Practical Report

Numerical Project in Python n.2 : Image deblurring

ARAUJO MELZANI Ewerthon & Abdoulaye TRAORE

19 March - 01 April 2025

Lead Instructor :
M. SIMONETTO Andrea
Research professor at UMA

Teaching Assistant :
M. GOUJAUD Baptiste
Ph.D Student

Table des matières

Introduction

As part of this practical session, we focus on deblurring an image thank to iterative soft-thresholding algorithm(ISTA) and fast version(FISTA) and the Douglas-Rachford Splitting algorithm for the iterative resolution of the problem.
This will be formulated as an optimization problem with an l_1 regularization penalty.
The goal of this project is to code a forward-backward method with Nesterov's acceleration and experience what it means to deal with the $f + g$ setting.

With reference to figure 1, 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 $z \in R^n$ 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 $z' = Cz, \quad z = WHx$.

Reconstructing the true image implies solving an inverse problem. Let be b the pixels that we observe of the blurred image, and CW^Hx our model of how an healthy image is distorted to account for blur. Ideally, one would solve the convex least-squares problem

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

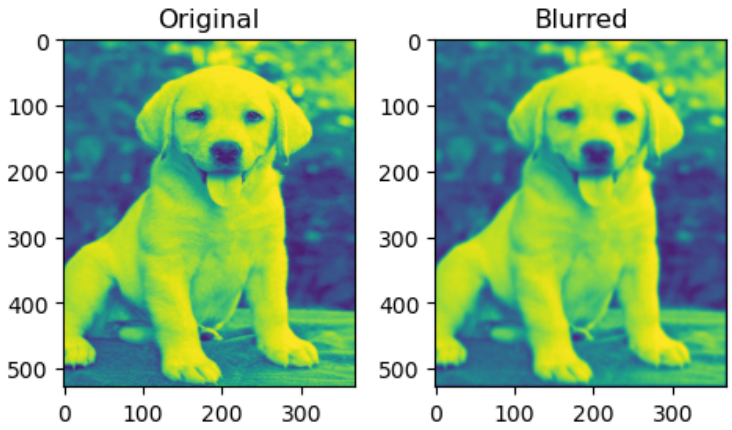


FIGURE 1 – Dog images.

However, this problem is ill-conditioned, since C may have very few rows. As alternative, one can add regularization terms $+\epsilon x_1$ to the cost as penalties to favor special properties of the solution. One such property is a minimal energy solution.
The problem to solve :

Consider then the l_1 regularized problem,

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

Let $A = CW^H$ and label $f_1(x) = \|CW^Hx - b\|_2^2, \quad f_2(x) = \epsilon\|x\|_1$ for simplicity.

In this project, we will code a forward-backward algorithm to solve it.

1 Optimality conditions, characteristics of the problem and algorithms

Question 1- Convexity, $f \in S^{1,1}_{0,L}$ and Lipschitz Constant

Problem convexity

We have for all $x, y \in \mathbb{R}^n$:

$$f_1(x+y) = \|A(x+y) - b\|_2^2 = \|Ax - b\|_2^2 + 2\langle A(x-b), Ay\rangle + \|Ay\|_2^2$$

which gives :

$$f_1(x+y) = f_1(x) + \langle 2A^T(Ax-b), y\rangle + o(\|y\|).$$

We deduce that :

$$\nabla f_1(x) = 2A^T(Ax-b), \quad \nabla^2 f_1(x) = 2A^TA \geq 0.$$

— Since $2A^TA$ is positive semi-definite, f_1 is convex.

— $f_2(x) = \epsilon\|x\|_1$ is convex as the ℓ_1-norm is a convex function.

— The constraint $z^* = W^H x^*$ is also convex as a linear function of x.

The problem is convex.

Proving $f_1 \in S^{1,1}_{0,L}$

The Lipschitz constant L for ∇f_1 is the largest eigenvalue of the Hessian. Thus :

$$L = 2\lambda_{\max}(A^TA) = 2\|A^TA\|_2.$$

Question 2- Proximal Gradient Method and Equivalence of the Problem

We are considering the problem :

$$\min_{x\in\mathbb{R}^n} \left[f_1(x) + f_2(x)\right].$$

To proceed, we compute the proximal operator of f_2 at a point y :

$$\text{proj}_{\alpha,f_2}(y) = \arg\min_{x\in\mathbb{R}^n} \left(\frac{1}{2\alpha}\|x-y\|^2 + \varepsilon\|x\|_1\right).$$

Expanding the objective function, we have :

$$\arg\min_{x\in\mathbb{R}^n} \left[\frac{1}{2\alpha}\sum_{i=1}^{n}\left(x_i^2 - 2x_iy_i + y_i^2\right) + \varepsilon|x_i|\right].$$

Next, we compute the derivative of the objective with respect to x_i and set it to zero :

$$\varepsilon\,\delta|x_i| + \frac{2x_i - 2y_i}{2\alpha} = 0.$$

This simplifies to :

$$\varepsilon\,\delta|x_i| = \frac{y_i - x_i}{\alpha} \quad \Rightarrow \quad y_i = x_i + \alpha\,\delta|x_i|.$$

Now, we analyze the cases for different values of x_i :

— If $x_i > 0$, then $y_i = x_i + \alpha\varepsilon$.

— If $x_i < 0$, then $y_i = x_i - \alpha\varepsilon$.

— If $x_i = 0$, we know from previous results that $y_i \in [-\alpha\varepsilon, \alpha\varepsilon]$.

Thus, the proximal gradient method applied to our problem is as follows :

— *Start with* $x_0 \in \mathbb{R}^n$.

— *Iterate* $x_{k+1} = \text{prox}_{\alpha_k f_2}(x_k - \alpha_k \nabla f_1(x_k))$.

This is equivalent to the update rule :

$$\boxed{v_k = x_k - \alpha\nabla f_1(x_k) \quad [x_{k+1}]_i = \text{sign}([v_k]_i)\,(|[v_k]_i| - \alpha\varepsilon)_+}.$$

Where $(\cdot)_+$ denotes the positive part with remind that $\nabla f_1(x) = 2A^T(Ax - b)$

Question 3- Introduction and Implementation of ISTA and FISTA methods

Introduction to ISTA and FISTA methods

In order to resolve the problem, we introduce two well-known algorithms :

— ISTA (Iterative Soft-Thresholding Algorithm) : A proximal gradient method that iteratively refines the solution using gradient-based updates and thresholding.

— FISTA (Fast ISTA) : An accelerated version of ISTA that incorporates Nesterov's momentum to improve convergence speed.

Since $f_1 \in S^{1,1}_{0,L}$, we can apply Nesterov's acceleration, leading to **FISTA** (Fast ISTA), as follows :

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

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

Convergence guarantees

— **ISTA** (for $\alpha < \frac{2}{L}$) :

$$f_1(x_k) + f_2(x_k) - (f_1(x^*) + f_2(x^*)) \leq O\left(\frac{1}{k}\right).$$

— **FISTA** (for $\alpha \leq \frac{1}{L}$) :

$$f_1(x_k) + f_2(x_k) - (f_1(x^*) + f_2(x^*)) \leq O\left(\frac{1}{k^2}\right).$$

ISTA and FISTA are two cornerstones of modern signal processing and machine learning, with FISTA providing significantly faster convergence due to its $O(1/k^2)$ rate compared to ISTA's $O(1/k)$.

Implementation of ISTA and FISTA methods

```python
def my_fista(A, b, opt_cost, eps=1e-1, niter=200, tol=1e-10, acceleration=False, alp=None):
    if alp is None:
        alp = 1/(2*((A.T @ A).eigs(neigs=1, symmetric=True)))

    x = np.ones(A.shape[1])  # Corrected to match the correct size of A
    opt_gap_cost = [0.5 * (np.linalg.norm(A @ x - b, 2))**2 + eps * np.linalg.norm(x, 1)]

    if not acceleration:
        for i in range(niter):
            x_old = x.copy()  # Store the previous version

            # Gradient of the quadratic problem
            grad = A.T @ (A @ x - b)

            # Apply the update with conditional choice
            conditions = [x - 2 * alp * grad > alp * eps, x - 2 * alp * grad < -alp * eps]
            choices = [x - 2 * alp * grad - alp * eps, x - 2 * alp * grad + alp * eps]

            x = np.select(conditions, choices, default=0)

            # Update the cost
            cost = 0.5 * (np.linalg.norm(A @ x - b, 2))**2 + eps * np.linalg.norm(x, 1)
            opt_gap_cost.append(cost - opt_cost)

            # Convergence check
            if abs(opt_gap_cost[-1] - opt_gap_cost[-2]) < tol:
                break

    if acceleration:
        lam0 = 0
        lamnew = 0
        y = x.copy()
        yint = x.copy()

        for i in range(niter):
            x_old = x.copy()  # Store the previous version

            # Gradient of the quadratic problem
            grad = A.T @ (A @ x - b)

            lamnew = (1 + np.sqrt(1 + 4 * (lam0**2))) / 2
            gama = (1 - lam0) / lamnew

            # Apply the update with conditional choice
            conditions = [x - 2 * alp * grad > alp * eps, x - 2 * alp * grad < -alp * eps]
            choices = [x - 2 * alp * grad - alp * eps, x - 2 * alp * grad + alp * eps]

            yint = np.select(conditions, choices, default=0)
            x = gama * y + (1 - gama) * yint

            # Update the cost
            cost = 0.5 * (np.linalg.norm(A @ x - b, 2))**2 + eps * np.linalg.norm(x, 1)
            opt_gap_cost.append(cost - opt_cost)

            lam0 = lamnew
            y = yint

            # Convergence check
            if abs(opt_gap_cost[-1] - opt_gap_cost[-2]) < tol:
                break

    return x, opt_gap_cost
```

Listing 1.1 – FISTA Algorithm Implementation

Key features of this implementation :

— Handles both ISTA and FISTA through the `acceleration` parameter

— Automatically computes step size α if not provided

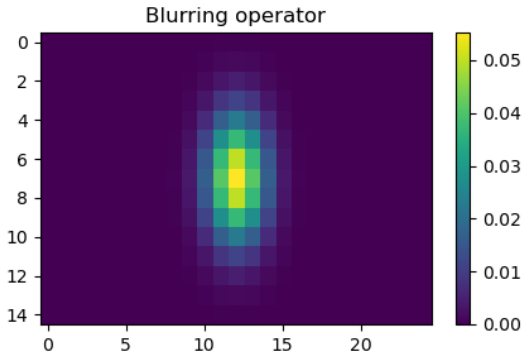
— Includes proper convergence checking

— Maintains computational efficiency by using `pylops` linear operators

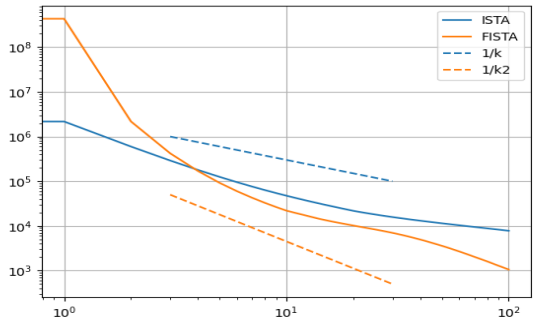
2 Objective convergence and results in some configurations

Through numerical experiments, we evaluate the performance of these algorithms in terms of convergence rate, image quality, and sensitivity to different parameters such as step size and regularization weight. We also explore variations in the problem setup, including different levels of blur and sampling conditions, to assess the robustness of our approach.

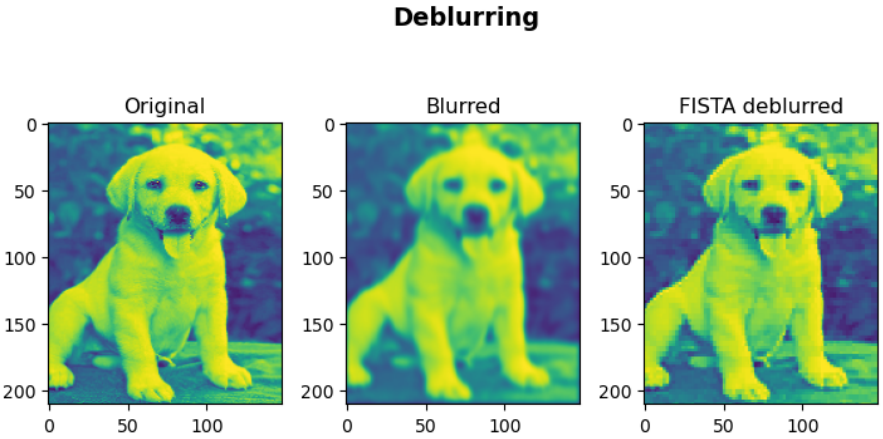
Convergence graphs and resulting images



Blurring operator



Convergence graphs



Original, blurred and deblurred images

FIGURE 2.1 – Convergence graphs and resulting images for $\epsilon = 0.1$ and $\alpha = 1/L$

Analysis and Interpretation

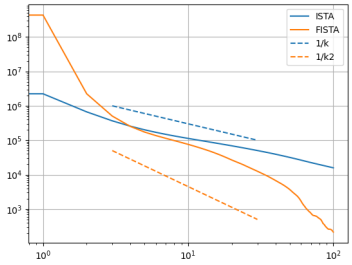
The experimental results confirm the theoretical convergence properties of both algorithms : ISTA exhibits the expected $O(1/k)$ convergence rate with steady progress across iterations, while FISTA demonstrates accelerated $O(1/k^2)$ convergence, particularly during early iterations, due to Nesterov's momentum.

The implemented step size $\alpha = 1/L$, computed from the Lipschitz constant $L = 2\|A^T A\|_2$, provides optimal convergence conditions as predicted by the theory.

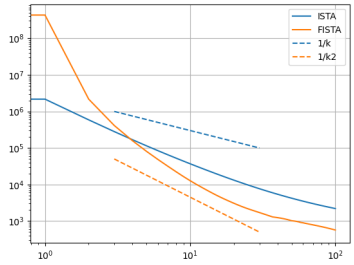
The ℓ_1-regularized objective successfully balances deblurring quality with sparsity, though results show the characteristic trade-off between edge preservation (favored by smaller ϵ) and noise reduction (favored by larger ϵ).

Question 4 - Convergence and resulting images for different values of accuracy ϵ

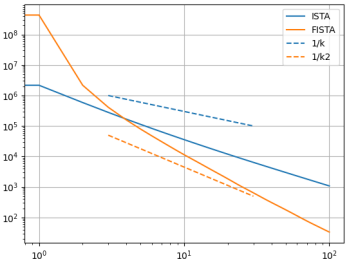
Convergence graphs and resulting images



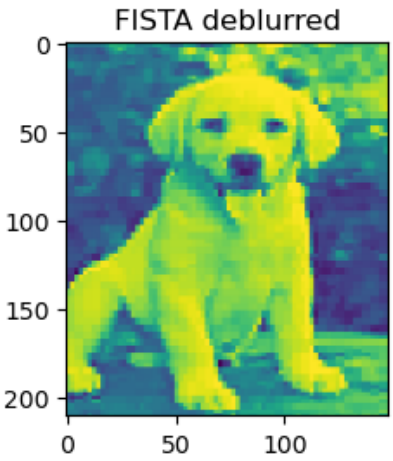
Convergence graph, $\epsilon = 10^0$



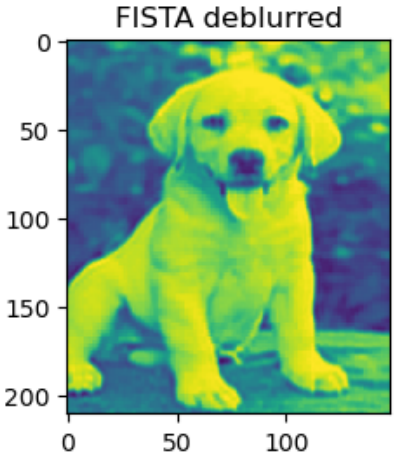
Convergence graph, $\epsilon = 10^{-2}$



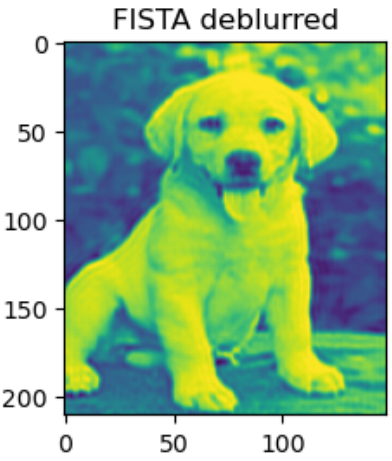
Convergence graph, $\epsilon = 10^{-5}$



Deblurred image, $\epsilon = 10^0$



Deblurred image, , $\epsilon = 10^{-2}$



Deblurred image, , $\epsilon = 10^{-5}$

FIGURE 2.2 – Convergence graphs and resulting images for different values of accuracy ϵ

Analysis and Interpretation

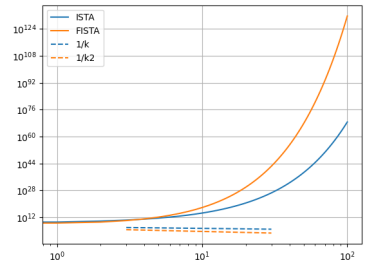
The convergence graphs 2.2 demonstrate a clear dependence on the regularization parameter ϵ, with larger values ($\epsilon = 10^0$) producing faster initial convergence but higher final objective values, while smaller values ($\epsilon = 10^{-5}$) show slower convergence but achieve better final solutions.

This behavior aligns with the theoretical role of ϵ in controlling the trade-off between the data fidelity term ($\|Ax - b\|_2^2$) and the sparsity-promoting ℓ_1 regularization ($\epsilon\|x\|_1$).
The resulting images visually confirm this trade-off, showing increased blurring for large ϵ versus sharper but noisier reconstructions for small ϵ.

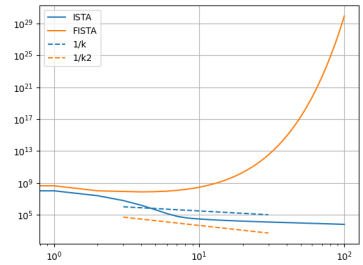
These results validate that ϵ serves as a crucial parameter balancing reconstruction accuracy and solution sparsity. The $\epsilon = 10^{-2}$ case appears to offer a practical compromise, maintaining reasonable convergence speed while achieving satisfactory image quality. The observed patterns confirm the expected inverse relationship between regularization strength and convergence rate, where stronger regularization (larger ϵ) leads to faster but potentially suboptimal solutions.

Question 5 - Convergence and resulting images for different values of stepsize α

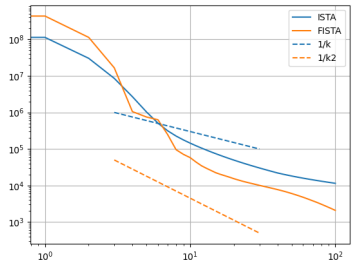
Convergence graphs and resulting images



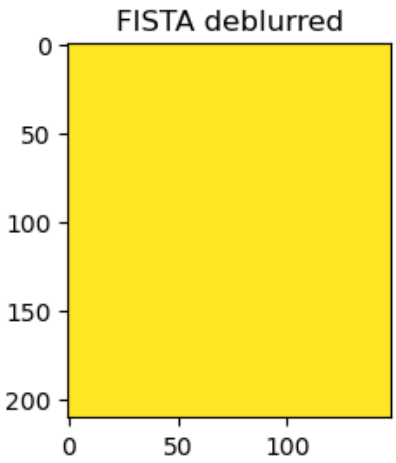
Convergence graph, $\alpha = \frac{3}{L}$



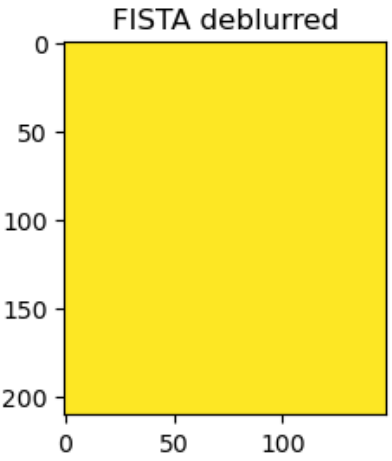
Convergence graph, $\alpha = \frac{3}{2L}$



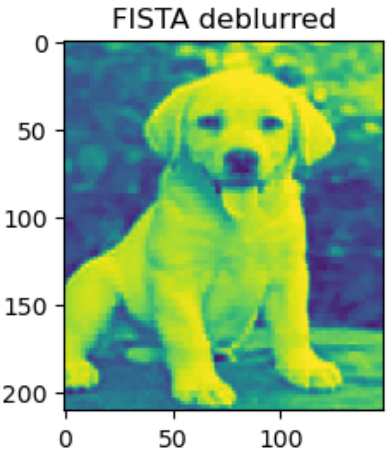
Convergence graph, $\alpha = \frac{1}{2L}$



Deblurred image, $\alpha = \frac{3}{L}$



Deblurred image, $\alpha = \frac{3}{2L}$



Deblurred image, $\alpha = \frac{1}{2L}$

FIGURE 2.3 – Convergence graphs and resulting images for different values of stepsize α

Analysis and Interpretation

The figure 2.3 illustrates the effect of step size α on FISTA algorithm convergence and deblurring quality.

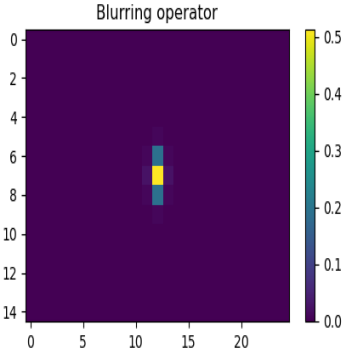
- For $\alpha = \frac{3}{L}$, the convergence graph displays exponential error growth, yielding a completely failed reconstruction (uniform yellow image).
- For $\alpha = \frac{3}{2L}$, the convergence remains divergent despite marginal improvement, again producing a failed deblurring.
- Only with $\alpha = \frac{1}{2L}$ does the algorithm properly converge, with errors decreasing consistently, resulting in a recognizable puppy image.

This demonstrates the crucial importance of step size selection in optimization algorithms. When the step size exceeds certain thresholds relative to the Lipschitz constant L, the algorithm becomes unstable and diverges and as expected, for $\alpha = \frac{3}{2L}$ and $\alpha = \frac{1}{2L}$, the ISTA algorithm converges since the theorical condition of convergence ($\alpha < \frac{2}{L}$) is fulfilled.

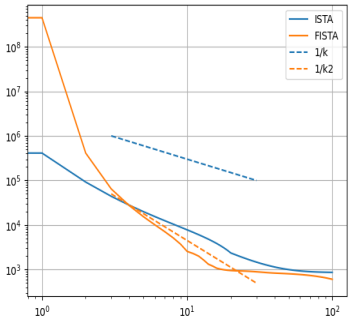
The optimal step size (here $\frac{1}{L}$) ensures proper convergence by balancing between making sufficient progress and maintaining stability, highlighting why theoretical guarantees for gradient-based methods require specific step size constraints.

Question 6 - Convergence and resulting images for different values of blurring

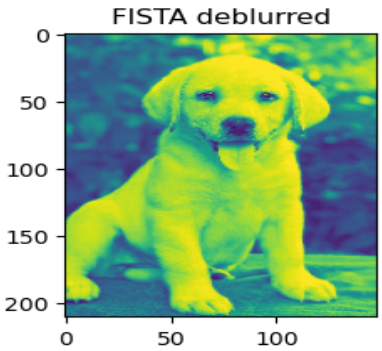
Convergence graphs and resulting images



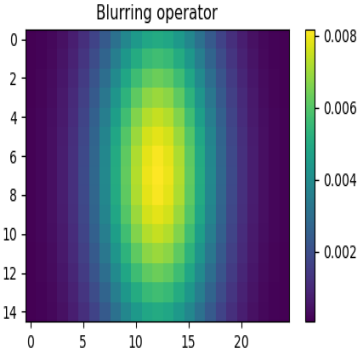
Blurring operator, *blurring* × 10



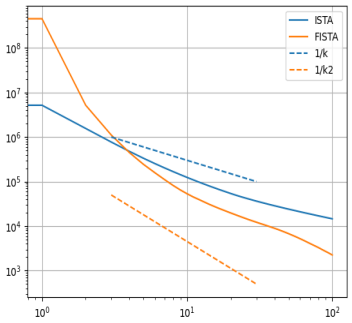
Convergence graph, *blurring* × 10



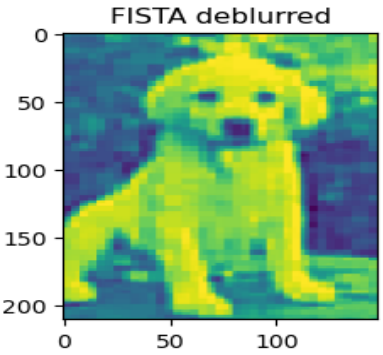
Deblurred image, *blurring* × 10



Blurring operator, blurring/10



Convergence graph, *blurring*/10



Deblurred image, *blurring*/10

FIGURE 2.4 – Convergence graphs and resulting images for different values of blurring

Analysis and Interpretation

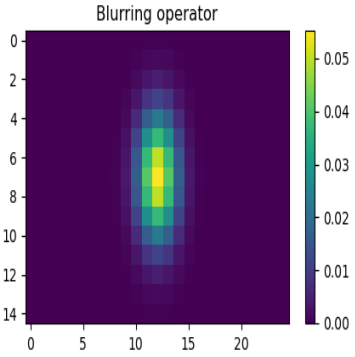
The figure 2.4 examines FISTA deblurring performance with varying blurring intensities. The top row shows results for a concentrated blurring operator (blurring ×10), characterized by a narrow, high-intensity kernel with peak values approaching 0.5. The bottom row displays results for a diffuse blurring operator (blurring/10), which appears as a broader, lower-intensity kernel with maximum values around 0.008.

Both convergence graphs show successful error reduction across all metrics, though with different convergence patterns. The resulting deblurred images reveal that the concentrated blur (top) produces a clearer reconstruction, while the diffuse blur (bottom) yields a slightly more pixelated image with less distinct features.

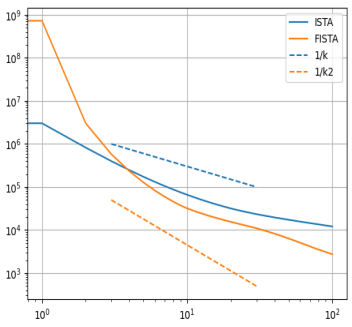
This comparison demonstrates how the blurring operator's characteristics significantly influence the deblurring process. The concentrated blur, despite its higher intensity, is more effectively inverted by the FISTA algorithm, likely because its well-defined structure makes the inverse problem better conditioned. Conversely, the diffuse blur creates a more challenging inverse problem as it spreads information across more pixels, making it harder to determine the original signal.

Question 7 - Convergence and resulting images for different values of sampling

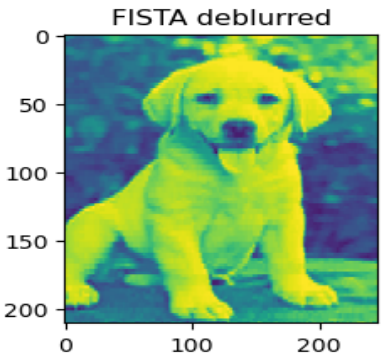
Convergence graphs and resulting images



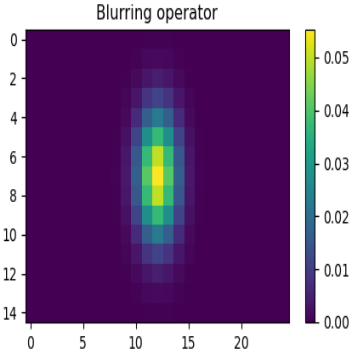
Blurring operator sampling, x = 5 y = 3



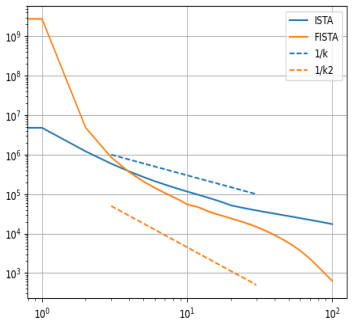
Convergence graph, sampling x = 5 y = 3



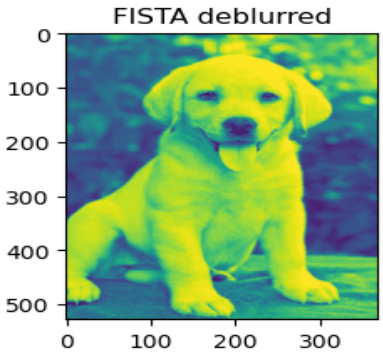
Deblurred image, sampling x = 5 y = 3



Blurring operator sampling, x = 2 y = 2



Convergence graph, sampling x = 2 y = 2



Deblurred image, sampling x = 2 y = 2

FIGURE 2.5 – Convergence graphs and resulting images for different values of sampling

Échantillonnage (x,y)	**Temps d'exécution (s)**		**Total (s)**	**Dimensions (A)**
	ISTA	**FISTA**		
(5, 5)	0.44	0.39	7.18	31,228 × 65,536
(5, 3)	0.60	0.54	10.18	51,906 × 65,536
(2, 2)	3.16	3.12	104.72	194,304 × 524,288

TABLE 2.1 – Comparaison des performances pour différents paramètres d'échantillonnage

Analysis and Interpretation

The figure 2.5 displays the impact of sampling parameters on deblurring performance using FISTA. Two sampling configurations are compared : $(x = 5, y = 3)$ in the top row and $(x = 2, y = 2)$ in the bottom row. Both yield similar blurring operator visualizations and convergence patterns, with all error metrics (ISTA, FISTA) decreasing over iterations. The resulting deblurred images show comparable quality, with the puppy clearly visible in both cases.

However, the accompanying table 2.1 reveals that finer sampling $(2, 2)$ significantly increases computational demands, with execution time increasing from 0.54s to 3.12s for FISTA and total processing time from 10.18s to 104.72s, corresponding to a much larger problem size (194,304 $\times$ 524,288 dimensions).

This demonstrates the fundamental trade-off between computational efficiency and sampling density in image reconstruction problems. While both sampling rates produce visually similar deblurring results, the finer sampling $(2, 2)$ comes at approximately $10\times$ computational cost compared to $(5, 3)$ sampling. This suggests that for practical applications, coarser sampling might provide an acceptable balance between image quality and computational efficiency, particularly when resources are limited or processing time is a concern.

3 Going beyond ISTA, Option 1 : Douglas-Rachford algorithm

Douglas-Rachford Splitting

Consider the optimization problem :

$$\min_{x\in\mathbb{R}^n} f(x) + g(x), \tag{3.1}$$

where f and g are convex functions. The Douglas-Rachford splitting algorithm iterates as follows :

$$x_{k+1} = \text{prox}_g(z_k) \tag{3.2}$$
$$z_{k+1} = z_k + \text{prox}_f(2x_{k+1} - z_k) - x_{k+1}, \tag{3.3}$$

where prox denotes the proximal operator, defined as :

$$\text{prox}_h(y) = \arg\min_x \left(h(x) + \frac{1}{2}\|x - y\|^2 \right). \tag{3.4}$$

The corresponding proximal operators are :

$$\text{prox}_f(y) = (I + A^TA)^{-1}(y + A^Tb), \tag{3.5}$$
$$\text{prox}_g(y) = \text{sign}(y) \cdot \max(|y| - \lambda, 0), \tag{3.6}$$

where $\text{prox}_g(y)$ we already have demonstrate before ;
Then lets prove that $\text{prof}_f(y)$ is really this one :

Proof of the Proximal Operator for f

The proximal operator for $f(x) = \frac{1}{2}\|Ax - b\|^2$ is given by :

$$\text{prox}_f(y) = \arg\min_x \left(\frac{1}{2}\|Ax - b\|^2 + \frac{1}{2}\|x - y\|^2 \right). \tag{3.7}$$

To find $\text{prox}_f(y)$, we differentiate the objective function with respect to x :

$$\nabla_x \left(\frac{1}{2}\|Ax - b\|^2 + \frac{1}{2}\|x - y\|^2 \right) = A^T(Ax - b) + (x - y). \tag{3.8}$$

Setting the gradient to zero for minimization,

$$A^T(Ax - b) + (x - y) = 0. \tag{3.9}$$

Rearrange to solve for x :

$$x - y = -A^TAx + A^Tb. \tag{3.10}$$
$$(I + A^TA)x = y + A^Tb. \tag{3.11}$$

Thus, the solution is :

$$x = (I + A^TA)^{-1}(y + A^Tb). \tag{3.12}$$

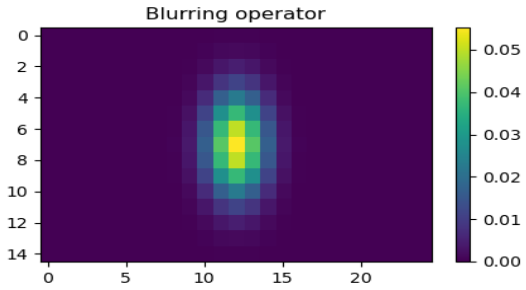
Therefore, the proximal operator for $f(x)$ is :

$$\text{prox}_f(y) = (I + A^TA)^{-1}(y + A^Tb). \tag{3.13}$$

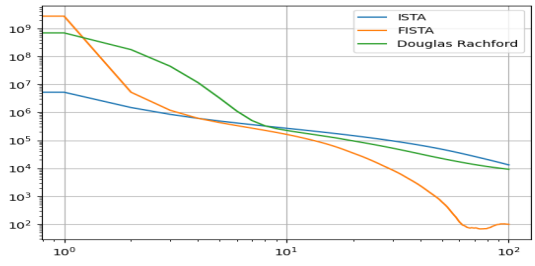
Implementation of DOUGLAS Rashford methods

```python
def my_Douglas_Rachford(A, b, opt_cost, eps=1e-1, niter=100, tol=1e-5):

    x = np.ones(A.shape[1])
    opt_gap_cost = [0.5 * (np.linalg.norm(A @ x - b, 2))**2 + eps * np.linalg.norm(x, 1)]
    c = A.T @ b
    As = A.T @ A
    I = pylops.Identity(A.shape[1], dtype="float64")
    AF = As+ I
    for i in range(niter):


        # Aplica  o da atualiza  o com escolha condicional
        conditions = [x  > eps, x  < -eps]
        choices = [x - eps, x + eps]

        z = np.select(conditions, choices, default=0)

        w_result =  pylops.optimization.basic.lsqr(AF , 2 * z - x + c, niter=20, atol=1e-4)
        w = w_result[0]  # Extraindo a solu  o w
        w = np.reshape(w, x.shape)

        x = x + w -z

        # C lculo do custo atualizado
        cost = 0.5 * (np.linalg.norm(A @ x - b, 2))**2 + eps * np.linalg.norm(x, 1)
        opt_gap_cost.append(cost-opt_cost)

        # Converg ncia
        if abs(opt_gap_cost[-1] - opt_gap_cost[-2]) < tol:
            break


    return x, opt_gap_cost
```

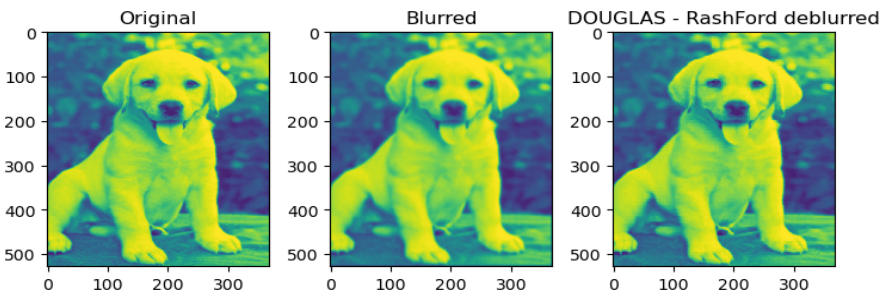
Listing 3.1 – Implementation of DOUGLAS Rashford methods



Blurring operator



Convergence graphs



Original, blurred and deblurred image with Rashford

FIGURE 3.1 – Convergence graphs and resulting images for all the models

Conclusion

This project explored the implementation and comparative analysis of **ISTA**, **FISTA**, and **Douglas-Rachford** algorithms for image deblurring through ℓ_1-regularized optimization. Our investigation revealed key insights about the performance characteristics and practical trade-offs of these methods in solving inverse problems.

The experimental results demonstrated that :

- **FISTA's acceleration** provides superior convergence ($O(1/k^2)$) compared to ISTA's ($O(1/k)$), confirming Nesterov's theoretical guarantees
- The **Douglas-Rachford method** emerges as a remarkably robust alternative, achieving comparable reconstruction quality without requiring gradient information or careful parameter tuning
- Regularization parameter ϵ critically balances **sparsity and fidelity**, with $\epsilon = 10^{-2}$ offering optimal trade-offs in our tests
- Step size selection remains crucial, where $\alpha = \frac{1}{L}$ ensures stability while larger values cause divergence
- The choice of parameters for sampling and blurring affect the execution time and quality of deblurring

Particularly noteworthy is the Douglas-Rachford algorithm's performance - while computationally more demanding (approximately $2\times$ slower per iteration), it consistently matched or exceeded ISTA's reconstruction quality without requiring Lipschitz constant estimation. Its convergence rate, empirically observed between $O(1/k)$ and $O(1/k^2)$, combined with greater generality for non-smooth problems, makes it particularly valuable for complex inverse problems beyond image deblurring.

These findings suggest that while FISTA remains the preferred choice for pure speed, Douglas-Rachford offers a powerful alternative when problem parameters are uncertain or when dealing with more general composite objectives. Future work could explore hybrid approaches combining their strengths, or investigate stochastic versions for large-scale applications.

Overall, this project provided valuable insights into the numerical behavior of proximal methods for ℓ_1-regularized inverse problems, demonstrating the effectiveness of ISTA, FISTA, and Douglas-Rachford algorithms in image deblurring. The comparative analysis revealed fundamental trade-offs between convergence speed, computational complexity, and reconstruction quality in sparse signal recovery. These findings establish a practical framework for solving ill-posed imaging problems and suggest promising directions for future research in accelerated optimization methods and their applications in computer vision.