Practical work on image restoration
Constrained approach

Prerequisite: read the paper in advance, end-to-end, several times.

In the previous practical works you have solved an image deblurring problem. In order to recover a sharp image starting from the blurred (and noisy) version of it, you have worked on two regularized methods. Both relies on a penalized convex criterion itself based on a quadratic least-squares term (quantifying the discrepancy between the solution and the data) and a penalization term (accounting for a prior information regarding the regularity of the solution, up to a certain extend). The first one, called the Wiener-Hunt method, involve a quadratic penalty (in order to give advantage to smooth solutions rather than explosive ones) and the second method, involve a Huber penalty (in view of edge preserving properties and resolution enhancement). Nevertheless, when analysing the results, one can observe that the value of some pixels is negative whereas all pixels of the true image have a positive value. Moreover, unlike the true image, the value of pixels in the "black area" outside the brain is not zero. The purpose of this practical work is to develop a constrained approach which leads to a better physical modelling and an improvement of the resolution.

1 Quadratic penalty and constraints

We still use the following model to describe the acquisition process:

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{e}$$

where the vector $\boldsymbol{y}$ denotes the data (blurred image), the vector $\boldsymbol{x}$ represents the unknown image (sharp image), $\boldsymbol{H}$ is the convolution matrix and $\boldsymbol{e}$ is the vector accounting for measurement and modelling errors.

To regularise the deconvolution problem, we account for additional information pertaining to the spatial regularity of the unknown image by introducing a penalty of strong spatial variations of the gray levels. We define the following criterion:

$$\mathcal{J}(\boldsymbol{x}) = \|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}\|^2 + \mu \, \|\boldsymbol{D}\boldsymbol{x}\|^2$$

where $\boldsymbol{D}$ is a "difference" matrix, *e.g.,* a first order difference or any other linear "differential" operator (*e.g.,* Laplacian, Sobel, Prewitt,...). Additionally, we account for another prior information available about the unknown image:

1. the pixel values are positive, and
2. the pixel values are zero outside a given support $\mathcal{S}$.

Thus, the reconstructed image $\widehat{\boldsymbol{x}}$ is defined as:

$$\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in\mathbb{R}^P}{\arg\min} \begin{cases} \mathcal{J}(\boldsymbol{x}) \\ \text{s.t.} \begin{cases} x_p \geq 0 & \text{for all } p \\ x_p = 0 & \text{for } p \notin \mathcal{S} \end{cases} \end{cases} \tag{1}$$

that is to say an image that minimizes the criterion and meets the constraints. It is noteworthy that it is rigorously defined: the criterion is strictly convex and the constraints set is convex so there exist a unique solution.

2 Optimisation

Generally speaking, an optimisation problem consists of an objective function (a criterion) and possibly a set of constraints often expressed in the form of a system of equalities or inequalities. The aim is to find an optimal value, *i.e.,* that minimizes the objective function and meets the set of constraints, by using an algorithm. Various algorithms exist to solve such problems, see for example [1–3] and among the possibilities we can cite:

- Gradient projection and constrained gradient,
- Interior points and barrier,
- Pixel-wise descent,
- *Lagrange multipliers*.

They concern a huge range of applications in engineering, imaging (*e.g.,* astronomy, molecular,. . . and more broadly physics), computer science (computer graphics, computer vision, augmented reality,. . .), medicine and biology, economics and finance, statistics, inference and machine learning,. . .

2.1 The ADMM : an efficient algorithm for constrained problems

In this practical work we focus on an algorithm based on *Lagrange multipliers* and the idea of *criterion augmentation*. The algorithm itself called Alternating Direction Method of Multipliers (ADMM) is particularly efficient to solve large-scale convex constrained problems. The following text is taken from [3].

> *The method was developed in the 1970s, with roots in the 1950s, and is equivalent or closely related to many other algorithms, such as dual decomposition, the method of multipliers, Douglas-Rachford splitting, Spingarn's method of partial inverses, Dykstra's alternating projections, Bregman iterative algorithms for ℓ_1 problems, proximal methods, and others. [...] It takes the form of a decomposition-coordination procedure, in which the solutions to small local subproblems are coordinated to find a solution to a large global problem. ADMM can be viewed as an attempt to blend the benefits of dual decomposition and augmented Lagrangian methods for constrained optimization [...]*

It takes the form of decomposition-coordination procedure, in which the solutions to small local subproblems are coordinated to find solution to a large global problem. ADMM can be viewed as an attempt to blend the benefits of dual decomposition and augmented Lagrangian methods for constrained optimization, two earlier approaches that we review in §2.a It

It is possible to introduce ADMM in multiple ways[1]. Here, let us say that basically ADMM solves this kind of general convex problem with linear equality constraints:

$$(\mathcal{P}_\mathrm{p}) \qquad (\,\widehat{\boldsymbol{x}}\,,\widehat{\boldsymbol{z}}\,) = \underset{\boldsymbol{x},\boldsymbol{z}}{\arg\min} \begin{cases} \mathcal{F}(\boldsymbol{x}) + \mathcal{G}(\boldsymbol{z}) \\ \text{s.t. } \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} = \boldsymbol{c} \end{cases} \tag{2}$$

where $\mathcal{F}$ and $\mathcal{G}$ are convex functions, $\boldsymbol{A}$ and $\boldsymbol{B}$ are matrices and $\boldsymbol{c}$ is a vector with appropriate sizes. It is called the *primal problem* and $\boldsymbol{x}$, $\boldsymbol{z}$ are the *primal variables*.

First, we write the so called augmented Lagrangian function $\mathcal{L}$, associated to the primal problem (2):

$$\mathcal{L}(\boldsymbol{x},\boldsymbol{z},\boldsymbol{\lambda}) = \mathcal{F}(\boldsymbol{x}) + \mathcal{G}(\boldsymbol{z}) + \boldsymbol{\lambda}^\mathrm{t}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c}) + \frac{\rho}{2}\|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c}\|^2 \tag{3}$$

[1]The presentation here is slightly different than the one of the lecture.

where $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers and $\rho > 0$ a penalty parameter. Then we minimise $\mathcal{L}_\rho$ w.r.t. $(\boldsymbol{x}, \boldsymbol{z})$ for a given value of $\boldsymbol{\lambda}$ and it yields the minimiser

$$(\,\bar{\boldsymbol{x}}_{\boldsymbol{\lambda}}\,,\bar{\boldsymbol{z}}_{\boldsymbol{\lambda}}\,) = \underset{\boldsymbol{x},\boldsymbol{z}}{\arg\min}\, \mathcal{L}(\boldsymbol{x},\boldsymbol{z},\boldsymbol{\lambda})\,, \tag{4}$$

and the minimum:

$$\widetilde{\mathcal{L}}_\rho(\boldsymbol{\lambda}) = \min_{\boldsymbol{x},\boldsymbol{z}} \mathcal{L}(\boldsymbol{x},\boldsymbol{z},\boldsymbol{\lambda}) = \mathcal{L}(\bar{\boldsymbol{x}}_{\boldsymbol{\lambda}},\bar{\boldsymbol{z}}_{\boldsymbol{\lambda}},\boldsymbol{\lambda})\,, \tag{5}$$

that is a function of $\boldsymbol{\lambda}$, namely the *dual function*. The next point is then to solve the so called *dual problem*:

$$(\mathcal{P}_\mathrm{d}) \qquad \bar{\boldsymbol{\lambda}} = \underset{\boldsymbol{\lambda}}{\arg\max}\, \widetilde{\mathcal{L}}_\rho(\boldsymbol{\lambda}) \tag{6}$$

that yields the correct value of the Lagrange multipliers. The last step consists in substitution of (6) in (4):

$$(\,\widehat{\boldsymbol{x}}\,,\widehat{\boldsymbol{z}}\,) = (\,\bar{\boldsymbol{x}}_{\bar{\boldsymbol{\lambda}}}\,,\bar{\boldsymbol{z}}_{\bar{\boldsymbol{\lambda}}}\,) \tag{7}$$

that is the solution of the primal problem (2).

Practically, the algorithm acts numerically: it alternates minimisation w.r.t. the primal variables $\boldsymbol{x}, \boldsymbol{z}$ and the dual ones $\boldsymbol{\lambda}$ that is why it is referred to as a primal-dual scheme. The three key steps of the algorithm are the following.

$$\boxed{\begin{aligned}
&\text{(a)}\;\; \boldsymbol{x}^{k+1} = \arg\min\nolimits_{\boldsymbol{x}}\;\mathcal{L}(\boldsymbol{x}^k,\boldsymbol{z}^k,\boldsymbol{\lambda}^k)\\
&\text{(b)}\;\; \boldsymbol{z}^{k+1} = \arg\min\nolimits_{\boldsymbol{z}}\;\mathcal{L}(\boldsymbol{x}^{k+1},\boldsymbol{z}^k,\boldsymbol{\lambda}^k)\\
&\text{(c)}\;\; \boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \rho(\boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z}^{k+1} - \boldsymbol{c})
\end{aligned}}$$

Note that the $\boldsymbol{\lambda}$-update (c) is separable and direct and corresponds to a gradient step ascent of the dual function $\widetilde{\mathcal{L}}_\rho$ defined in (5).

2.2 Application to our quadratic criterion with linear (inequality and equality) constraints

We now consider again the original problem (1) including both inequality and equality constraints:

$$\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in\mathbb{R}^P}{\arg\min} \begin{cases} \mathcal{J}(\boldsymbol{x}) \\ \text{s.t.} \begin{cases} x_p \geq 0 & \text{for all } p \\ x_p = 0 & \text{for } \; p \;\notin\; \mathcal{S} \end{cases} \end{cases}$$

and we rewrite it in a strictly equivalent form including equality constraints only. It is possible by considering first the constraints subset of $\mathbb{R}^P$

$$\mathcal{C} = \{\boldsymbol{x} \in \mathbb{R}^P \text{such that : } x_p \geq 0 \text{ for all } p \text{ and } x_p = 0 \text{ for } \; p \;\notin\; \mathcal{S}\}\,,$$

then its indicator

$$\mathcal{I}_\mathcal{C}(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \boldsymbol{x} \in \mathcal{C} \\ +\infty & \text{if } \boldsymbol{x} \notin \mathcal{C} \end{cases}$$

so, the solution clearly reads:

$$\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in\mathbb{R}^P}{\arg\min}\, \mathcal{J}(\boldsymbol{x}) + \mathcal{I}_\mathcal{C}(\boldsymbol{x})\,,$$

without additional constraints. The original constraints is replaced by a penalty, more specifically an infinite penalty of object that do not meet the constraints.

At first sight it seems superfluous…but we will see it is not the case. Let us now introduce a vector of auxiliary variable $\boldsymbol{z} \in \mathbb{R}^P$ and write an equivalent problem including an additional equality constraint:

$$\widehat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in\mathbb{R}^P}{\arg\min} \begin{cases} \mathcal{J}(\boldsymbol{x}) + \mathcal{I}_{\mathcal{C}}(\boldsymbol{z}) \\ \text{s.t.} \quad \boldsymbol{x} = \boldsymbol{z} \end{cases} \tag{8}$$

It also seems superfluous…but we are now able to take advantage of the ADMM given in the previous section. Then, we can write the Augmented Lagrangian of (8):

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}) = \mathcal{J}(\boldsymbol{x}) + \mathcal{I}_{\mathcal{C}}(\boldsymbol{z}) + \boldsymbol{\lambda}^{\mathrm{t}}(\boldsymbol{x} - \boldsymbol{z}) + \frac{\rho}{2}\left\|\boldsymbol{x} - \boldsymbol{z}\right\|^2 , \tag{9}$$

and the ADMM steps are the following

```
• Initialize $\boldsymbol{z}^0 = \mathbf{0}$, $\boldsymbol{\lambda}^0 = \mathbf{0}$ and $k = 0$
• Set $\varepsilon$, for instance $\varepsilon = 10^{-5}$
• For $k = 1, 2, \ldots$ repeat
    (a) $\boldsymbol{x}^{k+1} = \arg\min_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}^k, \boldsymbol{z}^k, \boldsymbol{\lambda}^k)$
    (b) $\boldsymbol{z}^{k+1} = \arg\min_{\boldsymbol{z}} \mathcal{L}(\boldsymbol{x}^{k+1}, \boldsymbol{z}^k, \boldsymbol{\lambda}^k)$
    (c) $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \rho(\boldsymbol{x}^{k+1} - \boldsymbol{z}^{k+1})$
• Until $\left\|\boldsymbol{x}^k - \boldsymbol{x}^{k-1}\right\| < \varepsilon$
```

that are described in the next sections.

2.2.1 Update of the object $\boldsymbol{x}$

Let us consider the update of the object $\boldsymbol{x}$, that is step (a) above, given the current values of $\boldsymbol{z}$ and $\boldsymbol{\lambda}$. Considering the structure of the Lagrangian function (9), one can clearly leave $\boldsymbol{z}$ out, and minimise:

$$\mathcal{K}_{\mathrm{o}}(\boldsymbol{x}) = \mathcal{J}(\boldsymbol{x}) + \boldsymbol{\lambda}^{\mathrm{t}}(\boldsymbol{x} - \boldsymbol{z}) + \frac{\rho}{2}\left\|\boldsymbol{x} - \boldsymbol{z}\right\|^2 .$$

It is noteworthy that $\mathcal{K}_{\mathrm{o}}$ is *quadratic* with respect to $\boldsymbol{x}$ and the minimisation problem is *unconstrained*.

1. *Explicit the $\boldsymbol{x}$-update step (a). Explain how it can be efficiently computed (see your previous practical works).*

2.2.2 Update of the auxiliary variable $\boldsymbol{z}$

We now turn to the update of the auxiliary variable $\boldsymbol{z}$, referred to as step (b) above, the current values of $\boldsymbol{x}$ and $\boldsymbol{\lambda}$ being given. Considering once more the Lagrangian function (9), one can clearly leave $\boldsymbol{x}$ out, and minimise:

$$\mathcal{K}_{\mathrm{a}}(\boldsymbol{z}) = \mathcal{I}_{\mathcal{C}}(\boldsymbol{z}) + \boldsymbol{\lambda}^{\mathrm{t}}(\boldsymbol{x} - \boldsymbol{z}) + \frac{\rho}{2}\left\|\boldsymbol{x} - \boldsymbol{z}\right\|^2 .$$

The main points here are:

a. it is a separable function of the z_p, for $p = 1, \ldots P$, and

b. as a function of each z_p it is a simple second order polynomial (for $z_p \geq 0$) or $+\infty$ (for $z_p \leq 0$), as illustrated by the figure on the next page.

2. *Meditate that in-depth and make explicit the $\boldsymbol{z}$-update step (b).*

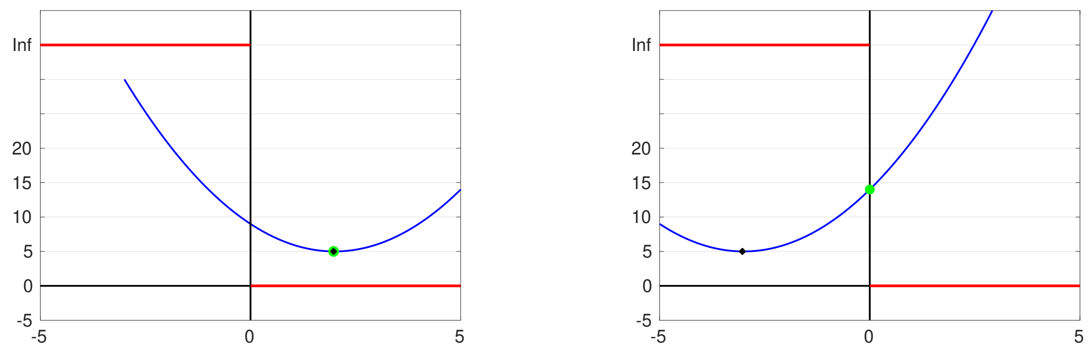

Figure 1: Indicator function $\mathcal{I}_\mathcal{C}$ (red) and second order polynomial (blue). Consider the summation of both of them. The unconstrained minimiser is black and the constrained one is green.

2.2.3 Update of the Lagrange multipliers $\boldsymbol{\lambda}$

The $\boldsymbol{\lambda}$-update (c) is separable and direct, very easy to implement in parallel within a one-line *Matlab* code. It is in direct relation with the residual for the equality constraint $\boldsymbol{x} - \boldsymbol{z}$ and pragmatically $\boldsymbol{\lambda}$ reaches a stable value when the constraint in met.

2.3 Practical implementation

This section deals with the implementation within the *Matlab* computing environment and the analysis of the results. The expected implementation accounts for positivity constraint only.

3. *Implement the optimisation method from the previous section. Take time to properly structure and comment your code. The $\boldsymbol{z}$-update should look like that code:*

```
% Unconstrained minimiser
    Auxiliary = Object + Lambda/rho;

% Account for positivity constraint
    Auxiliary(Auxiliary<0) = 0;
```

4. *Compare the result to the one obtained using the standard (unconstrained) Wiener-Hunt method, specifically regarding the image resolution and the black area around the brain.*

References

[1] J. Nocedal et S. J. Wright, *Numerical Optimization*, Series in Operations Research. Springer Verlag, New York, 2008.

[2] D. P. Bertsekas, *Nonlinear programming*, Athena Scientific, Belmont, MA, USA, 2nd edition, 1999.

[3] S. Boyd, N. Parikh, E. Chu, B. Peleato et J. Eckstein, *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*, vol. 3 de *Foundations and Trends in Machine Learning*, Now Publishers Inc, Hanover, MA, USA, janvier 2011.