analysis.

4.6.1 Low-pass Filter Constraints

We introduce the following three conditions on a filter h

$$\hat{h}(0) = \sqrt{2} \tag{C_1}$$

$$|\hat{h}(\omega)|^2 + |\hat{h}(\omega + \pi)|^2 = 2, \tag{C_2}$$

$$\inf_{\omega \in [-\pi/2, \pi/2]} |\hat{h}(\omega)| > 0. \tag{C^*}$$

Here we are using the Fourier series associated to a filter $h \in \mathbb{R}^{\mathbb{Z}}$

$$\forall \omega \in \mathbb{R}/2\pi\mathbb{Z}, \quad \hat{h}(\omega) \stackrel{\text{def.}}{=} \sum_{n\in\mathbb{Z}} h_n e^{-\mathrm{i}n\omega}.$$

If $h \in \ell^1(\mathbb{Z})$, this defines a continuous periodic fonction $\hat{h} \in \mathcal{C}^0(\mathbb{R}/2\pi\mathbb{Z})$, and this definition can be extended to $h \in \ell^2(\mathbb{Z})$ and defines $\hat{h} \in L^2(\mathbb{R}/2\pi\mathbb{Z})$.

Theorem 23. *If φ defines multi-resolution approximation spaces, then (C_1) and (C_2) holds for h defined in (4.6). Conversely, if (C_1), (C_2) and (C^*) holds, then there exists a φ defining multi-resolution approximation spaces so that associated filter is h as defined in (4.6).*

Proof. We only prove the first statement of the theorem. The converse statement is much more difficult to prove.

We now prove condition (C_1). The refinement equation reads like a discrete-continuous convolution (or equivalently a convolution with a distribution)

$$\frac{1}{\sqrt{2}}\varphi\left(\frac{t}{2}\right) = \sum_{n\in\mathbb{Z}} h_n \varphi(t-n). \tag{4.12}$$

Denoting $h \star \varphi$ such a convolution, assuming $h \in \ell_1(\mathbb{Z})$ and $\varphi \in L^1(\mathbb{R})$, one check that one can apply Fubini and that $h \star \varphi \in L^1(\mathbb{R})$ and then

$$\begin{aligned}\mathcal{F}(\sum_{n\in\mathbb{Z}} h_n\varphi(t-n))(\omega) &= \int_{\mathbb{R}} \sum_{n\in\mathbb{Z}} h_n\varphi(t-n)e^{-\mathrm{i}\omega t}\mathrm{d}t = \sum_{n\in\mathbb{Z}} h_n \int_{\mathbb{R}} \varphi(t-n)e^{-\mathrm{i}\omega t}\mathrm{d}t \\ &= \sum_{n\in\mathbb{Z}} h_n e^{-\mathrm{i}n\omega} \int_{\mathbb{R}} \varphi(x)e^{-\mathrm{i}\omega x}\mathrm{d}x = \hat{\varphi}(\omega)\hat{h}(\omega)\end{aligned}$$

where we made the change of variable $x = t-n$. Note that here, $h(\omega)$ is the 2π-periodic Fourier transform (i.e. Fourier series) of infinite filters defined in (4.6.1), whereas $\hat{\varphi}(\omega)$ is the Fourier transform of function. This is thus a product of a 2π-periodic function $\hat{h}$ and a non-periodic function $\hat{\varphi}$. We recall that $\mathcal{F}(f(\cdot/s)) = s\hat{f}(s\cdot)$. Over the Fourier domain, equation (4.12) thus reads

$$\hat{\varphi}(2\omega) = \frac{1}{\sqrt{2}}\hat{h}(\omega)\hat{\varphi}(\omega). \tag{4.13}$$

One can show that $\hat{\varphi}(0) \neq 0$ (actually, $|\hat{\varphi}(0)| = 1$), so that this relation implies the first condition (C_1).

We now prove condition (C_2). The orthogonality of $\varphi(\cdot-n)\}_n$ is rewritten using a continuous convolution as (see also Proposition 9)

$$\forall\, n \in \mathbb{Z}, \quad \varphi \star \bar{\varphi}(n) = \delta_0$$

where $\bar{\varphi}(x) = \varphi(-x)$, and thus over the Fourier domain, using (4.13) which shows $\hat{\varphi}\omega) = \frac{1}{\sqrt{2}}\hat{h}(\omega/2)\hat{\varphi}(\omega/2)$

$$1 = \sum_k |\hat{\varphi}(\omega+2k\pi)|^2 = \frac{1}{2}\sum_k |\hat{h}(\omega/2+k\pi)|^2|\hat{\varphi}(\omega/2+k\pi)|^2.$$

Since $\hat{h}$ is 2π-periodic, one can split even and odd k and obtain

$$2 = |h(\omega/2)|^2 \sum_k |\hat{\varphi}(\omega/2+2k\pi)|^2 + |h(\omega/2+\pi)|^2 \sum_k |\hat{\varphi}(\omega/2+2k\pi+\pi)|^2$$

This leads to condition (C_2). Re-using the fact that $\sum_k |\hat{\varphi}(\omega+2k\pi)|^2 = 1$ for $\omega' = \omega/2$ in place of ω, one thus has

$$|h(\omega')|^2 + |h(\omega'+\pi)|^2 = 2.$$

We do not prove the converse statement, which requires to "create" a function φ from the filter h. The intuition is that iterating (4.13) leads informally to

$$\varphi(\omega) = \prod_{k<0} \frac{\hat{h}(\omega/2^k)}{\sqrt{2}}. \tag{4.14}$$

Condition (C^*) can be shown to imply that this infinite product converge, and define a (non-periodic) function in $L^2(\mathbb{R})$. □

Note that for the converse statement of this theorem to holds, condition (C^*) imposes a control on the behavior of $\hat{h}$ near 0.

4.6.2 High-pass Filter Constraints

We now introduce the following two conditions on a pair of filter (g, h)

$$|\hat{g}(\omega)|^2 + |\hat{g}(\omega+\pi)|^2 = 2 \tag{C_3}$$

$$\hat{g}(\omega)\hat{h}(\omega)^* + \hat{g}(\omega+\pi)\hat{h}(\omega+\pi)^* = 0. \tag{C_4}$$

Theorem 24. *If (φ, ψ) defines a multi-resolution analysis, then (C_3) and (C_4) holds for (h, g) defined in (4.6). Conversely, if (C_1) to (C_4) hold, then there exists a (φ, ψ) defining multi-resolution analysis so that associated filters are (h, g) as defined in (4.6). Furthermore,*

$$\hat{\psi}(\omega) = \frac{1}{\sqrt{2}}\hat{g}(\omega/2)\hat{\varphi}(\omega/2). \tag{4.15}$$

Proof. We prove condition (C_3). The refinement equation for the wavelet reads

$$\frac{1}{\sqrt{2}}\psi\left(\frac{t}{2}\right) = \sum_{n\in\mathbb{Z}} g_n \varphi(t-n)$$

and thus over the Fourier domain

$$\hat{\psi}(2\omega) = \frac{1}{\sqrt{2}}\hat{g}(\omega)\hat{\varphi}(\omega). \tag{4.16}$$

The orthogonality of $\{\psi(\cdot - n)\}_n$ is re-written

$$\forall\, n \in \mathbb{Z}, \quad \psi \star \bar{\psi}(n) = \delta_0$$

and thus over the Fourier domain (using Poisson formula, see also Proposition 9)

$$\sum_k |\hat{\psi}(\omega + 2k\pi)|^2 = 1.$$

Using the Fourier domain refinement equation (4.16), similarely to the proof of Theorem 23 for (C_1), this is equivalent to condition (C_3). Figure 4.20 shows the Fourier transform of two filters that satisfy this complementary condition.

We now prove condition (C_4). The orthogonality between $\{\psi(\cdot - n)\}_n$ and $\{\varphi(\cdot - n)\}_n$ is written as

$$\forall\, n \in \mathbb{Z}, \quad \psi \star \bar{\varphi}(n) = 0$$

and hence over the Fourier domain (using Poisson formula, similarly to Proposition 9)

$$\sum_k \hat{\psi}(\omega + 2k\pi)\hat{\varphi}^*(\omega + 2k\pi) = 0.$$

Using the Fourier domain refinement equations (4.13) and (4.16), this is equivalent to condition (C_4). □

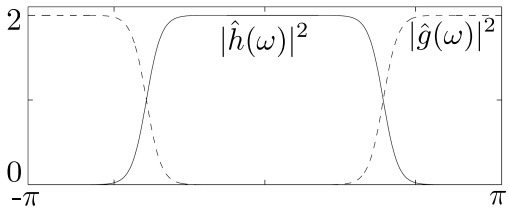


Figure 4.20: Complementarity between a low pass and a high pass wavelet filters h and g that satisfy condition (C_3).

Quadrature mirror filters. Quadrature mirror filters (QMF) defines g as a function of h so that the conditions of Theorem 24 are automatically satisfy. This choice is the natural choice to build wavelet filters, and is implicitly assumed in most constructions (other choices leading to the same wavelet function anyway, since it safisfies (4.15)).

Proposition 12. *For a filter $h \in \ell^2(\mathbb{Z})$ satisfying (C_1), defining $g \in \ell^2(\mathbb{Z})$ as*

$$\forall\, n \in \mathbb{Z}, \quad g_n = (-1)^{1-n} h_{1-n} \tag{4.17}$$

satisfies conditions (C_3) and (C_4).

Proof. One indeed has that

$$\hat{g}(\omega) = e^{-i\omega}\hat{h}(\omega+\pi)^*, \tag{4.18}$$

so that

$$|\hat{g}(\omega)|^2 + |\hat{g}(\omega+\pi)|^2 = |e^{-i\omega}\hat{h}(\omega+\pi)^*|^2 + |e^{-i(\omega+\pi)}\hat{h}(\omega+2\pi)^*|^2 = |\hat{h}(\omega+\pi)|^2 + |\hat{h}(\omega+2\pi)|^2 = 2.$$

where we used the fact that $\hat{h}$ is 2π-periodic, and also

$$\begin{aligned}\hat{g}(\omega)\hat{h}(\omega)^* + \hat{g}(\omega+\pi)\hat{h}(\omega+\pi)^* &= e^{-i\omega}\hat{h}(\omega+\pi)^*\hat{h}(\omega)^* + e^{-i(\omega+\pi)}\hat{h}(\omega+2\pi)^*\hat{h}(\omega+\pi)^* \\ &= (e^{-i\omega} + e^{-i(\omega+\pi)})\hat{h}(\omega+\pi)^*\hat{h}(\omega)^* = 0.\end{aligned}$$

□

4.6.3 Wavelet Design Constraints

According to the previous sections, the construction of a multi-resolution analysis (i.e. of functions (φ, ψ) is obtained by designing a filter h satisfying conditions (C_1) and (C_2). The function φ is obtained by an infinite cascade of filtering, or equivalently in the Fourier domain by (4.14), there is in general (put aside special case such as the Haar multiresolution) no closed form expression for φ. Once φ is defined, ψ is automatically defined by the relation (4.15) (and g can be defined as (4.18)).

There exists only one Fourier transform, but there is a large choice of different mother wavelet functions ψ. They are characterized by

– Size of the support.
– Number of oscillations (the so called number p of vanishing moments).
– Symmetry (only possible for non-orthogonal bases).
– Smoothness (number of derivatives).

We now detail how these constraints are integrated together with conditions (C_1)- (C_4).

Vanishing moments. A wavelet ψ has p vanishing moments if

$$\forall\, k \leqslant p-1, \quad \int_{\mathbb{R}} \psi(x)\, x^k \mathrm{d}x = 0. \tag{4.19}$$

This ensures that $\langle f, \psi_{j,n}\rangle$ is small if f is C^α, $\alpha < p$ on $\mathrm{Supp}(\psi_{j,n})$.

This condition can be equivalently expressed over Fourier as followed.

Proposition 13. *Assuming enough regularity of ψ, and using the QMF construction (4.18), it has p vanishing moments if and only if*

$$\forall\, k \leqslant p-1, \quad \frac{\mathrm{d}^k \hat{h}}{\mathrm{d}\omega^k}(\pi) = \frac{\mathrm{d}^k \hat{g}}{\mathrm{d}\omega^k}(0) = 0. \tag{4.20}$$

Proof. Since ψ is regular, one has that $\hat{\psi}^{(k)} = \mathcal{F}((-\mathrm{i}\cdot)^k \psi(\cdot))$, so that

$$(-\mathrm{i})^k \int_{\mathbb{R}} x^k \psi(x) \mathrm{d}x = \hat{\psi}^{(k)}(0).$$

Relation (4.15) and (4.18) implies

$$\hat{\psi}(2\omega) = \hat{h}(\omega+\pi)^* \rho(\omega) \quad \text{where} \quad \rho(\omega) \overset{\text{def.}}{=} \frac{1}{\sqrt{2}} e^{-i\omega} \hat{\varphi}(\omega).$$

and differentiating this relation shows

$$2\hat{\psi}^{(1)}(2\omega) = \hat{h}(\omega+\pi)^* \rho^{(1)}(\omega) + \hat{h}^{(1)}(\omega+\pi)^* \rho(\omega)$$

which shows that, since $\rho(0) = \hat{\varphi}(0) \neq 0$, $\psi^{(1)}(0) = 0 \Leftrightarrow \hat{h}^{(1)}(\pi)^* = 0$. Recursing this argument and iterating the derrivative, one obtains that $\psi^{(k)}(0) = 0 \Leftrightarrow \hat{h}^{(k)}(\pi)^* = 0$ (assuming this hold for previous derivatives). □

Note that conditions (C_1) and (C_1) implies that $\hat{h}(\pi) = 0$, so that an admissible wavelet necessarily has 1 vanishing moment, i.e. $\int \psi = 0$. Condition (4.20) shows that having more vanishing moment is equivalent to having a Fourier transform $\hat{h}$ which is "flatter" arround $\omega = \pi$.

Support. Figure 4.21 shows the wavelet coefficients of a piecewise smooth signal. Coefficients of large magnitude are clustered near the singularities, because the wavelet ψ has enough vanishing moments.

To avoid that many wavelets create large coefficients near singularities, one should choose ψ with a small support. One can show that the size of the support of ψ is proportional to the size of the support of h. This requirement is however contradictory with the vanishing moment property (4.19). Indeed, one can prove that for an orthogonal wavelet basis with p vanishing moments

$$|\mathrm{Supp}(\psi)| \geqslant 2p-1,$$

where $\sup(a)$ is the largest closed interval outside of which the function f is zero.

Chapter 6 studies in details the tradeoff of support size and vanishing moment to perform non-linear approximation of piecewise smooth signals.

Smoothness. In compression or denoising applications, an approximate signals is recovered from a partial set I_M of coefficients,

$$f_M = \sum_{(j,n)\in I_M} \langle f, \psi_{j,n}\rangle \psi_{j,n}.$$

This approximation f_M has the same smoothness as ψ.

To avoid visually unpleasant artifacts, one should thus choose a smooth wavelet function ψ. This is only for cosmetic reasons, since increasing smoothness does not leads to a better approximation. However, for most wavelet family, increasing the number of vanishing moments also increases the smoothness of the wavelets. This is for instance the case of the Daubechies family exposed in the next section.

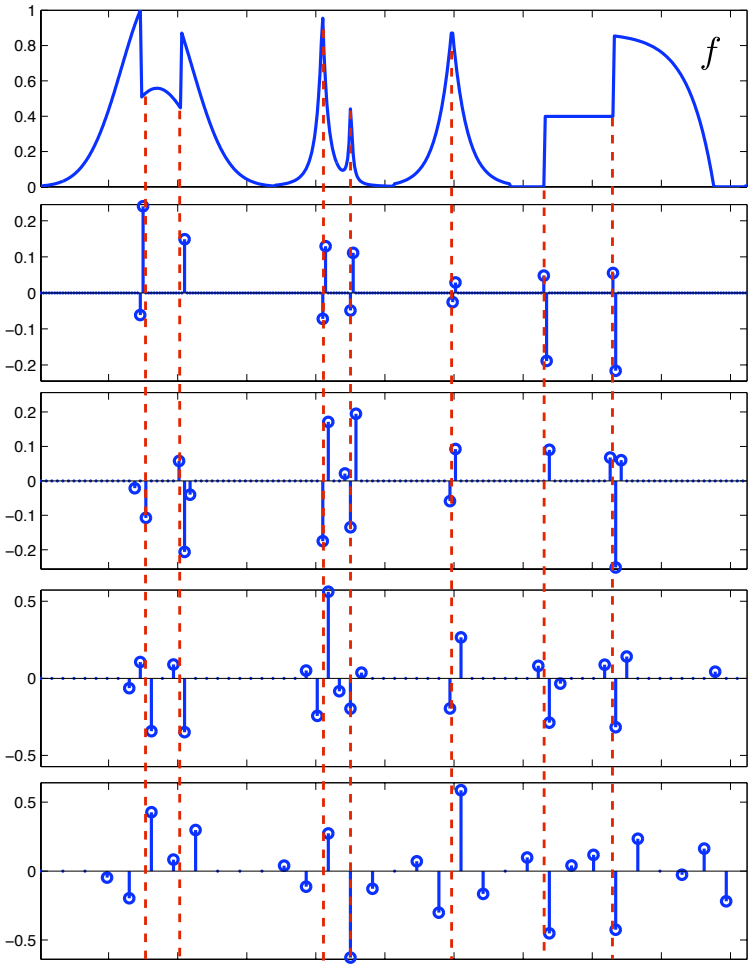


Figure 4.21: Location of large wavelet coefficients.

4.6.4 Daubechies Wavelets

To build a wavelet ψ with a fixed number p of vanishing moments, one designs the filter h, and use the quadrature mirror filter relation (4.18) to compute g. One thus look for h such that

$$|\hat{h}(\omega)|^2 + |\hat{h}(\omega+\pi)|^2 = 2, \quad \hat{h}(0) = \sqrt{2}, \quad \text{and} \quad \forall\, k < p, \ \frac{\mathrm{d}^k \hat{h}}{\mathrm{d}\omega^k}(\pi) = 0.$$

This corresponds to algebraic relationships between the coefficients of h, and it turns out that they can be solved explicitly using the Euclidean division algorithm for polynomials.

This leads to Daubechies wavelets with p vanishing moments, which are orthogonal wavelets with a minimum support length of $2p-1$.

For $p = 1$, it leads to the Haar wavelet, with

$$h = [h_0 = 0.7071; 0.7071].$$

For $p = 2$, one obtains the celebrated Daubechies 4 filter

$$h = [0.4830; h_0 = 0.8365; 0.2241; -0.1294],$$

and for $p = 3$,

$$h = [0; 0.3327; 0.8069; h_0 = 0.4599; -0.1350; -0.0854; 0.0352].$$

Wavelet display. Figure 4.22 shows examples of Daubechies mother wavelet functions with an increasing number of vanishing moments. These displays are obtained by computing in fact a discrete wavelet $\bar{\psi}_{j,n}$ defined in (**??**) for a very large number of samples N. This discrete wavelet is computed by applying the inverse wavelet transform to the coefficients $d_{j',n'} = \delta_{j-j'}\delta_{n-n'}$.

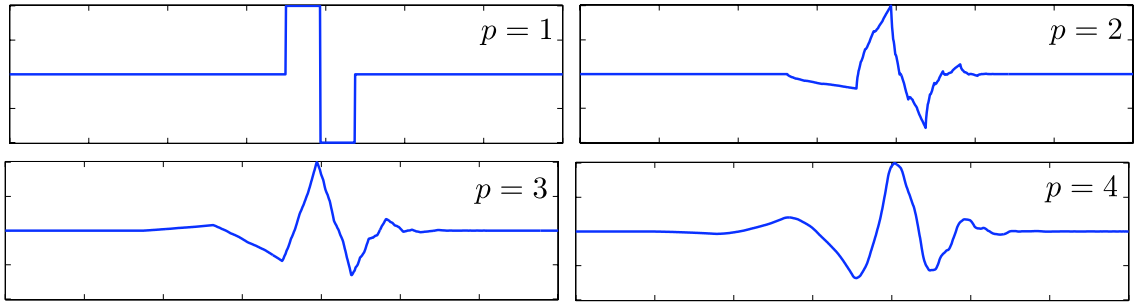


Figure 4.22: Examples of Daubechies mother wavelets ψ with an increasing number p of vanishing moments.

Chapter 5

Multiresolution Mesh Processing

This chapter shows how computations on a mesh can be performed in a multiscale manner, by considering meshes of increasing resolutions. This leads to the notion of subdivision surfaces and wavelet transform, which are two different tools to interpolates and decompose functions on meshes. Both methods rely on a special kind of meshes whose triangulations can be obtained by applying a regular refinement rule.

5.1 Semi-regular Meshes

5.1.1 Nested Multiscale Grids.

In order to perform multiscale mesh processing, one needs to pack the vertices V of a topological mesh $M = (V, E, F)$ in sets of increasing resolution. As explained in section 3.1.2, it is important to remember that this construction is purely combinatorial, in that no geometrical information (such as actual positions of the vertices in $\mathbb{R}^3$) is required to build the set of multi-resolution meshes. In fact these multiscale grids can be used to actually process the geometrical realization $\mathcal{M}$ of the mesh M as three real valued functions (the three coordinates of the points).

We thus consider a set of nested indexes

$$V_0 \subset V_{-1} \subset \ldots \subset V_L = V$$

which are split according to

$$V_j = V_{j+1} \cup H_{j+1}.$$

Next section describes how to actually compute this set of nested grids using a triangular split, but most of the mathematical tools are in fact valid for arbitrary set of indices, as long as they are embedded in one each other through scales.

For mesh processing, an index $\ell \in V_j$ corresponds to a vertex $x_\ell \in \mathcal{V} \subset \mathbb{R}^3$. The signals to be processed are vectors $f \in \mathbb{R}^n$ of size $n = |V_L|$ defined on the grid V_L. We sometimes write $f \in \ell^2(V_L)$ instead of $f \in \mathbb{R}^n$ to emphasis the domain on which f is indexed. This chapter describes transforms for signals $f \in \ell^2(V_L)$ sampled on the finest grid V_L.

5.1.2 Semi-regular Triangulation.

The combinatorial structure of a triangular mesh is defined in section 3.1.2. This chapter considers only a certain class of meshes $M = (V, E, F)$ that can be obtained by a regular split of faces, starting from an initial coarse triangulation. This splitting leads to a set of multiresolution meshes $M_j = (V_j, E_j, F_j)$ for $J \leqslant j \leqslant 0$, where the full mesh is $M_J = M$.

Starting from this coarse triangulation, one defines by subdivision a multiscale triangulation $(V_j, E_j, F_j)_{L \leqslant j \leqslant 0}$ where

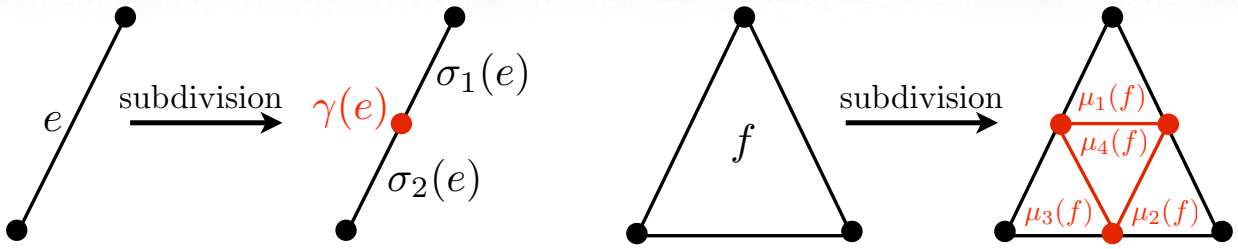


Figure 5.1: Edge-splitting subdivision.

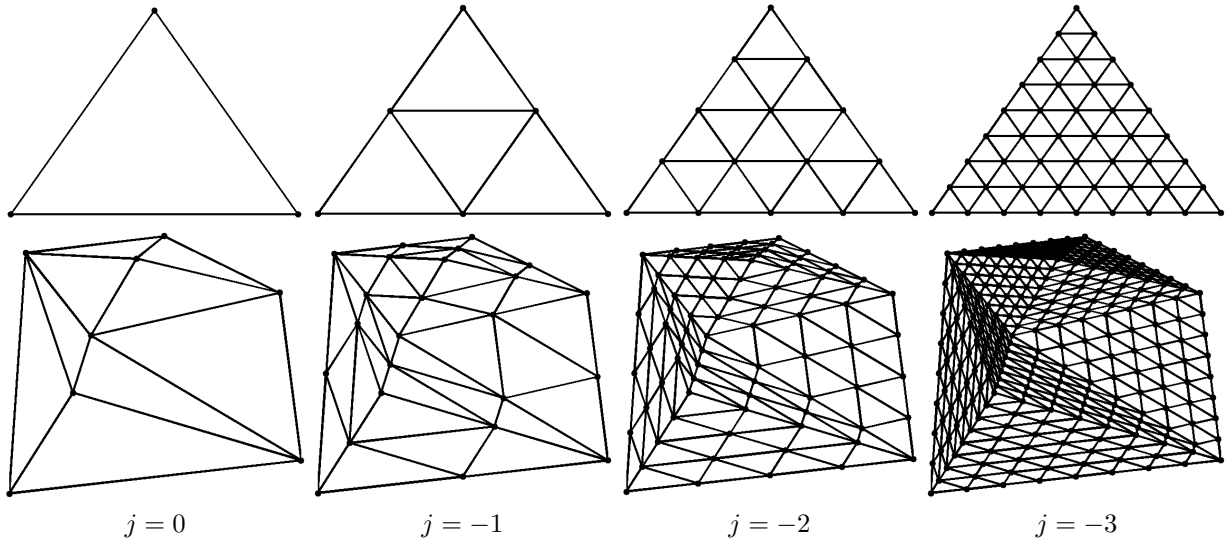


Figure 5.2: Regular subdivision 1:4 of a single triangle. Regular subdivision of a planar triangulation M_0.

– For each edge $e \in E_j$, a central index $\gamma(e) \in V_{j-1}$ is added to the vertices

$$V_{j-1} = V_j \cup \{\gamma(e)\ ;\ e \in E_j\}\,.$$

– Each edge is subdivided into two finer edges

$$\forall\, e = (a,b) \in E_j, \quad \sigma_1(e) = (a, \gamma(e)) \quad \text{and} \quad \sigma_2(e) = (b, \gamma(e)).$$

The subdivided set of edges is then

$$E_{j-1} = \{\sigma_i(e)\ ;\ i = 1, 2 \quad \text{and} \quad e \in E_j\}\,.$$

– Each face $f = (a,b,c) \in F_j$ is subdivided into four faces

$$\left\{ \begin{array}{l} \mu_1(f) = (a, \gamma(a,b), \gamma(a,c)),\ \mu_2(f) = (b, \gamma(b,a), \gamma(b,c)), \\ \mu_3(f) = (c, \gamma(c,a), \gamma(c,b)),\ \mu_4(f) = (\gamma(a,b), \gamma(b,c), \gamma(c,a)). \end{array} \right.$$

The subdivided set of faces is then

$$F_{j-1} = \{\mu_i(f)\ ;\ i = 1, 2, 3, 4 \quad \text{and} \quad f \in F_j\}\,.$$

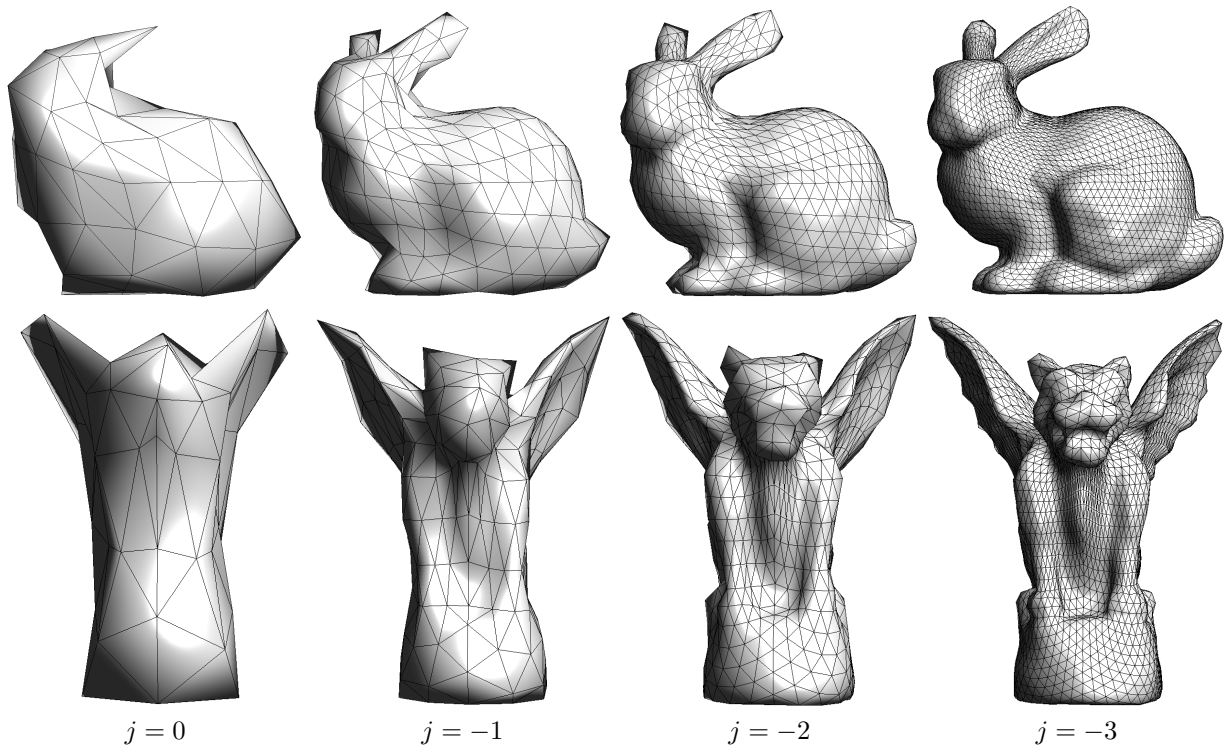


Figure 5.3: Examples of semi-regular meshes $(V_j)_j$ for increasing scale j (from left to right).

Figure 5.1 shows the notations related to the subdivision process. Figure 5.2 shows an example of recursive splitting of a triangle and a coarse triangulation. Figure 5.3 shows examples of semi-regular triangulation using a geometric realization (position of the vertices) to create a 3D surface.

The set of vertices can be classified as

- **Regular vertices** are those who belong neither to the coarse mesh V_0 nor to a boundary of a mesh M_j. These vertices have always 6 neighbors.
- **Extraordinary vertices** are the initial vertices of V_0. They exhibit arbitrary connectivity.
- **Boundary vertices** are those belonging to a mesh boundary. Boundary vertices not in V_0 always have 4 immediate neighbors.

Obviously not every meshes can be obtained from such a subdivision process. In practice, an arbitrary mesh, obtained from CAD design or range scanning usually does not have any multiscale structure. It is thus necessary to remesh it in order to modify the connectivity of the mesh. During this process, the position of the vertices in $\mathbb{R}^3$ is modified in order for the geometrical realization to stay close from the original piecewise linear surface. One can see [2] fur a survey of various semi-regular remeshing methods.

5.1.3 Spherical Geometry Images

Starting from some input surfaces $\mathcal{S} \subset \mathbb{R}^3$, one typically wants to compute a semi-regular meshes $(M_j)_{j \geqslant L}$ that approximate $\mathcal{S}$. In most case, the surface $\mathcal{S}$ is actually given as an arbitrary triangulated mesh and this process corresponds to a semi-regular remeshing. Many algorithm have been devised for surface remeshing and we describe here a method [35] that works for surfaces that have the topology of a sphere. It means that the surface has genus 0, without boundary and without handles.

This methods works by computing several intermediate surface-wise parameterization.

- *Spherical parameterization:* each points of the original triangulation of $\mathcal{S}$ is mapped onto the unit sphere.

This create a bijective parameterization

$$\varphi_S : S^2 \to \mathcal{S}.$$

This is a non-linear process that differ from the planar parameterization introduced in section 3.3.7. We do not give the details of such a process, but it requires minimizing the smoothness of the mapping φ_S^{-1} under the constraint that it maps points of $\mathcal{S}$ to unit length vectors (point on the sphere S^2). The algorithm is explained in details in [35].

– *Spherical-tetraedron flattening:* one flatten each quadrant (1/8) of the sphere in order to have a mapping

$$\varphi_T : \text{Octaedron} \to S^2.$$

One can use for instance a mapping between spherical barycentric coordinate on each quadrant and Euclidean barycentric coordinates on each face of the octahedron.

– *Tetraedron unfolding:* One maps each equilateral face of the octaedron on a rectangular triangle that corresponds to 1/8th of the square $[0,1]^2$

$$\varphi_U : [0,1]^2 \to \text{Octaedron}.$$

– *Regular sampling:* the geometry image is obtained by regularly sampling the square on a uniform grid

$$x_\ell = \varphi_S \circ \varphi_T \circ \varphi_U(\ell/n) \quad \text{for} \quad \ell_i = 0, \ldots, n-1.$$

The mapping $\ell \mapsto x_\ell \in \mathbb{R}^3$ is the geometry image, which can be stored as a 3-channel (color) image.

From such a geometry image x_ℓ, one can easily compute a semi-regular mesh by simply performing a regular 1:4 subdivision of the octaedron. Figure 5.4 shows the steps of the construction of a geometry image, and the resulting semi-regular mesh.

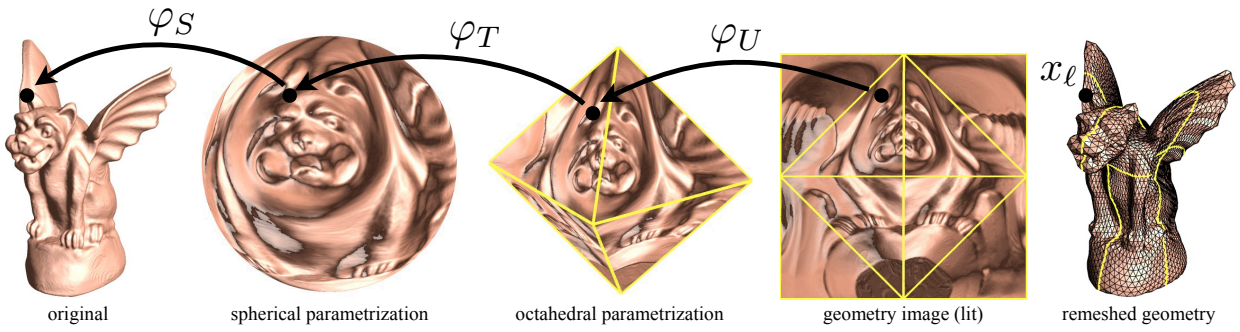


Figure 5.4: Spherical geometry image construction, taken from [35].

5.2 Subdivision Curves

Before getting into the detail of subdivision surfaces, we describe the subdivision process in the simpler setting of 1D signals. This leads to the construction of subdivision of 1D functions and subdivision curves.

In this 1D setting, the grid point indexes are dyadic sub-grids of $\mathbb{Z}$

$$\forall\, j \geqslant L, \quad V_j = \left\{ \ell 2^{j-L} \;;\; 0 \leqslant \ell < s_0 2^{-j} \right\},$$

where $s_0 = |V_0|$ is the size of the initial vector f_0 to be subdivided.

Each subdivision steps computes, from a set $f_j(\ell) \in \ell^2(V_j)$ of coarse values, a refined vector $f_{j-1} \in \ell^2(V_{j-1})$ defined by

$$\begin{cases} \forall\, k \in H_j,\ f_{j-1}(k) = \sum_t f_j((k-1)/2 + t) h(t), \\ \forall\, \ell \in V_j,\ f_{j-1}(\ell) = \sum_t f_j(\ell + t) \tilde{h}(t). \end{cases}$$

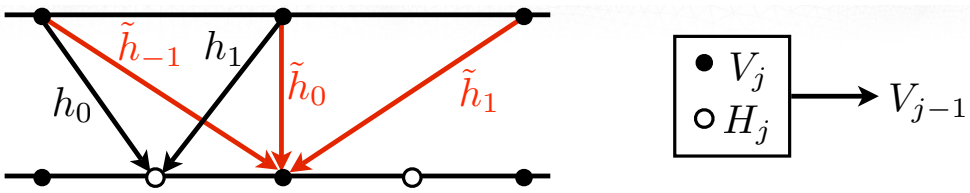


Figure 5.5: 1D subdivision scheme with filters h and $\tilde{h}$. The red curve represent the original signal f^0.

where the set of weights h and $\tilde{h}$ acts as local averaging operators. This averaging should be corrected at the boundary, and we use here cyclic boundary conditions which identifies 0 and $s_0 2^{-j}$ in V_j. Figure 5.5 shows a graphical display of these averaging operators.

One can write this subdivision steps as convolution by introducing the global set of weights

$$g = [\ldots, \tilde{h}(-1), h(0), \tilde{h}(0), h(1), \tilde{h}(1), \ldots]$$

since one has

$$f_{j-1} = (f_j \uparrow 2) * g \quad \text{where} \quad a \uparrow 2 = [\ldots, 0, a(-1), 0, a(0), 0, a(1), 0, \ldots].$$

This corresponds to the traditional description of the wavelet low-pass filtering [28].

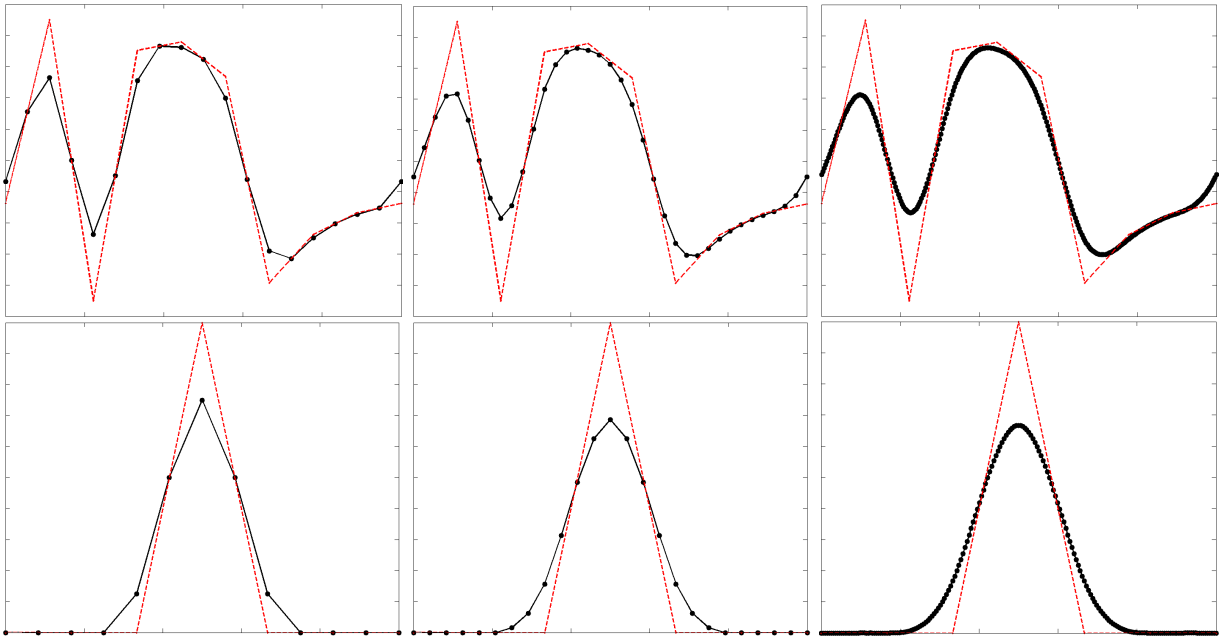


Figure 5.6: 1D subdivision of a signal. Bottom row shows the subdivision from an impulse signal, converging to the scaling function φ.

Figure 5.6 shows several steps of subdivision, starting from an initial vector of size $|V_0| = 10$.

One can apply this subdivision of functions to a pair of signals

$$(X_0, Y_0) : V_0 \to \mathbb{R}^2$$

which is a control polygon composed of points located in the plane. The subdivision curve converges to the limiting curve

$$(X_j, Y_j) \overset{j \to -\infty}{\longrightarrow} (X(t), Y(t))_{t=0}^{1} \subset \mathbb{R}^2.$$

An interesting property is that this curve is included in the convex hull of the control polygon

$$(X(t), Y(t))_t \subset \mathrm{Conv}(X_0, Y_0).$$

Figure 5.7 shows examples of subdivision curves.

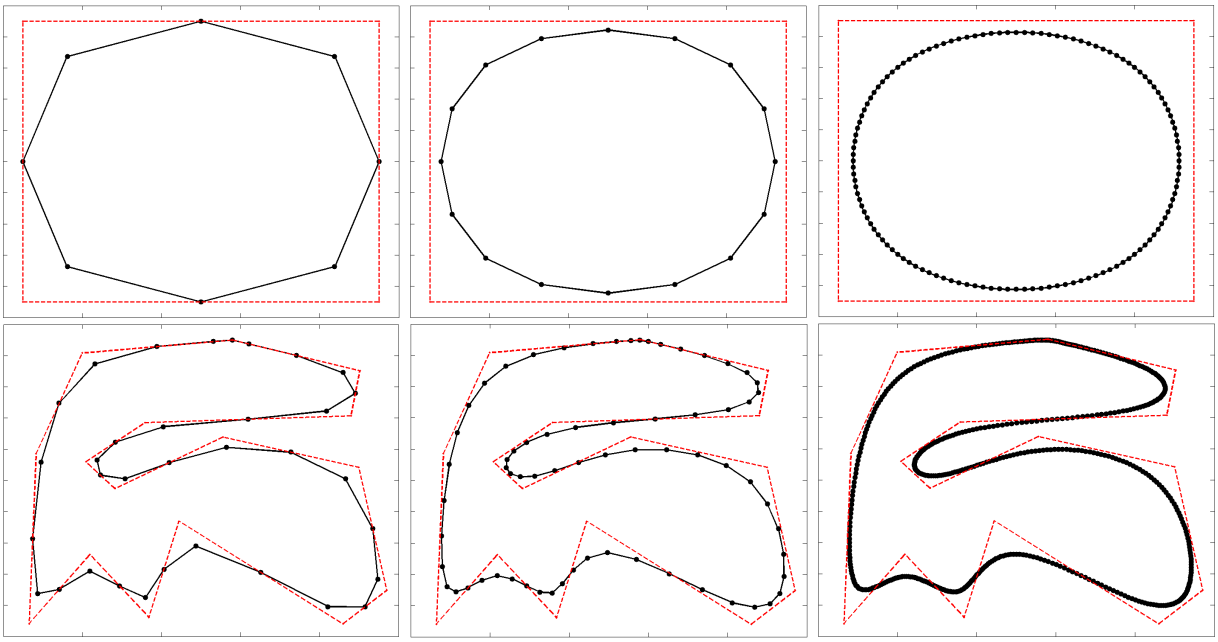


Figure 5.7: Two examples of subdivision curves. The red curve represent the original curve (X^0, Y^0).

5.3 Subdivision Surfaces

Subdivision schemes allows to compute a set of progressively refined vectors on a semi-regular mesh. More precisely, from an initial vector $f_0 \in \mathbb{R}^{|V_0|}$ defined on the coarse mesh M_0, local interpolation kernels computes iteratively vectors $f_j \in \mathbb{R}^{|V_j|}$ of finer resolution. When applied to 3 function $(f_0^i)_{i=1,2,3}$ defining the geometrical position of points in $\mathbb{R}^3$, this hierarchical construction defines a subdivision surface. These subdivisions surfaces are used extensively in computer aided geometry and computer graphics. One can see [16] for a survey of subdivision surfaces and their applications.

5.3.1 Interpolation Operators

In order to refine a vector $f_j \in \mathbb{R}^{|V_j|}$ defined on the vertex V_j of the mesh M_j, one uses two interpolators

$$P_j : \ell^2(V_j) \longrightarrow \ell^2(H_j) \qquad \text{and} \qquad \tilde{P}_j : \ell^2(V_j) \longrightarrow \ell^2(V_j). \tag{5.1}$$

A new refined function $f_{j-1} \in \mathbb{R}^{|V_{j-1}|}$ defined on the vertices $V_{j-1} = V_j \cup H_j$ of M_{j-1} is defined by applying these two refinement operators:

$$\forall\, \ell \in V_{j-1}, \quad f_{j-1}(\ell) = \begin{cases} (P_j f_j)(\ell) & \text{if} \quad \ell \in V_j, \\ (\tilde{P}_j f_j)(\ell) & \text{if} \quad \ell \in H_j. \end{cases}$$

Since $V_j \subset V_{j-1}$, the operator $\tilde{P}_j$ only modify slightly the value at vertex in V_j. On the other hand, the operator P_j creates new value at the vertices of H_j that are inserted between V_j and V_{j-1}.

In practical applications, these interpolating operators are local, meaning that the value of $(P_j f_j)(\ell)$ and $(\tilde{P}_j f_j)(\ell)$ depends only on values $f_j(\ell')$ for $\ell' \in V_j$ being close to $\ell \in V_{j-1}$, typically in the 1-ring or 2-ring vertex neighborhood.

A particularly important setting for subdivision scheme is when one apply the subdivision steps in parallel to three vectors (X_j, Y_j, Z_j) starting from three initial vectors describing the position in 3D space of a coarse mesh M_0. This allows to defines finer and finer spacial localization for the vertex of the refined meshes M_j. Figure 5.9 shows an example of such a subdivision surface. In order for the resulting infinitely refined surface to have good properties such as being continuous and even smooth, one needs to design carefully the interpolation operators. Next section gives examples of such operators.

5.3.2 Some Classical Subdivision Stencils

In order to define the interpolation operators P_j and $\tilde{P}_j$ of equation (5.1), one needs to use a naming convention for the neighborhoods of vertices.

For a vertex $\ell \in V_j$, the one ring neighborhood V_ℓ has already been defined in equation (3.1). It is the set of vertices adjacent to ℓ. In a regular point (that does not belongs to V_0 and not on a boundary of the mesh), its size is $|V_\ell| = 6$ since a point has 6 neighbors. This 1-ring is used to define $\tilde{P}_j$.

For a vertex $k \in H_j \subset V_{j-1}$, the butterfly neighborhood is a set of vertices in V_j close to k. This neighborhood is used to define P_j. The two immediate neighbors are

$$(v_k^1, v_k^2) \overset{\text{def.}}{=} \{v \in V_j \,;\, (v,k) \in E_{j-1}\}\,.$$

Two other vertices (w_k^1, w_k^2) are defined using the two faces adjacent to edge $(v_k^2, v_k^2) \in E_j$

$$f_k^1 = (v_k^1, v_k^2, w_k^1) \in F_j \quad \text{and} \quad f_k^2 = (v_k^1, v_k^2, w_k^2) \in F_j.$$

For edges E_j on the boundary of M_j, one one face is available, in which case we implicitly assume that $f_1 = f_2$ (reflecting boundary conditions). The four last vertices are defined using faces adjacent to f_1 and f_2:

$$\forall\, i,j = 1,2, \quad f_k^{i,j} \overset{\text{def.}}{=} (z_k^{i,j}, v_k^j, w_k^j) \in F_j \quad \text{with} \quad f_k^{i,j} \neq f_j.$$

Once again, reflecting boundary condition are applied for faces on the boundary of the mesh. The butterfly neighborhood is depicted on figure 5.8.

Linear Interpolating Scheme The simplest subdivision rule compute values along edge mide point using a simple linear interpolation as follow

$$\begin{cases} \forall\, k \in H_j, & (P_j f_j)(k) = \frac{1}{2}(f(v_k^1) + f(v_k^2)), \\ \forall\, \ell \in V_j, & (\tilde{P}_j f_j)(\ell) = f_j(\ell). \end{cases} \tag{5.2}$$

Since $\tilde{P}_j$ is the identity operator, this scheme is called interpolating. It means that value of f_0 on points of the coarse triangulation are kept during iteration of the subdivision.

Butterfly Interpolating Scheme The linear scheme creates function that are piecewise linear on each face of the coarse triangulation F_0. In order to create smooth surface, one needs to use more points in the butterfly neighborhood as follow

$$\begin{cases} \forall\, k \in H_j, & (P_j f_j)(k) = \frac{1}{2}\sum_{i=1}^2 f(v_k^i) + \frac{1}{8}\sum_{i=1}^2 f(w_k^i) - \frac{1}{16}\sum_{i,j=1}^2 f(z_k^{i,j}), \\ \forall\, \ell \in V_j, & (\tilde{P}_j f_j)(\ell) = f_j(\ell). \end{cases} \tag{5.3}$$

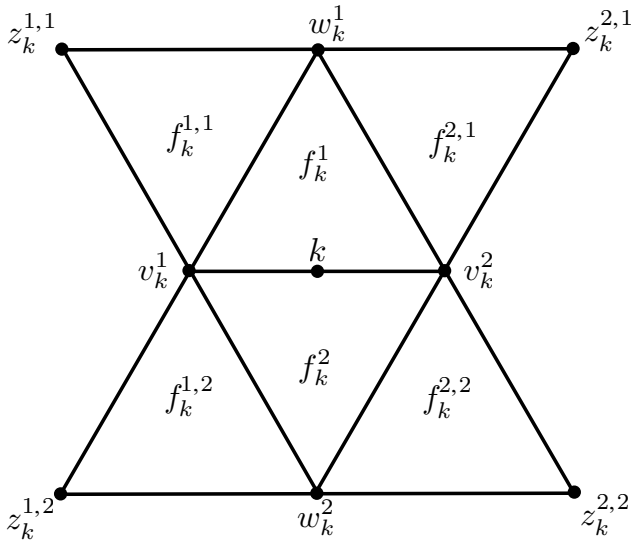


Figure 5.8: The butterfly neighborhood of a vertex $k \in H_j$.

Loop Approximating Scheme In order to gain flexibility in the subdivision design, one can also modify points in V_j during the iterations. This means that $\tilde{P}_j$ is not any more the identity, and that all the values will evolves during the iterations. The question of wether these iterated modification actually converge to a limit value is studied in the next section.

The Loop subdivision rule is defined as

$$\begin{cases} \forall\, k \in H_j, & (P_j f_j)(k) = \frac{3}{8} \sum_{i=1}^2 f(v_k^i) + \frac{1}{8} \sum_{i=1}^2 f(w_k^i), \\ \forall\, \ell \in V_j, & (\tilde{P}_j f_j)(\ell) = (1 - |V_\ell| \beta_{|V_\ell|}) f_j(\ell) + \beta_{|V_\ell|} \sum_{\ell' \in V_\ell} f_j(\ell'). \end{cases}$$

where the weights depends on the number of neighbors and are defined as

$$\beta_m \overset{\text{def.}}{=} \frac{1}{m} \left(\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4} \cos(2\pi/m) \right)^2 \right).$$

Other schemes. It is possible to define subdivision schemes using rules that do not involve a regular 1:4 splitting of each coarse face. For instance, in dual schemes such as the one depicted in figure 5.11, the faces of F_j are not included in F_{j-1} but only in F_{j-2}.

5.3.3 Invariant Neighborhoods

In order to study the convergence of subdivision schemes, one needs to consider independently each vertex $x \in V_{j_0(x)}$, where $j_0(x)$ is the coarser scale at which x appears

$$j_0(x) = \max \{ j \,;\, x \in V_j \}.$$

Original vertices satisfy $j_0(x) = 0$ and are the only one (except boundary vertices) that have a non-regular connectivity.

The vertex x belongs to the mesh $M_{j_0(x)}$ which is going to be refined through scales $j < j_0(x)$. In order to analyze this refinement, one needs to define an invariant neighborhood $V_j^x \subset V_j$ of x for each scale $j \leqslant j_0(x)$. These neighborhood are the set of points that are required to compute the operators P_j and $\tilde{P}_j$.

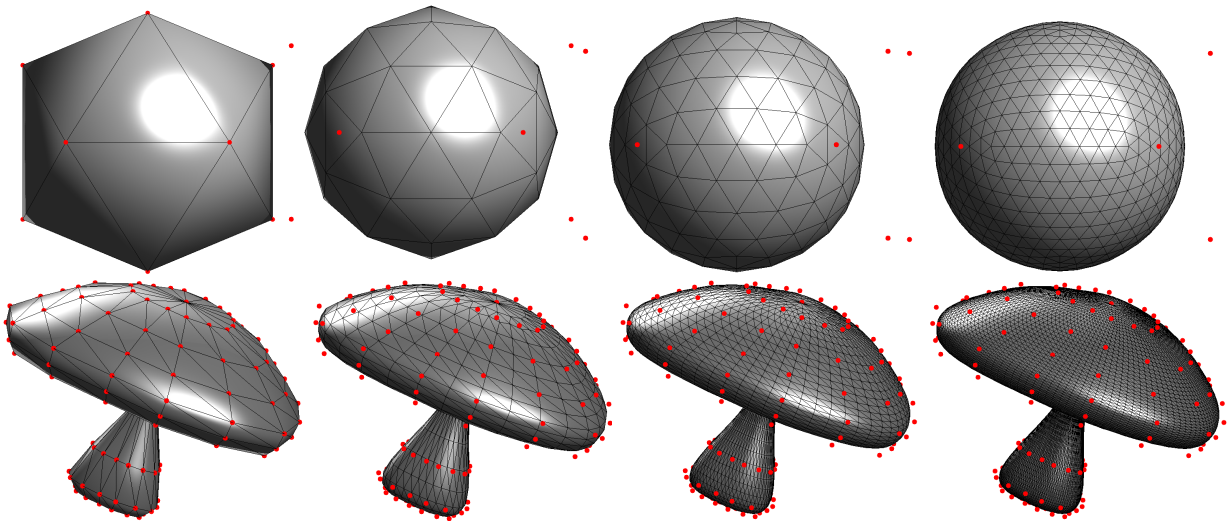


Figure 5.9: Examples of iterative subdivision using Loop scheme. The points (X_0, Y_0, Z_0) of the initial coarse mesh M_0 are shown in red.

More precisely, given a vector $f \in \ell^2(V_{j-1})$, the neighborhoods are required to satisfy

$$\begin{cases} \forall\, \ell \in V_{j-1}^x \cap V_j, & (\tilde{P}_j f)(\ell) \text{ depends only on } V_j^x \\ \forall\, k \in V_{j-1}^x \cap H_j, & (P_j f)(k) \text{ depends only on } V_j^x. \end{cases}$$

We further impose that all the invariant neighborhoods have the same size

$$\forall\, j \leqslant j_0(x), \quad \#V_j^x = m_x.$$

Figure 5.12 shows an example of invariant neighborhood which corresponds to the 2-ring $V_\ell^{(2)}$, as defined in (3.2).

Thanks to the invariance of these neighborhood systems, one can restrict the predictors around x and define

$$P_j^x : V_j^x \longrightarrow V_{j-1}^x \cap V_j \qquad \text{and} \qquad \tilde{P}_j^x : V_j^x \longrightarrow V_{j-1}^x \cap H_j.$$

The subdivision matrix $S_j^x \in \mathbb{R}^{m_x \times m_x}$ is then defined as matrix of the following mapping

$$(\tilde{P}_j^x, P_j^x) : V_x^j \longrightarrow V_x^{j-1}.$$

All the subdivision schemes studied in this chapter are invariant, meaning that the subdivision rule does not change through the scales j. This impose that the subdivision matrices are constant $S_j^x = S^x$. In fact, in all the examples given in the previous section, they only depends on the number $|V_x|$ of neighbors in the one ring of x.

5.3.4 Convergence of Subdivisions

The value at $x \in V_{j_0(x)}$ of a function $f_j \in \ell^2(V_j)$ obtained by subdividing at scale $j \leqslant j_0(x)$ an initial vector $f_0 \in \ell^2(V_0)$ can be computed as

$$f_j(x) = \left(S^x f_{j+1}^x\right)(x) = \left(\left(S^x\right)^{j_0(x)-j} f_{j_0(x)}^x\right)(x),$$

where the vector $f_i^x \in \mathbb{R}^{m_x}$ is the restriction of f_i to the set V_j^x.

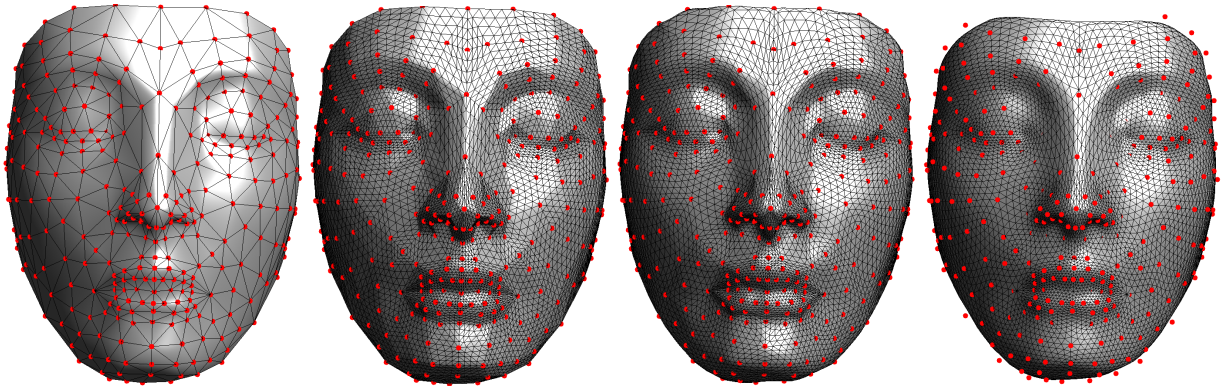


Figure 5.10: Examples of subdivision schemes. The points (X_0, Y_0, Z_0) of the initial coarse mesh M_0 are shown in red. Since the linear and butterfly scheme are interpolating, these points actually belongs to the limiting surface.

In order to analyze the limiting function resulting from an infinite number of subdivision, one can use the eigen vector decomposition of the matrix S^x

$$S^x = \tilde{\Phi} V \Lambda \Phi^{\mathrm{T}} \quad \text{where} \quad \begin{cases} \Phi^{\mathrm{T}} = \tilde{\Phi}^{-1}, \\ \Lambda = \mathrm{diag}(\lambda_i), \lambda_1 \leqslant \lambda_2 \leqslant \ldots \leqslant \lambda_{m_x}. \end{cases}$$

Since the subdivision matrix S_x is not symmetric, some of the eigenvalues might be complex, and we shall ignore this difficulty here. The fact that P_j and $\tilde{P}_j$ are predictor implies that the subdivision matrix has to satisfy $S^x 1 = 1$, meaning that $\tilde{\varphi}_1 = 1$ is an eigenvector associated to the eigenvalue 1. In the following we further makes the following assumption

$$1 = \lambda_1 < \lambda \stackrel{\text{def.}}{=} \lambda_2 = \lambda_3 < \lambda_4. \tag{5.4}$$

This hypothesis is satisfied by all the subdivision rules introduced in the previous section.

If one write $\Phi = (\varphi_i)_{i=1}^m$ and $\Phi = (\varphi_i)_{i=1}^m$, one has the following decomposition of a vector $f \in \mathbb{R}^{m_x}$

$$f = \sum_{i=1}^{m_x} \langle f, \varphi_i \rangle \tilde{\varphi}_i \quad \text{and} \quad (S^x)^k(x) = \sum_{i=1}^{m_x} \lambda_i^k \langle f, \varphi_i \rangle \tilde{\varphi}_i.$$

One thus has the following asymptotic expansion

$$\frac{1}{\lambda^k} (f - \langle f, \varphi_1 \rangle 1) = \langle f, \varphi_2 \rangle \tilde{\varphi}_2 + \langle f, \varphi_3 \rangle \tilde{\varphi}_3 + o(1). \tag{5.5}$$

This expression describes the asymptotic behavior of the subdivision scheme at zero order (position) and first order (tangents).

Theorem 25 (Convergence of the subdivision scheme)**.** *If the subdivision matrix S^x of a point x satisfies (5.4) then the subdivision process converges at x to the value*

$$f^j(x) \stackrel{j \to -\infty}{\longrightarrow} \langle f^x_{j_0(x)}, \varphi_1 \rangle.$$

The smoothness of the resulting function is more difficult to analyze. A particularly important setting is when one computes the subdivision of 3 function $p_0 = (X_0, Y_0, Z_0) \in \ell^2(V_0)^3$ corresponding to the position

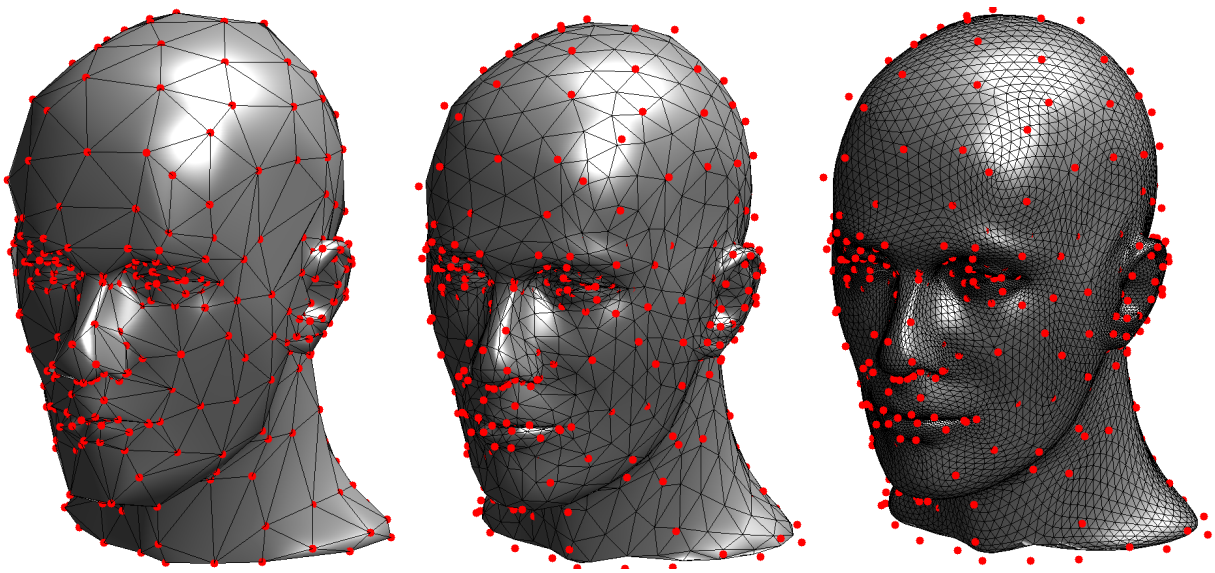


Figure 5.11: Surface after 0, 1 and 3 step of $\sqrt{3}$ subdivision [26].

in $\mathbb{R}^3$ (geometrical realization) of a coarse mesh M_0. In this case, the subdivided functions $p_j = (X_j, Y_j, Z_j)$ gives refined 3D meshes that converge uniforlmy to a continuous surfaces

$$p(x) = (X(x), Y(x), Z(x)) = (\langle X^x_{j_0}, \varphi_1\rangle, \langle Y^x_{j_0}, \varphi_1\rangle, \langle Z^x_{j_0}, \varphi_1\rangle).$$

Condition (5.4) nearly implies that the resulting surface is smooth. Indeed, the asymptotic expansion (5.5) shows that for a point x' near x in the subdivision domain, the differential vector can be well approximated as a projection on a 2D plane

$$p(x) - p(x') + o(1) \in \mathrm{Span}(\tau^x_2, \tau^x_3) \quad \text{where} \quad \tau^i(x) \stackrel{\text{def.}}{=} (\langle X^x_{j_0}, \varphi_i\rangle, \langle Y^x_{j_0}, \varphi_i\rangle, \langle Z^x_{j_0}, \varphi_i\rangle).$$

If the vectors τ^x_2 and τ^x_3 are linearly independent, they form a basis of the tangent plane at $p(x)$.

Example of the Loop subdivision. For the Loop interpolation operators defined in equation (5.3.2), the invariant neighborhood V^x_j correspond to the 2-ring of x in the triangulation G_j, as shown in figure 5.12. For a vertex with k neighbors, $|V_x| = k$, the size of these invariant neighborhood is $m_x = 3k + 1$. A particular neighboring for $k = 3$ is depicted in figure 5.12, together with an indexing in $\{0, \ldots, 3k = 9\}$ of the points in V^x_j and V^x_{j-1}. For this indexing, the subdivision matrix reads

$$\begin{pmatrix}
7 & & & & & & 3 & 3 & 3 \\
1 & 1 & & & 1 & & 1 & 10 & 1 & 1 \\
1 & & 1 & & 1 & 1 & & 1 & 10 & 1 \\
1 & & & 1 & & 1 & 1 & 1 & 1 & 10 \\
1 & & & & 1 & & & 3 & 3 & \\
1 & & & & & 1 & & & 3 & 3 \\
1 & & & & & & 1 & 3 & & 3 \\
1 & & & & & & & 3 & 1 & 1 \\
1 & & & & & & & 1 & 3 & 1 \\
1 & & & & & & & 2 & 1 & 3
\end{pmatrix}$$

where the 0's have been omitted and where the rows should be rescaled to sum to 1. The eigenvalues of this matrix satisfy $\lambda_1 = 1$ and $\lambda_2 = \lambda_3 = 1/3 > \lambda_4$.

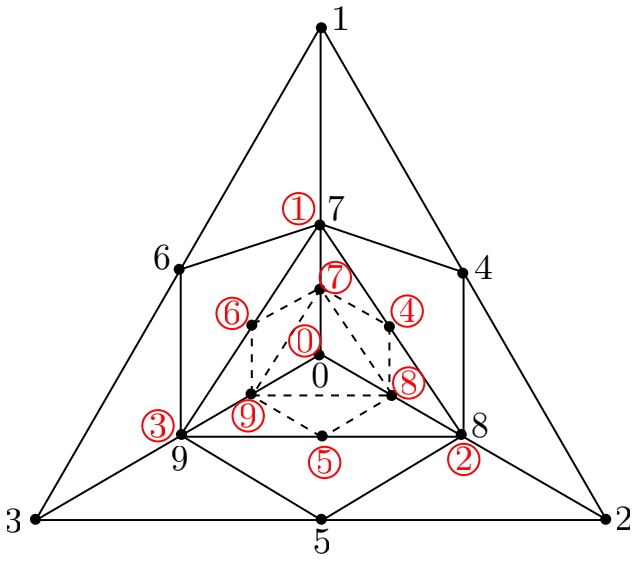


Figure 5.12: Invariant neighborhood V_j^x and V_{j-1}^x (indexing with red circles) of the Loop subdivision scheme for a vertex of valence $|V_\ell| = 0$. The number in $\{0, \dots, 9\}$ refers to the numbering of the vertices in V_j^x and V_{j-1}^x

5.4 Wavelets on Meshes

5.4.1 Multiscale Biorthogonal Bases on Meshes

The transforms considered in this section are multiscale and indexed by the set of nested grids $(V_j)_{L<j\leqslant J}$. This corresponds to computing a set of coefficients $(d_j)_{L<j\leqslant J} \cup f_J$ from an initial input signal f. These coefficients corresponds to inner products with basis vectors

$$\begin{cases} d_j \in \ell^2(H_j) & \text{where} \quad \forall\, k \in H_j,\ d_j(k) = \langle f,\, \psi_{j,k} \rangle, \\ f_J \in \ell^2(V_J) & \text{where} \quad \forall\, \ell \in V_J,\ f_J(\ell) = \langle f,\, \varphi_{J,\ell} \rangle. \end{cases}$$

By analogy with the wavelet setting, the vectors $\psi_{j,k} \in \mathbb{R}^n$ corresponds to primal wavelets and are intended to capture the details present in the signal f at a scale j, whereas the scaling vectors $\varphi_{J,k} \in \mathbb{R}^n$ capture the missing coarse approximation of f at scale J. This decomposition is stopped at any coarse scale $L < J \leqslant 0$.

In order to reconstruct the function f from this set of transformed coefficients, one needs to use a set of bi-orthogonal basis vectors

$$f = \sum_{L<j\leqslant J, k\in H_j} d_j(k)\tilde{\psi}_{j,k} + \sum_{\ell\in V_J} f_J(\ell)\tilde{\varphi}_{J,\ell}.$$

If this reconstruction formula holds for any scale $L < J \leqslant 0$, the set of vectors

$$(\psi_{j,k}, \varphi_{j,\ell})_{k\in H_j, \ell\in V_j}^{L<j\leqslant 0} \qquad \text{and} \qquad (\tilde{\psi}_{j,k}, \tilde{\varphi}_{j,\ell})_{k\in H_j, \ell\in V_j}^{L<j\leqslant 0}, \tag{5.6}$$

is said to be a pair of primal and dual multiscale bases (together with their scaling functions).

The following paragraph shows how one can modify such a pair of multiscale bases while still maintaining the biorthogonality property. This lifting process is useful to design multiscale bases with various properties on complicated domains.

5.4.2 The Lifting Scheme

The lifting scheme is a construction of multiscale biorthogonal bases introduced by Sweldens [43, 44]. It extends the traditional construction of wavelets in two main directions:

– As explained in [18], it allows to implement already existing filter banks more efficiency by splitting the computation into elementary blocks. This computational gain is described at the end of the section together with the factorization of wavelets into lifting steps.
– It allows to define multiscale transforms over domains that are not translation invariant. This section gives two examples of such transforms: a non-separable 2D wavelet transform and wavelets on triangulated meshes.

In order to build wavelets on triangulation, one can specialize the lifting scheme to a particular setting where only two lifting steps are applied.

Forward lifting scheme. The forward algorithm performs the transform

$$(f_{j-1}(\ell))_{\ell\in V_{j-1}} \quad \longrightarrow \quad (d_j(k))_{k\in H_j} \cup (f_j(\ell))_{\ell\in V_j}$$

by applying the following steps
– *Splitting:* this corresponds selecting the coefficient of $f_{j-1}(\ell)$ that are in V_j or in H_j

$$(f_{j-1}(\ell))_{\ell\in V_{j-1}} = (f_j(\ell))_{\ell\in V_j} \cup (f_j(\ell))_{\ell\in H_j}.$$

These two sets of coefficients are treated differently in the two remaining steps of the transform.
– *Predict step:* creates wavelets coefficients d_j by computing local differences between each coefficient in V_j and its neighbors in H_j

$$\forall\, k \in H_j, \quad d_j(k) = f_{j-1}(k) - \sum_{\ell\in V_j} p_j(k,\ell) f_{j-1}(\ell).$$

The coefficients $p_j(k,\ell)$ are weights that determine the predict operator

$$P_j : \left\{ \begin{array}{ccc} \ell^2(V_j) & \longrightarrow & \ell^2(H_j) \\ g & \longmapsto & h = P_j g \end{array} \right. \quad \text{where} \quad h(k) = \sum_{k\in H_j} p_j(k,\ell) g(\ell).$$

– *Update step:* enhance the properties of each remaining low pass coefficients $f_{j-1}(\ell)$ for $\ell \in V_j$ by pooling locally the wavelets coefficients $d_j(k)$ for k around ℓ

$$\forall\, \ell \in V_j, \quad f_j(\ell) = f_{j-1}(\ell) + \sum_{k\in H_j} u_j(\ell,k) d_j(k).$$

The coefficients $u_j(\ell,k)$ are weights that determine the update operator

$$U_j : \left\{ \begin{array}{ccc} \ell^2(H_j) & \longrightarrow & \ell^2(V_j) \\ h & \longmapsto & g = U_j h \end{array} \right. \quad \text{where} \quad g(\ell) = \sum_{k\in H_j} u_j(\ell,k) h(k).$$

Figure 5.13, top row, shows the block diagram associated to this forward lifting wavelet transform.

The iterations of the forward lifting transform can also be written in vector and operator format

$$\left\{ \begin{array}{l} d_j = f_{j-1}^{H_j} - P_j f_{j-1}^{V_j}, \\ f_j = f_{j-1}^{V_j} + U_j d_j = (\mathrm{Id}_{V_j} - U_j P_j) f_{j-1}^{V_j} + U_j f_{j-1}^{H_j}, \end{array} \right.$$

where g^A is the restriction of some vector g to the set A.

Backward lifting scheme. The backward transform algorithm does the reverse computation

$$(d_j(k))_{k\in H_j} \cup (f_j(\ell))_{\ell\in V_j} \quad \longrightarrow \quad (f_{j-1}(\ell))_{\ell\in V_{j-1}}$$

One of the main feature of the lifting scheme is that this is achieved by simply reversing the order of the lifting steps and interchanging +/- signs.

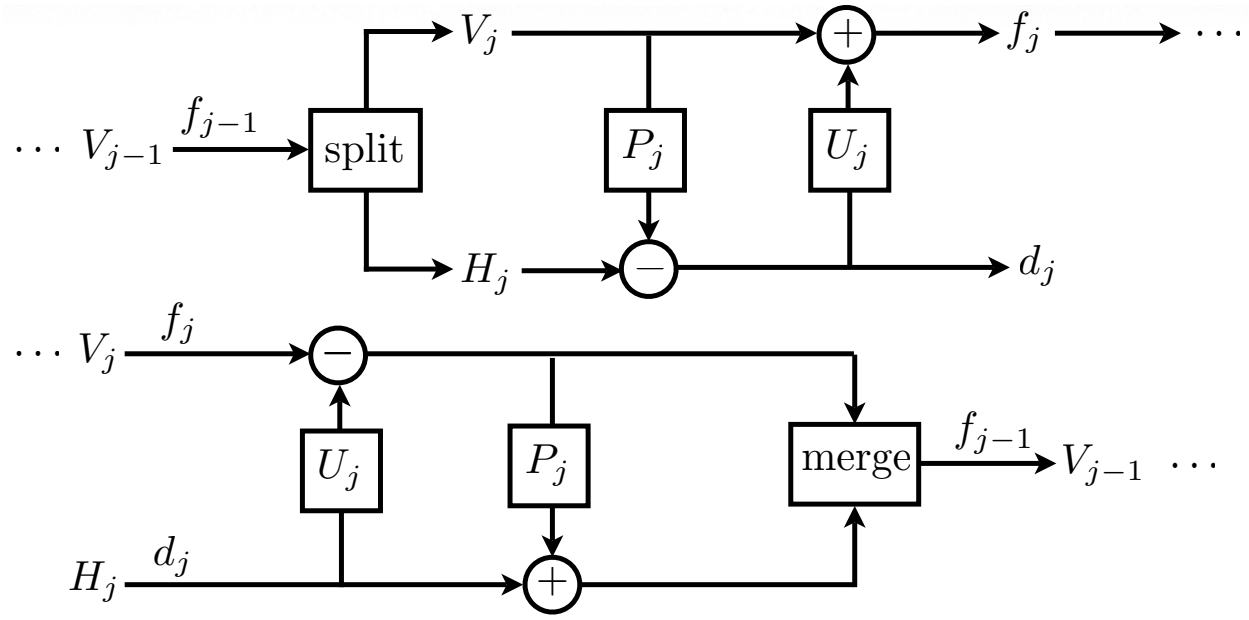


Figure 5.13: Block diagrams for the forward and backward lifting scheme.

– *Inverse update step:*

$$\forall\, \ell \in V_j, \quad f_{j-1}(\ell) = f_j(\ell) - \sum_{k \in H_j} u_j(\ell, k) d_j(k).$$

– *Inverse predict step:*

$$\forall\, k \in H_j, \quad f_{j-1}(k) = d_j(k) + \sum_{\ell \in V_j} p_j(k, \ell) f_{j-1}(\ell).$$

– *Merging:* makes the union of the coefficients computed in the two previous steps

$$(f_{j-1}(\ell))_{\ell \in V_{j-1}} = (f_j(\ell))_{\ell \in V_j} \cup (f_j(\ell))_{\ell \in H_j}.$$

Figure 5.13, top row, shows the block diagram associated to this backward lifting wavelet transform.

The lifting scheme is more general than the algorithm described in this section since several passes of predict/update steps can be applied to further enhance the properties of the resulting transform. However, the steps beyond the two initial ones are difficult to analyze, except in the notable exception of points sampled evenly on a 1D axes, where a factorization algorithm [18] allows to recover traditional wavelet filters.

5.4.3 Imposing vanishing moments.

The operator P_j is called a predictor since the values of $P_j f_{j-1}^{V_j}$ should typically be close to $f_{j-1}^{H_j}$ for the wavelet coefficients d_j to be small. Such predictors have already been constructed in equations (5.2), (5.3) and (5.3.2).

The operator U_j is called an update operator since the additional term $U_j d_j$ should enhance the properties of $f_{j-1}^{V_j}$. This update steps does not appears in the theory of subdivision surface and this section considers a local update operator which guaranty the conservation of the mean value when switching from f_{j-1} to f_j.

Polynomial vectors. In order to select predict and update operator that have good properties, one follow the insight gained from the analysis of the wavelet approximation of signal on the real line. In order to do

so, one need analyze the effect of a lifting wavelet transform on polynomials. The most basic constraint enforces one vanishing moment by imposing orthogonality with the constant vector $\Phi_0 = 1$. This constraint does not require to known the spacial location x_ℓ of each index $\ell \in V_L$. In order to impose higher order vanishing moments, one needs to assume some sampling pattern, for instance

$$\forall\, \ell \in V_L, \quad f(\ell) = \bar{f}(x_\ell) \qquad \text{where} \qquad x_\ell \in \mathbb{R}^q$$

and where $\bar{f}$ is a function defined on $\mathbb{R}^q$. For instance, the points x_ℓ might corresponds to a regular sampling of the line (this is the traditional wavelet setting) or to an irregular sampling of a 2D surface embedded in $\mathbb{R}^3$. The next paragraphs describe several situations with different sampling grids. Once the precise locations of the samples are known, one can for instance select Φ_s as some monomials of degree $(s_1, \ldots, s_q)$ over $\mathbb{R}^q$.

Vanishing moment and polynomials reproduction. Having defined these polynomial vectors, one requires that the following constraints are fulfilled.

– *Vanishing moments:* the wavelet coefficients of a low order polynomial should be 0, which implies that

$$\forall\, k \in H_j,\ \langle \Phi_s,\, \psi_{j,\ell} \rangle = 0. \tag{5.7}$$

– *Polynomial reproduction:* coarse coefficients f_j computed from a polynomial f_{f-1} should also be polynomials, which implies that

$$\forall\, \ell \in V_j,\ \langle \Phi_s,\, \varphi_{j,\ell} \rangle = \Phi_s(\ell) \tag{5.8}$$

In order for the wavelets and scaling function to satisfy conditions (5.7) and (5.8), the predict operator P_j and update operator U_j should be designed carefully. One can impose these constraint from the fine scale $j = L$ until the coarse scale $j = 0$. Indeed, if $(\varphi_{j-1,\ell}, \psi_{j-1,\ell})_{k,\ell}$ satisfy conditions (5.7) and (5.8), then, for the scale j

$$\forall\, s \in S, \quad \begin{cases} (5.7) & \iff & P_j \Phi_s^{V_j} = \Phi_s^{H_j}, \\ (5.8) & \iff & {U_j}^{\mathrm{T}} \left(\Phi_s^{V_j} + {P_j}^{\mathrm{T}} \Phi_s^{H_j} \right) = \Phi_s^{H_j}. \end{cases}$$

where $\Phi_s^A \in \ell^2(A)$ is the restriction of Φ_s to A.

In contrast, the constraint (5.8) on the update operator P_j is more involved and the next section shows how to handle it on a triangulation situations for only one vanishing moment $|S| = 1$.

5.4.4 Lifted Wavelets on Meshes

The lifted wavelet bases can be used to process signals $f \in \ell^2(V_L)$ where $\ell \in V_L$ index a sampling x_ℓ of an arbitrary surface. The construction of biorthogonal wavelets on triangulated mesh has been first proposed by Lounsbery et al. [27] and re-casted into the lifting scheme framework by Schroeder and Sweldens [38, 39].

Designing predict operators. The constraints (5.7) on the predictor P_j is easily solved. For instance, for each k, one selects only $|S|$ non vanishing weights $(p_j(k, \ell))_\ell$ and solves a small $|S| \times |S|$ linear system. Furthermore, in the case of a regular triangulation with edges of constant length, predictors with several vanishing moments have been already defined in (5.2), (5.3) and (5.3.2). Figure 5.14 shows the weights for these predictors.

One can choose any of these operators, and creates respectively linear, butterfly and Loop wavelets bases. All these predictors have one vanishing moment since they satisfy $P_j 1^{H_j} = 1^{V_j}$. In fact they have more vanishing moments if one consider polynomials Φ_s sampled at points $x_\ell \in \mathbb{R}^2$ of an hexagonal tiling with constant edge length. In practice, if the triangulation under consideration have edges with smoothly varying length, the resulting predictor are efficient to predict the value of smooth functions on the triangulation.

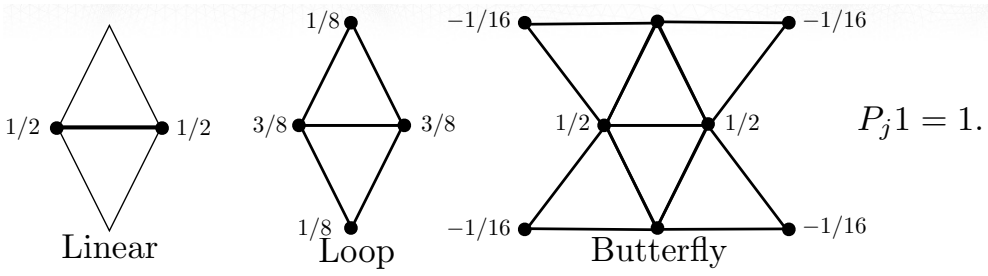


Figure 5.14: Predict operators on a triangulation.

Designing update operators. In order to ensure the reproduction of constant polynomials, we design the update operator so that it depends only on the direct neighbors in H_j of each point in V_j

$$\forall\, \ell \in V_j, \quad V_\ell = \{\gamma(\ell, \ell')\ ;\ (\ell, \ell') \in E_j\}\,.$$

One then looks for a valid update operator in the following form

$$\forall\, h \in \ell^2(H_j),\ \forall\, \ell \in V_j, \quad (U_j h)(\ell) = \lambda_\ell \sum_{k \in V_\ell} h(k), \tag{5.9}$$

where each λ_ℓ should be fixed in order for condition (5.8) to be satisfied.

In an semi-regular triangulation, $|V_\ell| = 6$ excepted maybe for some points in the coarse grid $\ell \in V_0$. In this setting, the values of λ_ℓ can be computed by a recursion through the scales. In an ideal triangulation where $|V_\ell| = 6$ for all ℓ, one can use a constant weight $\lambda_\ell = \lambda$.

For the predictors defined in (5.2), (5.3) and (5.3.2), one has

$$P_j^{\mathrm{T}} 1^{H_j} = 3 \times 1^{V_j} \qquad \text{and} \qquad U_j^{\mathrm{T}} 1^{V_j} = 6\lambda 1^{H_j}$$

so setting $\lambda_\ell = 1/24$ solves equation (5.8). Figure 5.15 shows examples of butterfly wavelets on a planar semi-regular triangulation.

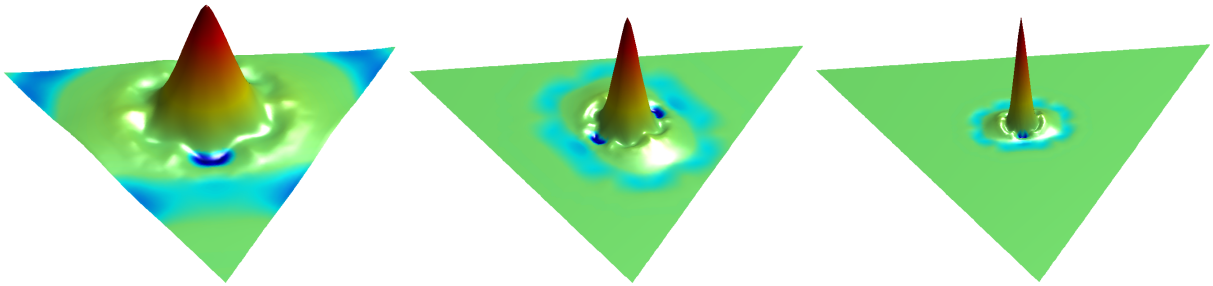


Figure 5.15: Example of wavelets $\psi_{j,k}$ on a semi-regular triangulation. The height over the triangle (together with the color) indicates the value of the wavelet vector.

5.4.5 Non-linear Mesh Compression

These wavelets can be used to perform an approximation of a function $f \in \ell^2(V_L)$ defined on the fine triangulation. For instance a wavelet approximation can be applied to each coordinate $f_i, i = 1, 2, 3$ of the

actual position $x_\ell = (f_1(\ell), f_2(\ell), f_3(\ell)) \in \mathbb{R}^3$ of the surface points, as done in [24, 25]. This leads to a scheme to approximate and compress a 3D surface using the lifted biorthogonal wavelets associated to the semi-regular triangulation. This is possible because these wavelets depend only on the combinatorial grids V_j and not on the precise position of the samples x_ℓ in 3D.

In order to perform a wavelet approximation in this biorthogonal basis, one uses a non-linear thresholding at $T > 0$

$$f = \sum_{(j,k)\in I_T} \langle f, \psi_{j,k}\rangle \tilde{\psi}_{j,k}$$

$$\text{where} \quad I_T = \left\{ (j,k) \;;\; k \in H_j \quad \text{and} \quad |\langle f, \psi_{j,k}\rangle| > T |\operatorname{supp}(\psi_{j,k})|^{-1/2} \right\}.$$

Note that for each coefficient the threshold T is scaled according to the size of the support of the wavelet in order to approximately normalize the wavelets in $\ell^2(V_L)$ norm.

Figure 5.16 shows an example of compression of the position of a vertex in 3D spaces as 3 functions defined on a semi-regular mesh. Figure 5.17 shows an example of compression of a spherical texture map which is a single function defined at each vertex of a semi-regular mesh obtained by subdividing an icosaedron.

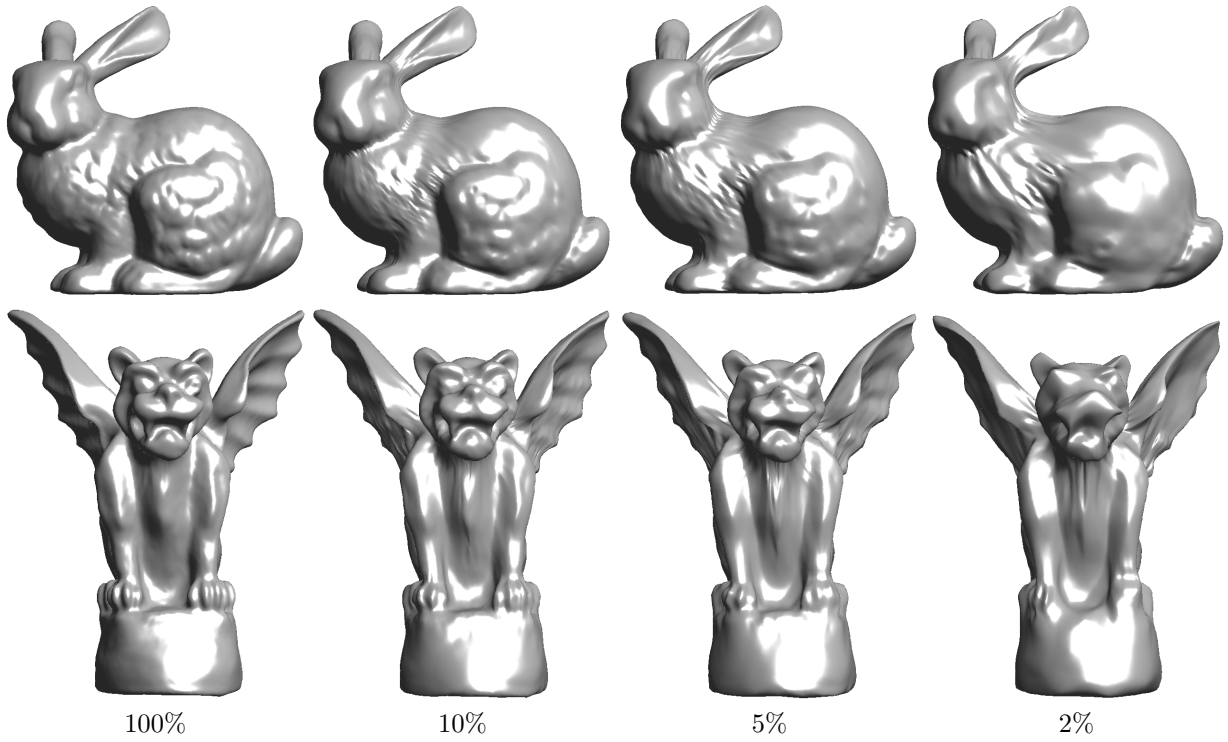


Figure 5.16: Non-linear wavelet mesh compression with a decreasing number of coefficients.

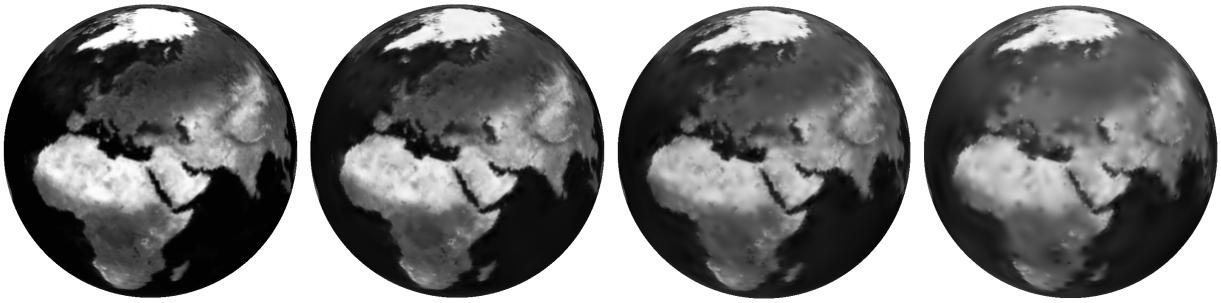


Figure 5.17: Non-linear spherical wavelet compression with a decreasing number of coefficients.

Chapter 6

Linear and Non-linear Approximation

This chapter studies the theory of signal and image approximation, and gives an application to lossy compression. This theoritical analysis is performed for continuous functions $f \in \mathrm{L}^2([0,1]^d)$ for $d = 1, 2$. This analysis is important to studies the performance of compression, denoising, and super-resolution applications.

6.1 Approximation

6.1.1 Approximation in an Ortho-basis

We consider an orthogonal basis $\mathcal{B} = \{\psi_m\}_m$ of $\mathrm{L}^2([0,1]^d)$, with for instance $d = 1$ (signals) or $d = 2$ (images). We recall that the decomposition of a signal in an orthonormal basis

$$f = \sum_{m \in \mathbb{Z}} \langle f, \psi_m \rangle \psi_m$$

gives back the original signal and thus produces no error. Processing algorithms modify the coefficients $\langle f, \psi_m \rangle$ and introduce some error.

The simplest processing computes an approximation by considering only a sub-set $I_M \subset \mathbb{Z}$ of M coefficients and performing the reconstruction from this subset

$$f_M \stackrel{\text{def.}}{=} \sum_{m \in I_M} \langle f, \psi_m \rangle \psi_m, \quad \text{where} \quad M = |I_M|.$$

The reconstructed signal f_M is the orthogonal projection of f onto the space

$$V_M \stackrel{\text{def.}}{=} \mathrm{Span}\{\psi_m \ ; \ m \in I_M\}.$$

Since V_M might depend on f, this projection $f \mapsto f_M$ might be non-linear.

Since the basis is orthogonal, the approximation error is

$$\|f - f_M\|^2 = \sum_{m \notin I_M} |\langle f, \psi_m \rangle|^2.$$

The important question is now to choose the set I_M, which might depend on the signal f itself.

6.1.2 Linear Approximation

Linear approximation is obtained by fixing once for all I_M, and thus using the same set of coefficients for all f. The mapping $f \mapsto f_M$ is a linear orthogonal projection on V_M, and it satisfies

$$(f + g)_M = f_M + g_M$$

For the Fourier basis, one usually selects the low-frequency atoms

$$I_M = \{-M/2+1, \ldots, M/2\}.$$

For a 1-D wavelet basis, one usually selects the coarse wavelets

$$I_M = \{m = (j,m) \; ; \; j \geqslant j_0\}$$

where j_0 is selected such that $|I_M| = M$.

Original f | Linear f_M^ℓ | Non-linear f_M^n

Figure 6.1: Linear versus non-linear wavelet approximation.

Figure 6.1, center, shows an example of such a linear approximation with wavelets. Linear approximation tends to performs poorly near singularities, because they introduce some blurring.

6.1.3 Non-linear Approximation

A non-linear approximation is obtained by choosing I_M depending on f. In particular, one would like to choose I_M to minimize the approximation error $\|f - f_M\|$. Since the basis is orthogonal, this is achieved by selecting the M largest coefficients in magnitude

$$I_M = \{M \text{ largest coefficients } |\langle f, \psi_m \rangle|\}.$$

This can be equivalently obtained using a thresholding

$$I_M = \{m \; ; \; |\langle f, \psi_m \rangle| > T\}$$

where T depends on the number of coefficients M,

$$M = \#\{m \; ; \; |\langle f, \psi_m \rangle| > T\}.$$

Computation of the threshold. There is a bijective 1:1 mapping between T and M obtained by ordering the coefficient magnitudes $|\langle f, \psi_m \rangle|$ by decaying order,

$$T = d_M \quad \text{where} \quad \{d_m\}_{m=0}^{N-1} = \{|\langle f, \psi_m \rangle|\}_0^{N-1} \quad \text{and} \quad d_m \geqslant d_{m+1}. \tag{6.1}$$

Figure 6.2 shows this mapping between M and T.

The following proposition show that the decay of the ordered coefficients is linked to the non-linear approximation decay.

Proposition 14. *One has*

$$d_m = O(m^{-\frac{\alpha+1}{2}}) \quad \Longleftrightarrow \quad \|f - f_M\|^2 = O(M^{-\alpha}). \tag{6.2}$$

Proof. One has

$$\|f - f_M\|^2 = \sum_{m>M} d_m^2$$

and

$$d_M^2 \leqslant \frac{2}{M} \sum_{m=M/2+1}^{M} d_m^2 \leqslant \frac{2}{M} \sum_{m>M/2} d_m^2 = \frac{2}{M} \|f - f_{M/2}\|^2.$$

□

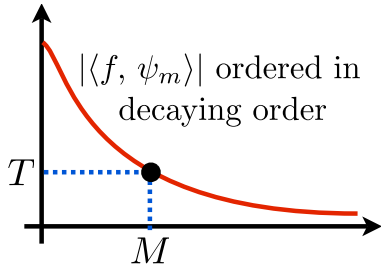


Figure 6.2: Decay of the ordered coefficients and determination of the threshold for non-linear approxiation.

Hard thresholding. The non-linear approximation is re-written as

$$f_M = \sum_{|\langle f, \psi_m \rangle| > T} \langle f, \psi_m \rangle \psi_m = \sum_m S_T(\langle f, \psi_m \rangle) \psi_m, \tag{6.3}$$

where

$$S_T(x) = \begin{cases} x & \text{if} \quad |x| > T \\ 0 & \text{if} \quad |x| \leqslant T \end{cases} \tag{6.4}$$

is the hard thresholding, that is displayed in Figure 6.3.

6.2 Signal and Image Modeling

A signal model is a constraint $f \in \Theta$, where $\Theta \subset \mathrm{L}^2([0,1]^d)$ is a set of signals one is interested in. Figure 6.4 shows different class of models for images, that we describe in the following paragraph.

6.2.1 Uniformly Smooth Signals and Images

Signals with derivatives. The simplest model is made of uniformly smooth signals, that have bounded derivatives

$$\Theta = \left\{ f \in \mathrm{L}^2([0,1]^d) \; ; \quad \|f\|_{\mathrm{C}^\alpha} \leqslant C \right\}, \tag{6.5}$$

where $C > 0$ is a fixed constant, and where in 1-D

$$\|f\|_{\mathrm{C}^\alpha} \overset{\text{def.}}{=} \max_{k \leqslant \alpha} \left\| \frac{\mathrm{d}^k f}{\mathrm{d}t^k} \right\|_\infty.$$

This extends to higher dimensional signals by considering partial derivatives along each direction.

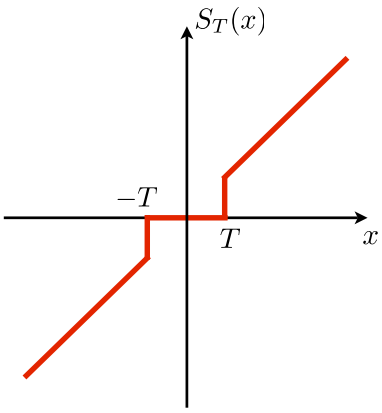
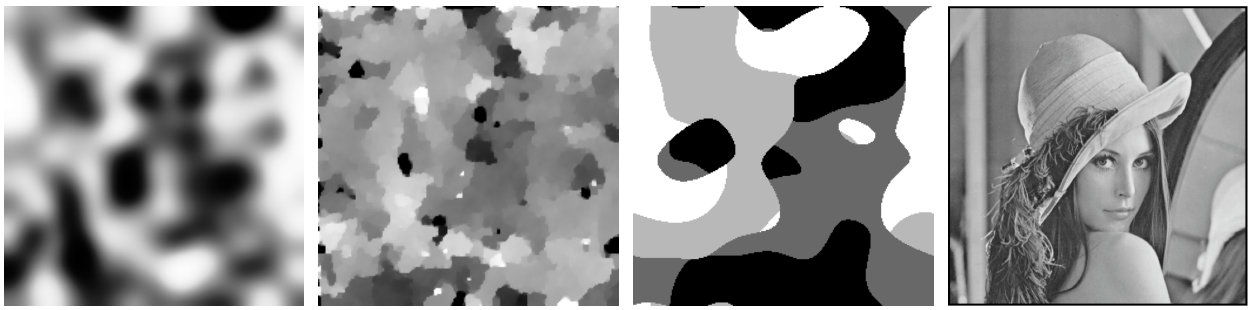


Figure 6.3: Hard thresholding.



Regular Bounded variations Cartoon Natural

Figure 6.4: Examples of different kinds of image models.

Sobolev smooth signals and images. A smooth C^α signals in (6.5) has derivatives with bounded energy, so that

$$\frac{\mathrm{d}^\alpha f}{\mathrm{d}t^\alpha}(t) = f^{(\alpha)}(t) \in \mathrm{L}^2([0,1]).$$

Using the fact that

$$\hat{f}_m^{(\alpha)} = (2i\pi m)^\alpha \hat{f}_m$$

where $\hat{f}$ is the Fourier coefficient defined in (2.2), except we are here on $\mathbb{R}/\mathbb{Z}$ in place of $\mathbb{R}/2\pi\mathbb{Z}$,

$$\hat{f}_n \overset{\text{def.}}{=} \int_0^1 e^{-2\mathrm{i}\pi n x} f(x)\mathrm{d}x,$$

one defines a so-called Sobolev functional

$$\|f\|_{\mathrm{Sob}(\alpha)}^2 = \sum_{m\in\mathbb{Z}} |2\pi m|^{2\alpha} |\langle f,\, e_m\rangle|^2, \tag{6.6}$$

that satisfies $\|f\|_{\mathrm{Sob}(\alpha)} = \|f^{(\alpha)}\|$ for smooth functions. This Sobolev functional is extended to signals that have derivatives in the sense of distribution in $\mathrm{L}^2([0,1])$.

This definition extends to distributions and signals $f \in \mathrm{L}^2([0,1]^d)$ of arbitrary dimension d as

$$\|f\|_{\mathrm{Sob}(\alpha)}^2 = \sum_{m\in\mathbb{Z}^d} (2\pi\|m\|)^{2\alpha} |\langle f,\, e_m\rangle|^2, \tag{6.7}$$

The C^α-Sobolev model

$$\Theta = \left\{ f \in \ell^2([0,1]^d) \; ; \; \max_{k \leqslant \alpha} \|f\|^2_{\mathrm{Sob}(k)} \leqslant C \right\} \tag{6.8}$$

generalizes the C^α smooth image model (6.5).

Figure 6.5 shows images with an increasing Sobolev norm for $\alpha = 2$.

Figure 6.5: Images with increasing Sobolev norm.

6.2.2 Piecewise Regular Signals and Images

Piecewise smooth signals. Piecewise smooth signals in 1-D are functions $f \in \mathrm{L}^2([0,1])$ that are C^α smooth outside a set of less than K pointwise discontinuities

$$\Theta = \left\{ f \in \mathrm{L}^2([0,1]) \; ; \; \exists\, (t_i)_{i=0}^{K-1}, \; \|f_{(t_i,t_{i+1})}\|_{\mathrm{C}^\alpha} \leqslant C \right\} \tag{6.9}$$

where $f_{(t_i,t_{i+1})}$ is the restriction of f to the open interval (t_i, t_{i+1}).

Piecewise smooth images. Piecewise smooth images are 2-D functions $f \in \mathrm{L}^2([0,1]^2)$ that are C^α regular outside a set of less than K curves that have a finite perimeter

$$\Theta = \left\{ f \in \mathrm{L}^2([0,1]^2) \; ; \; \exists\, \Gamma = (\gamma_i)_{i=0}^{K-1}, \; \|f\|_{\mathrm{C}^\alpha(\Gamma^c)} \leqslant C_1 \quad \text{and} \quad |\gamma_i| \leqslant C_2 \right\} \tag{6.10}$$

where $|\gamma_i|$ is the length of the curve γ_i and where $\|f\|_{\mathrm{C}^\alpha(\Gamma^c)}$ is the maximum norm of the derivatives of f outside the set of curves Γ.

Segmentation methods such as the one proposed by Mumford and Shah [30] implicitly assume such a piecewise smooth image modem.

6.2.3 Bounded Variation Signals and Images

Signals with edges are obtained by considering functions with bounded variations

$$\Theta = \left\{ f \in \mathrm{L}^2(\mathbb{R}^d) \; ; \; \|f\|_\infty \leqslant C_1 \quad \text{and} \quad \|f\|_{\mathrm{TV}} \leqslant C_2 \right\}. \tag{6.11}$$

For $d = 1$ and $d = 2$, this model generalizes the model of piecewise smooth signals (6.9) and images (6.10).

The total variation of a smooth function is

$$\int \|\nabla f(x)\| \mathrm{d}x$$

where

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_i} \right)_{i=0}^{d-1} \in \mathbb{R}^d$$

is the gradient vector at x. The total variation is extended to discontinuous images, that might for instance exhibit jumps across singular curves (the edges of the image). In particular, the total variation of a piecewise smooth image is the sum of the lengths of its level sets

$$\|f\|_{\mathrm{TV}} = \int_{-\infty}^{\infty} |\mathcal{L}_t(f)| \mathrm{d}t < +\infty \quad \text{where} \quad \mathcal{L}_t(f) = \{x \,;\, f(x) = t\}\,, \tag{6.12}$$

and where $|\mathcal{L}_t(f)|$ is the length of $\mathcal{L}_t(f)$. For a set $\Omega \subset \mathbb{R}^2$ with finite perimeter $|\partial\Omega|$, then

$$\|1_\Omega\|_{\mathrm{TV}} = |\partial\Omega|.$$

The model of bounded variation was introduced in image processing by Rudin, Osher and Fatemi [36].

6.2.4 Cartoon Images

The bounded variation image model (6.11) does not constrain the smoothness of the level set curves $\mathcal{L}_t(f)$. Geometrical images have smooth contour curves, which should be taken into account to improve the result of processing methods.

The model of C^α cartoon images is composed of 2-D functions that are C^α regular outside a set of less that K regular edge curves γ_i

$$\Theta = \left\{ f \in \mathrm{L}^2([0,1]^2) \,;\, \exists\, \Gamma = (\gamma_i)_{i=0}^{K-1},\ \|f\|_{\mathrm{C}^\alpha(\Gamma^c)} \leqslant C_1 \quad \text{and} \quad \|\gamma_i\|_{\mathrm{C}^\alpha} \leqslant C_2 \right\} \tag{6.13}$$

where each γ_i is a arc-length parameterization of the curve $\gamma_i : [0, A] \mapsto [0,1]^2$. Figure 6.6 shows cartoon images with increasing total variation $\|f\|_{\mathrm{TV}}$.

Figure 6.6: Cartoon image with increasing total variation.

Typical images might also be slightly blurred by optical diffraction, so that one might consider a blurred cartoon image model

$$\tilde{\Theta} = \left\{ \tilde{f} = f \star h \in \mathrm{L}^2([0,1]^2) \,;\, f \in \Theta \quad \text{and} \quad h \in \mathcal{H} \right\} \tag{6.14}$$

where Θ is the model of sharp (unblurred) images (6.13) and $\mathcal{H}$ is a set of constraints on the blurring kernel, for instance $h \geqslant 0$ should be smooth, localized in space and frequency. This unknown blurring makes difficult brute force approaches that detects the edges location Γ and then process the regular parts in $[0,1]^2 \backslash \Gamma$.

Figure 6.7 shows examples of images in Θ and $\tilde{\Theta}$.

6.3 Efficient approximation

6.3.1 Decay of Approximation Error

To perform an efficient processing of signals or images in Θ, the goal is to design an orthogonal basis such that the non-linear approximation error $\|f - f_M\|$ decays as fast as possible to 0 when M increases.

Figure 6.7: Examples of cartoon images: sharp discontinuities (left and center) and smooth discontinuities (right).

Polynomial error decay. This error decay measured using a power law

$$\forall\, f \in \Theta,\ \forall\, M, \quad \|f - f_M\|^2 \leqslant C_f M^{-\alpha} \tag{6.15}$$

where α is independent of f and should be as large as possible. The parameter α depends on the basis and on Θ. It is a class-complexity parameter that describes the overall complexity of signals in Θ with respect to the orthogonal basis one considers for approximation. The parameter C_f depends on f and describes the complexity of the signal f within its class Θ.

Relevance for compression, denoising and inverse problems. Monitoring the decay of approximation error is not only interesting from a mathematical point of view. Section 7.1 shows that the compression error is close to the non-linear approximation error. Bases that are efficient for approximation are thus also efficient for compression.

Chapter **??** shows that a similar conclusion holds for non-linear denoising with thresholding. Efficient denoisers are obtained by performing a non-linear approximation of the noisy image in a well chosen basis. The average denoising error with respect to a random noise is closely related to the approximation error.

Chapter **??** shows that ill-posed inverse problems such as super-resolution of missing information can be solved by taking advantage of the compressibility of the signal or the image in a well chosen basis. A basis that is efficient for approximation of the high resolution signal is needed to recover efficiently missing information. The performance of these schemes is difficult to analyze, and the basis atoms must also be far enough from the kernel of the operator that removes information.

Comparison of signals. For a fixed basis (for instance wavelets), the decay of $\|f - f_M\|$ allows one to compare the complexity of different images. Figure 6.9 shows that natural images with complicated geometric structures and textures are more difficult to approximate using wavelets.

Since the approximation error often decays in a power-low fashion (6.15), the curves are displayed in a log-log plot, so that

$$\log(\|f - f_M\|^2) = \mathrm{cst} - \alpha \log(M)$$

and hence one should expect an affine curve with slope $-\alpha$. Due to discretization issue, this is only the case for value of $M \ll N$, since the error quickly drops to zero for $M \approx N$.

6.3.2 Comparison of bases.

For a given image f, one can compare different ortho-bases using the decay of $\|f - f_M\|$. Figure 6.11 shows the efficiency of several bases to approximate a fixed natural image with contours and textures. The Fourier basis described in Section **??** is highly innefficient because of periodic boundary artifact and the global

Smooth Cartoon Natural #1 Natural #2

Figure 6.8: Several different test images.

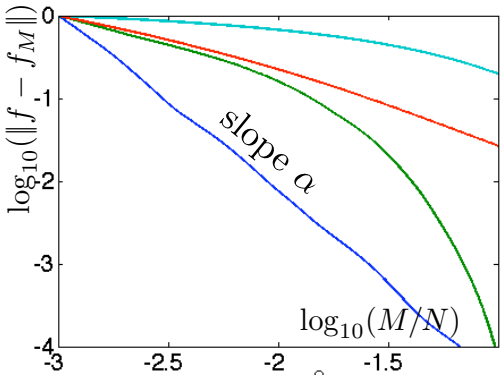


Figure 6.9: Comparison of approximation error decay in wavelets for different images shown in Figure 6.8.

support of its atoms that fail to capture contours. The cosine basis uses symmetric boundary conditions and thus removes the boundary artifacts, but it still not able to resolve efficiently localized features. The local DCT basis corresponds to the union of local cosine bases defined on small square patches. It is more efficient since its atoms have a local support. However, it gives bad approximation for a small number M of kept coefficients, because of blocking artifacts. The isotropic wavelet basis detailed in Section **??** gives the best approximation results because its is both composed of localized atoms and does not have a block structure but rather a multiresolution structure.

6.4 Fourier Linear Approximation of Smooth Functions

The smooth signal and image model (6.5) assumed that the analog function have bounded α continuous derivatives. A function f with a large α has more smoothness, and is thus simpler to approximate. Figure 6.5 shows images with increasing smoothness.

6.4.1 1-D Fourier Approximation

A 1-D signal $f \in \mathrm{L}^2([0,1])$ is associated to a 1-periodic function $f(t+1) = f(t)$ defined for $t \in \mathbb{R}$.

Fourier
SNR=17.1dB

Cosine
SNR=17.5dB

Local DCT
SNR=18.4dB

Wavelets
SNR=19.3dB

Figure 6.10: Comparison of approximation errors for different bases using the same number $M = N/50$ of coefficients.

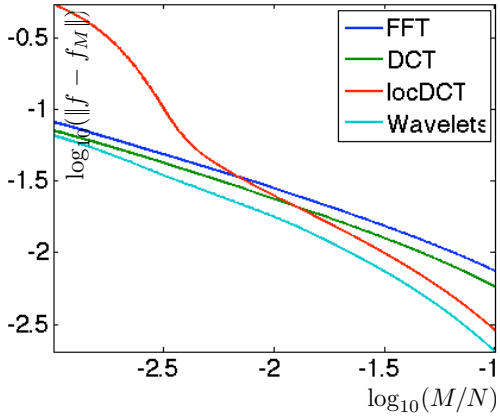


Figure 6.11: Comparison of approximation error decay for different bases.

Low pass approximation. We consider a linear Fourier approximation, that only retains low frequencies

$$f_M^{\text{lin}} = \sum_{m=-M/2}^{M/2} \langle f, e_m \rangle e_m$$

where we use the 1-D Fourier atoms

$$\forall\, n \in \mathbb{Z}, \quad e_m(t) \stackrel{\text{def.}}{=} e^{2i\pi m t}.$$

We note that f_M actually requires $M+1$ Fourier atoms.

Figure 6.12 shows examples of such linear approximation for an increasing value of M. Since the original function f is singular (no derivative), this produces a large error and one observe ringing artifacts near singularities.

This low pass approximation corresponds to a filtering, since

$$f_M = \sum_{m=-M/2}^{M/2} \langle f, e_m \rangle e_m = f \star h_M \quad \text{where} \quad \hat{h}_M \stackrel{\text{def.}}{=} 1_{[-M/2, M/2]}.$$

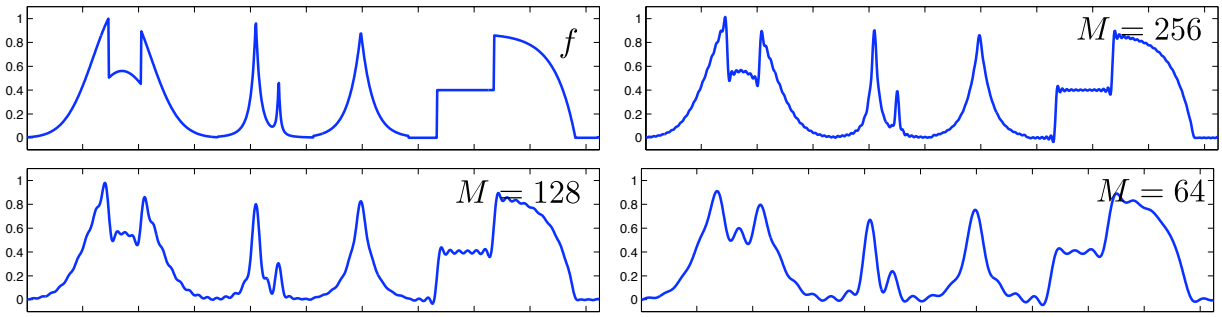


Figure 6.12: Fourier approximation of a signal.

Here, h_M is the so-called Dirichlet kernel (see also the chapter on Fourier).

The following proposition shows that this approximation decays fast for $\mathcal{C}^\alpha$ signals.

Proposition 15. *One has for $f \in \mathcal{C}^\alpha(\mathbb{R}/\mathbb{Z})$*

$$\|f - f_M^{lin}\| = O(M^{-2\alpha+1}).$$

Proof. Using integration by part, one shows that for a C^α function f in the smooth signal model (6.5), one has

$$|\langle f, e_m\rangle| \leqslant |2\pi m|^{-\alpha}\|f^{(\alpha)}\|.$$

This implies

$$\|f - f_M^{\mathrm{lin}}\| = \sum_{|m|>M/2} |\langle f, e_m\rangle|^2 \leqslant \|f^{(\alpha)}\| \sum_{|m|>M/2} |2\pi m|^{-2\alpha} = O(M^{-2\alpha+1}).$$

□

We show next that a slightly modified proof gives a better decay (assuming $f^{(\alpha)}$ is in $L^2(\mathbb{R}/\mathbb{Z})$) and that this conclusion is valid for a larger class of Sobolev functions.

Proposition 16. *For signal f in the Sobolev model (6.8), one has*

$$\|f - f_M^{lin}\|^2 \leqslant C\|f^{(\alpha)}\|^2 M^{-2\alpha}. \tag{6.16}$$

Proof. One has

$$\|f^{(\alpha)}\|^2 = \sum_m |2\pi m|^{2\alpha}|\langle f, e_m\rangle|^2 \geqslant \sum_{|m|>M/2} |2\pi m|^{2\alpha}|\langle f, e_m\rangle|^2 \tag{6.17}$$

$$\geqslant (\pi M)^{2\alpha} \sum_{m>M/2} |\langle f, e_m\rangle|^2 \geqslant (\pi M)^{2\alpha}\|f - f_M^{\mathrm{lin}}\|^2. \tag{6.18}$$

□

One can also shows that this asymptotic error decay is optimal, and that the non-linear approximation error in Fourier also decays like $O(M^{-2\alpha})$.

For a signal in the piecewise smooth model (6.9), such as the one shows in Figure 6.12, one only has a slow decay of the linear and non-linear approximation error

$$\|f - f_M\|^2 \leqslant C_f M^{-1} \tag{6.19}$$

and Fourier atoms are not anymore optimal for approximation.

6.4.2 Sobolev Images

This analysis caries over to images and higher dimensional datasets by considering a Sobolev functional (6.7) for $d > 1$.

The linear and non-linear approximation of an α-regular Sobolev image then satisfy

$$\|f - f_M\|^2 \leqslant C\|f^\alpha\|^2 M^{-\alpha}.$$

For d-dimensional data $f : [0,1]^d \to \mathbb{R}$, one would have an error decay of $O(M^{-2\alpha/d})$.

For an image in the piecewise smooth model (6.10), the linear and non-linear error decays are slow,

$$\|f - f_M\|^2 \leqslant C_f M^{-1/2}, \tag{6.20}$$

and Fourier atoms are not anymore optimal for approximation.

Figure 6.13: Linear (top row) and non-linear (bottom row) Fourier approximation.

6.5 Wavelet Approximation of Piecewise Smooth Functions

Wavelet approximation improve significantly over Fourier approximation to capture singularities. This is due to the localized support of wavelets.

6.5.1 Decay of Wavelet Coefficients

To efficiently approximate regular parts of signals and images, one uses wavelet with a large enough number p of vanishing moments

$$\forall\, k < p, \quad \int \psi(x) x^k \mathrm{d}x = 0.$$

This number p should be larger than the regularity α of the signal outside singularities (for instance jumps or kinks).

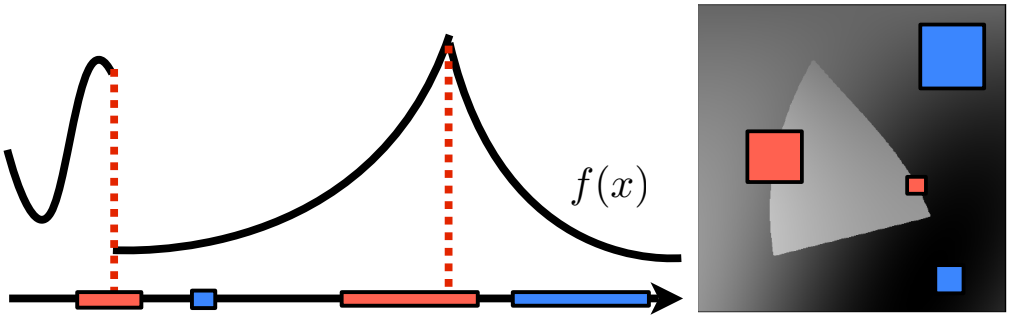


Figure 6.14: Singular par of signals (left) and image (right).

To quantify the approximation error decay for piecewise smooth signals (6.9) and images (6.10), one needs to treat differently wavelets that are in regular and singular areas. Figure 6.14 shows for a signal and an image the localization of singular and regular parts.

Proposition 17. *If f is C^α on $\mathrm{supp}(\psi_{j,n})$, with $p \geqslant \alpha$, one has*

$$|\langle f,\, \psi_{j,n}\rangle| \leqslant C_f \|\psi\|_1 2^{j(\alpha+d/2)}. \tag{6.21}$$

In general, one always has for bounded f

$$|\langle f,\, \psi_{j,n}\rangle| \leqslant \|f\|_\infty \|\psi\|_1 2^{j\frac{d}{2}}.$$

Proof. If f is C^α on $\mathrm{supp}(\psi_{j,n})$, with $p \geqslant \alpha$, then one can perform a Taylor expansion of f around the point $2^j n$

$$f(x) = P(x - 2^j n) + R(x - 2^j n) = P(2^j t) + R(2^j t)$$

where $\deg(P) < \alpha$ and

$$|R(x)| \leqslant C_f \|x\|^\alpha.$$

One then bounds the wavelet coefficient

$$\langle f,\, \psi_{j,n}\rangle = \frac{1}{2^{j\frac{d}{2}}} \int f(x)\psi\left(\frac{x - 2^j n}{2^j}\right) \mathrm{d}x = 2^{j\frac{d}{2}} \int R(2^j t)\psi(t)\mathrm{d}t$$

where we have performed the change of variable $t = \frac{x-2^j n}{2^j}$. This shows that (6.21) holds. Property (17) is straightforward. □

6.5.2 1-D Piecewise Smooth Approximation

For 1-D signal in the piecewise regular model (6.9), large wavelets coefficients $\langle f,\, \psi_{j,n}\rangle$ are clustered around the singularities of the signal. We call $\mathcal{S} \subset [0,1]$ the finite set of singular points.

Theorem 26. *f is in the piecewise smooth signal model (6.9), the non-linear approximation error in wavelet obeys*

$$\|f - f_M\|^2 = O(M^{-2\alpha}). \tag{6.22}$$

Proof. The proof is split in several parts.

Step 1. Coefficient segmentation. The singular support at scale 2^j is the set of coefficients corresponding to wavelets that are crossing a singularity

$$\mathcal{C}_j = \{n \; ; \; \mathrm{supp}(\psi_{j,n}) \cap \mathcal{S} \neq \emptyset\} \tag{6.23}$$

It has a constant size because of the dyadic translation of wavelets

$$|\mathcal{C}_j| \leqslant K|\mathcal{S}| = \text{constant}.$$

Using (6.21) for $d = 1$, the decay of regular coefficients is bounded as

$$\forall\, n \in \mathcal{C}_j^c, \quad |\langle f, \psi_{j,n}\rangle| \leqslant C 2^{j(\alpha+1/2)}.$$

Using (17) for $d = 1$, the decay of singular coefficients is bounded as

$$\forall\, n \in \mathcal{C}_j, \quad |\langle f, \psi_{j,n}\rangle| \leqslant C 2^{j/2}.$$

Once a fixed threshold T is fixed to compute the non-linear approximation, one defines cut-off scales for regular and singular coefficients that depend on T

$$2^{j_1} = (T/C)^{\frac{1}{\alpha+1/2}} \quad \text{and} \quad 2^{j_2} = (T/C)^2.$$

Figure 6.15 shows a schematic segmentation of the set of wavelet coefficients into regular and singular parts, and also using the cut-off scales.

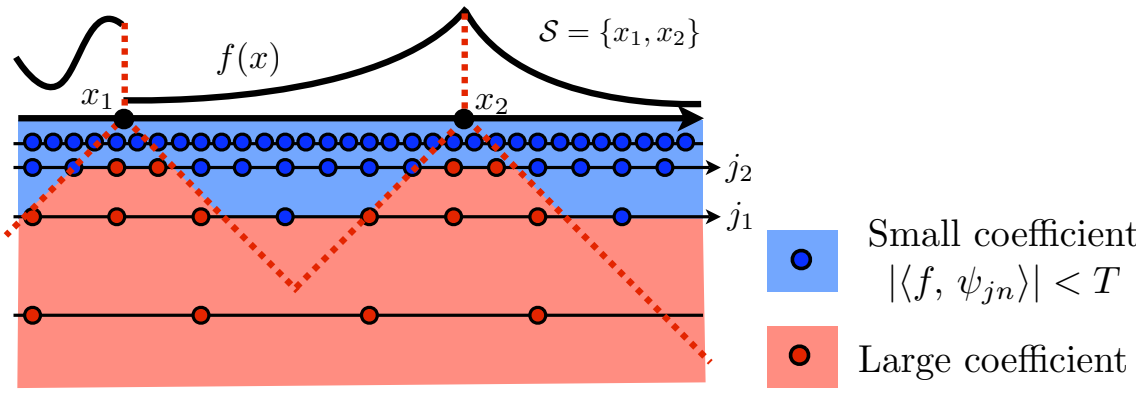


Figure 6.15: Segmentation of the wavelet coefficients into regular and singular parts.

Step 2. Counting the error. These cut-off scales allow us to define a hand-crafted approximation signal

$$\tilde{f}_M = \sum_{j \geqslant j_2} \sum_{n \in \mathcal{C}_j} \langle f, \psi_{j,n}\rangle \psi_{j,n} + \sum_{j \geqslant j_1} \sum_{n \in \mathcal{C}_j^c} \langle f, \psi_{j,n}\rangle \psi_{j,n}. \tag{6.24}$$

The approximation error generated by this M-term approximation $\tilde{f}_M$ is larger than the best M-term approximation error, and hence

$$\|f - f_M\|^2 \leqslant \|f - \tilde{f}_M\|^2 \leqslant \sum_{j<j_2, n \in \mathcal{C}_j} |\langle f, \psi_{j,n}\rangle|^2 + \sum_{j<j_1, n \in \mathcal{C}_j^c} |\langle f, \psi_{j,n}\rangle|^2 \tag{6.25}$$

$$\leqslant \sum_{j<j_2} (K|\mathcal{S}|) \times C^2 2^j + \sum_{j<j_1} 2^{-j} \times C^2 2^{j(2\alpha+1)} \tag{6.26}$$

$$= O(2^{j_2} + 2^{2\alpha j_1}) = O(T^2 + T^{\frac{2\alpha}{\alpha+1/2}}) = O(T^{\frac{2\alpha}{\alpha+1/2}}). \tag{6.27}$$

Step 3. Counting the number of measurements. The number of coefficients needed to build the approximating signal $\tilde{f}_M$ is

$$M \leqslant \sum_{j \geqslant j_2} |\mathcal{C}_j| + \sum_{j \geqslant j_1} |\mathcal{C}_j^c| \leqslant \sum_{j \geqslant j_2} K|\mathcal{S}| + \sum_{j \geqslant j_1} 2^{-j} \tag{6.28}$$

$$= O(|\log(T)| + T^{\frac{-1}{\alpha+1/2}}) = O(T^{\frac{-1}{\alpha+1/2}}). \tag{6.29}$$

Step 3. Putting everything together. Putting equations (6.25) and (6.28) together gives the desired result. □

This theorem improves significantly over the $O(M^{-1})$ decay of Fourier approximation (6.19). Furthermore, this decay is the same as the error decay of uniformly smooth signal (6.16). In 1-D, wavelet approximations do not "see" the singularities. The error decay (6.22) can be shown to be asymptotically optimal.

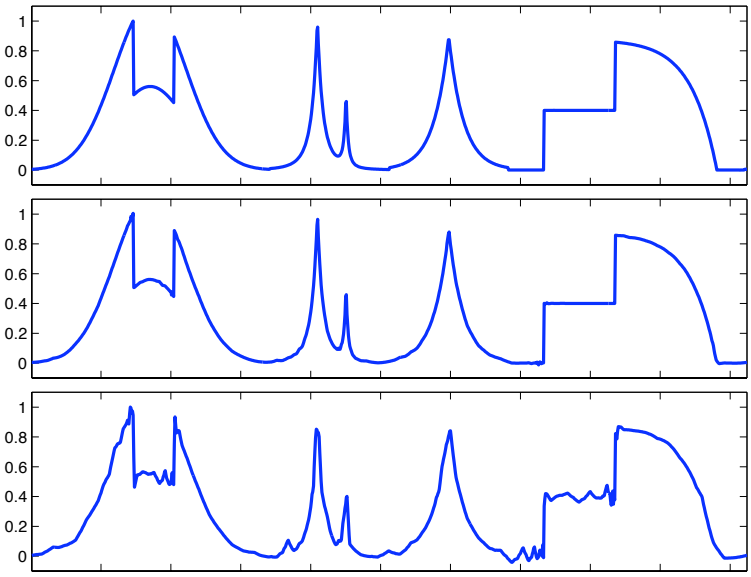


Figure 6.16: 1-D wavelet approximation.

Figure 6.16 shows examples of wavelet approximation of singular signals.

6.5.3 2-D Piecewise Smooth Approximation

We now give the equivalent of Theorem 26 but for 2-D functions.

Theorem 27. *f is in the piecewise smooth signal model (6.10), the non-linear approximation error in wavelet obeys*

$$\|f - f_M\|^2 = O(M^{-1}). \tag{6.30}$$

Proof. For an image in the piecewise smooth model (6.10), we define the singular support $\mathcal{C}_j$ as in (6.23). The major difference with the 1-D setting, is that for 2-D images, the size of the singular support grows when the scale 2^j goes to zero

$$|\mathcal{C}_j^\omega| \leqslant 2^{-j} K|\mathcal{S}|,$$

where $|\mathcal{S}|$ is the perimeter of the singular curves $\mathcal{S}$, and $\omega \in \{V, H, D\}$ is the wavelet orientation. Using (6.21) for $d = 2$, the decay of regular coefficients is bounded as

$$\forall\, n \in (\mathcal{C}_j^\omega)^c, \quad \left|\langle f,\, \psi_{j,n}^\omega \rangle\right| \leqslant C 2^{j(\alpha+1)}.$$

Using (17) for $d = 2$, the decay of singular coefficients is bounded as

$$\forall\, n \in \mathcal{C}_j^\omega, \quad \left|\langle f,\, \psi_{j,n}^\omega \rangle\right| \leqslant C 2^{j}.$$

After fixing T, the cut-off scales are defined as

$$2^{j_1} = (T/C)^{\frac{1}{\alpha+1}} \quad \text{and} \quad 2^{j_2} = T/C.$$

We define similarly to (6.24) a hand-made approximation. Similarly to (6.25), we bound the approximation error as

$$\|f - f_M\|^2 \leqslant \|f - \tilde{f}_M\|^2 = O(2^{j_2} + 2^{2\alpha j_1}) = O(T + T^{\frac{2\alpha}{\alpha+1}}) = O(T)$$

and the number of coefficients as

$$M = O(T^{-1} + T^{\frac{-1}{\alpha+1}}) = O(T^{-1}).$$

This leads to the announced decay of the non-linear wavelet approximation error. □

This improves significantly over the $O(M^{-1/2})$ decay of Fourier approximation (6.20). This result is however deceiving, since it does not take advantage of the C^α regularity of the image outside the edge curves.

This error decay is still valid for the more general model of images with bounded variations (6.11). One can shows that wavelets are asymptotically optimal to approximate images with bounded variations.

Figure 6.17: 2-D wavelet approximation.

Figure 6.17 shows wavelet approximations of a bounded variation image.

6.6 Cartoon Images Approximation

The square support of wavelet makes them inefficient to approximate geometric images (6.13), whose edges are more regular than the level set of bounded variation images (6.11), which are only assumed to be of finite length.

6.6.1 Wavelet Approximation of Cartoon Images

Result (6.30) shows that wavelet approximation of images in the cartoon models (6.13) decays at least like $O(M^{-1})$. One can show that simple cartoon images like $f = 1_\Omega$ where Ω is a disk reach this low decay speed. This is because the square support of wavelets forbid them to take advantage of the regularity of edge curves. The approximation error for the smoothed cartoon model (6.14) is also slow if the width of the blurring kernel is small with respect to the number M of coefficients.

Figure 6.18 shows that many large coefficients are located near edge curves, and retaining only a small number leads to a bad approximation with visually unpleasant artifacts.

Figure 6.18: Wavelet approximation of a cartoon image.

6.6.2 Finite Element Approximation

To improve over the wavelet approximation, one can design an adapted triangulation that is highly anisotropic near edges. Figure 6.19 shows an example of such a triangulation.

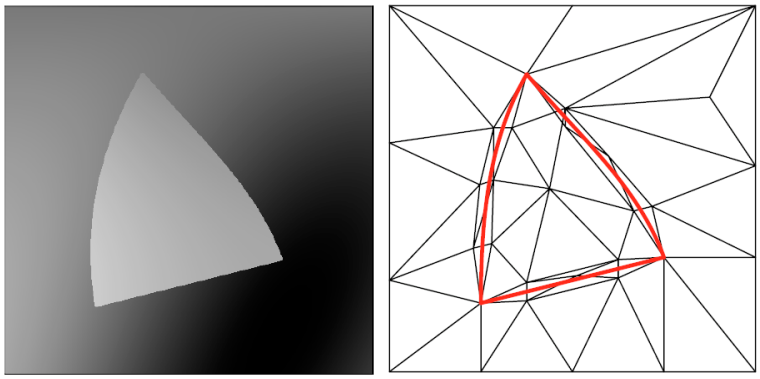


Figure 6.19: Left: cartoon image, right: adaptive triangulation.

A triangulation is obtain by sampling M points over the image domain $[0,1]^2$ and then connecting them using triangles. One then defines a piecewise linear interpolation $\tilde{f}_M$ over these triangles.

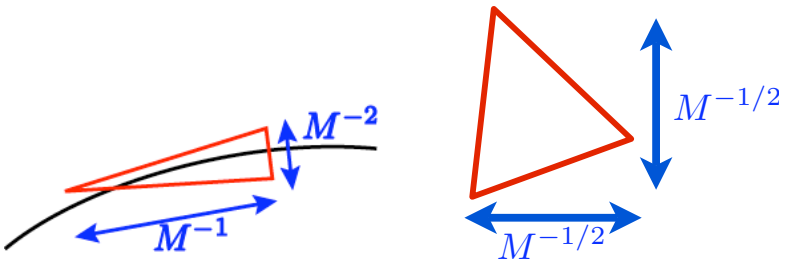


Figure 6.20: Aspect ratio of triangle away from edges (left) and near an edge (right).

As shown in Figure 6.20, an efficient approximation of a C^2-cartoon image (6.13) for $\alpha = 2$ is obtained by seeding $\approx M/2$ approximately equilateral triangles of width $\approx M^{-1/2}$ in the areas where the image is regular. Near the edges, using the C^2 regularity of the singular curve, one can seed $\approx M/2$ anisotropic triangles of length M^{-1} and width $\approx M^{-1/2}$. One can show that such an adaptive triangulation leads to an approximation error

$$\|f - f_M\|^2 = O(M^{-2}), \tag{6.31}$$

which improves over the wavelet approximation error decay (6.30).

This scheme is however difficult to implement in practice, since the edge curves are not known and difficult to find. This is in particular the case for smooth cartoon images when the smoothing kernel h is unknown.

There is currrently no known algorithm that can automatically produces the error decay (6.31). One thus has to use heuristics and greedy algorithm to find the location of the sampling points and computes the triangles. Figure (6.21) shows an example of compression using such a greedy seeding algorithm, that works well in practice.

Adpated triangulation | $\tilde{f}_M$ | JPEG-2000

Figure 6.21: Comparison of adaptive triangulation and JPEG-2000, with the same number of bits.

6.6.3 Curvelets Approximation

Instead of using an adaptive approximation scheme such as finite elements, one can replace the wavelet basis by a set of oriented anisotropic atoms. The curvelet frame was proposed by Candès and Donoho for this purpose [6].

Curvelets. The curvelet construction starts from a curvelet function c that is oriented along the horizontal direction, and perform stretching

$$c_{2^j}(x_1, x_2) \approx 2^{-3j/4} c(2^{-j/2} x_1, 2^{-j} x_2),$$

translation and rotation

$$c^{\theta}_{2^j,u}(x_1, x_2) = c_{2^j}(R_\theta(x - u))$$

where R_θ is the rotation of angle θ.

The atoms $c^{\theta}_{2^j,u}$ is located near u, with an orientation θ, and has an aspect ratio "width $\approx$ length^2", which is the same aspect used to build an adaptive finite element approximation. This aspect ratio is essential to capture the anisotropic regularity near edges for images in the cartoon model (6.13) for $\alpha = 2$.

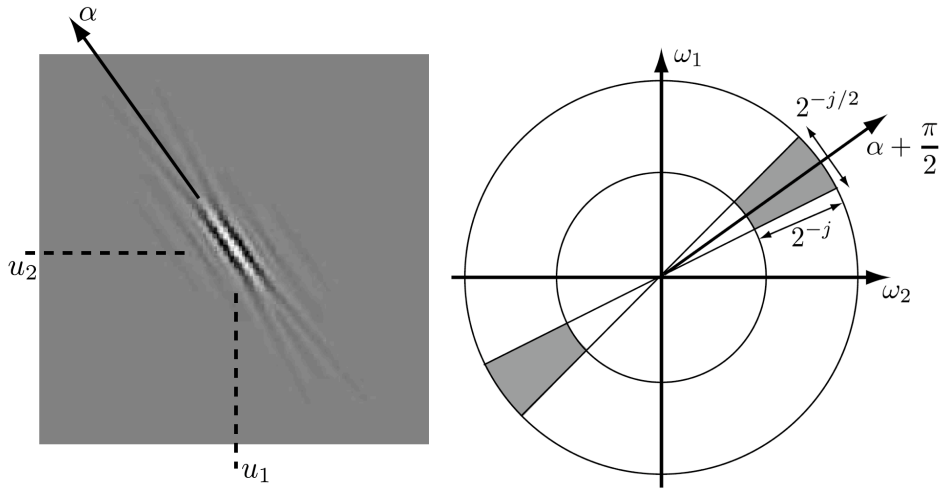


Figure 6.22: Left: a curvelet c_m, right: its Fourier transform localization.

Figure 6.23 shows the spacial and frequency localization of curvelets.

Parameter discretization. To build an image representation, one need to sample the u and θ parameter. To maintain a stable representation, the sub-sampling of the angles depends on the scale

$$\forall\, 0 \leqslant k < 2^{-\lceil j/2 \rceil + 2}, \quad \theta^{(j)}_k = k\pi 2^{\lceil j/2 \rceil - 1}$$

and the spacial grid depends on the scale and on the angles

$$\forall\, m = (m_1, m_2) \in \mathbb{Z}^2, \quad u^{(j,\theta)}_m = R_\theta(2^{j/2} m_1, 2^j m_2).$$

Figure 6.23 shows this sampling pattern.

Curvelet tight frame. This sampling leads to a stable redundant family

$$C_{j,m,k}(x) = c^{\theta}_{2^j,u}(x) \quad \text{where} \quad \theta = \theta^{(j)}_k \quad \text{and} \quad u = u^{(j,\theta)}_m,$$

that obeys a conservation of energy

$$\|f\|^2 = \sum_{j \in \mathbb{Z}} \sum_{k=0}^{2^{-\lceil j/2 \rceil + 2}} \sum_{m \in \mathbb{Z}^2} |\langle f,\, C_{j,m,k} \rangle|^2$$

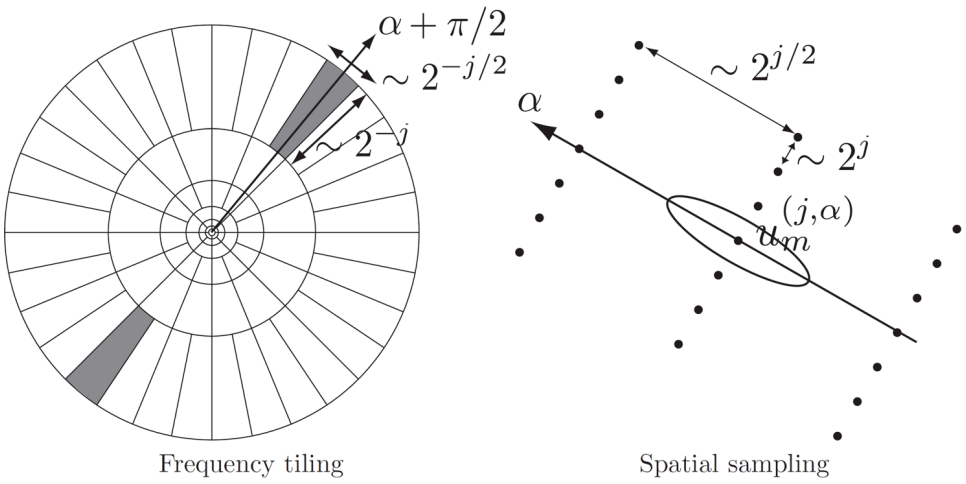


Figure 6.23: Sampling pattern for the curvelet positions.

and a reconstruction formula

$$f = \sum_{j \in \mathbb{Z}} \sum_{k=0}^{2^{-\lceil j/2 \rceil + 2}} \sum_{m \in \mathbb{Z}^2} \langle f, C_{j,m,k} \rangle C_{j,m,k}$$

that extends the properties of orthogonal basis (tight frame), although the representation is redundant (the atoms are not orthogonal).

A numerical implementation of this tight frame also defines a discrete tight frame for image of N pixels, that is made of $\approx 5N$ atoms [8].

Curvelet approximation. A non-linear M-term approximation in curvelets is defined as

$$f_M = \sum_{|\langle f, C_{j,m,k} \rangle| > T} \langle f, C_{j,m,k} \rangle C_{j,m,k}$$

where T is a threshold that depends on M. One should note that f_M is not necessarily the best M-term curvelet approximation since the curvelet frame is not orthogonal.

For position $u_m^{(j,\theta)}$ that are far away from an edges, the vanishing moments of the curvelets create a small coefficient $\langle f, C_{j,m,k} \rangle$. If $u_m^{(j,\theta)}$ is close to an edge curve whose tangent has direction $\tilde{\theta}$, then the coefficient $\langle f, C_{j,m,k} \rangle$ decays very fast to zero when $|\theta - \tilde{\theta}|$ increases. Figure 6.24 shows the principle of this curvelet approximation, and compares it with directional wavelets that have a square support.

Using these two properties together with the sparse sampling of the curvelet in space and orientation leads to the following approximation error decay

$$\|f - f_M\|^2 = O(\log^3(M) M^{-2})$$

for image in the cartoon model (6.13) for $\alpha = 2$. This is close to the decay of adaptive triangulations (6.31), but this time one computes f_M with a fast $O(N \log(N))$ algorithm for an image of N pixels.

In practice, the redundancy of the curvelet frame makes it not suitable for image compression. Its efficiency is however useful for denoising purpose, where it can improve over wavelet to denoise geometric images and textures, see Figure 6.25. The result is obtained by using a thresholding denoiser as detailed in Section 8.3.1.

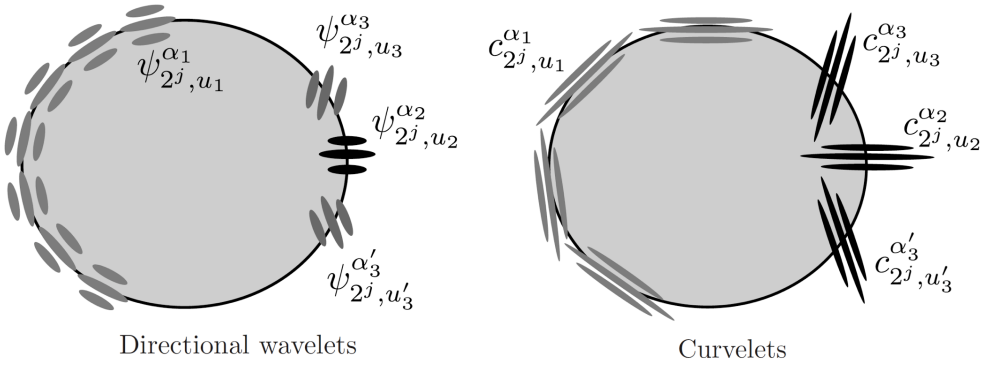


Figure 6.24: Comparison of the principle of wavelets (left) and curvelet (right) approximations of a cartoon image.

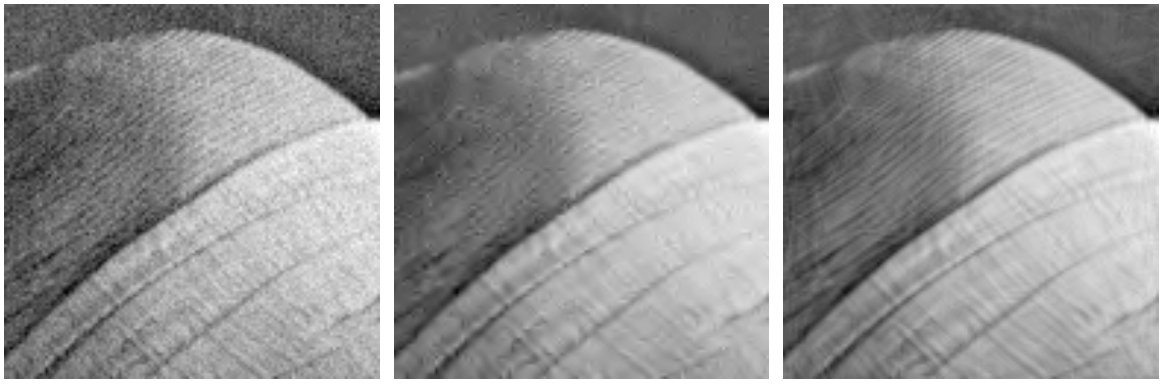


Figure 6.25: Comparison of wavelets (translation invariant) and curvelet denoising.

Chapter 7

Compression

7.1 Transform Coding

7.1.1 Coding

State of the art compression schemes correspond to transform coders, that code quantized coefficients in an ortho-basis. They first computes the coefficients of the decomposition of the signal into an well-chosen basis (for instance wavelets)

$$a_m = \langle f, \psi_m \rangle \in \mathbb{R}.$$

Quantization corresponds to rounding the coefficients to an integer using a step size $T > 0$

$$q_m = Q_T(a_m) \in \mathbb{Z} \quad \text{where} \quad Q_T(x) = \text{sign}(x) \left\lfloor \frac{|x|}{T} \right\rfloor.$$

We note that this quantizer has a twice larger zero bin, so that coefficients in $[-T, T]$ are set to zero.

This quantizer nonlinearity should be compared to the hard thresholding nonlinearity (6.4) used to perform non-linear approximation. The quantizer not only set to zero small coefficients that are smaller than T in magnitude, it also modifies larger coefficients by rounding, see Figure 7.1.

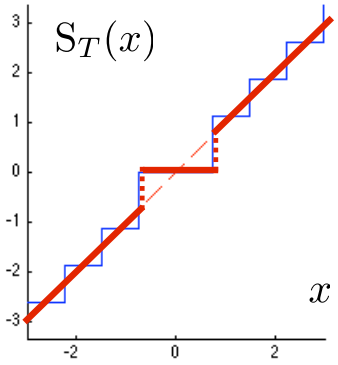


Figure 7.1: Thresholding and quantization non-linearity mappings.

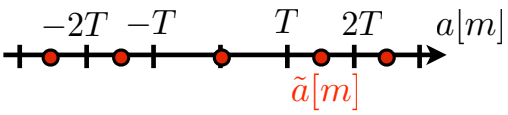
The resulting integer values $(q_m)_m$ are stored into a binary file of length R, which corresponds to a number of bits. Sections 7.1.3 and 7.2 detail two different approach to perform this transformation from integer to bits. The goal is to reduce as much as possible the number R of bits.

7.1.2 De-coding

The decoder retrieves the quantized coefficients q_m from the binary file, and dequantizes the coefficients using

$$\tilde{a}_m = \mathrm{sign}(q_m)\left(|q_m| + \frac{1}{2}\right)T. \tag{7.1}$$

This corresponds to retrieving the value from quantization at the center of quantization bins:



The compressed-decompressed image is then reconstructed as

$$\mathcal{Q}_T(f) \stackrel{\text{def.}}{=} \sum_{m\in I_T} \tilde{a}_m \psi_m = \sum_{m\in I_T} Q_T(\langle f, \psi_m\rangle)\psi_m,$$

thus producing a decompression error $\|f - \mathcal{Q}_T(f)\|$.

This decompression reconstruction (7.1.2) should be compared with the non-linear approximation formula (6.3). One necessarily has $\|f - f_M\| \leqslant \|f - \mathcal{Q}_T(f)\|$, but in practice these two errors have comparable magnitudes.

Proposition 18. *One has*

$$\|f - \mathcal{Q}_T(f)\|^2 \leqslant \|f - f_M\|^2 + MT^2/4 \quad \textit{where} \quad M = \#\left\{m\,;\, \tilde{a}_m \neq 0\right\}. \tag{7.2}$$

Proof. Indeed, the de-quantization formula (7.1) implies that for $|a_m| > T$,

$$|a_m - \tilde{a}_m| \leqslant \frac{T}{2}.$$

One thus has

$$\|f - \mathcal{Q}_T(f)\|^2 = \sum_m (a_m - \tilde{a}_m)^2 \leqslant \sum_{|a_m|<T} |a_m|^2 + \sum_{|a_m|\geqslant T} \left(\frac{T}{2}\right)^2,$$

which leads to the desired bound. □

If $\|f - f_M\|^2 \sim O(M^{-\alpha})$, then it is easy to see that $\|f - f_M\|^2 \sim MT^2$ so that one has

$$\|f - \mathcal{Q}_T(f)\|^2 \sim \|f - f_M\|^2.$$

7.1.3 Support Coding

To measure how large is the additional error term $MT^2/4$ in (7.2), one needs to choose a method to store the quantized coefficients q_m into a file.

For aggressive compression scenario, where R and M are small with respect to the size N of the image, the support

$$I_M = \{m\,;\, \tilde{a}_m \neq 0\}$$

is highly sparse. It thus make sense to code first this support and then to code the actual value $q_m \neq 0$ for $m \in I_M$.

The remaining of this section proves the following theorem.

Theorem 28. *We assume $\|f - f_M\|^2 \sim O(M^{-\alpha})$ where f_M is the M-term non-linear approximation of f in $\{\psi_m\}_m$. Then for all T is a coding strategy of $\mathcal{Q}_T(f)$ using $R = R(T)$ bits such that*

$$\|f - \mathcal{Q}_T(f)\|^2 = O(\log^\alpha(R)R^{-\alpha}).$$

Comments on the signals constraints. First, let us notice that, thanks to Proposition 14 the error decay hypothesis $\|f - f_M\|^2 \sim O(M^{-\alpha})$ is equivalent to imposing a fast decay of the ordered coefficients d_m defined in (6.1)

$$d_m \sim m^{-\frac{\alpha+1}{2}}. \tag{7.3}$$

Furthermore, a practical compression algorithm is only capable of dealing with discrete signals of size N. We thus considers that the algorithm has access to N inner products $\{\langle f, \psi_m\rangle\}_{0\leqslant m<N}$ that are computed using a decomposition algorithm from a discretized signal or image of size N. For instance, Section **??** details a discrete wavelet transform, and introduces a compatibility condition (**??**) on the sampling operator for this inner product computation to be possible from discrete data.

For the compression from the discrete signals to be the same as a compression of a continuous signal, we impose that N is large enough so that

$$\forall\, m \geqslant N, \quad |\langle f, \psi_m\rangle| < T$$

so that the coefficients not quantized to 0 are contained within the set $\{\langle f, \psi_m\rangle\}_{0\leqslant m<N}$ of the N computed coefficients.

The other hypothesis beyond (7.3) is that the sampling precision N is not too large, and in particular, that there is a polynomial grows of N with respect to the number M of coefficients to code

$$N \sim M^{\beta} \tag{7.4}$$

for some $\beta > 0$. For the wavelet and Fourier bases, and for all the classes of signals and images detailed in Section 6.2, one can show that this is indeed the case, if one orders the basis elements $\{\psi_m\}_m$ from low frequencies (or coarse scales) to high frequencies (or fine scales). This is because in these bases, the coefficients $\langle f, \psi_m\rangle$ have a polynomial decay with m if f as some smoothness. See Sections **??** and 6.5.1 for a proof of this decay for Sobolev and piecewise regular signals.

Support coding. Since the size of the support is $|I_M| = M$, one can code the entries of I_M as being a set of size M within a larger set of N coefficients using a number of bits

$$R_{\text{ind}} \leqslant \log_2 \binom{N}{M} = O(M \log_2 \left(\frac{N}{M}\right)) = O(M \log_2(M)) \tag{7.5}$$

where $\binom{N}{M}$ is the number of possible choices for M elements in a set of N elements, and where we have used hypothesis (7.3) to derive the last equality.

Values coding. The quantized values satisfy $q_m \in \{-A, \ldots, A\}$, with

$$A \leqslant \frac{1}{T} \max_m |\langle f, \psi_m\rangle| = O(T^{-1}),$$

so one can code them using a number of bits

$$R_{\text{val}} = O(M|\log_2(T)|) = O(M \log_2(M)) \tag{7.6}$$

where we have used hypothesis (7.3) that implies $|\log_2(T)| \sim \log_2(M)$.

Total number of bits. Putting (7.5) and (7.6) together, the total number of bits for this support coding approach is thus

$$R = R_{\text{ind}} + R_{\text{val}} = O(M \log_2(M)).$$

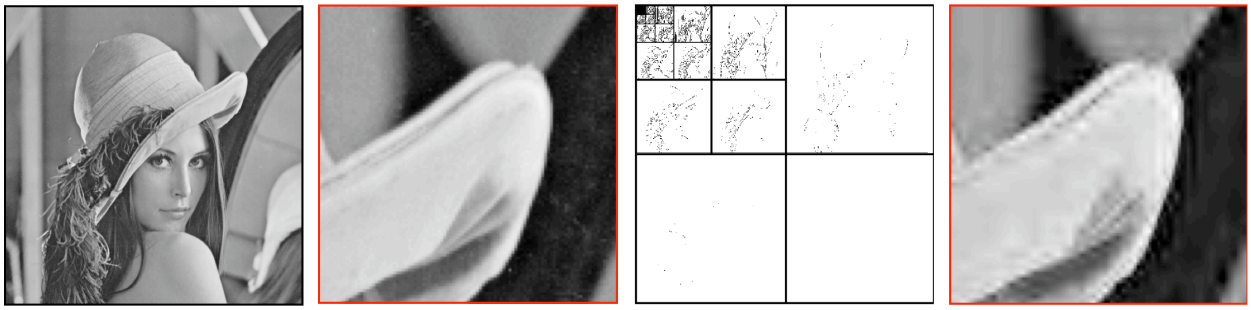
Inverting this relationship proves that

$$M = O(R \log_2(R)). \tag{7.7}$$

Rate/distortion error decay. Condition (7.3) implies that $MT^2 \sim M^{-\alpha}$, so that putting (7.2) and (7.7) together proves

$$\|f - \mathcal{Q}_T(f)\|^2 = O(\log^\alpha(R) R^{-\alpha}).$$

This shows the importance of the study of non-linear approximation, and in particular the design of bases that are efficient for approximation of a given signal model Θ.



Original f | Zoom | Wavelet support | Compressed $\mathcal{Q}_T(f)$

Figure 7.2: Image compression using wavelet support codding.

7.2 Entropic Coding

To further reduce the file size R (in bits), one can use an entropic coder to transform the integer values q_m into bits. Such coding scheme makes use of the statistical redundancy of the quantized values, that have a large number of zero entries and very few large entries. The theory of coding was formalized by Shannon [40].

We refer to Section 1.3 for the theoretical foundation associated to what we now describe.

Probabilistic modeling. The quantized coefficients $q_m \in \{-A, \ldots, A\}$ are assumed to take values in an alphabet of $Q = 2A + 1$ elements. A coding scheme performs the transformation

$$\{q_m\}_m \longmapsto \{0, 1, 1, \ldots, 0, 1\} \in \{0, 1\}^R.$$

To reduce the average value of R, one makes use of a statistical model, which assumes that the q_m are drawn independently at random from a known probability distribution

$$\mathbb{P}(q_m = i) = p_i \in [0, 1].$$

Huffman code. A Huffman code is a code with variable length, since it perform a mapping from symbols to binary strings

$$q_m = i \in \{-A, \ldots, A\} \longmapsto c_i \in \{0, 1\}^{|c_i|}$$

where $|c_i|$ is the length of the binary code word c_i, that should be larger if p_i is small. A Huffman tree algorithm is able to build a code such that

$$|c_i| \leqslant \lceil \log_2(p_i) \rceil$$

so that

$$R \leqslant (\mathcal{E}(p) + 1)N$$

where $\mathcal{E}$ is the entropy of the distribution, defined as

$$\mathcal{E}(p) = -\sum_i p_i \log_2(p_i).$$

Figure 7.3 shows different probability distribution. The entropy is small for highly sparse distribution. Wavelet coefficients of natural images tend to have a low entropy because many coefficients are small.

The Huffman scheme codes symbols independently, leading to a sub-optimal code if some of the p_i are large, which is usually the case for wavelet coefficients. One usually prefers arithmetic coding schemes, that codes groups of symbols, and are able to get close to the entropy bound $R \approx \mathcal{E}(p)N$ for large N.

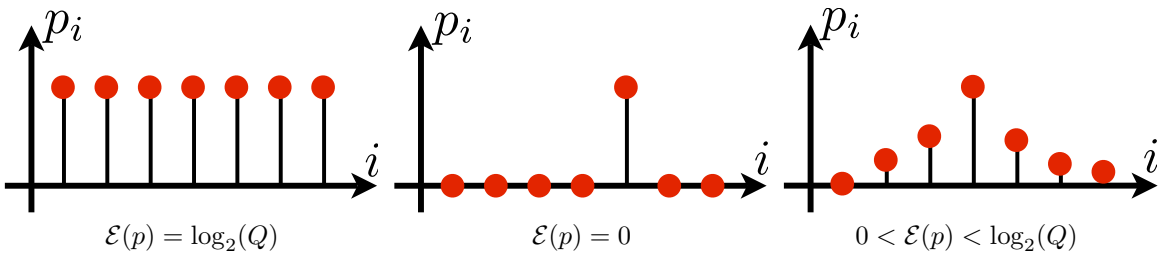


Figure 7.3: Three different probability distributions.

7.3 JPEG-2000

JPEG-2000 is the latest still image compression standard. It corresponds to a wavelet transform coder that performs a clever adaptive entropy coding that makes use of the statistical redundancy of wavelet coefficients of natural images. The wavelet transform is not orthogonal, it is a symmetric 7/9 biorthogonal, with symmetric boundary condition and a lifting implementation. This transform is however close to orthogonality, so that the previous discussion about orthogonal approximation and coding is still relevant.

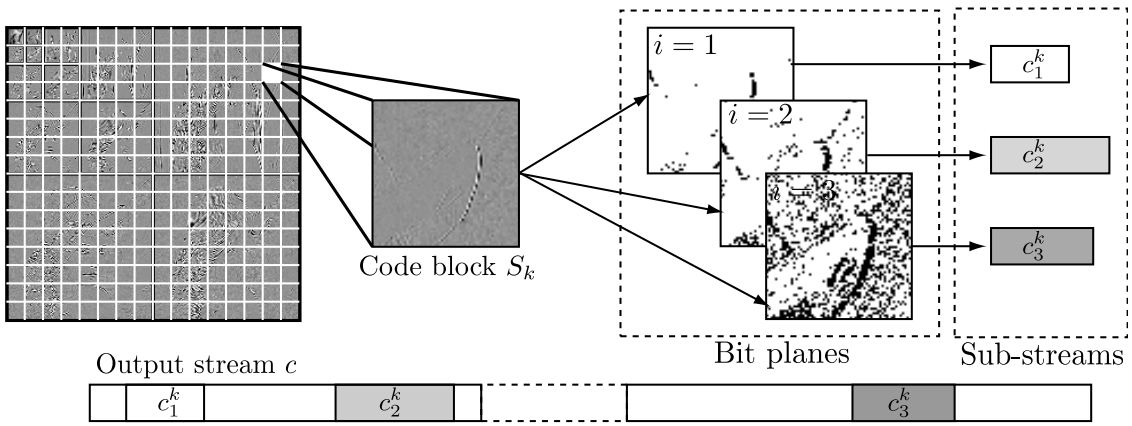


Figure 7.4: JPEG-2000 coding architecture.

Figure 7.4 shows an overview of JPEG-2000 architecture. Figure 7.5 shows a comparison between JPEG and JPEG-2000 compressors. JPEG is based on a local DCT transform, and suffers from blocking artifacts at low bit rates, which is not the case of JPEG-2000. This new standard also comes with several important features, such as regions of interest, which allows to refine the coding in some specific parts of the image.

Figure 7.5: Comparison of JPEG (left) and JPEG-2000 (right) coding.

Dyadic quantization. The wavelet coefficients are quantized with a varying quantization step $T_i = 2^{-i}T_0$. This allows one to progressively increase the precision of the coded coefficients. For each i, a bit plane coding pass produces new bits to refine the value of the coefficients when i increases.

Steam packing. The bits obtained using the bit plane pass with quantizer $T_i = 2^{-i}T_0$ are entropy coded using a contextual coder. This coder processes square blocks S_k of coefficients. This local coding enhances the parallelization of the method. This local block coding produces a bit stream c_i^k, and these streams are optimally packed into the final coded file to reduce the distortion $\|f - \mathcal{Q}_T(f)\|$ for almost every possible number R of bits. This stream packing ensures scalability of the output bit stream. It means that one can receive only the R first bits of a large coded file and get a low resolution decoded image $\mathcal{Q}_T(f)$ that has an almost minimal distortion $\|f - \mathcal{Q}_T(f)\|$.

Bit plane coding pass. For each threshold T_i, for each scale and orientation $(j, \omega \in \{V, H, D\})$, for each coefficient location $n \in S_k$, JPEG-2000 coder encodes several bit reflecting the value of the wavelet coefficient $d_j^\omega[n]$. In the following we drop the dependancy on (j, ω) for simplicity.

- If $d_j^\omega[n] < T_{i-1}$, the coefficient was not significant at bit-plane $i-1$. It thus encodes a significance bit $b_i^1[n]$ to tell wether $d_j^\omega[n] \geqslant T_i$ or not.
- If $b_i^1[n] = 1$, meaning that the coefficient has became significant, it codes it sign as a bit $b_i^2[n]$.

– For every position n that was previously significant, meaning $d_j^\omega[n] \geqslant T_{i-1}$, it codes a value refinement bit $b_i^3[n]$ to tell wether $d_j^\omega[n] \geqslant T_i$ or not.

Contextual coder. The final bits streams c_i^k are computed from the produced bits $\{b_i^s[n]\}_{s=1}^3$ for $n \in S_k$ using a contextual coder. The contextual coding makes use of spacial redundancies in wavelet coefficients, especially near edges and geometric singularities that create clusters of large coefficients. The coefficients $n \in S_k$ are traversed in zig-zag order as shown on Figure 7.6.

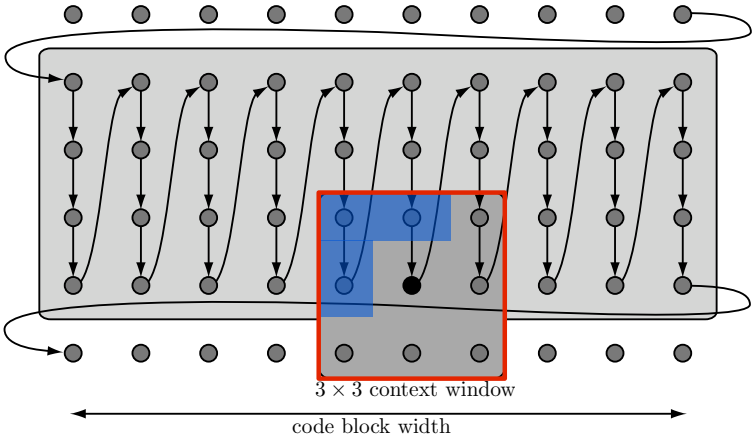


Figure 7.6: Coding order and context for JPEG-2000 coding.

For each coefficient location $n \in S_k$, the context value $v_i^s[n]$ of the bit $b_i^s[n]$ to code at position x is an integer computed over a 3×3 window

$$w_n = \{(n_1 + \varepsilon_1, n_2 + \varepsilon_2)\}_{\varepsilon_i = \pm 1}.$$

This local context $v_i^s[n]$ integrates in a complicated way the previous bit plane values $\{b_{i-1}^s[\tilde{n}]\}_{\tilde{n} \in w_n}$, and neighboring bits at plane $\{b_i^s[\tilde{n}]\}_{\tilde{n} \in w_n, \tilde{n} \text{ coded}}$ that have already been coded.

The bit value $b_i^s[n]$ is then coded with an arithmetic coding by making use of the conditional probability distribution $\mathbb{P}(b_i^s[n]|v_i^s[n])$. The choice made for the computation $v_i^s[n]$ allows to reduce significantly the entropy of this conditional probability condition with respect to the original distribution $\mathbb{P}(b_i^s[n])$, thus reducing the overall number of bits.

Chapter 8

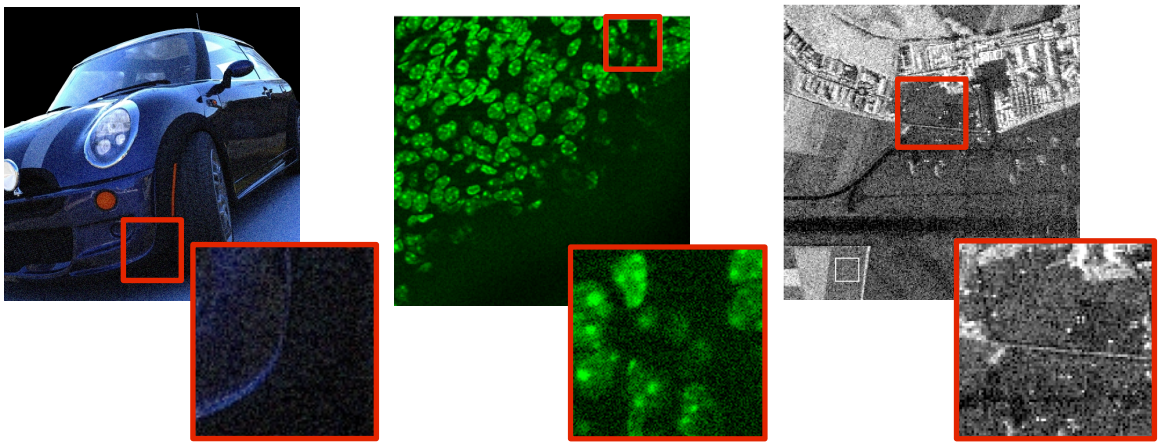
Denoising

Together with compression, denoising is the most important processing application, that is pervasive in almost any signal or image processing pipeline. Indeed, data acquisition always comes with some kind of noise, so modeling this noise and removing it efficiently is crucial.

8.1 Noise Modeling

8.1.1 Noise in Images

Image acquisition devices always produce some noise. Figure 8.1 shows images produced by different hardware, where the regularity of the underlying signal and the statistics of the noise is very different.



Digital camera Confocal imaging SAR imaging

Figure 8.1: Example of noise in different imaging device.

One should thus model both the acquisition process and the statistics of the noise to fit the imaging process. Then one should also model the regularity and geometry of the clean signal to choose a basis adapted to its representation. This chapter describes how thresholding methods can be used to perform denoising in some specific situations where the noise statistics are close to being Gaussian and the mixing operator is a sum or can be approximated by a sum.

Since noise perturbs discrete measurements acquired by some hardware, in the following, we consider only finite dimensional signal $f \in \mathbb{C}^N$.

8.1.2 Image Formation

Figure 8.2 shows an idealized view of the image formation process, that mixes a clean image f_0 with a noise w to obtain noisy observations $f = f_0 \oplus w$, where $\oplus$ might for instance be a sum or a multiplication.

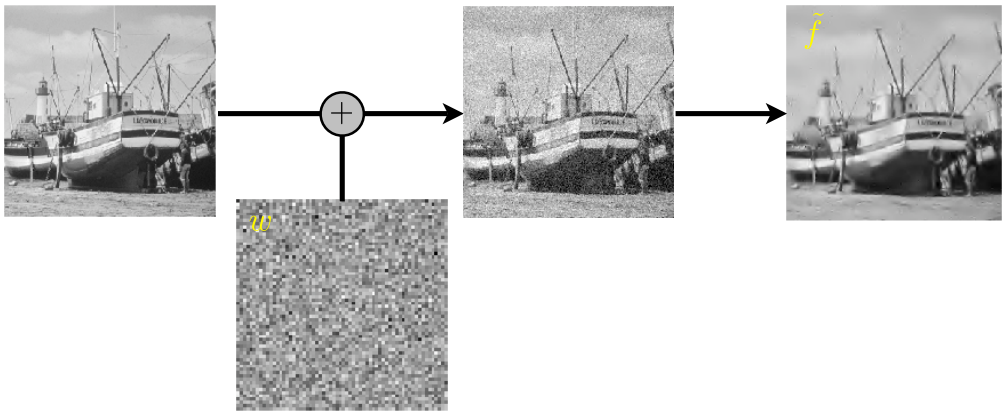


Figure 8.2: Image formation with noise modeling and denoising pipepline.

Statistical modeling considers w as a random vector with known distribution, while numerical computation are usually done on a single realization of this random vector, still denoted as w.

Additive Noise. The simplest model for such image formation consists in assuming that it is an additive perturbation of a clean signal f_0

$$f = f_0 + w$$

where w is the noise residual. Statistical noise modeling assume that w is a random vector, and in practice one only observes a realization of this vector. This modeling thus implies that the image f to be processed is also a random vector. Figure 8.3 and 8.4 show examples of noise addition to a clean signal and a clean image.

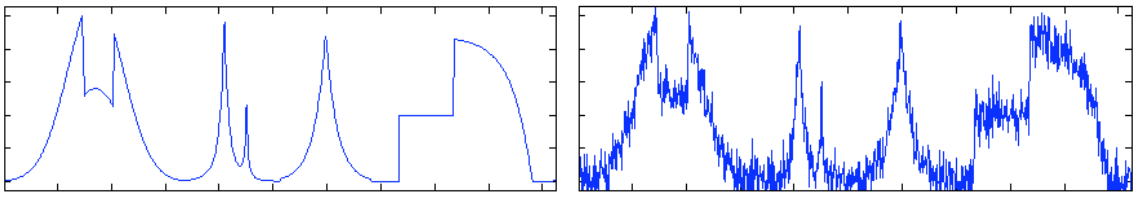


Figure 8.3: 1-D additive noise example.

The simplest noise model assumes that each entry w_n of the noise is a Gaussian random variable of variance σ^2, and that the w_n are independent, i.e. $w \sim \mathcal{N}(0, \mathrm{Id}_N)$. This is the white noise model.

Depending on the image acquisition device, one should consider different noise distributions, such as for instance uniform noise $w_n \in [-a, a]$ or Impulse noise

$$\mathbb{P}(w_n = x) \propto e^{-|x/\sigma|^\alpha} \quad \text{where} \quad \alpha < 2$$

In many situations, the noise perturbation is not additive, and for instance its intensity might depend on the intensity of the signal. This is the case with Poisson and multiplicative noises considered in Section 8.4.

f_0 f

Figure 8.4: 2-D additive noise example.

8.1.3 Denoiser

A denoiser (also called estimator) is an estimation $\tilde{f}$ of f_0 computed from the observation f alone. It is thus also a random vector that depends on the noise w. Since f is a random vector of mean f_0, the numerical denoising process corresponds to the estimation of the mean of a random vector from a single realization. Figure 8.5 shows an example of denoising.

The quality of a denoiser is measured using the average mean square risk $\mathbb{E}_w(\|f_0 - \tilde{f}\|^2)$. where $\mathbb{E}_w$ is the esperance (averaging) with respect to the noise w. Since f_0 is unknown, this corresponds to a theoretical measure of performance, that is bounded using a mathematical analysis. In the numerical experiments, one observes a single realization $f^r \sim f_0 + w$, and the performance is estimated from this single denoising using the SNR

$$\mathrm{SNR}(\tilde{f}^r, f0) = -20\log_{10}(\|\tilde{f}^r - f_0\|/\|f_0\|),$$

where $\tilde{f}^r$ should be computed from the single realization f^r of f. In the following, with an abuse of notation, when displaying single realization, we ignore the exponent ^{r}. The SNR is expressed in "decibels", denoted dB. This measure of performance requires the knowledge of the clean signal f_0, and should thus only be considered as an experimentation tool, that might not be available in a real life denoising scenario where clean data are not available. Furthermore, the use of an ℓ^2 measure of performance is questionable, and one should also observe the result to judge of the visual quality of the denoising.

8.2 Linear Denoising using Filtering

8.2.1 Translation Invariant Estimators

A linear estimator $\mathcal{E}(f) = \tilde{f}$ of f_0 depends linearly on f, so that $\mathcal{E}(f+g) = \mathcal{E}(f) + \mathcal{E}(g)$. A translation invariant estimator commutes with translation, so that $\mathcal{E}(f_\tau) = \mathcal{E}(f)_\tau$, where $f_\tau(t) = f(t-\tau)$. Such a denoiser can always be written as a filtering

$$\tilde{f} = f \star h$$

where $h \in \mathbb{R}^N$ is a (low pass) filter, that should satisfy at least

$$\sum_n h_n = \hat{h}_0 = 1$$

where $\hat{h}$ is the discrete Fourier transform.

Figure 8.6 shows an example of denoising using a low pass filter.

f_0 f $\tilde{f}$

Figure 8.5: Left: clean image, center: noisy image, right: denoised image.

The filtering strength is usually controlled the width s of h. A typical example is the Gaussian filter

$$\forall\ -N/2 < i \leqslant N/2, \quad h_{s,i} = \frac{1}{Z_s} \exp\left(-\frac{i^2}{2s^2}\right) \tag{8.1}$$

where Z_s ensures that $\sum_i h_{s,i} = 1$ (low pass). Figure 8.6 shows the effect of Gaussian filtering over the spacial and Fourier domains.

Figure 8.7 shows the effect of low pass filtering on a signal and an image with an increasing filter width s. Linear filtering introduces a blur and are thus only efficient to denoise smooth signals and image. For signals and images with discontinuities, this blur deteriorates the signal. Removing a large amount of noise necessitates to also smooth significantly edges and singularities.

8.2.2 Optimal Filter Selection

The selection of an optimal filter is a difficult task. Its choice depends both on the regularity of the (unknown) data f_0 and the noise level σ. A simpler option is to optimize the filter width s among a parametric family of filters, such as for instance the Gaussian filters defined in (8.1).

The denoising error can be decomposed as

$$\|\tilde{f} - f_0\| \leqslant \|h_s \star f_0 - f_0\| + \|h_s \star w\|$$

The filter width s should be optimized to perform a tradeoff between removing enough noise ($\|h_s \star w\|$ decreases with s) and not smoothing too much the singularities (($\|h_s \star f_0 - f_0\|$ increases with s).

Figure (8.8) shows the oracle SNR performance, defined in (**??**).

Figure 8.9 and 8.10 show the results of denoising using the optimal filter width $s^\star$ that minimizes the SNR for a given noisy observation.

These optimal filtering appear quite noisy, and the optimal SNR choice is usually quite conservative. Increasing the filter width introduces a strong blurring that deteriorates the SNR, although it might look visually more pleasant.

8.2.3 Wiener Filter

If one has a random model both for the noise $w \sim W$ and for for the signal $f_0 \sim F$, one can derives an optimal filters in average over both the noise and the signal realizations. One further assumes that w and

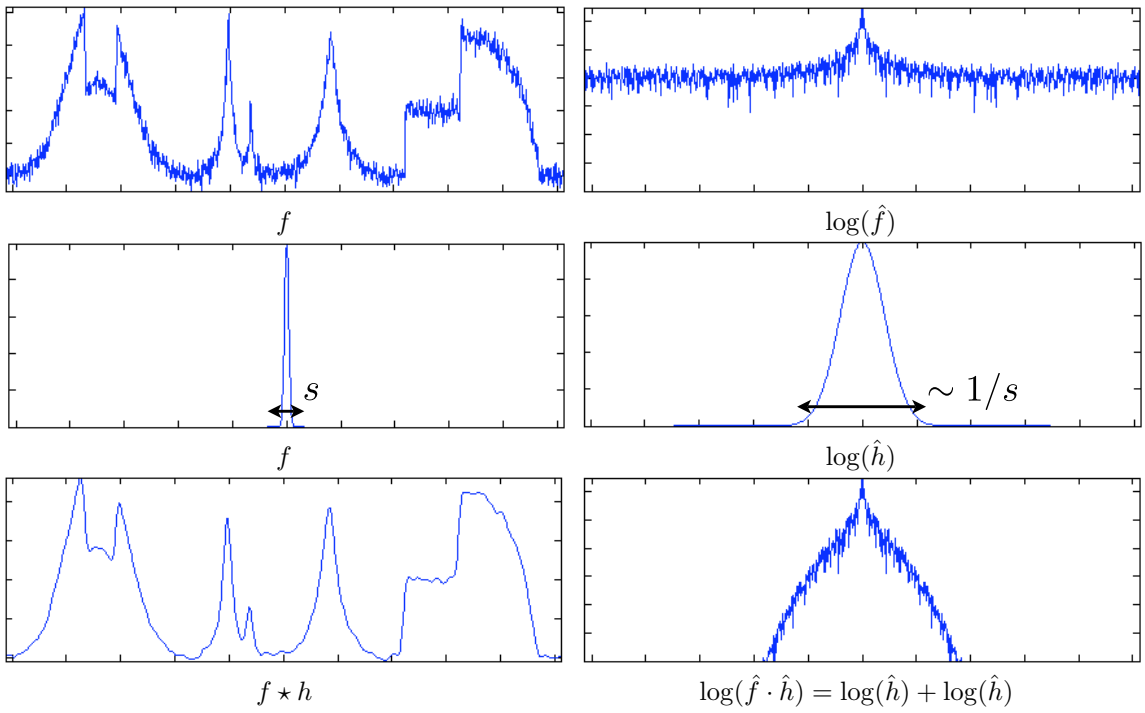


Figure 8.6: Denoising by filtering over the spacial (left) and Fourier (right) domains.

f_0 are independent realization. The optimal h thus minimizes

$$\mathbb{E}_{W,F}(\|h \star (F+W) - F\|^2)$$

If both F is wide-sense stationary, and W is a Gaussian white noise of variance σ^2, then the optimal filer is known as the Wiener filter

$$\hat{h}_\omega = \frac{|\hat{F}_\omega|^2}{|\hat{F}_\omega|^2 + \sigma^2}$$

where $|\hat{F}|^2$ is the power spectrum of F,

$$\hat{F}_\omega = \hat{C}_\omega \quad \text{where} \quad C_n = \mathbb{E}(\langle F, F[\cdot + n] \rangle),$$

the Fourier transform of an infinite vector is defined in Section 2.3.

In practice, one rarely has such a random model for the signal, and interesting signals are often not stationary. Most signals exhibit discontinuities, and are thus poorly restored with filtering.

8.2.4 Denoising and Linear Approximation

In order to study linear (and also non-linear, see the section bellow) denoising without assuming a random signal model, one should use approximation theory as studied in Chapter 6. We thus consider an ortho-basis $\mathcal{B} = (\psi_m)_m$ of $\mathbb{R}^N$, and consider a simple denoising obtained by keeping only the M first term elements of the approximation of the noisy observation in $\mathcal{B}$

$$\tilde{f} \stackrel{\text{def.}}{=} \sum_{m=1}^{M} \langle f, \psi_m \rangle \psi_m. \tag{8.2}$$

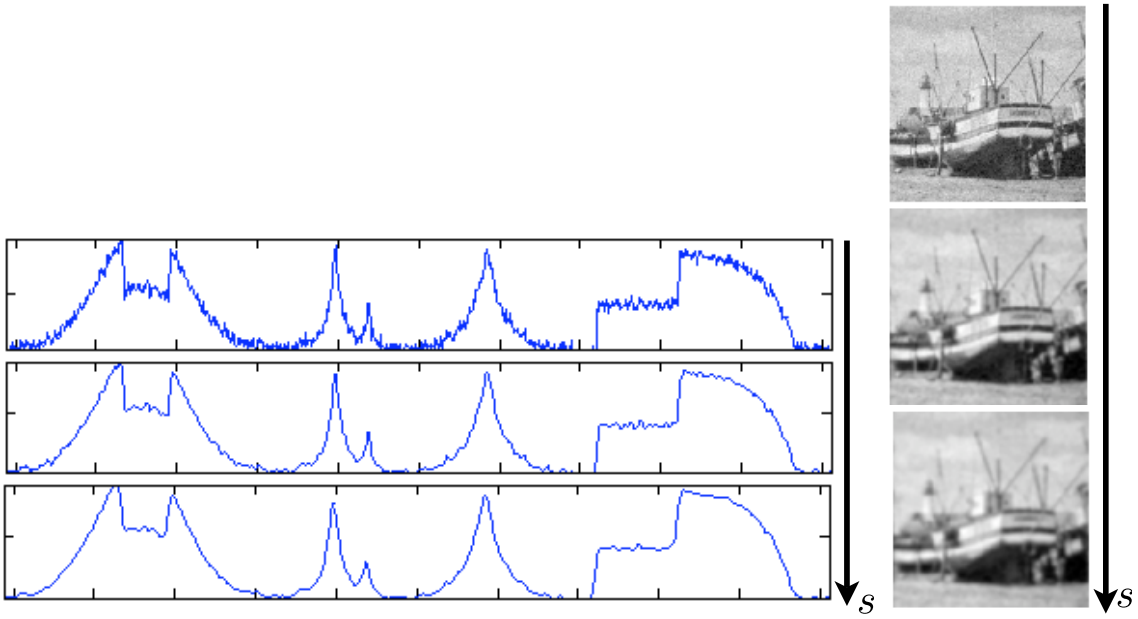


Figure 8.7: Denoising using a filter of increasing width s.

This is a linear projection on the space spanned by $(\psi_m)_{m=1}^M$. This denoising scheme is thus parameterized by some integer $M > 0$, increasing M increases the denoising strength. For instance, when $\mathcal{B}$ is the discrete Fourier basis, this corresponds to an ideal low-pass filter against a (discretized) Dirichlet kernel.

More advanced linear denoising operator can be designed by computing weighted average $\sum_m \lambda_m \langle f, \psi_m \rangle \psi_m$, and (8.2) is retrieve when using binary weights $\alpha_n = 1$ for $n \leqslant M$, and $\alpha_n = 0$ otherwise. The asymptotic theoretical performances described by the following theorem are however not improved by using non-binary weights.

Theorem 29. *We assume that $f_0 \in \mathbb{R}^N$ has a linear approximation error decay that satisfies*

$$\forall\, M, \quad \|f_0 - f_{0,M}^{lin}\|^2 \leqslant C M^{-2\beta} \quad \textit{where} \quad f_{0,M}^{lin} \overset{\text{def.}}{=} \sum_{m=1}^{M} \langle f_0, \psi_m \rangle \psi_m$$

for some constant C. Then the linear denoising error using (8.2) satisfies

$$\mathbb{E}(\|f_0 - \tilde{f}\|^2) \leqslant 2 C^{\frac{1}{2\beta+1}} \sigma^{2 - \frac{1}{\beta+1/2}},$$

when choosing

$$M = C^{\frac{1}{2\beta+1}} \sigma^{-\frac{2}{2\beta+1}}. \tag{8.3}$$

Proof. One has, thanks to the ortho-normality of $(\psi_m)_m$

$$\begin{aligned}
\mathbb{E}(\|f_0 - \tilde{f}\|^2) &= \mathbb{E}(\sum_m \langle f_0 - \tilde{f}, \psi_m \rangle^2) = \mathbb{E}(\sum_{m=1}^{M} \langle f_0 - f, \psi_m \rangle^2 + \sum_{m>M} \langle f_0, \psi_m \rangle^2) \\
&= \mathbb{E}\left(\sum_{m=1}^{M} \langle w, \psi_m \rangle^2 \right) + \sum_{m>M} \langle f_0, \psi_m \rangle^2 = M\sigma^2 + \|f_0 - f_{0,M}^{\text{lin}}\|^2 \\
&\leqslant M\sigma^2 + C M^{-2\beta}.
\end{aligned}$$

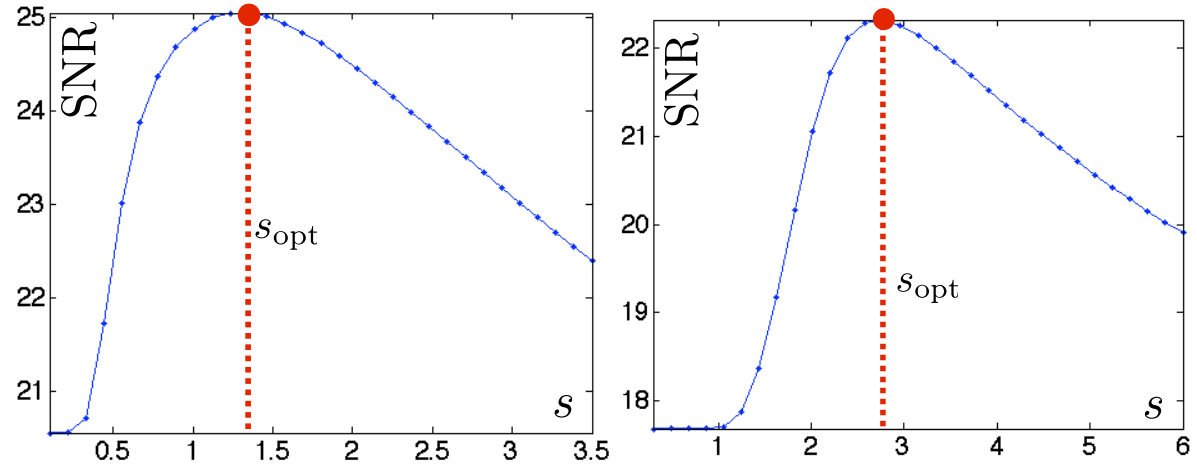


Figure 8.8: Curves of SNR as a function of the filtering width in 1-D (left) and 2-D (right).

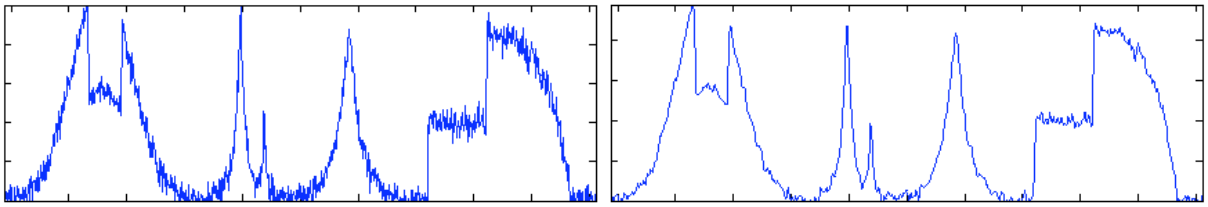


Figure 8.9: Noisy image (left) and denoising (right) using the optimal filter width.

Here we use the fundamental fact that $(\langle w, \psi_m \rangle)_m$ is also $\mathcal{N}(0, \sigma^2 \mathrm{Id}_N)$. Choosing M such that $M\sigma^2 = CM^{-2\beta}$, i.e. $M = C^{\frac{1}{2\beta+1}} \sigma^{-\frac{2}{2\beta+1}}$ leads to

$$\mathbb{E}(\|f_0 - \tilde{f}\|^2) = 2CM^{-2\beta} = 2CC^{-\frac{2\beta}{2\beta+1}} \sigma^{\frac{4\beta}{2\beta+1}} = 2C^{\frac{1}{2\beta+1}} \sigma^{2-\frac{1}{\beta+1/2}}.$$

□

There are several important remark regarding this simple but important result:

– Thanks to the decay of the linear approximation error, the denoising error $\mathbb{E}(\|f_0 - \tilde{f}\|^2)$ is bounded *independently* of the sampling size N, although the input noise level $\mathbb{E}(\|w\|^2) = N\sigma^2$ growth with N.

– If the signal is well approximated linearly, i.e. if β is large, then the denoising error decays fast when the noise level σ drops to zero. The upper bound approaches the optimal rate σ^2 by taking β large enough.

– This theory is finite dimensional, i.e. this computation makes only sense when introducing some discretization step N. This is natural because random noise vectors of finite energy are necessarily finite dimensional. For the choice (8.3) to be realizable, one should however have $M \leqslant N$, i.e. $N \geqslant C^{\frac{1}{2\beta+1}} \sigma^{-\frac{2}{2\beta+1}}$. Thus N should increase when the noise diminish for the denoising effect to kick-in.

– Section 6.3.1 bounds the linear approximation error for infinite dimensional signal and image model. This theory can be applied provided that the discretization error is smaller than the denoising error, i.e. once again, one should use N large enough.

A typical setup where this denoising theorem can be applied is for the Sobolev signal and image model detailed in Section 6.2.1. In the discrete setting, where the sampling size N is intended to grow (specially if σ diminishes), one can similarly consider a "Sobolev-like" model, and similarely as for Proposition 16, this model implies a decay of the linear approximation error.

Figure 8.10: Noisy image (left) and denoising (right) using the optimal filter width.

Proposition 19. *Assuming that*

$$\sum_{m=1}^{N} m^{2\alpha} |\langle f_0, \psi_m \rangle|^2 \leqslant C \tag{8.4}$$

then

$$\forall\, M, \quad \|f_0 - f_{0,M}^{lin}\|^2 \leqslant C M^{-2\alpha}$$

Proof.

$$C \geqslant \sum_{m=1}^{N} m^{2\alpha} |\langle f_0, \psi_m \rangle|^2 \geqslant \sum_{m>M} m^{2\alpha} |\langle f_0, \psi_m \rangle|^2 \geqslant M^{2\alpha} \sum_{m>M} |\langle f_0, \psi_m \rangle|^2 \geqslant M^{2\alpha} \|f_0 - f_{0,M}^{\mathrm{lin}}\|^2.$$

□

If ψ_m is the discrete Fourier basis defined in (2.8), then this discrete Sobolev model (8.4) is equivalent to the continuous Sobolev model of Section 6.2.1, up to a discretization error which tends to 0 as N increase. Choosing N large enough shows that smooth signals and image are thus efficiently denoised by a simple linear projection on the first M element of the Fourier basis.

8.3 Non-linear Denoising using Thresholding

8.3.1 Hard Thresholding

We consider an orthogonal basis $\{\psi_m\}_m$ of $\mathbb{C}^N$, for instance a discrete wavelet basis. The noisy coefficients satisfy

$$\langle f, \psi_m \rangle = \langle f_0, \psi_m \rangle + \langle w, \psi_m \rangle. \tag{8.5}$$

Since a Gaussian white noise is invariant under an orthogonal transformation, $\langle w, \psi_m \rangle$ is also a Gaussian white noise of variance σ^2. If the basis $\{\psi_m\}_m$ is efficient to represent f_0, then most of the coefficients $\langle f_0, \psi_m \rangle$ are close to zero, and one observes a large set of small noisy coefficients, as shown on Figure 8.11. This idea of using thresholding estimator for denoising was first systematically explored by Donoho and Jonhstone [19].

A thresholding estimator removes these small amplitude coefficients using a non-linear hard thresholding

$$\tilde{f} = \sum_{|\langle f, \psi_m \rangle| > T} \langle f, \psi_m \rangle \psi_m = \sum_m S_T(\langle f, \psi_m \rangle) \psi_m.$$

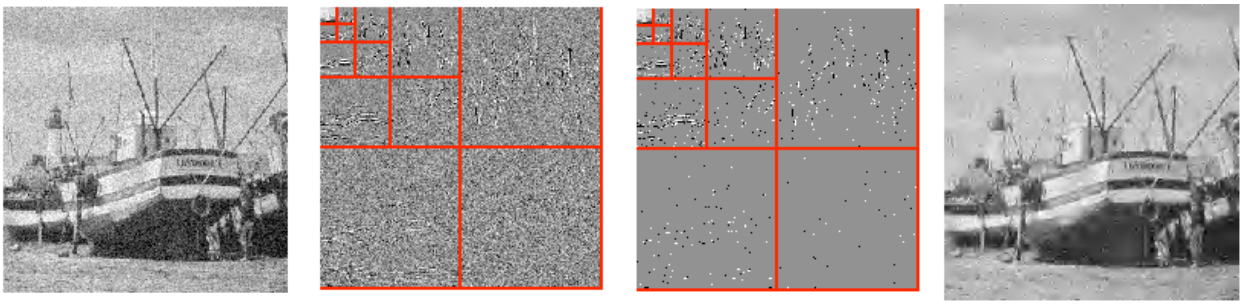


Figure 8.11: Denoising using thresholding of wavelet coefficients.

where S_T is defined in (6.4). This corresponds to the computation of the best M-term approximation $\tilde{f} = f_M$ of the noisy function f. Figure 8.11 shows that if T is well chose, this non-linear estimator is able to remove most of the noise while maintaining sharp features, which was not the case with linear filtering estimatiors.

8.3.2 Soft Thresholding

We recall that the hard thresholding operator is defined as

$$S_T(x) = S_T^0(x) = \begin{cases} x & \text{if } |x| > T, \\ 0 & \text{if } |x| \leqslant T. \end{cases} \tag{8.6}$$

This thresholding performs a binary decision that might introduces artifacts. A less aggressive nonlinearity is the soft thresholding

$$S_T^1(x) = \max(1 - T/|x|, 0)x. \tag{8.7}$$

Figure 8.12 shows the 1-D curves of these 1-D non-linear mapping.

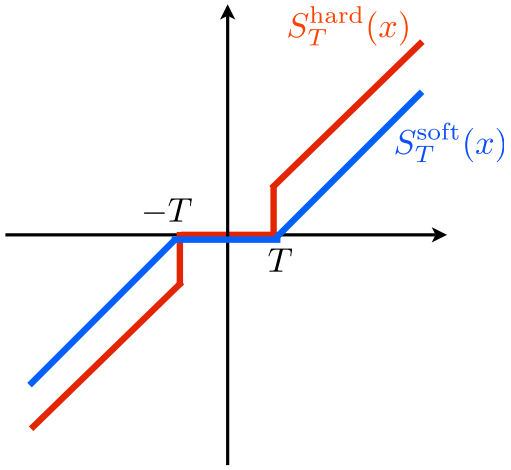


Figure 8.12: Hard and soft thresholding functions.

For $q = 0$ and $q = 1$, these thresholding defines two different estimators

$$\tilde{f}^q = \sum_m S_T^q(\langle f, \psi_m \rangle)\psi_m \tag{8.8}$$

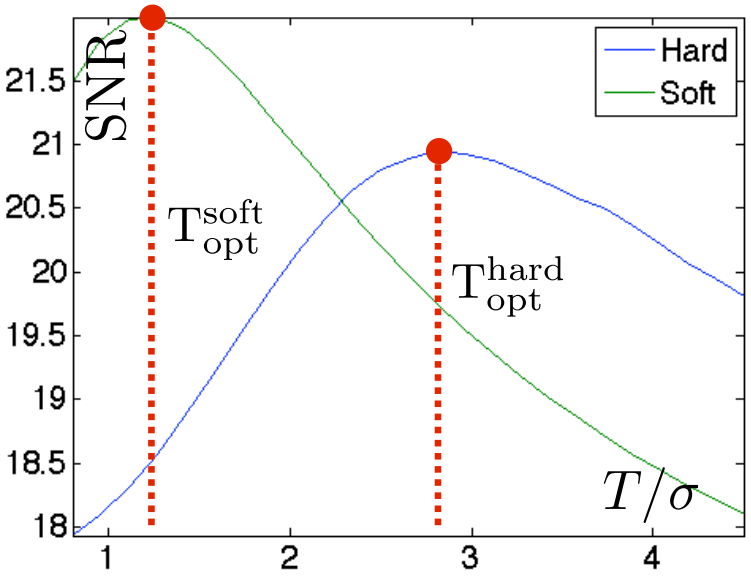


Figure 8.13: Curves of SNR with respect to T/σ for hard and soft thresholding.

Coarse scale management. The soft thresholded S^1_T introduces a bias since it diminishes the value of large coefficients. For wavelet transforms, it tends to introduces unwanted low-frequencies artifacts by modifying coarse scale coefficients. If the coarse scale is 2^{j_0}, one thus prefers not to threshold the coarse approximation coefficients and use, for instance in 1-D,

$$\tilde{f}^1 = \sum_{0 \leqslant n < 2^{-j_0}} \langle f, \varphi_{j_0,n} \rangle \varphi_{j_0,n} + \sum_{j=j_0}^{0} \sum_{0 \leqslant n < 2^{-j}} S^1_T(\langle f, \psi_{j_0,n} \rangle) \psi_{j_0,n}.$$

Empirical choice of the threshold. Figure 8.13 shows the evolution of the SNR with respect to the threshold T for these two estimators, for a natural image f_0. For the hard thresholding, the best result is obtained around $T \approx 3\sigma$, while for the soft thresholding, the optimal choice is around $T \approx 3\sigma/2$. These results also shows that numerically, for thresholding in orthogonal bases, soft thresholding is slightly superior than hard thresholding on natural signals and images.

Although these are experimental conclusions, these results are robust across various natural signals and images, and should be considered as good default parameters.

8.3.3 Minimax Optimality of Thresholding

Sparse coefficients estimation. To analyze the performance of the estimator, and gives an estimate for the value of T, we first assumes that the coefficients

$$a_{0,m} = \langle f_0, \psi_m \rangle \in \mathbb{R}^N$$

are sparse, meaning that most of the $a_{0,m}$ are zero, so that its ℓ^0 norm

$$\|a_0\|_0 = \#\{m \,;\, a_{0,m} \neq 0\}$$

is small. As shown in (8.5), noisy coefficients

$$\langle f, \psi_m \rangle = a_m = a_{0,m} + z_m$$

20.9dB 21.8dB

Figure 8.14: Comparison of hard (left) and soft (right) thresholding.

are perturbed with an additive Gaussian white noise of variance σ^2. Figure 8.15 shows an example of such a noisy sparse signal.

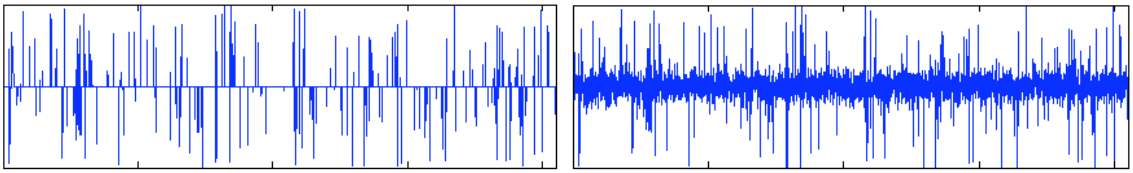


Figure 8.15: Left: sparse signal a, right: noisy signal.

Universal threshold value. If

$$\min_{m:a_{0,m}\neq 0} |a_{0,m}|$$

is large enough, then $\|f_0 - \tilde{f}\| = \|a_0 - S_T(a)\|$ is minimum for

$$T \approx \tau_N = \max_{0\leqslant m<N} |z_m|.$$

τ_N is a random variable that depends on N. One can show that its mean is $\sigma\sqrt{2\log(N)}$, and that as N increases, its variance tends to zero and τ_N is highly concentrated close to its mean. Figure 8.16 shows that this is indeed the case numerically.

Asymptotic optimality. Donoho and Jonhstone [19] have shown that the universal threshold $T = \sigma\sqrt{2\log(N)}$ is a good theoretical choice for the denoising of signals that are well approximated non-linearly in $\{\psi_m\}_m$. The obtain denoising error decay rate with σ can also be shown to be in some sense optimal.

Theorem 30. *We assume that $f_0 \in \mathbb{R}^N$ has a non-linear approximation error decay that satisfies*

$$\forall\, M, \quad \|f_0 - f_{0,M}^{nlin}\|^2 \leqslant CM^{-2\beta} \quad \textit{where} \quad f_{0,M}^{nlin} \stackrel{\text{def.}}{=} \sum_{r=1}^{M} \langle f_0,\, \psi_{m_r}\rangle \psi_{m_r}$$

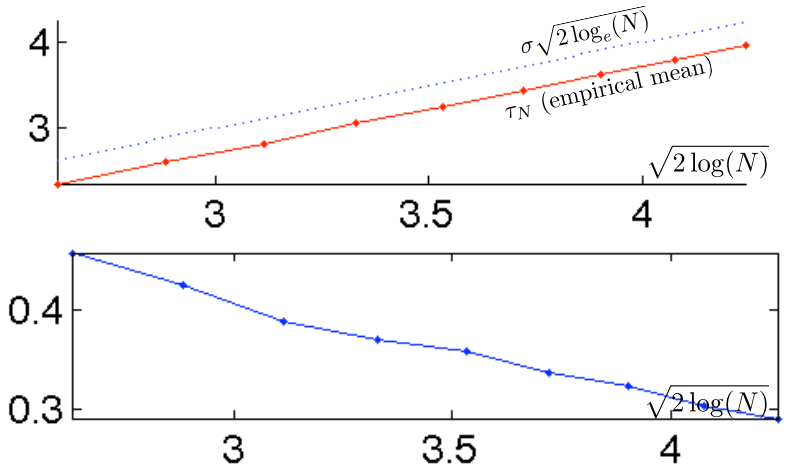


Figure 8.16: Empirical estimation of the mean of Z_n (top) and standard deviation of Z_n (bottom)

for some constant C, where here $(\langle f_0, \psi_{m_r}\rangle)_r$ are the coefficient sorted by decaying magnitude. Then the non-linear denoising error using (8.2) satisfies

$$\mathbb{E}(\|f_0 - \tilde{f}^q\|^2) \leqslant C' \ln(N) \sigma^{2-\frac{1}{\beta+1/2}},$$

for some constant C', when choosing $T = \sqrt{2\ln(N)}$, where $\tilde{f}^q$ is defined in (8.8) for $q \in \{0,1\}$.

This universal threshold choice $T = \sqrt{2\ln(N)}$ is however very conservative since it is guaranteed to remove almost all the noise. In practice, as shown in Figure 8.14, better results are obtained on natural signals and images by using $T \approx 3\sigma$ and $T \approx 3\sigma/2$ for hard and soft thresholdings.

8.3.4 Translation Invariant Thresholding Estimators

Translation invariance. Let $f \mapsto \tilde{f} = \mathcal{D}(f)$ by a denoising method, and $f_\tau(x) = f(x-\tau)$ be a translated signal or image for $\tau \in \mathbb{R}^d$, ($d = 1$ or $d = 2$). The denoising is said to be translation invariant at precision Δ if

$$\forall\, \tau \in \Delta, \quad \mathcal{D}(f) = \mathcal{D}(f_\tau)_{-\tau}$$

where Δ is a lattice of $\mathbb{R}^d$. The denser Δ is, the more translation invariant the method is. This corresponds to the fact that $\mathcal{D}$ computes with the translation operator.

$$\begin{array}{ccc} f & \xrightarrow{\mathcal{D}} & \mathcal{D}(f) \\ \tau \downarrow & & \uparrow -\tau \\ f_\tau & \xrightarrow[\mathcal{D}]{} & \mathcal{D}(f_\tau) \end{array}$$

Imposing translation invariance for a fine enough set Δ is a natural constraint, since intuitively the denoising results should not depend on the location of features in the signal or image. Otherwise, some locations might be favored by the denoising process, which might result in visually unpleasant denoising artifacts.

For denoising by thresholding

$$\mathcal{D}(f) = \sum_m S_T(\langle f, \psi_m\rangle)\psi_m.$$

then translation invariance is equivalent to asking that the basis $\{\psi_m\}_m$ is translation invariant at precision Δ,

$$\forall\, m,\ \forall\, \tau \in \Delta,\ \exists m,\ \exists \lambda \in \mathbb{C}, \qquad (\psi_{m'})_\tau = \lambda \psi_m$$

where $|\lambda| = 1$.

The Fourier basis is fully translation invariant for $\Delta = \mathbb{R}^d$ over $[0,1]^d$ with periodic boundary conditions and the discrete Fourier basis is translation invariant for all interger translations $\Delta = \{0, \ldots, N_0 - 1\}^d$ where $N = N_0$ is the number of points in 1-D, and $N = N_0 \times N_0$ is the number of pixels in 2-D.

Unfortunately, an orthogonal wavelet basis

$$\{\psi_m = \psi_{j,n}\}_{j,n}$$

is not translation invariant both in the continuous setting or in the discrete setting. For instance, in 1-D,

$$(\psi_{j',n'})_\tau \notin \{\psi_{j,n}\} \quad \text{for} \quad \tau = 2^j/2.$$

Cycle spinning. A simple way to turn a denoiser Δ into a translation invariant denoiser is to average the result of translated images

$$\mathcal{D}_{\text{inv}}(f) = \frac{1}{|\Delta|} \sum_{\tau \in \Delta} \mathcal{D}(f_\tau)_{-\tau}. \tag{8.9}$$

One easily check that

$$\forall\, \tau \in \Delta, \quad \mathcal{D}_{\text{inv}}(f) = \mathcal{D}_{\text{inv}}(f_\tau)_{-\tau}$$

To obtain a translation invariance up to the pixel precision for a data of N samples, one should use a set of $|\Delta| = N$ translation vectors. To obtain a pixel precision invariance for wavelets, this will result in $O(N^2)$ operations.

Figure 8.17 shows the result of applying cycle spinning to an orthogonal hard thresholding denoising using wavelets, where we have used the following translation of the continuous wavelet basis $\Delta = \{0, 1/N, 2/N, 3/N\}^2$, which corresponds to discrete translation by $\{0,1,2,3\}^2$ on the discretized image. The complexity of the denoising scheme is thus 16 wavelet transforms. The translation invariance brings a very large SNR improvement, and significantly reduces the oscillating artifacts of orthogonal thresholding. This is because this artifacts pop-out at random locations when τ changes, so that the averaging process reduces significantly these artifacts.

Figure 8.18 shows that translation invariant hard thresholding does a slightly better job than translation invariant soft thresholding. The situation is thus reversed with respect to thresholding in an orthogonal wavelet basis.

Translation invariant wavelet frame. An equivalent way to define a translation invariant denoiser is to replace the orthogonal basis $\mathcal{B} = \{\psi_m\}$ by a redundant family of translated vectors

$$\mathcal{B}_{\text{inv}} = \{(\psi_m)_\tau\}_{m, \tau \in \Delta}. \tag{8.10}$$

One should be careful about the fact that $\mathcal{B}_{\text{inv}}$ is not any more an orthogonal basis, but it still enjoy a conservation of energy formula

$$\|f\|^2 = \frac{1}{|\Delta|} \sum_{m,\tau \in \Delta} |\langle f, (\psi_m)_\tau \rangle|^2 \quad \text{and} \quad f = \frac{1}{|\Delta|} \sum_{m,\tau \in \Delta} \langle f, (\psi_m)_\tau \rangle (\psi_m)_\tau.$$

This kind of redundant family are called tight frames.

One can then define a translation invariant thresholding denoising

$$\mathcal{D}_{\text{inv}}(f) = \frac{1}{|\Delta|} \sum_{m,\tau \in \Delta} S_T(\langle f, (\psi_m)_\tau \rangle)(\psi_m)_\tau. \tag{8.11}$$

21.8dB 23.4dB

Figure 8.17: Comparison of wavelet orthogonal soft thresholding (left) and translation invariant wavelet hard thresholding (right).

This denoising is the same as the cycle spinning denoising defined in (8.9).

The frame $\mathcal{B}_{\text{inv}}$ might contain up to $|\Delta||\mathcal{B}|$ basis element. For a discrete basis of signal with N samples, and a translation lattice of $|\Delta| = N$ vectors, it corresponds to up to N^2 elements in $\mathcal{B}_{\text{inv}}$. Hopefully, for a hierarchical basis such as a discrete orthogonal wavelet basis, one might have

$$(\psi_m)_\tau = (\psi_{m'})_{\tau'} \quad \text{for} \quad m \neq m' \quad \text{and} \quad \tau \neq \tau',$$

so that the number of elements in $\mathcal{B}_{\text{inv}}$ might be much smaller than N^2. For instance, for an orthogonal wavelet basis, one has

$$(\psi_{j,n})_{k2^j} = \psi_{j,n+k},$$

so that the number of basis elements is $|\mathcal{B}_{\text{inv}}| = N \log_2(N)$ for a 2-D basis, and $3N \log_2(N)$ for a 2-D basis. The fast translation invariant wavelet transform, also called "a trou" wavelet transform, computes all the inner products $\langle f, (\psi_m)_\tau \rangle$ in $O(N \log_2(N))$ operations. Implementing formula (8.11) is thus much faster than applying the cycle spinning (8.9) equivalent formulation.

Translation invariant wavelet coefficients are usually grouped by scales in $\log_2(N)$ (for $d = 1$) or by scales and orientations $3 \log_2(N)$ (for $d = 2$) sets of coefficients. For instance, for a 2-D translation invariant transform, one consider

$$\forall\, n \in \{0, \ldots, 2^j N_0 - 1\}^2,\ \forall\, k \in \{0, \ldots, 2^{-j}\}^2, \quad d_j^\omega[2^{-j}n + k] = \langle f, (\psi_{j,n})_{k2^j} \rangle$$

where $\omega \in \{V, H, D\}$ is the orientation. Each set d_j^ω has N coefficients and is a band-pass filtered version of the original image f, as shown on Figure 8.19.

Figure 8.20 shows how these set of coefficients are hard thresholded by the translation invariant estimator.

8.3.5 Exotic Thresholdings

It is possible to devise many thresholding nonlinearities that interpolate between the hard and soft thresholder. We present here two examples, but many more exist in the literature. Depending on the statistical distribution of the wavelet coefficients of the coefficients of f in the basis, these thresholders might produce slightly better results.

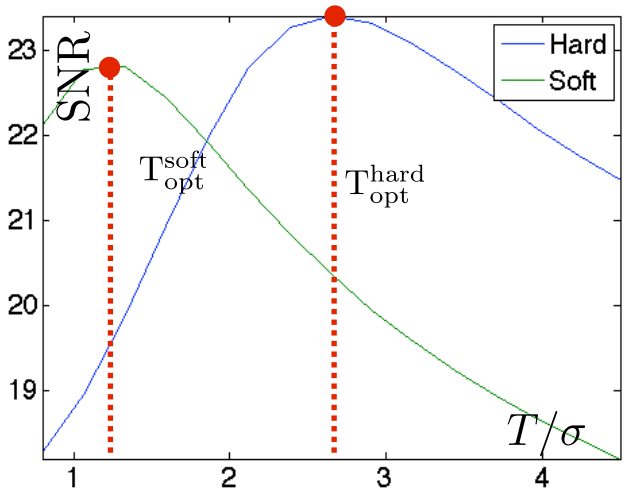


Figure 8.18: Curve of SNR with respect to T/σ for translation invariant thresholding.

Semi-soft thresholding. One can define a family of intermediate thresholder that depends on a parameter $\mu > 1$

$$S_T^\theta(x) = g_{\frac{1}{1-\theta}}(x) \quad \text{where} \quad g_\mu(x) = \begin{cases} 0 & \text{if} \quad |x| < T \\ x & \text{if} \quad |x| > \mu T \\ \text{sign}(x)\frac{|x|-T}{\mu-1} & \text{otherwise.} \end{cases}$$

One thus recovers the hard thresholding as S_T^0 and the soft thresholding as S_T^1. Figure 8.21 display an example of such a non-linearity.

Figure 8.22 shows that a well chosen value of μ might actually improves over both hard and soft thresholders. The improvement is however hardly noticeable visually.

Stein thresholding. The Stein thresholding is defined using a quadratic attenuation of large coefficients

$$S_T^{\text{Stein}}(x) = \max\left(1 - \frac{T^2}{|x|^2}, 0\right) x.$$

This should be compared with the linear attenuation of the soft thresholding

$$S_T^1(x) = \max\left(1 - \frac{T}{|x|}, 0\right) x.$$

The advantage of the Stein thresholder with respect to the soft thresholding is that

$$|S_T^{\text{Stein}}(x) - x| \to 0 \quad \text{whereas} \quad |S_T^1(x) - x| \to T,$$

where $x \to \pm\infty$. This means that Stein thresholding does not suffer from the bias of soft thresholding.

For translation invariant thresholding, Stein and hard thresholding perform similarly on natural images.

8.3.6 Block Thresholding

The non-linear thresholding method presented in the previous section are diagonal estimators, since they operate a coefficient-by-coefficient attenuation

$$\tilde{f} = \sum_m A_T^q(\langle f, \psi_m \rangle)\langle f, \psi_m \rangle \psi_m$$

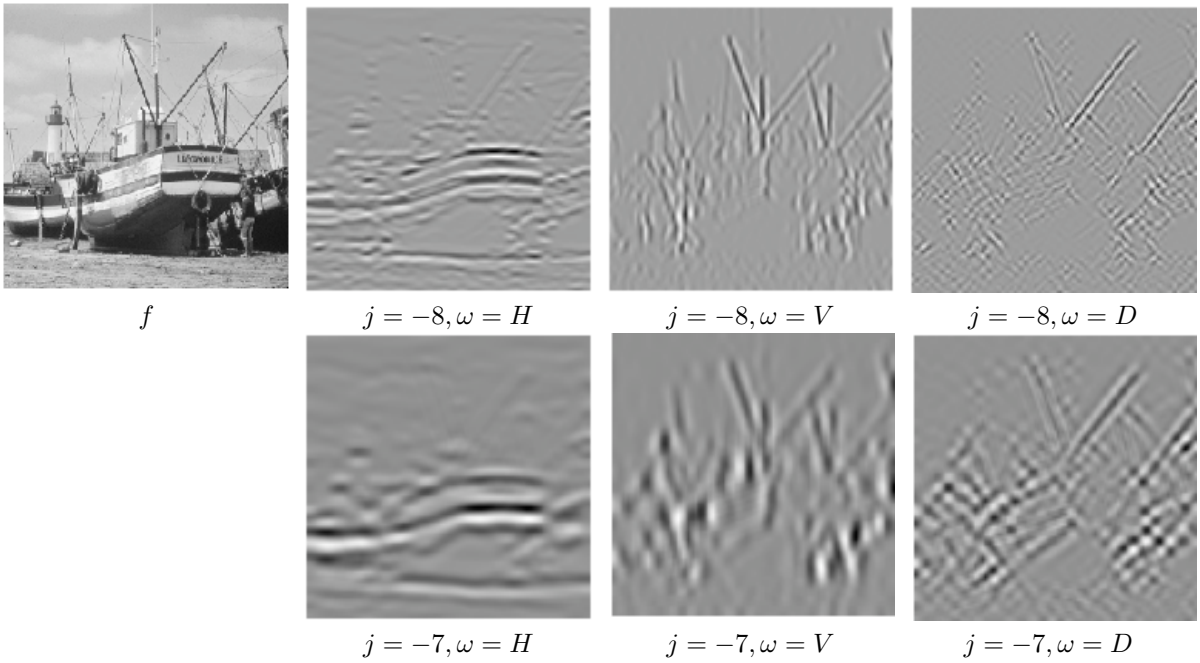


Figure 8.19: Translation invariant wavelet coefficients.

where

$$A_T^q(x) = \begin{cases} \max(1-x^2/T^2,0) & \text{for} \quad q=\text{Stein} \\ \max(1-|x|/T,0) & \text{for} \quad q=1 \text{ (soft)} \\ 1_{|x|>T} & \text{for} \quad q=0 \text{ (hard)} \end{cases}$$

Block thresholding takes advantage of the statistical dependancy of wavelet coefficients, by computing the attenuation factor on block of coefficients. This is especially efficient for natural images, where edges and geometric features create clusters of high magnitude coefficients. Block decisions also help to remove artifacts due to isolated noisy large coefficients in regular areas.

The set of coefficients is divided into disjoint blocks, and for instance for 2-D wavelet coefficients

$$\{(j,n,\omega)\}_{j,n,\omega} = \bigcup_k B_k,$$

where each B_k is a square of $s \times s$ coefficients, where the block size s is a parameter of the method. Figure 8.24 shows an example of such a block.

The block energy is defined as

$$B_k = \frac{1}{s^2} \sum_{m \in B_k} |\langle f, \psi_m \rangle|^2,$$

and the block thresholding

$$\tilde{f} = \sum_m S_T^{\text{block},q}(\langle f, \psi_m \rangle)\psi_m$$

makes use of the same attenuation for all coefficients within a block

$$\forall\, m \in B_k, \quad S_T^{\text{block},q}(\langle f, \psi_m \rangle) = A_T^q(E_k)\langle f, \psi_m \rangle.$$

for $q \in \{0, 1, \text{stein}\}$. Figure 8.24 shows the effect of this block attenuation, and the corresponding denoising result.

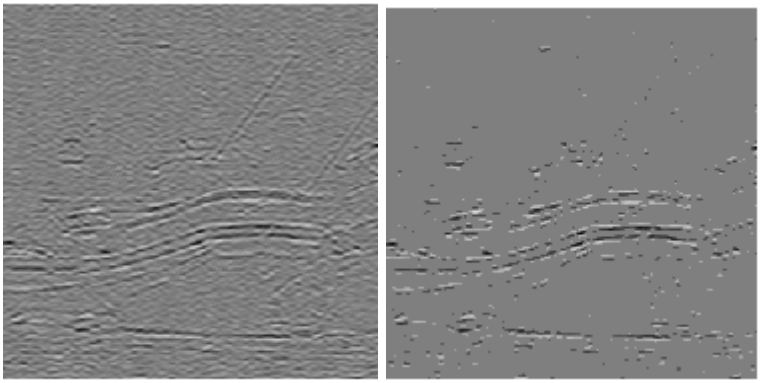


Figure 8.20: Left: translation invariant wavelet coefficients, for $j = -8, \omega = H$, right: tresholded coefficients.

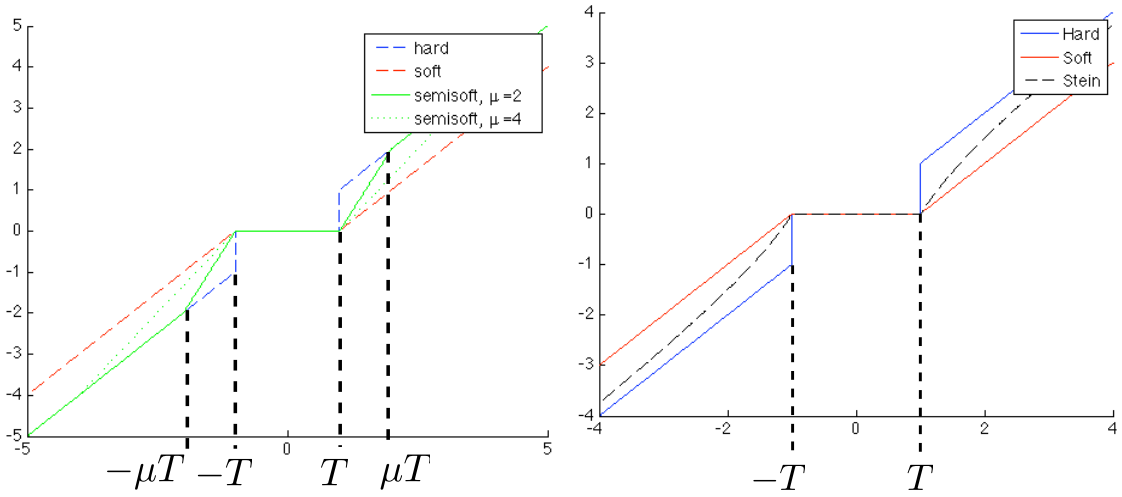


Figure 8.21: Left: semi-soft thresholder, right: Stein thresholder.

Figure 8.25, left, compares the three block thresholding obtained for $q \in \{0, 1, \text{stein}\}$. Numerically, on natural images, Stein block thresholding gives the best results. Figure 8.25, right, compares the block size for the Stein block thresholder. Numerically, for a broad range of images, a value of $s = 4$ works well.

Figure 8.26 shows a visual comparison of the denoising results. Block stein thresholding of orthogonal wavelet coefficients gives a result nearly as good as a translation invariant wavelet hard thresholding, with a faster algorithm. The block thresholding strategy can also be applied to wavelet coefficients in translation invariant tight frame, which produces the best results among all denoisers detailed in this book.

Code **??** implement this block thresholding.

One should be aware that more advanced denoisers use complicated statistical models that improves over the methods proposed in this book, see for instance [34].

8.4 Data-dependant Noises

For many imaging devices, the variance of the noise that perturbs $f_{0,n}$ depends on the value of $f_{0,n}$. This is a major departure from the additive noise formation model considered so far. We present here two popular examples of such non-additive models.

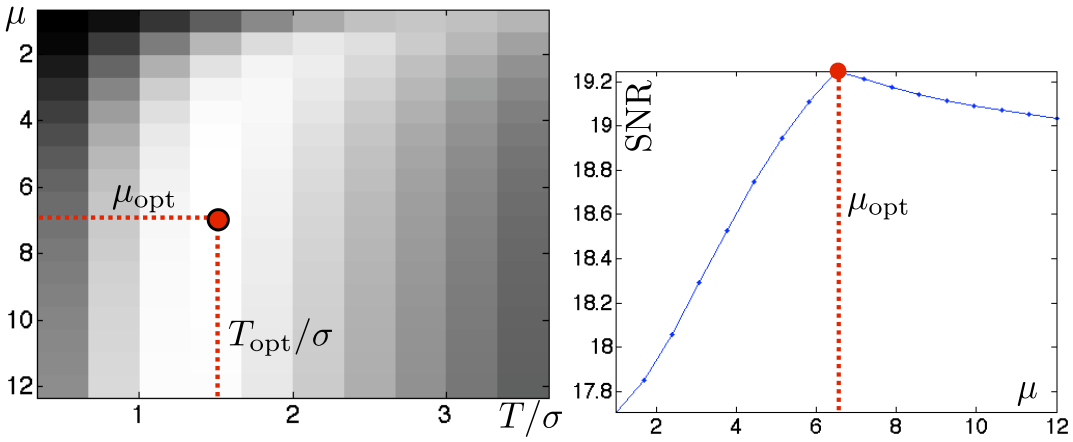


Figure 8.22: Left: image of SNR with respect to the parameters μ and T/σ, right: curve of SNR with respect to μ using the best T/σ for each μ.

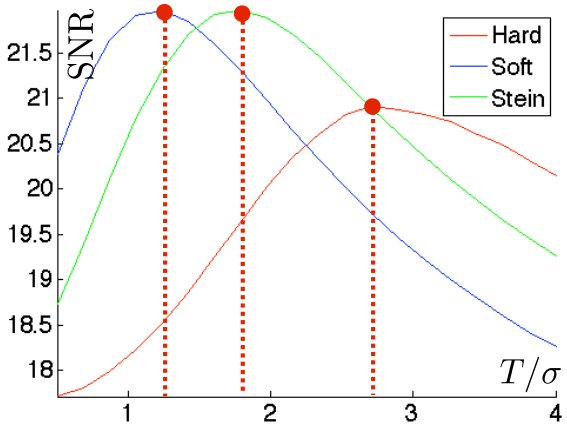


Figure 8.23: SNR curves with respect to T/σ for Stein threhsolding.

8.4.1 Poisson Noise

Many imaging devices sample an image through a photons counting operation. This is for instance the case in digital camera, confocal microscopy, TEP and SPECT tomography.

Poisson model. The uncertainty of the measurements for a quantized unknown image $f_{0,n} \in \mathbb{N}$ is then modeled using a Poisson noise distribution

$$f_n \sim \mathcal{P}(\lambda) \quad \text{where} \quad \lambda = f_{0,n} \in \mathbb{N},$$

and where the Poisson distribution $\mathcal{P}(\lambda)$ is defined as

$$\mathbb{P}(f_n = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

and thus varies from pixel to pixel. Figure 8.27 shows examples of Poisson distributions.

One has

$$\mathbb{E}(f_n) = \lambda = f_{0,n} \quad \text{and} \quad \text{Var}(f_n) = \lambda = f_{0,n}$$

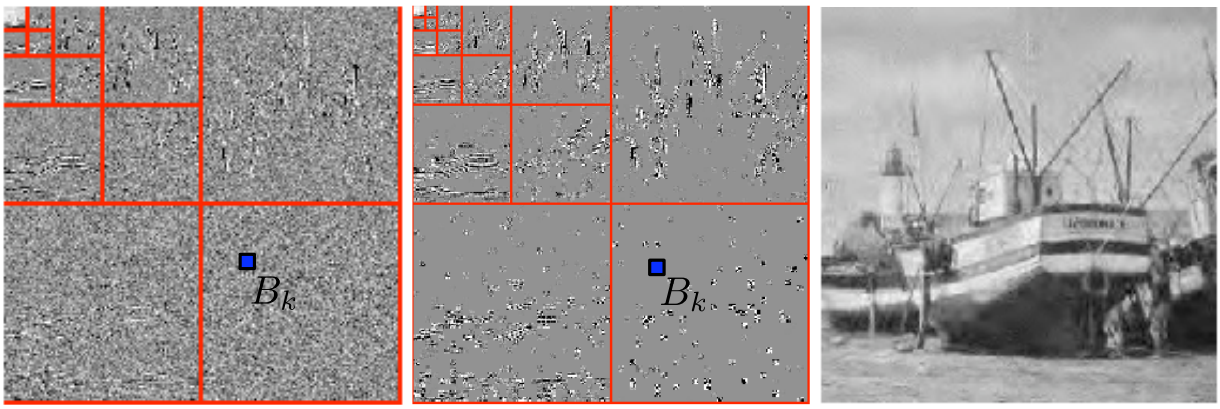


Figure 8.24: Left: wavelet coefficients, center: block thresholded coefficients, right: denoised image.

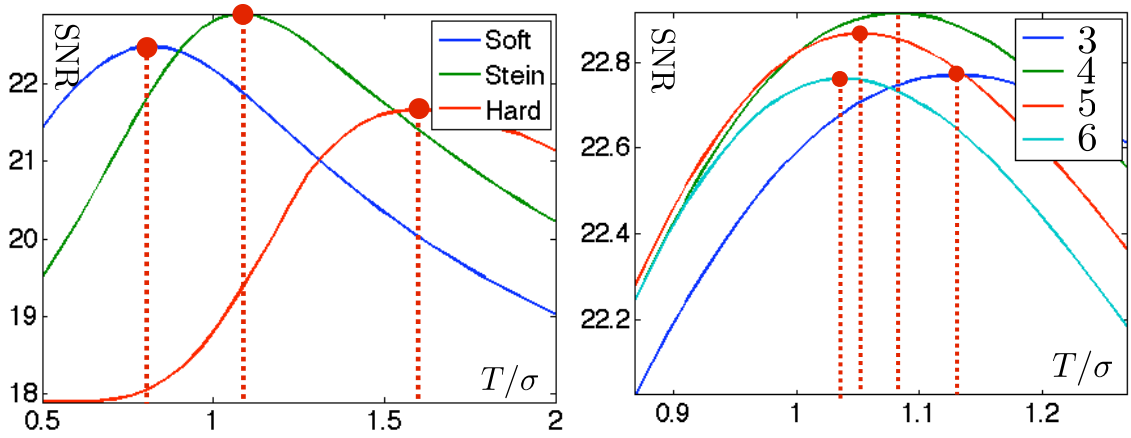


Figure 8.25: Curve of SNR with respect to T/σ (left) and comparison of SNR for different block size (right).

so that the denoising corresponds to estimating the mean of a random vector from a single observation, but the variance now depends on the pixel intensity. This shows that the noise level increase with the intensity of the pixel (more photons are coming to the sensor) but the relative variation $(f_n - f_{0,n})/f_{0,n}$ tends to zero in expectation when $f_{0,n}$ increases.

Figure 8.28 shows examples of a clean image f_0 quantized using different values of $\lambda_{\max}$ and perturbed with the Poisson noise model.

Variance stabilization. Applying thresholding estimator

$$\mathcal{D}(f) = \sum_m S_T^q(\langle f, \psi_m \rangle)\psi_m$$

to f might give poor results since the noise level fluctuates from point to point, and thus a single threshold T might not be able to capture these variations. A simple way to improve the thresholding results is to first apply a variance stabilization non-linearity $\varphi : \mathbb{R} \to \mathbb{R}$ to the image, so that $\varphi(f)$ is as close as possible to an additive Gaussian white noise model

$$\varphi(f) \approx \varphi(f_0) + w \tag{8.12}$$

SNR=23.4dB 22.8dB 23.8dB

Figure 8.26: Left: translation invariant wavelet hard thresholding, center: block orthogonal Stein thresholding, right: block translation invariant Stein thresholding.

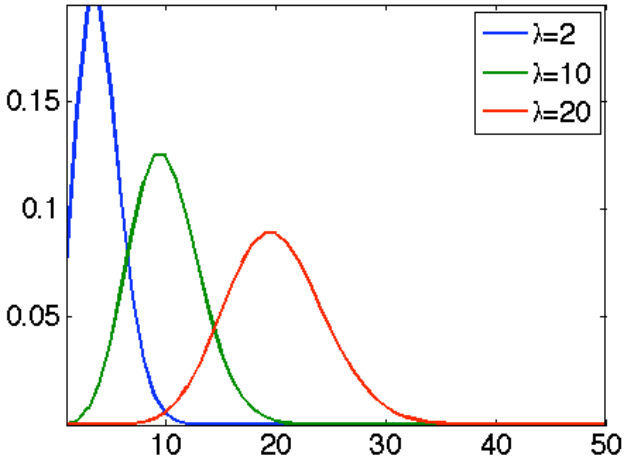


Figure 8.27: Poisson distributions for various λ.

where $w_n \sim \mathcal{N}(0, \sigma)$ is a Gaussian white noise of fixed variance σ^2.

Perfect stabilization is impossible, so that (8.12) only approximately holds for a limited intensity range of $f_{0,n}$. Two popular variation stabilization functions for Poisson noise are the Anscombe mapping

$$\varphi(x) = 2\sqrt{x + 3/8}$$

and the mapping of Freeman and Tukey

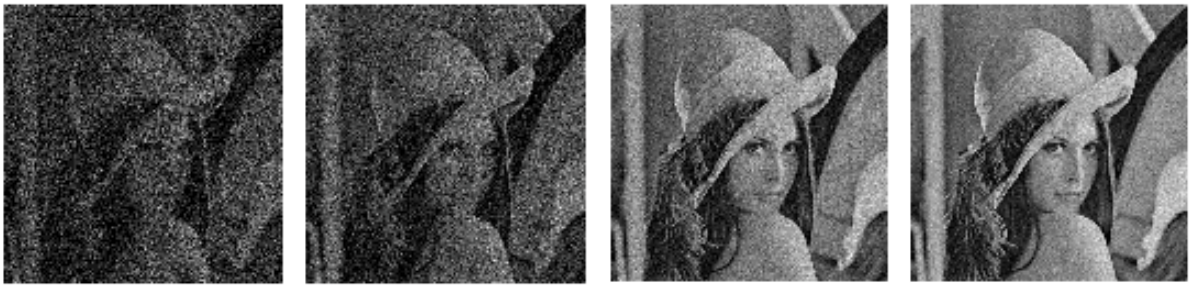
$$\varphi(x) = \sqrt{x+1} + \sqrt{x}.$$

Figure 8.29 shows the effect of these variance stabilizations on the variance of $\varphi(f)$.

A variance stabilized denoiser is defined as

$$\Delta^{\text{stab},q}(f) = \varphi^{-1}(\sum_m S_T^q(\langle \varphi(f),\, \psi_m \rangle)\psi_m)$$

where φ^{-1} is the inverse mapping of φ.



$\lambda_{\max} = 5$ $\lambda_{\max} = 50$ $\lambda_{\max} = 50$ $\lambda_{\max} = 100$

Figure 8.28: Noisy image with Poisson noise model, for various $\lambda_{\max} = \max_n f_{0,n}$.

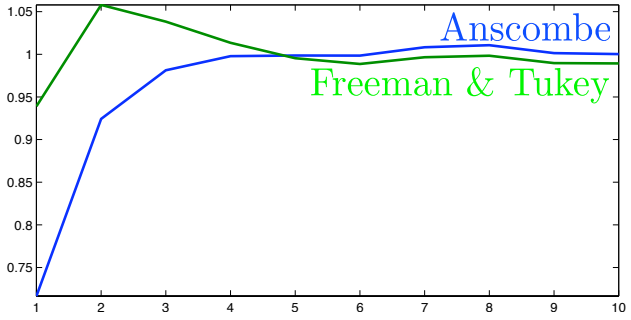


Figure 8.29: Comparison of variariance stabilization: display of $\mathrm{Var}(\varphi(f_n))$ as a function of $f_{0,n}$.

Figure 8.30 shows that for moderate intensity range, variance stabilization improves over non-stabilized denoising.

8.4.2 Multiplicative Noise

Multiplicative image formation. A multiplicative noise model assumes that

$$f_n = f_{0,n} w_n$$

where w is a realization of a random vector with $\mathbb{E}(w) = 1$. Once again, the noise level depends on the pixel value

$$\mathbb{E}(f_n) = f_{0,n} \quad \text{and} \quad \mathrm{Var}(f_n) = f_{0,n}^2 \sigma^2 \quad \text{where} \quad \sigma^2 = \mathrm{Var}(w).$$

Such a mutiplicative noise is a good model for SAR satellite imaging, where f is obtained by averaging S images

$$\forall\, 0 \leqslant s < K, \quad f_n^{(s)} = f_{0,n} w_n^{(s)} + r_n^{(s)}$$

where $r^{(s)}$ is a Gaussian white noise, and $w_n^{(s)}$ is distributed according to a one-sided exponential distribution

$$\mathcal{P}(w_n^{(s)} = x) \propto e^{-x}\, \mathbb{I}_{x>0}.$$

For K large enough, averaging the images cancels the additive noise and one obtains

$$f_n = \frac{1}{K} \sum_{s=1}^{K} f_n^{(s)} \approx f_{0,n} w_n$$

Figure 8.30: Left: noisy image, center: denoising without variance stabilization, right: denoising after variance stabilization.

where w is distributed according to a Gamma distribution

$$w \sim \Gamma(\sigma = K^{-\frac{1}{2}}, \mu = 1) \quad \text{where} \quad \mathbb{P}(w = x) \propto x^{K-1} e^{-Kx},$$

One should note that increasing the value of K reduces the overall noise level.

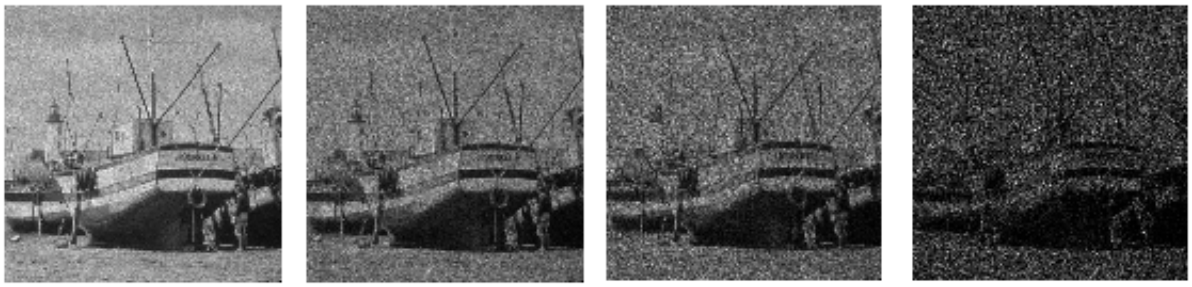


Figure 8.31: Noisy images with multiplicative noise, with varying σ.

Figure **??** shows an example of such image formation for a varying number $K = 1/\sigma^2$ of averaged images.

A simple variance stabilization transform is

$$\varphi(x) = \log(x) - c$$

where

$$c = \mathrm{E}(\log(w)) = \psi(K) - \log(K) \quad \text{where} \quad \psi(x) = \Gamma'(x)/\Gamma(x)$$

and where Γ is the Gamma function that generalizes the factorial function to non-integer. One thus has

$$\varphi(f)_n = \varphi(f_0)_n + z_n,$$

where $z_n = \log(w) - c$ is a zero-mean additive noise.

Figure 8.32 shows the effect of this variance stabilization on the repartition of w and z.

Figure 8.33 shows that for moderate noise level σ, variance stabilization improves over non-stabilized denoising.

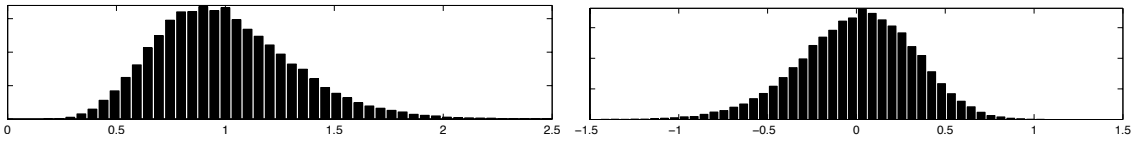


Figure 8.32: Histogram of multiplicative noise before (left) and after (right) stabilization.

Figure 8.33: Left: noisy image, center: denoising without variance stabilization, right: denoising after variance stabilization.

Chapter 9

Variational Priors and Regularization

9.1 Sobolev and Total Variation Priors

The simplest prior are obtained by integrating local differential quantity over the image. They corresponds to norms in functional spaces that imposes some smoothness on the signal of the image. We detail here the Sobolev and the total variation priors, that are the most popular in image processing.

9.1.1 Continuous Priors

In the following, we consider either continuous functions $f \in \mathrm{L}^2([0,1]^2)$ or discrete vectors $f \in \mathbb{R}^N$, and consider continuous priors and there discrete counterparts in Section 9.1.2.

Sobolev prior. The prior energy $J(f) \in \mathbb{R}$ is intended to be low for images in a class $f \in \Theta$. The class of uniformly smooth functions detailed in Section 6.2.1 corresponds to functions in Sobolev spaces. A simple prior derived from this Sobolev class is thus

$$J_{\mathrm{Sob}}(f) = \frac{1}{2}\|f\|_{\mathrm{Sob}}^2 = \frac{1}{2}\int \|\nabla f(x)\|^2 \mathrm{d}x, \tag{9.1}$$

where ∇f is the gradient in the sense of distributions.

Total variation prior. To take into account discontinuities in images, one considers a total variation energy, introduced in Section 6.2.3. It was introduced for image denoising by Rudin, Osher and Fatemi [36]

The total variation of a smooth image f is defined as

$$J_{\mathrm{TV}}(f) = \|f\|_{\mathrm{TV}} = \int \|\nabla_x f\| \mathrm{d}x. \tag{9.2}$$

This energy extends to non-smooth functions of bounded variations $f \in \mathrm{BV}([0,1]^2)$. This class contains indicators functions $f = 1_\Omega$ of sets Ω with a bounded perimeter $|\partial\Omega|$.

The total variation norm can be computed alternatively using the co-area formula (6.12), which shows in particular that $\|1_\Omega\|_{\mathrm{TV}} = |\partial\Omega|$.

9.1.2 Discrete Priors

An analog image $f \in \mathrm{L}^2([0,1]^2)$ is discretized through an acquisition device to obtain a discrete image $f \in \mathbb{R}^N$. Image processing algorithms work on these discrete data, and we thus need to define discrete priors for finite dimensional images.

Discrete gradient. Discrete Sobolev and total variation priors are obtained by computing finite differences approximations of derivatives, using for instance forward differences

$$\delta_1 f_{n_1,n_2} = f_{n_1+1,n_2} - f_{n_1,n_2}$$
$$\delta_2 f_{n_1,n_2} = f_{n_1,n_2+1} - f_{n_1,n_2},$$

and one can use higher order schemes to process more precisely smooth functions. One should be careful with boundary conditions, and we consider here for simplicity periodic boundary conditions, which correspond to computing the indexes $n_i + 1$ modulo N. More advanced symmetric boundary conditions can be used as well to avoid boundary artifacts.

A discrete gradient is defined as

$$\nabla f_n = (\delta_1 f_n, \delta_2 f_n) \in \mathbb{R}^2$$

which corresponds to a mapping from images to vector fields

$$\nabla : \mathbb{R}^N \longrightarrow \mathbb{R}^{N \times 2}.$$

Figure 9.1 shows examples of gradient vectors. They point in the direction of the largest slope of the function discretized by f. Figure 9.2 shows gradients and their norms displayed as an image. Regions of high gradients correspond to large intensity variations, and thus typically to edges or textures.

Figure 9.1: Discrete gradient vectors.

Discrete divergence. One can also use backward differences,

$$\tilde{\delta}_1 f_{n_1,n_2} = f_{n_1,n_2} - f_{n_1-1,n_2}$$
$$\tilde{\delta}_2 f_{n_1,n_2} = f_{n_1,n_2} - f_{n_1,n_2-1}.$$

They are dual to the forward differences, so that

$$\delta_i^* = -\tilde{\delta}_i,$$

which means that

$$\forall\, f, g \in \mathbb{R}^N, \quad \langle \delta_i f,\, g \rangle = -\langle f,\, \tilde{\delta}_i g \rangle,$$

which is a discrete version of the integration by part formula

$$\int_0^1 f' g = -\int_0^1 f g'$$

for smooth periodic functions on $[0,1]$.

A divergence operator is defined using backward differences,

$$\mathrm{div}(v)_n = \tilde{\delta}_1 v_{1,n} + \tilde{\delta}_2 v_{2,n},$$

and corresponds to a mapping from vector fields to images

$$\mathrm{div} : \mathbb{R}^{N\times 2} \longrightarrow \mathbb{R}^N.$$

It is related to the dual of the gradient

$$\mathrm{div} = -\nabla^*$$

which means that

$$\forall\, f \in \mathbb{R}^N,\ \forall\, v \in \mathbb{R}^{N\ \times 2},\quad \langle \nabla f,\, v\rangle_{\mathbb{R}^{N\times 2}} = -\langle f,\, \mathrm{div}(v)\rangle_{\mathbb{R}^N}$$

which corresponds to a discrete version of the divergence theorem.

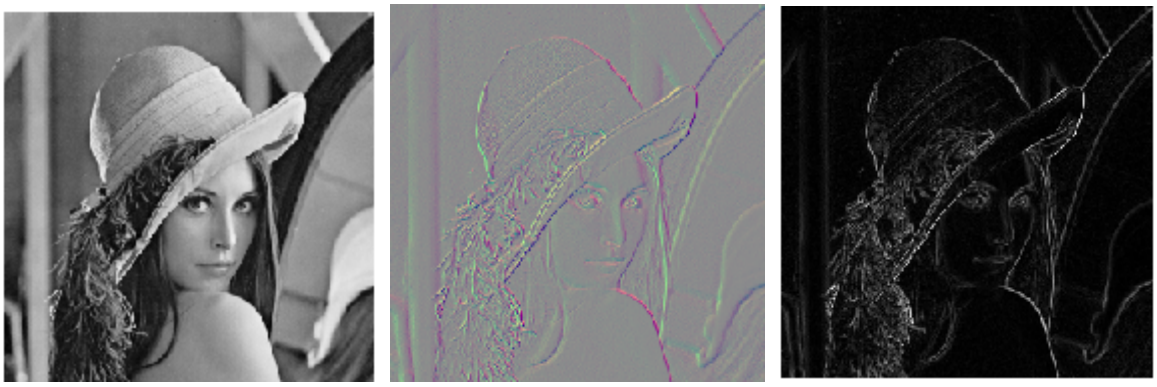


Image f $\quad$ ∇f $\quad$ $\|\nabla f\|$

Figure 9.2: Discrete operators.

Discrete laplacian. A general definition of a Laplacian is

$$\Delta f = \mathrm{div}(\nabla f),$$

which corresponds to a semi-definite negative operator.

For discrete images, and using the previously defined gradient and divergence, it is a local high pass filter

$$\Delta f_n = \sum_{p\in V_4(n)} f_p - 4 f_n, \tag{9.3}$$

that approximates the continuous second order derivative

$$\frac{\partial^2 f}{\partial x_1^2}(x) + \frac{\partial^2 f}{\partial x_2^2} \approx N^2 \Delta f_n \quad \text{for} \quad x = n/N.$$

Lapalacian operators thus correspond to filterings. A continuous Laplacian is equivalently defined over the Fourier domain in diagonal form as

$$g = \Delta f \quad \Longrightarrow \quad \hat{g}(\omega) = \|\omega\|^2 \hat{f}(\omega)$$

and the discrete Laplacian (9.3) as

$$g = \Delta f \quad \Longrightarrow \quad \hat{g}_\omega = \rho_\omega^2 \hat{f}(\omega) \quad \text{where} \quad \rho_\omega^2 = \sin\left(\frac{\pi}{N}\omega_1\right)^2 + \sin\left(\frac{\pi}{N}\omega_2\right)^2. \tag{9.4}$$

Discrete energies. A discrete Sobolev energy is obtained by using the ℓ^2 norm of the discrete gradient vector field

$$J_{\mathrm{Sob}}(f) = \frac{1}{2}\sum_n (\delta_1 f_n)^2 + (\delta_2 f_n)^2 = \frac{1}{2}\|\nabla f\|^2. \tag{9.5}$$

Similarly, a discrete TV energy is defined as the ℓ^1 norm of the gradient field

$$J_{\mathrm{TV}}(f) = \sum_n \sqrt{(\delta_1 f_n)^2 + (\delta_2 f_n)^2} = \|\nabla f\|_1 \tag{9.6}$$

where the ℓ^1 norm of a vector field $v \in \mathbb{R}^{N\times 2}$ is

$$\|v\|_1 = \sum_n \|v_n\| \tag{9.7}$$

where $v_n \in \mathbb{R}^2$.

9.2 PDE and Energy Minimization

Image smoothing is obtained by minimizing the prior using a gradient descent.

9.2.1 General Flows

The gradient of the prior $J : \mathbb{R}^N \to \mathbb{R}$ is a vector $\operatorname{grad} J(f)$. It describes locally up to the first order the variation of the prior

$$J(f+\varepsilon) = J(f) + \langle \varepsilon, \operatorname{grad} J(f)\rangle + o(\|\varepsilon\|).$$

If J is a smooth function of the image f, a discrete energy minimization is obtained through a gradient descent

$$f^{(k+1)} = f^{(k)} - \tau \operatorname{grad} J(f^{(k)}), \tag{9.8}$$

where the step size τ must be small enough to guarantee convergence.

For infinitesimal step size τ, one replaces the discrete parameter k by a continuous time, and the flow

$$t > 0 \longmapsto f_t \in \mathbb{R}^N$$

solves the following partial differential equation

$$\frac{\partial f_t}{\partial t} = -\operatorname{grad} J(f_t) \quad \text{and} \quad f_0 = f. \tag{9.9}$$

The gradient descent can be seen as an explicit discretization in time of this PDE at times $t_k = k\tau$.

9.2.2 Heat Flow

The heat flow corresponds to the instantiation of the generic PDE (9.9) to the case of the Sobolev energies $J_{\mathrm{Sob}}(f)$ defined for continuous function in (9.1) and for discrete images in (9.5).

Since it is a quadratic energy, its gradient is easily computed

$$J(f+\varepsilon) = \frac{1}{2}\|\nabla f + \nabla \varepsilon\|^2 = J(f) - \langle \Delta f, \varepsilon\rangle + o(\|\varepsilon\|^2),$$

so that

$$\operatorname{grad} J_{\mathrm{Sob}}(f) = -\Delta f.$$

Figure 9.4, left, shows an example of Laplacian. It is typically large (positive or negative) near edges.

The heat flow is thus

$$\frac{\partial f_t}{\partial t}(x) = -(\operatorname{grad} J(f_t))(x) = \Delta f_t(x) \quad \text{and} \quad f_0 = f. \tag{9.10}$$

Figure 9.3: Display of f_t for increasing time t for heat flow (top row) and TV flow (bottom row).

Continuous in space. For continuous images and an unbounded domain $\mathbb{R}^2$, the PDE (9.10) has an explicit solution as a convolution with a Gaussian kernel of increasing variance as time increases

$$f_t = f \star h_t \quad \text{where} \quad h_t(x) = \frac{1}{4\pi t} e^{-\frac{-\|x\|^2}{4t}}. \tag{9.11}$$

This shows the regularizing property of the heat flow, that operates a blurring to make the image more regular as time evolves.

Discrete in space. The discrete Sobolev energy (9.5) minimization defined a PDE flow that is discrete in space

$$\frac{\partial f_{n,t}}{\partial t} = -(\operatorname{grad} J(f_t))_n = (\Delta f_t)_n.$$

It can be further discretized in time as done in (9.8) and leads to a fully discrete flow

$$f_n^{(k+1)} = f_n^{(k)} + \tau\Big(\sum_{p \in V_4(n)} f_p - 4 f_n \Big) = (f \star h)_n$$

where $V_4(n)$ are the four neighbor to a pixel n. The flow thus corresponds to iterative convolutions

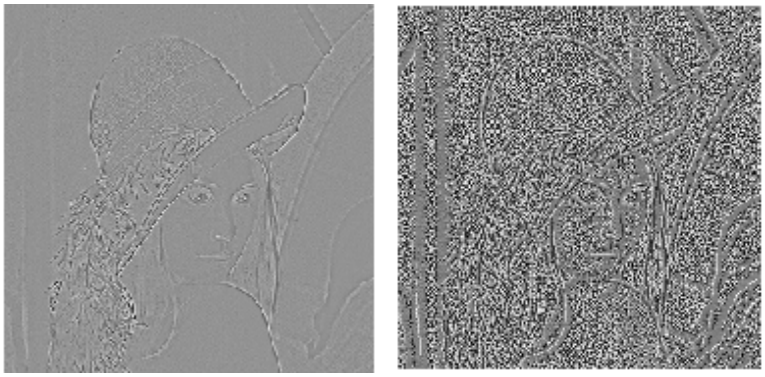
$$f^{(k)} = f \star h \star \ldots \star h = f \star^k h.$$

where h is a discrete filter.

It can be shown to be stable and convergent if $\tau < 1/4$.

9.2.3 Total Variation Flows

Total variation gradient. The total variation energy J_{TV}, both continuous (9.2) and discrete (9.6) is not a smooth function of the image. For instance, the discrete J_{TV} is non-differentiable at an image f such that there exists a pixel n where $\nabla f_n = 0$.



$\mathrm{div}(\nabla f)$ $\mathrm{div}(\nabla f/\|\nabla f\|)$

Figure 9.4: Discrete Laplacian and discrete TV gradient.

If $\nabla f_n \neq 0$, one can compute the gradient of the TV energy specialized at pixel n as

$$(\mathrm{grad}\, J(f))_n = -\,\mathrm{div}\left(\frac{\nabla f}{\|\nabla f\|}\right)_n$$

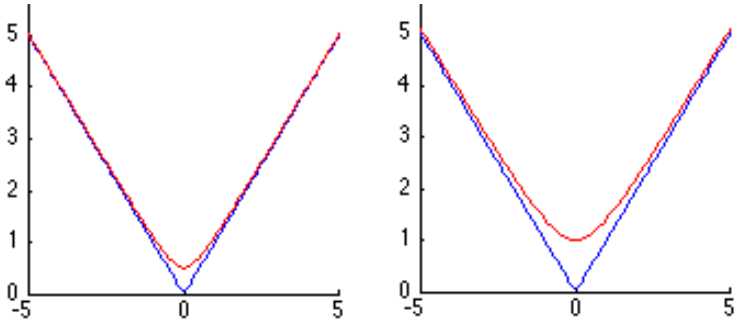
which exhibits a division by zero singularity for a point with vanishing gradient. Figure 9.4 shows an example of TV gradient, which appears noisy in smooth areas, because $\|\nabla f_n\|$ is small in such regions.

This non-differentiabilty makes impossible the definition of a gradient descent and a TV flow.

Regularized total variation. To avoid this issue, one can modify the TV energy, and define a smoothed TV prior

$$J^{\varepsilon}_{\mathrm{TV}}(f) = \sum_n \sqrt{\varepsilon^2 + \|\nabla f_n\|^2} \tag{9.12}$$

where $\varepsilon > 0$ is a small regularization parameter. Figure 9.5 shows this effect of this regularization on the absolute value function.



$\varepsilon = 0.1$ $\varepsilon = 0.01$

Figure 9.5: Regularized absolute value $x \mapsto \sqrt{x^2 + \varepsilon^2}$.

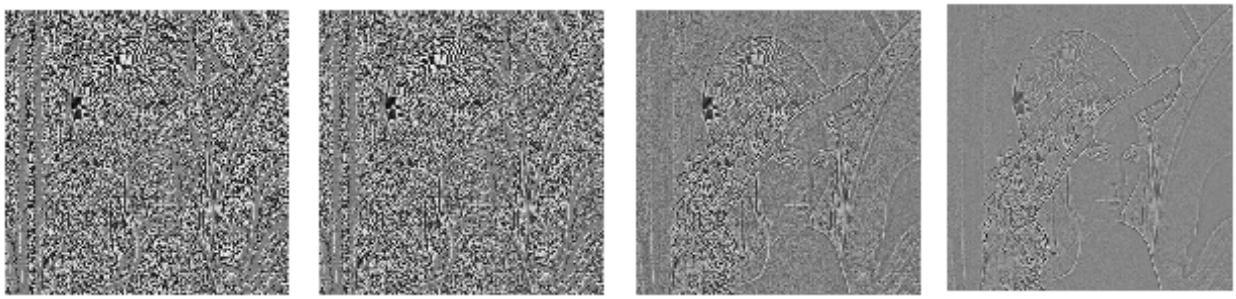
This smoothed TV energy is a differentiable function of the image, and its gradient is

$$\mathrm{grad}\, J^{\varepsilon}_{\mathrm{TV}}(f) = -\,\mathrm{div}\left(\frac{\nabla f}{\sqrt{\varepsilon^2 + \|\nabla f\|^2}}\right). \tag{9.13}$$

One can see that this smoothing interpolate between TV and Sobolev, as

$$\operatorname{grad}_f^\varepsilon \sim -\Delta/\varepsilon \quad \text{when} \quad \varepsilon \to +\infty.$$

Figure 9.6 shows the evolution of this gradient for several value of the smoothing parameter.



$\varepsilon = 0.01$ $\varepsilon = 0.05$ $\varepsilon = 0.1$ $\varepsilon = 0.5$

Figure 9.6: Regularized gradient norm $\sqrt{\|\nabla f(x)\|^2 + \varepsilon^2}$.

Regularized total variation flow. The smoothed total variation flow is then defined as

$$\frac{\partial f_t}{\partial t} = \operatorname{div}\left(\frac{\nabla f_t}{\sqrt{\varepsilon^2 + \|\nabla f_t\|^2}}\right). \tag{9.14}$$

Choosing a small ε makes the flow closer to a minimization of the total variation, but makes the computation unstable.

In practice, the flow is computed with a discrete gradient descent (9.8). For the smoothed total variation flow to converge, one needs to impose that $\tau < \varepsilon/4$, which shows that being more faithful to the TV energy requires smaller time steps and thus slower algorithms.

Figure 9.3 shows a comparison between the heat flow and the total variation flow for a small value of ε. This shows that the TV flow smooth less the edges than heat diffusion, which is consistent with the ability of the TV energy to better characterize sharp edges.

9.2.4 PDE Flows for Denoising

PDE flows can be used to remove noise from an observation $f = f_0 + w$. As detailed in Section 8.1.2 a simple noise model assumes that each pixel is corrupted with a Gaussian noise $w_n \sim \mathcal{N}(0, \sigma)$, and that these perturbations are independent (white noise).

The denoising is obtained using the PDE flow within initial image f at time $t = 0$

$$\frac{\partial f_t}{\partial t} = -\operatorname{grad}_{f_t} J \quad \text{and} \quad f_{t=0} = f.$$

An estimator $\tilde{f} = f_{t_0}$ is obtained for a well chose $t = t_0$. Figure 9.7 shows examples of Sobolev and TV flows for denoising.

Since f_t converges to a constant image when $t \to +\infty$, the choice of t_0 corresponds to a tradeoff between removing enough noise and not smoothing too much the edges in the image. This is usually a difficult task. During simulation, if one has access to the clean image f_0, one can monitor the denoising error $\|f_0 - f_t\|$ and choose the $t = t_0$ that minimizes this error. Figure 9.8, top row, shows an example of this oracle estimation of the best stopping time.

Figure 9.7: Denoising using f_t displayed for various time t for Sobolev (top) and TV (bottom) flows.

9.3 Regularization for Denoising

Instead of performing a gradient descent flow for denoising as detailed in Section 9.2.4 and select a stopping time, one can formulate an optimization problem. The estimator is then defined as a solution of this optimization. This setup has the advantage as being well defined mathematically even for non-smooth priors such as the TV prior J_{TV} or the sparsity prior J_1. Furthermore, this regularization framework is also useful to solve general inverse problems as detailed in Chapter **??**.

9.3.1 Regularization

Given some noisy image $f = f_0 + w$ of N pixels and a prior J, we consider the convex optimization problem

$$f_\lambda^\star \in \operatorname*{argmin}_{g \in \mathbb{R}^N} \frac{1}{2}\|f - g\|^2 + \lambda J(g), \tag{9.15}$$

where $\lambda > 0$ is a Lagrange multiplier parameter that weights the influence of the data fitting term $\|f - g\|^2$ and the regularization term $J(g)$.

If one has at his disposal a clean original image f_0, one can minimize the denoising error $\|f_\lambda^\star - f_0\|$, but it is rarely the case. In practice, this parameter should be adapted to the noise level and to the regularity of the unknown image f_0, which might be a non trivial task.

We note that since we did not impose J to be convex, the problem (9.15) might have several optimal solutions.

An estimator is thus defined as

$$\tilde{f} = f_\lambda^\star$$

for a well chosen λ.

If J is differentiable and convex, one can compute the solution of (9.15) through a gradient descent

$$f^{(k+1)} = f^{(k)} - \tau \left(f^{(k)} - \lambda \operatorname{grad} J(f^{(k)}) \right) \tag{9.16}$$

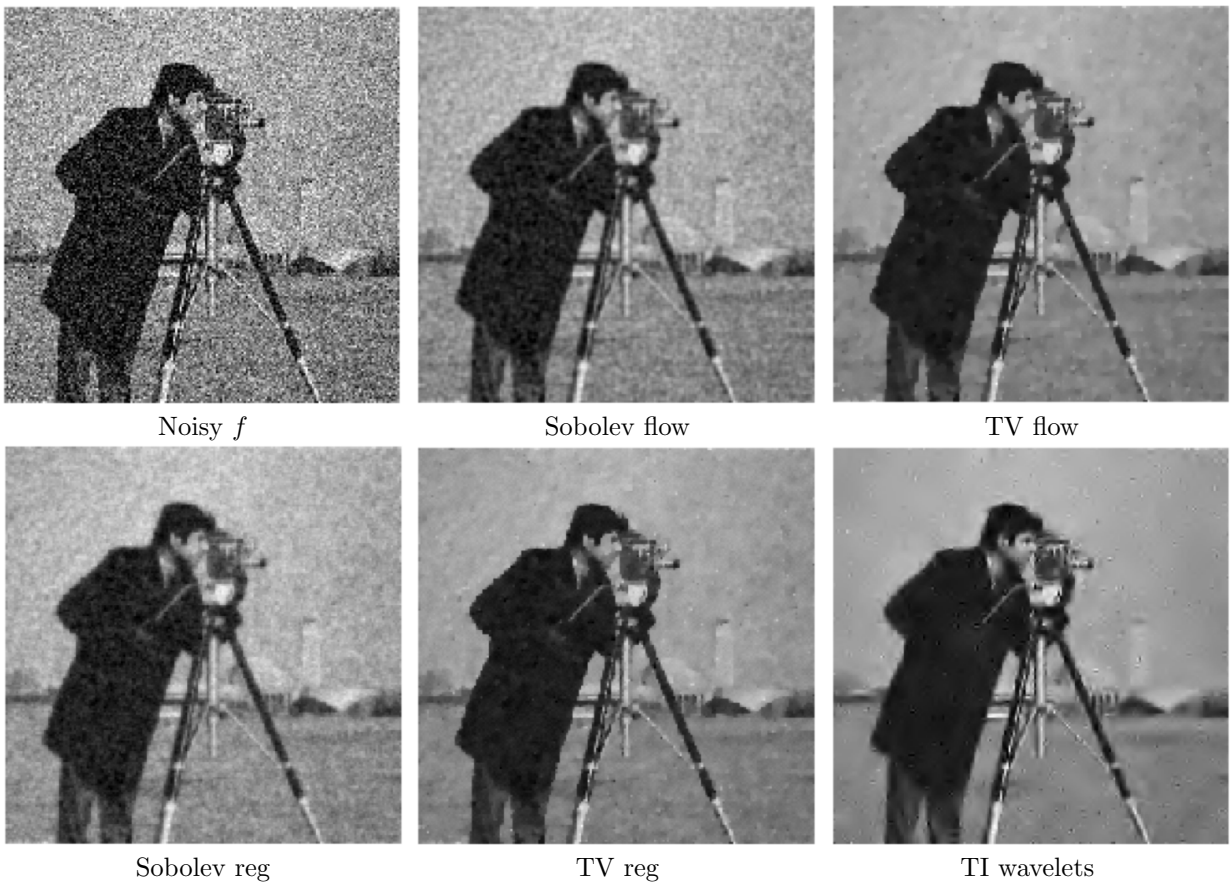


Figure 9.8: Denoising using PDE flows and regularization.

where the descent step size $\tau > 0$ should be small enough. This gradient descent is similar to the time-discretized minimization flow (9.8), excepted that the data fitting term prevent the flow to converge to a constant image.

Unfortunately, priors such as the total variation J_{TV} or the sparsity J_1 are non-differentiable. Some priors such as the ideal sparsity J_0 might even be non-convex. This makes the simple gradient descent not usable to solve for (9.15). In the following Section we show how to compute $f_\lambda^\star$ for several priors.

9.3.2 Sobolev Regularization

The discrete Sobolev prior defined in (9.5) is differentiable, and the gradient descent (9.16) reads

$$f^{(k+1)} = (1-\tau) f^{(k)} + \tau f - \tau\lambda\Delta J(f^{(k)}.$$

Since $J(f) = \|\nabla f\|^2$ is quadratic, one can use a conjugate gradient descent, which converges faster.

The solution $f_\lambda^\star$ can be computed in closed form as the solution of a linear system

$$f_\lambda^\star = (\mathrm{Id}_N - \lambda\Delta)^{-1} f,$$

which shows that the regularization (9.15) is computing an estimator that depends linearly on the observations f.

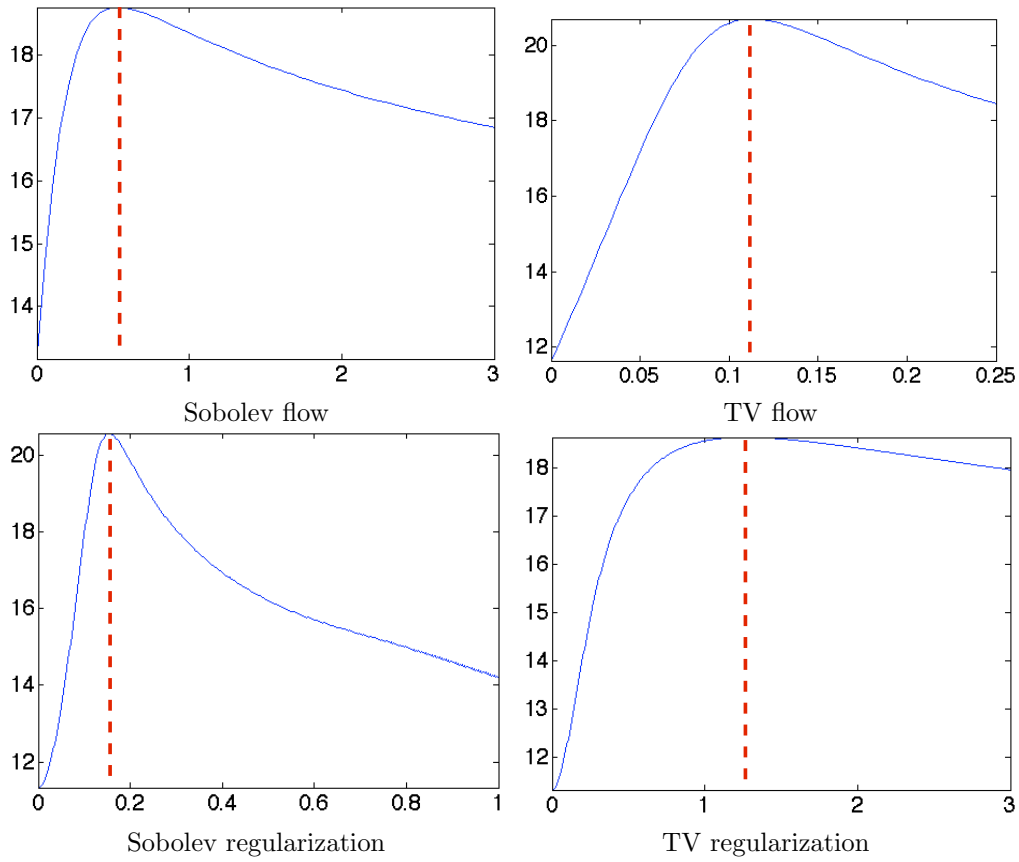


Figure 9.9: SNR as a function of time t for flows (top) and λ for regularization (bottom).

If the differential operators are computed with periodic boundary conditions, this linear system can be solved exactly over the Fourier domain

$$(\hat{f}^\star_\lambda)_\omega = \frac{1}{1+\lambda\rho_\omega^2}\hat{f}_\omega \tag{9.17}$$

where ρ_ω depends on the discretization of the Laplacian, see for instance (9.4).

Equation (9.17) shows that denoising using Sobolev regularization corresponds to a low pass filtering, whose strength is controlled by λ. This should be related to the solution (9.11) of the heat equation, which also corresponds to a filtering, but using a Gaussian low pass kernel parameterized by its variance t^2.

This Sobolev regularization denoising is a particular case of the linear estimator considered in Section 8.2. The selection of the parameter λ is related to the selection of an optimal filter as considered in Section 8.2.2, but with the restriction that the filter is computed in a parametric family.

9.3.3 TV Regularization

The total variation prior J_{TV} defined in (9.6) is non-differentiable. One can either use a smoothed approximation of the prior, or use an optimization algorithm not based on a gradient descent.

The TV prior can be approximated to obtain the prior $J^\varepsilon_{\mathrm{TV}}(g)$ defined in (9.12), which is differentiable with respect to g. Using the gradient of this prior defined in (9.13), one obtains the following instantiation

of the gradient descent (9.16)

$$f^{(k+1)} = (1-\tau) f^{(k)} + \tau f + \lambda \tau \operatorname{div}\left(\frac{\nabla f_t}{\sqrt{\varepsilon^2 + \|\nabla f_t\|^2}} \right). \tag{9.18}$$

which converge to the unique minimizer $f_\lambda^\star$ of (9.15).

Section 13.1 details a better alternative which does not require introducing this ε smoothing.

Chapter 10

Inverse Problems

The main references for this chapter are [29, 37, 20].

10.1 Inverse Problems Regularization

Increasing the resolution of signals and images requires to solve an ill posed inverse problem. This corresponds to inverting a linear measurement operator that reduces the resolution of the image. This chapter makes use of convex regularization introduced in Chapter **??** to stabilize this inverse problem.

We consider a (usually) continuous linear map $\Phi : \mathcal{S} \to \mathcal{H}$ where $\mathcal{S}$ can be an Hilbert or a more general Banach space. This operator is intended to capture the hardware acquisition process, which maps a high resolution unknown signal $f_0 \in \mathcal{S}$ to a noisy low-resolution obervation

$$y = \Phi f_0 + w \in \mathcal{H}$$

where $w \in \mathcal{H}$ models the acquisition noise. In this section, we do not use a random noise model, and simply assume that $\|w\|_{\mathcal{H}}$ is bounded.

In most applications, $\mathcal{H} = \mathbb{R}^P$ is finite dimensional, because the hardware involved in the acquisition can only record a finite (and often small) number P of observations. Furthermore, in order to implement numerically a recovery process on a computer, it also makes sense to restrict the attention to $\mathcal{S} = \mathbb{R}^N$, where N is number of point on the discretization grid, and is usually very large, $N \gg P$. However, in order to perform a mathematical analysis of the recovery process, and to be able to introduce meaningful models on the unknown f_0, it still makes sense to consider infinite dimensional functional space (especially for the data space $\mathcal{S}$).

The difficulty of this problem is that the direct inversion of Φ is in general impossible or not advisable because Φ^{-1} have a large norm or is even discontinuous. This is further increased by the addition of some measurement noise w, so that the relation $\Phi^{-1}y = f_0 + \Phi^{-1}w$ would leads to an explosion of the noise $\Phi^{-1}w$.

We now gives a few representative examples of forward operators Φ.

Denoising. The case of the identity operator $\Phi = \mathrm{Id}_{\mathcal{S}}$, $\mathcal{S} = \mathcal{H}$ corresponds to the classical denoising problem, already treated in Chapters **??** and **??**.

De-blurring and super-resolution. For a general operator Φ, the recovery of f_0 is more challenging, and this requires to perform both an inversion and a denoising. For many problem, this two goals are in contradiction, since usually inverting the operator increases the noise level. This is for instance the case for the deblurring problem, where Φ is a translation invariant operator, that corresponds to a low pass filtering with some kernel h

$$\Phi f = f \star h. \tag{10.1}$$

One can for instance consider this convolution over $\mathcal{S} = \mathcal{H} = L^2(\mathbb{T}^d)$, see Proposition 3. In practice, this convolution is followed by a sampling on a grid $\Phi f = \{(f \star h)(x_k) \; ; \; 0 \leqslant k < P\}$, see Figure 10.1, middle, for an example of a low resolution image Φf_0. Inverting such operator has important industrial application to upsample the content of digital photos and to compute high definition videos from low definition videos.

Interpolation and inpainting. Inpainting corresponds to interpolating missing pixels in an image. This is modelled by a diagonal operator over the spacial domain

$$(\Phi f)(x) = \begin{cases} 0 & \text{if} \quad x \in \Omega, \\ f(x) & \text{if} \quad x \notin \Omega. \end{cases} \tag{10.2}$$

where $\Omega \subset [0,1]^d$ (continuous model) or $\{0, \ldots, N-1\}$ which is then a set of missing pixels. Figure 10.1, right, shows an example of damaged image Φf_0.

Original f_0 Low resolution Φf_0 Masked Φf_0

Figure 10.1: Example of inverse problem operators.

Medical imaging. Most medical imaging acquisition device only gives indirect access to the signal of interest, and is usually well approximated by such a linear operator Φ. In scanners, the acquisition operator is the Radon transform, which, thanks to the Fourier slice theorem, is equivalent to partial Fourier mesurments along radial lines. Medical resonance imaging (MRI) is also equivalent to partial Fourier measures

$$\Phi f = \left\{ \hat{f}(x) \; ; \; x \in \Omega \right\}. \tag{10.3}$$

Here, Ω is a set of radial line for a scanner, and smooth curves (e.g. spirals) for MRI.

Other indirect application are obtained by electric or magnetic fields measurements of the brain activity (corresponding to MEG/EEG). Denoting $\Omega \subset \mathbb{R}^3$ the region around which measurements are performed (e.g. the head), in a crude approximation of these measurements, one can assume $\Phi f = \{(\psi \star f)(x) \; ; \; x \in \partial\Omega\}$ where $\psi(x)$ is a kernel accounting for the decay of the electric or magnetic field, e.g. $\psi(x) = 1/\|x\|^2$.

10.2 Theoretical Study of Quadratic Regularization

We now give a glimpse on the typical approach to obtain theoretical guarantee on recovery quality in the case of Hilbert space. The goal is not to be exhaustive, but rather to insist on the modelling hypotethese, namely smoothness implies a so called "source condition", and the inherent limitations of quadratic methods (namely slow rates and the impossibility to recover information in $\ker(\Phi)$, i.e. to achieve super-resolution).

10.2.1 Singular Value Decomposition

Finite dimension. Let us start by the simple finite dimensional case $\Phi \in \mathbb{R}^{P\times N}$ so that $\mathcal{S} = \mathbb{R}^N$ and $\mathcal{H} = \mathbb{R}^P$ are Hilbert spaces. In this case, the Singular Value Decomposition (SVD) is the key to analyze the operator very precisely, and to describe linear inversion process.

Proposition 20 (SVD). *There exists $(U, V) \in \mathbb{R}^{N\times R} \times \mathbb{R}^{P\times R}$, where $R = \mathrm{rank}(\Phi) = \dim(\mathrm{Im}(\Phi))$, with $U^\top U = V^\top V = \mathrm{Id}_R$, i.e. having orthogonal columns $(u_m)_{m=1}^R \subset \mathbb{R}^N, (v_m)_{m=1}^R \subset \mathbb{R}^P$, and $(\sigma_m)_{m=1}^R$ with $\sigma_m > 0$, such that*

$$\Phi = U \,\mathrm{diag}_m(\sigma_m) V^\top = \sum_{m=1}^R \sigma_m u_m v_m^\top. \tag{10.4}$$

Proof. We first analyze the problem, and notice that if $\Phi = U\Sigma V^\top$ with $\Sigma = \mathrm{diag}_m(\sigma_m)$, then $\Phi\Phi^\top = U\Sigma^2 U^\top$ and then $V^\top = \Sigma^{-1}U^\top\Phi$. We can use this insight. Since $\Phi\Phi^\top$ is a positive symmetric matrix, we write its eigendecomposition as $\Phi\Phi^\top = U\Sigma^2U^\top$ where $\Sigma = \mathrm{diag}_{m=1}^R(\sigma_m)$ with $\sigma_m > 0$. We then define $V \overset{\text{def.}}{=} \Phi^\top U\Sigma^{-1}$. One then verifies that

$$V^\top V = (\Sigma^{-1}U^\top\Phi)(\Phi^\top U\Sigma^{-1}) = \Sigma^{-1}U^\top(U\Sigma^2U^\top)U\Sigma^{-1} = \mathrm{Id}_P \quad \text{and} \quad U\Sigma V^\top = U\Sigma\Sigma^{-1}U^\top\Phi = \Phi.$$

□

This theorem is still valid with complex matrice, replacing $^\top$ by *. Expression (10.4) describes Φ as a sum of rank-1 matrices $u_m v_m^\top$. One usually order the singular values $(\sigma_m)_m$ in decaying order $\sigma_1 \geqslant \ldots \geqslant \sigma_R$. If these values are different, then the SVD is unique up to ± 1 sign change on the singular vectors.

The left singular vectors U is an orthonormal basis of $\mathrm{Im}(\Phi)$, while the right singular values is an orthonormal basis of $\mathrm{Im}(\Phi^\top) = \ker(\Phi)^\perp$. The decomposition (10.4) is often call the "reduced" SVD because one has only kept the R non-zero singular values. The "full" SVD is obtained by completing U and V to define orthonormal bases of the full spaces $\mathbb{R}^P$ and $\mathbb{R}^N$. Then Σ becomes a rectangular matrix of size $P \times N$.

A typical example is for $\Phi f = f \star h$ over $\mathbb{R}^P = \mathbb{R}^N$, in which case the Fourier transform diagonalizes the convolution, i.e.

$$\Phi = (u_m)_m^* \,\mathrm{diag}(\hat{h}_m)(u_m)_m \tag{10.5}$$

where $(u_m)_n \overset{\text{def.}}{=} \frac{1}{\sqrt{N}} e^{\frac{2\mathrm{i}\pi}{N}nm}$ so that the singular values are $\sigma_m = |\hat{h}_m|$ (removing the zero values) and the singular vectors are $(u_m)_n$ and $(v_m\theta_m)_n$ where $\theta_m \overset{\text{def.}}{=} |\hat{h}_m|/\hat{h}_m$ is a unit complex number.

Computing the SVD of a full matrix $\Phi \in \mathbb{R}^{N\times N}$ has complexity N^3.

Compact operators. One can extend the decomposition to compact operators $\Phi : \mathcal{S} \to \mathcal{H}$ between separable Hilbert space. A compact operator is such that ΦB_1 is pre-compact where $B_1 = \{s \in \mathcal{S} \,;\, \|s\| \leqslant 1\}$ is the unit-ball. This means that for any sequence $(\Phi s_k)_k$ where $s_k \in B_1$ one can extract a converging sub-sequence. Note that in infinite dimension, the identity operator $\Phi : \mathcal{S} \to \mathcal{S}$ is never compact.

Compact operators Φ can be shown to be equivalently defined as those for which an expansion of the form (10.4) holds

$$\Phi = \sum_{m=1}^{+\infty} \sigma_m u_m v_m^\top \tag{10.6}$$

where $(\sigma_m)_m$ is a decaying sequence onverging to 0, $\sigma_m \to 0$. Here in (10.6) convergence holds in the operator norm, which is the algebra norm on linear operator inherited from those of $\mathcal{S}$ and $\mathcal{H}$

$$\|\Phi\|_{\mathcal{L}(\mathcal{S},\mathcal{H})} \overset{\text{def.}}{=} \min_{\|\Phi u\|_{\mathcal{H}}} \|u\|_{\mathcal{S}} \leqslant 1.$$

For Φ having an SVD decomposition (10.6), $\|\Phi\|_{\mathcal{L}(\mathcal{S},\mathcal{H})} = \sigma_1$.

When $\sigma_m = 0$ for $m > R$, Φ has a finite rank $R = \dim(\mathrm{Im}(\Phi))$. As we explain in the sections bellow, when using *linear* recovery methods (such as quadratic regularization), the inverse problem is equivalent to

a finite dimensional problem, since one can restrict its attention to functions in $\ker(\Phi)^\perp$ which as dimension R. Of course, this is not true anymore when one can retrieve function inside $\ker(\Phi)$, which is often referred to as a "super-resolution" effect of non-linear methods. Another definition of compact operator is that they are the limit of finite rank operator. They are thus in some sense the extension of finite dimensional matrices, and are the correct setting to model ill-posed inverse problems. This definition can be extended to linear operator between Banach spaces, but this conclusion does not holds.

Typical example of compact operator are matrix-like operator with a continuous kernel $k(x,y)$ for $(x,y) \in \Omega$ where Ω is a compact sub-set of $\mathbb{R}^d$ (or the torus $\mathbb{T}^d$), i.e.

$$\Phi f = \int_\Omega k(x,y) f(y) \mathrm{d}y$$

where $\mathrm{d}y$ is the Lebesgue measure. An example of such a setting which generalizes (10.5) is when $\Phi f = h \star h$ on $\mathbb{T}^d = (\mathbb{R}/2\pi\mathbb{Z})^d$, which is corresponds to a translation invariant kernel $k(x,y) = h(x-y)$, in which case $u_m(x) = (2\pi)^{-d/2} e^{\mathrm{i}\omega x}$, $\sigma_m = |\hat{f}_m|$.

Pseudo inverse. In the case where $w = 0$, it makes to try to directly solve $\Phi f = y$. The two obstruction for this is that one not necessarily has $y \in \mathrm{Im}(\Phi)$ and even so, there are an infinite number of solutions if $\ker(\Phi) \neq \{0\}$. The usual workaround is to solve this equation in the least square sense

$$f^+ \overset{\text{def.}}{=} \operatorname*{argmin}_{\Phi f = y^+} \|f\|_{\mathcal{S}} \quad \text{where} \quad y^+ = \mathrm{Proj}_{\mathrm{Im}(\Phi)}(y) = \operatorname*{argmin}_{z \in \mathrm{Im}(\Phi)} \|y - z\|_{\mathcal{H}}.$$

The following proposition shows how to compute this least square solution using the SVD and by solving linear systems involving either $\Phi\Phi^*$ or $\Phi^*\Phi$.

Proposition 21. *One has*

$$f^+ = \Phi^+ y \quad \textit{where} \quad \Phi^+ = V \,\mathrm{diag}_m(1/\sigma_m) U^*. \tag{10.7}$$

In case that $\mathrm{Im}(\Phi) = \mathcal{H}$*, one has* $\Phi^+ = \Phi^*(\Phi\Phi^*)^{-1}$*. In case that* $\ker(\Phi) = \{0\}$*, one has* $\Phi^+ = (\Phi^*\Phi)^{-1}\Phi^*$*.*

Proof. Since U is an ortho-basis of $\mathrm{Im}(\Phi)$, $y^+ = UU^*y$, and thus $\Phi f = y^+$ reads $U\Sigma V^* f = UU^* y$ and hence $V^* f = \Sigma^{-1} U^* y$. Decomposition orthogonally $f = f_0 + r$ where $f_0 \in \ker(\Phi)^\perp$ and $r \in \ker(\Phi)$, one has $f_0 = VV^* f = V\Sigma^{-1}U^* y = \Phi^+ y$ is a constant. Minimizing $\|f\|^2 = \|f_0\|^2 + \|r\|^2$ is thus equivalent to minimizing $\|r\|$ and hence $r = 0$ which is the desired result. If $\mathrm{Im}(\Phi) = \mathcal{H}$, then $R = N$ so that $\Phi\Phi^* = U\Sigma^2 U^*$ is the eigen-decomposition of an invertible and $(\Phi\Phi^*)^{-1} = U\Sigma^{-2}U^*$. One then verifies $\Phi^*(\Phi\Phi^*)^{-1} = V\Sigma U^* U \Sigma^{-2} U^*$ which is the desired result. One deals similarly with the second case. □

For convolution operators $\Phi f = f \star h$, then

$$\Phi^+ y = y \star h^+ \quad \text{where} \quad \hat{h}^+_m = \begin{cases} \hat{h}_m^{-1} & \text{if} \quad \hat{h}_m \neq 0 \\ 0 & \text{if} \quad \hat{h}_m = 0. \end{cases}.$$

10.2.2 Tikonov Regularization

Regularized inverse. When there is noise, using formula (10.7) is not acceptable, because then

$$\Phi^+ y = \Phi^+ \Phi f_0 + \Phi^+ w = f_0^+ + \Phi^+ w \quad \text{where} \quad f_0^+ \overset{\text{def.}}{=} \mathrm{Proj}_{\ker(\Phi)^\perp},$$

so that the recovery error is $\|\Phi^+ y - f_0^+\| = \|\Phi^+ w\| \geqslant \|w\|/\sigma_R$. The noise is thus amplified by the inverse $1/\sigma_R$ of the smallest amplitude non-zero singular values, which can be very large. In infinite dimension, one typically has $R = +\infty$, so that the inverse is actually not bounded (discontinuous). It is thus mendatory to replace Φ^+ by a regularized approximate inverse, which should have the form

$$\Phi^+_\lambda = V \,\mathrm{diag}_m(\mu_\lambda(\sigma_m)) U^* \tag{10.8}$$

where μ_λ, indexed by some parameter $\lambda > 0$, is a regularization of the inverse, that should typically satisfies

$$\mu_\lambda(\sigma) \leqslant C_\lambda < +\infty \quad \text{and} \quad \lim_{\lambda \to 0} \mu_\lambda(\sigma) = \frac{1}{\sigma}$$

Variational regularization. A typical example of such regularized inverse is obtained by considering a penalized least square involving a regularization functional

$$f_\lambda \overset{\text{def.}}{=} \operatorname*{argmin}_{f\in\mathcal{S}} \|y-\Phi f\|_{\mathcal{H}}^2 + \lambda J(f)^2 \tag{10.9}$$

where J is some regularization functional which should at least be continuous on $\mathcal{S}$. The simplest example is the quadratic norm $J=\|\cdot\|_{\mathcal{S}}^2$,

$$f_\lambda \overset{\text{def.}}{=} \operatorname*{argmin}_{f\in\mathcal{S}} \|y-\Phi f\|_{\mathcal{H}}^2 + \lambda \|f\|^2 \tag{10.10}$$

which can be conveniently rewritten in the basis of singular vectors as

$$f_\lambda = \operatorname*{argmin}_{f\in\mathrm{Im}(\Phi^*)} \sum_m (\sigma_m\langle f,\, v_m\rangle - \langle y,\, v_m\rangle)^2 + \lambda \langle f,\, v_m\rangle^2 \tag{10.11}$$

where we have use the fact that necessarily $f_\lambda \in \mathrm{Im}(\Phi^*)$ because of the square penalty. The minimization (10.11) boils down to independent scalar minimization over each coordinate $f_m \overset{\text{def.}}{=} \langle f,\, v_m\rangle$ and the first order condition reads

$$\sigma_m(\sigma_m f_m - y_m) + \lambda f_m = 0 \quad \text{where} \quad y_m \overset{\text{def.}}{=} \langle y,\, v_m\rangle.$$

One thus has $f_\lambda = \Phi_\lambda^+$ where Φ_λ^+ is in the form (10.8) for the specific choice of function

$$\forall\, \sigma\in\mathbb{R}, \quad \mu_\lambda(\sigma) = \frac{\sigma}{\sigma^2+\lambda}.$$

The question is to understand how to choose λ as a function of the noise level $\|w\|_{\mathcal{H}}$ in order to guarantees that $f_\lambda \to f_0$ and furthermore establish convergence speed. One first needs to ensure at least $f_0 = f_0^+$, which in turns requires that $f_0 \in \mathrm{Im}(\Phi^*) = \ker(\Phi)^\perp$. Indeed, an important drawback of linear recovery methods (such as quadratic regularization) is that necessarily $f_\lambda \in \mathrm{Im}(\Phi^*) = \ker(\Phi^\perp)$ so that no information can be recovered inside $\ker(\Phi)$. Non-linear methods must be used to achieve a "super-resoltution" effect and recover this missing information.

Source condition. In order to ensure convergence speed, one quantify this condition and impose a so-called source condition of order β, which reads

$$f_0 \in \mathrm{Im}((\Phi^*\Phi)^\beta) = \mathrm{Im}(V\,\mathrm{diag}(\sigma_m^{2\beta})V^*).$$

This condition means that there should exists $z\in\mathcal{S}$ such that $f_0 = V\,\mathrm{diag}(\sigma_m^{2\beta})V^* z$, i.e. $z = V\,\mathrm{diag}(\sigma_m^{-2\beta})V^* f_0$. Denoting $\rho \overset{\text{def.}}{=} \|z\|$, we thus in fact impose the following constraint

$$\sum_m \sigma_m^{-2\beta}\langle f,\, v_m\rangle^2 \leqslant \rho^2 < +\infty. \tag{$S_{\beta,\rho}$}$$

This is a Sobolev-type constraint, similar to those imposed in 8.4. A prototypical example is for a low-pass filter $\Phi f = f\star h$ where h as a slow polynomial-like decay of frequency, i.e. $|\hat{h}_m| \sim 1/m^\alpha$ for large m. In this case, since v_m is the Fourier basis, the source condition ($S_{\beta,\rho}$) reads

$$\sum_m \|m\|^{2\alpha\beta}|\hat{f}_m|^2 \leqslant \rho^2 < +\infty. \tag{$S_{\beta,\rho}$}$$

which is a Sobolev ball of radius ρ and differential order $\alpha\beta$.

Sublinear convergence speed. The following theorem shows that this source condition leads to a convergence speed of the regularization.

Theorem 31. *Assuming the source condition* ($S_{\beta,\rho}$) *with* $0 < \beta \leqslant 2$, *then the solution of* (10.10) *for* $\|w\| \leqslant \delta$ *satisfies*

$$\|f_\lambda - f_0\| \leqslant C\rho^{\frac{1}{\beta+1}}\delta^{\frac{\beta}{\beta+1}}$$

for a constant C *which depends only on* β, *and for a choice*

$$\lambda \sim \delta^{\frac{2}{\beta+1}}\rho^{-\frac{2}{\beta+1}}.$$

Proof. Because of the source condition, $f_0 \in \mathrm{Im}(\Phi^*)$. Denoting

$$f_\lambda^0 \overset{\text{def.}}{=} \Phi_\lambda^+(\Phi f_0)$$

so that $f_\lambda = f_\lambda^0 + \Phi_\lambda^+ w$, one has for any regularized inverse of the form (10.8)

$$\|f_\lambda - f_0\| \leqslant \|f_\lambda - f_\lambda^0\| + \|f_\lambda^0 - f_0\|. \tag{10.12}$$

The term $\|f_\lambda - f_\lambda^0\|$ is a variance term which account for residual noise, and thus decays when λ increases (more regularization). The term $\|f_\lambda^0 - f_0\|$ is independent of the noise, it is a bias term coming from the approximartion (smoothing) of f_0, and thus increases when λ increases. The choice of an optimal λ thus results in a bias-variance tradeoff between these two terms. Assuming

$$\forall\, \sigma \geqslant 0, \quad \mu_\lambda(\sigma) \leqslant C_\lambda$$

the variance terme is bounded as

$$\|f_\lambda - f_\lambda^0\|^2 = \|\Phi_\lambda^+ w\|^2 = \sum_m \mu_\lambda(\sigma_m)^2 w_m^2 \leqslant C_\lambda^2 \|w\|_{\mathcal{H}}^2.$$

The bias term is bounded as

$$\|f_\lambda^0 - f_0\|^2 = \sum_m (1 - \mu_\lambda(\sigma_m)\sigma_m)^2 f_{0,m}^2 = \sum_m \left(\frac{1 - \mu_\lambda(\sigma_m)\sigma_m}{\sigma^\beta}\right)^2 \sigma^{2\beta} f_{0,m}^2 \leqslant D_{\lambda,\beta}^2 \rho^2 \tag{10.13}$$

where we assumed

$$\forall\, \sigma \geqslant 0, \quad \left|\frac{1 - \mu_\lambda(\sigma)\sigma}{\sigma^\beta}\right| \leqslant D_{\lambda,\beta}. \tag{10.14}$$

Putting (10.13) and (10.14) together, one obtains

$$\|f_\lambda - f_0\| \leqslant C_\lambda \delta + D_{\lambda,\beta}\rho. \tag{10.15}$$

In the case of the regularization (10.10), one has $\mu_\lambda(\sigma) = \frac{\sigma}{\sigma^2+\lambda}$, and thus $\frac{1-\mu_\lambda(\sigma)\sigma}{\sigma^\beta} = \frac{\lambda\sigma^\beta}{\sigma^2+\lambda}$. For $\beta \leqslant 2$, one verifies that

$$C_\lambda = \frac{1}{2\sqrt{\lambda}} \quad \text{and} \quad D_{\lambda,\beta} = C_\beta \lambda^{\frac{\beta}{2}},$$

for some constant C_β. Equalizing the contributions of the two terms in (10.15) (a better constant would be reached by finding the best λ) leads to selecting $\frac{\delta}{\sqrt{\lambda}} = \lambda^{\frac{\beta}{2}}\rho$ i.e. $\lambda = (\delta/\rho)^{\frac{2}{\beta+1}}$. With this choice,

$$\|f_\lambda - f_0\| = O(\delta/\sqrt{\lambda}) = O(\delta(\delta/\rho)^{-\frac{1}{\beta+1}}) = O(\delta^{\frac{\beta}{\beta+1}}\rho^{\frac{1}{\beta+1}}).$$

□

This theorem shows that using larger $\beta \leqslant 2$ leads to faster convergence rates as $\|w\|$ drops to zero. The rate (10.12) however suffers from a "saturation" effect, indeed, choosing $\beta > 2$ does not helps (it gives the same rate as $\beta = 2$), and the best possible rate is thus

$$\|f_\lambda - f_0\| = O(\rho^{\frac{1}{3}}\delta^{\frac{2}{3}}).$$

By choosing more alternative regularization functional μ_λ and choosing β large enough, one can show that it is possible to reach rate $\|f_\lambda - f_0\| = O(\delta^{1-\kappa})$ for an arbitrary small $\kappa > 0$, but one cannot reach a linear rate $\|f_\lambda - f_0\| = O(\|w\|)$. Such rates are achievable using non-linear sparse ℓ^1 regularizations as detailed in Chapter 11.

10.3 Quadratic Regularization

After this theoretical study in infinite dimension, we now turn our attention to more practical matters, and focus only on the finite dimensional setting.

Convex regularization. Following (10.9), the ill-posed problem of recovering an approximation of the high resolution image $f_0 \in \mathbb{R}^N$ from noisy measures $y = \Phi f_0 + w \in \mathbb{R}^P$ is regularized by solving a convex optimization problem

$$f^\star \in \operatorname*{argmin}_{f \in \mathbb{R}^N} \mathcal{E}(f) \overset{\text{def.}}{=} \frac{1}{2}\|y - \Phi f\|^2 + \lambda J(f) \tag{10.16}$$

where $\|y - \Phi f\|^2$ is the data fitting term (here $\|\cdot\|$ is the ℓ^2 norm on $\mathbb{R}^P$) and $J(f)$ is a convex functional on $\mathbb{R}^N$.

The Lagrange multiplier λ weights the importance of these two terms, and is in practice difficult to set. Simulation can be performed on high resolution signal f_0 to calibrate the multiplier by minimizing the super-resolution error $\|f_0 - \tilde{f}\|$, but this is usually difficult to do on real life problems.

In the case where there is no noise, $w = 0$, the Lagrange multiplier λ should be set as small as possible. In the limit where $\lambda \to 0$, the unconstrained optimization problem (10.16) becomes a constrained optimization

$$f^\star = \operatorname*{argmin}_{f \in \mathbb{R}^N} \{J(f)\ ;\ \Phi f = y\}. \tag{10.17}$$

Quadratic Regularization. The simplest class of prior functional are quadratic, and can be written as

$$J(f) = \frac{1}{2}\|Gf\|^2_{\mathbb{R}^K} = \frac{1}{2}\langle Lf,\, f\rangle_{\mathbb{R}^N} \tag{10.18}$$

where $G \in \mathbb{R}^{K \times N}$ and where $L = G^*G \in \mathbb{R}^{N \times N}$ is a positive semi-definite matrix. The special case (10.10) is recovered when setting $G = L = \mathrm{Id}_N$.

Writing down the first order optimality conditions for (10.16) leads to

$$\nabla \mathcal{E}(f) = \Phi^*(\Phi f - y) + \lambda L f = 0,$$

hence, if

$$\ker(\Phi) \cap \ker(G) = \{0\},$$

then (10.18) has a unique minimizer f_λ, which is obtained by solving a linear system

$$f_\lambda = (\Phi^*\Phi + \lambda L)^{-1}\Phi^* y. \tag{10.19}$$

In the special case where L is diagonalized by the singular basis $(v_m)_m$ of Φ, i.e. $L = V\,\mathrm{diag}(\alpha_m^2)V^*$, then f_λ reads in this basis

$$\langle f_\lambda,\, v_m\rangle = \frac{\sigma_m}{\sigma_m^2 + \lambda\alpha_m^2}\langle y,\, v_m\rangle. \tag{10.20}$$

Example of convolution. A specific example is for convolution operator

$$\Phi f = h \star f, \tag{10.21}$$

and using $G = \nabla$ be a discretization of the gradient operator, such as for instance using first order finite differences (2.16). This corresponds to the discrete Sobolev prior introduced in Section 9.1.2. Such an operator compute, for a d-dimensional signal $f \in \mathbb{R}^N$ (for instance a 1-D signal for $d = 1$ or an image when $d = 2$), an approximation $\nabla f_n \in \mathbb{R}^d$ of the gradient vector at each sample location n. Thus typically, $\nabla : f \mapsto (\nabla f_n)_n \in \mathbb{R}^{N \times d}$ maps to d-dimensional vector fields. Then $-\nabla^* : \mathbb{R}^{N \times d} \to \mathbb{R}^N$ is a discretized divergence operator. In this case, $\Delta = -GG^*$ is a discretization of the Laplacian, which is itself a convolution operator. One then has

$$\hat{f}_{\lambda,m} = \frac{\hat{h}_m^* \hat{y}_m}{|\hat{h}_m|^2 - \lambda \hat{d}_{2,m}}, \tag{10.22}$$

where $\hat{d}_2$ is the Fourier transform of the filter d_2 corresponding to the Laplacian. For instance, in dimension 1, using first order finite differences, the expression for $\hat{d}_{2,m}$ is given in (2.18).

10.3.1 Solving Linear System

When Φ and L do not share the same singular spaces, using (10.20) is not possible, so that one needs to solve the linear system (10.19), which can be rewritten as

$$Af = b \quad \text{where} \quad A \overset{\text{def.}}{=} \Phi^*\Phi + \lambda L \quad \text{and} \quad b = \Phi^* y.$$

It is possible to solve exactly this linear system with direct methods for moderate N (up to a few thousands), and the numerical complexity for a generic A is $O(N^3)$. Since the involved matrix A is symmetric, the best option is to use Choleski factorization $A = BB^*$ where B is lower-triangular. In favorable cases, this factorization (possibly with some re-re-ordering of the row and columns) can take advantage of some sparsity in A.

For large N, such exact resolution is not an option, and should use approximate iterative solvers, which only access A through matrix-vector multiplication. This is especially advantageous for imaging applications, where such multiplications are in general much faster than a naive $O(N^2)$ explicit computation. If the matrix A is highly sparse, this typically necessitates $O(N)$ operations. In the case where A is symmetric and positive definite (which is the case here), the most well known method is the conjugate gradient methods, which is actually an optimization method solving

$$\min_{f \in \mathbb{R}^N} \mathcal{E}(f) \overset{\text{def.}}{=} \mathcal{Q}(f) \overset{\text{def.}}{=} \langle Af, f \rangle - \langle f, b \rangle \tag{10.23}$$

which is equivalent to the initial minimization (10.16). Instead of doing a naive gradient descent (as studied in Section 10.4.2 bellow), stating from an arbitrary $f^{(0)}$, it compute a new iterate $f^{(\ell+1)}$ from the previous iterates as

$$f^{(\ell+1)} \overset{\text{def.}}{=} \operatorname*{argmin}_f \left\{ \mathcal{E}(f) \;;\; f \in f^{(\ell)} + \operatorname{Span}(\nabla\mathcal{E}(f^{(0)}), \ldots, \nabla\mathcal{E}(f^{(\ell)})) \right\}.$$

The crucial and remarkable fact is that this minimization can be computed in closed form at the cost of two matrix-vector product per iteration, for $k \geqslant 1$ (posing initially $d^{(0)} = \nabla\mathcal{E}(f^{(0)}) = Af^{(0)} - b$)

$$f^{(\ell+1)} = f^{(\ell)} - \tau_\ell d^{(\ell)} \quad \text{where} \quad d^{(\ell)} = g_\ell + \frac{\|g^{(\ell)}\|^2}{\|g^{(\ell-1)}\|^2} d^{(\ell-1)} \quad \text{and} \quad \tau_\ell = \frac{\langle g_\ell, d^{(\ell)} \rangle}{\langle Ad^{(\ell)}, d^{(\ell)} \rangle} \tag{10.24}$$

$g^{(\ell)} \overset{\text{def.}}{=} \nabla\mathcal{E}(f^{(\ell)}) = Af^{(\ell)} - b$. It can also be shown that the direction $d^{(\ell)}$ are orthogonal, so that after $\ell = N$ iterations, the conjugate gradient computes the unique solution $f^{(\ell)}$ of the linear system $Af = b$. It is however rarely used this way (as an exact solver), and in practice much less than N iterates are computed. It should also be noted that iterations (10.24) can be caried over for an arbitrary smooth convex function $\mathcal{E}$, and it typically improves over the gradient descent (although in practice quasi-Newton method are often preferred).

10.4 Non-Quadratic Regularization

10.4.1 Total Variation Regularization

A major issue with quadratic regularization such as (10.18) is that they typically leads to blurry recovered data f_λ, which is thus not a good approximation of f_0 when it contains sharp transition such as edges in images. This is can clearly be seen in the convolutive case (10.22), this the restoration operator $\Phi_\lambda^+\Phi$ is a filtering, which tends to smooth sharp part of the data.

This phenomena can also be understood because the restored data f_λ always belongs to $\mathrm{Im}(\Phi^*) = \ker(\Phi)^\perp$, and thus cannot contains "high frequency" details that are lost in the kernel of Φ. To alleviate this shortcoming, and recover missing information in the kernel, it is thus necessarily to consider non-quadratic and in fact non-smooth regularization.

Total variation. The most well know instance of such a non-quadratic and non-smooth regularization is the total variation prior. For smooth function $f : \mathbb{R}^d \mapsto \mathbb{R}$, this amounts to replacing the quadratic Sobolev energy (often called Dirichlet energy)

$$J_{\mathrm{Sob}}(f) \overset{\text{def.}}{=} \frac{1}{2} \int_{\mathbb{R}^d} \|\nabla f\|_{\mathbb{R}^d}^2 \mathrm{d}x,$$

where $\nabla f(x) = (\partial_{x_1} f(x), \ldots, \partial_{x_d} f(x))^\top$ is the gradient, by the (vectorial) L^1 norm of the gradient

$$J_{\mathrm{TV}}(f) \overset{\text{def.}}{=} \int_{\mathbb{R}^d} \|\nabla f\|_{\mathbb{R}^d} \mathrm{d}x.$$

We refer also to Section 9.1.1 about these priors. Simply "removing" the square ^{2} inside the integral might seems light a small change, but in fact it is a game changer.

Indeed, while $J_{\mathrm{Sob}}(1_\Omega) = +\infty$ where 1_Ω is the indicator a set Ω with finite perimeter $|\Omega| < +\infty$, one can show that $J_{\mathrm{TV}}(1_\Omega) = |\Omega|$, if one interpret ∇f as a distribution Df (actually a vectorial Radon measure) and $\int_{\mathbb{R}^d} \|\nabla f\|_{\mathbb{R}^d} \mathrm{d}x$ is replaced by the total mass $|Df|(\Omega)$ of this distribution $m = Df$

$$|m|(\Omega) = \sup \left\{ \int_{\mathbb{R}^d} \langle h(x), \mathrm{d}m(x) \rangle \;;\; h \in \mathcal{C}(\mathbb{R}^d \mapsto \mathbb{R}^d), \forall\, x, \|h(x)\| \leqslant 1 \right\}.$$

The total variation of a function such that Df has a bounded total mass (a so-called bounded variation function) is hence defined as

$$J_{\mathrm{TV}}(f) \overset{\text{def.}}{=} \sup \left\{ \int_{\mathbb{R}^d} f(x) \,\mathrm{div}(h)(x) \mathrm{d}x \;;\; h \in \mathcal{C}_c^1(\mathbb{R}^d; \mathbb{R}^d), \|h\|_\infty \leqslant 1 \right\}.$$

Generalizing the fact that $J_{\mathrm{TV}}(1_\Omega) = |\Omega|$, the functional co-area formula reads

$$J_{\mathrm{TV}}(f) = \int_{\mathbb{R}} \mathcal{H}_{d-1}(L_t(f)) \mathrm{d}t \quad \text{where} \quad L_t(f) = \{x \;;\; f(x) = t\}$$

and where $\mathcal{H}_{d-1}$ is the Hausforf measures of dimension $d-1$, for instance, for $d = 2$ if L has finite perimeter $|L|$, then $\mathcal{H}_{d-1}(L) = |L|$ is the perimeter of L.

Discretized Total variation. For discretized data $f \in \mathbb{R}^N$, one can define a discretized TV semi-norm as detailed in Section 9.1.2, and it reads, generalizing (9.6) to any dimension

$$J_{\mathrm{TV}}(f) = \sum_n \|\nabla f_n\|_{\mathbb{R}^d}$$

where $\nabla f_n \in \mathbb{R}^d$ is a finite difference gradient at location indexed by n.

The discrete total variation prior $J_{\mathrm{TV}}(f)$ defined in (9.6) is a convex but non differentiable function of f, since a term of the form $\|\nabla f_n\|$ is non differentiable if $\nabla f_n = 0$. We defer to chapters 12 and 13 the study of advanced non-smooth convex optimization technics that allows to handle this kind of functionals.

In order to be able to use simple gradient descent methods, one needs to smooth the TV functional. The general machinery proceeds by replacing the non-smooth ℓ^2 Euclidean norm $\|\cdot\|$ by a smoothed version, for instance

$$\forall\, u \in \mathbb{R}^d, \quad \|u\|_\varepsilon \overset{\text{def.}}{=} \sqrt{\varepsilon^2 + \|u\|}.$$

This leads to the definition of a smoothed approximate TV functional, already introduced in (9.12),

$$J^\varepsilon_{\mathrm{TV}}(f) \overset{\text{def.}}{=} \sum_n \|\nabla f_n\|_\varepsilon$$

One has the following asymptotics for $\varepsilon \to \{0, +\infty\}$

$$\|u\|_\varepsilon \overset{\varepsilon\to 0}{\longrightarrow} \|u\| \quad \text{and} \quad \|u\|_\varepsilon = \varepsilon + \frac{1}{2\varepsilon}\|u\|^2 + O(1/\varepsilon^2)$$

which suggest that $J^\varepsilon_{\mathrm{TV}}$ interpolates between J_{TV} and J_{Sob}.

The resulting inverse regularization problem (10.16) thus reads

$$f_\lambda \overset{\text{def.}}{=} \operatorname*{argmin}_{f\in\mathbb{R}^N} \mathcal{E}(f) = \frac{1}{2}\|y - \Phi f\|^2 + \lambda J^\varepsilon_{\mathrm{TV}}(f) \tag{10.25}$$

It is a strictly convex problem (because $\|\cdot\|_\varepsilon$ is strictly convex for $\varepsilon > 0$) so that its solution f_λ is unique.

10.4.2 Gradient Descent Method

The optimization program (10.25) is a example of smooth unconstrained convex optimization of the form

$$\min_{f\in\mathbb{R}^N} \mathcal{E}(f) \tag{10.26}$$

where $\mathcal{E} : \mathbb{R}^N \to \mathbb{R}$ is a $\mathcal{C}^1$ function. Recall that the gradient $\nabla\mathcal{E} : \mathbb{R}^N \mapsto \mathbb{R}^N$ of this functional (not to be confound with the discretized gradient $\nabla f \in \mathbb{R}^N$ of f) is defined by the following first order relation

$$\mathcal{E}(f + r) = \mathcal{E}(f) + \langle f,\, r\rangle_{\mathbb{R}^N} + O(\|r\|^2_{\mathbb{R}^N})$$

where we used $O(\|r\|^2_{\mathbb{R}^N})$ in place of $o(\|r\|_{\mathbb{R}^N})$ (for differentiable function) because we assume here $\mathcal{E}$ is of class $\mathcal{C}^1$ (i.e. the gradient is continuous).

For such a function, the gradient descent algorithm is defined as

$$f^{(\ell+1)} = f^{(\ell)} - \tau_\ell \nabla\mathcal{E}(f^{(\ell)}),$$

where the step size $\tau_\ell > 0$ should be small enough to guarantee convergence, but large enough for this algorithm to be fast.

To allow for not-too-small steps,

One needs to quantify the smoothness of $\mathcal{E}$. This is enforced by requiring that the gradient is L-Lipschitz, i.e.

$$\forall\,(f, g) \in (\mathbb{R}^N)^2, \quad \|\nabla\mathcal{E}(f) - \nabla\mathcal{E}(g)\| \leqslant L\|f - g\|. \tag{$\mathcal{R}_L$}$$

In order to obtain fast convergence of the iterates themselve, it is needed that the function has enough "curvature" (i.e. is not too flat), which corresponds to imposing that $\mathcal{E}$ is M-strongly convex

$$\forall\,(f, g), \in (\mathbb{R}^N)^2, \quad \langle \nabla\mathcal{E}(f) - \nabla\mathcal{E}(g),\, f - g\rangle \geqslant M\|f - g\|^2. \tag{$\mathcal{S}_L$}$$

The following proposition express these conditions as constraints on the hessian for $\mathcal{C}^2$ functions.

Proposition 22. *If $\mathcal{E}$ is of class $\mathcal{C}^2$, conditions $(\mathcal{R}_L)$ and $(\mathcal{S}_L)$ are equivalent to*

$$\forall\, f, \quad M\mathrm{Id}_{N\times N} \preceq \partial^2\mathcal{E}(f) \preceq L\mathrm{Id}_{N\times N} \tag{10.27}$$

where $\partial^2\mathcal{E}(f) \in \mathbb{R}^{N\times N}$ is the Hessian of $\mathcal{E}$, and where $\preceq$ is the natural order on symmetric matrices, i.e.

$$A \preceq B \quad \Longleftrightarrow \quad \forall\, u \in \mathbb{R}^N, \quad \langle Au,\, u\rangle \leqslant \langle Bu,\, u\rangle.$$

Condition (10.27) thus reads that the singular values of $\partial^2\mathcal{E}(f)$ should be contained in the interval $[M, L]$. The upper bound is also equivalent to $\|\partial^2\mathcal{E}(f)\|_{\mathrm{op}} \leqslant L$ where $\|\cdot\|_{\mathrm{op}}$ is the operator norm, i.e. the largest singular value. In the special case of a quadratic function $\mathcal{Q}$ of the form (10.23), $\partial^2\mathcal{E}(f) = A$ is constant, so that $[M, L]$ can be chosen to be the range of the singular values of A.

The following theorem ensure the convergence of the gradient descent with a linear speed.

Theorem 32. *If f satisfy conditions $(\mathcal{R}_L)$ and $(\mathcal{S}_L)$, assuming there exists $(\tau_{\min}, \tau_{\max})$ such that*

$$0 < \tau_{\min} \leqslant \tau_\ell \leqslant \tau_{\max} < \frac{2M}{L} \tag{10.28}$$

then there exists $0 \leqslant \rho < 1$ such that

$$\|f^{(\ell)} - f^\star\| \leqslant \rho^\ell \|f^{(0)} - f^\star\| \tag{10.29}$$

where $f^\star$ is the unique solution to (10.26).

Proof. Since $\nabla\mathcal{E}(f^\star) = 0$, one has

$$f^{(\ell+1)} - f^\star = (f^{(\ell)} - f^\star) - \tau_\ell(\nabla\mathcal{E}(f^{(\ell)}) - \nabla\mathcal{E}(f^\star)).$$

Hence, using strong convexity and Lipschitz gradient

$$\begin{aligned}\|f^{(\ell+1)} - f^\star\|^2 &= \|f^{(\ell)} - f^\star\|^2 - 2\tau_\ell\langle f^{(\ell)} - f^\star,\, \nabla\mathcal{E}(f^{(\ell)}) - \nabla\mathcal{E}(f^\star)\rangle + \tau_\ell^2\|\nabla\mathcal{E}(f^{(\ell)}) - \nabla\mathcal{E}(f^\star)\|^2 \\ &\leqslant P(\tau_\ell)\|f^{(\ell)} - f^\star\|^2 \quad \text{where} \quad P(\tau) = 1 - 2M\tau + L^2\tau^2.\end{aligned}$$

Figure **??** shows visually the shape of the second order polynomial P, which shows that condition (10.34) on τ_ℓ implies

$$P(\tau_\ell)^{\frac{1}{2}} \leqslant \rho \overset{\text{def.}}{=} \max(P(\tau_{\min}), P(\tau_{\max}))^{\frac{1}{2}} < 1,$$

which shows the desired result. □

The error decay rate (10.32), although it is geometrical $O(\rho^\ell)$ is called a "linear rate" in the optimization litterature. It is a "global" rate because it hold for all ℓ (and not only for large enough ℓ). The best (smallest) rate ρ is obtained when choosing

$$\tau_\ell = \frac{M}{L^2} \quad \Longrightarrow \quad \rho = 1 - \frac{M^2}{L^2}. \tag{10.30}$$

In the case of a quadratic functional of the form (10.23), one can sharpen the convergence proof because the iterates are computed in closed form using matrix multiplication

$$f^{(\ell)} - f^\star = (\mathrm{Id}_N - \tau_\ell A)(f^{(0)} - f^\star)$$

which immediately leads to the following proposition.

Proposition 23. *For $\mathcal{E}(f) = \langle A,\, f\rangle - \langle b,\, f\rangle$ with the singular values of A upper-bounded by L, assuming there exists $(\tau_{\min}, \tau_{\max})$ such that*

$$0 < \tau_{\min} \leqslant \tau_\ell \leqslant \tilde{\tau}_{\max} < \frac{2}{L} \tag{10.31}$$

then there exists $0 \leqslant \tilde{\rho} < 1$ *such that*

$$\|f^{(\ell)} - f^\star\| \leqslant \tilde{\rho}^\ell \|f^{(0)} - f^\star\|. \tag{10.32}$$

If the singular values are lower bounded by M*, then the best rate* $\tilde{\rho}$ *is obtained for*

$$\tau_\ell = \frac{1}{L+M} \quad \Longrightarrow \quad \tilde{\rho} \overset{\text{def.}}{=} \frac{L-M}{L+M}. \tag{10.33}$$

The maximum allowable step size $\tilde{\tau}_{\max}$ in (10.34) is much larger than $\tau_{\max}$ given in (10.34), and the optimal rate (10.33) is also much better (smaller) than the one in (10.30). In particular, if $\varepsilon \overset{\text{def.}}{=} M/L \ll 1$ (which is the typical setup for ill-posed problems), then

$$\rho \sim 1 - \varepsilon^2 \quad \text{and} \quad \tilde{\rho} \sim 1 - 2\varepsilon.$$

These two results are however complementary. Indeed, if the gradient descent converges, then ultimately $f^{(\ell)}$ is close to $f^\star$, so that one can approximate up to second order $\mathcal{E}(f) \approx \mathcal{E}(f^\star) + \langle Af,\, f\rangle - \langle f,\, b\rangle$ with $A = \partial^2 \mathcal{E}(f^\star)$ and $b = -\nabla \mathcal{E}(f^\star)$. So that the "local" rate, the one obtained after a large enough of iterations, is actually driven by $\tilde{\rho}$ and not ρ. It is thus important to distinguish between the global rate and the local rate. In practice, descent algorithm typically have two phase: a first "slow" phase govern by the global rate, and a second "fast" phase governed by the local rate. Unfortunately, the optimal step sizes τ_ℓ are in general different for the two phase, so that optimal adaptation of step size is a difficult problems. This is why more advanced users typically use various line search strategies (to find the optimal step size at each iteration) or use second order information using quasi-Newton technics (BFGS).

The convergence result of Proposition 23 does not requires strong convexity, while Theorem 32 does. In the general non-strongly convex case, it is still possible to prove convergence, but the rate is only sub-linear, and is only on the value of $\mathcal{E}$, not on the iterate $f^{(\ell)}$ themselves. Note that in this case, the solution of the minimization problem is not necessarily unique. The proof is more technical.

Theorem 33. *If* f *satisfy conditions* $(\mathcal{R}_L)$*, assuming there exists* $(\tau_{\min}, \tau_{\max})$ *such that*

$$0 < \tau_{\min} \leqslant \tau_\ell \leqslant \tau_{\max} < \frac{2}{L}, \tag{10.34}$$

then $f^{(\ell)}$ *converges to a solution* $f^\star$ *of* (10.26) *and there exists* $C > 0$ *such that*

$$\mathcal{E}(f^{(\ell)}) - \mathcal{E}(f^\star) \leqslant \frac{C}{\ell}.$$

10.4.3 Examples of Gradient Computation

Note that the gradient of a quadratic function $\mathcal{Q}(f)$ of the form (10.23) reads

$$\nabla \mathcal{G}(f) = Af - b.$$

In particular, one retrieves that the first order optimality condition $\nabla \mathcal{G}(f) = 0$ is equivalent to the linear system $Af = b$.

For the quadratic fidelity term $\mathcal{G}(f) = \frac{1}{2}\|\Phi f - y\|^2$, one thus obtains

$$\nabla \mathcal{G}(f) = \Phi^*(\Phi y - y).$$

In the special case of the regularized TV problem (10.25), the gradient of $\mathcal{E}$ reads

$$\nabla \mathcal{E}(f) = \Phi^*(\Phi y - y) + \lambda \nabla J^\varepsilon_{\mathrm{TV}}(f).$$

Recall the chain rule for differential reads $\partial(\mathcal{G}_1 \circ \mathcal{G}_2) = \partial \mathcal{G}_1 \circ \partial \mathcal{G}_2$, but that gradient vectors are actually transposed of differentials, so that for $\mathcal{E} = \mathcal{F} \circ \mathcal{H}$ where $\mathcal{F} : \mathbb{R}^P \to \mathbb{R}$ and $\mathcal{H} : \mathbb{R}^N \to \mathbb{R}^P$, one has

$$\nabla \mathcal{E}(f) = [\partial \mathcal{H}(f)]^* \left(\nabla \mathcal{F}(\mathcal{H} f)\right).$$

Since $J_{\mathrm{TV}}^{\varepsilon} = \|\cdot\|_{1,\varepsilon} \circ \nabla$, one thus has

$$\nabla J_{\mathrm{TV}}^{\varepsilon} = \nabla^{\star} \circ (\partial \|\cdot\|_{1,\varepsilon}) \quad \text{where} \quad \|u\|_{1,\varepsilon} = \sum_n \|u_n\|_{\varepsilon}$$

so that

$$J_{\mathrm{TV}}^{\varepsilon}(f) = -\operatorname{div}(\mathcal{N}^{\varepsilon}(\nabla f)),$$

where $\mathcal{N}^{\varepsilon}(u) = (u_n/\|u_n\|_{\varepsilon})_n$ is the smoothed-normalization operator of vector fields (the differential of $\|\cdot\|_{1,\varepsilon}$), and where $\operatorname{div} = -\nabla^*$ is minus the adjoint of the gradient.

Since $\operatorname{div} = -\nabla^*$, their Lipschitz constants are equal $\|\operatorname{div}\|_{\mathrm{op}} = \|\nabla\|_{\mathrm{op}}$, and is for instance equal to $\sqrt{2d}$ for the discretized gradient operator. Computation shows that the Hessian of $\|\cdot\|_{\varepsilon}$ is bounded by $1/\varepsilon$, so that for the smoothed-TV functional, the Lipschitz constant of the gradient is upper-bounded by

$$L = \frac{\|\nabla\|^2}{\varepsilon} + \|\Phi\|_{\mathrm{op}}^2.$$

Furthermore, this functional is strongly convex because $\|\cdot\|_{\varepsilon}$ is ε-strongly convex, and the Hessian is lower bounded by

$$M = \varepsilon + \sigma_{\min}(\Phi)^2$$

where $\sigma_{\min}(\Phi)$ is the smallest singular value of Φ. For ill-posed problems, typically $\sigma_{\min}(\Phi) = 0$ or is very small, so that both L and M degrades (tends respectively to 0 and $+\infty$) as $\varepsilon \to 0$, so that gradient descent becomes prohibitive for small ε, and it is thus required to use dedicated non-smooth optimization methods detailed in the following chapters. On the good news side, note however that in many case, using a small but non-zero value for ε often leads to better a visually more pleasing results, since it introduce a small blurring which diminishes the artifacts (and in particular the so-called "stair-casing" effect) of TV regularization.

10.5 Examples of Inverse Problems

We detail here some inverse problem in imaging that can be solved using quadratic regularization or non-linear TV.

10.5.1 Deconvolution

The blurring operator (10.1) is diagonal over Fourier, so that quadratic regularization are easily solved using Fast Fourier Transforms when considering periodic boundary conditions. We refer to (10.21) and the correspond explanations. TV regularization in contrast cannot be solved with fast Fourier technics, and is thus much slower.

10.5.2 Inpainting

For the inpainting problem, the operator defined in (10.3) is diagonal in space

$$\Phi = \operatorname{diag}_m(\delta_{\Omega^c}[m]),$$

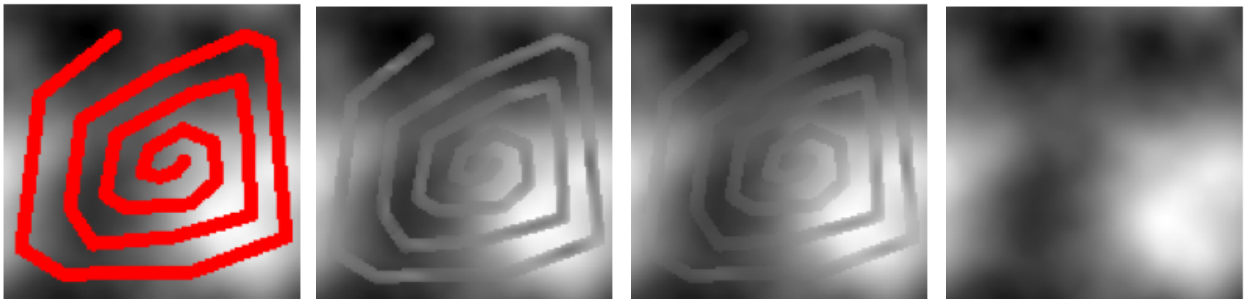
and is an orthogonal projector $\Phi^* = \Phi$.

In the noiseless case, to constrain the solution to lie in the affine space $\{f \in \mathbb{R}^N \;;\; y = \Phi f\}$, we use the orthogonal projector

$$\forall\, x, \quad P_y(f)(x) = \begin{cases} f(x) & \text{if} \quad x \in \Omega, \\ y(x) & \text{if} \quad x \notin \Omega. \end{cases}$$

In the noiseless case, the recovery (10.17) is solved using a projected gradient descent. For the Sobolev energy, the algorithm iterates

$$f^{(\ell+1)} = P_y(f^{(\ell)} + \tau \Delta f^{(\ell)}).$$



$k = 1$ $k = 10$ $k = 20$ $k = 100$

Figure 10.2: Sobolev projected gradient descent algorithm.

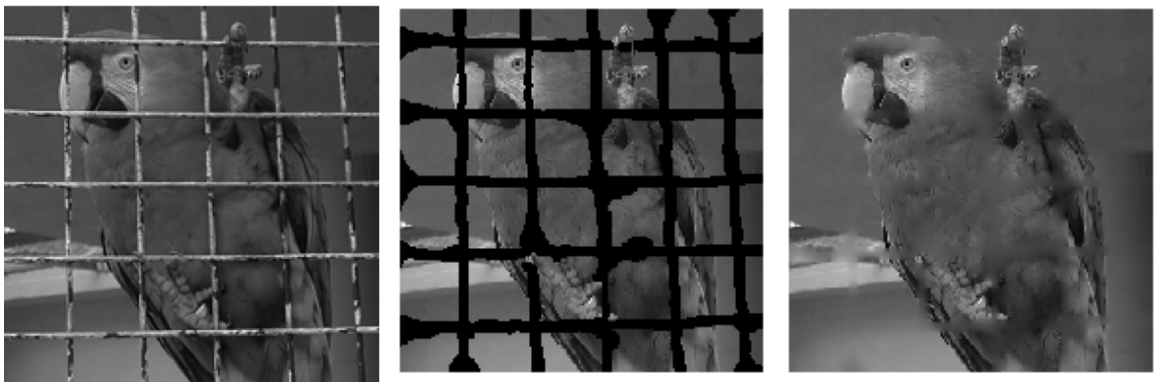


Image f_0 Observation $y = \Phi f_0$ Sobolev $f^\star$

Figure 10.3: Inpainting the parrot cage.

which converges if $\tau < 2/\|\Delta\| = 1/4$. Figure 10.2 shows some iteration of this algorithm, which progressively interpolate within the missing area.

Figure 10.3 shows an example of Sobolev inpainting to achieve a special effect.

For the smoothed TV prior, the gradient descent reads

$$f^{(\ell+1)} = P_y\left(f^{(\ell)} + \tau \operatorname{div}\left(\frac{\nabla f^{(\ell)}}{\sqrt{\varepsilon^2 + \|\nabla f^{(\ell)}\|^2}}\right)\right)$$

which converges if $\tau < \varepsilon/4$.

Figure 10.4 compare the Sobolev inpainting and the TV inpainting for a small value of ε. The SNR is not improved by the total variation, but the result looks visually slightly better.

10.5.3 Tomography Inversion

In medical imaging, a scanner device compute projection of the human body along rays $\Delta_{t,\theta}$ defined

$$x \cdot \tau_\theta = x_1 \cos\theta + x_2 \sin\theta = t$$

where we restrict ourself to 2D projection to simplify the exposition.

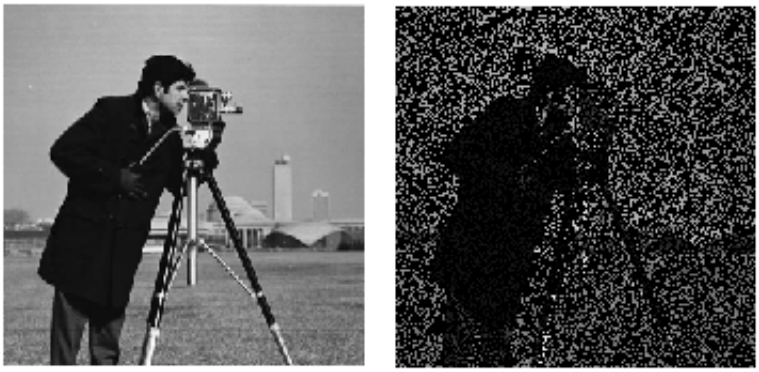


Image f_0 | Observation $y = \Phi f_0$

Sobolev $f^\star$
SNR=?dB | TV $f^\star$
SNR=?dB

Figure 10.4: Inpainting with Sobolev and TV regularization.

The scanning process computes a Radon transform, which compute the integral of the function to acquires along rays

$$\forall\, \theta \in [0,\pi), \forall\, t \in \mathbb{R}, \quad p_\theta(t) = \int_{\Delta_{t,\theta}} f(x)\, ds = \iint f(x)\, \delta(x \cdot \tau_\theta - t)\, dx$$

see Figure (10.5)

The Fourier slice theorem relates the Fourier transform of the scanned data to the 1D Fourier transform of the data along rays

$$\forall\, \theta \in [0,\pi)\ ,\ \forall\, \xi \in \mathbb{R} \quad \hat{p}_\theta(\xi) = \hat{f}(\xi\cos\theta, \xi\sin\theta). \tag{10.35}$$

This shows that the pseudo inverse of the Radon transform is computed easily over the Fourier domain using inverse 2D Fourier transform

$$f(x) = \frac{1}{2\pi} \int_0^\pi p_\theta \star h(x \cdot \tau_\theta)\, d\theta$$

with $\hat{h}(\xi) = |\xi|$.

Imaging devices only capture a limited number of equispaced rays at orientations $\{\theta_k = \pi/k\}_{0 \leqslant k < K}$. This defines a tomography operator which corresponds to a partial Radon transform

$$Rf = (p_{\theta_k})_{0 \leqslant k < K}.$$

Relation (10.35) shows that knowing Rf is equivalent to knowing the Fourier transform of f along rays,

$$\{\hat{f}(\xi\cos(\theta_k), \xi\sin(\theta_k))\ \}_k.$$

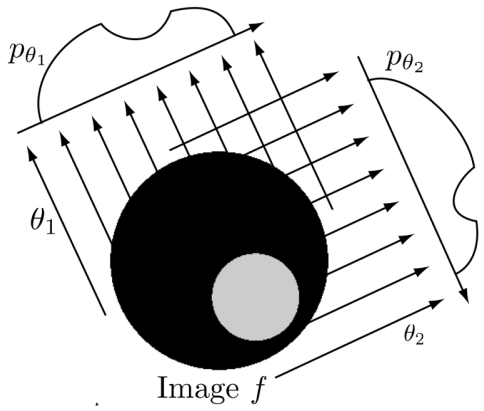


Figure 10.5: Principle of tomography acquisition.

We thus simply the acquisition process over the discrete domain and model it as computing directly samples of the Fourier transform

$$\Phi f = (\hat{f}[\omega])_{\omega \in \Omega} \in \mathbb{R}^P$$

where Ω is a discrete set of radial lines in the Fourier plane, see Figure 10.6, right.

In this discrete setting, recovering from Tomography measures $y = Rf_0$ is equivalent in this setup to inpaint missing Fourier frequencies, and we consider partial noisy Fourier measures

$$\forall\, \omega \in \Omega, \quad y[\omega] = \hat{f}[\omega] + w[\omega]$$

where $w[\omega]$ is some measurement noise, assumed here to be Gaussian white noise for simplicity.

The peuso-inverse $f^+ = R^+ y$ defined in (10.7) of this partial Fourier measurements reads

$$\hat{f}^+[\omega] = \begin{cases} y[\omega] & \text{if} \quad \omega \in \Omega, \\ 0 & \text{if} \quad \omega \notin \Omega. \end{cases}$$

Figure 10.7 shows examples of pseudo inverse reconstruction for increasing size of Ω. This reconstruction exhibit serious artifact because of bad handling of Fourier frequencies (zero padding of missing frequencies).

The total variation regularization (**??**) reads

$$f^\star \in \operatorname*{argmin}_f \frac{1}{2} \sum_{\omega \in \Omega} |y[\omega] - \hat{f}[\omega]|^2 + \lambda \|f\|_{\mathrm{TV}}.$$

It is especially suitable for medical imaging where organ of the body are of relatively constant gray value, thus resembling to the cartoon image model introduced in Section 6.2.4. Figure 10.8 compares this total variation recovery to the pseudo-inverse for a synthetic cartoon image. This shows the hability of the total variation to recover sharp features when inpainting Fourier measures. This should be contrasted with the difficulties that faces TV regularization to inpaint over the spacial domain, as shown in Figure 11.9.

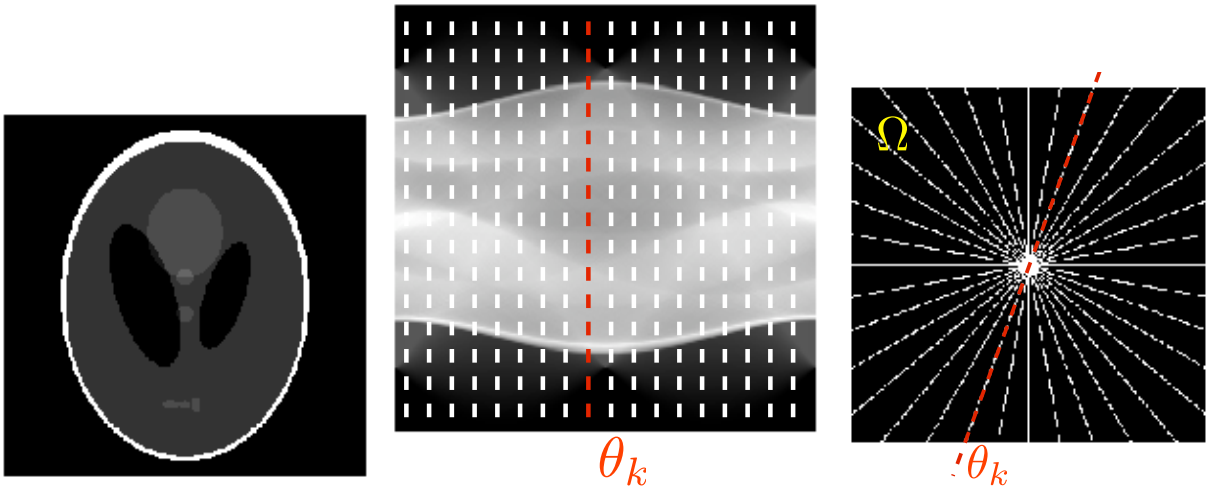


Image f | Radon sub-sampling | Fourier domain

Figure 10.6: Partial Fourier measures.

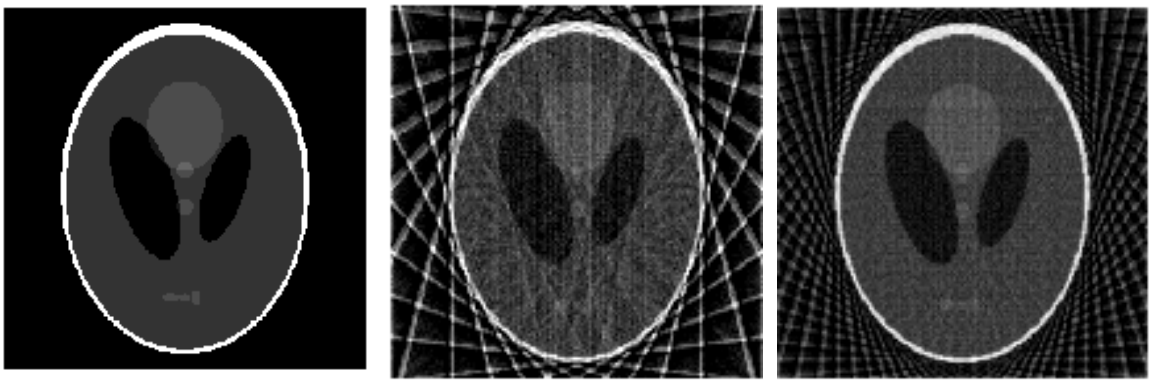


Image f_0 | 13 projections | 32 projections.

Figure 10.7: Pseudo inverse reconstruction from partial Radon projections.

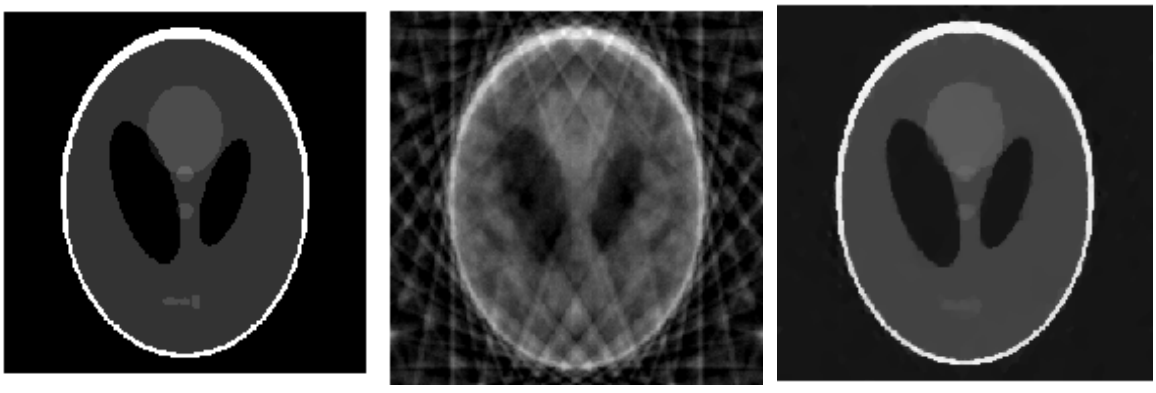


Image f_0 | Pseudo-inverse | TV

Figure 10.8: Total variation tomography inversion.

Chapter 11

Sparse Regularization

TODO.
Ref [29, 42, 37]

11.1 Sparsity Priors

11.1.1 Ideal sparsity prior.

As detailed in Chapter **??**, it is possible to use an orthogonal basis $\mathcal{B} = \{\psi_m\}_m$ to efficiently approximate an image f in a given class $f \in \Theta$ with a few atoms from $\mathcal{B}$.

To measure the complexity of an approximation with $\mathcal{B}$, we consider the ℓ^0 prior, which counts the number of non-zero coefficients in $\mathcal{B}$

$$J_0(f) = \#\{m \,;\, \langle f, \psi_m\rangle \neq 0\} = \|a\|_0 \quad \text{where} \quad a[m] = \langle f, \psi_m\rangle.$$

We have introduced the ℓ^0 pseudo-norm $\|a\|_0$, which we treat here as an ideal sparsity measure for the coefficients a of f in $\mathcal{B}$.

Natural images are not exactly composed of a few atoms, but they can be well approximated by a function f_M with a small ideal sparsity $M = J_0(f)$. In particular, the best M-term approximation defined in (6.3) is defined by

$$f_M = \sum_{|\langle f, \psi_m\rangle| > T} \langle f, \psi_m\rangle \psi_m \quad \text{where} \quad M = \#\{m \,;\, |\langle f, \psi_m\rangle| > T\}\,.$$

As detailed in Section 6.2, discontinuous images with bounded variation have a fast decay of the approximation error $\|f - f_M\|$. Natural images f are well approximated by images with a small value of the ideal sparsity prior J_0.

Figure 11.1 shows an examples of decomposition of a natural image in a wavelet basis, $\psi_m = \psi^{\omega}_{j,n}$ $m = (j, n, \omega)$. This shows that most $\langle f, \psi_m\rangle$ are small, and hence the decomposition is quite sparse.

11.1.2 Convex relaxation

Unfortunately, the ideal sparsity prior J_0 is difficult to handle numerically because $J_0(f)$ is not a convex function of f. For instance, if f and g have non-intersecting supports of there coefficients in $\mathcal{B}$, then $J_0((f+g)/2) = J_0(f) + J_0(g)$, which shows the highly non-convex behavior of J_0.

This ideal sparsity J_0 is thus not amenable to minimization, which is an issue to solve general inverse problems considered in Section **??**.

We consider a family of ℓ^q priors for $q > 0$, intended to approximate the ideal prior J_0

$$J_q(f) = \sum_m |\langle f, \psi_m\rangle|^q.$$

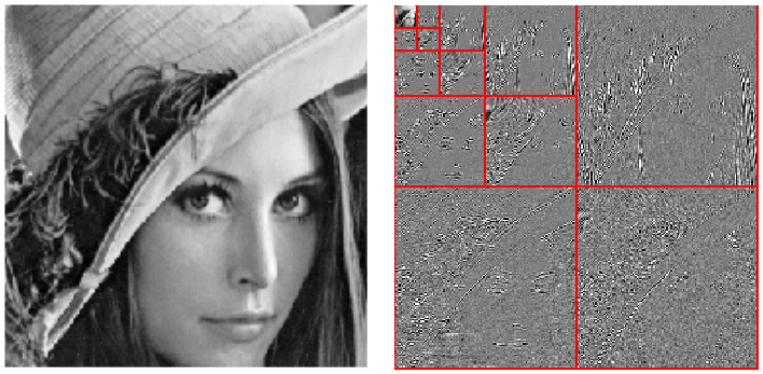


Image f Coefficients $\langle f, \psi_m \rangle$

Figure 11.1: Wavelet coefficients of natural images are relatively sparse.

As shown in Figure 11.2, the unit balls in $\mathbb{R}^2$ associated to these priors are shrinking toward the axes, which corresponds to the unit ball for the ℓ^0 pseudo norm. In some sense, the J_q priors are becoming closer to J_0 as q tends to zero, and thus J_q favors sparsity for small q.

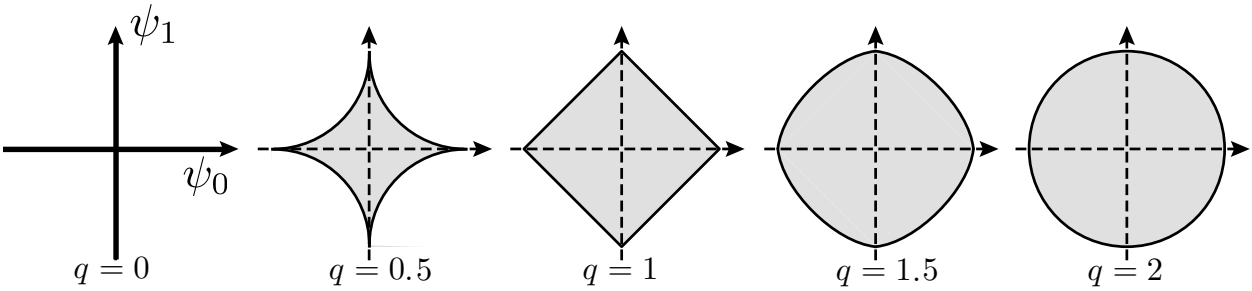


Figure 11.2: ℓ^q balls $\{x\ ;\ J_q(x) \leqslant 1\}$ for varying q.

The prior J_q is convex if and only if $q \geqslant 1$. To reach the highest degree of sparsity while using a convex prior, we consider the ℓ^1 sparsity prior J_1, which is thus defined as

$$J_1(f) = \|(\langle f, \psi_m \rangle)\|_1 = \sum_m |\langle f, \psi_m \rangle|. \tag{11.1}$$

In the following, we consider discrete orthogonal bases $\mathcal{B} = \{\psi_m\}_{m=0}^{N-1}$ of $\mathbb{R}^N$.

11.1.3 Sparse Regularization and Thresholding

Given some orthogonal basis $\{\psi_m\}_m$ of $\mathbb{R}^N$, the denoising by regularization (9.15) is written using the sparsity J_0 and J_1 as

$$f^\star_{\lambda,q} = \operatorname*{argmin}_{g \in \mathbb{R}^N} \frac{1}{2}\|f - g\|^2 + \lambda J_q(f)$$

for $q = 0$ or $q = 1$. It can be re-written in the orthogonal basis as

$$f^\star_{\lambda,q} = \sum_m a^\star_{\lambda,q}[m]\psi_m$$

$$a^\star_{\lambda,q} = \operatorname*{argmin}_{b\in\mathbb{R}^N} \sum_m \frac{1}{2}|a[m]-b[m]|^2 + \lambda\varphi_q(b[m])$$

where $a[m] = \langle f, \psi_m\rangle$, and with

$$\varphi_1(x) = |x| \quad \text{and} \quad \varphi_0(x) = \begin{cases} 0 & \text{if} \quad x = 0, \\ 1 & \text{otherwise.} \end{cases}$$

Each coefficients of the denoised image is the solution of

$$a^\star_{\lambda,q}[m] = \operatorname*{argmin}_{\alpha\in\mathbb{R}} \frac{1}{2}|a[m]-\beta|^2 + \lambda\varphi_q(\beta)$$

and one can shows that this optimization is solved exactly in closed form using thresholding

$$a^\star_{\lambda,q}[m] = S^q_T(a[m]) \quad \text{where} \quad \begin{cases} T = \sqrt{2\lambda} & \text{for} \quad q = 0, \\ T = \lambda & \text{for} \quad q = 1, \end{cases} \tag{11.2}$$

where S^0_T is the hard thresholding introduced in (8.6), and S^1_T is the soft thresholding introduced in (8.7).

One thus has

$$f_{\lambda,q} = \sum_m S^q_T(\langle f, \psi_m\rangle)\psi_m.$$

As detailed in Section 8.3, these denoising methods has the advantage that the threshold is simple to set for Gaussian white noise w of variance σ^2. Theoritical values indicated that $T = \sqrt{2\log(N)}\sigma$ is asymptotically optimal, see Section 8.3.3. In practice, one should choose $T \approx 3\sigma$ for hard thresholding (ℓ^0 regularization), and $T \approx 3\sigma/2$ for soft thresholding (ℓ^1 regularization), see Figure 8.14.

11.2 Sparse Regularization of Inverse Problems

Sparse ℓ^1 regularization in an orthogonal basis $\{\psi_m\}_m$ of $\mathbb{R}^N$ makes use of the J_1 prior defined in (11.1), so that the inversion is obtained by solving the following convex program

$$f^\star \in \operatorname*{argmin}_{f\in\mathbb{R}^N} \frac{1}{2}\|y-\Phi f\|^2 + \lambda \sum_m |\langle f, \psi_m\rangle|. \tag{11.3}$$

This corresponds to the basis pursuit denoising for sparse approximation introduced by Chen, Donoho and Saunders in [12]. The resolution of (11.3) can be perform using an iterative thresholding algorithm as detailed in Section 11.3.

For noiseless measurements $y = \Phi f_0$, one solves a constraint basis pursuit problem

$$f^\star \in \operatorname*{argmin}_{\Phi f = y} \sum_m |\langle f, \psi_m\rangle|.$$

This can be recasted as a convex linear program, which can in turn by solved by various solver such as simplex, interior points, or Douglas-Rachford iterations.

11.3 Proximal Gradient Algorithm

This section details an iterative algorithm that compute a solution of (10.16) for either the TV prior $J = J_{\mathrm{TV}}$ or the sparse ℓ^1 prior, which corresponds respectively to the minimizations (**??**) and (11.3).

This algorithm was derived by several authors, among which [21, 17], and belongs to the general family of forward-backward splitting in proximal iterations [15]. We note that faster algorithms can be used, such as Nesterov scheme [31].

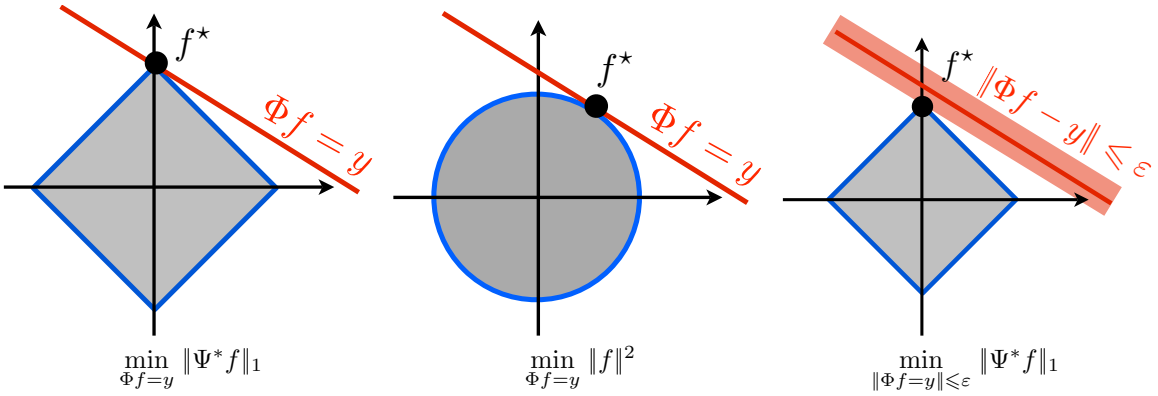


Figure 11.3: Geometry of convex optimizations.

Surrogate functionals. The energy to minimize in (**??**) and (11.3) is written as

$$E(f) = \frac{1}{2}\|\Phi f - y\|^2 + \lambda J(f).$$

The difficulty is the presence of the operator Φ in the ℓ^2 norm, which makes this problem significantly more difficult than the simple denoising by regularization (9.15).

To derive an iterative algorithm, we modify the energy $E(f)$ to obtain a surrogate functional $E(f, f^{(k)})$ whose minimization corresponds to a simpler denoising problem.

Given some guess $f^{(k)} \in \mathbb{R}^N$ of the solution $f^\star$, the surrogate functional is defined as

$$E(f, f^{(k)}) = E(f) - \frac{1}{2}\|\Phi f - \Phi f^{(k)}\|^2 + \frac{1}{2\tau}\|f - f^{(k)}\|^2.$$

One has

$$E(f, f^{(k)}) \geqslant E(f) \quad \text{and} \quad E(f^{(k)}, f^{(k)}) = E(f^{(k)}) \tag{11.4}$$

so that $E(f, f^{(k)})$ is a proxy for the minimization of $E(f)$.

Proximal iterations. A proximal iterative algorithm computes

$$f^{(k+1)} = \operatorname*{argmin}_{f \in \mathbb{R}^N} E(f, f^{(k)}).$$

Property (11.4) guarantees that E is decaying

$$E(f^{(k+1)}) \leqslant E(f^{(k)}).$$

Furtheremore, one has

$$E(f, f^{(k)}) = C + \frac{1}{2\tau}\|f - f^{(k)} + \tau\Phi^*(\Phi f^{(k)} - y)\|^2 + \lambda \sum_m |\langle f, \psi_m \rangle|$$

where C is independent of f. Defining a proximal operator

$$\operatorname{prox}_{\lambda J}(\tilde{f}) = \operatorname*{argmin}_{f \in \mathbb{R}^N} \frac{1}{2}\|\tilde{f} - f\|^2 + \lambda J(f), \tag{11.5}$$

that corresponds to the variational denoiser introduced in (9.15), one thus has

$$f^{(k+1)} = \operatorname{prox}_{\lambda\tau J}\left(\tilde{f}^{(k)}\right) \quad \text{where} \quad \tilde{f}^{(k)} = f^{(k)} - \tau\Phi^*(\Phi f^{(k)} - y).$$

One can prove, see [], that if $\tau < 2/\|\Phi^*\Phi\|_S$, then

$$f^{(k)} \to f^\star.$$

Noiseless case. If $\sigma = 0$, so that one observe noiseless measures $y = \Phi f_0$, an heuristic to compute approximately the solution $f^\star$ of (10.17) is to use a decaying value of $\lambda = \lambda^{(k)}$ during the iterations. One can for instance use $\lambda_k = \lambda_{\max}/k$, although there is no proof of convergence to $f^\star$.

constrained problem. The constrained problem

$$f_\varepsilon^\star \in \operatorname*{argmin}_{\|\Phi f - y\| \leqslant \varepsilon} \sum_m |\langle f, \psi_m \rangle|.$$

is equivalent to the problem (10.16), in the sense that $f^\star$ is a solution of (10.16) for a suitable value of λ that depends on ε. Unfortunately, the correspondence between λ and ε is unknown and depends on y.

An heuristic to automatically find this correspondence is to iteratively update the value of $\lambda = \lambda^{(k)}$

$$\lambda^{(k+1)} = \lambda^{(k)} \frac{\varepsilon}{\|\Phi f^{(k)} - y\|}.$$

Sparse regularization. For the case of $J = J_1$, the proximal denoising operator (11.5) is computed in closed form using a soft thresholding, as already noticed in (11.2).

The resulting proximal iterative algorithm corresponds to the iterative soft thresholding algorithm, that alternates a gradient descent step

$$\tilde{f}^{(k)} = f^{(k)} - \tau \Phi^*(\Phi f^{(k)} - y). \tag{11.6}$$

and soft thresholding

$$f^{(k+1)} = \sum_m S^1_{\lambda\tau}(\langle \tilde{f}^{(k)}, \psi_m \rangle) \psi_m. \tag{11.7}$$

Table **??** details the implementation of this method, when the data is assumed to be sparse in the trivial identity basis.

TV regularization. For the case of $J = J_{\mathrm{TV}}$, the proximal operator (11.5) does not have a closed form solution. One thus has to use inner iteration of Chambolle's scheme (13.3) to compute the proximal map.

11.4 Example: Sparse Deconvolution

11.4.1 Sparse Spikes Deconvolution

Sparse spikes deconvolution makes use of sparsity in the spacial domain, which corresponds to the orthogonal basis of Diracs $\psi_m[n] = \delta[n - m]$. This sparsity was first introduced in the seismic imaging community [], where the signal f_0 represent the change of density in the underground and is assumed to be composed of a few Diracs impulse.

In a simplified linearized 1D set-up, ignoring multiple reflexions, the acquisition of underground data f_0 is modeled as a convolution $y = h \star f_0 + w$, where h is a so-called "wavelet" signal sent in the ground. This should not be confounded with the construction of orthogonal wavelet bases detailed in Chapter **??**, although the term "wavelet" originally comes from seismic imaging.

The wavelet filter h is typically a band pass signal that perform a tradeoff between space and frequency concentration especially tailored for seismic exploration. Figure (11.4) shows a typical wavelet that is a second derivative of a Gaussian, together with its Fourier transform. This shows the large amount of information removed from f during the imaging process.

The sparse ℓ^1 regularization in the Dirac basis reads

$$f^\star = \operatorname*{argmin}_{f \in \mathbb{R}^N} \frac{1}{2}\|f \star h - y\|^2 + \lambda \sum_m |f[m]|.$$

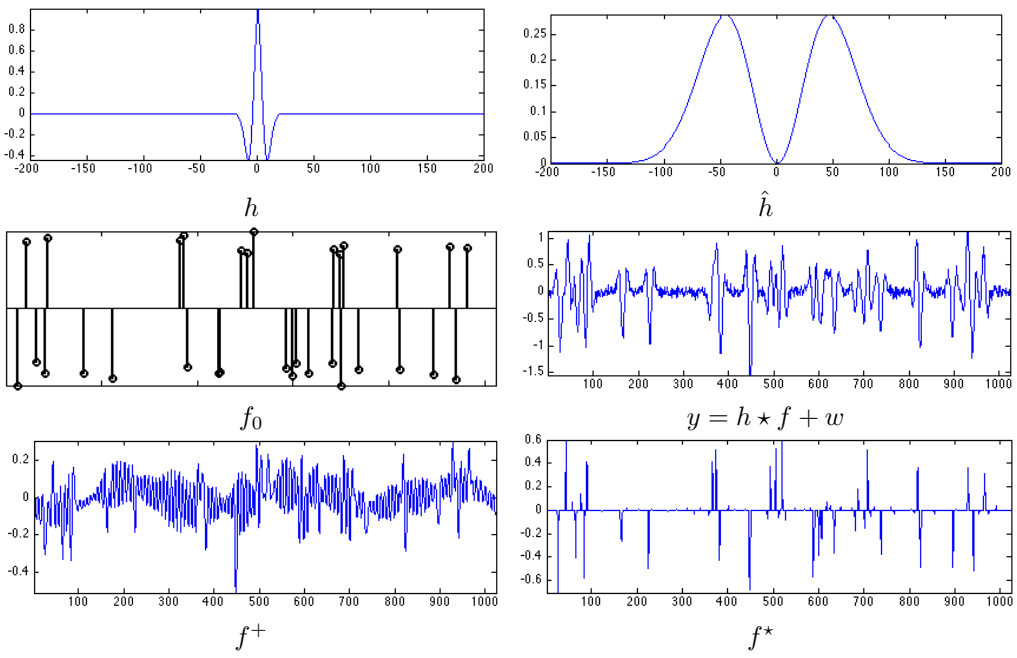


Figure 11.4: Pseudo-inverse and ℓ^1 sparse spikes deconvolution.

Figure 11.4 shows the result of ℓ^1 minimization for a well chosen λ parameter, that was optimized in an oracle manner to minimize the error $\|f^\star - f_0\|$.

The iterative soft thresholding for sparse spikes inversion iterates

$$\tilde{f}^{(k)} = f^{(k)} - \tau h \star (h \star f^{(k)} - y)$$

and

$$f^{(k+1)}[m] = S^1_{\lambda\tau}(\tilde{f}^{(k)}[m])$$

where the step size should obeys

$$\tau < 2/\|\Phi^*\Phi\| = 2/\max_\omega |\hat{h}(\omega)|^2$$

to guarantee convergence. Figure 11.5 shows the progressive convergence of the algorithm, both in term of energy minimization and iterates. Since the energy is not strictly convex, we note that convergence in energy is not enough to guarantee convergence of the algorithm.

11.4.2 Sparse Wavelets Deconvolution

Signal and image acquired by camera always contain some amount of blur because of objects being out of focus, movements in the scene during exposure, and diffraction. A simplifying assumption assumes a spatially invariant blur, so that Φ is a convolution

$$y = f_0 \star h + w.$$

In the following, we consider h to be a Gaussian filter of width $\mu > 0$. The number of effective measurements can thus be considered to be $P \sim 1/\mu$, since Φ nearly set to 0 large enough Fourier frequencies. Table **??** details the implementation of the sparse deconvolution algorithm.

Figures 11.6 and 11.7 shows examples of signal and image acquisition with Gaussian blur.

Sobolev regularization (9.17) improves over ℓ^2 regularization (**??**) because it introduces an uniform smoothing that reduces noise artifact. It however fail to recover sharp edge and thus does a poor job

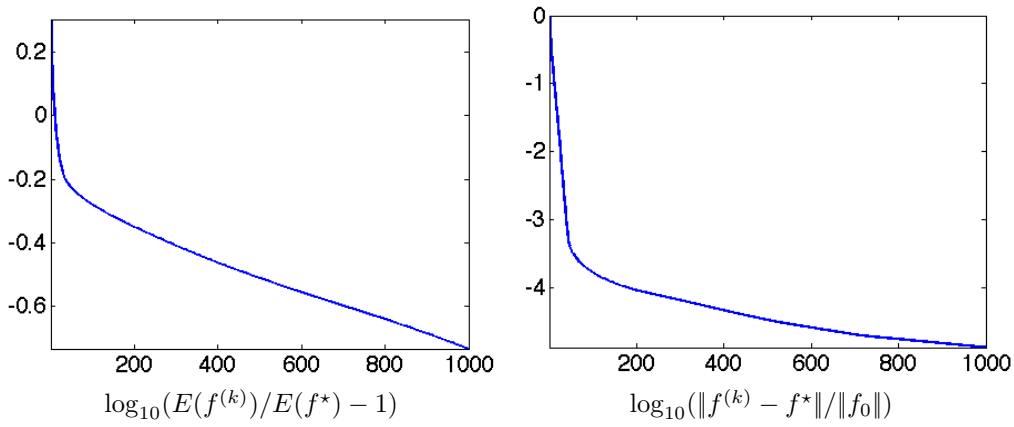


Figure 11.5: Decay of the energy and convergence through the iterative thresholding iterations.

in inverting the operator. To recover sharper transition and edges, one can use either a TV regularization or a sparsity in an orthogonal wavelet basis.

Figure 11.6 shows the improvement obtained in 1D with wavelets with respect to Sobolev. Figure 11.7 shows that this improvement is also visible for image deblurring. To obtain a better result with fewer artifact, one can replace the soft thresholding in orthogonal wavelets in during the iteration (11.7) by a thresholding in a translation invariant tight frame as defined in (8.10).

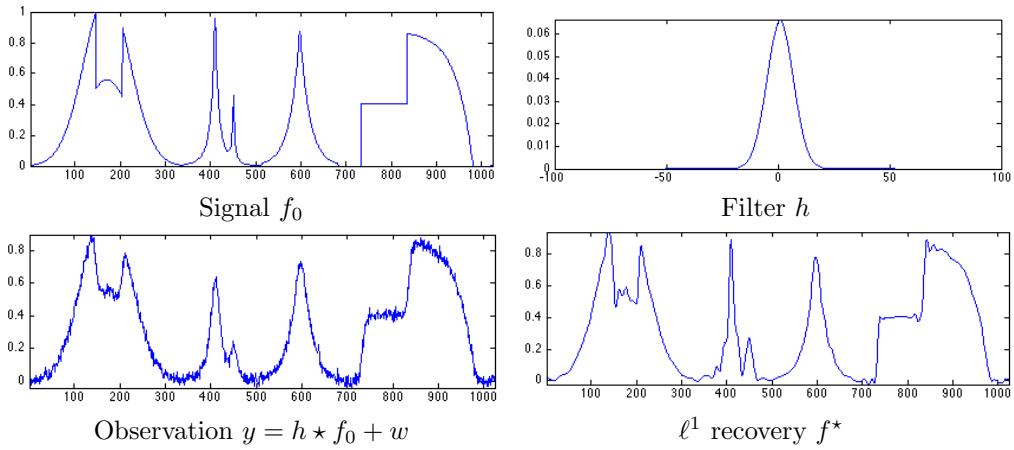


Figure 11.6: Sparse 1D deconvolution using orthogonal wavelets.

Figure 11.8 shows the decay of the SNR as a function of the regularization parameter λ. This SNR is computed in an oracle manner since it requires the knowledge of f_0. The optimal value of λ was used in the reported experiments.

11.4.3 Sparse Inpainting

This section is a follow-up of Section 10.5.2.

To inpaint using a sparsity prior without noise, we use a small value for λ. The iterative thresholding

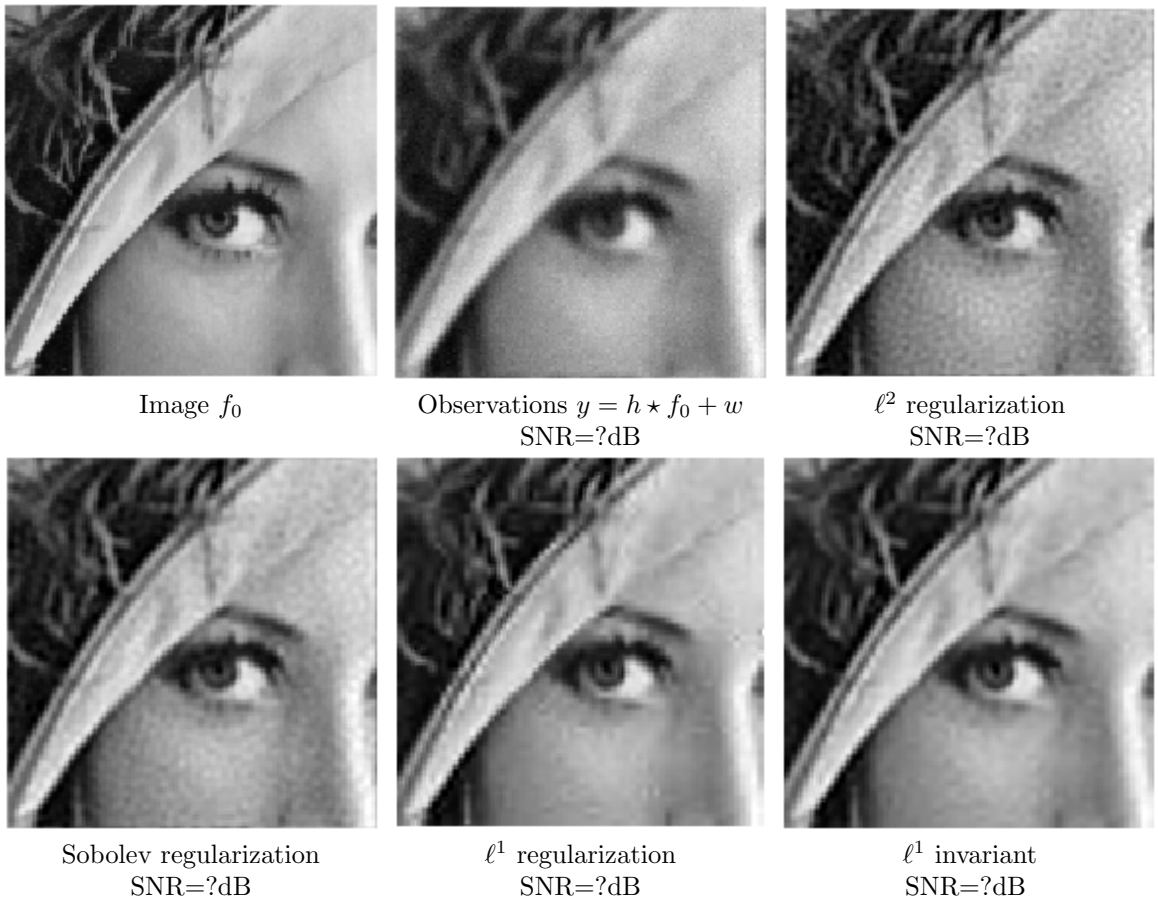


Figure 11.7: Image deconvolution.

algorithm (11.7) is written as follow for $\tau = 1$,

$$f^{(k+1)} = \sum_m S^1_\lambda(\langle P_y(f^{(k)}),\, \psi_m\rangle)\psi_m$$

Figure 11.9 shows the improvevement obtained by the sparse prior over the Sobolev prior if one uses soft thresholding in a translation invariant wavelet frame.

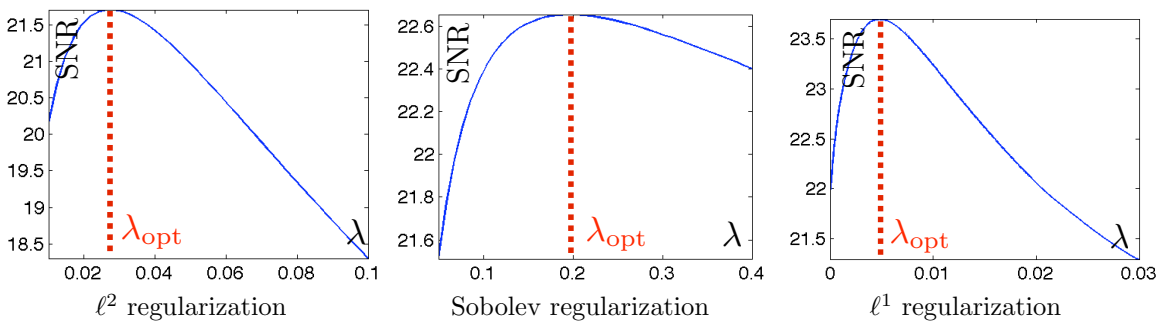


Figure 11.8: SNR as a function of λ.

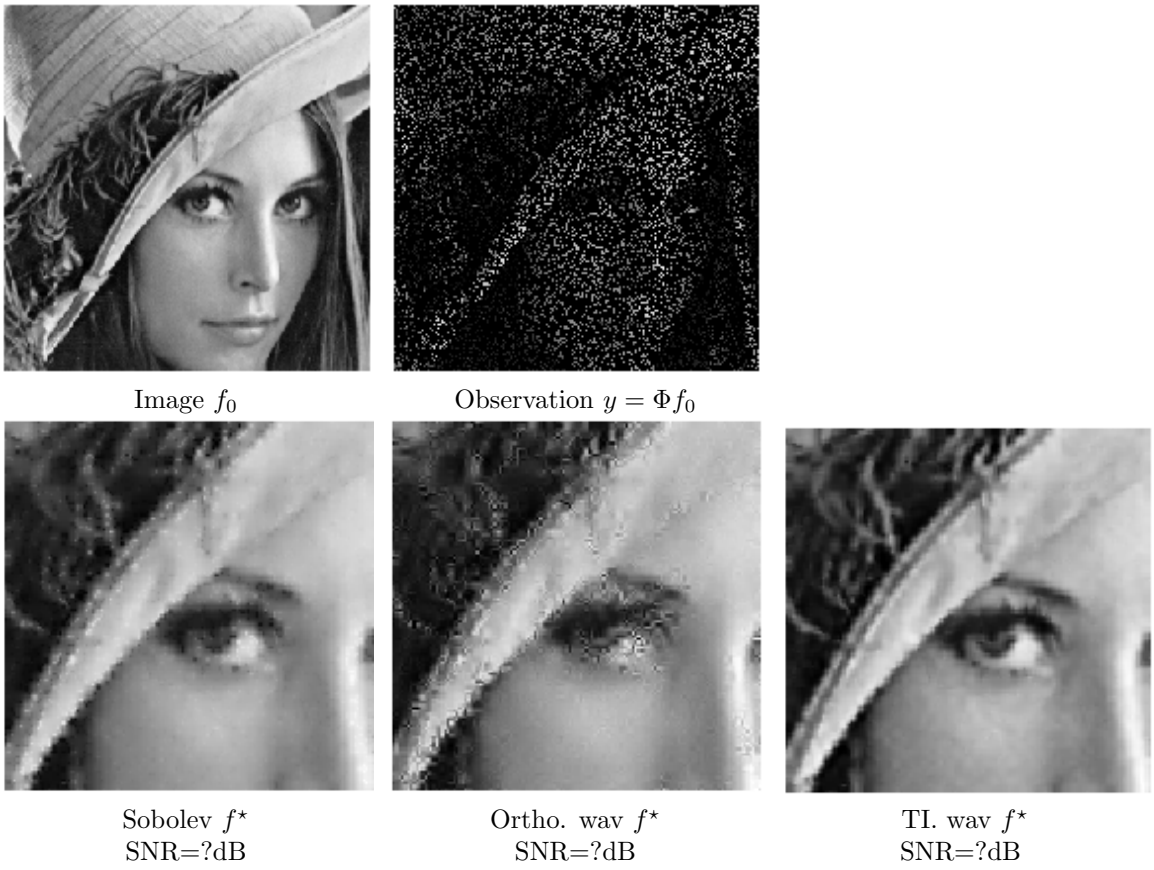


Figure 11.9: Inpainting with Sobolev and sparsity.

Chapter 12

Convex Optimization

TODO.
Main ref: [14, 5]

Chapter 13

Convex Duality

Main ref [10, 11, 5], [32, 4, 3]
TODO.

13.1 Forward-backward on the Dual

Chambolle's algorithm. Chambolle in [9] detail an algorithm to minimize exactly the TV denoising problem

$$f_\lambda^\star = \operatorname*{argmin}_{g\in\mathbb{R}^N} \frac{1}{2}\|f-g\|^2 + \lambda\|g\|_{\mathrm{TV}}. \tag{13.1}$$

It uses a relationship between the vectorial ℓ^1 and ℓ^∞ norms

$$\|v\|_1 = \sum_{m=0}^{N-1} \|v_m\| \quad \text{and} \quad \|v\|_\infty = \max_{0\leqslant m<N} \|v_m\|$$

where each $v_m \in \mathbb{R}^2$ and $v \in \mathbb{R}^{N\times 2}$. One has

$$\|v\|_1 = \max_{\|w\|_\infty\leqslant 1} \langle w,\, u\rangle$$

which allows one to re-write the optimization (13.1) as

$$\min_{g\in\mathbb{R}^N} \max_{\|w\|_\infty\leqslant 1} \frac{1}{2}\|f-g\|^2 + \lambda\langle w,\, \nabla g\rangle.$$

Exchanging the roles of the min and the max, one proves that the solution of (13.1) is re-written as

$$f_\lambda^\star = f + \lambda\,\mathrm{div}(w^\star) \tag{13.2}$$

where

$$w^\star \in \operatorname*{argmin}_{\|w\|_\infty\leqslant 1} \|f+\lambda\,\mathrm{div}(w^\star)\|^2. \tag{13.3}$$

The convex optimization problem (13.3) computes a dual vector field $w^\star \in \mathbb{R}^{N\times 2}$, from which the denoised image is recovered using (13.2).

The dual problem (13.3) is the minimization of a quadratic functional subject to a convex ℓ^∞ constraint. It can thus be solved using for instance a projected gradient descent

$$w_m^{(k+1)} = \frac{\tilde{w}_m^{(k)}}{\max(|\tilde{w}_m^{(k)}|,1)} \quad \text{where} \quad \tilde{w}^{(k)} = w^{(k)} + \tau\nabla(f/\lambda + \mathrm{div}(w^{(k)})).$$

If the gradient step size satisfy $0 < \tau < 1/4$, one can prove that

$$f + \lambda \operatorname{div}(w^{(k)}) \longrightarrow f_\lambda^\star \quad \text{when} \quad k \to +\infty.$$

Chapter 14

Compressed Sensing

Main ref: [29, 23, 37]
TODO.

14.1 Motivation and Potential Applications

14.2 Dual Certificate Theory and Non-Uniform Guarantees

Polytopes, descent cone and Gaussian width.

14.3 RIP Theory for Uniform Guarantees

14.3.1 RIP Constants

The RIP constant δ_s of a matrix $\Phi \in \mathbb{R}^{P\times N}$ is defined as

$$\forall\, z \in \mathbb{R}^N, \quad \|z\|_0 \leqslant s \quad \Longrightarrow \quad (1-\delta_s)\|z\|^2 \leqslant \|\Phi z\|^2 \leqslant (1+\delta_s)\|z\|^2 \tag{14.1}$$

14.3.2 RIP implies stable recovery

RIP implies dual certificate guarantees.

In the following, we fixe a vector $x \in \mathbb{R}^N$ and denote $y = \Phi x + w$ the measurement, with $\|w\| \leqslant \varepsilon$. We denote $x_s \in \mathbb{R}^N$ the best s-term approximation of x, obtained by only keeping the s largest coefficients in magnitude from x and setting the others to 0.

We consider a solution $x^\star$ of

$$\min_{\|\Phi\tilde{x}-y\|\leqslant\varepsilon} \|\tilde{x}\|_1.$$

This note recall the proof from [7] of the following theorem

Theorem 34 ([7]). *If $\delta_{2s} \leqslant \sqrt{2}-1$ then there exists C_0, C_1 such that*

$$\|x^\star - x\|_1 \leqslant \frac{C_0}{\sqrt{s}}\|x_s - x\| + C_1\varepsilon.$$

The remaining of this section is devoted to proving this theorem.

Notations. We denote in the following $h = x^\star - x$ and denote T_0 the largest s coefficients of x in magnitude (so that $x_s = x_{T_0}$), T_1 the s largest coefficients of $h_{T_0^c}$, T_2 the following s largest coefficients of $h_{T_0^c}$ and so on. We denote $T = T_0 \cup T_1$ which is an index set of size $2s$.

Lemma 3. *One has*

$$\sum_{j \geqslant 2} \|h_{T_j}\| \leqslant \frac{1}{\sqrt{s}} \|h_{T_0^c}\|_1$$

Proof. By the definition of T_j for $j \geqslant 2$, one has, for all $j \geqslant 2$

$$\forall\, i \in T_{j-1}, \quad \|h_{T_j}\|_\infty \leqslant h_i,$$

and hence

$$\|h_{T_j}\|_\infty \leqslant \frac{1}{s} \|h_{T_{j-1}}\|_1.$$

This proves that

$$\|h_{T_j}\| \leqslant \sqrt{s} \|h_{T_j}\|_\infty \leqslant \frac{1}{\sqrt{s}} \|h_{T_{j-1}}\|_1$$

and thus

$$\sum_{j \geqslant 2} \|h_{T_j}\| \leqslant \frac{1}{\sqrt{s}} \sum_{j \geqslant 1} \|h_{T_j}\|_1 = \frac{1}{\sqrt{s}} \|h_{T_0^c}\|_1.$$

□

Lemma 4. *One has*

$$\|h_{T_0^c}\|_1 \leqslant \|h_{T_0}\|_1 + 2\|x_{T_0^c}\|_1$$

Proof. One has

$$\begin{aligned}
\|x\|_1 &\geqslant \|x + h\| && \text{because } x^\star \text{ is a minimizer} \\
&= \|(x+h)_{T_0}\|_1 + \|(x+h)_{T_0^c}\|_1 \\
&\geqslant \|x_{T_0}\|_1 - \|h_{T_0}\|_1 + \|h_{T_0^c}\|_1 - \|x_{T_0^c}\|_1 && \text{using the triangular inequality.}
\end{aligned}$$

Decomposing the left hand size $\|x\|_1 = \|x_{T_0}\|_1 + \|x_{T_0^c}\|_1$, one obtains the result. □

Lemma 5. *If z and z' have disjoints supports and $\|z\| \leqslant s$ and $\|z'\|_0 \leqslant s$,*

$$|\langle \Phi z,\, \Phi z' \rangle| \leqslant \delta_{2s} \|z\| \|z'\|.$$

Proof. Using the RIP (14.1) since $z \pm z'$ has support of size $2s$ and the fact that $\|z \pm z'\|^2 = \|z\|^2 + \|z'\|^2$, one has

$$(1 - \delta_{2s}) \left(\|z\|^2 + \|z'\|^2 \right) \leqslant \|\Phi z \pm \Phi z'\|^2 \leqslant (1 + \delta_{2s}) \left(\|z\|^2 + \|z'\|^2 \right).$$

One thus has using the parallelogram equality

$$|\langle \Phi z,\, \Phi z' \rangle| = \frac{1}{4} |\|\Phi z + \Phi z'\|^2 - \|\Phi z - \Phi z'\|^2|^2 \leqslant \delta_{2s} \|z\| \|z'\|.$$

□

Theorem 34 requires bounding $\|h\|$. We bound separately $\|h_T\|$ and $\|h_{T^c}\|$.

Part 1: bounding $\|h_T\|$. One has

$$\begin{aligned}
\|h_{T^c}\| = \|\sum_{j\geqslant 2} h_{T_j}\| &\leqslant \sum_{j\geqslant 2}\|h_{T_j}\| && \text{using the triangular inequality}\\
&\leqslant \frac{1}{\sqrt{s}}\|h_{T_0^c}\|_1 && \text{using Lemma 3}\\
&\leqslant \frac{1}{\sqrt{s}}\|h_{T_0}\|_1 + \frac{2}{\sqrt{s}}\|x_{T_0^c}\|_1 && \text{using Lemma 4}\\
&\leqslant \frac{1}{\sqrt{s}}\|h_{T_0}\|_1 + 2e_0 && \text{denoting } e_0 = \frac{1}{\sqrt{s}}\|x_{T_0^c}\|_1\\
&\leqslant \|h_{T_0}\| + 2e_0 && \text{using Cauchy-Schwartz}\\
&\leqslant \|h_T\| + 2e_0 && \text{because } T_0 \subset T.
\end{aligned}$$

The final bound reads

$$\|h_{T^c}\| \leqslant \|h_T\| + 2e_0. \tag{14.2}$$

Part 2: bounding $\|h_{T^c}\|$. One has

$$\begin{aligned}
\|h_T\|^2 &\leqslant \frac{1}{1-\delta_{2s}}\|\Phi h_T\|^2 && \text{using the RIP (14.1)}\\
&= \frac{A-B}{1-\delta_{2s}} && \text{using } \Phi h_T = \Phi h - \sum_{j\geqslant 2}\Phi h_{T_j},
\end{aligned}$$

where we have introduced

$$A = \langle \Phi h_T,\, \Phi h\rangle \quad \text{and} \quad B = \langle \Phi h_T,\, \sum_{j\geqslant 2}\Phi h_{T_j}\rangle.$$

One has

$$\begin{aligned}
|A| &\leqslant \|\Phi h_T\|\|\Phi h\| && \text{using Cauchy-Schwartz}\\
&\leqslant \sqrt{1+\delta_{2s}}\|h_T\|\|\Phi h\| && \text{using the RIP (14.1)}\\
&\leqslant \sqrt{1+\delta_{2s}}\|h_T\| 2\varepsilon && \text{using } \|\Phi h\| \leqslant \|\Phi x - y\| + \|\Phi x^\star - y\| \leqslant 2\varepsilon
\end{aligned}$$

The final bound reads

$$|A| \leqslant 2\varepsilon\sqrt{1+\delta_{2s}}\|h_T\| \tag{14.3}$$

One has

$$\begin{aligned}
|B| &\leqslant |\langle \Phi h_{T_0}, \sum_{j\geqslant 2}\Phi h_{T_j}\rangle| + |\langle \Phi h_{T_1}, \sum_{j\geqslant 2}\Phi h_{T_j}\rangle| && \text{using the triangular inequality}\\
&\leqslant \sum_{j\geqslant 2} |\langle \Phi h_{T_0},\, \Phi h_{T_j}\rangle| + |\langle \Phi h_{T_0},\, \Phi h_{T_j}\rangle| && \text{using the triangular inequality}\\
&\leqslant \sum_{j\geqslant 2} \delta_{2s}\|h_{T_0}\|\|h_{T_j}\| + \delta_{2s}\|h_{T_1}\|\|h_{T_j}\| && \text{using Lemma 5}\\
&= \delta_{2s}(\|h_{T_0}\| + \|h_{T_1}\|)\sum_{j\geqslant 2}\|h_{T_j}\|\\
&\leqslant \delta_{2s}\sqrt{2}\|h_T\|\sum_{j\geqslant 2}\|h_{T_j}\| && T_0 \text{ and } T_1 \text{ are disjoints}\\
&\leqslant \frac{\sqrt{2}\ \delta_{2s}}{\sqrt{s}}\|h_T\|\|h_{T_0^c}\|_1 && \text{using Lemma 3}
\end{aligned}$$

The final bound reads

$$|B| \leqslant \frac{\sqrt{2}\ \delta_{2s}}{\sqrt{s}} \|h_T\| \|h_{T_0^c}\|_1. \tag{14.4}$$

Putting together (14.3) and (14.4) one obtains

$$\|h_T\|^2 \leqslant \frac{\|h_T\|}{1-\delta_{2s}} \left(\sqrt{1+\delta_{2s}} 2\varepsilon + \frac{2}{\sqrt{s}} \delta_{2s} \|h_{T_0^c}\|_1 \right)$$

thus

$$\begin{aligned}
\|h_T\| &\leqslant \alpha\varepsilon + \frac{\rho}{\sqrt{s}} \|h_{T_0^c}\|_1 && \text{denoting} \begin{cases} \alpha = 2\frac{\sqrt{1+\delta_{2s}}}{1-\delta_{2s}}, \\ \rho = \frac{\sqrt{2}\delta_{2s}}{1-\delta_{2s}}. \end{cases} \\
&\leqslant \alpha\varepsilon + \frac{\rho}{\sqrt{s}} \|h_{T_0}\|_1 + \frac{2\rho}{\sqrt{s}} \|x_{T_0^c}\|_1 && \text{using Lemma 4} \\
&\leqslant \alpha\varepsilon + \rho \|h_{T_0}\| + 2\rho e_0 && \text{using Cauchy-Schwartz} \\
&\leqslant \alpha\varepsilon + \rho \|h_T\| + 2\rho e_0 && \text{because } T_0 \subset T.
\end{aligned}$$

Note that since $\delta_{2s} < \sqrt{2} - 1$, one has $\rho < 1$. This implies

$$\|h_T\| \leqslant \frac{\alpha}{1-\rho}\varepsilon + \frac{2\rho}{1-\rho} e_0. \tag{14.5}$$

Conclusion. One has

$$\begin{aligned}
\|h\| &\leqslant \|h_T\| + \|h_{T^c}\| && \text{using the triangular inequality} \\
&\leqslant 2\|h_T\| + 2e_0 && \text{using (14.2)} \\
&\leqslant \frac{2\alpha}{1-\rho}\varepsilon + 2\frac{1+\rho}{1-\rho} e_0 && \text{using (14.5)}
\end{aligned}$$

which proves the theorem.

14.3.3 Gaussian Matrices RIP

14.3.4 Fourier sampling RIP

Chapter 15

Machine Learning

15.1 Supervised vs Unsupervised Learning

TOOD: describe the general settings and problems.

15.2 PCA, Nearest-Neighbors Classification and Clustering

15.2.1 Dimenionality Reduction and PCA

We detail Principal Component Analysis (dimentionality reduction), supervised classification using nearest neighbors and unsupervised clustering using k-means.

We use here the famous IRIS dataset of Fisher. The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimetres.

The feature are stored as the row of a matrix $X \in \mathbb{R}^{n \times p}$

n is the number of samples, p is the dimensionality of the features, k is the number of classes.

In order to display in 2-D or 3-D the data, dimensionality reduction is needed. The simplest method is the Principal Component Analysis (PCA), which perform an orthogonal linear projection on the principal axis (eigenvector) of the covariance matrix.

We compute the empirical mean

$$m = \frac{1}{n} \sum_{i=1}^{n} x_i \in \mathbb{R}^p$$

and covariance

$$C = \frac{1}{n} \sum_{i=1}^{n} (x_i - m)(x_i - m)^\top \in \mathbb{R}^{p \times p}.$$

Denoting $\tilde{X} = X - 1_p m^\top$, one has $C = \tilde{X}^\top \tilde{X}$.

We compute PCA ortho-basis using the SVD decomposition

$$\tilde{X} = U \operatorname{diag}(d) V$$

where $U \in \mathbb{R}^{n \times p}$ and $V \in \mathbb{R}^{p \times p}$ have orthonormal columns. V are the principal directions of variance and are order by decreasing variances.

We then compute the feature in the PCA basis, $z_i = V^\top (x_i - m)$, stored in matrix format as $Z = \tilde{X} V$.

One can plot the singular values of the covariances, which corresponds to the standard deviation of the data along the principal directions.

The first dimensions of the z_i are the optimal way to linearly embed the data in a low dimensional space. This can be used for display in 2-D using the first two dimension. One can do a similar display in 3-D.

15.2.2 Supervised Learning: Nearest Neighbor Classification

Probably the simplest method for supervised classification is Nearest Neighbor (R-NN), where R is a parameter indexing the number of neighbor. Increasing R is important to cope with noise and obtain smoother decision boundary, and hence better generalization performance.

The class predicted for a point x is the one which is the most represented among the R points $(x_i)_i$ which are the closed to x.

The data needs to be split into training and testing.

One then compute Euclidean distance between some x and all other $x_{1,j}$ in the training set. Then sort the distance and generate the list of sorted classes $y_\sigma = (y_{\sigma(i)})_i$. This generate an indexing σ (a permutation of $\{1, \ldots, n\}$) such that

$$\|x - x_{\sigma(1)}\| \leqslant \|x - x_{\sigma(2)}\| \leqslant \ldots \leqslant \|x - x_{\sigma(n)}\|.$$

For a given R, one can compute the histogram of class apparition

$$h_\ell \stackrel{\text{def.}}{=} \frac{1}{R} \{i \,;\, \sigma(i) \in \{1, \ldots, R\}\} = \sharp \sigma^{-1}(\{1, \ldots, R\}).$$

The decision class for x is then the maximum of the histogram

$$c(x) \stackrel{\text{def.}}{=} \operatorname*{argmax}_\ell h_\ell$$

One can display the histogram $(h_\ell)_\ell$ of repartition of class indexes as R grows. Then perform the NN classification for all the points in the test set, and for varying R. And show how the classification score S (number of correctly classified points) evolves with R.

One can also display, as a function of the position in 2-D PCA space, the class output by the R-NN method when applied in 2-D.

15.2.3 Unsupervised Learning: k-means

In an un-supervised setting, the class information y is not available. The basic problem is then to recover class information from the knowledge of x only. This corresponds to the clustering problem.

The most basic algorithm is the k-means, which tries to recover the class index $\bar{y}_i = \ell$ from the distance $\|x_i - c_\ell\|$ between the feature point x_i and the class centroid c_ℓ (which are the unknown of the problem).

It does so by minimizing the following non-convex energy

$$\min_{(c_\ell)_\ell} \sum_i \min_\ell \|x_i - c_\ell\|^2$$

We first initialize the class centroids $(c_\ell)_\ell$ at random among the points. They are stored in as the row of a matrix $C \in \mathbb{R}^{k \times p}$.

The k-means algorithm iterate between first determining the class of each point using the distance to the centroids

$$\forall i \in \{1, \ldots, n\}, \quad \bar{y}_i \leftarrow \operatorname*{argmin}_\ell \|x_i - c_\ell\|.$$

One can display the centroids and the classes using colors. This corresponds to a Voronoi diagram segmentation in the high dimensional space, but here the display is done in 2D.

The second step of the k-means algorithm is to update the centroids position to be the mean of the points inside each class

$$\forall \ell \in \{1, \ldots, k\}, \quad c_\ell \leftarrow \frac{\sum_{i : y_i = \ell} x_i}{\sharp\{i : y_i = \ell\}}.$$

One can then perform several step of the k-means algorithm. And display the histogram of (true, i.e. according to y) class inside each estimated class (i.e. according to $\bar{y}$).

It is possible to implement better initialization strategies such as farthest point sampling or k-means++.

15.3 Linear Regression and Kernel Methods

We now study linear regression method, and its non-linear variant using kernelization.

15.3.1 Linear Regression

We test the method on the Boston house prices dataset, consisting in $n = 506$ samples with features $x_i \in \mathbb{R}^p$ in dimension $p = 13$. The goal is to predict the price value $y_i \in \mathbb{R}$.

We separate the features X from the data y to predict information. n is the number of samples, p is the dimensionality of the features,

In order to display in 2-D or 3-D the data, dimensionality is needed. The simplest method is the principal component analysis, which perform an orthogonal linear projection on the principal axsis (eigenvector) of the covariance matrix.

We look for a linear relationship $y_i = \langle w, x_i \rangle$ written in matrix format $y = Xw$ where the rows of $X \in \mathbb{R}^{n \times p}$ stores the features $x_i \in \mathbb{R}^p$.

Since here $n > p$, this is an over-determined system, which can solved in the least square sense

$$\min_w \|Xw - y\|^2,$$

whose solution is given using the Moore-Penrose pseudo-inverse

$$w = (X^\top X)^{-1} X^\top y.$$

Regularization is obtained by introducing a penalty. It is often called ridge regression, and is defined as

$$\min_w \|Xw - y\|^2 + \lambda \|w\|^2$$

where $\lambda > 0$ is the regularization parameter.

The solution is given using the following equivalent formula

$$w = (X^\top X + \lambda \mathrm{Id}_p)^{-1} X^\top y,$$

$$w = X^\top (X X^\top + \lambda \mathrm{Id}_n)^{-1} y,$$

When $p < n$ (which is the case here), the first formula should be preferred.

In contrast, when the dimensionality p of the feature is very large and there is little data, the second is faster. Furthermore, this second expression is generalizable to Kernel Hilbert space setting, corresponding possibly to $p = +\infty$ for some kernels.

15.3.2 Kernelized Ridge Regression

In order to perform non-linear and non-parametric regression, it is possible to use kernelization. It is non-parametric in the sense that the number of parameter grows with the number n of samples (while for the initial linear method, the number of parameter is p). This allows in particular to generate estimator of arbitrary complexity.

Given a kernel $\kappa(x, z) \in \mathbb{R}$ defined for $(x, z) \in \mathbb{R}^p \times \mathbb{R}^p$, the kernelized method replace the linear approximation functional $f(x) = \langle x, w \rangle$ by a sum of kernel centered on the samples

$$f_h(x) = \sum_{i=1}^n h_i k(x_i, x)$$

where $h \in \mathbb{R}^n$ is the unknown vector of weight to find.

When using the linear kernel $\kappa(x, y) = \langle x, y \rangle$, one retrieves the previously studied linear method.

The gaussian kernel is the most well known and used kernel

$$\kappa(x,y) \overset{\text{def.}}{=} e^{-\frac{\|x-y\|^2}{2\sigma^2}}.$$

The bandwidth parameter $\sigma > 0$ is crucial and controls the locality of the model. It is typically tuned through cross validation.

Once evaluated on grid points, the kernel define a matrix

$$K = (\kappa(x_i, x_j))_{i,j=1}^n \in \mathbb{R}^{n\times n}.$$

The weights $h \in \mathbb{R}^n$ are solutions of

$$\min_h \|Kh - y\|^2 + \lambda \langle Kh,\, h\rangle$$

and hence can be computed by solving a linear system

$$h = (K + \lambda \mathrm{Id}_n)^{-1} y$$

15.4 Logistic Classification

We now detail the logistic classification method (for 2 classes and multi-classes).

Note thatlLogisitic classification is actually called "logistic regression" in the literature, but it is in fact a classification method.

15.4.1 Two Classes Logistic Classification

Logistic classification is, with support vector machine (SVM), the baseline method to perform classification. Its main advantage over SVM is that is is a smooth minimization problem, and that it also output class probabity, offering a probabilistic interpretation of the classification.

To understand the behavior of the method, we generate synthetic data distributed according to a mixture of Gaussian with an overlap governed by an offset ω. Here classes indexes are set to $y_i \in \{-1, 1\}$ to simplify the equations.

Logistic classification minimizes a logistic loss in place of the usual ℓ^2 loss for regression

$$\min_w E(w) \overset{\text{def.}}{=} \frac{1}{n}\sum_{i=1}^n L(\langle x_i,\, w\rangle, y_i)$$

where the logistic loss reads

$$L(s,y) \overset{\text{def.}}{=} \log(1 + \exp(-sy))$$

This corresponds to a smooth convex minimization. If X is injective, this is also strictly convex, hence it has a single global minimum.

One can compare the binary (ideal) 0-1 loss, the logistic loss and the hinge loss (the one used for SVM).

This can be interpreted as a maximum likelihood estimator when one models the probability of belonging to the two classes for sample x_i as

$$h(x_i) \overset{\text{def.}}{=} (\theta(x_i), 1-\theta(x_i)) \quad \text{where} \quad \theta(s) \overset{\text{def.}}{=} \frac{e^s}{1+e^s} = (1+e^{-s})^{-1}$$

Re-writting the energy to minimize

$$E(w) = \mathcal{L}(Xw, y) \quad \text{where} \quad \mathcal{L}(s,y) = \frac{1}{n}\sum_i L(s_i, y_i),$$

its gradient reads

$$\nabla E(w) = X^\top \nabla \mathcal{L}(Xw, y) \quad \text{where} \quad \nabla \mathcal{L}(s, y) = \frac{y}{n} \odot \theta(-y \odot s),$$

where $\odot$ is the pointwise multiplication operator, i.e. `.*` in Matlab.

In order to improve performance, it is important (especially in low dimension p) to add a constant bias term $w_{p+1} \in \mathbb{R}$, and replace $\langle x_i, w \rangle$ by $\langle x_i, w \rangle + w_{p+1}$. This is equivalently achieved by adding an extra $(p+1)^{\text{th}}$ dimension equal to 1 to each x_i, which we do using a convenient macro.

With this added bias term, once $w_{\ell=0} \in \mathbb{R}^{p+1}$ initialized (for instance at 0_{p+1}),

one step of gradient descent reads

$$w_{\ell+1} = w_\ell - \tau_\ell \nabla E(w_\ell).$$

One can then implement a gradient descent

$$w_{\ell+1} = w_\ell - \tau_\ell \nabla E(w_\ell).$$

One can display the data overlaid on top of the classification probability, this highlight the separating hyperplane $\{x \,;\, \langle w, x \rangle = 0\}$.

15.4.2 Kernelized Logistic Classification

Logistic classification tries to separate the classes using a linear separating hyperplane $\{x \,;\, \langle w, x \rangle = 0\}$.

In order to generate a non-linear descision boundary, one can replace the parametric linear model by a non-linear non-parametric model, thanks to kernelization. It is non-parametric in the sense that the number of parameter grows with the number n of sample (while for the basic method, the number of parameter is p. This allows in particular to generate decision boundary of arbitrary complexity.

The downside is that the numerical complexity of the method grows (at least) quadratically with n.

The good news however is that thanks to the theory of reproducing kernel Hilbert spaces (RKHS), one can still compute this non-linear decision function using (almost) the same numerical algorithm.

Given a kernel $\kappa(x, z) \in \mathbb{R}$ defined for $(x, z) \in \mathbb{R}^p$, the kernelized method replace the linear decision functional $f(x) = \langle x, w \rangle$ by a sum of kernel centered on the samples

$$f_h(x) = \sum_{i=1}^{p} h_i k(x_i, x)$$

where $h \in \mathbb{R}^n$ is the unknown vector of weight to find.

When using the linear kernel $\kappa(x, y) = \langle x, y \rangle$, one retrieves the previously studied linear method.

The gaussian kernel is the most well known and used kernel

$$\kappa(x, y) \overset{\text{def.}}{=} e^{-\frac{\|x-y\|^2}{2\sigma^2}}.$$

The bandwidth parameter $\sigma > 0$ is crucial and controls the locality of the model. It is typically tuned through cross validation.

Once evaluated on grid points, the kernel define a matrix

$$K = (\kappa(x_i, x_j))_{i,j=1}^{n} \in \mathbb{R}^{n \times n}.$$

Valid kernels are those that gives rise to positive symmetric matrices K. The linear and Gaussian kernel are valid kernel functions. Other popular kernels include the polynomial kernel $\langle x, y \rangle^a$ for $a \geqslant 1$ and the Laplacian kernel $\exp(-\|x-y\|^2/\sigma)$.

The kernelized Logistic minimization reads

$$\min_h F(h) \overset{\text{def.}}{=} \mathcal{L}(Kh, y).$$

This minimization can be related to an infinite dimensional optimization problem where one minimizes directly over the function f. This is shown to be equivalent to the above finite-dimenisonal optimization problem thanks to the theory of RKHS. One can then use a gradient descent to minimize $F(h)$.

Once this optimal h has been found, class probability at a point x are obtained as

$$(\theta(f_h(x)), 1-\theta(f_h(x)))$$

where f_h has been defined above.

One can separate the dataset into a training set and a testing set. Evaluate the classification performance for varying σ. Try to introduce regularization and minmize

$$\min_h F(h) \stackrel{\text{def.}}{=} \mathcal{L}(Kh, y) + \lambda R(h)$$

where for instance $R = \|\cdot\|_2^2$ or $R = \|\cdot\|_1$.

15.4.3 Multi-Classes Logistic Classification

The logistic classification method is extended to an arbitrary number k of classes by considering a family of weight vectors $(w_\ell)_{\ell=1}^k$, which are conveniently stored as columns of matrix $W \in \mathbb{R}^{p\times k}$.

This allows to model probabilistically the belonging of a point $x \in \mathbb{R}^p$ to a the classes using an exponential model

$$h(x) = \left(\frac{e^{-\langle x,\, w_\ell\rangle}}{\sum_m e^{-\langle x,\, w_m\rangle}} \right)_\ell$$

This vector $h(x) \in [0,1]^k$ describes the probability of x belonging to the different classes, and $\sum_\ell h(x)_\ell = 1$.

The computation of w is obtained by solving a maximum likelihood estimator

$$\max_{w\in\mathbb{R}^k} \frac{1}{n}\sum_{i=1}^n \log(h(x_i)_{y_i})$$

where we recall that $y_i \in \{1,\ldots,k\}$ is the class index of point x_i.

This is conveniently rewritten as

$$\min_w \sum_i \mathrm{LSE}(XW)_i - \langle XW,\, D\rangle$$

where $D \in \{0,1\}^{n\times k}$ is the binary class index matrices

$$D_{i,\ell} = \left\{ \begin{array}{l} 1 \quad \text{if} \quad y_i = \ell, \\ 0\text{otherwise.} \end{array} \right.$$

and LSE is the log-sum-exp operator

$$\mathrm{LSE}(S) = \log\left(\sum_\ell \exp(S_{i,\ell}) \right) \in \mathbb{R}^n.$$

The computation of LSE is unstable for large value of $S_{i,\ell}$ (numerical overflow, producing NaN), but this can be fixed by substracting the largest element in each row, since $\mathrm{LSE}(S+a) = \mathrm{LSE}(S) + a$ if a is constant along rows. This is the LSE trick.

The gradient of the LSE operator is the soft-max operator

$$\nabla\mathrm{LSE}(S) = \mathrm{SM}(S) \stackrel{\text{def.}}{=} \left(\frac{e^{S_{i,\ell}}}{\sum_m e^{S_{i,m}}} \right)$$

Similarly to the LSE, it needs to be stabilized.

We load a dataset of n images of size $p = 8 \times 8$, representing digits from 0 to 9 (so there are $k = 10$ classes). Separate the features X from the data y to predict information. n is the number of samples, p is the dimensionality of the features, k the number of classes. One can display a few samples digits

We compute the D matrix and define the energy $E(W)$. Then define its gradients

$$\nabla E(W) = \frac{1}{n} X^\top (\mathrm{SM}(XW) - D).$$

and one can use a gradient descent

$$W_{\ell+1} = W_\ell - \tau_\ell \nabla E(W_\ell).$$

We can evaluate class probability associated to weight vectors on this grid.

Chapter 16

Deep Learning

Before detailing deep architecture and their use, we start this chapter by presenting two essential computational tools that are used to train these model: stochastic optimization methods and automatic differentiation. In practice, they work hand-in-hand to be able to learn painlessly complicated non-linear model on large-scale datasets.

16.1 Stochastic Optimization

We detail stochastic Gradient Descent, with an application to the binary logistic classification problem.

We set the classes indexes to be $\{-1, +1\}$, and remove empty features, normalize X. n is the number of samples, p is the dimensionality of the features,

16.1.1 Batch Gradient Descent (BGD)

We first recall the usual deterministic (batch) gradient descent (BGD) on the problem of supervised logistic classification.

We refer to the dedicated section on logistic classification for background and more details about the derivations of the energy and its gradient.

Logistic classification aims at solving the following convex program

$$\min_w E(w) \stackrel{\text{def.}}{=} \frac{1}{n} \sum_{i=1}^{n} L(\langle x_i, w \rangle, y_i)$$

where the logistic loss reads

$$L(s, y) \stackrel{\text{def.}}{=} \log(1 + \exp(-sy))$$

We can define energy E and its gradient ∇E and use a vanilla gradient descent

$$w_{\ell+1} = w_\ell - \tau_\ell \nabla E(w_\ell).$$

16.1.2 Stochastic Gradient Descent (SGD)

As any empirical risk minimization procedure, the logistic classification minimization problem can be written as

$$\min_w E(w) = \frac{1}{n} \sum_i E_i(w) \quad \text{where} \quad E_i(w) = L(\langle x_i, w \rangle, y_i).$$

For very large n (which could in theory even be infinite, in which case the sum needs to be replaced by an expectation or equivalenty an integral), computing ∇E is prohebitive. It is possible instead to use a

stochastic gradient descent (SGD) scheme

$$w_{\ell+1} = w_\ell - \tau_\ell \nabla E_{i(\ell)}(w_\ell)$$

where, for each iteration index ℓ, $i(\ell)$ is drawn uniformly at random in $\{1, \dots, n\}$. Note that here

$$\nabla E_i(w) = x_i \nabla L(\langle x_i, w \rangle, y_i) \quad \text{where} \quad \nabla L(u, v) = v \odot \theta(-u)$$

Note that each step of a batch gradient descent has complexity $O(np)$, while a step of SGD only has complexity $O(p)$. SGD is thus advantageous when n is very large, and one cannot afford to do several passes through the data. In some situation, SGD can provide accurate results even with $\ell \ll n$, exploiting redundancy between the samples.

A crucial question is the choice of step size schedule τ_ℓ. It must tends to 0 in order to cancel the noise induced on the gradient by the stochastic sampling. But it should not go too fast to zero in order for the method to keep converging.

A typical schedule that ensures both properties is to have asymptically $\tau_\ell \sim \ell^{-1}$ for $\ell \to +\infty$. We thus propose to use

$$\tau_\ell \overset{\text{def.}}{=} \frac{\tau_0}{1 + \ell/\ell_0}$$

where ℓ_0 indicates roughly the number of iterations serving as a "warmup" phase.

One can prove the following convergence result

$$\mathbb{E}(E(w_\ell)) - E(w^\star) = O\left(\frac{1}{\sqrt{\ell}}\right),$$

where $\mathbb{E}$ indicates an expectation with respect to the i.i.d. sampling performed at each iteration.

We can perform the Stochastic gradient descent and display the evolution of the energy $E(w_\ell)$. One can overlay on top (black dashed curve) the convergence of the batch gradient descent, with a carefull scaling of the number of iteration to account for the fact that the complexity of a batch iteration is n times larger.

16.1.3 Stochastic Gradient Descent with Averaging (SGA)

Stochastic gradient descent is slow because of the fast decay of τ_ℓ toward zero.

To improve somehow the convergence speed, it is possible to average the past iterate, i.e. run a "classical" SGD on auxiliary variables $(\tilde{w}_\ell)_\ell$

$$\tilde{w}_{\ell+1} = \tilde{w}_\ell - \tau_\ell \nabla E_{i(\ell)}(\tilde{w}_\ell)$$

and output as estimated weight vector the average

$$w_\ell \overset{\text{def.}}{=} \frac{1}{\ell} \sum_{k=1}^{\ell} \tilde{w}_\ell.$$

This defines the Stochastic Gradient Descent with Averaging (SGA) algorithm.

Note that it is possible to avoid explicitly storing all the iterates by simply updating a running average as follow

$$w_{\ell+1} = \frac{1}{\ell} \tilde{w}_\ell + \frac{\ell - 1}{\ell} w_\ell.$$

In this case, a typical choice of decay is rather of the form

$$\tau_\ell \overset{\text{def.}}{=} \frac{\tau_0}{1 + \sqrt{\ell/\ell_0}}.$$

Notice that the step size now goes much slower to 0, at rate $\ell^{-1/2}$.

Typically, because the averaging stabilizes the iterates, the choice of (ℓ_0, τ_0) is less important than for SGD.

Bach proves that for logistic classification, it leads to a faster convergence (the constant involved are smaller) than SGD, since on contrast to SGD, SGA is adaptive to the local strong convexity of E.

We can display the evolution of the energy $E(w_\ell)$.

16.1.4 Stochastic Averaged Gradient Descent (SAG)

For problem size n where the dataset (of size $n \times p$) can fully fit into memory, it is possible to further improve the SGA method by bookeeping the previous gradient. This gives rise to the Stochastic Averaged Gradient Descent (SAG) algorithm.

We stored all the previously computed gradient in $(G^i)_{i=1}^n$, which necessitate $n \times p$ memory. The iterates are defined by using a proxy g for the batch gradient, which is progressively enhanced during the iterates.

The algorithm reads

$$h \leftarrow \nabla E_{i(\ell)}(\tilde{w}_\ell),$$

$$g \leftarrow g - G^{i(\ell)} + h,$$

$$G^{i(\ell)} \leftarrow h,$$

$$w_{\ell+1} = w_\ell - \tau g.$$

Note that in contrast to SGD and SGA, this method uses a fixed step size τ. Similarely to the BGD, in order to ensure convergence, the step size τ should be of the order of $1/L$ where L is the Lipschitz constant of E.

This algorithm improves over SGA and BGD since it has a convergence rate of $O(1/\ell$. Furthermore, in the presence of strong convexity (for instance when X is injective for logistic classification), it has a linear convergence rate, i.e.

$$\mathbb{E}(E(w_\ell)) - E(w^\star) = O\left(\rho^\ell\right),$$

for some $0 < \rho < 1$.

Note that this improvement over SGD and SGA is made possible only because SAG explicitly use the fact that n is finite (while SGD and SGA can be extended to infinite n and more general minimization of expectations).

We display the evolution of the energy $E(w_\ell)$.

16.2 Automatic Differentiation

16.3 Deep Discriminative Models

16.4 Deep Generative Models

16.4.1 Density Fitting

Fitting and MLE

Generative Models

16.4.2 Auto-encoders

16.4.3 GANs

Bibliography

[1] P. Alliez and C. Gotsman. Recent advances in compression of 3d meshes. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in multiresolution for geometric modelling*, pages 3–26. Springer Verlag, 2005.

[2] P. Alliez, G. Ucelli, C. Gotsman, and M. Attene. Recent advances in remeshing of surfaces. In *AIM@SHAPE repport*. 2005.

[3] Amir Beck. *Introduction to Nonlinear Optimization: Theory, Algorithms, and Applications with MATLAB*. SIAM, 2014.

[4] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[5] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[6] E. Candès and D. Donoho. New tight frames of curvelets and optimal representations of objects with piecewise C^2 singularities. *Commun. on Pure and Appl. Math.*, 57(2):219–266, 2004.

[7] E. J. Candès. The restricted isometry property and its implications for compressed sensing. *Compte Rendus de l'Académie des Sciences*, Serie I(346):589–592, 2006.

[8] E. J. Candès, L. Demanet, D. L. Donoho, and L. Ying. Fast discrete curvelet transforms. *SIAM Multiscale Modeling and Simulation*, 5:861–899, 2005.

[9] A. Chambolle. An algorithm for total variation minimization and applications. *J. Math. Imaging Vis.*, 20:89–97, 2004.

[10] Antonin Chambolle, Vicent Caselles, Daniel Cremers, Matteo Novaga, and Thomas Pock. An introduction to total variation for image analysis. *Theoretical foundations and numerical methods for sparse recovery*, 9(263-340):227, 2010.

[11] Antonin Chambolle and Thomas Pock. An introduction to continuous optimization for imaging. *Acta Numerica*, 25:161–319, 2016.

[12] S.S. Chen, D.L. Donoho, and M.A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1999.

[13] F. R. K. Chung. Spectral graph theory. *Regional Conference Series in Mathematics, American Mathematical Society*, 92:1–212, 1997.

[14] Philippe G Ciarlet. Introduction à l'analyse numérique matricielle et à l'optimisation. 1982.

[15] P. L. Combettes and V. R. Wajs. Signal recovery by proximal forward-backward splitting. *SIAM Multiscale Modeling and Simulation*, 4(4), 2005.

[16] P. Schroeder et al. D. Zorin. Subdivision surfaces in character animation. In *Course notes at SIGGRAPH 2000*, July 2000.

[17] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Commun. on Pure and Appl. Math.*, 57:1413–1541, 2004.

[18] I. Daubechies and W. Sweldens. Factoring wavelet transforms into lifting steps. *J. Fourier Anal. Appl.*, 4(3):245–267, 1998.

[19] D. Donoho and I. Johnstone. Ideal spatial adaptation via wavelet shrinkage. *Biometrika*, 81:425–455, Dec 1994.

[20] Heinz Werner Engl, Martin Hanke, and Andreas Neubauer. *Regularization of inverse problems*, volume 375. Springer Science & Business Media, 1996.

[21] M. Figueiredo and R. Nowak. An EM Algorithm for Wavelet-Based Image Restoration. *IEEE Trans. Image Proc.*, 12(8):906–916, 2003.

[22] M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in multiresolution for geometric modelling*, pages 157–186. Springer Verlag, 2005.

[23] Simon Foucart and Holger Rauhut. *A mathematical introduction to compressive sensing*, volume 1. Birkhäuser Basel, 2013.

[24] I. Guskov, W. Sweldens, and P. Schröder. Multiresolution signal processing for meshes. In Alyn Rockwood, editor, *Proceedings of the Conference on Computer Graphics (Siggraph99)*, pages 325–334. ACM Press, August8–13 1999.

[25] A. Khodakovsky, P. Schröder, and W. Sweldens. Progressive geometry compression. In *Proceedings of the Computer Graphics Conference 2000 (SIGGRAPH-00)*, pages 271–278, New York, July 23–28 2000. ACMPress.

[26] L. Kobbelt. $\sqrt{3}$ subdivision. In Sheila Hoffmeyer, editor, *Proc. of SIGGRAPH'00*, pages 103–112, New York, July 23–28 2000. ACMPress.

[27] M. Lounsbery, T. D. DeRose, and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Trans. Graph.*, 16(1):34–73, 1997.

[28] S. Mallat. *A Wavelet Tour of Signal Processing, 3rd edition.* Academic Press, San Diego, 2009.

[29] Stephane Mallat. *A wavelet tour of signal processing: the sparse way.* Academic press, 2008.

[30] D. Mumford and J. Shah. Optimal approximation by piecewise smooth functions and associated variational problems. *Commun. on Pure and Appl. Math.*, 42:577–685, 1989.

[31] Y. Nesterov. Smooth minimization of non-smooth functions. *Math. Program.*, 103(1, Ser. A):127–152, 2005.

[32] Neal Parikh, Stephen Boyd, et al. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

[33] Gabriel Peyré. *L'algèbre discrète de la transformée de Fourier.* Ellipses, 2004.

[34] J. Portilla, V. Strela, M.J. Wainwright, and Simoncelli E.P. Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Trans. Image Proc.*, 12(11):1338–1351, November 2003.

[35] E. Praun and H. Hoppe. Spherical parametrization and remeshing. *ACM Transactions on Graphics*, 22(3):340–349, July 2003.

[36] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Phys. D*, 60(1-4):259–268, 1992.

[37] Otmar Scherzer, Markus Grasmair, Harald Grossauer, Markus Haltmeier, Frank Lenzen, and L Sirovich. *Variational methods in imaging*. Springer, 2009.

[38] P. Schröder and W. Sweldens. Spherical Wavelets: Efficiently Representing Functions on the Sphere. In *Proc. of SIGGRAPH 95*, pages 161–172, 1995.

[39] P. Schröder and W. Sweldens. Spherical wavelets: Texture processing. In P. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95*. Springer Verlag, Wien, New York, August 1995.

[40] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[41] A. Sheffer, E. Praun, and K. Rose. Mesh parameterization methods and their applications. *Found. Trends. Comput. Graph. Vis.*, 2(2):105–171, 2006.

[42] Jean-Luc Starck, Fionn Murtagh, and Jalal Fadili. *Sparse image and signal processing: Wavelets and related geometric multiscale analysis*. Cambridge university press, 2015.

[43] W. Sweldens. The lifting scheme: A custom-design construction of biorthogonal wavelets. *Applied and Computation Harmonic Analysis*, 3(2):186–200, 1996.

[44] W. Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM J. Math. Anal.*, 29(2):511–546, 1997.