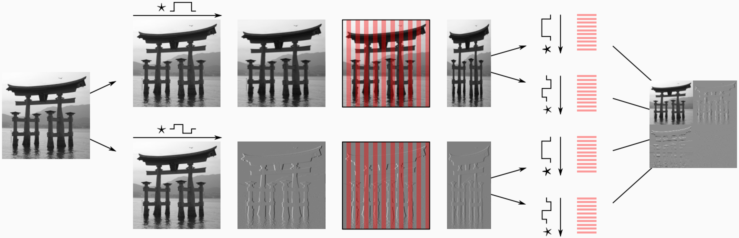
ECE 285

Image and video restoration

Chapter VI – Sparsity, shrinkage and wavelets

Charles Deledalle

June 7, 2019



Motivations

(a) $y = x + w$ (b) $z = \boldsymbol{F} y$ (c) $\frac{\lambda_i^2}{\lambda_i^2 + \sigma^2}$ (d) $\hat{z}_i = \frac{\lambda_i^2}{\lambda_i^2 + \sigma^2} z_i$ (e) $\hat{x} = \boldsymbol{F}^{-1} \hat{z}$

Wiener filter (LMMSE in the Fourier domain)

- Assume Fourier coefficients to be decorrelated (white),
- Modulate frequencies based on the mean power spectral density λ_i^2.

Limits

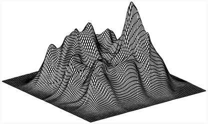
- Linear: no adaptation to the content $\Rightarrow \begin{cases} \text{Unable to preserve edges,} \\ \text{Blurry solutions.} \end{cases}$

Facts and consequences

- Assume Fourier coefficients to be decorrelated (white)
- Removing Gaussian noise $\Rightarrow$ need to be adaptive $\Rightarrow$ Non linear
- Assuming Gaussian noise + Gaussian prior $\Rightarrow$ Linear

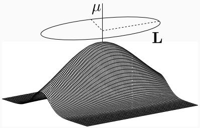
Deductive reasoning

Fourier coefficients of clean images are not Gaussian distributed



Underlying prior $x \mapsto p(x)$

$\not\approx$

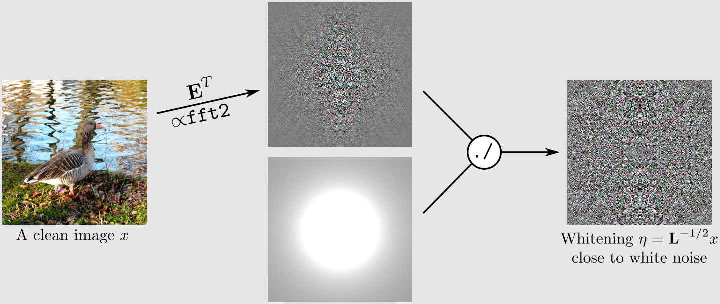


Gaussian prior $x \sim \mathcal{N}(\mu; \mathbf{L})$

How are Fourier coefficients distributed?

How are Fourier coefficients distributed?

1. Perform whitening with DFT



A clean image x

Standard deviation λ_i for each frequency

Whitening $\eta = \mathbf{L}^{-1/2}x$ close to white noise

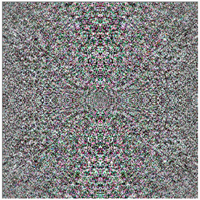
$$\mathrm{Var}[x] = \boldsymbol{L} = \boldsymbol{E}\boldsymbol{\Lambda}\boldsymbol{E}^* \quad \text{with} \quad \boldsymbol{E} = \tfrac{1}{\sqrt{n}}\boldsymbol{F}$$
$$\mathrm{diag}(\boldsymbol{\Lambda}) = (\lambda_1^2, \ldots, \lambda_n^2) = n^{-1}\mathsf{MPSD}$$

How are Fourier coefficients distributed?

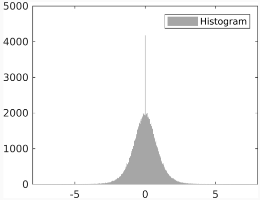
2. Look at the histogram

- The histogram of η has a symmetric bell shape around 0.
- It has a peak at 0 (a large number of Fourier coefficients are zero).
- It has large/heavy tails (many coefficients are "outliers"/abnormal).

(a) x



(b) Whitening η of x



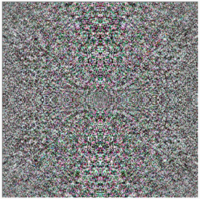
(c) Histogram of η

How are Fourier coefficients distributed?

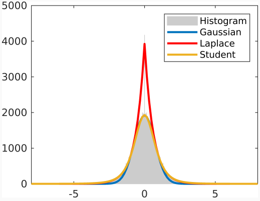
3. Look for the distribution that best fits (in log scale)

- Gaussian: bell shape ✓, peak ×, tail ×
- Laplacian: bell shape ×, peak ✓, tail ✓
- Student: bell shape ✓, peak ×, tail ✓ (heavier)

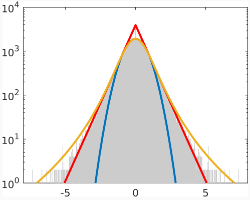
- Others: alpha stables and generalized Gaussian distributions



(a) Whitening η of x



(b) Histogram of η



(c) Log-histogram of η

Motivations – Distribution of Fourier coefficients

Model expression (zero mean, variance $= 1$)

- Gaussian: bell shape $\checkmark$, peak $\times$, tail $\times$

$$p(\eta_i) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\eta_i^2}{2}\right)$$

- Laplacian: bell shape $\times$, peak $\checkmark$, tail $\checkmark$

$$p(\eta_i) = \frac{1}{\sqrt{2}} \exp\left(-\sqrt{2}|\eta_i|\right)$$

- Student: bell shape $\checkmark$, peak $\times$, tail $\checkmark$ (heavier)

$$p(\eta_i) = \frac{1}{Z} \left(\frac{1}{(2r-2)+\eta_i^2}\right)^{r+1/2}$$

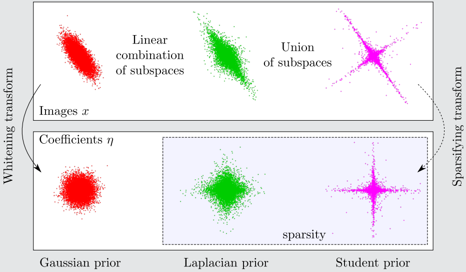
(Z normalization constant, $r > 1$ controls the tails)

How do they look in multiple-dimensions?

Motivations – Distribution of Fourier coefficients

- Gaussian prior
  - images are concentrated in an elliptical cluster,
  - outliers are rare (images outside the cluster).

- Peaky & heavy tailed priors: shape **between a diamond and a star**.



- **union of subspaces**: most images lie in one of the branches of the star,
- **sparsity**: most of their coefficients η_i are zeros,
- **robustness**: outlier coefficients are frequent.

Shrinkage functions

Consider the following Gaussian denoising problem

- Let $y \in \mathbb{R}^n$ and $x \in \mathbb{R}^p$ be two random vectors such that

$$y \,|\, x \sim \mathcal{N}(x, \sigma^2 \mathrm{Id}_n)$$

$$\mathbb{E}[x] = 0 \quad \text{and} \quad \mathrm{Var}[x] = \boldsymbol{L} = \boldsymbol{E}\boldsymbol{\Lambda}\boldsymbol{E}^*$$

- Let $\eta = \boldsymbol{\Lambda}^{-1/2}\boldsymbol{E}^* x$ (whitening / decorrelation of x)

Goal: estimate x from y
assuming a non-Gaussian prior p_η for η.
(such as Laplacian or Student)

Bayesian shrinkage functions

- Assume η_i are also **independent and identically distributed (iid)**.
- Then, the MMSE and MAP estimators both read as

$$\hat{x}^\star = \underbrace{\boldsymbol{E}\hat{z}}_{\text{Come back}} \quad \text{where} \quad \underbrace{\hat{z}_i = s(z_i;\ \lambda_i, \sigma)}_{\text{shrinkage}} \quad \text{and} \quad z = \underbrace{\boldsymbol{E}^* y}_{\text{Change of basis}}$$

- The function $z_i \mapsto s(z_i;\ \lambda_i, \sigma)$ is called **shrinkage function**.

- Unlike the LMMSE, s will depend on the prior distribution of η_i.
- As for the LMMSE, the solution can be computed in the eigenspace.
- We say that the estimator is **separable** in the eigenspace (ex: Fourier).

Remark

$$\boxed{\begin{array}{c} \text{independence} \Rightarrow \text{uncorrelation} \\ \neg\text{uncorrelation} \Rightarrow \neg\text{independence} \\ \text{correlation} \Rightarrow \text{dependence} \end{array}} \Rightarrow$$

Whitening is a necessarily step for independence but not a sufficient one.
(Except in the Gaussian case)

How are the shrinkage functions defined for the MMSE and MAP?

Shrinkage functions

- Recall that the MMSE is the posterior mean

$$\hat{x}^\star = \int_{\mathbb{R}^n} x p(x|y)\,\mathrm{d}x = \frac{\int_{\mathbb{R}^n} x p(y|x) p(x)\,\mathrm{d}x}{\int_{\mathbb{R}^n} p(y|x) p(x)\,\mathrm{d}x}$$

MMSE Shrinkage functions

- Under the previous assumptions

$$\hat{x}^\star = \underbrace{\boldsymbol{E}\hat{z}}_{\text{Come back}} \quad \text{where} \quad \underbrace{\hat{z}_i = s(z_i;\ \lambda_i, \sigma)}_{\text{shrinkage}} \quad \text{and} \quad z = \underbrace{\boldsymbol{E}^* y}_{\text{Change of basis}}$$

$$\text{with} \quad s(z;\ \lambda, \sigma) = \frac{\int_{\mathbb{R}} \tilde{z} \exp\left(-\frac{(z-\tilde{z})^2}{2\sigma^2}\right) p_\eta\left(\frac{\tilde{z}}{\lambda}\right)\,\mathrm{d}\tilde{z}}{\int_{\mathbb{R}} \exp\left(-\frac{(z-\tilde{z})^2}{2\sigma^2}\right) p_\eta\left(\frac{\tilde{z}}{\lambda}\right)\,\mathrm{d}\tilde{z}}$$

where p_η is the prior distribution on the entries of η.

- Separability: n **dimensional optimization** $\longrightarrow n \times$ **1d integrations.**

Shrinkage functions

- Recall that the MAP is the optimization problem

$$\hat{x}^\star \in \operatorname*{argmax}_{x \in \mathbb{R}^n} p(x|y) = \operatorname*{argmin}_{x \in \mathbb{R}^n} \left[-\log p(y|x) - \log p(x)\right]$$

MAP Shrinkage functions

- Under the previous assumptions

$$\hat{x}^\star = \underbrace{\boldsymbol{E}\hat{z}}_{\text{Come back}} \quad \text{where} \quad \underbrace{\hat{z}_i = s(z_i;\ \lambda_i, \sigma)}_{\text{shrinkage}} \quad \text{and} \quad z = \underbrace{\boldsymbol{E}^* y}_{\text{Change of basis}}$$

$$\text{with} \quad s(z;\ \lambda, \sigma) = \operatorname*{argmin}_{\tilde{z} \in \mathbb{R}} \left[\frac{(z - \tilde{z})^2}{2\sigma^2} - \log p_\eta\left(\frac{\tilde{z}}{\lambda}\right)\right]$$

where p_η is the prior distribution on the entries of η.

- Separability: n **dimensional integration** $\longrightarrow$ $n \times$ **1d optimisations.**

Shrinkage functions

Example (Gaussian noise + Gaussian prior)

- MMSE Shrinkage

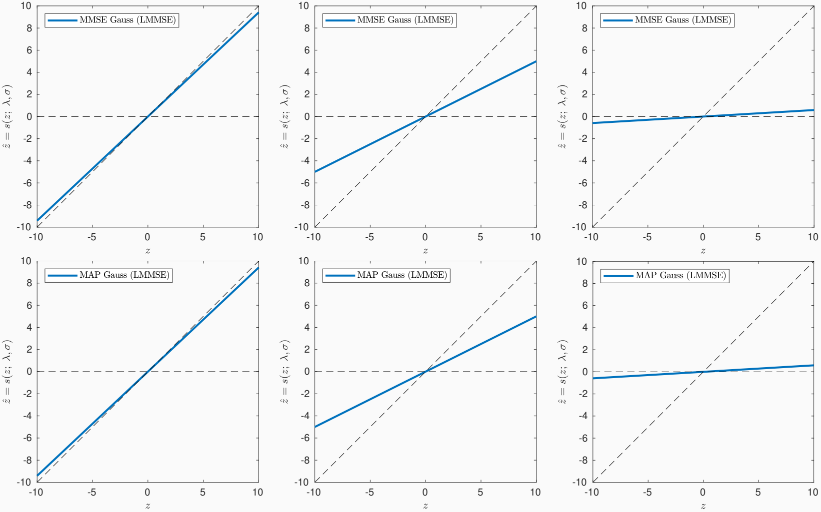
$$s(z;\ \lambda, \sigma) = \frac{\int_{\mathbb{R}} \tilde{z} \exp\left(-\frac{(z-\tilde{z})^2}{2\sigma^2} - \frac{\tilde{z}^2}{2\lambda^2}\right) \mathrm{d}\tilde{z}}{\int_{\mathbb{R}} \exp\left(-\frac{(z-\tilde{z})^2}{2\sigma^2} - \frac{\tilde{z}^2}{2\lambda^2}\right) \mathrm{d}\tilde{z}} = \frac{\lambda^2}{\lambda^2 + \sigma^2} z$$

- MAP Shrinkage

$$s(z;\ \lambda, \sigma) = \underset{\tilde{z} \in \mathbb{R}}{\operatorname{argmin}} \left[\frac{(z-\tilde{z})^2}{2\sigma^2} + \frac{\tilde{z}^2}{2\lambda^2}\right] = \frac{\lambda^2}{\lambda^2 + \sigma^2} z$$

- Gaussian prior: MAP = MMSE = Linear shrinkage.
- We retrieve the LMMSE as expected.

Gaussian noise + Gaussian prior



$\mathrm{SNR} = \lambda/\sigma = 4$ $\lambda/\sigma = 1$ $\lambda/\sigma = 1/4$

Posterior mean – Shrinkage functions – Examples

Example (Gaussian noise + Laplacian prior)

- MMSE Shrinkage

$$s(z;\ \lambda, \sigma) = \frac{\int \tilde{z} \exp\left(-\frac{(z-\tilde{z})^2}{2\sigma^2} - \frac{\sqrt{2}|\tilde{z}|}{\lambda}\right) \mathrm{d}\tilde{z}}{\int \exp\left(-\frac{(z-\tilde{z})^2}{2\sigma^2} - \frac{\sqrt{2}|\tilde{z}|}{\lambda}\right) \mathrm{d}\tilde{z}}$$

$$= z - \frac{\gamma\left(\operatorname{erf}\left(\frac{z-\gamma}{\sqrt{2}\sigma}\right) - \exp\left(\frac{2\gamma z}{\sigma^2}\right)\operatorname{erfc}\left(\frac{\gamma+z}{\sqrt{2}\sigma}\right) + 1\right)}{\operatorname{erf}\left(\frac{z-\gamma}{\sqrt{2}\sigma}\right) + \exp\left(\frac{2\gamma z}{\sigma^2}\right)\operatorname{erfc}\left(\frac{\gamma+z}{\sqrt{2}\sigma}\right) + 1}, \quad \gamma = \frac{\sqrt{2}\sigma^2}{\lambda}$$

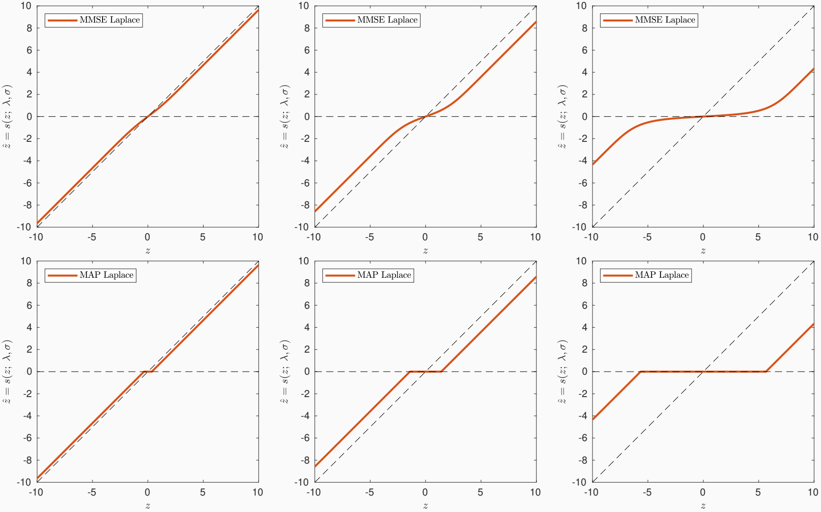
- MAP Shrinkage (soft-thresholding)

$$s(z;\ \lambda, \sigma) = \underset{\tilde{z}\in\mathbb{R}}{\operatorname{argmin}} \left[\frac{(z-\tilde{z})^2}{2\sigma^2} + \frac{\sqrt{2}|\tilde{z}|}{\lambda}\right] = \underbrace{\begin{cases} 0 & \text{if} \quad |z| < \gamma \\ z-\gamma & \text{if} \quad z > \gamma \\ z+\gamma & \text{if} \quad z < -\gamma \end{cases}}_{\text{Soft-T}(z,\gamma)}$$

Non-gaussian prior: MAP $\neq$ MMSE $\rightarrow$ **Non-linear shrinkage**.

Shrinkage functions

Gaussian noise + Laplacian prior



SNR $= \lambda/\sigma = 4$ $\lambda/\sigma = 1$ $\lambda/\sigma = 1/4$

Posterior mean – Shrinkage functions – Examples

Example (Gaussian noise + Student prior)

- MMSE Shrinkage

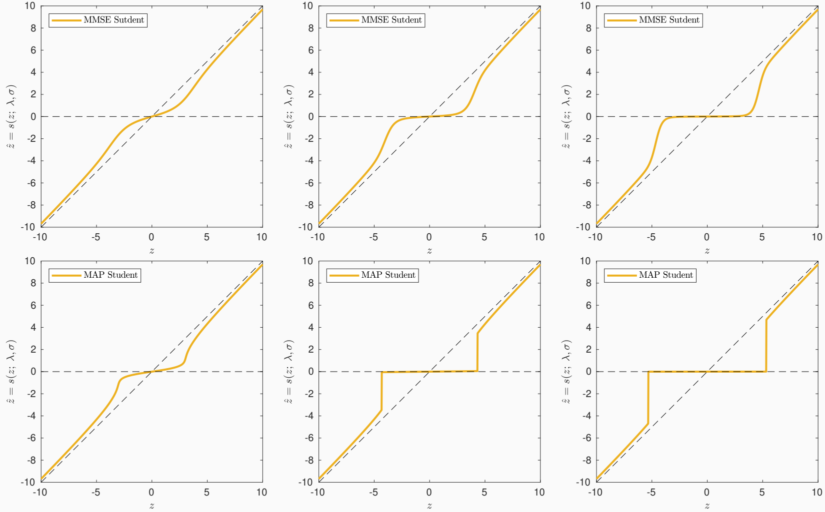
  No simple expression, requires 1d numerical integration

- MAP Shrinkage

  No simple expression, requires 1d numerical optimization

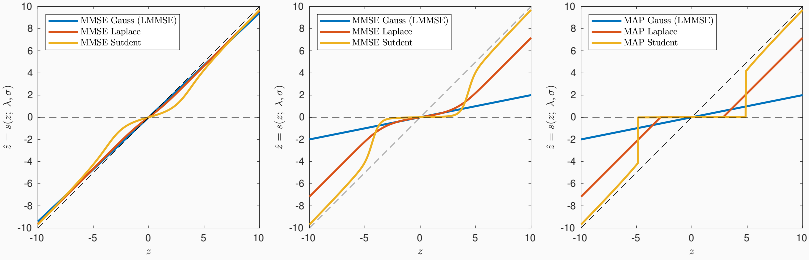
For efficiency, the 1d functions can be evaluated offline and stored in a look-up-table.

Gaussian noise + Student prior



SNR $= \lambda/\sigma = 4$ $\qquad$ $\lambda/\sigma = 1$ $\qquad$ $\lambda/\sigma = 1/4$

Posterior mean – Shrinkage functions – Examples

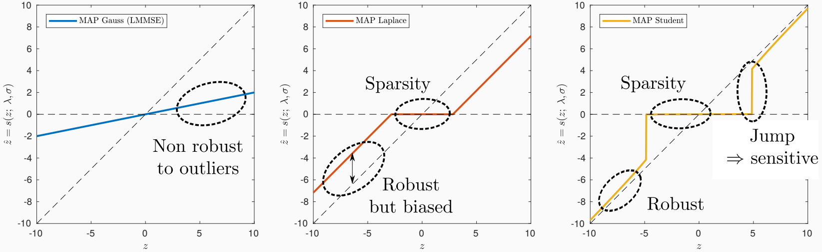


SNR $= \lambda/\sigma = 4$ | $\lambda/\sigma = 1/2$ | $\lambda/\sigma = 1/2$

- Coefficients are **shrunk** towards zero
- Signs are preserved
- Non-Gaussian priors leads to non-linear filtering:
  - **sparsity**: small coefficients are shrunk (likely due to noise)
  - **robustness**: large coefficients are preserved (likely encoding signal)
- Larger SNR $= \frac{\lambda}{\sigma} \Rightarrow$ shrinkage becomes close to identity.

Posterior mean – Shrinkage functions – Examples

Interpretation



Sparsity: zero for small values.

Robustness: remain close to the identity for large values.

Transition: bias/variance tradeoff.

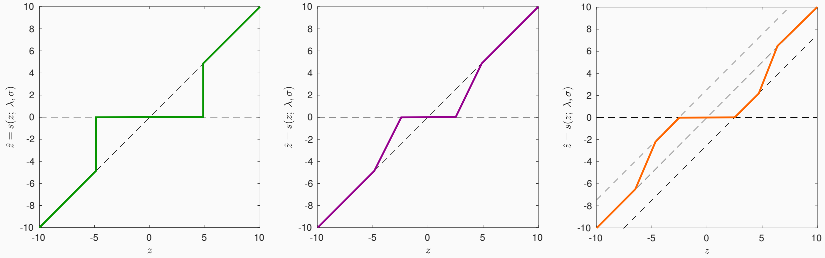
Can we design our own shrinkage according to what we want?

Shrinkage functions

Shrinkage functions (*a.k.a*, thresholding functions)

- Pick a shrinkage function s satisfying
  - **Shrink:** $|s(z)| \leqslant |z|$ (non-expansive)
  - **Preserve sign:** $z \cdot s(z) \geqslant 0$
  - **Kill low SNR:** $\lim\limits_{\frac{\lambda}{\sigma} \to 0} s(z;\ \lambda, \sigma) = 0$
  - **Keep high SNR:** $\lim\limits_{\frac{\lambda}{\sigma} \to \infty} s(z;\ \lambda, \sigma) = z$
  - **Increasing:** $z_1 \leqslant z_2 \quad \Leftrightarrow \quad s(z_1) \leqslant s(z_2)$
- **Beyond Bayesian:** No need to relate s to a prior distribution p_η.

A few examples (among many others)



Hard-thresholding MCP SCAD

- Though not necessarily related to a prior distribution,
- Often related to a **penalized least square problem**, ex:

$$\text{Hard-T}(z) = \underset{\tilde{z} \in \mathbb{R}}{\operatorname{argmin}} \left[(z - \tilde{z})^2 + \tau^2 \mathbf{1}_{\{\tilde{z} \neq 0\}} \right] = \begin{cases} 0 & \text{if} \quad |z| < \tau \\ z & \text{otherwise} \end{cases}$$

- Hard-thresholding: similar behavior to Student's shrinkage.

Link with penalized least square (1/2)

- $\boldsymbol{D} = \boldsymbol{L}^{1/2} = \boldsymbol{E}\boldsymbol{\Lambda}^{1/2}$ is an **orthogonal dictionary** of n atoms/words

$$\boldsymbol{D} = (d_1, d_2, \ldots, d_n) \quad \text{with} \quad \|d_i\| = \lambda_i \quad \text{and} \quad \langle d_i, d_j \rangle = 0 \text{ (for } i \neq j)$$

- Goal: Look for the n coefficients η_i, such that $\hat{x}$ close to y

$$\hat{x} = \boldsymbol{D}\eta = \sum_{i=1}^{n} \eta_i d_i = \text{"linear comb. of the } \textbf{orthogonal atoms } d_i \text{ of } D\text{"}$$

- Choosing $\eta_i = \left\langle \frac{d_i}{\|d_i\|^2}, y \right\rangle$, *i.e.*, $\eta = \boldsymbol{\Lambda}^{-1/2}\boldsymbol{E}^* y$, is optimal:

$$\hat{x} = y$$

  but, it also reconstructs the noise component.

- Idea: **penalize the coeffs to prevent from reconstructing the noise**.

Shrinkage functions

Link with penalized least square (2/2)

- Penalization on the coefficients controls shrinkage and sparsity:
  - $\frac{1}{2}\|y - \boldsymbol{D}\eta\|_2^2 + \frac{\tau^2}{2}\|\eta\|_2^2 \quad \Rightarrow \quad \hat{z}_i = \frac{\lambda_i^2}{\lambda_i^2 + \tau^2} z_i$
  - $\frac{1}{2}\|y - \boldsymbol{D}\eta\|_2^2 + \tau\|\eta\|_1 \quad \Rightarrow \quad \hat{z}_i = \text{Soft-T}\left(z_i, \gamma_i\right) \quad \text{with} \quad \gamma_i = \frac{\tau}{\lambda_i}$
  - $\frac{1}{2}\|y - \boldsymbol{D}\eta\|_2^2 + \frac{\tau^2}{2}\|\eta\|_0 \quad \Rightarrow \quad \hat{z}_i = \text{Hard-T}\left(z_i, \gamma_i\right) \quad \text{with} \quad \gamma_i = \frac{\tau}{\lambda_i}$

ℓ_0 **pseudo-norm:** $\|\eta\|_0 = \lim_{p \to 0} \left(\sum_{i=1}^{n} \eta_i^p \right)^{1/p}$ = "**# of non-zero coefficients**"

Sparsity: $\|\eta\|_0$ small compared to n

Posterior mean – Shrinkage in the Fourier domain

Shrinkage in the discrete Fourier domain

(a) x

(b) $y = x + w$

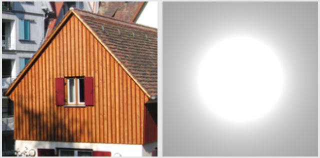
```
  sig    = 20
▶ y      = x + sig * np.random.randn(x.shape)
  n1, n2 = y.shape[:2]
  n      = n1 * n2
  lbd    = np.sqrt(prior_mpsd(n1, n2) / n)

  z      = nf.fft2(y, axes=(0, 1)) / np.sqrt(n)
  zhat   = shrink(z, lbd, sig)
  xhat   = np.real(nf.ifft2(zhat, axes=(0, 1))) * np.sqrt(n)
```

$$z = \boldsymbol{F} y / \sqrt{n}$$

$$\hat{z}_i = s(z_i;\, \lambda_i, \sigma)$$

$$\hat{x} = \sqrt{n} \boldsymbol{F}^{-1} \hat{z}$$

Shrinkage in the discrete Fourier domain



(a) x (b) λ

```
  sig     = 20
  y       = x + sig * np.random.randn(x.shape)
  n1, n2  = y.shape[:2]
  n       = n1 * n2
▶ lbd     = np.sqrt(prior_mpsd(n1, n2) / n)

  z       = nf.fft2(y, axes=(0, 1)) / np.sqrt(n)
  zhat    = shrink(z, lbd, sig)
  xhat    = np.real(nf.ifft2(zhat, axes=(0, 1))) * np.sqrt(n)
```

$$z = \boldsymbol{F} y / \sqrt{n}$$

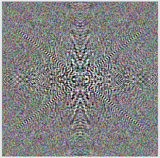
$$\hat{z}_i = s(z_i;\, \lambda_i, \sigma)$$

$$\hat{x} = \sqrt{n} \boldsymbol{F}^{-1} \hat{z}$$

Posterior mean – Shrinkage in the Fourier domain

Shrinkage in the discrete Fourier domain

(a) x



(b) z

```
sig    = 20
y      = x + sig * np.random.randn(x.shape)
n1, n2 = y.shape[:2]
n      = n1 * n2
lbd    = np.sqrt(prior_mpsd(n1, n2) / n)

z      = nf.fft2(y, axes=(0, 1)) / np.sqrt(n)
zhat   = shrink(z, lbd, sig)
xhat   = np.real(nf.ifft2(zhat, axes=(0, 1))) * np.sqrt(n)
```

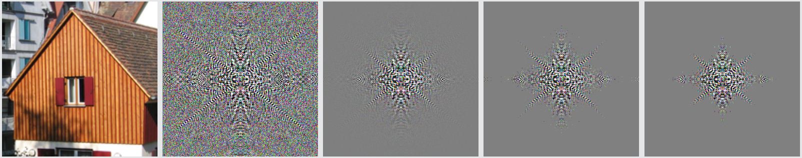
$$z = \boldsymbol{F} y / \sqrt{n}$$

$$\hat{z}_i = s(z_i;\, \lambda_i, \sigma)$$

$$\hat{x} = \sqrt{n} \boldsymbol{F}^{-1} \hat{z}$$

Posterior mean – Shrinkage in the Fourier domain

Shrinkage in the discrete Fourier domain



(a) x (b) z (c) Linear/Wiener (=Gaussian) (d) Soft-T (=Laplacian) (e) Hard-T (≈Student)

```
  sig     = 20
  y       = x + sig * np.random.randn(x.shape)
  n1, n2 = y.shape[:2]
  n       = n1 * n2
  lbd     = np.sqrt(prior_mpsd(n1, n2) / n)

  z       = nf.fft2(y, axes=(0, 1)) / np.sqrt(n)
▶ zhat    = shrink(z, lbd, sig)
  xhat    = np.real(nf.ifft2(zhat, axes=(0, 1))) * np.sqrt(n)
```

$$z = \boldsymbol{F} y / \sqrt{n}$$

$$\hat{z}_i = s(z_i;\, \lambda_i, \sigma)$$

$$\hat{x} = \sqrt{n} \boldsymbol{F}^{-1} \hat{z}$$

Posterior mean – Shrinkage in the Fourier domain

Shrinkage in the discrete Fourier domain

(a) x (b) $y = x + w$ (c) Linear/Wiener (=Gaussian) (d) Soft-T (=Laplacian) (e) Hard-T (≈Student)

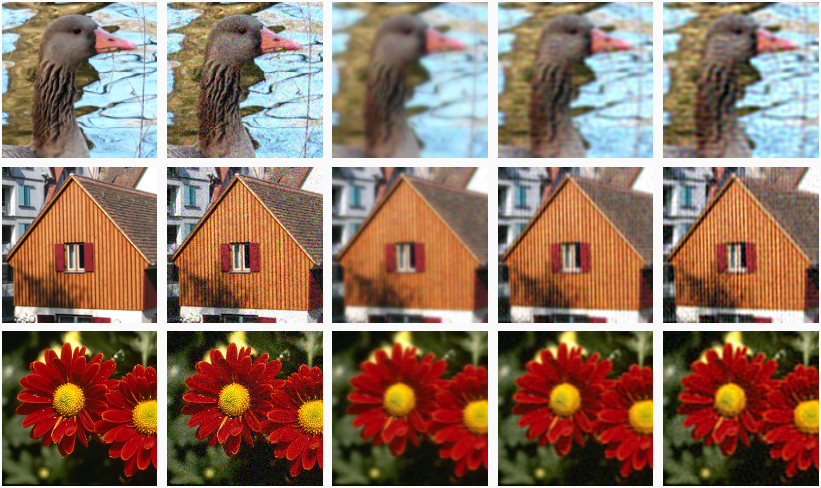
```
sig     = 20
y       = x + sig * np.random.randn(x.shape)
n1, n2 = y.shape[:2]
n       = n1 * n2
lbd     = np.sqrt(prior_mpsd(n1, n2) / n)

z       = nf.fft2(y, axes=(0, 1)) / np.sqrt(n)
zhat    = shrink(z, lbd, sig)
xhat    = np.real(nf.ifft2(zhat, axes=(0, 1))) * np.sqrt(n)
```

$$z = \boldsymbol{F} y / \sqrt{n}$$

$$\hat{z}_i = s(z_i;\, \lambda_i, \sigma)$$

$$\hat{x} = \sqrt{n} \boldsymbol{F}^{-1} \hat{z}$$

Shrinkage functions – Fourier domain – Results



(a) x (b) y $(\sigma = 20)$ (c) Linear/Wiener (=Gaussian) (d) Soft-T (=Laplacian) (e) Hard-T (≈Student)

Bias $\longleftrightarrow$ **Variance**

Shrinkage functions – Fourier domain – Results

(a) x (b) y ($\sigma = 40$) (c) Linear/Wiener (=Gaussian) (d) Soft-T (=Laplacian) (e) Hard-T (≈Student)

Bias ⟷ **Variance**

Shrinkage functions – Fourier domain – Results

(a) x (b) y $(\sigma = 60)$ (c) Linear/Wiener (=Gaussian) (d) Soft-T (=Laplacian) (e) Hard-T (≈Student)

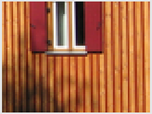
Bias ⟷ **Variance**

Shrinkage functions – Fourier domain – Results

(a) x (b) y ($\sigma = 120$) (c) Linear/Wiener (=Gaussian) (d) Soft-T (=Laplacian) (e) Hard-T (≈Student)

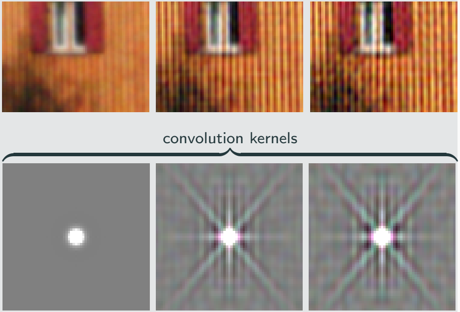
Bias $\longleftrightarrow$ **Variance**

Limits of shrinkage in the discrete Fourier domain



(a) x

(b) y



(c) Linear (=Gaussian) (d) Soft-T (=Laplacian) (e) Hard-T (≈Student)

- Linear shrinkage (Wiener)
  $\Rightarrow$ Non-adaptive,
- Non-linear shrinkage
  $\Rightarrow$ **Adaptive convolution**,
- Adapts to the frequency content,
- but not to the spatial content.

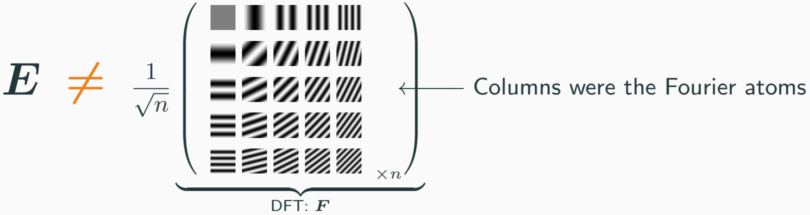
$$\hat{z}_i = s(z_i; \tau, \sigma) = \underbrace{\frac{s(z_i; \tau, \sigma)}{z_i} \times z_i}_{\text{element-wise product}} \quad \Leftrightarrow \quad \underbrace{\hat{x} = \nu(y) * y}_{\substack{\text{spatial average} \\ \text{adapted to the spectrum of } y.}}$$

Consequences

- Modulating Fourier coefficients $\Rightarrow$ Non spatially adaptive
- Assuming Fourier coefficients to be white+sparse $\Rightarrow$ Shrinkage in Fourier

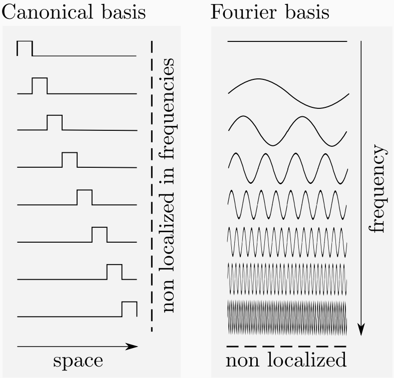
Deductive reasoning

Need another representation for sparsifying clean images



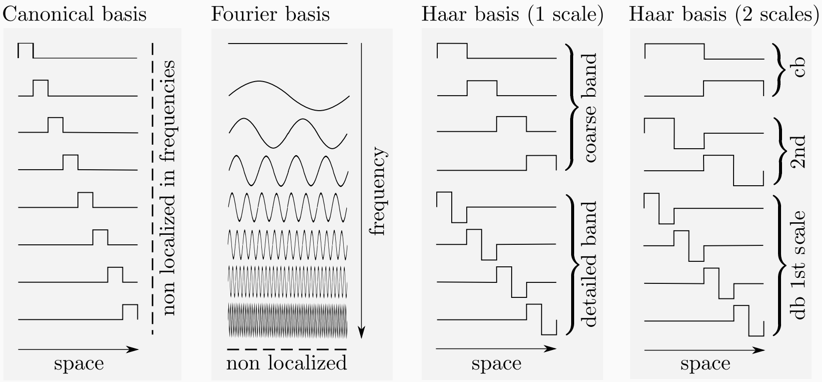
What transform can make signal white and sparse and captures both spatial and spectral contents?

Wavelet transforms



$$\mathrm{Id} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad \boldsymbol{F} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & e^{-2\pi i 1/4} & e^{-2\pi i 2/4} & e^{-2\pi i 3/4} \\ 1 & e^{-2\pi i 2/4} & e^{-2\pi i 4/4} & e^{-2\pi i 6/4} \\ 1 & e^{-2\pi i 3/4} & e^{-2\pi i 6/4} & e^{-2\pi i 9/4} \end{pmatrix}$$

Introduction to wavelets – Haar (1d case) [Alfréd Haar (1909)]



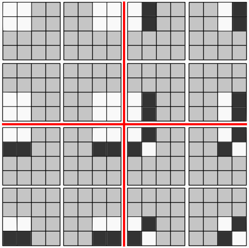
$$\mathcal{H}^{\text{1st}} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix} \quad \text{and} \quad \mathcal{H}^{\text{2nd}} = \begin{array}{r} 1/2 \\ 1/2 \\ 1/\sqrt{2} \\ 1/\sqrt{2} \end{array} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix}$$

(a) $\mathcal{H}^{1\text{st}}$ (4×4 image) (b) x (c) $\mathcal{H}^{1\text{st}} x$

2d Haar representation

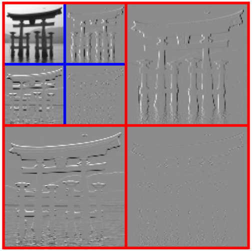
4 sub-bands
- Coarse sub-band
- Vertical detailed sub-band
- Horizontal detailed sub-band
- Diagonal detailed sub-band

Introduction to wavelets – Haar (2d case)



(a) $\mathcal{H}^{\text{1st}}$ (4×4 image)

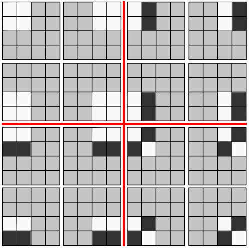
(b) x



(c) $\mathcal{H}^{\text{2nd}} x$

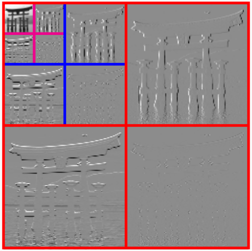
Multi-scale 2d Haar representation

- Repeat recursively J times
- Dyadic decomposition
- Multi-scale representation
- Related to scale spaces



(a) $\mathcal{H}^{\text{1st}}$ (4×4 image)

(b) x

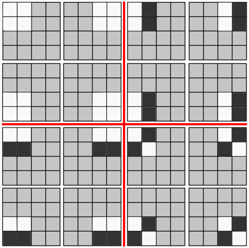


(c) $\mathcal{H}^{\text{3rd}} x$

Multi-scale 2d Haar representation

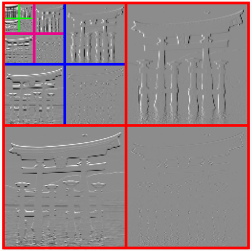
- Repeat recursively J times
- Dyadic decomposition
- Multi-scale representation
- Related to scale spaces

Introduction to wavelets – Haar (2d case)



(a) $\mathcal{H}^{1\text{st}}$ (4×4 image)

(b) x

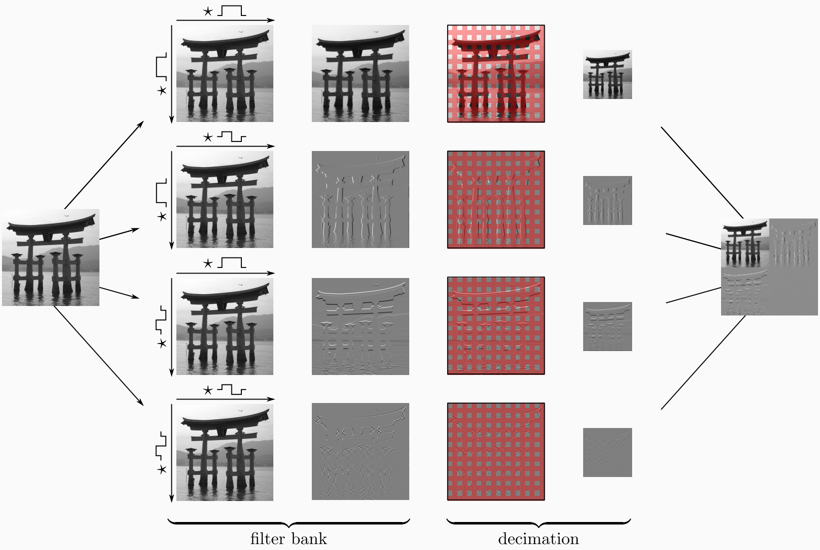


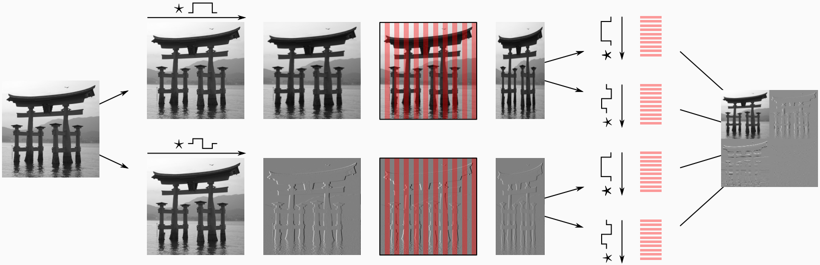
(c) $\mathcal{H}^{4\text{th}} x$

Multi-scale 2d Haar representation

- Repeat recursively J times
- Dyadic decomposition
- Multi-scale representation
- Related to scale spaces

Introduction to wavelets – Haar transform - Filter bank





Properties of the 2d Haar transform

- Separable: 1d Haar transforms in horizontal and next vertical direction
- First: perform a **low pass** and **high pass** filtering
- Next: perform decimation by a factor of 2

Can we choose other low and high pass filters to get a better transform?

Discrete wavelets

Discrete wavelet transform (DWT) (1/3) (1d and n even)

- Let $h \in \mathbb{R}^n$ (with periodical boundary conditions) satisfying

$$\sum_{i=0}^{n-1} h_i = 0$$

$$\sum_{i=0}^{n-1} h_i^2 = 1$$

$$\text{and} \quad \sum_{i=0}^{n-1} h_i h_{i+2k} = 0 \quad \text{for all integer } k \neq 0$$

Example (Haar as a particular case)

$$h = \frac{1}{\sqrt{2}}(0 \ldots 0 \quad -1 \quad +1 \quad 0 \ldots 0)$$

Discrete wavelet transform (DWT) (2/3) **(1d and n even)**

- Define the high and low pass filters $H : \mathbb{R}^n \to \mathbb{R}^n$ and $G : \mathbb{R}^n \to \mathbb{R}^n$ as

$$(Hx)_k = (h * x)_k = \sum_{i=0}^{n-1} h_i x_{k-i}$$

$$(Gx)_k = (g * x)_k = \sum_{i=0}^{n-1} g_i x_{k-i} \quad \text{where } g_i = (-1)^i h_{n-1-i}$$

- Note: necessarily $\sum_{i=0}^{n-1} g_i = \sqrt{2}$

Example (Haar as a particular case)

$$h = \frac{1}{\sqrt{2}}(0 \;\dots\; 0 \quad -1 \quad +1 \quad 0 \;\dots\; 0)$$

$$g = \frac{1}{\sqrt{2}}(0 \dots 0 \quad +1 \quad +1 \quad 0 \dots 0)$$

- Define the decimation by 2 of a matrix $\boldsymbol{M} \in \mathbb{R}^{n \times n}$ as

$$\boldsymbol{M}{\downarrow_2} = \text{“}\texttt{M[::2, :]}\text{”} \in \mathbb{R}^{n/2 \times n}$$

i.e., the matrix obtained by removing every two rows.

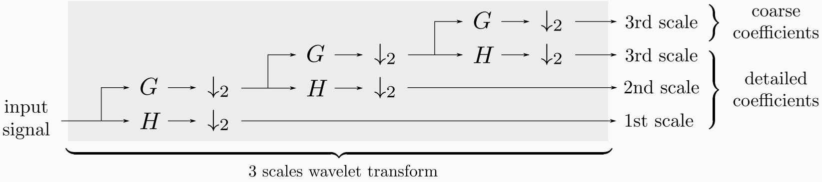
- $\boldsymbol{M}{\downarrow_2}\, x$: apply $\boldsymbol{M}$ to x and next remove every two entries.

Discrete wavelet transform (DWT) (3/3) **(1d and n even)**

$$\text{Let } \boldsymbol{W} = \begin{pmatrix} G{\downarrow_2} \\ H{\downarrow_2} \end{pmatrix} \in \mathbb{R}^{n \times n}$$

Then
- $x \mapsto \boldsymbol{W}x$: **orthonormal discrete wavelet transform**,
- Columns of $\boldsymbol{W}$: **orthonormal discrete wavelet basis**,
- $z = \boldsymbol{W}x$: **wavelet coefficients of x**.

Multi-scale discrete wavelets



Multi-scale DWT (1d and n multiple of 2^J) **[Mallat, 1989]**

Defined recursively as $\boldsymbol{W}^{J\text{-th}} = \begin{pmatrix} \boldsymbol{W}^{(J\text{-}1)\text{-th}} & O \\ 0 & \mathrm{Id} \end{pmatrix} \boldsymbol{W}$

Multi-scale discrete wavelets

Implementation of 2D DWT (n_1 and n_2 multiple of 2^J)

```
def dwt(x, J, h, g):                          # 2d and multi-scale
    if J == 0:
        return x
    n1, n2 = x.shape[:2]
    m1, m2 = (int(n1 / 2), int(n2 / 2))
    z = dwt1d(x, h, g)
    z = flip(dwt1d(flip(z), h, g))
    z[:m1, :m2] = dwt(z[:m1, :m2], J - 1, h, g)
    return z
```

```
def dwt1d(x, h, g):                           # 1d and 1scale
    coarse = convolve(x, g)
    detail = convolve(x, h)
    z = np.concatenate((coarse[::2, :], detail[::2, :]), axis=0)
    return z
```

Use separability

Multi-scale discrete wavelets

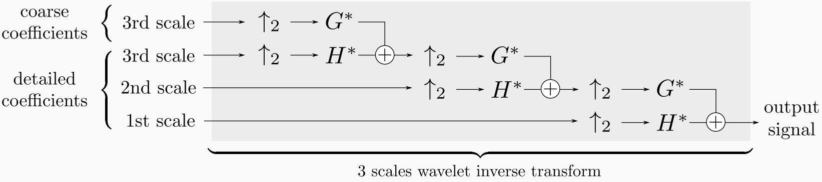
Multi-scale Inverse DWT (1d and n multiple of 2^J)

$$\text{Defined recursively as } (\boldsymbol{W}^{J\text{-th}})^{-1} = \boldsymbol{W}^{-1}\begin{pmatrix} (\boldsymbol{W}^{(J\text{-}1)\text{-th}})^{-1} & O \\ 0 & \mathrm{Id} \end{pmatrix}$$

$$\text{where } \boldsymbol{W}^{-1} = \boldsymbol{W}^{*} = \begin{pmatrix} G^*\!\uparrow_2 & H^*\!\uparrow_2 \end{pmatrix} \in \mathbb{R}^{n \times n}$$

and $\boldsymbol{M}\!\uparrow_2$: remove every two columns.

$\boldsymbol{M}\!\uparrow_2 x$: insert 0 every two entries in x and next apply $\boldsymbol{M}$.



Implementation of 2D IDWT (n_1 and n_2 multiple of 2^J)

```python
def idwt(z, J, h, g):                    # 2d and multi-scale
    if J == 0:
        return z
    n1, n2 = z.shape[:2]
    m1, m2 = (int(n1 / 2), int(n2 / 2))
    x = z.copy()
    x[:m1, :m2] = idwt(x[:m1, :m2], J - 1, h, g)
    x = flip(idwt1d(flip(x), h, g))
    x = idwt1d(x, h, g)
    return x
```

```python
def idwt1d(z, h, g):                     # 1d and 1scale
    n1 = z.shape[0]
    m1 = int(n1 / 2)
    coarse, detail = np.zeros(z.shape), np.zeros(z.shape)
    coarse[::2, :], detail[::2, :] = z[:m1, :], z[m1:, :]
    x = convolve(coarse, g[::-1]) + convolve(detail, h[::-1])
    return x
```

Discrete wavelet with limited support

- Consider a high pass filter with **finite support** of size $m = 2p$ (even). For instance for $m = 4$

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

- Then h defines a wavelet transform if it satisfies the three conditions

$$\sum h_i = 0 \quad \text{and} \quad \sum h_i^2 = 1 \quad \text{and} \quad \sum h_i h_{i+2k} = 0 \quad \text{for } k = 1 \text{ to } p-1$$

- This system has $2p$ unknowns and $1 + p$ independent equations.
- If $p = 1$, $2p = 1 + p$, this implies that the solution is unique (Haar).
- Otherwise, one has **$p - 1$ degrees of freedom**.

Daubechies' wavelets (1988)

- Daubechies suggests adding the $p-1$ constraints

$$\sum_{i=0}^{2p-1} i^q h_i = 0 \quad \text{for } q = 1 \text{ to } p-1 \qquad \text{(vanishing } q\text{-order moments)}$$

- For $p = 2$, the (orthonormal) Daubechies' wavelets are defined as

$$\begin{cases} h_0^2 + h_1^2 + h_2^2 + h_3^2 & = 1 \\ h_0 + h_1 + h_2 + h_3 & = 0 \\ h_0 h_2 + h_1 h_3 & = 0 \\ h_1 + 2h_2 + 3h_3 & = 0 \end{cases} \Leftrightarrow h = \pm \frac{1}{\sqrt{2}} \begin{pmatrix} \frac{1+\sqrt{3}}{4} \\ \frac{3+\sqrt{3}}{4} \\ \frac{3-\sqrt{3}}{4} \\ \frac{1-\sqrt{3}}{4} \end{pmatrix}$$

- The corresponding DWT is referred to as Daubechies-2 (or Db2).

As for the Fourier transform, there also exists a continuous version.

Continuous wavelets

Continuous wavelet transform (CWT) (1d)

- Continuum of locations $t \in \mathbb{R}$ and scales $a > 0$,
- **Continuous wavelet transform** of $x : \mathbb{R} \to \mathbb{R}$

$$\underbrace{c(a,t)}_{\text{wavelet coefficient}} = \int_{-\infty}^{+\infty} \psi_{a,t}^*(t')\, x(t')\, \mathrm{d}t' = \langle \underbrace{x}_{\text{signal}}, \underbrace{\psi_{a,t}}_{\text{wavelet}} \rangle$$

where * is the complex conjugate.

- $\psi_{a,t}$: **daughter wavelets**, translated and scaled versions of Ψ

$$\psi_{a,t}(t') = \frac{1}{\sqrt{a}} \Psi\left(\frac{t'-t}{a}\right)$$

- Ψ: **the mother wavelets** satisfying

$$\int_{-\infty}^{+\infty} \Psi(t)\, \mathrm{d}t = 0 \quad \text{and} \quad \int_{-\infty}^{+\infty} |\Psi(t)|^2\, \mathrm{d}t = 1 < \infty$$

(zero-mean) (unit-norm / square-integrable)

Continuous wavelets

Inverse CWT (1d)

- The **inverse continuous wavelet transform** is given by

$$x(t) = \frac{1}{C_\Psi} \int_{-\infty}^{+\infty} \int_{0}^{+\infty} \frac{1}{|a|^2} c(a, t') \psi_{a,t}\left(t'\right) \, \mathrm{d}a \, \mathrm{d}t'$$

with $C_\Psi = \displaystyle\int_0^{+\infty} \frac{|\hat{\Psi}(u)|^2}{u} \mathrm{d}u$ where $\hat{\Psi}$ is the Fourier transform of Ψ.

Relation between CWT/DWT (1d)

- The DWT can be seen as the discretization of the CWT
  - **Diadic discretization** in scale: $a = 1, 2, 4, \ldots, 2^J$
  - Uniform discretization in time at scale j with step 2^j: $t = 1{:}2^j{:}n$

Twin-scale relation **(1d)**

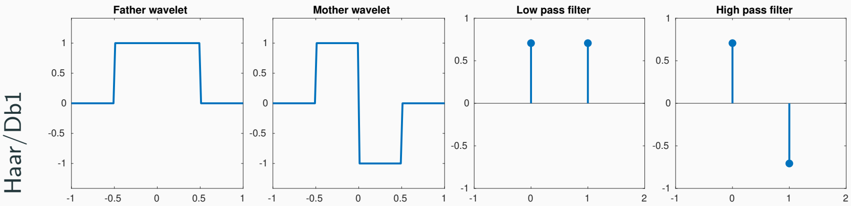
- The CWT is orthogonal (inverse = adjoint), if and only if Ψ satisfies

$$\Psi(t) = \sqrt{2}\sum_{i=0}^{m-1} h_i \Phi(2t-i) \quad \text{and} \quad \Phi(t) = \sqrt{2}\sum_{i=0}^{m-1} g_i \Phi(2t-i)$$

  where h and g are high- and low-pass filters defining a DWT.
- Φ is called **father wavelet or scaling function**.
- Note: potentially $m = \infty$.

Twin-scale relation: allows to define a CWT from DWT and vice-versa.
The CWT may not have a closed form (approximated by the cascade algorithm)

Continuous and discrete wavelets



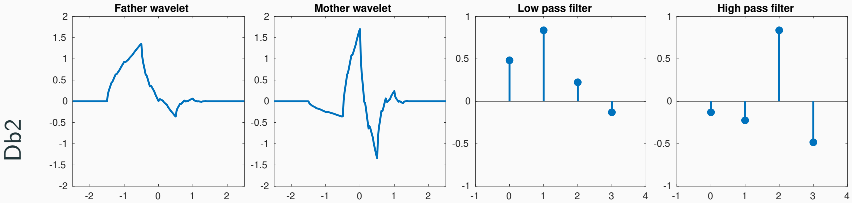
Popular wavelets are:

Haar (1909); Gabor wavelet (1946); Mexican hat/Marr wavelet (1980); Morlet wavelet (1984); Daubechies (1988); Meyer wavelet (1990); Binomial quadrature mirror filter (1990); Coiflets (1991); Symlets (1992).

Some classical wavelet transforms are not orthogonal.

Bi-orthogonal: Non orthogonal but invertible (ex: for symmetric wavelets). Two filters for the direct, and two others for the inverse.

Continuous and discrete wavelets



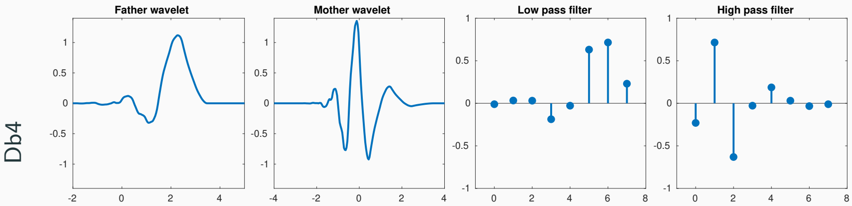
Popular wavelets are:

Haar (1909); Gabor wavelet (1946); Mexican hat/Marr wavelet (1980); Morlet wavelet (1984); Daubechies (1988); Meyer wavelet (1990); Binomial quadrature mirror filter (1990); Coiflets (1991); Symlets (1992).

Some classical wavelet transforms are not orthogonal.

Bi-orthogonal: Non orthogonal but invertible (ex: for symmetric wavelets).
Two filters for the direct, and two others for the inverse.

Continuous and discrete wavelets



Popular wavelets are:

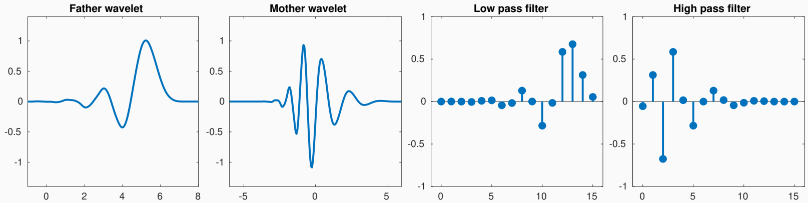
Haar (1909); Gabor wavelet (1946); Mexican hat/Marr wavelet (1980); Morlet wavelet (1984); Daubechies (1988); Meyer wavelet (1990); Binomial quadrature mirror filter (1990); Coiflets (1991); Symlets (1992).

Some classical wavelet transforms are not orthogonal.

Bi-orthogonal: Non orthogonal but invertible (ex: for symmetric wavelets). Two filters for the direct, and two others for the inverse.

Continuous and discrete wavelets

Db8



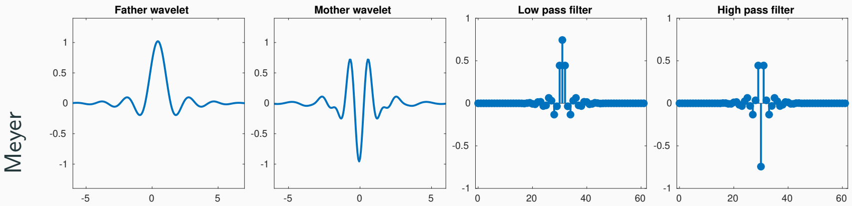
Popular wavelets are:

Haar (1909); Gabor wavelet (1946); Mexican hat/Marr wavelet (1980); Morlet wavelet (1984); Daubechies (1988); Meyer wavelet (1990); Binomial quadrature mirror filter (1990); Coiflets (1991); Symlets (1992).

Some classical wavelet transforms are not orthogonal.

Bi-orthogonal: Non orthogonal but invertible (ex: for symmetric wavelets). Two filters for the direct, and two others for the inverse.

Continuous and discrete wavelets



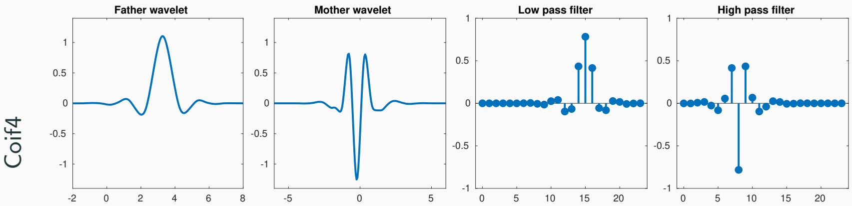
Popular wavelets are:

Haar (1909); Gabor wavelet (1946); Mexican hat/Marr wavelet (1980); Morlet wavelet (1984); Daubechies (1988); Meyer wavelet (1990); Binomial quadrature mirror filter (1990); Coiflets (1991); Symlets (1992).

Some classical wavelet transforms are not orthogonal.

Bi-orthogonal: Non orthogonal but invertible (ex: for symmetric wavelets). Two filters for the direct, and two others for the inverse.

Continuous and discrete wavelets



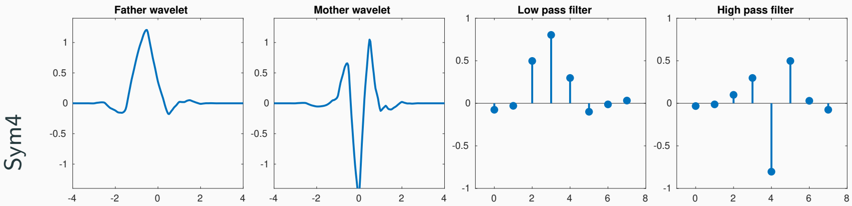
Popular wavelets are:

Haar (1909); Gabor wavelet (1946); Mexican hat/Marr wavelet (1980); Morlet wavelet (1984); Daubechies (1988); Meyer wavelet (1990); Binomial quadrature mirror filter (1990); Coiflets (1991); Symlets (1992).

Some classical wavelet transforms are not orthogonal.

Bi-orthogonal: Non orthogonal but invertible (ex: for symmetric wavelets). Two filters for the direct, and two others for the inverse.

Continuous and discrete wavelets



Popular wavelets are:

Haar (1909); Gabor wavelet (1946); Mexican hat/Marr wavelet (1980); Morlet wavelet (1984); Daubechies (1988); Meyer wavelet (1990); Binomial quadrature mirror filter (1990); Coiflets (1991); Symlets (1992).

Some classical wavelet transforms are not orthogonal.

Bi-orthogonal: Non orthogonal but invertible (ex: for symmetric wavelets). Two filters for the direct, and two others for the inverse.

Wavelets and sparsity

Wavelets perform image compression

- Haar encodes constant signals with one coefficient,
- Db-p encodes $(p\text{-}1)$-order polynomials with p coefficients.

Consequences:

- Polynomial/Smooth signals are encoded with very few coefficients,
- Coarse coefficients encode the smooth underlying signal,
- Detailed coefficients encode non-smooth content of the signal,
- Typical signals are concentrated on few coefficients,
- The remaining coefficients capture only noise components.

Wavelets and sparsity

Wavelets perform image compression

- Haar encodes constant signals with one coefficient,
- Db-p encodes $(p\text{-}1)$-order polynomials with p coefficients.

Consequences:

- Polynomial/Smooth signals are encoded with very few coefficients,
- Coarse coefficients encode the smooth underlying signal,
- Detailed coefficients encode non-smooth content of the signal,
- Typical signals are concentrated on few coefficients,
- The remaining coefficients capture only noise components.

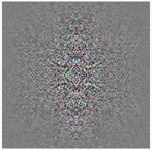
$\Rightarrow$ **Heavy tailed distribution with a peak at zero,**

i.e., **wavelets favor sparsity.**

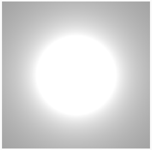
Wavelets as a sparsifying transform

Fourier

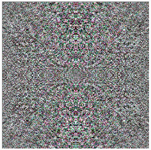
(a) x



(b) $\boldsymbol{F}x$



(c) λ

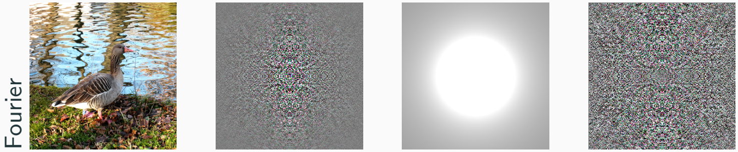


(d) $(\boldsymbol{F}x)_i/\lambda_i$

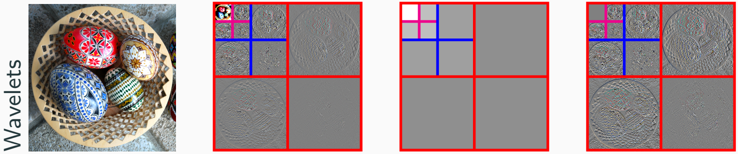
Fourier $\quad (u_i, v_i$ freq. of component $i)$

- $\boldsymbol{E}^* = \boldsymbol{F}/\sqrt{n}$
- $\lambda_i^2 = n^{-1}\mathrm{MPSD}$ and ∞ if $i = 0$
- Arbitrary DC component

Wavelets as a sparsifying transform



(a) x (b) $\boldsymbol{F}x$ (c) λ (d) $(\boldsymbol{F}x)_i/\lambda_i$



(e) x (f) $\boldsymbol{W}x$ (g) λ (h) $(\boldsymbol{W}x)_i/\lambda_i$

Fourier $(u_i, v_i$ freq. of component $i)$

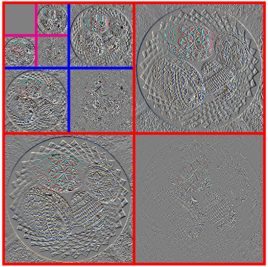
- $\boldsymbol{E}^* = \boldsymbol{F}/\sqrt{n}$
- $\lambda_i^2 = n^{-1}\mathrm{MPSD}$ and ∞ if $i = 0$
- Arbitrary DC component

Wavelets $(j_i$ scale of component $i)$

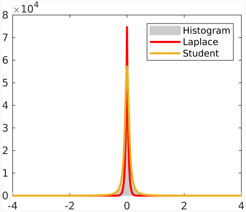
- $\boldsymbol{E}^* = \boldsymbol{W}$
- $\lambda_i = \alpha 2^{j_i - 1}$ and ∞ if $j_i = J$
- Arbitrary coarse component

Distribution of wavelet coefficients

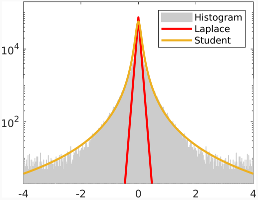
(a) x



(b) $\eta_i = (\boldsymbol{W}x)_i/\lambda_i$

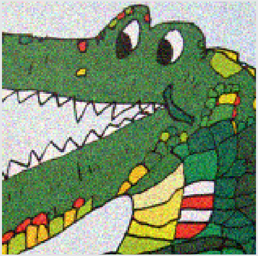


(c) Histogram of η



(d) Histogram of η

Shrinkage in the discrete wavelet domain



(a) y

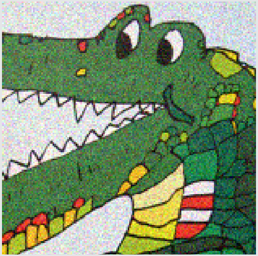
```
sig  = 20
y    = x + sig * nr.randn(*x.shape)

z    = im.dwt(y, 3, h, g)
zhat = shrink(z, lbd, sig)
xhat = im.idwt(zhat, 3, h, g)
```

$$z = \boldsymbol{W} y$$

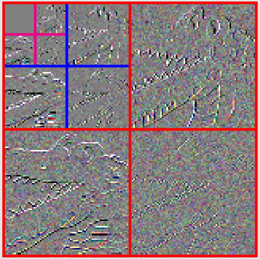
$$\hat{z}_i = s(z_i;\, \lambda_i, \sigma)$$

$$\hat{x} = \boldsymbol{W}^{-1} \hat{z}$$

Shrinkage in the discrete wavelet domain



(a) y



(b) z (Haar)

```
sig  = 20
y    = x + sig * nr.randn(*x.shape)

z    = im.dwt(y, 3, h, g)
zhat = shrink(z, lbd, sig)
xhat = im.idwt(zhat, 3, h, g)
```

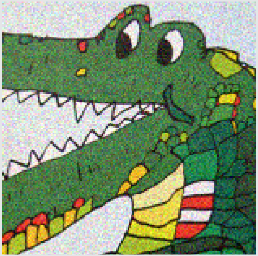
$$z = \boldsymbol{W} y$$

$$\hat{z}_i = s(z_i;\, \lambda_i, \sigma)$$

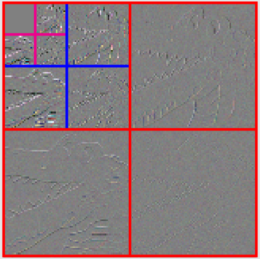
$$\hat{x} = \boldsymbol{W}^{-1} \hat{z}$$

Shrinkage in the wavelet domain

Shrinkage in the discrete wavelet domain



(a) y



(b) $\hat{z}$ (Haar+LMMSE)

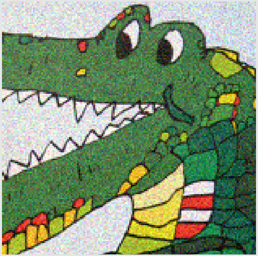
```
    sig  = 20
    y    = x + sig * nr.randn(*x.shape)

    z    = im.dwt(y, 3, h, g)
▶   zhat = shrink(z, lbd, sig)
    xhat = im.idwt(zhat, 3, h, g)
```

$$z = \boldsymbol{W} y$$

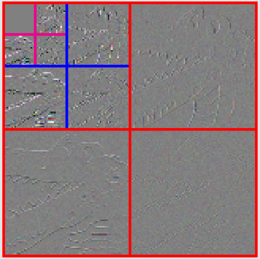
$$\hat{z}_i = s(z_i;\, \lambda_i, \sigma)$$

$$\hat{x} = \boldsymbol{W}^{-1} \hat{z}$$

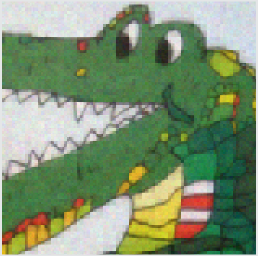
Shrinkage in the discrete wavelet domain



(a) y



(b) $\hat{z}$ (Haar+LMMSE)



(c) $\hat{x}$

```
  sig  = 20
  y    = x + sig * nr.randn(*x.shape)

  z    = im.dwt(y, 3, h, g)
  zhat = shrink(z, lbd, sig)
▶ xhat = im.idwt(zhat, 3, h, g)
```

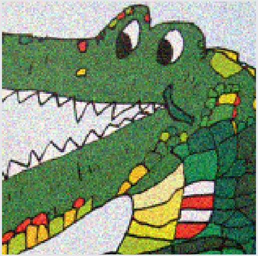
$$z = \boldsymbol{W} y$$

$$\hat{z}_i = s(z_i;\, \lambda_i, \sigma)$$

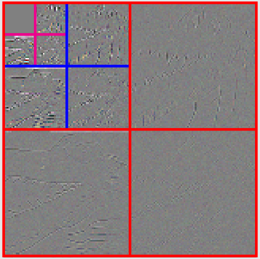
$$\hat{x} = \boldsymbol{W}^{-1} \hat{z}$$

Shrinkage in the wavelet domain

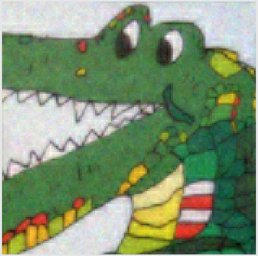
Shrinkage in the discrete wavelet domain



(a) y



(b) $\hat{z}$ (Daubechies+LMMSE)



(c) $\hat{x}$

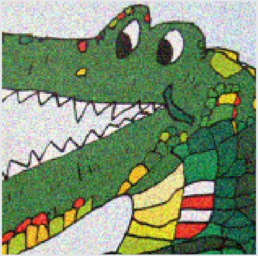
```
    sig  = 20
    y    = x + sig * nr.randn(*x.shape)

    z    = im.dwt(y, 3, h, g)
    zhat = shrink(z, lbd, sig)
▶   xhat = im.idwt(zhat, 3, h, g)
```

$$z = \boldsymbol{W} y$$

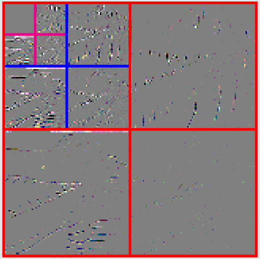
$$\hat{z}_i = s(z_i;\ \lambda_i, \sigma)$$

$$\hat{x} = \boldsymbol{W}^{-1} \hat{z}$$

Shrinkage in the discrete wavelet domain



(a) y



(b) $\hat{z}$ (Daubechies+Soft-T)

(c) $\hat{x}$

```
  sig  = 20
  y    = x + sig * nr.randn(*x.shape)

  z    = im.dwt(y, 3, h, g)
  zhat = shrink(z, lbd, sig)
▶ xhat = im.idwt(zhat, 3, h, g)
```

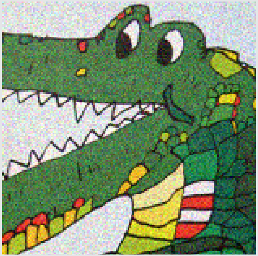
$$z = \boldsymbol{W} y$$

$$\hat{z}_i = s(z_i;\ \lambda_i, \sigma)$$

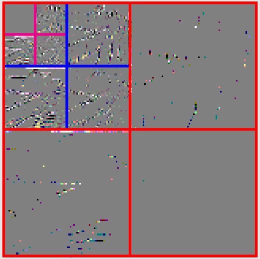
$$\hat{x} = \boldsymbol{W}^{-1} \hat{z}$$

Shrinkage in the wavelet domain

Shrinkage in the discrete wavelet domain



(a) y



(b) $\hat{z}$ (Daubechies+Hard-T)

(c) $\hat{x}$

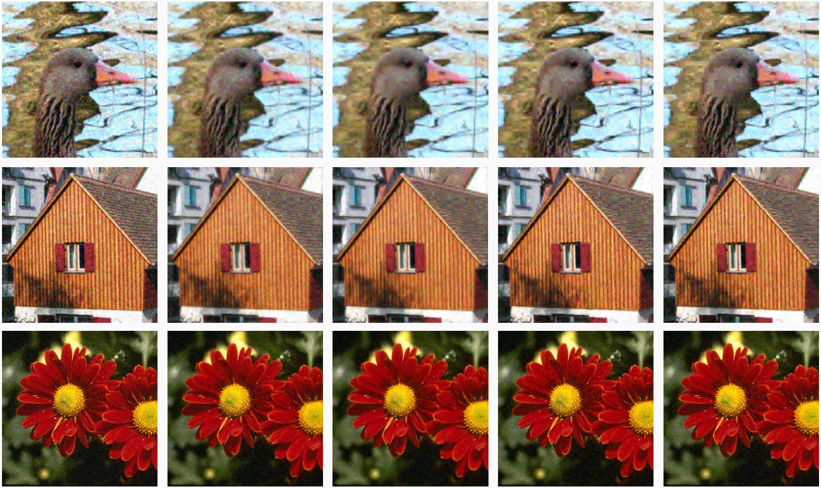
```
sig  = 20
y    = x + sig * nr.randn(*x.shape)

z    = im.dwt(y, 3, h, g)
zhat = shrink(z, lbd, sig)
▶ xhat = im.idwt(zhat, 3, h, g)
```

$$z = \boldsymbol{W} y$$

$$\hat{z}_i = s(z_i;\, \lambda_i, \sigma)$$

$$\hat{x} = \boldsymbol{W}^{-1} \hat{z}$$

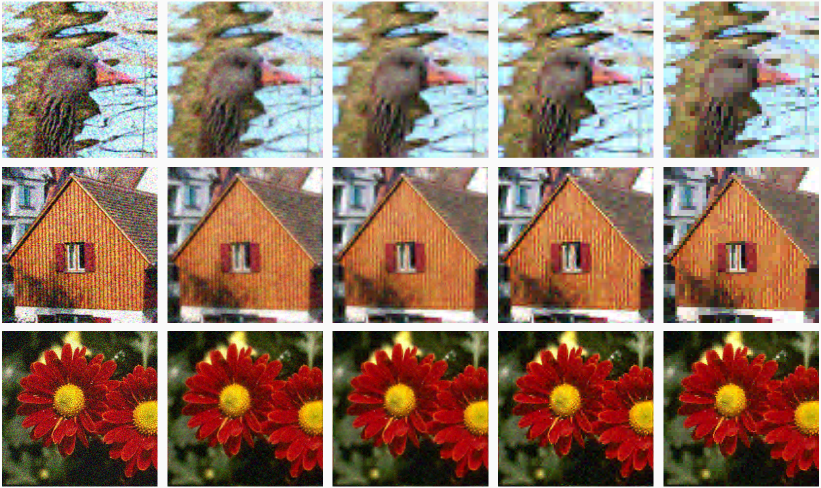
Shrinkage in the wavelet domain



(a) y ($\sigma = 20$) (b) Db2+LMMSE (c) Db2+Soft-T (d) Db2+Hard-T (e) Haar+Hard-T

For large noise: **Blocky effects** and **Ringing artifacts**

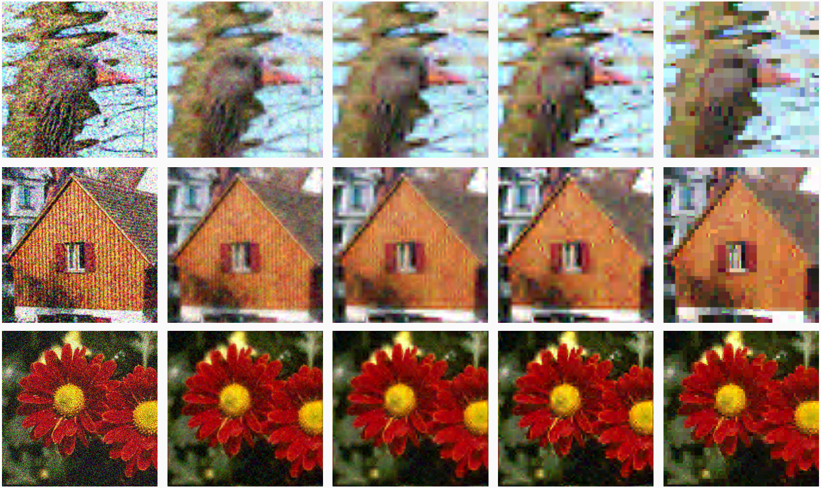
Shrinkage in the wavelet domain



(a) y ($\sigma = 40$) (b) Db2+LMMSE (c) Db2+Soft-T (d) Db2+Hard-T (e) Haar+Hard-T

For large noise: **Blocky effects** and **Ringing artifacts**

Shrinkage in the wavelet domain



(a) y ($\sigma = 60$) (b) Db2+LMMSE (c) Db2+Soft-T (d) Db2+Hard-T (e) Haar+Hard-T

For large noise: **Blocky effects** and **Ringing artifacts**

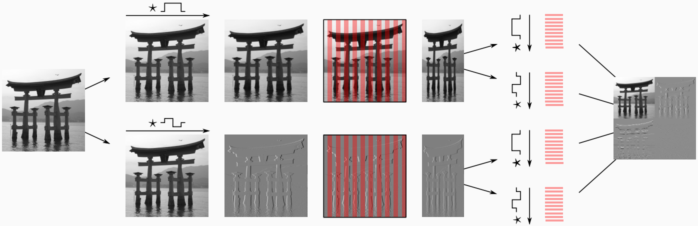
Shrinkage in the wavelet domain

(a) y ($\sigma = 120$) (b) Db2+LMMSE (c) Db2+Soft-T (d) Db2+Hard-T (e) Haar+Hard-T

For large noise: **Blocky effects** and **Ringing artifacts**

Undecimated wavelet transforms

Limits of the discrete wavelet transform



- While Fourier shrinkage is translation invariant:

$$\psi(y^\tau) = \psi(y)^\tau \quad \text{where} \quad y^\tau(s) = y(s+\tau)$$

- Wavelet shrinkage is **not translation invariant**.
- This is due to the decimation step:

$$\boldsymbol{W} = \begin{pmatrix} G\downarrow_2 \\ H\downarrow_2 \end{pmatrix} \in \mathbb{R}^{n\times n} \quad \text{where} \quad M\downarrow_2 = \text{“}\texttt{M[::2, :]}\text{”}$$

- This explains the **blocky artifacts** that we observe.

Undecimated discrete wavelet transform (UDWT)

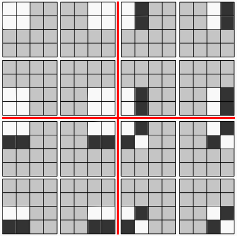


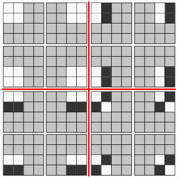
Figure 1 – Haar DWT

- Haar transform groups pixels by clusters of 4.
- Blocks are treated independently to each other.
- When similar neighbor blocks are shrunk differently, it becomes clearly visible in the image.
- This arises all the more as the noise level is large.

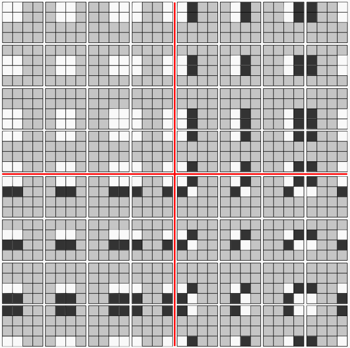
What if we do not decimate?

$\Rightarrow$ **UDWT,** ***aka*****, stationary or translation-invariant wavelet transform.**

Undecimated discrete wavelet transform (UDWT)



Haar discrete wavelet transform (DWT)



Haar undecimated discrete wavelet transform (UDWT)

1-scale DWT

- For a 4×4 image:
  4×4 coefficients.
- For n pixels: $K = n$ coefficients.

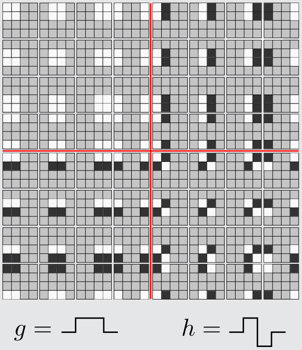
1-scale UDWT

- For a 4×4 image:
  8×8 coefficients.
- For n pixels: $K = 4n$ coeffs.

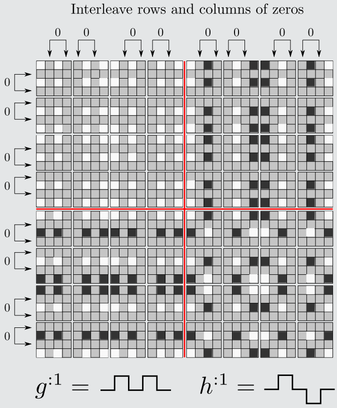
What about multi-scale?

Undecimated discrete wavelet transform (UDWT)

A trous **algorithm (with holes)** **(Holschneider *et al.*, 1989)**



Haar UDWT, first scale

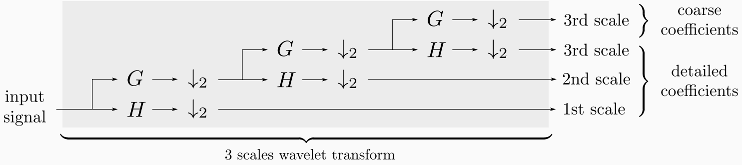


Haar UDWT, second scale

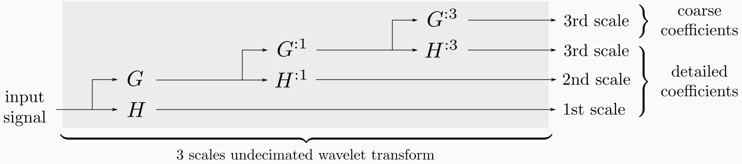
Instead of decimating the coefficients at each scale j, upsample the filters h and g by injecting $2^j - 1$ zeros between each entries.

Undecimated discrete wavelet transform (UDWT)

DWT: Mallat's dyadic pyramidal multi-resolution scheme



UDWT: *A trous* algorithm – $G^{:p}$: inject p zeros between each filter coeffs



Multi-scales: $K = (1 + J(2^d - 1))n$ coeffs (J: #scales, $d = 2$ for images)

Undecimated discrete wavelet transform (UDWT)

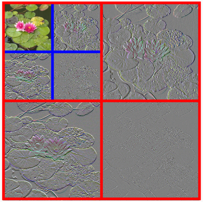
Implementation of 2D UDWT (*A trous* algorithm)

```
def udwt(x, J, h, g):
    if J == 0:
        return x[:, :, np.newaxis]
    tmph = flip(convolve(flip(x), h)) / 2
    tmpg = flip(convolve(flip(x), g)) / 2
    detail = np.stack((convolve(tmpg, h),
                       convolve(tmph, g),
                       convolve(tmph, h)), axis=2)
    coarse = convolve(tmpg, g)
    h2 = interleave0(h)
    g2 = interleave0(g)
    z = np.concatenate((udwt(coarse, J - 1, h2, g2), detail), axis=2)
    return z
```

Linear complexity.

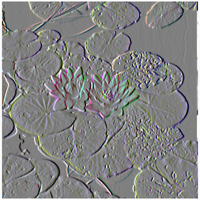
Can be easily modified to reduce memory usage.

Undecimated discrete wavelet transform (UDWT)

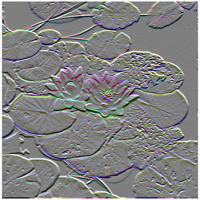


(a) DWT (J=2)

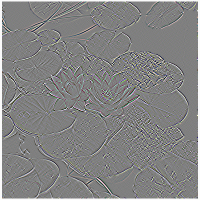
(b) UDWT Coarse Sc.



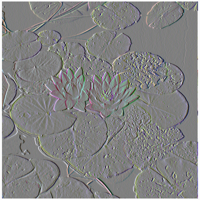
(c) Detailed Scale #1



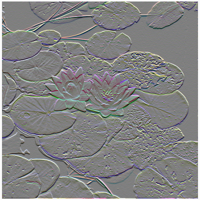
(d) Detailed Scale #2



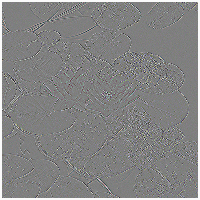
(e) Detailed Scale #3



(f) Detailed Scale #4



(g) Detailed Scale #5



(h) Detailed Scale #6

What about its inverse transform?

Undecimated discrete wavelet transform (UDWT)

DWT – Wavelet basis – and inverse DWT

- The DWT $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ has n columns and n rows.
- The n columns/rows of $\boldsymbol{W}$ are orthonormal.
- The inverse DWT is $\boldsymbol{W}^{-1} = \boldsymbol{W}^*$.
- One-to-one relationship between an image and its wavelet coefficients.

UDWT – Redundant wavelet dictionary

- The UDWT $\bar{\boldsymbol{W}} \in \mathbb{R}^{K \times n}$ has $K = (1 + J(2^d - 1))n$ rows and n columns.
- The rows of $\bar{\boldsymbol{W}}$ cannot be linearly independent: not a basis.
- They are said to form a **redundant/overcomplete wavelet dictionary**.
- Since $\bar{\boldsymbol{W}}$ is non square, it is not invertible.

Note: **redundant dictionaries necessarily favor sparsity.**

Undecimated discrete wavelet transform (UDWT)

Pseudo-inverse UDWT

- Nevertheless, the n columns are orthonormal, then: $\bar{\boldsymbol{W}}^* = \bar{\boldsymbol{W}}^+$
- It satisfies $\bar{\boldsymbol{W}}^+ \bar{\boldsymbol{W}} = \mathrm{Id}_n$, but $\bar{\boldsymbol{W}} \bar{\boldsymbol{W}}^+ \neq \mathrm{Id}_K$
  - image $\xrightarrow{\bar{\boldsymbol{W}}}$ coefficients $\overset{\bar{\boldsymbol{W}}^+}{\to}$ back to the original image,
  - coefficients $\overset{\bar{\boldsymbol{W}}^+}{\to}$ image $\overset{\bar{\boldsymbol{W}}}{\to}$ not necessarily the same coefficients.
- Satisfies the Parseval equality

$$\left\langle \bar{\boldsymbol{W}} x,\, \bar{\boldsymbol{W}} y \right\rangle = \left\langle x,\, \bar{\boldsymbol{W}}^* \bar{\boldsymbol{W}} y \right\rangle = \left\langle x,\, \bar{\boldsymbol{W}}^+ \bar{\boldsymbol{W}} y \right\rangle = \langle x,\, y \rangle$$

- In the vocabulary of linear algebra: $\bar{\boldsymbol{W}}$ is called a **tight-frame**.

Consequence: an algorithm for $\bar{\boldsymbol{W}}^+$ can be obtained.

Undecimated discrete wavelet transform (UDWT)

Implementation of 2D Inverse UDWT

```
def iudwt(z, J, h, g):
    if J == 0:
        return z[:, :, 0]
    h2 = interleave0(h)
    g2 = interleave0(g)
    coarse = iudwt(z[:, :, :-3], J - 1, h2, g2)
    tmpg = convolve(coarse, g[::-1]) + \
           convolve(z[:, :, -3], h[::-1])
    tmph = convolve(z[:, :, -2], g[::-1]) + \
           convolve(z[:, :, -1], h[::-1])
    x = (flip(convolve(flip(tmpg), g[::-1])) +
         flip(convolve(flip(tmph), h[::-1]))) / 2
    return x
```

Linear complexity again.

Can also be easily modified to reduce memory usage.

Can we be more efficient?

Filter bank

- The UDWT of x for subband k, $x \mapsto (\boldsymbol{W}x)_k$ is

  linear and translation invariant (LTI)

  $\Rightarrow$ It's a convolution.

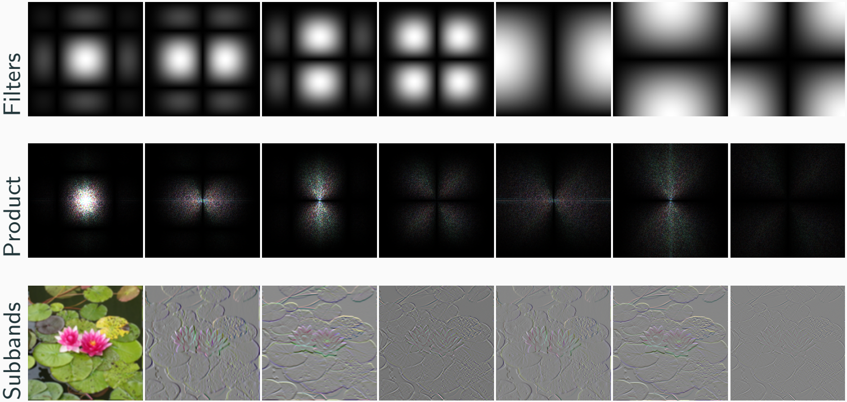
- The UDWT is a filter bank:

  a **set of band-pass filters** that separates
  the input image into multiple components.

- Each filter can be **represented by its frequential response**.

- Direct and inverse transform: **implementation in the Fourier domain**.

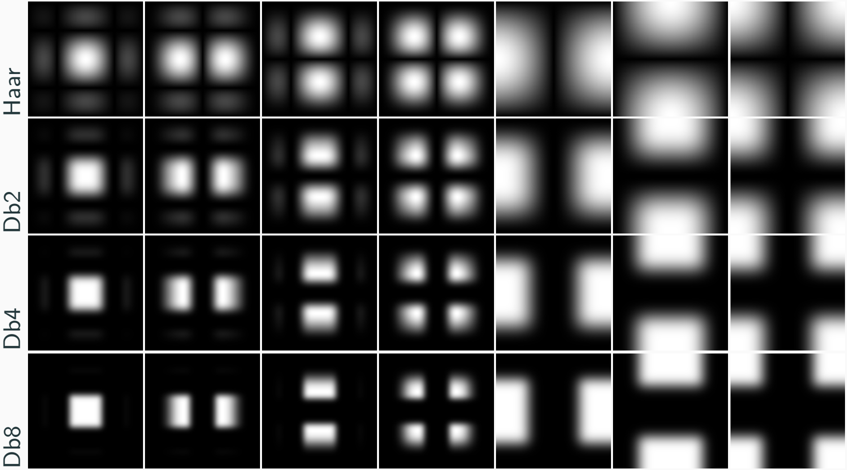
Undecimated discrete wavelet transform (UDWT)



(a) Coarse 2 (b) Details 2 (c) Details 2 (d) Details 2 (e) Details 1 (f) Details 1 (g) Details 1

Haar with $J = 2$ levels of decomposition

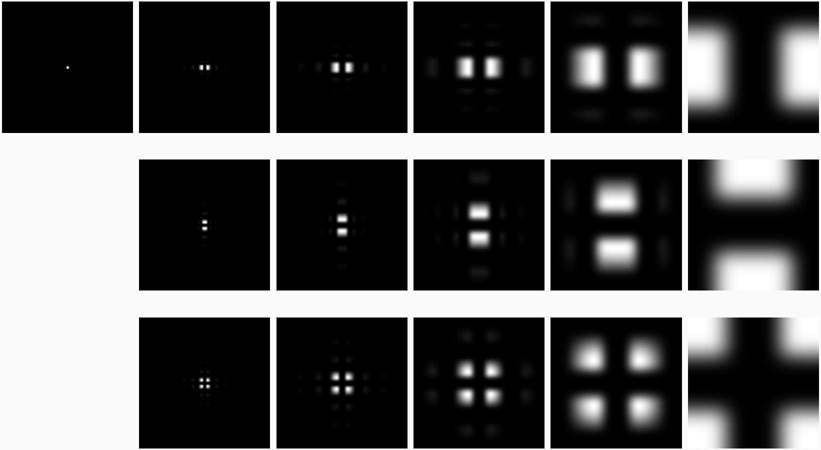
Undecimated discrete wavelet transform (UDWT)



(a) Coarse 2 (b) Details 2 (c) Details 2 (d) Details 2 (e) Details 1 (f) Details 1 (g) Details 1

Haar: band pass with side lobes. **Db8:** closer to ideal band pass.

Undecimated discrete wavelet transform (UDWT)



Db4 with $J = 6$ levels of decomposition

How to create such a filter bank?

Undecimated discrete wavelet transform (UDWT)

UDWT: Creation of the filter bank (offline)

```
def udwt_create_fb(n1, n2, J, h, g, ndim=3):
    if J == 0:
        return np.ones((n1, n2, 1, *[1] * (ndim - 2)))
    h2 = interleave0(h)
    g2 = interleave0(g)
    fbrec = udwt_create_fb(n1, n2, J - 1, h2, g2, ndim=ndim)
    gf1 = nf.fft(fftpad(g, n1), axis=0)
    hf1 = nf.fft(fftpad(h, n1), axis=0)
    gf2 = nf.fft(fftpad(g, n2), axis=0)
    hf2 = nf.fft(fftpad(h, n2), axis=0)
    fb = np.zeros((n1, n2, 4), dtype=np.complex128)
    fb[:, :, 0] = np.outer(gf1, gf2) / 2
    fb[:, :, 1] = np.outer(gf1, hf2) / 2
    fb[:, :, 2] = np.outer(hf1, gf2) / 2
    fb[:, :, 3] = np.outer(hf1, hf2) / 2
    fb = fb.reshape(n1, n2, 4, *[1] * (ndim - 2))
    fb = np.concatenate((fb[:, :, 0:1] * fbrec, fb[:, :, -3:]),
                        axis=2)
    return fb
```

Undecimated discrete wavelet transform (UDWT)

UDWT: Direct transform using the filter bank (online)

```
def fb_apply(x, fb):
    x = nf.fft2(x, axes=(0, 1))
    z = fb * x[:, :, np.newaxis]
    z = np.real(nf.ifft2(z, axes=(0, 1)))
    return z
```

UDWT: Inverse transform using the filter bank (online)

```
def fb_adjoint(z, fb):
    z = nf.fft2(z, axes=(0, 1))
    x = (np.conj(fb) * z).sum(axis=2)
    x = np.real(nf.ifft2(x, axes=(0, 1)))
    return x
```

Much more efficient than previous implementation when $J > 1$

Reconstruction with the UDWT

Shrinkage with UDWT

- Consider a denoising problem $y = x + w$ with noise variance σ^2.
- Shrink the $K \geqslant n$ coefficients independently.

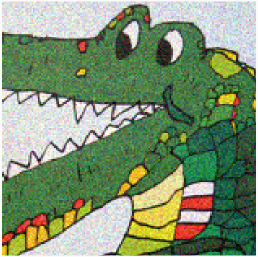
$$\hat{x}^\star = \underbrace{\bar{\boldsymbol{W}}^+}_{\text{Pseudo-inverse}} \hat{z} \quad \text{where} \quad \underbrace{\hat{z}_i = s(z_i;\ \lambda_i, \sigma_i)}_{\text{shrinkage}} \quad \text{and} \quad z = \underbrace{\bar{\boldsymbol{W}}}_{\text{Redundant representation}} y$$

Rule of thumb for soft-thresholding:

- For the orthonormal DWT $\boldsymbol{W}$: increase λ_i as $\sqrt{2}^{d(j_i-1)}$.
- For the tight-frame UDWT $\bar{\boldsymbol{W}}$: increase λ_i as: $2^{d(j_i-1/2)}$.

(j_i scale for coefficient i, $d = 2$ for images).

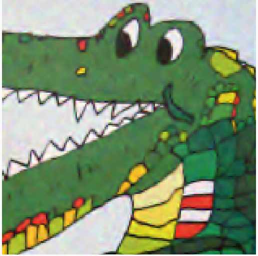
Reconstruction with the UDWT



(a) y

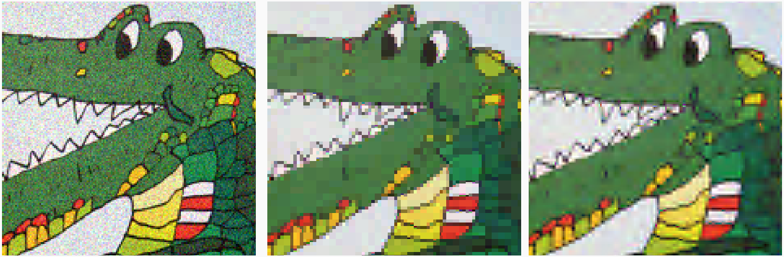
(b) DWT(3)+Haar+HT

(c) DWT(3)+Db2+HT

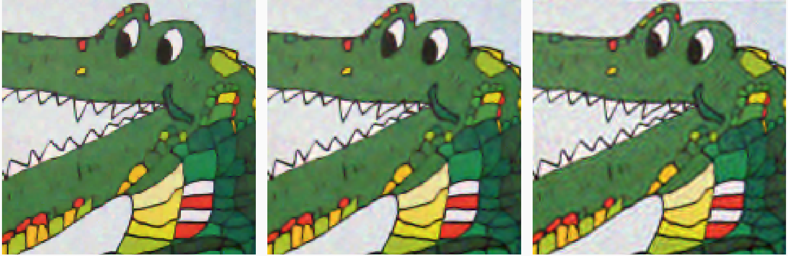


(e) UDWT(3)+Db2+HT

Reconstruction with the UDWT

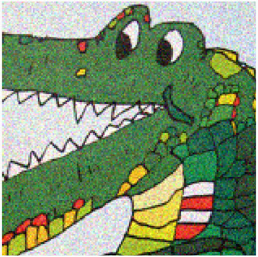


(a) y (b) DWT(3)+Haar+HT (c) DWT(3)+Db2+HT



(d) UDWT(3)+Haar+HT (e) UDWT(3)+Db2+HT (f) UDWT(3)+Db8+HT

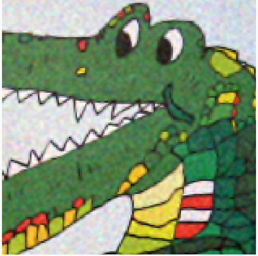
Reconstruction with the UDWT



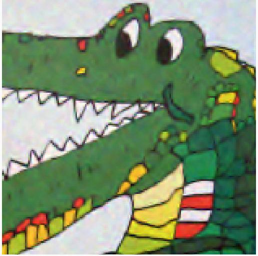
(a) y

(b) DWT(3)+Haar+HT

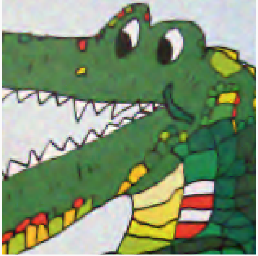
(c) DWT(3)+Db2+HT



(d) UDWT(1)+Db2+HT

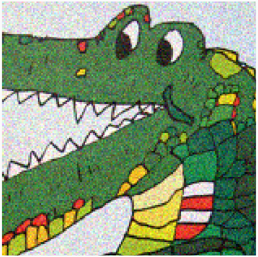


(e) UDWT(3)+Db2+HT



(f) UDWT(5)+Db2+HT

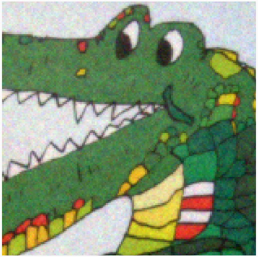
Reconstruction with the UDWT



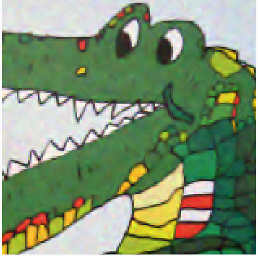
(a) y

(b) DWT(3)+Haar+HT

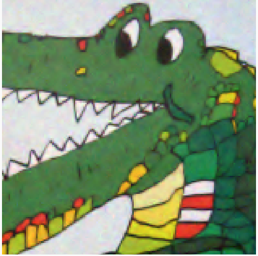
(c) DWT(3)+Db2+HT



(d) UDWT(3)+Db2+Linear



(e) UDWT(3)+Db2+HT



(f) UDWT(3)+Db2+ST

Reconstruction with the UDWT

(a) y ($\sigma = 20$) (b) UDWT+Lin. (c) UDWT+HT (d) DWT+HT

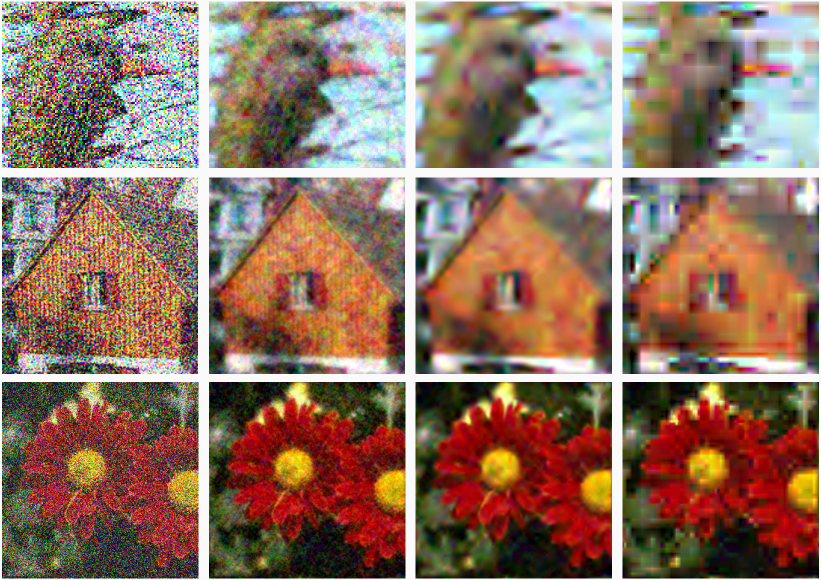
Reconstruction with the UDWT

(a) y ($\sigma = 40$) (b) UDWT+Lin. (c) UDWT+HT (d) DWT+HT

Reconstruction with the UDWT

(a) y ($\sigma = 60$) (b) UDWT+Lin. (c) UDWT+HT (d) DWT+HT

Reconstruction with the UDWT



(a) y ($\sigma = 120$) (b) UDWT+Lin. (c) UDWT+HT (d) DWT+HT

Reconstruction with the UDWT

$$\hat{x}^\star = \underbrace{\bar{\boldsymbol{W}}^+}_{\text{Pseudo-inverse}} \hat{z} \quad \text{where} \quad \underbrace{\hat{z}_i = s(z_i;\ \lambda_i, \sigma_i)}_{\text{shrink } K \text{ coefficients}} \quad \text{and} \quad z = \underbrace{\bar{\boldsymbol{W}}}_{\text{Redundant representation}} y$$

Connection with Bayesian shrinkage?

- Since the rows of $\bar{\boldsymbol{W}}$ are linearly dependent, the **coefficients z_i are necessarily correlated** (non-white).
- Shrink the $K \geqslant n$ coefficients independently, even though they cannot be assumed independent.
- This estimator has **no Bayesian interpretation**, it does not correspond to the MMSE or MAP.

How to use the UDWT in the Bayesian context?

Reconstruction with the UDWT

Bayesian analysis model

Whitening model: Consider $\eta = \mathbf{\Lambda}^{-1/2} \boldsymbol{W} x$ (η coeffs)
such that $\mathbb{E}[\eta] = 0_n$ and $\mathrm{Var}[\eta] = \mathrm{Id}_n$

Analysis: images can be transformed to white coeffs.

⚠ Non-sense when rows of $\boldsymbol{W}$ are redundant.

Bayesian synthesis model

Generative model: Consider $x = \bar{\boldsymbol{W}}^{+} \mathbf{\Lambda}^{1/2} \eta$ (η code)
such that $\mathbb{E}[\eta] = 0_K$ and $\mathrm{Var}[\eta] = \mathrm{Id}_K$

Synthesis: images can be generated from a white code.

☺ Always well-founded.

Reconstruction with the UDWT

Forward model: $y = x + w$

Maximum a Posteriori for the Synthesis model

- Instead of looking for x, consider the MAP for the code η

$$\begin{aligned}\hat{\eta}^\star &\in \underset{\eta \in \mathbb{R}^K}{\operatorname{argmax}}\ p(\eta|y) \\ &= \underset{\eta \in \mathbb{R}^K}{\operatorname{argmin}}\ [-\log p(y|\eta) - \log p(\eta)] \\ &= \underset{\eta \in \mathbb{R}^K}{\operatorname{argmin}}\ \left[\frac{1}{2}\|y - \bar{\boldsymbol{W}}^+ \boldsymbol{\Lambda}^{1/2}\eta\|_2^2 - \log p(\eta)\right]\end{aligned}$$

- Once you get $\hat{\eta}^\star$, generate the image $\hat{x}^\star$ as

$$\hat{x}^\star = \bar{\boldsymbol{W}}^+ \boldsymbol{\Lambda}^{1/2} \hat{\eta}^\star$$

What interpretation?

Reconstruction with the UDWT

Penalized least square with redundant dictionary

- Consider the **redundant wavelet dictionary** $\boldsymbol{D} = \bar{\boldsymbol{W}}^{+}\boldsymbol{\Lambda}^{1/2}$

$$\boldsymbol{D} = (\underbrace{d_1, d_2, \ldots, d_K}_{\text{linearly dependent atoms}}), \quad \|d_i\| = \lambda_i, \quad K \geqslant n$$

- Goal: Look for a code $\eta \in \mathbb{R}^K$, such that $\hat{x}$ close to y

$$\hat{x} = \boldsymbol{D}\eta = \sum_{i=1}^{K} \eta_i d_i = \text{“linear comb. of the } \textbf{redundant atoms } d_i \text{ of } \boldsymbol{D}\text{”}$$

- Since $\boldsymbol{D}$ is redundant, different codes η produce the same image x.

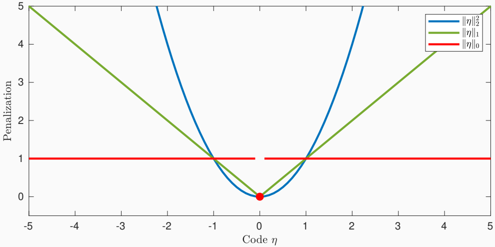
- Penalize independently each η_i to select a relevant one

$$\hat{\eta}^{\star} \in \underset{\eta \in \mathbb{R}^K}{\operatorname{argmin}} \left[\frac{1}{2}\|y - \bar{\boldsymbol{W}}^{+}\boldsymbol{\Lambda}^{1/2}\eta\|_2^2 - \sum_{i=1}^{K} \log p(\eta_i) \right]$$

What choice for $\log p(\eta_i)$?

Penalized least square with redundant dictionary

- $\frac{1}{2}\|y - \boldsymbol{D}\eta\|_2^2 + \frac{\tau^2}{2}\|\eta\|_2^2$, $\quad \|\eta\|_2^2 = \sum_i \eta_i^2$ $\quad \leftarrow$ Ridge regression
- $\frac{1}{2}\|y - \boldsymbol{D}\eta\|_2^2 + \tau\|\eta\|_1$, $\quad \|\eta\|_1 = \sum_i |\eta_i|$ $\quad \leftarrow$ LASSO
- $\frac{1}{2}\|y - \boldsymbol{D}\eta\|_2^2 + \frac{\tau^2}{2}\|\eta\|_0$, $\quad \|\eta\|_0 = \sum_i \mathbf{1}_{\{\eta_i \neq 0\}}$ $\quad \leftarrow$ Sparse regression



When D is redundant, these problems are no longer separable.

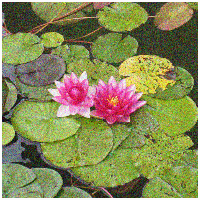
They require large-scale optimization techniques.

Regularizations and optimization

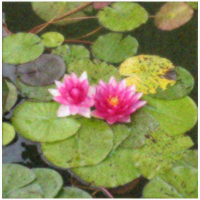
Ridge regression

Ridge/Smooth regression

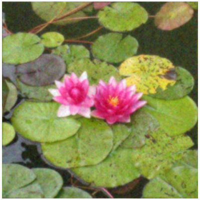
- Convex energy: $E(\eta) = \frac{1}{2}\|y - \boldsymbol{D}\eta\|_2^2 + \frac{\tau^2}{2}\|\eta\|_2^2$
- Gradient: $\nabla E(\eta) = \boldsymbol{D}^*(\boldsymbol{D}\eta - y) + \tau^2\eta$
- Optimality conditions: $\hat{\eta}^\star = (\boldsymbol{D}^*\boldsymbol{D} + \tau^2 \mathrm{Id}_K)^{-1}\boldsymbol{D}^* y$
- For UDWT: this is an LTI filter $\equiv$ convolution (non adaptive)



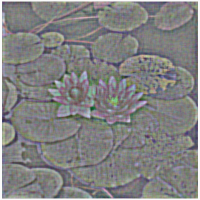
(a) y



(b) Linear shrink



(c) Ridge

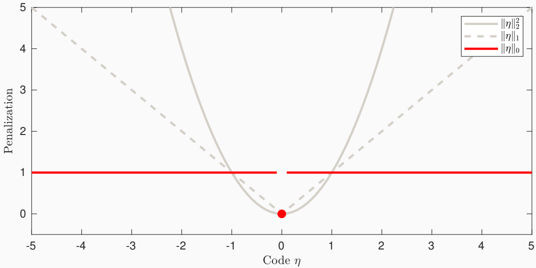


(d) Difference

Ridge $\neq$ Linear shrinkage (except if D is orthogonal).

Sparse regression / ℓ_0 regularization (1/3)

- Energy: $E(\eta) = \frac{1}{2}\|y - \boldsymbol{D}\eta\|_2^2 + \frac{\tau^2}{2}\|\eta\|_0$
- Penalty: $\|\eta\|_0 = \#$non zero elements in η
- Non-convex: $0.5 = \frac{1}{2}(\|0\|_0 + \|1\|_0) < \|0.5\|_0 = 1$
- Produces optimal sparse solutions adapted to the signal ☺
- But, non-differentiable and discontinuous. ☹



Sparse regression / ℓ_0 regularization (2/3)

- If $\boldsymbol{D}$ is orthogonal: solution given by the Hard-Thresholding.
- Otherwise, exact solution obtained by **brute force**:
  - For all possible support $\mathcal{I} \subseteq \{1, \ldots, K\}$ (set of non-zero coefficients)
  - Solve the least square estimation problem:

$$\underset{(\eta_i)_{i \in \mathcal{I}}}{\operatorname{argmin}} \frac{1}{2} \|y - \sum_{i \in \mathcal{I}} \eta_i a_i\|_2^2$$

  - Pick the solution that minimizes E.
- **NP-hard combinatorial problem**:

$$\#\text{subsets} = \sum_{k=0}^{K} \binom{K}{k} = 2^K$$

Sparse regression

Sparse regression / ℓ_0 regularization (3/3)

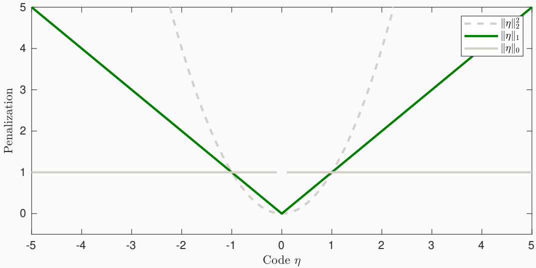
- Sub-optimal solutions can be obtained by **greedy** algorithms.
- **Matching pursuit (MP):** (Mallat, 1993)
  1. Initialization: $r \leftarrow y$, $\eta \leftarrow 0$, $k \leftarrow 0$
  2. Choose i maximizing $|\boldsymbol{D}^* r|_i = |\langle d_i, r \rangle|$
  3. Compute $\alpha = \langle r, d_i \rangle / \|d_i\|_2^2$
  4. Update $r \leftarrow r - \alpha d_i$
  5. Update $\eta_i = \alpha$
  6. Update $k \leftarrow k + 1$
  7. Back to step 2 while $E(\eta) = \frac{1}{2}\|r\|_2^2 + \frac{\tau^2}{2} k$ decreases
- Lots of iterations: complexity $O(kn)$, with k the sparsity of the solution.
- Each iteration requires to compute an UDWT.
- **Extensions**: OMP (Tropp & Gilbert, 2007), CoSaMP (Needel & Tropp, 2009)

Least Absolute Shrinkage and Selection Operator (LASSO)

Convex relaxation: **Take the best of both worlds: sparsity and convexity**

LASSO / ℓ_1 regularization **(Tibshirani 1996)**

- Convex energy: $E(\eta) = \frac{1}{2}\|y - \boldsymbol{D}\eta\|_2^2 + \tau\|\eta\|_1$
- Non-smooth penalty: $\|\eta\|_1 = \sum_{i=1}^{K} |\eta_i|$
- If $\boldsymbol{D}$ is orthogonal: solution given by the Soft-Thresholding.
- Produces also sparse solutions adapted to the signal ☺

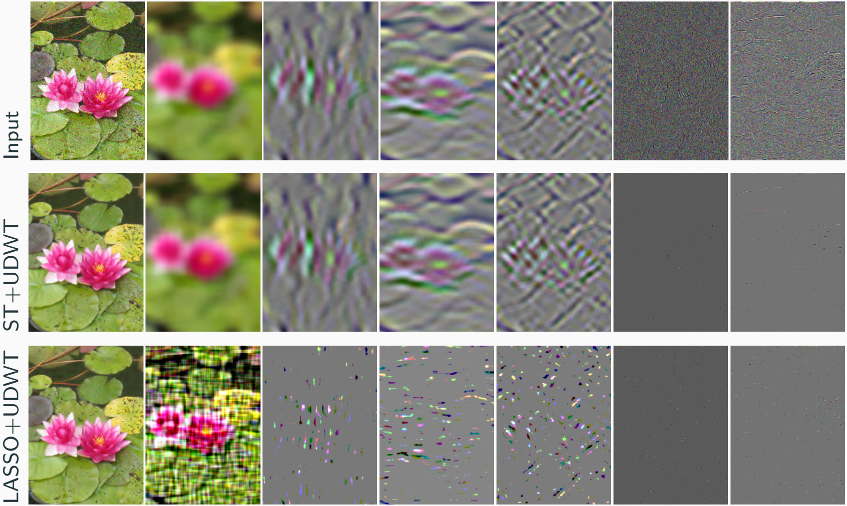


Least Absolute Shrinkage and Selection Operator

(a) Input (b) ST+UDWT (1s) (c) LASSO+UDWT (30s) (d) Difference

Though the solutions look alike, their codes η are very different.

Least Absolute Shrinkage and Selection Operator



(a) Image (b) Coarse 5 (c) Scale 5 (d) Scale 5 (e) Scale 5 (f) Scale 1 (g) Scale 1

The LASSO creates much sparser codes than ST only.

Why use the LASSO if shrinkage in the UDWT provides similar results?

- Shrinkage in the UDWT domain can only be applied for denoising problems.

- The LASSO can be adapted to **inverse-problems**:

$$\hat{x}^\star = \boldsymbol{D}\hat{\eta}^\star \quad \text{with} \quad \hat{\eta}^\star \in \underset{\eta \in \mathbb{R}^K}{\operatorname{argmin}} \left[\frac{1}{2} \|y - \boldsymbol{H}\boldsymbol{D}\eta\|_2^2 + \tau \|\eta\|_1 \right]$$

But it requires solving a non-smooth convex optimization problem.

Solution: use **sub-differential** and **Fermat's rule**.

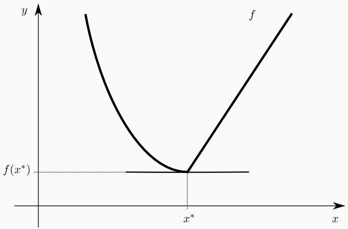
Definition (Sub-differential)

- Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function, $u \in \mathbb{R}^n$ is a **sub-gradient** of f at x^*, if for all $x \in \mathbb{R}^n$

$$f(x) \geqslant f(x^*) + \langle u, x - x^* \rangle .$$

- The **sub-differential** is the set of sub-gradients

$$\partial f(x^*) = \{u \in \mathbb{R}^n : \forall x \in \mathbb{R}^n, f(x) \geqslant f(x^*) + \langle u, x - x^* \rangle\}.$$



If the sub-gradient is unique, f is differentiable and $\partial f(x) = \{\nabla f(x)\}$.

Non-smooth convex optimization

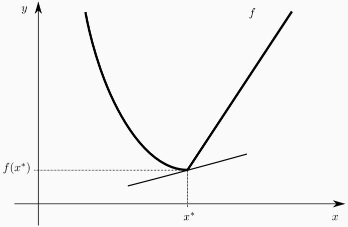
Definition (Sub-differential)

- Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function, $u \in \mathbb{R}^n$ is a **sub-gradient** of f at x^*, if for all $x \in \mathbb{R}^n$

$$f(x) \geqslant f(x^*) + \langle u, x - x^* \rangle .$$

- The **sub-differential** is the set of sub-gradients

$$\partial f(x^*) = \{u \in \mathbb{R}^n : \forall x \in \mathbb{R}^n, f(x) \geqslant f(x^*) + \langle u, x - x^* \rangle\}.$$



If the sub-gradient is unique, f is differentiable and $\partial f(x) = \{\nabla f(x)\}$.

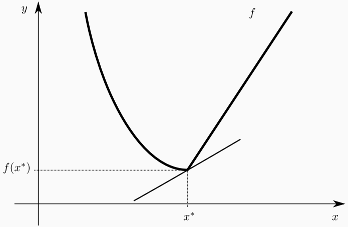
Definition (Sub-differential)

- Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function, $u \in \mathbb{R}^n$ is a **sub-gradient** of f at x^*, if for all $x \in \mathbb{R}^n$

$$f(x) \geqslant f(x^*) + \langle u, x - x^* \rangle .$$

- The **sub-differential** is the set of sub-gradients

$$\partial f(x^*) = \{u \in \mathbb{R}^n : \forall x \in \mathbb{R}^n, f(x) \geqslant f(x^*) + \langle u, x - x^* \rangle\}.$$



If the sub-gradient is unique, f is differentiable and $\partial f(x) = \{\nabla f(x)\}$.

Non-smooth convex optimization

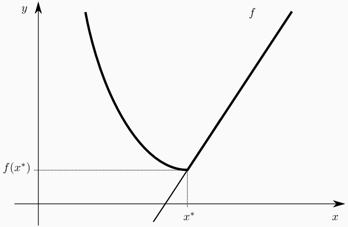
Definition (Sub-differential)

- Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function, $u \in \mathbb{R}^n$ is a **sub-gradient** of f at x^*, if for all $x \in \mathbb{R}^n$

$$f(x) \geqslant f(x^*) + \langle u, x - x^* \rangle .$$

- The **sub-differential** is the set of sub-gradients

$$\partial f(x^*) = \{u \in \mathbb{R}^n : \forall x \in \mathbb{R}^n, f(x) \geqslant f(x^*) + \langle u, x - x^* \rangle\}.$$



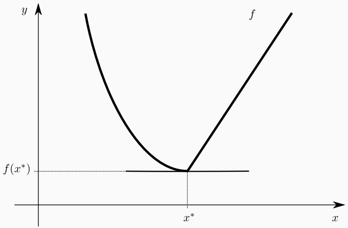
If the sub-gradient is unique, f is differentiable and $\partial f(x) = \{\nabla f(x)\}$.

Theorem (Fermat's rule)

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function, then

$$x^* \in \operatorname*{argmin}_{x \in \mathbb{R}^n} f(x) \quad \Leftrightarrow \quad 0_n \in \partial f(x^*)$$

If f is also differentiable, this corresponds to the standard rule $\nabla f(x^*) = 0_n$.



Minimizers are the only points with a horizontal tangent

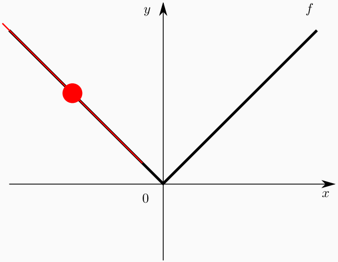
Non-smooth convex optimization

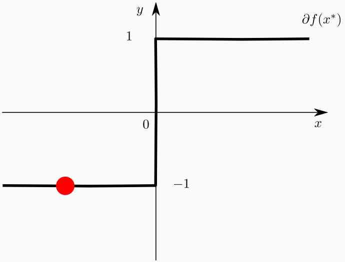
Function (abs):

$$f : \begin{cases} \mathbb{R} & \to \mathbb{R} \\ x & \mapsto |x| \end{cases}$$

Sub-differential (sign)

$$\partial f(x^*) = \begin{cases} \{-1\} & \text{if } x^* \in (-\infty, 0) \\ \{+1\} & \text{if } x^* \in (0, \infty) \\ [-1, 1] & \text{if } x^* = 0 \end{cases}$$

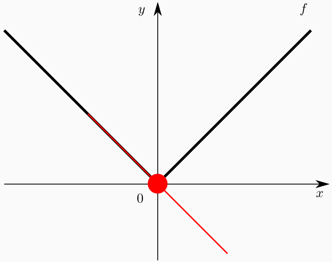




Non-smooth convex optimization

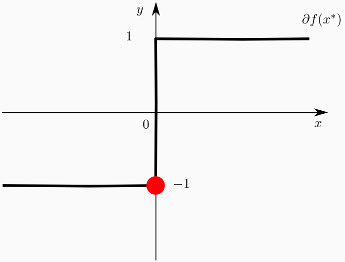
Function (abs):

$$f : \begin{cases} \mathbb{R} & \to \mathbb{R} \\ x & \mapsto |x| \end{cases}$$



Sub-differential (sign)

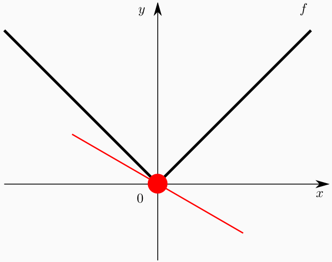
$$\partial f(x^*) = \begin{cases} \{-1\} & \text{if } x^* \in (-\infty, 0) \\ \{+1\} & \text{if } x^* \in (0, \infty) \\ [-1, 1] & \text{if } x^* = 0 \end{cases}$$



Non-smooth convex optimization

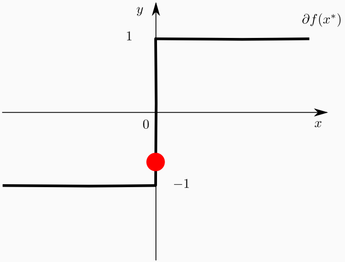
Function (abs):

$$f : \begin{cases} \mathbb{R} & \to \mathbb{R} \\ x & \mapsto |x| \end{cases}$$



Sub-differential (sign)

$$\partial f(x^*) = \begin{cases} \{-1\} & \text{if } x^* \in (-\infty, 0) \\ \{+1\} & \text{if } x^* \in (0, \infty) \\ [-1, 1] & \text{if } x^* = 0 \end{cases}$$

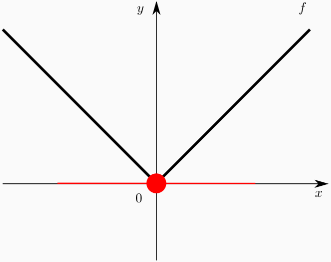


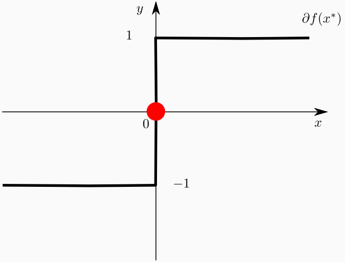
Function (abs):

$$f : \begin{cases} \mathbb{R} & \to \mathbb{R} \\ x & \mapsto |x| \end{cases}$$

Sub-differential (sign)

$$\partial f(x^*) = \begin{cases} \{-1\} & \text{if } x^* \in (-\infty, 0) \\ \{+1\} & \text{if } x^* \in (0, \infty) \\ [-1, 1] & \text{if } x^* = 0 \end{cases}$$

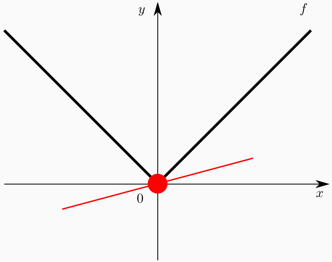




Non-smooth convex optimization

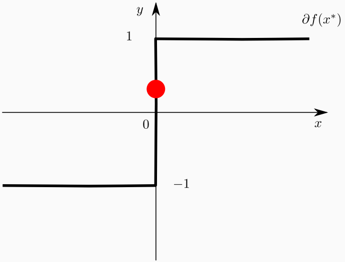
Function (abs):

$$f : \begin{cases} \mathbb{R} & \to \mathbb{R} \\ x & \mapsto |x| \end{cases}$$



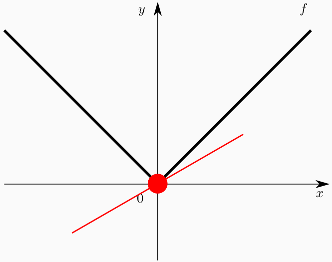
Sub-differential (sign)

$$\partial f(x^*) = \begin{cases} \{-1\} & \text{if } x^* \in (-\infty, 0) \\ \{+1\} & \text{if } x^* \in (0, \infty) \\ [-1, 1] & \text{if } x^* = 0 \end{cases}$$



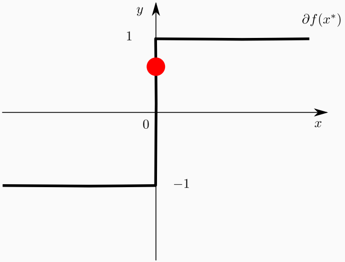
Function (abs):

$$f : \begin{cases} \mathbb{R} & \to \mathbb{R} \\ x & \mapsto |x| \end{cases}$$



Sub-differential (sign)

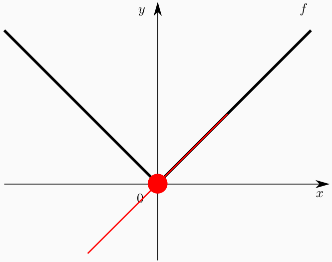
$$\partial f(x^*) = \begin{cases} \{-1\} & \text{if } x^* \in (-\infty, 0) \\ \{+1\} & \text{if } x^* \in (0, \infty) \\ [-1, 1] & \text{if } x^* = 0 \end{cases}$$



Non-smooth convex optimization

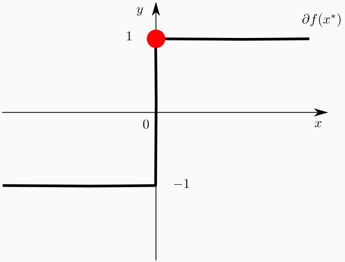
Function (abs):

$$f : \begin{cases} \mathbb{R} & \to \mathbb{R} \\ x & \mapsto |x| \end{cases}$$



Sub-differential (sign)

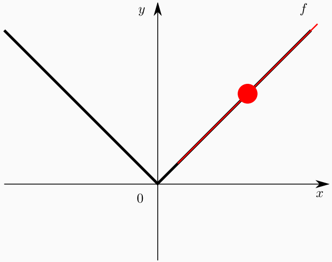
$$\partial f(x^*) = \begin{cases} \{-1\} & \text{if } x^* \in (-\infty, 0) \\ \{+1\} & \text{if } x^* \in (0, \infty) \\ [-1, 1] & \text{if } x^* = 0 \end{cases}$$



Non-smooth convex optimization

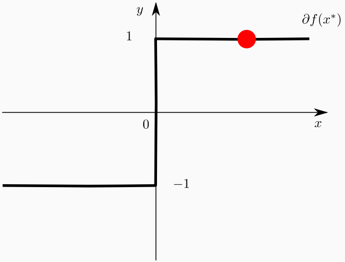
Function (abs):

$$f : \begin{cases} \mathbb{R} & \to \mathbb{R} \\ x & \mapsto |x| \end{cases}$$



Sub-differential (sign)

$$\partial f(x^*) = \begin{cases} \{-1\} & \text{if } x^* \in (-\infty, 0) \\ \{+1\} & \text{if } x^* \in (0, \infty) \\ [-1, 1] & \text{if } x^* = 0 \end{cases}$$



Non-smooth convex optimization

Proximal operator

- Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function (+ some technical conditions). The **proximal operator** of f is

$$\mathrm{Prox}_f(x) = \operatorname*{argmin}_{z \in \mathbb{R}^n} \frac{1}{2}\|z - x\|_2^2 + f(z)$$

- Remark: this minimization problem always has a unique solution, so the proximal operator is without ambiguity a function $\mathbb{R}^n \to \mathbb{R}^n$.
- Always non-expansive:

$$\|\mathrm{Prox}_f(x_1) - \mathrm{Prox}_f(x_2)\| \leqslant \|x_1 - x_2\|$$

- Can be interpreted as a denoiser/shrinkage for the regularity f.

Property

$$\mathrm{Prox}_{\gamma f}(x) = (\mathrm{Id} + \gamma \partial f)^{-1} x$$

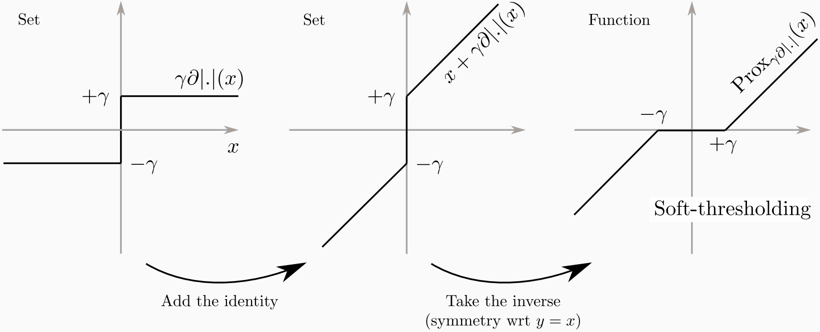
Proof.

$$
\begin{aligned}
\underset{z}{\operatorname{argmin}} \frac{1}{2}\|z-x\|_2^2 + \gamma f(z) \quad &\Leftrightarrow \quad 0 \in \partial\left[\frac{1}{2}\|z-x\|_2^2 + \gamma f(z)\right] \\
&\Leftrightarrow \quad 0 \in \partial\left[\frac{1}{2}\|z-x\|_2^2\right] + \gamma\partial f(z) \\
&\Leftrightarrow \quad 0 \in z - x + \gamma\partial f(z) \\
&\Leftrightarrow \quad x \in z + \gamma\partial f(z) \\
&\Leftrightarrow \quad x \in (\mathrm{Id} + \gamma\partial f)(z) \\
&\Leftrightarrow \quad z = (\mathrm{Id} + \gamma\partial f)^{-1} x
\end{aligned}
$$

Even though $\partial f(x)$ is a set, the pre-image by $\mathrm{Id} + \gamma\partial f$ is unique. $\square$

Soft-thresholding

$$\begin{aligned}\operatorname{Prox}_{\gamma|.|}(x) &= \underset{z\in\mathbb{R}}{\operatorname{argmin}}\ \frac{1}{2}(z-x)^2 + \gamma|z| \\ &= (\mathrm{Id} + \gamma\partial|.|)^{-1}x = \begin{cases} x-\gamma & \text{if} \quad x>\gamma \\ x+\gamma & \text{if} \quad x<-\gamma \\ 0 & \text{otherwise} \end{cases}\end{aligned}$$



Non-smooth convex optimization

Proximal operator of simple functions

Name	$f(x)$	$\mathrm{Prox}_{\gamma f}(x)$
Indicator of convex set $\mathcal{C}$	$\begin{cases} 0 & \text{if } x \in \mathcal{C} \\ \infty & \text{otherwise} \end{cases}$	$\mathsf{Proj}_{\mathcal{C}}(x)$
Square	$\frac{1}{2}\\|x\\|_2^2$	$\dfrac{x}{1+\gamma}$
Abs	$\\|x\\|_1$	Soft-T(x, γ)
Euclidean	$\\|x\\|_2$	$\left(1 - \dfrac{\gamma}{\max(\\|x\\|_2, \gamma)}\right) x$
Square+Affine	$\frac{1}{2}\\|Ax + b\\|_2^2$	$(\mathrm{Id} + \gamma A^* A)^{-1}(x - \gamma A^* b)$
Separability for $x = \binom{x_1}{x_2}$	$g(x_1) + h(x_2)$	$\begin{pmatrix} \mathrm{Prox}_{\gamma g}(x_1) \\ \mathrm{Prox}_{\gamma h}(x_2) \end{pmatrix}$

More exhaustive list: `http://proximity-operator.net`

Proximal minimization

- Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function (+ some technical conditions). Then, whatever the initialization x^0 and $\gamma > 0$, the sequence

$$x^{k+1} = \operatorname{Prox}_{\gamma f}(x^k)$$

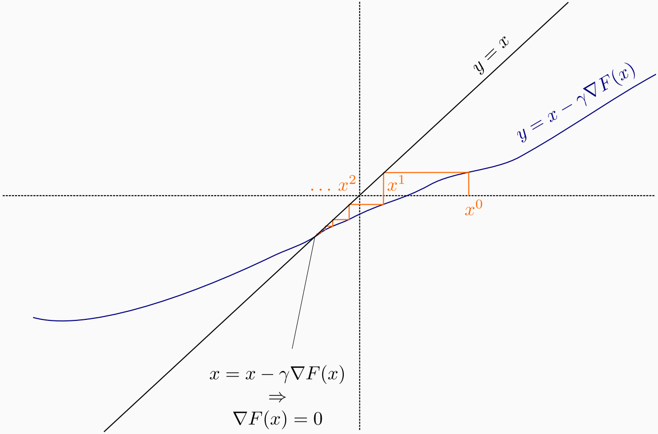
converges towards a **global minimizer** of f.

$$\operatorname{Prox}_{\gamma f}(x^k) = (\mathrm{Id} + \gamma \partial f)^{-1} x^k = \operatorname*{argmin}_z \frac{1}{2}\|z - x^k\|_2^2 + \gamma f(z)$$

Compared to gradient descent

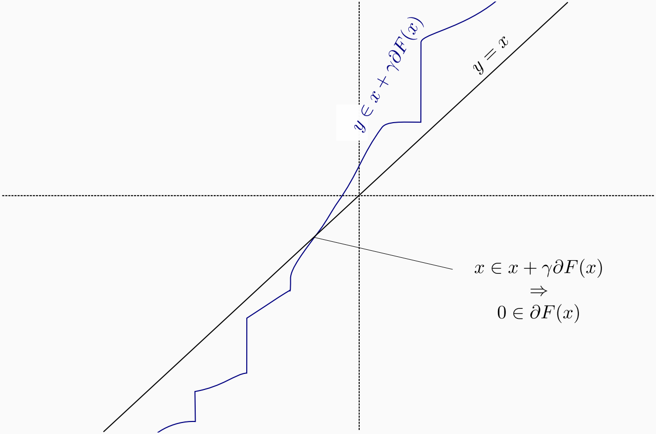
- No need to be differentiable,
- No need to have Lipschitz gradient,
- Works whatever the parameter γ,
- Requires to solve an optimization problem at each step.

Non-smooth convex optimization



Gradient descent:
read x^k on the x-axis and evaluate its image by the function $x - \gamma\nabla F(x)$.

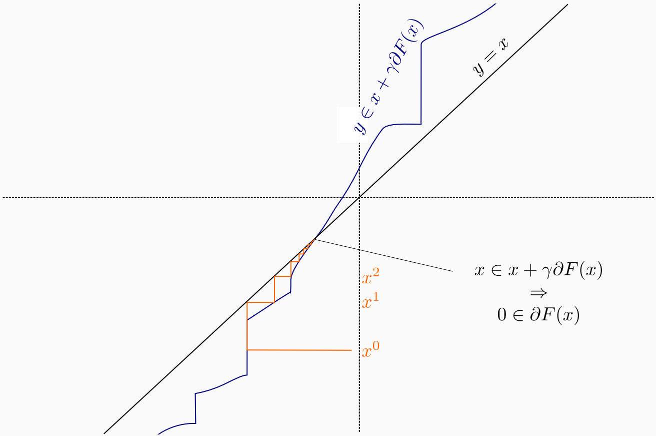
Non-smooth convex optimization



Proximal minimization:

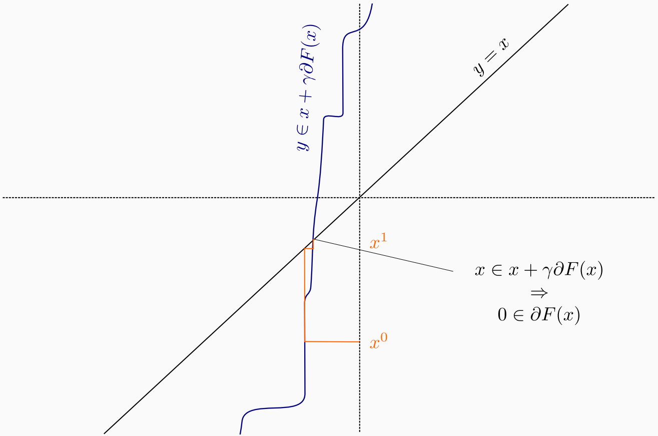
Look at the set $x + \gamma\partial F(x)$

Non-smooth convex optimization



Proximal minimization:
read x^k on the y-axis and evaluate its pre-image by $x + \gamma\nabla F(x)$.

Non-smooth convex optimization



Proximal minimization:

the larger γ the faster, but the inversion becomes harder (ill-conditioned).

Toy example

- Consider the smoothing regularization problem

$$F(x) = \frac{1}{2}\|\nabla x\|_{2,2}^2$$

- Its sub-gradient is thus given by

$$\partial F(x) = \{\nabla F(x) = -\Delta x\}$$

- The proximal minimization reads as

$$\begin{aligned} x^{k+1} &= (\mathrm{Id} + \gamma \partial F)^{-1} x^k \\ &= (\mathrm{Id} - \gamma \Delta)^{-1} x^k \end{aligned}$$

- This is exactly the implicit Euler scheme for the Heat equation.

Can we apply proximal minimization for the LASSO?

Proximal splitting methods

- The proximal operator may not have a closed form.
- Computing it may be as difficult as solving the original problem ☹
- Solution: use **proximal splitting methods**, a family of techniques developed for non-smooth convex problems.
- Idea: split the problem into subproblems, that involve
  - gradient descent steps for smooth terms,
  - proximal steps for simple convex terms.

Non-smooth convex optimization

$$\min_{x \in \mathbb{R}^n} \{E(x) = F(x) + G(x)\}$$

Proximal forward-backward algorithm

- Assume F is convex and differentiable with **L-Lipschitz gradient**

$$\|\nabla F(x_1) - \nabla F(x_2)\|_2 \leqslant L\|x_1 - x_2\|_2, \quad \text{for all } x_1,\ x_2\ .$$

- Assume G is convex and **simple**, *i.e.*, its prox is known in closed form

$$\mathrm{Prox}_{\gamma G}(x) = \operatorname*{argmin}_z \frac{1}{2}\|z - x\|_2^2 + \gamma G(z)$$

- The **proximal forward-backward algorithm** reads

$$x^{k+1} = \mathrm{Prox}_{\gamma G}(x^k - \gamma \nabla F(x^k))$$

- For $0 < \gamma < 2/L$, it converges to a minimizer of $E = F + G$.

Aka, explicit-implicit scheme by analogy with PDE discretization schemes.

Non-smooth convex optimization

The LASSO problem: $$E(\eta) = \underbrace{\frac{1}{2}\|y - \boldsymbol{A}\eta\|_2^2}_{F(\eta)} + \underbrace{\tau\|\eta\|_1}_{G(\eta)=\sum_i |\eta_i|}, \quad \boldsymbol{A} = \boldsymbol{H}\boldsymbol{D}$$

Iterative Soft-Thresholding Algorithm (ISTA) **(Daubechies, 2004)**

- F is convex and differentiable with **L-Lipschitz gradient**

$$\nabla F(\eta) = \boldsymbol{A}^*(\boldsymbol{A}\eta - y) \quad \text{with} \quad L = \|\boldsymbol{A}\|_2^2$$

- G is convex and **simple**, in fact separable:

$$\mathrm{Prox}_{\gamma G}(\eta)_i = \text{Soft-T}(\eta_i, \gamma\tau)$$

- The proximal forward-backward algorithm reads for $0 < \gamma < 2/L$

$$\eta^{k+1} = \text{Soft-T}(\eta^k - \gamma(\boldsymbol{A}^*\boldsymbol{A}\eta^k - \boldsymbol{A}^*y), \gamma\tau)$$

and is known as **Iterative Soft-Thresholding Algorithm (ISTA)**.

- Finally: $$\hat{x}^\star = \bar{\boldsymbol{D}}\hat{\eta}^\star$$

Preconditioned ISTA (1/2)

- Remark

$$\hat{\eta}^\star \in \underset{\eta \in \mathbb{R}^K}{\operatorname{argmin}} \frac{1}{2}\|y - \boldsymbol{A}\eta\|_2^2 + \tau\|\eta\|_1, \quad \boldsymbol{A} = \boldsymbol{H}\underbrace{\bar{\boldsymbol{W}}^+\boldsymbol{\Lambda}^{1/2}}_{\boldsymbol{D}}$$

$$\in \underset{\eta \in \mathbb{R}^K}{\operatorname{argmin}} \frac{1}{2}\|y - \boldsymbol{H}\bar{\boldsymbol{W}}^+\boldsymbol{\Lambda}^{1/2}\eta\|_2^2 + \tau\|\eta\|_1$$

- $\boldsymbol{\Lambda}^{1/2}$ invertible: bijection between $z = \boldsymbol{\Lambda}^{1/2}\eta$ and $\eta = \boldsymbol{\Lambda}^{-1/2}z$
- Solving for η is equivalent to solve a weighted LASSO for z

$$\hat{z}^\star \in \underset{z \in \mathbb{R}^K}{\operatorname{argmin}} \frac{1}{2}\|y - \boldsymbol{H}\bar{\boldsymbol{W}}^+ z\|_2^2 + \tau\|\boldsymbol{\Lambda}^{-1/2}z\|_1$$

$$\in \underset{z \in \mathbb{R}^K}{\operatorname{argmin}} \frac{1}{2}\|y - \boldsymbol{B}z\|_2^2 + \sum_{i=1}^{K} \frac{\tau}{\lambda_i}|z_i|, \quad \boldsymbol{B} = \boldsymbol{H}\bar{\boldsymbol{W}}^+$$

- In practice, this equivalent problem has **better conditioning**.

Equivalent to: $$E(z) = \underbrace{\frac{1}{2}\|y - \boldsymbol{B}z\|_2^2}_{F(z)} + \underbrace{\tau\|\boldsymbol{\Lambda}^{-1/2}z\|_1}_{G(z)=\sum_i \frac{\tau}{\lambda_i}|z_i|}, \quad \boldsymbol{B} = \boldsymbol{H}\bar{\boldsymbol{W}}^+$$

Preconditioned ISTA (2/2)

$$\nabla F(z) = \boldsymbol{B}^*(\boldsymbol{B}z - y) \quad \text{with} \quad L = \|\boldsymbol{B}\|_2^2$$

$$\mathrm{Prox}_{\gamma G}(z)_i = \text{Soft-T}\left(z_i, \frac{\gamma\tau}{\lambda_i}\right)$$

- ISTA becomes for $0 < \gamma < 2/L$

$$z^{k+1} = \text{Soft-T}\left(z^k - \gamma(\boldsymbol{B}^*\boldsymbol{B}z^k - \boldsymbol{B}^*y), \frac{\gamma\tau}{\lambda_i}\right)$$

- Finally: $$\hat{x}^\star = \bar{\boldsymbol{W}}^+\hat{z}^\star$$

- Leads to **larger steps γ, better conditioning, and faster convergence**.

Non-smooth convex optimization

$$z^{k+1} = \mathrm{Prox}_{\gamma G}(z^k - \gamma \nabla F(z^k))$$
$$= \text{Soft-T}\left(z^k - \gamma(\boldsymbol{B}^* \boldsymbol{B} z^k - \boldsymbol{B}^* y), \frac{\gamma\tau}{\lambda_i}\right) \quad \text{with} \quad \boldsymbol{B} = \boldsymbol{H}\bar{\boldsymbol{W}}^+$$

Bredies & Lorenz (2007): **$E(z^k) - E(z^\star)$ decays with rate $O(1/k)$**

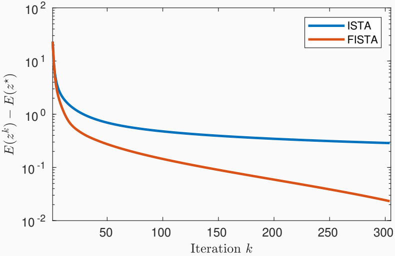
Fast ISTA (FISTA)

$$z^{k+1} = \mathrm{Prox}_{\gamma G}\left(\tilde{z}^k - \gamma \nabla F(\tilde{z}^k)\right)$$
$$\tilde{z}^{k+1} = z^{k+1} + \frac{t_k - 1}{t_{k+1}}(z^{k+1} - z^k)$$
$$t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}, t_0 = 1$$

Beck & Teboulle (2009): **$E(z^k) - E(z^\star)$ decays with rate $O(1/k^2)$**

Non-smooth convex optimization

(a) Input y: motion blur + noise ($\sigma = 2$)

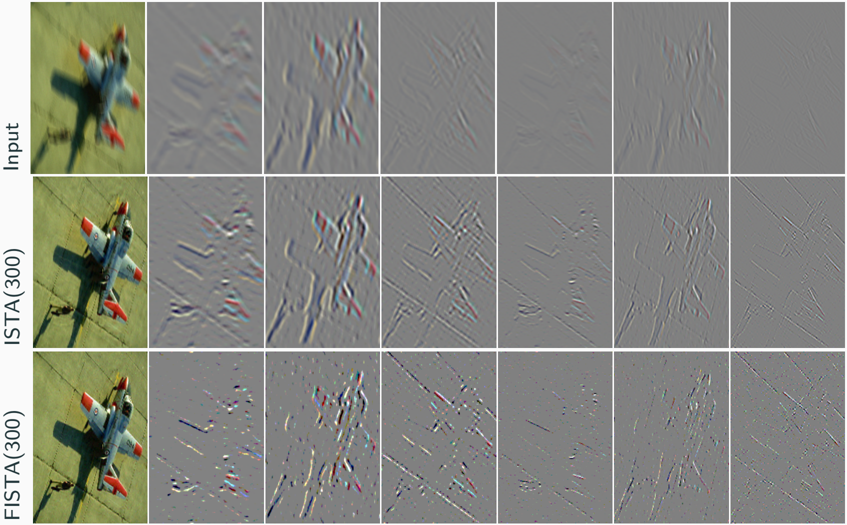


(b) Convergence profiles

(c) Deconvolution ISTA(300)+UDWT

(d) Deconvolution FISTA(300)+UDWT

Non-smooth convex optimization



(a) Image (b) Scale 4 (c) Scale 4 (d) Scale 4 (e) Scale 3 (f) Scale 3 (g) Scale 3

FISTA converges faster: sparser codes given a limited time budget

Sparsity: synthesis vs analysis

Sparse reconstruction: synthesis vs analysis

Sparse synthesis model with UDWT

- LASSO: $$\hat{\eta}^\star \in \operatorname*{argmin}_{\eta \in \mathbb{R}^K} \frac{1}{2}\|y - \boldsymbol{H}\bar{\boldsymbol{W}}^+ \boldsymbol{\Lambda}^{1/2}\eta\|_2^2 + \tau\|\eta\|_1$$
- Using the change of variable $\eta = \boldsymbol{\Lambda}^{-1/2} z$:

$$\hat{z}^\star \in \operatorname*{argmin}_{z \in \mathbb{R}^K} \frac{1}{2}\|y - \boldsymbol{H}\bar{\boldsymbol{W}}^+ z\|_2^2 + \tau\|\boldsymbol{\Lambda}^{-1/2} z\|_1$$

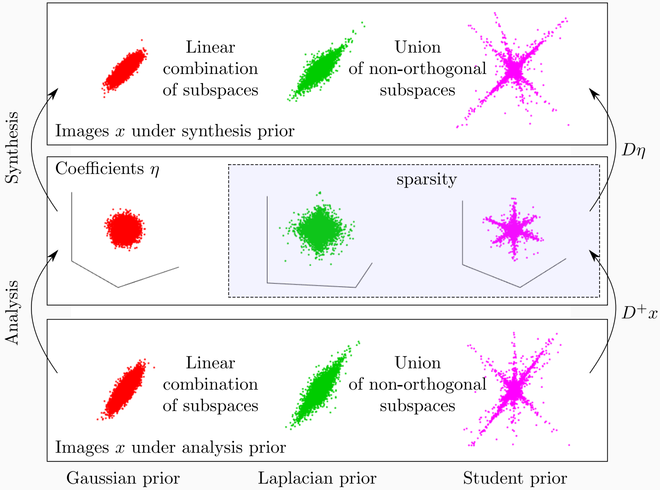
Sparse analysis model with UDWT

- What about?

$$\hat{x}^\star \in \operatorname*{argmin}_{x \in \mathbb{R}^n} \frac{1}{2}\|y - \boldsymbol{H}x\|_2^2 + \tau\|\boldsymbol{\Lambda}^{-1/2}\bar{\boldsymbol{W}}x\|_1$$

- The change of variable $\eta = \boldsymbol{\Lambda}^{-1/2}\bar{\boldsymbol{W}}x$ is not one-to-one.
- **The two problems are not equivalent** (unless $\bar{\boldsymbol{W}}$ is invertible).

Sparse reconstruction: synthesis vs analysis



Analysis versus synthesis (Elad, Milanfar, Rubinstein, 2007)

Generative: generate good images

$$\hat{x}^\star = \boldsymbol{D}\hat{\eta}^\star \quad \text{with} \quad \hat{\eta}^\star \in \underset{\eta \in \mathbb{R}^K}{\operatorname{argmin}} \frac{1}{2}\|y - \boldsymbol{H}\boldsymbol{D}\eta\|_2^2 + \tau\|\eta\|_p^p, \quad p \geqslant 0$$

Synthesis: images are linear combinations of a few columns of $\boldsymbol{D}$.

Bayesian interpretation: MAP for the sparse code η.

Discriminative: discriminate between good and bad images

$$\hat{x}^\star \in \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2}\|y - \boldsymbol{H}x\|_2^2 + \tau\|\boldsymbol{\Gamma}x\|_p^p, \quad p \geqslant 0$$

Analysis: images are correlated with a few rows of $\boldsymbol{\Gamma}$.

Bayesian interpretation: MAP for x with an improper *Gibbs prior*.

Sparse reconstruction: synthesis vs analysis

$$\hat{\eta}^\star \in \underset{\eta \in \mathbb{R}^K}{\operatorname{argmin}} \frac{1}{2}\|y - \boldsymbol{H}\boldsymbol{D}\eta\|_2^2 + \tau\|\eta\|_p^p \qquad (\ell_p^p\text{-synthesis})$$

$$\hat{x}^\star \in \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2}\|y - \boldsymbol{H}x\|_2^2 + \tau\|\boldsymbol{\Gamma}x\|_p^p \qquad (\ell_p^p\text{-analysis})$$

Common properties

	Solution	Problem
$p = 0$	Optimal sparse	Non-convex & discontinuous (NP-hard)
$0 < p < 1$	Sparse	Non-convex & continuous but non-smooth
$p = 1$	Sparse	Convex & continuous but non-smooth
$p > 1$	Smooth	Convex & differentiable
$p = 2$	Linear	Quadratic

- $\boldsymbol{\Gamma}$ square and invertible $\Rightarrow$ equivalent for $\boldsymbol{D} = \boldsymbol{\Gamma}^{-1}$.
- $\boldsymbol{\Gamma}$ full-rank and $p = 2$ $\Rightarrow$ equivalent for $\boldsymbol{D} = \boldsymbol{\Gamma}^{+}$.
- LTI dictionaries $\Rightarrow$ redundant filter bank.

Sparse reconstruction: synthesis vs analysis

$$\hat{\eta}^\star \in \operatorname*{argmin}_{\eta \in \mathbb{R}^K} \frac{1}{2}\|y - \boldsymbol{H}\boldsymbol{D}\eta\|_2^2 + \tau\|\eta\|_p^p \qquad (\ell_p^p\text{-synthesis})$$

$$\hat{x}^\star \in \operatorname*{argmin}_{x \in \mathbb{R}^n} \frac{1}{2}\|y - \boldsymbol{H}x\|_2^2 + \tau\|\boldsymbol{\Gamma}x\|_p^p \qquad (\ell_p^p\text{-analysis})$$

Synthesis

- $\boldsymbol{D}$: synthesis dictionary.
- Atoms need to span images.

  $\Rightarrow$ Low- & high-pass filters

  $\Rightarrow \mathrm{Im}[\boldsymbol{D}] \approx \mathbb{R}^n$
- Redundancy favor sparsity.
- K dimensional problem ($> n$).
- Prior separable.

Analysis

- $\boldsymbol{\Gamma}$: analysis dictionary.
- Atoms need to sparsify images.

  $\Rightarrow$ High-pass filters only

  $\Rightarrow \mathrm{Ker}[\boldsymbol{\Gamma}] \neq \emptyset$ ($\supset$ DC, coarse)
- Redundancy decreases sparsity.
- n dimensional problem ($< K$).
- Prior non-separable.

Quiz: What analysis dictionary is LTI and not too redundant?

Sparse reconstruction: synthesis vs analysis

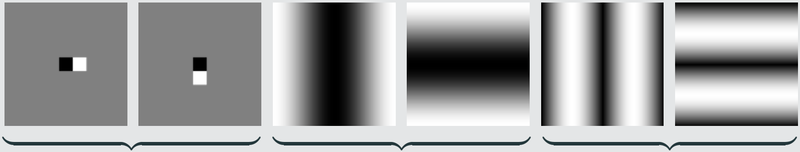
$$\hat{\eta}^\star \in \underset{\eta \in \mathbb{R}^K}{\operatorname{argmin}} \frac{1}{2}\|y - \boldsymbol{H}\boldsymbol{D}\eta\|_2^2 + \tau\|\eta\|_p^p \qquad (\ell_p^p\text{-synthesis})$$

$$\hat{x}^\star \in \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2}\|y - \boldsymbol{H}x\|_2^2 + \tau\|\boldsymbol{\Gamma}x\|_p^p \qquad (\ell_p^p\text{-analysis})$$

Link between analysis models and variational methods

- $p = 2$: Analysis model = Tikhonov regularization.
- $p = 1$ & $\boldsymbol{\Gamma} = \nabla$: Analysis model = anisotropic Total-Variation (TV)

TV filter bank = Horizontal and vertical gradient



Spatial filter bank | Spectral f.b. (real part) | Spectral f.b. (imaginary part)

Can we use proximal forward-backward for ℓ_1-analysis prior?

Non-smooth optimization

$$\hat{x}^\star \in \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \; \underbrace{\frac{1}{2}\|y - \boldsymbol{H}x\|_2^2}_{F(x)} + \underbrace{\tau\|\boldsymbol{\Gamma}x\|_1}_{G(x)} \qquad (\ell_1\text{-analysis})$$

Proximal forward-backward for the ℓ_1-analysis problem?

- F convex and differentiable
- G convex but **not simple** (not separable)

$\longrightarrow$ cannot use proximal forward backward ☹

- **Exception:** for denoising $\boldsymbol{H} = \mathrm{Id}_n$ (see: Chambolle algorithm, 2004)

Need another proximal optimization technique.

Non-smooth optimization

$$\min_{x \in \mathbb{R}^n} \{E(x) = F(x) + G(x)\}$$

Alternating direction method of multipliers (ADMM) **(∼1970)**

- Assume F and G are **convex and simple** (+ some mild conditions).
- For any initialization $x^0, \tilde{x}^0$ and d^0, the ADMM algorithm reads as

$$\begin{aligned} x^{k+1} &= \mathrm{Prox}_{\gamma F}(\tilde{x}^k + d^k) \\ \tilde{x}^{k+1} &= \mathrm{Prox}_{\gamma G}(x^{k+1} - d^k) \\ d^{k+1} &= d^k - x^{k+1} + \tilde{x}^{k+1} \end{aligned}$$

- For $\gamma > 0$, x^k converges to a minimizer of $E = F + G$.

Fast version: FADMM, similar idea as for FISTA (Goldstein *et al.*, 2014).

Related concepts: Lagrange multipliers, Duality, Legendre transform.

How to use it for ℓ_1 analysis priors?

Non-smooth optimization

$$\hat{x}^\star \in \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \; \frac{1}{2}\|y - \boldsymbol{H}x\|_2^2 + \tau\|\boldsymbol{\Gamma}x\|_1 \qquad (\ell_1\text{-analysis})$$

ADMM + Variable splitting (1/3)

- Define: $X = \begin{pmatrix} x \\ z \end{pmatrix} \in \mathbb{R}^{n+K}$
- Consider: $E(X) = F(X) + G(X)$

with:
$$\begin{cases} F\begin{pmatrix} x \\ z \end{pmatrix} = \|y - \boldsymbol{H}x\|_2^2 + \tau\|z\|_1 \\ G\begin{pmatrix} x \\ z \end{pmatrix} = \begin{cases} 0 & \text{if} \quad \boldsymbol{\Gamma}x = z \\ \infty & \text{otherwise} \end{cases} \end{cases}$$

- Remark 1: Minimizing E solves the ℓ_1-analysis problem.
- Remark 2: F and G are convex and simple $\Rightarrow$ ADMM applies.

Non-smooth optimization

$$\hat{x}^\star \in \underset{x \in \mathbb{R}^n}{\operatorname{argmin}}\ \frac{1}{2}\|y - \boldsymbol{H}x\|_2^2 + \tau\|\boldsymbol{\Gamma}x\|_1 \qquad (\ell_1\text{-analysis})$$

ADMM + Variable splitting (2/3)

Applying formula from slide 92:

$$F\begin{pmatrix} x \\ z \end{pmatrix} = \|y - \boldsymbol{H}x\|_2^2 + \tau\|z\|_1 \quad \longrightarrow \quad \mathrm{Prox}_{\gamma F}\begin{pmatrix} x \\ z \end{pmatrix} = \begin{pmatrix} (\mathrm{Id}_n + \gamma \boldsymbol{H}^*\boldsymbol{H})^{-1}(x + \gamma \boldsymbol{H}^* y) \\ \text{Soft-T}(z, \gamma\tau) \end{pmatrix}$$

$$G\begin{pmatrix} x \\ z \end{pmatrix} = \underbrace{\begin{cases} 0 & \text{if} \quad \boldsymbol{\Gamma}x = z \\ \infty & \text{otherwise} \end{cases}}_{\substack{\text{Indicator of the convex set} \\ \mathcal{C}=\{(x,z)\ ;\ \boldsymbol{\Gamma}x=z\}}} \quad \longrightarrow \quad \mathrm{Prox}_{\gamma G}\begin{pmatrix} x \\ z \end{pmatrix} = \underbrace{\begin{pmatrix} \mathrm{Id}_n \\ \boldsymbol{\Gamma} \end{pmatrix} (\mathrm{Id}_n + \boldsymbol{\Gamma}^*\boldsymbol{\Gamma})^{-1}(x + \boldsymbol{\Gamma}^* z)}_{\text{Projection on } \mathcal{C}}$$

Non-smooth optimization

$$\hat{x}^\star \in \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \; \frac{1}{2}\|y - \boldsymbol{H}x\|_2^2 + \tau\|\boldsymbol{\Gamma}x\|_1 \qquad (\ell_1\text{-analysis})$$

ADMM + Variable splitting (3/3)

$$\begin{aligned}
x^{k+1} &= (\mathrm{Id}_n + \gamma \boldsymbol{H}^*\boldsymbol{H})^{-1}(\tilde{x}^k + d_x^k + \gamma \boldsymbol{H}^* y)\\
z^{k+1} &= \mathsf{Soft\text{-}T}(\tilde{z}^k + d_z^k, \gamma\tau)\\
\tilde{x}^{k+1} &= (\mathrm{Id}_n + \boldsymbol{\Gamma}^*\boldsymbol{\Gamma})^{-1}(x^{k+1} - d_x^k + \boldsymbol{\Gamma}^*(z^{k+1} - d_z^k))\\
\tilde{z}^{k+1} &= \boldsymbol{\Gamma}\tilde{x}^{k+1}\\
d_x^{k+1} &= d_x^k - x^{k+1} + \tilde{x}^{k+1}\\
d_z^{k+1} &= d_z^k - z^{k+1} + \tilde{z}^{k+1}
\end{aligned}$$

If $\boldsymbol{H}$ is a blur, and $\boldsymbol{\Gamma}$ a filter bank, $(\mathrm{Id}_n + \gamma\boldsymbol{H}^*\boldsymbol{H})^{-1}$ and $(\mathrm{Id}_n + \boldsymbol{\Gamma}^*\boldsymbol{\Gamma})^{-1}$ can be computed in the Fourier domain in $O(n \log n)$.

Non-smooth optimization

$$\hat{x}^\star \in \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \; \frac{1}{2}\|y - \boldsymbol{H}x\|_2^2 + \tau\|\boldsymbol{\Gamma}x\|_1 \qquad (\ell_1\text{-analysis})$$

Application to Total-Variation $\boldsymbol{\Gamma} = \nabla$

$$x^{k+1} = (\mathrm{Id}_n + \gamma \boldsymbol{H}^* \boldsymbol{H})^{-1}(\tilde{x}^k + d_x^k + \gamma \boldsymbol{H}^* y)$$

$$z^{k+1} = \mathsf{Soft\text{-}T}(\tilde{z}^k + d_z^k, \gamma\tau)$$

$$\tilde{x}^{k+1} = (\mathrm{Id}_n + \nabla^*\nabla)^{-1}(x^{k+1} - d_x^k + \nabla^*(z^{k+1} - d_z^k))$$

$$\tilde{z}^{k+1} = \nabla\tilde{x}^{k+1}$$

$$d_x^{k+1} = d_x^k - x^{k+1} + \tilde{x}^{k+1}$$

$$d_z^{k+1} = d_z^k - z^{k+1} + \tilde{z}^{k+1}$$

$$\nabla^* = -\operatorname{div} \quad \text{and} \quad \nabla^*\nabla = -\Delta$$

$$\hat{x}^\star \in \operatorname*{argmin}_{x \in \mathbb{R}^n} \frac{1}{2}\|y - \boldsymbol{H}x\|_2^2 + \tau\|\boldsymbol{\Gamma}x\|_1 \qquad (\ell_1\text{-analysis})$$

Application to Total-Variation $\boldsymbol{\Gamma} = \nabla$

$$\begin{aligned}
x^{k+1} &= (\mathrm{Id}_n + \gamma \boldsymbol{H}^*\boldsymbol{H})^{-1}(\tilde{x}^k + d_x^k + \gamma \boldsymbol{H}^* y)\\
z^{k+1} &= \mathsf{Soft\text{-}T}(\tilde{z}^k + d_z^k, \gamma\tau)\\
\tilde{x}^{k+1} &= (\mathrm{Id}_n - \Delta)^{-1}(x^{k+1} - d_x^k - \mathrm{div}(z^{k+1} - d_z^k))\\
\tilde{z}^{k+1} &= \nabla \tilde{x}^{k+1}\\
d_x^{k+1} &= d_x^k - x^{k+1} + \tilde{x}^{k+1}\\
d_z^{k+1} &= d_z^k - z^{k+1} + \tilde{z}^{k+1}
\end{aligned}$$

$$\nabla^* = -\mathrm{div} \quad \text{and} \quad \nabla^*\nabla = -\Delta$$

$$\hat{x}^\star \in \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \ \frac{1}{2}\|y - \boldsymbol{H}x\|_2^2 + \tau\|\boldsymbol{\Gamma}x\|_1 \qquad (\ell_1\text{-analysis})$$

Application to sparse analysis with UDWT $\boldsymbol{\Gamma} = \boldsymbol{\Lambda}^{-1/2}\bar{\boldsymbol{W}}$

$$\begin{aligned}
x^{k+1} &= (\mathrm{Id}_n + \gamma \boldsymbol{H}^*\boldsymbol{H})^{-1}(\tilde{x}^k + d_x^k + \gamma \boldsymbol{H}^* y) \\
z^{k+1} &= \text{Soft-T}(\tilde{z}^k + d_z^k, \frac{\gamma\tau}{\lambda_i}) \\
\tilde{x}^{k+1} &= (\mathrm{Id}_n + \bar{\boldsymbol{W}}^*\bar{\boldsymbol{W}})^{-1}(x^{k+1} - d_x^k + \bar{\boldsymbol{W}}^*(z^{k+1} - d_z^k)) \\
\tilde{z}^{k+1} &= \bar{\boldsymbol{W}}\tilde{x}^{k+1} \\
d_x^{k+1} &= d_x^k - x^{k+1} + \tilde{x}^{k+1} \\
d_z^{k+1} &= d_z^k - z^{k+1} + \tilde{z}^{k+1}
\end{aligned}$$

Tight-frame: $\bar{\boldsymbol{W}}^*\bar{\boldsymbol{W}} = \mathrm{Id}_n$

Non-smooth optimization

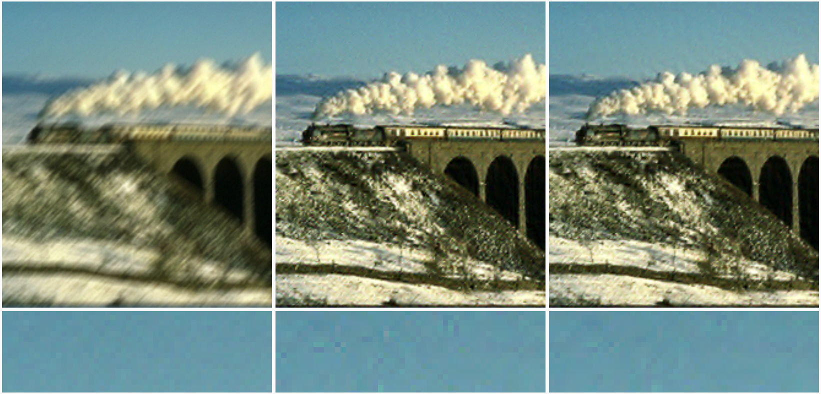
$$\hat{x}^\star \in \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \ \frac{1}{2}\|y - \boldsymbol{H}x\|_2^2 + \tau\|\boldsymbol{\Gamma}x\|_1 \qquad (\ell_1\text{-analysis})$$

Application to sparse analysis with UDWT $\quad \boldsymbol{\Gamma} = \boldsymbol{\Lambda}^{-1/2}\bar{\boldsymbol{W}}$

$$\begin{aligned}
x^{k+1} &= (\mathrm{Id}_n + \gamma \boldsymbol{H}^*\boldsymbol{H})^{-1}(\tilde{x}^k + d_x^k + \gamma \boldsymbol{H}^* y)\\
z^{k+1} &= \text{Soft-T}(\tilde{z}^k + d_z^k, \frac{\gamma\tau}{\lambda_i})\\
\tilde{x}^{k+1} &= \frac{1}{2}(x^{k+1} - d_x^k + \bar{\boldsymbol{W}}^*(z^{k+1} - d_z^k))\\
\tilde{z}^{k+1} &= \bar{\boldsymbol{W}}\tilde{x}^{k+1}\\
d_x^{k+1} &= d_x^k - x^{k+1} + \tilde{x}^{k+1}\\
d_z^{k+1} &= d_z^k - z^{k+1} + \tilde{z}^{k+1}
\end{aligned}$$

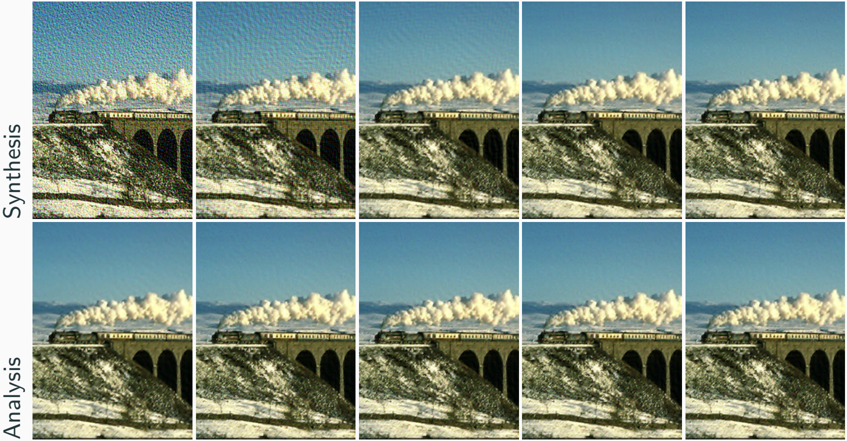
Tight-frame: $\quad \bar{\boldsymbol{W}}^*\bar{\boldsymbol{W}} = \mathrm{Id}_n$

Deconvolution with UDWT (5 levels, Db2)



(a) Blurry image y (noise $\sigma = 2$) (b) Synthesis (FISTA) (c) Analysis (FADMM)

Sparse analysis – Results



(a) 1 level (b) 2 level (c) 3 level (d) 4 level (e) 5 level

Analysis allows for less decomposition levels.

$\Rightarrow$ **leads to faster algorithms.**

Sparse analysis – Results

(a) Noisy ($\sigma = 40$)   (b) Analysis UDWT(4)   (c) +block (orien.+col.)   (d) Difference

- As for TV, **group coefficients** across orientations/color using $\ell_{2,1}$ norms:

$$\|\mathbf{\Gamma} z\|_{2,1}$$

- The soft-thresholding becomes the **group soft-thresholding**:

$$\left[\mathrm{Prox}_{\gamma\|\cdot\|_{2,1}}(z)\right]_i = \begin{cases} z_i - \gamma \frac{z_i}{\|z_i\|_2} & \text{if} \quad \|z_i\|_2 > \gamma \\ 0 & \text{otherwise} \end{cases}$$

Reminder from last class:

Modeling the distribution of images is complex (large degree of freedom).
Applying LMMSE on patches $\rightarrow$ increase performance

Next class:

What if we use sparse priors, not for the distribution of images, but for the distribution of patches?

For further reading

Sparsity, shrinkage and recovery guarantee:

- Donoho & Johnstone (1994); Moulin & Liu (1999); Donoho and Elad (2003); Gribonval and Nielsen (2003); Candès and Tao (2005); Zhang (2008); Candès and Romberg (2007).
- Book: Statistical Learning with Sparsity (Hastie, Tibshirani, Wainwright, 2015).

Wavelet related transforms:

- Warblet/Chirplet (Mann, Mihovilovic et al., 1991–1992), Curvelet (Candès & Donoho, 2000), Noiselet (Coifman, 2001), Contourlet (Do & Vetterli, 2002), Ridgelet (Do & Vetterli, 2003), Shearlets (Kanghui et al., 2005), Bandelet (Le Pennec, Peyré, Mallat, 2005), Empirical wavelets (Gilles, 2013).
- Book: A wavelet tour of signal processing (Mallat, 2008)

Non-smooth convex optimization:

- Douglas-Rachford splitting (Combettes & Pesquet, 2007), Split Bregman (Goldstein & Osher, 2009), Primal-Dual (Chambolle & Pock, 2011), Generalized FB (Raguet et al., 2013), Condat algorithm (2014).
- Book: Convex Optimization (Boyd, 2004).

Questions?

Next class: Patch models and dictionary learning

Sources, images courtesy and acknowledgment

L. Condat

A. Horodniceanu

G. Peyré

J. Salmon

Wikipedia