Linear Inverse Problems (2/2)

Mathematical Foundations of Signal Processing

Dr. Matthieu Simeoni

November 16, 2020

EPFL

Table of contents

Functional Linear Inverse Problems

Functional Inverse Problems

In the previous lecture, we have constrained the signal f to be finite-dimensional:

$$f = \sum_{n=1}^{N} \alpha_n \psi_n = \Psi \boldsymbol{\alpha}, \qquad \boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_N] \in \mathbb{R}^N,$$

for some suitable basis functions $\{\psi_n,\, n=1,\dots,N\} \subset \mathscr{L}^2(\mathbb{R}^d)$.[1] The idea was to reduce the number of degrees of freedom of the signal f to something manageable given the finite-dimensional data. While sensible, it is unclear if this discretisation step can be done canonically:

- *How should we choose N?*
- *How should we choose the parametrising basis functions $\{\psi_n,\, n=1,\dots,N\}$?* (pixels, sines/cosines, radial basis functions, polynomials, splines...)

To answer these questions, we relax the finite-dimensional assumption and formulate the reconstruction problem directly in the continuous domain. We then characterise the form of the solutions and deduce canonical discretisation schemes.

[1] Typically chosen as indicator functions of regular rectangular tiles of $\mathbb{R}^d$ called pixels.

Functional Tikhonov Regularisation

Consider the following functional penalised Tikhonov problem:[2]

$$\min_{f\in\mathscr{H}^k} F\left(y,\Phi^* f\right) \quad + \quad \lambda \left\|D^k f\right\|_2^2, \tag{1}$$

where $k \geq 0$, $\lambda > 0$ and:

- D^k denotes the k-th derivate operator on $\mathbb{R}$.
- $\mathscr{H}^k := \left\{f \in \mathscr{L}^2(\mathbb{R}) : \left\|D^k f\right\|_2 < +\infty\right\}$ denotes the Hilbert space of functions with square-integrable k-th derivatives called Sobolev space.
- $\Phi^* : \mathscr{H}^k \to \mathbb{R}^L$ is the sampling operator associated with a linearly independent family of sampling functionals $\{\varphi_1, \ldots, \varphi_L\} \subset \mathscr{H}^k$ and such that $\mathscr{N}(\Phi^*) \cap \mathscr{N}(D^k) = \{0\}$.
- $F : \mathbb{R}^L \times \mathbb{R}^L \to \mathbb{R}_+ \cup \{+\infty\}$ is a cost functional assumed proper convex, coercive and lwsc w.r.t. its second argument.

[2]Note that the unknown signal $f : \mathbb{R} \to \mathbb{R}$ in (1) is a function and not a discrete vector anymore.

Functional Representer Theorem

Representer Theorem: (Functional Tikhonov) [1, Theorem 3]

Under the assumptions listed on Slide 5, the solution set of (1) is non empty, convex, compact. Moreover, any solution $f^\star \in \mathcal{V}$ can be written as:

$$f^\star(x) = \sum_{i=1}^{L} \alpha_i (\rho_k * \varphi_i)(x) + \sum_{j=0}^{k-1} \beta_j x^j, \qquad \forall x \in \mathbb{R}, \tag{2}$$

for some coefficients $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_L] \in \mathbb{R}^L$, $\boldsymbol{\beta} = [\beta_0, \ldots, \beta_{k-1}] \in \mathbb{R}^k$ such that

$$\sum_{i=1}^{L} \alpha_i \left\langle x^j, \varphi_i \right\rangle = \sum_{i=1}^{L} \alpha_i \int_{\mathbb{R}} \varphi_i(x) x^j \, dx = 0, \qquad \forall j = 0, \ldots, k-1. \tag{3}$$

and where

$$\rho_k(x) = \mathscr{F}^{-1}\left\{|\omega|^{-2k}\right\}(x) = \frac{|x|^{2k-1}}{2(-1)^k (2k-1)!}, \qquad x \in \mathbb{R}. \tag{4}$$

Moreover if $F(\mathbf{y}, \cdot)$ is strictly convex, then the solution is unique.

Canonical Discretisation

We can re-write (2) as

$$f^\star = \Psi\boldsymbol{\alpha} + \Lambda\boldsymbol{\beta},$$

where $\Psi : \mathbb{R}^L \to \mathscr{H}^k$ and $\Lambda : \mathbb{R}^k \to \mathscr{H}^k$ are the synthesis operators associated to the family of functions $\{\rho_k * \varphi_i,\ i = 1,\dots,L\}$ and $\{x^j,\ j = 0,\dots,k-1\}$ respectively. We have then:

$$\Phi^* f^\star = \Phi^*\Psi\boldsymbol{\alpha} + \Phi^*\Lambda\boldsymbol{\beta} = \boldsymbol{G}\boldsymbol{\alpha} + \boldsymbol{H}\boldsymbol{\beta}, \tag{5}$$

where $\boldsymbol{G} = \Phi^*\Psi \in \mathbb{R}^{L\times L}$ and $\boldsymbol{H} = \Phi^*\Lambda \in \mathbb{R}^{L\times k}$ are real matrices with entries given by:

$$G_{ij} := \langle \rho_k * \varphi_j, \varphi_i \rangle,\ i,j = 1,\dots,L, \quad \text{and} \quad H_{in} := \langle x^n, \varphi_i \rangle\ i = 1,\dots,L,\ n = 0,\dots k-1.$$

We have moreover:

$$\|D^k f^\star\|_2^2 = \left\langle D^k f^\star, D^k f^\star \right\rangle = \left\langle D^k\Psi\boldsymbol{\alpha} + \underbrace{D^k\Lambda\boldsymbol{\beta}}_{=0}, D^k\Psi\boldsymbol{\alpha} + \underbrace{D^k\Lambda\boldsymbol{\beta}}_{=0} \right\rangle = \left\langle D^{k*} D^k\Psi\boldsymbol{\alpha}, \Psi\boldsymbol{\alpha} \right\rangle$$

since $\Lambda\boldsymbol{\beta} = \sum_{j=0}^{k-1} \beta_j x^j$ is a polynomial of degree $k-1$ and hence $D^k\Lambda\boldsymbol{\beta} = 0$.

Canonical Discretisation (continued)

$D f(x) \leftrightarrow \widehat{Df}(\omega) = j\omega\, \hat{f}(\omega)$

$\widehat{D^k f} = (j\omega)^k \hat{f}(\omega)$

Additionally, we have from (4) and the convolution/multiplication theorem that:

$$D^{k*} D^k \Psi \boldsymbol{\alpha} = \sum_{i=1}^{L} \alpha_i \mathscr{F}^{-1}\left\{-j\omega^k j\omega^k \hat{\rho}_k \hat{\varphi}_i\right\} = \sum_{i=1}^{L} \alpha_i \mathscr{F}^{-1}\left\{\frac{|\omega|^{2k}}{|\omega|^{2k}} \hat{\varphi}_i\right\} = \sum_{i=1}^{L} \alpha_i \varphi_i = \Phi \boldsymbol{\alpha},$$

which yields:

$$\|D^k f^\star\|_2^2 = \left\langle D^{k*} D^k \Psi \boldsymbol{\alpha}, \Psi \boldsymbol{\alpha} \right\rangle = \langle \Phi \boldsymbol{\alpha}, \Psi \boldsymbol{\alpha} \rangle = \left\langle \boldsymbol{\alpha}, \Phi^* \Psi \boldsymbol{\alpha} \right\rangle = \boldsymbol{\alpha}^T \boldsymbol{G} \boldsymbol{\alpha}. \tag{6}$$

Finally, note that condition (3) translates into:

$$\sum_{i=1}^{L} \alpha_i \left\langle x^j, \varphi_i \right\rangle = 0, \qquad \forall j = 0, \ldots, k-1, \quad \Leftrightarrow \quad \boldsymbol{\alpha}^T \boldsymbol{H} = \boldsymbol{0} \quad \Leftrightarrow \quad \boldsymbol{\alpha} \in \mathscr{R}(\boldsymbol{H})^\perp \subset \mathbb{R}^L. \tag{7}$$

Plugging (5), (6) and (7) into (1) yields:

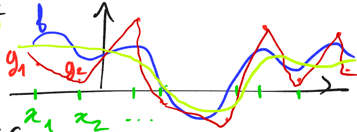
$$f^\star \in \underset{f \in \mathscr{H}^k}{\arg\min}\, F\left(\boldsymbol{y}, \Phi^* f\right) + \lambda \|D^k f\|_2^2 \quad \Leftrightarrow \quad (\boldsymbol{\alpha}, \boldsymbol{\beta}) \in \underset{\boldsymbol{\alpha} \in \mathscr{R}(\boldsymbol{H})^\perp, \boldsymbol{\beta} \in \mathbb{R}^k}{\arg\min} F(\boldsymbol{y}, \boldsymbol{G}\boldsymbol{\alpha} + \boldsymbol{H}\boldsymbol{\beta}) + \lambda \boldsymbol{\alpha}^T \boldsymbol{G} \boldsymbol{\alpha}.$$

We have hence shown that the functional penalised Tikhonov problem (1) can be discretised canonically. Moreover this discretisation is lossless: *the functional and discrete problems are both equivalent!*

Example: Ideal Sampling, $k=2$

$$\mathcal{H}^2 = \left\{ f : \mathbb{R} \to \mathbb{R},\ \int_{\mathbb{R}} |f''(x)|^2 dx < +\infty \right\}$$

$$\to \min_{f \in \mathcal{H}^2} \underbrace{\sum_{i=1}^{L} |y_i - f(x_i)|^2}_{\text{strict cvx, lwsc, proper}} + \lambda \underbrace{\|D^2 f\|_2^2}$$



$$f(x_i) = \int_{\mathbb{R}} f(x)\, \delta(x - x_i)\, dx = \langle f, \underbrace{\delta(\cdot - x_i)}_{a_i} \rangle$$

a_i linearly indep

$$\to f^* = \sum_{i=1}^{L} \alpha_i \underbrace{\rho_2 * \delta(\cdot - x_i)}_{\rho_2(x - x_i)} + \underbrace{\beta_0} + \underbrace{\beta_1} x$$

$$G_{ij} = \langle \rho_2(\cdot - x_i), \delta(\cdot - x_j) \rangle = \rho_2(x_j - x_i)$$

$$\rho_2(x) = \frac{|x|^3}{12}$$

Vandermond

$$\alpha^T H = 0 \iff \alpha^T \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_L \end{pmatrix} = 0 \qquad H_{nj} = \langle x^n, \delta(\cdot - x_j) \rangle = x_j^n$$

$\uparrow 2$

$L - 2 + 2 = L$

$\alpha \in \mathcal{R}(H)^{\perp} \leftarrow L - 2$ $\qquad \mathbb{R}^2$

Proximal Algorithms

Proximal Operator

Definition: (Proximal Operator)

Let $f : \mathbb{R}^N \to \mathbb{R} \cup \{+\infty\}$ be a proper convex and lwsc functional. Then, the proximal operator $\textbf{prox}_f : \mathbb{R}^N \to \mathbb{R}^N$ of f is defined as

$$\textbf{prox}_f(\boldsymbol{z}) := \underset{\boldsymbol{x}\in\mathbb{R}^N}{\arg\min}\, f(\boldsymbol{x}) + \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{z}\|_2^2, \quad \forall \boldsymbol{z} \in \mathbb{R}^N.$$

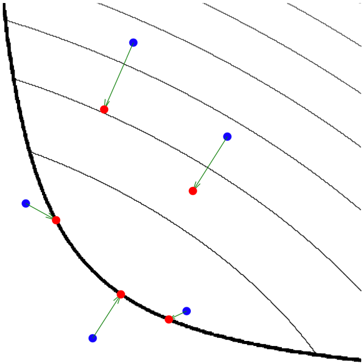
Since f proper convex and lwsc, it is easy to see that the objective functional defining the proximal operator is proper strictly convex, coercive, and lwsc, and hence $\textbf{prox}_f(\boldsymbol{z})$ exists and is unique for every $\boldsymbol{z} \in \mathbb{R}^N$. The proximal operator is hence well defined. We will often encounter the proximal operator of the scaled function τf, $\tau > 0$, which can be expressed as:

$$\textbf{prox}_{\tau f}(\boldsymbol{z}) := \underset{\boldsymbol{x}\in\mathbb{R}^N}{\arg\min}\, f(\boldsymbol{x}) + \frac{1}{2\tau}\|\boldsymbol{x}-\boldsymbol{z}\|_2^2, \quad \forall \boldsymbol{z} \in \mathbb{R}^N.$$

We say that a function is proximable if its proximal operator admits a simple closed-form expression.

Interpretation of Proximal Operator [2, Section 1.2]

- The thin black lines are level curves of f.
- The **bold** black line indicates the boundary of the domain of f.
- Evaluating $\mathbf{prox}_{\tau f}$ at the blue points moves them to the corresponding red points.
- The three points in the domain of the function stay in the domain and move towards the minimum of the function ($\simeq$ descent step).
- The two points outside of the domain move to the boundary of the domain and towards the minimum of the function ($\simeq$ projection step).
- The parameter τ controls the amount of displacement towards the minimum.



Properties of Proximal Operators

Proposition: (Properties of Proximal Operators)

1. **Separable Sum:** If $f : \mathbb{R}^{N_1} \times \cdots \times \mathbb{R}^{N_n} \to \mathbb{R} \cup \{+\infty\}$ is defined as: $f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \sum_{i=1}^n f_i(\boldsymbol{x}_i)$, $\forall (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^{N_1} \times \cdots \times \mathbb{R}^{N_n}$, then the proximal operator of f is given by:
$$\mathbf{prox}_{\tau f} = \left[\mathbf{prox}_{\tau f_1}(\boldsymbol{x}_1), \ldots, \mathbf{prox}_{\tau f_n}(\boldsymbol{x}_n)\right] \in \mathbb{R}^{N_1} \times \cdots \times \mathbb{R}^{N_n}.$$
2. **Precomposition:** If $f(\boldsymbol{x}) = g(\alpha \boldsymbol{x} + \boldsymbol{y})$, $\alpha > 0$, $\boldsymbol{y} \in \mathbb{R}^N$, then
$$\mathbf{prox}_{\tau f}(\boldsymbol{x}) = \frac{1}{\alpha}\left(\mathbf{prox}_{\tau \alpha^2 g}(\alpha \boldsymbol{x} + \boldsymbol{y}) - \boldsymbol{y}\right), \qquad \forall \boldsymbol{x} \in \mathbb{R}^N.$$
3. **Fixed Points & Minimisers:** $\boldsymbol{x}^\star \in \mathbb{R}^N$ minimises $f : \mathbb{R}^N \to \mathbb{R} \cup \{+\infty\}$ iff $\mathbf{prox}_f(\boldsymbol{x}^\star) = \boldsymbol{x}^\star$ [2, Section 2.3].

Additional useful results can be found in [2, Section 2].

Proof (Point 1)

$n = 2$

$$f(x,y) = f_1(\underset{\in \mathbb{R}^{N_1}}{x}) + f_2(\underbrace{y}_{\mathbb{R}^{N_2}})$$

$$\operatorname{prox}_{\tau f}(w,z) = \underset{(x,y)\in \mathbb{R}^{N_1}\times\mathbb{R}^{N_2}}{\operatorname{argmin}} f(x,y) + \frac{1}{2\tau}\|(x,y)-(w,z)\|_2^2$$

$$= \underset{(x,y)}{\operatorname{argmin}} f_1(x) + f_2(y) + \frac{1}{2\tau}\left[\|x-w\|_2^2 + \|y-z\|_2^2\right]$$

$$= \Big(\underbrace{\underset{x}{\operatorname{argmin}} f_1(x) + \frac{1}{2\tau}\|x-w\|_2^2}_{\operatorname{prox}_{\tau f_1}(w)},\ \underbrace{\underset{y}{\operatorname{argmin}} f_2(y) + \frac{1}{2\tau}\|y-z\|_2^2}_{\operatorname{prox}_{\tau f_2}(z)}\Big)$$

Proof (Point 2)

$$f(x) = g(\alpha x + y) \quad \forall$$

$$\mathrm{prox}_{\tau f}(z) = \underset{x \in \mathbb{R}^N}{\mathrm{argmin}}\ f(x) + \frac{1}{2\tau}\|x - z\|_2^2$$

$$u = \alpha x + y \iff \boxed{x = \frac{1}{\alpha}(u - y)}$$

$$= \underbrace{\underset{x \in \mathbb{R}^N}{\mathrm{argmin}}\ g(\alpha x + y) + \frac{1}{2\tau}\|x - z\|_2^2}$$

$$= \frac{1}{\alpha}\Big(\underbrace{\underset{u \in \mathbb{R}^N}{\mathrm{argmin}}\ g(u) + \frac{1}{2\tau}\left\|\frac{u}{\alpha} - \frac{y}{\alpha} - z\right\|_2^2}\ - y \Big)$$

$$\underbrace{\underset{u \in \mathbb{R}^N}{\mathrm{argmin}}\ g(u) + \frac{1}{2\tau\alpha^2}\|u - (\alpha z + y)\|_2^2}_{\frac{1}{\alpha}\left(\mathrm{prox}_{\tau\alpha^2 g}(\alpha z + y) - y\right)}$$

Examples of Simple Proximal Operators

Examples of Simple Proximal Operators ($y \in \mathbb{R}^N$):

- $f(\boldsymbol{x}) = \|\boldsymbol{x}-\boldsymbol{y}\|_2^2$:

$$\textbf{prox}_{\tau f}(\boldsymbol{x}) = \frac{\boldsymbol{x}-\boldsymbol{y}}{1+2\tau} + \boldsymbol{y}, \qquad \boldsymbol{x} \in \mathbb{R}^N.$$

- $f(\boldsymbol{x}) = \iota_C(\boldsymbol{x}-\boldsymbol{y})$ with $C \subset \mathbb{R}^N$ convex:

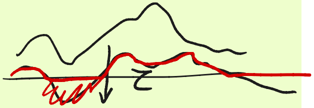
$$\textbf{prox}_{\tau f}(\boldsymbol{x}) = P_C(\boldsymbol{x}-\boldsymbol{y}) + \boldsymbol{y}, \qquad \boldsymbol{x} \in \mathbb{R}^N,$$

  where P_C is the projection operator onto the convex set C.

- $f(\boldsymbol{x}) = \|\boldsymbol{x}-\boldsymbol{y}\|_1$:

$$\textbf{prox}_{\tau f}(\boldsymbol{x}) = \textbf{soft}_{\tau \cancel{f}}(\boldsymbol{x}-\boldsymbol{y}) + \boldsymbol{y}, \qquad \boldsymbol{x} \in \mathbb{R}^N,$$

  where **soft**$_{\tau \cancel{f}}(\boldsymbol{x}) := \max\{|\boldsymbol{x}| - \tau, 0\}\,\mathrm{sgn}(\boldsymbol{x})$.



- $f(\boldsymbol{x}) = D_{KL}(\boldsymbol{y}||\boldsymbol{x})$:

$$\textbf{prox}_{\tau f}(\boldsymbol{x}) = \frac{\boldsymbol{x}-\tau+\sqrt{(\boldsymbol{x}-\tau)^2+4\boldsymbol{y}\tau}}{2}, \qquad \boldsymbol{x} \in \mathbb{R}^N.$$

Proximal Minimisation

Consider the following problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^N} \mathscr{G}(\boldsymbol{x}),$$

where $\mathscr{G}:\mathbb{R}^N \to \mathbb{R}$ is proper, lwsc and convex function~~s~~ with simple proximal operator. This optimisation problem can be solved by means of proximal minimisation:

Algorithm 1 Proximal Minimisation

procedure $\textsc{ProxMin}(\tau, \boldsymbol{x}_0)$
  for all $n \geq 1$ **do**
    $\boldsymbol{x}_n = \mathbf{prox}_{\tau\mathscr{G}}(\boldsymbol{x}_{n-1})$
  return $(\boldsymbol{x}_n)_{n\in\mathbb{N}}$

If $\mathscr{V} = \arg\min_{\boldsymbol{x}\in\mathbb{R}^N} \mathscr{G}(\boldsymbol{x})$ is non empty, the sequence $(\boldsymbol{x}_n)_{n\in\mathbb{N}}$ converges to an element of $\mathscr{V}$ for any $\tau > 0$ [2, Section 4.1].

Gradient Descent

Consider the following problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^N} \mathscr{F}(\boldsymbol{x}),$$

where $\mathscr{F}:\mathbb{R}^N \to \mathbb{R}$ is convex and differentiable, with β-Lipschitz continuous gradient:

$$\|\nabla\mathscr{F}(\boldsymbol{x}) - \nabla\mathscr{F}(\boldsymbol{x}')\|_2 \leq \beta\|\boldsymbol{x}-\boldsymbol{x}'\|_2, \qquad \forall(\boldsymbol{x},\boldsymbol{x}') \in \mathbb{R}^N \times \in \mathbb{R}^N, \tag{8}$$

for some Lipschitz constant $\beta \in [0,+\infty[$. This optimisation problem can be solved by means of gradient descent:

Algorithm 2 Gradient Descent

1: **procedure** GRADDESC($\tau, \boldsymbol{x}_0$)
2:     **for all** $n \geq 1$ **do**
3:         $\boldsymbol{x}_n = \boldsymbol{x}_{n-1} - \tau\nabla\mathscr{F}(\boldsymbol{x}_{n-1})$
4:     **return** $(\boldsymbol{x}_n)_{n\in\mathbb{N}}$

If $\mathscr{V} = \arg\min_{\boldsymbol{x}\in\mathbb{R}^N} \mathscr{F}(\boldsymbol{x})$ is non empty, the sequence $(\boldsymbol{x}_n)_{n\in\mathbb{N}}$ converges to an element of $\mathscr{V}$ for any $0 < \tau \leq \frac{1}{\beta}$ [3, Section 2].

Proximal Minimisation vs. Gradient Descent

Consider the least-squares minimisation problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^N} \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{G}\boldsymbol{x}\|_2^2,$$

with $\boldsymbol{G}\in\mathbb{R}^{L\times N}$, $\boldsymbol{y}\in\mathbb{R}^L$. We can minimise the functional $\mathscr{J}(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{G}\boldsymbol{x}\|_2^2$ in two ways:

- Via proximal minimisation since $\mathscr{J}$ is proper convex, and lwsc. This yields the following iterations:

$$\boldsymbol{x}_n = \textbf{prox}_{\tau\mathscr{J}}(\boldsymbol{x}_{n-1}) \Leftrightarrow \left(\tau\boldsymbol{G}^T\boldsymbol{G}+\boldsymbol{I}\right)\boldsymbol{x}_n = \boldsymbol{x}_{n-1} + \boldsymbol{G}^T\boldsymbol{y}, \qquad \boldsymbol{x}_0\in\mathbb{R}^N, n\geq 1.$$

  We must solve a linear system of size $N\times N$ at each iteration! Computationally expensive...

- Via gradient descent since $\mathscr{J}$ is differentiable and its gradient $\nabla\mathscr{J}(\boldsymbol{x}) = \boldsymbol{G}^T(\boldsymbol{G}\boldsymbol{x}-\boldsymbol{y})$ is moreover β-Lipschitz continuous with Lipschitz constant $\beta = \|\boldsymbol{G}^T\boldsymbol{G}\|_2$. This yields the following iterations:

$$\boldsymbol{x}_n = \boldsymbol{x}_{n-1} + \tau\boldsymbol{G}^T(\boldsymbol{y}-\boldsymbol{G}\boldsymbol{x}_{n-1}), \qquad \boldsymbol{x}_0\in\mathbb{R}^N, n\geq 1.$$

  The update equation only involves matrix/vector products with $\boldsymbol{G}$ and $\boldsymbol{G}^T$. Much cheaper!

Accelerated Proximal Gradient Descent

Consider the following problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^N} \mathscr{F}(\boldsymbol{x}) + \mathscr{G}(\boldsymbol{x}),$$

where $\mathscr{F}:\mathbb{R}^N \to \mathbb{R}$ is as in Slide 18 and $\mathscr{G}:\mathbb{R}^N \to \mathbb{R}\cup\{+\infty\}$ is as in Slide 17.This optimisation problem can be solved by means of Accelerated Proximal Gradient Descent (APGD):

Algorithm 3 Accelerated Proximal Gradient Descent (APGD)

1: **procedure** APGD$(\tau, \mathfrak{d}, \boldsymbol{x}_0 = \boldsymbol{z}_0)$
2:     **for all** $n \geq 1$ **do**
3:         $\boldsymbol{z}_n = \mathbf{prox}_{\tau\mathscr{G}}(\boldsymbol{x}_{n-1} - \tau\nabla\mathscr{F}(\boldsymbol{x}_{n-1}))$
4:         $\boldsymbol{x}_n = \boldsymbol{z}_n + \frac{n-1}{n+\mathfrak{d}}(\boldsymbol{z}_n - \boldsymbol{z}_{n-1})$
5:     **return** $(\boldsymbol{x}_n)_{n\in\mathbb{N}}$

The update equation at line 3 is the composition between a proximal step for $\mathscr{G}$ and a gradient step for $\mathscr{F}$. Line 4 is an acceleration step.

Convergence of APGD

For $\partial > 2$ and $0 < \tau \leq 1/\beta$, APGD achieves the following (optimal) convergence rates:

$$\lim_{n\to\infty} n^2 \left|\mathscr{J}(\boldsymbol{x}^\star) - \mathscr{J}(\boldsymbol{x}_n)\right| = 0 \qquad \& \qquad \lim_{n\to\infty} n^2 \|\boldsymbol{x}_n - \boldsymbol{x}_{n-1}\|_{\mathscr{X}}^2 = 0,$$

for some minimiser $\boldsymbol{x}^\star \in \arg\min_{\boldsymbol{x}\in\mathbb{R}^N} \left\{\mathscr{J}(\boldsymbol{x}) := \mathscr{F}(\boldsymbol{x}) + \mathscr{G}(\boldsymbol{x})\right\}$. [3]

In other words, both the objective functional and the APGD iterates $\{\boldsymbol{x}_n\}_{n\in\mathbb{N}}$ converge at a rate $o(1/n^2)$. Significant practical *speedup* can moreover be achieved for values of ∂ in the range $[50, 100]$ [4, 5].

[3] Assuming that the solution set is non empty.

Example: Fast Iterative Soft Thresholding Algorithm (FISTA)

Consider the LASSO problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^N} \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{G}\boldsymbol{x}\|_2^2 \quad + \quad \lambda\|\boldsymbol{x}\|_1,$$

with $\boldsymbol{G}\in\mathbb{R}^{L\times N}, \boldsymbol{y}\in\mathbb{R}^L, \lambda>0$. This problem can be solved via APGD with $\mathscr{F}(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{G}\boldsymbol{x}\|_2^2$ and $\mathscr{G}(\boldsymbol{x}) = \lambda\|\boldsymbol{x}\|_1$. We have:

$$\nabla\mathscr{F}(\boldsymbol{x}) = \boldsymbol{G}^T(\boldsymbol{G}\boldsymbol{x}-\boldsymbol{y}), \qquad \mathbf{prox}_{\lambda\|\cdot\|_1}(\boldsymbol{x}) = \mathbf{soft}_\lambda(\boldsymbol{x}).$$

This yields the so-called Fast Iterative Soft Thresholding Algorithm (FISTA) [6]:

Algorithm 4 Fast Iterative Soft Thresholding Algorithm (FISTA)

---

procedure FISTA$(\tau, \eth, \boldsymbol{x}_0 = \boldsymbol{z}_0)$
  for all $n \geq 1$ **do**
    $\boldsymbol{z}_n = \mathbf{soft}_{\tau\lambda}\left(\boldsymbol{x}_{n-1} + \tau\boldsymbol{G}^T(\boldsymbol{y}-\boldsymbol{G}\boldsymbol{x}_{n-1})\right)$
    $\boldsymbol{x}_n = \boldsymbol{z}_n + \frac{n-1}{n+\eth}(\boldsymbol{z}_n - \boldsymbol{z}_{n-1})$
  return $(\boldsymbol{x}_n)_{n\in\mathbb{N}}$

Convergence of FISTA is moreover guaranteed for $\eth > 2$ and $0 < \tau \leq \beta^{-1} = \|\boldsymbol{G}\|_2^{-2}$.

Lipschitzian, Proximable and Linear Composite Terms

Consider the following problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^N} \mathscr{F}(\boldsymbol{x}) \;+\; \mathscr{G}(\boldsymbol{x}) \;+\; \mathscr{H}(\boldsymbol{K}\boldsymbol{x}). \tag{9}$$

with the following assumptions:

1. $\mathscr{F}:\mathbb{R}^N \to \mathbb{R}$ is convex and differentiable, with β-Lipschitz continuous gradient:

$$\|\nabla\mathscr{F}(\boldsymbol{x}) - \nabla\mathscr{F}(\boldsymbol{x}')\|_{\mathscr{X}} \leq \beta\|\boldsymbol{x}-\boldsymbol{x}'\|_{\mathscr{X}}, \qquad \forall \boldsymbol{x},\boldsymbol{x}' \in \mathbb{R}^N,$$

   for some Lipschitz constant $\beta \in [0,+\infty[$.
2. $\mathscr{G}:\mathbb{R}^N \to \mathbb{R}\cup\{+\infty\}$ and $\mathscr{H}:\mathbb{R}^M \to \mathbb{R}\cup\{+\infty\}$ are two proper, lwsc and convex functions with simple proximal operators.
3. $\boldsymbol{K}:\mathbb{R}^N \to \mathbb{R}^M$ is a linear operator, with operator norm: $\|\boldsymbol{K}\|_2 = \sup_{\boldsymbol{x}\in\mathbb{R}^N, \|\boldsymbol{x}\|_2=1} \|\boldsymbol{K}\boldsymbol{x}\|_2$.
4. The problem (9) is feasible −i.e. there exists at least one solution.

Variable Splitting

Problem (9) cannot be solved via APGD: $\mathscr{G}$ and $\mathscr{H}$ have simple proximal operators, but the composite term $\mathscr{G}(\boldsymbol{x}) + \mathscr{H}(\boldsymbol{K}\boldsymbol{x})$ may not![4]

To circumvent this issue, we perform variable splitting by re-writing (9) in consensus form:

$$\min_{\boldsymbol{x}\in\mathbb{R}^N, \boldsymbol{w}\in\mathbb{R}^M} \mathscr{F}(\boldsymbol{x}) + \mathscr{G}(\boldsymbol{x}) + \mathscr{H}(\boldsymbol{w}), \quad \text{s.t.} \quad \boldsymbol{w} = \boldsymbol{K}\boldsymbol{x}. \tag{10}$$

The Lagrangian $\mathscr{L} : \mathbb{R}^N \times \mathbb{R}^M \times \mathbb{R}^M \to \mathbb{R} \cup \{+\infty\}$ associated to this problem is given by:

$$\mathscr{L}(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{z}) = \mathscr{F}(\boldsymbol{x}) + \mathscr{G}(\boldsymbol{x}) + \mathscr{H}(\boldsymbol{w}) + \boldsymbol{z}^T(\boldsymbol{K}\boldsymbol{x} - \boldsymbol{w}), \tag{11}$$

where the ancillary variable $\boldsymbol{z} \in \mathbb{R}^M$ is called a Lagrange multiplier. It is then possible to show that the saddle-point problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^N, \boldsymbol{w}\in\mathbb{R}^M} \max_{\boldsymbol{z}\in\mathbb{R}^M} \mathscr{L}(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{z})$$

is equivalent to (10). To this end, we introduce the notion of Fenchel conjugate of a function.

[4] For example the TV proximal problem $\arg\min_{\boldsymbol{x}\in\mathbb{R}^N} \|\boldsymbol{K}\boldsymbol{x}\|_1 + \frac{1}{2\tau}\|\boldsymbol{x} - \boldsymbol{z}\|_2^2$ does not admit a simple closed-form expression.

Fenchel Conjugate and Fenchel-Moreau Theorem

Definition: (Fenchel Conjugate/Biconjugate)

The Fenchel conjugate of a function $f : \mathbb{R}^N \to \mathbb{R} \cup \{-\infty, +\infty\}$ is defined as:

$$f^*(\boldsymbol{z}) := \sup_{\boldsymbol{x} \in \mathbb{R}^N} \langle \boldsymbol{z}, \boldsymbol{x} \rangle - f(\boldsymbol{x}), \qquad \forall \boldsymbol{z} \in \mathbb{R}^N.$$

The Fenchel biconjugate $f^{**} : \mathbb{R}^N \to \mathbb{R} \cup \{-\infty, +\infty\}$ is the Fenchel conjugate of the Fenchel conjugate:

$$f^{**}(\boldsymbol{x}) := \sup_{\boldsymbol{x} \in \mathbb{R}^N} \langle \boldsymbol{x}, \boldsymbol{z} \rangle - f^*(\boldsymbol{z}), \qquad \forall \boldsymbol{x} \in \mathbb{R}^N.$$

Theorem: (Fenchel-Moreau)

For f proper convex and lwsc we have $f = f^{**}$, i.e. f is equal to its Fenchel biconjugate.

Saddle-Point Problem is Equivalent to (9)

Using the Fenchel-Moreau theorem applied to $\mathscr{H}$ we get:

$$\begin{aligned}
\min_{\boldsymbol{x}\in\mathbb{R}^N,\boldsymbol{w}\in\mathbb{R}^M}\left(\max_{\boldsymbol{z}\in\mathbb{R}^M}\mathscr{L}(\boldsymbol{x},\boldsymbol{w},\boldsymbol{z})\right) &= \min_{\boldsymbol{x}\in\mathbb{R}^N}\max_{\boldsymbol{z}\in\mathbb{R}^M}\mathscr{F}(\boldsymbol{x}) + \mathscr{G}(\boldsymbol{x}) + \left(\min_{\boldsymbol{w}\in\mathbb{R}^M}\mathscr{H}(\boldsymbol{w}) - \boldsymbol{z}^T\boldsymbol{w}\right) + \boldsymbol{z}^T\boldsymbol{K}\boldsymbol{x}\\
&= \min_{\boldsymbol{x}\in\mathbb{R}^N}\max_{\boldsymbol{z}\in\mathbb{R}^M}\mathscr{F}(\boldsymbol{x}) + \mathscr{G}(\boldsymbol{x}) + \underbrace{\left(-\max_{\boldsymbol{w}\in\mathbb{R}^M}\boldsymbol{z}^T\boldsymbol{w} - \mathscr{H}(\boldsymbol{w})\right)}_{=-\mathscr{H}^*(\boldsymbol{z})} + \boldsymbol{z}^T\boldsymbol{K}\boldsymbol{x}\\
&= \min_{\boldsymbol{x}\in\mathbb{R}^N}\mathscr{F}(\boldsymbol{x}) + \mathscr{G}(\boldsymbol{x}) + \underbrace{\max_{\boldsymbol{z}\in\mathbb{R}^M}\boldsymbol{z}^T\boldsymbol{K}\boldsymbol{x} - \mathscr{H}^*(\boldsymbol{z})}_{=\mathscr{H}^{**}(\boldsymbol{K}\boldsymbol{x})}\\
&= \min_{\boldsymbol{x}\in\mathbb{R}^N}\mathscr{F}(\boldsymbol{x}) + \mathscr{G}(\boldsymbol{x}) + \mathscr{H}(\boldsymbol{K}\boldsymbol{x}).
\end{aligned}$$

We can hence solve (9) by solving the saddle-point problem (also called primal-dual problem):

$$\min_{\boldsymbol{x}\in\mathbb{R}^N,\boldsymbol{w}\in\mathbb{R}^M}\max_{\boldsymbol{z}\in\mathbb{R}^M}\mathscr{L}(\boldsymbol{x},\boldsymbol{w},\boldsymbol{z}) = \min_{\boldsymbol{x}\in\mathbb{R}^N}\max_{\boldsymbol{z}\in\mathbb{R}^M}\mathscr{F}(\boldsymbol{x}) + \mathscr{G}(\boldsymbol{x}) - \mathscr{H}^*(\boldsymbol{z}) + \boldsymbol{z}^T\boldsymbol{K}\boldsymbol{x}.$$

Primal-Dual Splitting Method

The primal-dual problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^N} \max_{\boldsymbol{z}\in\mathbb{R}^M} \mathscr{F}(\boldsymbol{x}) \; + \; \mathscr{G}(\boldsymbol{x}) \; - \mathscr{H}^*(\boldsymbol{z}) + \boldsymbol{z}^T \boldsymbol{K}\boldsymbol{x} \tag{12}$$

is much simpler to optimise:

1. $\mathscr{F}:\mathbb{R}^N \to \mathbb{R}$ is convex and differentiable, with β-Lipschitz continuous gradient.
2. $\boldsymbol{x} \mapsto \boldsymbol{z}^T\boldsymbol{K}\boldsymbol{x}$ and $\boldsymbol{z} \mapsto \boldsymbol{z}^T\boldsymbol{K}\boldsymbol{x}$ are convex and differentiable functionals, with 0-Lipschitz continuous gradients.
3. $\mathscr{G}:\mathbb{R}^N \to \mathbb{R}\cup\{+\infty\}$ and $\mathscr{H}^*:\mathbb{R}^M \to \mathbb{R}\cup\{+\infty\}$ are two proper, lwsc and convex functions with simple proximal operators. Indeed, the proximal operator of the Fenchel conjugate $\mathscr{H}^*$ is given by Moreau's identity:

$$\mathbf{prox}_{\sigma\mathscr{H}^*}(\boldsymbol{z}) = \boldsymbol{z} - \sigma\,\mathbf{prox}_{\mathscr{H}/\sigma}(\boldsymbol{z}/\sigma), \qquad \forall \boldsymbol{z}\in\mathbb{R}^M, \sigma > 0. \tag{13}$$

The primal-dual splitting method [7, Algorithm 3.1] can therefore be used to solve (12).

Primal-Dual Splitting Method

Algorithm 5 Primal-Dual Splitting (PDS) Method

procedure $\mathrm{PDS}(\tau, \sigma, \rho, \boldsymbol{x}_0, \boldsymbol{z}_0)$

  for all $n \geq 1$ **do**

    $\tilde{\boldsymbol{x}}_n = \mathbf{prox}_{\tau \mathscr{G}}\left(\boldsymbol{x}_{n-1} - \tau \nabla \mathscr{F}(\boldsymbol{x}_{n-1}) - \tau \boldsymbol{K}^* \boldsymbol{z}_{n-1}\right)$

    $\tilde{\boldsymbol{z}}_n = \mathbf{prox}_{\sigma \mathscr{H}^*}\left(\boldsymbol{z}_{n-1} + \sigma \boldsymbol{K}\left[2\tilde{\boldsymbol{x}}_n - \boldsymbol{x}_{n-1}\right]\right)$

    $\boldsymbol{x}_n = \rho \tilde{\boldsymbol{x}}_n + (1-\rho)\boldsymbol{x}_{n-1}$

    $\boldsymbol{z}_n = \rho \tilde{\boldsymbol{z}}_n + (1-\rho)\boldsymbol{z}_{n-1}$

  return $\{(\boldsymbol{x}_n, \boldsymbol{z}_n)\}_{n \in \mathbb{N}}$

Interpretation of PDS

The algorithm performs alternating proximal gradient/ascent steps:

- Given an estimate $\boldsymbol{z}_{n-1}$, Row 3 performs a proximal gradient descent with step size $\tau > 0$ to minimise

$$\min_{\boldsymbol{x}\in\mathbb{R}^N} \mathscr{F}(\boldsymbol{x}) \; + \; \mathscr{G}(\boldsymbol{x}) \; - \underbrace{\cancel{\mathscr{H}^*(\boldsymbol{z}_{n-1})}}_{=\,\text{cste}} + \boldsymbol{z}_{n-1}^T \boldsymbol{K}\boldsymbol{x}$$

  w.r.t. to the variable $\boldsymbol{x}$ (called primal variable).

- Row 4 uses the result of the proximal gradient descent step 3 and the previous primal estimate $\boldsymbol{x}_{n-1}$ and performs a proximal gradient ascent with step size $\sigma > 0$ to maximise

$$\max_{\boldsymbol{z}\in\mathbb{R}^M} \boldsymbol{z}^T \boldsymbol{K}(2\tilde{\boldsymbol{x}}_n - \boldsymbol{x}_{n-1}) - \mathscr{H}^*(\boldsymbol{z})$$

  w.r.t. to the variable $\boldsymbol{z}$ (called dual variable).

- $\rho > 0$ is a momentum term, used to combine the output of the gradient/ascent steps with previous estimates of the primal/dual variables.

Convergence of PDS ($\beta \neq 0$)

Theorem: (Convergence of PDS, $\beta \neq 0$) [7, Theorem 3.1]

Consider problem (12) under the assumptions of Slide 23 and let $\tau > 0, \sigma > 0$ and $\rho > 0$ be the *hyperparameters* of Algorithm 5. Suppose moreover that $\beta > 0$ and that the following holds:

1. $\frac{1}{\tau} - \sigma \|\boldsymbol{K}\|_{2}^2 \geq \frac{\beta}{2}$,
2. $\rho \in]0, \delta[$, where $\delta := 2 - \frac{\beta}{2}\left(\frac{1}{\tau} - \sigma\|\boldsymbol{K}\|_{2}^2\right)^{-1} \in [1, 2[$.

Then, there exists a pair $(\boldsymbol{x}^\star, \boldsymbol{z}^\star) \in \mathbb{R}^N \times \mathbb{R}^M s$ solution to (12), s.t. the primal and dual sequences of estimates $(\boldsymbol{x}_n)_{n\in\mathbb{N}}$ and $(\boldsymbol{z}_n)_{n\in\mathbb{N}}$ converge towards $\boldsymbol{x}^\star$ and $\boldsymbol{z}^\star$ respectively, i.e.

$$\lim_{n\to+\infty} \|\boldsymbol{x}^\star - \boldsymbol{x}_n\|_2 = 0, \quad \text{and} \quad \lim_{n\to+\infty} \|\boldsymbol{z}^\star - \boldsymbol{z}_n\|_2 = 0.$$

30

Convergence of PDS ($\beta = 0$)

Theorem: (Convergence of PDS, $\beta = 0$) [7, Theorem 3.1]

Consider problem (12) under the assumptions of Slide 23 and let $\tau > 0, \sigma > 0$ and $\rho > 0$ be the *hyperparameters* of Algorithm 5. Suppose moreover that $\beta = 0$ and that the following holds:

1. $\tau\sigma\|\boldsymbol{K}\|^2_{2\,\cancel{\mathcal{X},\mathcal{Y}}} \leq 1$,
2. $\rho \in [\epsilon, 2-\epsilon]$, for some $\epsilon > 0$.

Then, there exists a pair $(\boldsymbol{x}^\star, \boldsymbol{z}^\star) \in \mathbb{R}^N \times \mathbb{R}^M s$ solution to (12), s.t. the primal and dual sequences of estimates $(\boldsymbol{x}_n)_{n\in\mathbb{N}}$ and $(\boldsymbol{z}_n)_{n\in\mathbb{N}}$ converge towards $\boldsymbol{x}^\star$ and $\boldsymbol{z}^\star$ respectively, i.e.

$$\lim_{n\to+\infty} \|\boldsymbol{x}^\star - \boldsymbol{x}_n\|_2 = 0, \quad \text{and} \quad \lim_{n\to+\infty} \|\boldsymbol{z}^\star - \boldsymbol{z}_n\|_2 = 0.$$

Choosing the Step Sizes

In practice, the convergence speed of Algorithm 5 is improved by choosing σ and τ as large as possible and relatively well-balanced −so that both the primal and dual variables converge at the same pace. In practice, it is hence recommended to choose perfectly balanced parameters $\sigma = \tau$ saturating the inequalities 1 and 1. For $\beta > 0$ this yields:

$$\frac{1}{\tau} - \tau \|\boldsymbol{K}\|_2^2 = \frac{\beta}{2} \quad \Longleftrightarrow \quad -2\tau^2 \|\boldsymbol{K}\|_2^2 - \beta\tau + 2 = 0,$$

which admits one positive root

$$\tau = \sigma = \frac{1}{\|\boldsymbol{K}\|_2^2} \left(-\frac{\beta}{4} + \sqrt{\frac{\beta^2}{16} + \|\boldsymbol{K}\|_2^2} \right). \tag{14}$$

For $\beta = 0$, this yields

$$\tau = \sigma = \|\boldsymbol{K}\|_2^{-1}. \tag{15}$$

Computing the Lipschitz Constant β

Sometimes, computing the Lipschitz constant β of $\mathscr{F}$ can be difficult. In which case, it can be beneficial to overestimate it slightly using properties of sums/compositions of Lipschitz continuous functions:

- Let $\mathscr{F} = \mathscr{F}_1 \circ \mathscr{F}_2$ where $\mathscr{F}_1, \mathscr{F}_2$ are Lipschitz continuous functions with Lipschitz constants γ_1, γ_2 respectively. Then $\mathscr{F}$ is Lipschitz continuous with Lipschitz constant $\beta \leq \gamma_1 \gamma_2$.
- Let $\mathscr{F} = \mathscr{F}_1 + \mathscr{F}_2$ where $\mathscr{F}_1, \mathscr{F}_2$ are Lipschitz continuous functions with Lipschitz constants γ_1, γ_2 respectively. Then $\mathscr{F}$ is Lipschitz continuous with Lipschitz constant $\beta \leq \gamma_1 + \gamma_2$.

Example:

Assume that $\mathscr{F}(\boldsymbol{x}) = \mathscr{E}(\boldsymbol{G}\boldsymbol{x}) + \lambda \|\boldsymbol{D}\boldsymbol{x}\|^2$ where $\mathscr{E}$ is differentiable with γ-Lipschitz continuous gradient (γ known). Then,

$$\nabla \mathscr{F}(\boldsymbol{x}) = \boldsymbol{G}^T \nabla \mathscr{E}(\boldsymbol{G}\boldsymbol{x}) + 2\lambda \boldsymbol{D}^T \boldsymbol{D}\boldsymbol{x}, \qquad \forall \boldsymbol{x} \in \mathbb{R}^N.$$

We have moreover

$$\|\nabla \mathscr{F}(\boldsymbol{x}) - \nabla \mathscr{F}(\boldsymbol{x}')\| \leq \left(\gamma \|\boldsymbol{G}\|^2 + 2\lambda \|\boldsymbol{D}\|^2\right) \|\boldsymbol{x} - \boldsymbol{x}'\|, \quad \forall (\boldsymbol{x}, \boldsymbol{x}') \in \mathbb{R}^N \times \mathbb{R}^N,$$

and hence $\nabla \mathscr{F}$ is β-Lipschitz continuous, with $\beta \leq \gamma \|\boldsymbol{G}\|^2 + 2\lambda \|\boldsymbol{D}\|^2$.

Computing Operator Norms

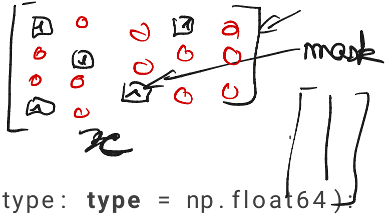
Computing the operator norm $\|\boldsymbol{K}\|_2$ of the linear operator $\boldsymbol{K}:\mathbb{R}^N \to \mathbb{R}^M$ amounts to finding its largest singular value. Performing this computation via a full SVD is wasteful and expensive: the full spectrum is computed when only the leading singular value is needed.

Instead, it is recommended to use the routine `scipy.linalg.svds()`[5] which is capable of computing only the leading (or more generally k leading) singular values.

This routine is moreover matrix-free: the operator $\boldsymbol{K}$ needs not be stored as an array, but can be an instance of the abstract class `scipy.sparse.linalg.LinearOperator` with methods `matvec()` and `rmatvec()` for matrix/vector products $\boldsymbol{K}\boldsymbol{x}$ and $\boldsymbol{K}^T\boldsymbol{x}$ respectively. This is particularly useful when N and M are very large (e.g. in computational imaging) and $\boldsymbol{K}$ cannot be stored in memory as a Numpy array.

[5] Or its companion routines `scipy.linalg.eigs()`, `scipy.linalg.eigsh()` for square/Hermitian matrices respectively

Example of a Matrix-Free Linear Operator




```
from scipy.sparse.linalg import LinearOperator

class DownSampling(LinearOperator):
    def __init__(self, size: int, mask: np.ndarray, dtype: type = np.float64):
        self.mask = np.asarray(mask).reshape(-1).astype(bool)
        self.in_size = size
        self.out_size = self.mask[self.mask == True].size
        super(Masking, self).__init__(shape=(self.out_size, self.in_size),\
                                      dtype=dtype)

    def matvec(self, x: np.ndarray) -> np.ndarray:
        return x[self.mask]

    def rmatvec(self, y: np.ndarray) -> np.ndarray:
        x = np.zeros(shape=self.in_size, dtype=self.dtype)
        x[self.mask] = y
        return x
```

Example 1: TV-Penalised Basis Pursuit

$$\min_{\boldsymbol{x}\in\mathbb{R}^N} \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{G}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{D}\boldsymbol{x}\|_1$$

$\mathcal{F}(x) = \frac{1}{2}\|y - Gx\|_2^2$, $\nabla\mathcal{F}(x) = G^T(Gx - y)$ $\quad \beta = \|G^TG\|_2 = \|G\|_2^2$

$\mathcal{G}(x) = 0$; $\mathcal{H}(x) = \lambda\|x\|_1$ $\quad K = D \in \mathbb{R}^{2N\times N}$

$\text{prox}_{\tau\mathcal{H}}(x) = \text{soft}_\tau(x) \leftarrow$

$\text{prox}_{\sigma\mathcal{H}^*}(x) = x - \sigma\, \text{prox}_{\mathcal{H}/\sigma}(x) \leftarrow$

$$\begin{cases} (x_0, z_0) \\ \tilde{x}_n = \boxed{x_{n-1} - \tau G^T(Gx_{n-1} - y) - \tau D^T z_{n-1}} \\ \tilde{z}_n = \text{prox}_{\sigma\mathcal{H}^*}\left(z_{n-1} + D(2\tilde{x}_n - x_{n-1})\right) \\ x_n = \rho\tilde{x}_n + (1-\rho)x_n \\ z_n = \rho\tilde{z}_n + (1-\rho)z_n \end{cases}$$

$\beta \in]0,1[$ $\quad \rho = 0,4$

Example 1: TV-Penalised Basis Pursuit

$$\beta = \|G\|_2^2$$

$$\tau = \sigma = \frac{1}{\|D\|_2^2}\left(-\frac{\|G\|_2^2}{4} + \sqrt{\frac{\|G\|_2^4}{16} + \|D\|_2^2}\right)$$

$$\mathcal{F}(x) = 0$$

$$\mathcal{G}_y(x) = \frac{1}{2}\|y - Gx\|_2^2 \rightarrow \mathrm{prox}_{\tau\mathcal{G}_y}(z) = \underset{x}{\mathrm{argmin}}\, \frac{1}{2}\|y - Gx\|_2^2 + \frac{1}{2\tau}\|z - x\|_2^2$$

$$\frac{\partial}{\partial x} \rightarrow G^TGx - G^Ty + \frac{1}{\tau}x - \frac{1}{\tau}z = 0$$

$$x = \left(G^TG + \frac{1}{\tau}I\right)^{-1}\left(G^Ty + \frac{1}{\tau}z\right)$$

PDS:

$$\begin{cases} \tilde{x}_n = \mathrm{prox}_{\tau G}\left(x_{n-1} - \tau D^T z_{n-1}\right) \\ \quad = \left(G^TG + \frac{1}{\tau}I\right)^{-1}\left(G^Ty + \frac{1}{\tau}\left(x_{n-1} - \tau D^T z_{n-1}\right)\right) \leftarrow \text{more complex} \\ \tilde{z}_n = \mathrm{prox}_{\sigma \mathcal{H}^*}\left(z_n + \sigma D\left(2\tilde{x}_n - x_{n-1}\right)\right) \\ x_n = \rho \tilde{x}_n + (1-\rho)\, x_{n-1} \\ z_n = \rho \tilde{z}_n + (1-\rho)\, z_{n-1} \end{cases}$$

Example 2: Tikhonov-Penalised Least Absolute Deviation

$$\min_{\boldsymbol{x}\in\mathbb{R}^N} \|\boldsymbol{y}-\boldsymbol{G}\boldsymbol{x}\|_1 + \frac{\lambda}{2}\|\boldsymbol{D}\boldsymbol{x}\|_2^2$$

$D : \mathbb{R}^N \to \mathbb{R}^{2N}$ gradient (image)

$\mathcal{G}(x) = 0$

$\mathcal{F}(x) = \frac{\lambda}{2}\|Dx\|_2^2 \to \nabla\mathcal{F}(x) = \lambda D^T D x \to \beta = \lambda \|D\|_2^2$

$\mathcal{H}(z) = \|y - z\|_1 \quad \forall z \in \mathbb{R}^L, \quad K = G \in \mathbb{R}^{L\times N}$

$\mathrm{prox}_{\tau\mathcal{H}}(z) = \mathrm{soft}_\tau(z-y) + y, \quad \mathrm{prox}_{\sigma\mathcal{H}^*}(\ ) =$ Moreau's identity

PDS
$$\left\{\begin{array}{l} (x_0, z_0) \in \mathbb{R}^N \times \mathbb{R}^L \\ \tilde{x}_n = x_{n-1} - \tau\lambda D^T D x - \tau G^T z_{n-1} \\ \tilde{z}_n = \mathrm{prox}_{\sigma\mathcal{H}^*}\left(z_{n-1} + \sigma G(2\tilde{x}_n - x_{n-1})\right) \\ \vdots \quad \rho \quad (1+\rho) \end{array}\right\}$$

Example 2: Tikhonov-Penalised Least Absolute Deviation

$$\beta = \lambda \|D\|_2^2$$

$$\sigma = \tau = \frac{1}{\|K\|_2^2}\left(-\frac{\beta}{4} + \sqrt{\frac{\beta^2}{16} + \|K\|_2^2}\right)$$

$$= \frac{1}{\|G\|_2^2}\left(-\frac{\lambda\|D\|_2^2}{4} + \sqrt{\frac{\lambda^2\|D\|_2^4}{16} + \|G\|_2^2}\right)$$

$$\rho \in \,]0,1[\;\rightarrow\; \rho = 0{,}4$$

$$\delta = 1$$

Example 3: KL-Divergence + TV

$D : \mathbb{R}^N \to \mathbb{R}^{2N}$

$$\min_{\boldsymbol{x}\in\mathbb{R}^N} D_{KL}(\boldsymbol{y}||\boldsymbol{G}\boldsymbol{x}) + \lambda\|\boldsymbol{D}\boldsymbol{x}\|_1$$

$\to \mathcal{X} = 0$

$\to \mathcal{Y} = 0$

$$\mathcal{H}(z,u) : \begin{cases} \mathbb{R}^L \times \mathbb{R}^{2N} \\ (z,u) \mapsto D_{KL}(y||z) + \lambda \|u\|_1 \end{cases}$$

$$K : \begin{cases} \mathbb{R}^N \longrightarrow \mathbb{R}^L \times \mathbb{R}^{2N} \\ x \longmapsto (Gx, Dx) \end{cases}$$

$$\mathcal{H}(Kx) = \mathcal{H}(Gx, Dx) = D_{KL}(y||Gx) + \lambda \|Dx\|_1$$

Example 3: KL-Divergence + TV

$$\mathcal{H}(z,u) = \underbrace{D_{KL}(y\|z)}_{\uparrow} + \underbrace{\lambda\|u\|_1}_{\uparrow}$$

$$\operatorname{prox}_{\tau\mathcal{H}}(z,u) = \left(\operatorname{prox}_{\tau \underbrace{D_{KL}}_{\mathcal{H}_1}}(z),\ \operatorname{prox}_{\underbrace{\lambda\|\cdot\|_1}_{\mathcal{H}_2}}(u)\right)$$

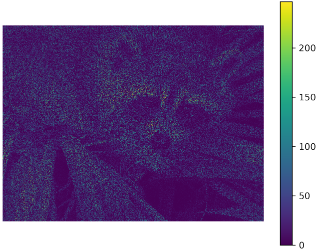
$$\operatorname{prox}_{\tau\mathcal{H}^*}$$

$$K^T = \begin{pmatrix} G^T \\ D^T \end{pmatrix}$$

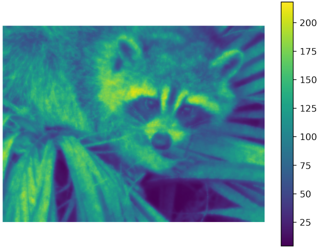
$$(z,u)$$

$$\text{PDS}\begin{cases} (z_0, u_0) \\ \tilde{x}_n = x_{n-1} - \underbrace{K^T(z_{n-1}, u_{n-1})}_{-G^T z_{n-1} - D^T u_{n-1}} = \\ \tilde{z}_{n-1} = \operatorname{prox}_{\sigma\mathcal{H}_1^*}\left(z_{n-1} + \sigma G(2\tilde{x}_n - x_{n-1})\right) \\ \tilde{u}_{n-1} = \operatorname{prox}_{\sigma\mathcal{H}_2^*}\left(u_{n-1} + \sigma D(2\tilde{x}_n - x_{n-1})\right) \end{cases}$$

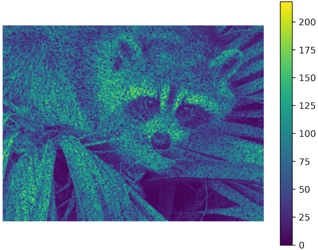
Effect of Regularisation Operator (Tikhonov)



(a) Data



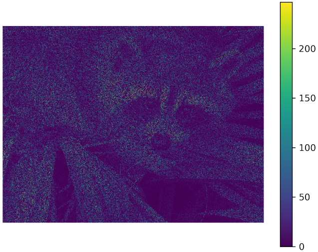
(b) $\boldsymbol{D} = \nabla$



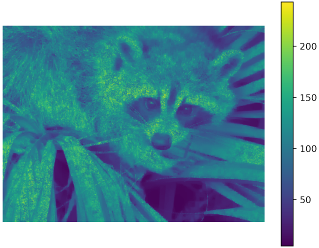
(c) $\boldsymbol{D} = \boldsymbol{\Delta}$

$$\min_{\boldsymbol{x}\in\mathbb{R}_+^N} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{G}\boldsymbol{x}\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{D}\boldsymbol{x}\|_2^2$$

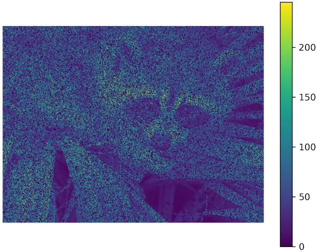
Effect of Regularisation Operator (TV)



(d) Data

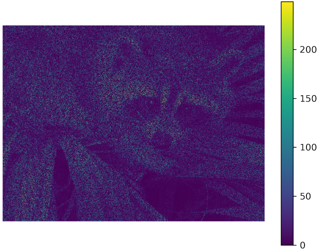


(e) $\boldsymbol{D} = \nabla$

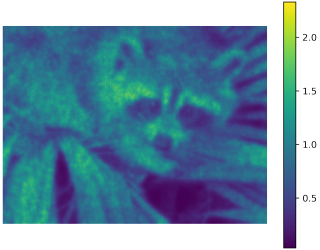


(f) $\boldsymbol{D} = \boldsymbol{\Delta}$

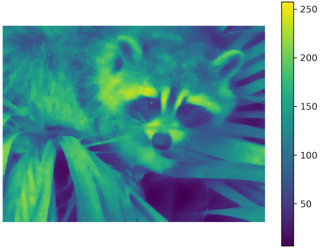
$$\min_{\boldsymbol{x} \in \mathbb{R}_+^N} \frac{1}{2} \|\boldsymbol{y} - \boldsymbol{G}\boldsymbol{x}\|_2^2 + \lambda \|\boldsymbol{D}\boldsymbol{x}\|_1$$



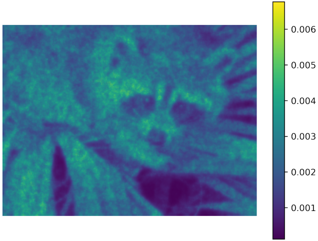
(g) Data



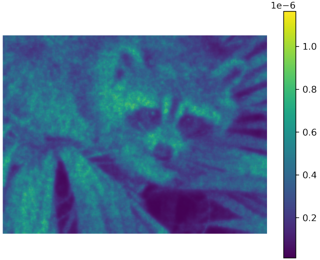
(h) Least-squares + gradient Tikhonov



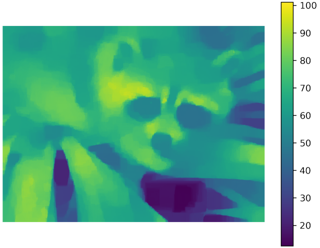
(i) Least-squares + gradient TV



(j) KL-divergence + gradient Tikhonov



(k) Least-deviations + gradient Tikhonov



(l) KL-divergence + gradient TV

References I

[1] Harshit Gupta, Julien Fageot, and Michael Unser.
Continuous-domain solutions of linear inverse problems with tikhonov versus generalized tv regularization.
IEEE Transactions on Signal Processing, 66(17):4670–4684, 2018.

[2] Neal Parikh and Stephen Boyd.
Proximal algorithms.
Foundations and Trends in optimization, 1(3):127–239, 2014.

[3] Brendan O'donoghue and Emmanuel Candes.
Adaptive restart for accelerated gradient schemes.
Foundations of computational mathematics, 15(3):715–732, 2015.

[4] Jingwei Liang, Jalal Fadili, and Gabriel Peyré.
Activity identification and local linear convergence of forward–backward-type methods.
SIAM Journal on Optimization, 27(1):408–437, 2017.

[5] Jingwei Liang and Carola-Bibiane Schönlieb.
Improving fista: Faster, smarter and greedier.
arXiv preprint arXiv:1811.01430, 2018.

References II

[6] Amir Beck and Marc Teboulle.
A fast iterative shrinkage-thresholding algorithm for linear inverse problems.
SIAM journal on imaging sciences, 2(1):183–202, 2009.

[7] Laurent Condat.
A primal–dual splitting method for convex optimization involving lipschitzian, proximable and linear composite terms.
Journal of Optimization Theory and Applications, 158(2):460–479, 2013.