Numerical Project in Python n.1

Image deblurring

Andrea Simonetto

version: January 7, 2026

We will consider the signal processing problem of deblurring an image. This will be formulated as an optimization problem with an ℓ_1 regularization penalty.

The goal of this project is for you to code a forward-backward method with Nesterov's acceleration and experience what it means to deal with the $f + g$ setting. **This project counts** 20% **of your grading.** The project will be graded as follows: 10 points for the technical exactness; 10 points for the clarity, formatting, and readability of the report; 10 points for the graphs (font size, explanations, selections).

Some instructions.

- You can do the project in pairs, as long as everybody does a comparable amount of work.
- After the first three hours, you need to submit what you have done at that point via the Moodle page of the course. Name the file: `LastName1_LastName2_Project1_After_three_hours.py`
- By **February, Tuesday 24, at 13H00 Paris time**, you need to submit a report in a single .pdf file, of no more than 6 pages, that is readable, and it has inside all the elements to assess your work. Name the file: `LastName1_LastName2_Project1_Final.pdf`

  You also need to submit a python script that we can run to assess the correctness of your plots and graphs. The script will have to generate the results that you have in the report. And it has to be readable. Name the file: `LastName1_LastName2_Project1_Final.py`

  No extension allowed, No resit allowed.
- As you figured already, you will use python, and in particular you will need the following packages (please refrain from using any other packages that are not strictly needed).

```
import numpy as np
import matplotlib.pyplot as plt
import pylops
```

- Do as much as you can during the TP session (3 hours) at ENSTA, but it is possible that you will need to do extra work in addition to it.
- Don't wait the last day to put together the project.

Writing a readable report is very important in life. The readability of the report will be very important for the final grade, as said. Some typesetting instructions follow.

1. Save and incorporate figures and graphs in .pdf format. (No .jpeg, no .png, no print-screen). In order to help you generate readable graphs, you have access to a setup example python script. It saves a sample figure in .pdf, and so must you.
2. The labels and legends of the graph should have a fontsize similar to the one used in the main text.
3. Equations should be punctuated. They are objects living in your text, so add commas, and full-stops.
4. Don't start a sentence with a symbol.
5. Don't mix French with English.
6. Don't put code in the text, or print-screen of results. Especially with black background.

Figure 1: The original and blurred image.

Getting started

You have access to a python script `project-deblur.py` who teaches you how to get the image and blur it, as below.

You can use either a dog image or a castle image (your choice).

An image can be thought of a series of pixels. Each pixel has three dimension, but here we consider only one channel, so each pixel is just a number between 0 and 255 (RBG scale). Let $z \in \mathbf{R}^n$ be the n pixels mapped into a column vector.

Blurring causes a linear transformation of the pixels into other pixels, and with the same notation of the notebook, you have

$$z' = Cz, \qquad z = W^H x.$$

Here it is not important what these matrices are, the only important information is that C has more column than rows (i.e., we are only observing a small portion of the image).

Reconstructing the true image implies solving an inverse problem. Let b be the pixels that we observe of the blurred image, and $CW^H x$ our model of how an healthy image is distorted to account for blur.

Ideally, one would solve the convex least-squares problem

$$x^* \in \arg\min_{x \in \mathbf{R}^n} \frac{1}{2}\|CW^H x - b\|_2^2, \qquad z^* = W^H x^*.$$

However, this problem is ill-conditioned, since C may have very few rows. As alternative, one can add regularization terms to the cost as penalties to favor special properties of the solution. One such property is a minimal energy solution (for which you would add $+\epsilon\|x\|_2^2$). Another is sparsity, for which you would add $+\epsilon\|x\|_1$. We look at the second choice, which is very important in signal processing.

The problem to solve

Consider then the ℓ_1 regularized problem,

$$x^* \in \arg\min_{x \in \mathbf{R}^n} \frac{1}{2}\|CW^H x - b\|_2^2 + \epsilon\|x\|_1, \qquad z^* = W^H x^*.$$

In this project, you will code a forward-backward algorithm to solve it. Let $A = CW^H$ for simplicity, and label $f_1(x) = \|CW^H x - b\|_2^2$, $f_2(x) = \epsilon\|x\|_1$.

1. **[2 points]** Is the problem convex? Prove that $f_1 \in \mathcal{S}_{0,L}^{1,1}$. Determine an expression for the Lipschitz constant L.

2. **[2 points]** Write the proximal gradient method applied to this problem and prove that it is equivalent to ($k \in \mathbf{N}$)

$$v_k = x_k - \alpha \nabla f_1(x_k) \qquad [x_{k+1}]_i = \text{sign}([v_k]_i)(|[v_k]_i| - \alpha\epsilon)_+ \tag{1}$$

   Compute the gradient of f_1.

When the proximal gradient method is applied to an ℓ_1 regularized least-squares problem, it yields (1) and these iterations are traditionally called the ISTA (i.e., iterative soft-thresholding algorithm).

Since $f_1 \in \mathcal{S}^{1,1}_{0,L}$, we can also think to apply a Nesterov's type acceleration, leading to FISTA (i.e., fast ISTA), as follows.

$$\lambda_0 = 0, \quad \lambda_{k+1} = \frac{1+\sqrt{1+4\lambda_k^2}}{2}, \quad \gamma_k = \frac{1-\lambda_k}{\lambda_{k+1}} \tag{2}$$

$$v_k = x_k - \alpha \nabla f_1(x_k) \quad [y_{k+1}]_i = \text{sign}([v_k]_i)(|[v_k]_i| - \alpha\epsilon)_+ \quad x_{k+1} = \gamma_k y_k + (1-\gamma_k) y_{k+1} \tag{3}$$

We have the following results,

ISTA (for $\alpha < 2/L$):

$$f_1(x_k) + f_2(x_k) - (f_1(x^*) + f_2(x^*)) \leq O(1/k).$$

FISTA (for $\alpha \leq 1/L$):

$$f_1(x_k) + f_2(x_k) - (f_1(x^*) + f_2(x^*)) \leq O(1/k^2).$$

ISTA and FISTA are two cornerstones of model signal processing and machine learning.

3. **[5 points]** Use `project-deblur.py` and code your own ISTA and FISTA algorithm in the function `my_fista`. Select $\alpha = 1/L$, and use the optimal solution of the already implemented

   `pylops.optimization.sparsity.fista` as optimal value (as shown in the code).

   Plot objective convergence, and the resulting images. You should obtain something similar to below.

   Careful. The matrices you are dealing with are very big and the reason to use `pylops` is to store and access them smartly. Do not convert the matrices in `numpy` arrays; just use the functions in `pylops`. For example, to compute the largest eigenvalue of a `pylops` matrix, use the function `A.eigs(neigs=1, symmetric=True)` if `A` is the symmetric matrix.

4. **[2 point]** Experiment with different ϵ values (now fixed at $\epsilon = 0.1$).

5. **[2 point]** Experiment with different step sizes.

6. **[2 point]** Experiment with different blurring (i.e., augment -0.1 and -0.3 by $\times 10$ and diminish them /10). The blurring can be found in

   ```
   hz = np.exp(-0.1 * np.linspace(-(nh[0] // 2), nh[0] // 2, nh[0]) ** 2)
   hx = np.exp(-0.3 * np.linspace(-(nh[1] // 2), nh[1] // 2, nh[1]) ** 2)
   ```

7. **[2 points]** In the second box, diminish the sampling to 3 (for dog) and to 1 (for castle), and comment on what happens. In fact, the dimension of your problem augments: describe how the computational time augments with the dimension of the problem.

Going beyond ISTA: Research questions

Choose one of the following research questions.

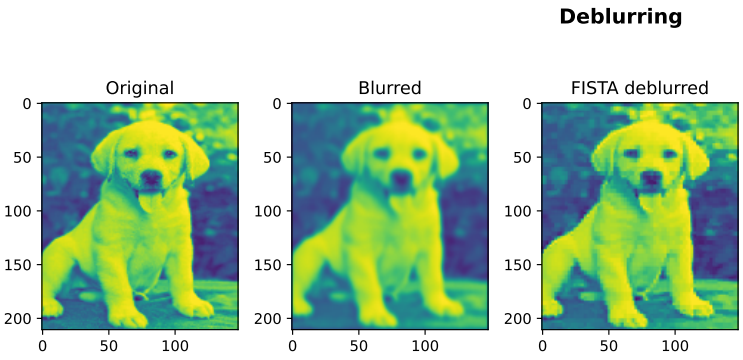
8. **[3 points]** Option I: Instead of proximal gradient, implement a Douglas-Rachford algorithm (See below) and comment on what happens. Now that we can select the step size freely, can we converge faster?

   Careful. You will have to be careful on how you compute the prox with `pylops`. Remember that inverting a matrix is never a good idea and find an alternative solution, if needed. Write down all the steps.

9. **[3 points]** Option II: Come back to the problem,

$$x^* \in \arg\min_{x \in \mathbf{R}^n} \frac{1}{2}\|CW^H x - b\|_2^2 + \epsilon\|x\|_1, \qquad z^* = W^H x^*,$$

   and solve it with the package `cvxpy`, which can solve convex optimization problems. You will have to pay particular attention at transforming the matrices into something that cvxpy can read.



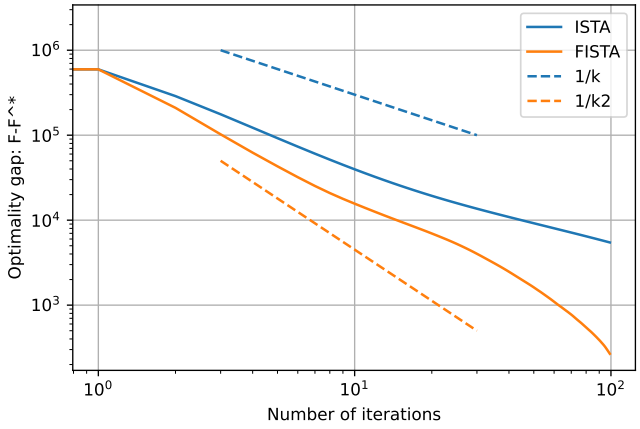


Figure 2: Sample of the results.

Careful. You will have to pay particular attention at transforming the matrices into something that `cvxpy` can read. In particular, you will need to use `numpy` arrays and sample down the image to obtain small enough matrices for the conversion to be possible. What is the largest image you can handle?

`cvxpy` uses an interior-point solver to solve the problem; what do you observe? Why do you think it is so?

Douglas-Rachford Splitting

Consider the problem,

$$\min_{x \in \mathbf{R}^n} f(x) + g(x),$$

with f and g convex closed and proper (CCP). Consider now the following method to find a solution x^*. Start at a certain z_0 and iterate for all $k \in \mathbb{N}$:

$$x_{k+1} = \operatorname{prox}_g(z_k) \tag{4a}$$

$$z_{k+1} = z_k + \operatorname{prox}_f(2x_{k+1} - z_k) - x_{k+1}, \tag{4b}$$

where prox is the usual prox operator. The method is called the Douglas-Rachford splitting and it converges in some defined sense. Note that no step size is used!

Total: /20 **Points.**