# SaS Deconvolution Report

Paul Caucheteux

March 2024

## Introduction

The Short-and-Sparse (SaS) deconvolution problem is crucial for improving image processing and signal analysis, helping to make signals clearer and more defined. The goal is to recover, from an observed signal (e.g., an image), motifs of interest (the kernel of convolution) and their distribution over space (the signal of convolution). For this problem, we consider that the kernel is short and that the distribution is sparse.

To tackle this problem, we reformulate the problem as an optimization problem where we want to minimize the distance between the observed signal and the convolution while respecting the short and sparse hypothesis. Unfortunately, this problem happens to be non-convex, meaning that it has many local minima, therefore making it difficult to conceive an effective algorithm.

We will try to solve the SaS problem in 1D by introducing a few techniques, particularly effective in this context, to solve non-convex objectives.

## 1 Question 1

Let $a \in \mathbb{R}^d, x \in \mathbb{R}^n, \lambda \in \mathbb{R}^*$,
We can express $\lambda a$ and $\frac{x}{\lambda}$ as :

$$\lambda a = \begin{pmatrix} \lambda a_1 \\ \vdots \\ \lambda a_d \end{pmatrix} \qquad \text{and} \qquad \frac{x}{\lambda} = \begin{pmatrix} \frac{x_1}{\lambda} \\ \vdots \\ \frac{x_n}{\lambda} \end{pmatrix}$$

then for all $k \in \{1 \dots n\}$ we can express the k-th element of the convolution $(\lambda a) * (\frac{x}{\lambda})$ as :

$$((\lambda a) * (\frac{x}{\lambda}))_k = \sum_{s=0}^{d-1} \lambda a_s \frac{1}{\lambda} x_{k-s} = \sum_{s=0}^{d-1} a_s x_{k-s} = (a * x)_k$$

And so we have shown that $\forall a \in \mathbb{R}^d, \forall x \in \mathbb{R}^n, \forall \lambda \in \mathbb{R}^*$:

$$a * x = (\lambda a) * (\frac{x}{\lambda})$$

It means that there is a multiplication factor ambiguity for the solution of :

$$\begin{aligned} &\text{find } a \in \mathbb{R}^d, x \in \mathbb{R}^n, \\ &\text{such that } a * x = y, \\ &\text{and } \|x\|_0 \leqslant k \end{aligned} \tag{SaS}$$

If $(a^*, x^*)$ is a solution of SaS then $(\lambda a^*, \frac{x^*}{\lambda})$ is also a solution of SaS. Therefore we can consider only the solution with an $a$ belonging to the unit sphere, it will leave a sign ambiguity in the solution. We can reformulate the problem as :

$$\begin{aligned} &\text{find } a \in \mathbb{R}^d, x \in \mathbb{R}^n, \\ &\text{such that } a * x = y, \\ &\text{and } \|x\|_0 \leqslant k, a \in \mathbb{S}^{d-1} \end{aligned} \tag{SaS'}$$

## 2 Question 2

The first step is to reformulate the problem SaS' as an optimization problem :

$$\min_{a,x} \Psi_{BL}(a,x) = \underbrace{\frac{1}{2}\|y - a * x\|_2^2}_{\text{fitting term } \psi(x)} + \underbrace{\lambda\|x\|_1}_{\text{sparsity term } g(x)}, \quad \text{s.t. } a \in \mathbb{S}^{n_0-1} \tag{2.1}$$

This is a non smooth and a non convex objective but we can hope to solve it by alternating between optimizing on $a$ and on $x$. Naturally we use an Alternating Descent method (ADM) to try to solve this problem. We use the version proposed in [LQK+19] :

**Algorithm 1** Alternating Descent Method (ADM)

**Input:** Measurement $\boldsymbol{y} \in R^n$; stepsizes $t_0$ and $\tau_0$; penalty $\lambda > 0$.
**Output:** Final iterate $\boldsymbol{a}_\star$, $\boldsymbol{x}_\star$.
Initialize $\boldsymbol{a}^{(0)}$, $\boldsymbol{x}^{(0)} \leftarrow \mathbf{0}$, and $k \leftarrow 0$.
**while** not converged **do**:
Fix $\boldsymbol{a}^{(k)}$ and take a proximal gradient step on $\boldsymbol{x}$ with stepsize $t_k$

$$\boldsymbol{x}^{(k+1)} \leftarrow \operatorname{prox}_g^{\lambda t}\left(\boldsymbol{x}^{(k)} - t\nabla\psi_{\boldsymbol{a}^{(k)}}(\boldsymbol{x}^{(k)})\right),$$

Fix $\boldsymbol{x}^{(k+1)}$ and take a Riemannian gradient step on $\boldsymbol{a}$ with stepsize $\tau_k$

$$\boldsymbol{a}^{(k+1)} \leftarrow R_{\boldsymbol{a}^{(k)}}^{\mathbb{S}^{n_0-1}}\left(-\tau \cdot \psi_{\boldsymbol{x}^{(k+1)}}(\boldsymbol{a}^{(k)})\right),$$

Update $k \leftarrow k+1$.
**end while**

This algorithm alternates gradient steps on the two variables. The first optimization is made by fixing $a$ and taking a proximal gradient on $x$ which is common when minimizing an expression involving $\|.\|_1$. The second optimization is made by fixing x and taking a Riemannian gradient, we project on $\mathbb{S}^{n-1}$ to ensures that the iterates stay on the sphere.

As one can see, we optimize for $a \in R^{n_0}$, we can take $n_0 = d$ but also $n_0 > d$ (e.g $n_0 = 3d - 2$). Optimizing on a larger set permits to gain interesting properties and doesn't change the problem as the convolution is shift invariant. Indeed the objective function is locally convex around a single shift of the ground truth $a$ and demonstrates favorable characteristics around multiple shifts. Finally we can recover the real $a \in \mathbb{R}$ by some simple manipulation.

## 3 Question 3

Our implementation heavily rely on [LQK+19] and [Qu19]. We implemented the vanilla version of the ADM and introduced a few additional algorithms which are modifications of ADM, aimed at achieving improved performance :

*Inertial Alternating Descent Method (iADM)*: It operates on the same principle as ADM but introduces an inertial term, e.g for $x$:

$$\boldsymbol{w}^{(k)} = \boldsymbol{x}^{(k)} + \beta\underbrace{(\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)})}_{\text{inertial term}}$$

We then apply the rest of the algorithm to $\boldsymbol{w}$. This term acts similarly to momentum, adding acceleration to the descent process.

*Homotopy Continuation*: Larger $\lambda$ values enhance sparsity in x, whereas smaller $\lambda$ values improve reconstruction. However, the approach becomes challenging as $\lambda$ diminishes due to $a$'s poor spectral conditioning.

To address this we apply a homotopy continuation method leveraging either ADM or iADM and dynamically adjust $\lambda$.

*ADM and iADM with adaptive lambda*: It builds on the foundation of either ADM or iADM but with a key enhancement : an adaptive $\lambda$ which is a decreasing function of the iteration. Initially, a larger $\lambda$ prioritizes sparsity, then gradually shifts focus towards accuracy with a smaller $\lambda$ in later iterations. It permits to find a balance between sparsity and precision and so it refines the algorithm's performance.

The code of the the algorithms are available in the *solving_algorithm.py* section.

# 4 Question 4

## 4.1 a)

cf *Experiments.ipynb*

## 4.2 b)

cf *Experiments.ipynb*

# 5 Question 5

## 5.1 a)

cf *Experiments.ipynb*

## 5.2 b)

cf *Experiments.ipynb*

# References


[LQK+19] Yenson Lau, Qing Qu, Han-Wen Kuo, Pengcheng Zhou, Yuqian Zhang, and John Wright. Short-and-sparse deconvolution – a geometric approach, October 2019. arXiv:1908.10959v2 [eess.SP] 1 Oct 2019.

[Qu19] Qing Qu. Sparse deconvolution. https://github.com/qingqu06/sparse_deconvolution, 2019.