COURSE PROJECT

ADVANCED CONVEX OPTIMIZATION

JAN-APR 2022

# A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems

*Author*
Bhartendu Kumar

*Instructor*
Dr. Kunal N. Chaudhury

April 22, 2022

## Abstract

The objective function that we have is:

$$F(x) = \frac{1}{2}\|b - Ax\|^2 + \lambda\|x\|_1 \tag{0.1}$$

Here the first term, $f(x)$ is a smooth, convex function with Lipschitz continuous gradient $A^T(Ax - b)$ and $L_f = \|A^T A\|$.

FISTA has a faster convergence rate as compared to ISTA. The main difference between the two is that the iterative shrinkage operator is not applied to the previous point alone but to another point which uses a specific linear combination of the previous two points.

Image denoising and deblurring problems can be modeled with the linear system

$$Ax + w = b, \tag{0.2}$$

where $b$ is the observed image, $w$ is some unknown noise, and $x$ is the true image we would like to recover. For a blurred image we let $A$ represent the blurring process, and for noisy images we let $A$ be the identity matrix.

# Contents

# 1 Introduction

The model of "Linear Inverse Problem" is :

Ax = b + w

Where Matrix A is $\mathcal{R}^{mxn}$, b is vector $\mathcal{R}^{m}$ , w and x are unknown vectors of dimensions $\mathcal{R}^{m}$ and $\mathcal{R}^{n}$ respectively. Here the inverse problem is to estimate vector x when we can observe $b + w$ and know A.
This general model can also represent **image deblurring problem** where image is vector x and Matrix A is blurring the vectorized image x. And we can observe blurred and noisy image $b + w$, thus deblurring is to estimate x in this setting.

Often **A** is $A = RW$ , $R$ is blurring Matrix and $W$ has wavelet basis. Multiplying by W on right can be seen as inverse wavelet transform. Thus before applying the blurring matrix **R** to image **x** we transform the image **x** to wavelet domain. Here L1 regularization is mostly effective as the image becomes relatively sparse when we apply the wavelet transformation

## 1.1 Objective Function

- Least Square

  (OLS): $\hat{x}_{OLS} = argmin_x ||Ax - b||^2$

- L2 regularized Least Square (Tikhonov regularization)

  (l2): $\hat{x}_{L2} = argmin_x ||Ax - b||^2 + \lambda ||Lx||^2$

- L1 regularized Least Square

  (l1): $\hat{x}_{L1} = argmin_x ||Ax - b||^2 + \lambda ||x||_1$

  - The importance of this objective function is that l1 regularization induces sparsity in $\hat{x}_{L1}$ and this is useful when we are dealing in wavelet domain.
  - ly l1 is less sensitive to outliers (sharp edges).

**Regularization Parameter** $\lambda$ controls the extent of importance given to regularization term w.r.t. the Least Squares term.

**L** is chosen as per Lipschitz constant of derivative of objective function.

## 1.2 Soft Thresholding Operator

**Proximal Operator for L1 regularized least squares:**

We have the Objective function of Optimization in L1 regularized least square as:

$$(l1) : \hat{x}_{L1} = argmin_x ||Ax - b||^2 + \lambda ||x||_1$$

Now,

$$prox_g(\hat{x}) = \arg\min_x \{\frac{1}{2}||x - \hat{x}||^2 + g(x)\} \tag{1.1}$$

Now when we have function $g$ as $\lambda||x||_1$, the proximal operator can be written as:

$\text{prox}_{\alpha||.||_1}(\hat{x}) = \arg\min_x \{\mathcal{H}''' \equiv \frac{1}{2\alpha}||x - \hat{x}||^2 + \lambda||x||_1\}$Analysing the function $\mathcal{H}'''$ as $x \in \mathcal{R}^n$
By definition:

$$\lambda||x||_1 + \frac{1}{2\alpha}||x - \hat{x}||_2^2 = \sum_{i=1}^{n} \lambda|x_i| + \frac{1}{2\alpha}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2$$

We can write solve the optimization as solving in each direction, i.e. a set of n scalar problems:

$$prox_{\alpha|.|}(\hat{x}_i) = \arg\min_x \{\lambda|x_i| + \frac{1}{2\alpha}(x_i - \hat{x}_i)^2\}$$

$\text{x}_i^* = prox_{\alpha|.|}(\hat{x}_i) \Leftrightarrow 0 \in \partial(\lambda|x_i| + \frac{1}{2\alpha}(x_i - \hat{x}_i)^2)$

$0 \in x_i - x_i + \alpha\hat{\lambda}\partial(|x_i|)$

with $\xi \in \partial|x_i|$

$0 = \text{x}_i - \hat{x}_i + \alpha\lambda\xi$

x$_i^* = \hat{x}_i \alpha \lambda \xi$

Thus we get,

$$x_i^* = S_{\alpha\lambda}(\hat{x}_i) = sign(\hat{x}_i) max(|\hat{x}_i| - \alpha\lambda, 0)$$

This implies,

$$prox_{\mu||.||_1}(x) = S_\mu(x)$$

## 1.3 TV for denoising

Optimization problem :

$$\min_X \lambda \|X\|_{TV} + \|F - X\|_2^2$$

Rationale behind it:
we need an item so that the difference between denoised image and Noisy image is not too large. So a new variable is introduced to this problem.

$$dissimilarity = \|F - X\|_2^2$$

By minimize this term, we can make the denoised image and Noisy image similar to each other.

### 1.3.1 Total Variance in Image denoising

Applying gradient descent (GD) algorithm as the first step of denoising.

$$\min_X \lambda \|X\|_{TV} + \|F - X\|_2^2.$$

Using gradient descent algorithm, the problem can be solved as:

Gradient Descent Algorithm

```
procedure GRADIENT DESCENT ALGORITHM WITH EXACT SEARCH
    given a starting point $x \in \mathrm{dom}\, f$
    repeat:
    1. $\Delta x := -\nabla f(x)$
    2. Line search. Choose step size $t$ via exact or backtracking line search.
    3. Update. $x := x + t\Delta x$
    *until* stopping criterion is satisfied.
**end procedure**
```

We need to compute $\Delta x := -\nabla f(x)$ every step, then we need to do exact line search in the direction of the negative side of the gradient.

### 1.3.2 Calculate the gradient of the objective function

Both of the following two items will be partially differentiated separately.

$$\|X\|_{TV} = \sum_{i=1}^{n-1}\sum_{j=1}^{n-1}\sqrt{(X_{i,j} - X_{i+1,j})^2 + (X_{i,j} - X_{i,j+1})^2}$$

$$dissimilarity = \|F - X\|_2^2$$

#### Calculate the partial differentiation of $\|X\|_{TV}$

According to the differential equation of the equation of TV, for every $x_{i,j}$, there will be three part of equations connected with it, and they are

$$\sqrt{(X_{i,j} - X_{i+1,j})^2 + (X_{i,j} - X_{i,j+1})^2},$$
$$\sqrt{(X_{i-1,j} - X_{i,j})^2 + (X_{i-1,j} - X_{i-1,j+1})^2},$$
$$\sqrt{(X_{i,j-1} - X_{i+1,j-1})^2 + (X_{i,j-1} - X_{i,j})^2},$$

Using the three equations connedted with $x_{i,j}$, it can be found that they can be seperated to to part, $\sqrt{(x-a)^2 + (x-b)^2}$ and $\sqrt{(a-x)^2 + (a-b)^2}$. To Calculate $\frac{\partial}{\partial x}\|X\|_{TV}$, the partial diffrenciation of these two equation should be calculated first.

$$\frac{\partial}{\partial x}(\sqrt{(x-a)^2 + (x-b)^2}) = \frac{-a - b + 2x}{\sqrt{a^2 - 2ax + b^2 - 2bx + 2x^2}}$$

$$\frac{\partial}{\partial x}(\sqrt{(a-x)^2 + (a-b)^2}) = -\frac{a - x}{\sqrt{(a-b)^2 + (a-x)^2}}$$

#### Calculate the partial differentiation of $\|F - X\|_2^2$

$$\frac{\partial}{\partial x}\left((a-x)^2\right) = 2x - 2a$$

# 2 Preliminary Lemmas and Proofs

## 2.1 Proximal view of Gradient Descent

Let $f$ be convex ,continuously differentiable, $f : \mathcal{R}^n \to \mathcal{R}$ To iteratively minimize the function $f$ we can use gradient descent algorithm, having the update rule:

$$x_k = x_{k-1} - t_k \nabla f(x_{k-1}) \tag{2.1}$$

, $t_k$ be suitable stepsize

The proximal view of this update is :

$$x_k = \arg\min_x \{H_{k-1}(x) \equiv f(x_{k-1}) + \nabla f(x_{k-1})^T (x - x_{k-1}) + \frac{1}{2t_k} || \ x - x_{k-1} \ ||_2^2\} \tag{2.2}$$

The blue part in the formula above is the tangent, or the first-order Taylor approximation at $x_{k-1}$, while the red part is a measure of proximity to $x_{k-1}$. In other words, the gradient step can be interpreted as:

> **find a point which balances between descending along the tangent at $x_{k-1}$, and staying in close proximity to $x_{k-1}$**

To show that this proximal characterization is equivalent to the gradient descent step, we optimize the $H_{k-1}(x)$ function for $x$.
By Fermat's principle we have $\nabla H_{k-1}(x_k) = 0$,

$\nabla f(x_{k-1}) + \frac{1}{t_k}(x_k - x_{k-1}) = 0$
giving us $x_k = x_{k-1} - t_k \nabla f(x_{k-1})$

## 2.2 L1 regularized objective

$$\min f(x) + \lambda||x||_1 : x \in \mathcal{R}^n \tag{2.3}$$

Now seeing this objective function and using the analogous proximal view point, we have :

$$x_k = \arg\min_x \{H'_{k-1}(x) \equiv f(x_{k-1}) + \nabla f(x_{k-1})^T(x - x_{k-1}) + \frac{1}{2t_k}||\ x - x_{k-1}\ ||_2^2 + \lambda||x||_1\}$$

We can rewrite as,

$$x_k = \arg\min_x \{H'_{k-1}(x) \equiv f(x_{k-1}) + \frac{2t_k}{2t_k}\nabla f(x_{k-1})^T(x - x_{k-1}) + \frac{1}{2t_k}\ ||\ x - x_{k-1}\ ||_2^2 + \lambda||x||_1\}$$

$$x_k = \arg\min_x \{H'_{k-1}(x) \equiv f(x_{k-1}) + \frac{1}{2t_k}||x - (x_{k-1} - t_k\nabla f(x_{k-1}))||^2 + \lambda||x||_1\}$$

Which can be written equivalently as,

$$x_k = \arg\min_x \{H'_{k-1}(x) \equiv \frac{1}{2t_k}||x - (x_{k-1} - t_k\nabla f(x_{k-1}))||^2 + \lambda||x||_1\} \tag{2.4}$$

ignoring the constant function $f(x_{k-1})$ in the optimization problem.

Using the definition of $\mathcal{P}roximal$ operator we can rewrite above equation 2.4 as ,

$$x_k = prox_{\lambda t_k}(x_{k-1} - t_k\nabla f(x_{k-1}))$$

because $prox_\lambda = \arg\min_{x_k}\{f(x_k) + \nabla f(x_{k-1})^T(x - x_{k-1}) + \frac{1}{2t_k\lambda}||x - x_{k-1}||^2\}$

## 2.3 Data fidelity Vs Penalty

As the problem is set up, the important components that need to be determined are the data fidelity term $f(x)$ and the penalty term $g(x)$. In this case, choices for both terms are flexible, but need to be reasonable. Furthermore, $g(x)$ must me convex but non-smooth. The traditional PGM methods (ISTA) tend to be quite slow on convergence, so an accelerated proximal gradient method will be used (FISTA).

The data fidelity term that is used is the squared L2 norm between the true image and the reconstructed estimate of the image at each iteration, symbolically:

$$f(x) = ||Ax - b||^2$$

This data fidelity term was chosen as it is strictly convex, continuously differentiable, and commonly used in various subfields of image processing such as denoising and deblurring.

The more intricate term to be chosen is the penalty term $g(x)$. There are many possible terms to choose from the literature, many of them include an L1 norm to ensure that the regularization term is at least non-smooth. Studies have shown that the L1-norm terms are better than other sparsity penalty terms at promoting the sharpness of the edges, an important factor to consider when reconstructing.

## 2.4 FISTA

### 2.4.1 Proximal Methods

Mainly these are solvers for the lasso and related methods.

This solver offers a good compromise between flexibility and performance and we can use it to solve all presented sparse regularization methods.

is able solve problems of the form

$$\min_{\alpha} F(\alpha) = f(\alpha) + g(\alpha), \tag{2.5}$$

i.e. minimizing functions that can be expressed as a sum of a convex, smooth function $f(\alpha)$ and another convex, possibly non-smooth function $g(\alpha)$. Lipschitz constant It is required that $f(\alpha)$ has a Lipschitz-continuous gradient. This means that the following condition holds for all possible vectors $x$ and $y$ for some positive constant $L$:

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|. \tag{2.6}$$

We call the smallest possible value of $L$ the Lipschitz constant.

For our goal given by 4.1 we set $f(\alpha)$ to

$$f(\alpha) = \frac{1}{2} \|\Phi\alpha - y\|_2^2 ,$$

such that the gradient of $f$ is given by

$$\nabla f(\alpha) = \Phi^T (\Phi\alpha - y) .$$

The Lipschitz constant for the gradient of $f$ is

$$L_{\nabla f} = (\sigma_{\max}(\Phi))^2, \tag{2.7}$$

where $\sigma_{\max}$ corresponds to the maximum singular value. Additionally we set $g$ to

$$g(\alpha) = n\lambda\mathcal{S}(\alpha) .$$

It is a regularized, smooth version of our function $g(\alpha)$ that has the same minimum as $g(\alpha)$. This implies that every point that minimizes $M_g(\alpha, \lambda)$ also minimizes our original function $g(\alpha)$. Finally, we are able to define the proximal operator that returns the infimum point of

$$M_g(\alpha, \lambda) = \inf_x \left\{ g(x) + (1/(2\lambda))\|x - \alpha\|_2^2 \right\} . \tag{2.8}$$

by

$$prox_g(\alpha\lambda) = \arg\min_x \left\{ g(x) + (1/(2\lambda))\|x - \alpha\|_2^2 \right\} . \tag{2.9}$$

This identity follows by rewriting 2.8 in terms of the proximal operator and calculating the gradient. We can use 2.9 to minimize $M_g(\alpha, \lambda)$ and thus also for optimizing $g(\alpha)$. In the most general case 2.9 would imply the need to solve a convex optimization problem. Fortunately we can find closed form solutions for many functions. Consider for example $g(\alpha) = 0$. In this case, the proximal operator is trivial and given by

$$\begin{aligned} &M_0(\alpha, \lambda) = \inf_x \left\{ 0 + 1/(2\lambda)\|\alpha - x\|_2^2 \right\} , \\ &prox_0 \alpha\lambda = \alpha. \end{aligned}$$

It is obvious that the minimum of $M_g(\alpha, \lambda)$ is equivalent to the minimum of $g(\alpha)$, the proximal operator corresponds to a gradient step on $M_g$.

We are now going to develop a minimizer for our composite goal that resembles a majorization-minimization algorithm. To do this, we first define an upper-bound of $F(\alpha)$ (*majorizing*) that we are then going to minimize (*minimization*).

We first give a regularized linearization of $f(\alpha)$ at an arbitrary but fixed point $y$ for an $L > 0$:

$$\hat{f}_L(\alpha, y) = f(y) + \langle \alpha - y, \nabla f(y) \rangle + L/2\|\alpha - y\|_2^2, \tag{2.10}$$

The first two terms are given by the first order Taylor expansion of $f(\alpha)$ at the point $y$, the last term can be interpreted as a trust-region or regularization that punishes large deviations from $y$. We then combine this linearization with our second function to archive an upper-bound of $F(\alpha)$:

$$Q_L(\alpha, y) = f(y) + \langle \alpha - y, \nabla f(y) \rangle + L/2\|\alpha - y\|_2^2 + g(\alpha). \tag{2.11}$$

We can see from 2.6 that $Q_L(\alpha, y)$ is an upper-bound of $F(\alpha)$ if L is equal to or greater than the Lipschitz constant of $\nabla f(\alpha)$.

The minimizer for this approximation is then given as the fixed-point equation

$$\begin{aligned} \pi_{g(\alpha)}(\alpha^*, L) &= \arg\min_x \{Q_L(x, \alpha^*)\} \\ &= prox_g \alpha^* - L^{-1}\nabla f(\alpha^*)L^{-1} \\ &= prox_g \alpha^* - L^{-1}\Phi^{(\Phi\alpha^* - y)}L^{-1}, \end{aligned} \tag{2.12}$$

where $\alpha^*$ denotes the optimal solution and $L$ is the Lipschitz constant of $\nabla f$ given by 2.7. In this equation $L$ is used to determine the optimal stepsize. This minimizer is called proximal gradient algorithm (or proximal-splitting) in the literature, because we first perform a gradient step on $f'_L(\alpha)$ given by 2.10 and then a proximal step on $g(\alpha)$. Using 2.12 repeatedly on a point will result in the fixed-point, i.e. the minimum of the upper bound, and thus also in the minimum of our original goal.

Iterative Shrinkage Tresholding Algorithm ()
**Require:** Lipschitz constant $L$ of $\nabla f$, regularization parameter $\lambda$

```
1: function ISTA($L, \alpha$)                              ▷ $\alpha$ is an initial guess
2:     while not converged do $\alpha\pi_{g(\alpha)}(\alpha, L)$
3:     end while
4:     return $\alpha$
5: end function
```

2.12 is all we need to define the simple iterative scheme called Iterative Shrinkage Tresholding Algorithm.

## 2.4.2 Proximal with Regularization

These proximal operators can be used to define a minimizer for a non-smooth function $g$. For example, combining 2.12 with the proximal operator for the lasso functional $g(\alpha) = n\lambda\|\alpha\|_1$ results in the minimizer

$$\begin{aligned} \pi_{n\lambda\|\alpha\|_1}(\alpha^*, L) &= prox_(\lambda\|\alpha\|_1\alpha^* - L^{-1}\nabla f(\alpha^*)L^{-1} \\ &= \left[\left(\alpha^* - L^{-1}\nabla f(\alpha^*)\right) - n\lambda L^{-1}\right]_+ - \left[-\left(\alpha^* - L^{-1}\nabla f(\alpha^*)\right) - n\lambda L^{-1}\right]_+, \end{aligned}$$

given as a fixed-point iteration. In this equation the function $[x]_+$ is applied element-wise on its input vector. We can then use this minimizer with 2.4.1 to compute a solution to the lasso problem.

ISTA always converges to the global maximum, but only does so linearly. To overcome this problem, Beck and Teboulle combined the algorithm with the accelerated gradient descent algorithm discovered by Nesterov. Nesterov's accelerated gradient descent is like to the vanilla gradient descent algorithm. The first step is identical, each following step carries some momentum of the step before, thus stabilizing the procedure.

It achieves quadratic convergence. This property is retained when combined with the proximal-splitting procedure 2.4.1, the result is called **FISTA**. Each step of **FISTA** evaluates the gradient and the proximal operator once, just as **ISTA** does.

Another problem with 2.4.1 is its dependence on the Lipschitz constant of $\nabla f$ to determine the optimal stepsize. For our choice of $f$, the best constant $L$ is given by 2.7. To avoid this expensive calculation, we use a backtracking line search to determine a suitable stepsize. In this line search procedure we use 2.11 as an upper bound for 4.1. We do this by iterating and finding the smallest $L$ for which 2.11 is an upper bound. This always results in the Lipschitz constant.

[] Linesearch **fista**

**Require:** $L > 0, \eta > 1, \alpha$

```
1: function LINESEARCH($\alpha, L$)
2:     $i \leftarrow 0$
3:     do
4:         L$\leftarrow \eta^i L$
5:         prox$\leftarrow \pi_\alpha(\alpha, L)$
6:         $i \leftarrow i + 1$
7: while $F(\text{prox}) < Q_L(\text{prox}, \alpha)$
8:     return prox and $L$                ▷ Also return prox to avoid duplicate calculations.
9: end function
```

We need to evaluate the line search once for each iteration step. It is possible that this procedure finds a non-optimal $L$, i.e. an $L$ that is larger than the Lipschitz constant. This leads to a smaller stepsize, which is not a problem in practice because our optimization procedure still converges, although slower than possible. We have to take that into consideration for our choice of linesearch parameters.

It is of course possible to use a constant stepsize like in 2.4.1. To do this, replace the line search with the minimal value of $L$ and calculate $\pi(\alpha, L)$ directly. We can also integrate the linesearch into the algorithm, by replacing the fixed $L$ with a call to the linesearch subroutine.

An alternative backtracking scheme for **FISTA**

[] Fast Iterative Shrinkage Tresholding Algorithm ()

**Require:** Initial guess for Lipschitz constant $L$ of $\nabla f$, regularization parameter $\lambda$

```
1: function FISTA($L, \alpha$)                                ▷ $\alpha$ is an initial guess for $\alpha^*$.
2:     $y \leftarrow \alpha$
3:     $t \leftarrow 1$
4:     while not converged do
5:         $\alpha_{\text{before}} \leftarrow \alpha$
6:         $\alpha, L \leftarrow$ LINESEARCH( y, L)        ▷ Linesearch returns $\pi_L(y)$ and the used L.
7:         $t_{\text{before}} \leftarrow t$
8:         $t \leftarrow \frac{1}{2}(1 + \sqrt{1 + 4t^2})$
9:         $y \leftarrow \alpha + (t_{\text{before}} - 1)\, t^{-1}\, (\alpha - \alpha_{\text{before}})$
10:    end while
11:    return $\alpha$
12: end function
```

# 3 Deblurring and denoising

For many inverse problem applications the least squares approximation,

$$\hat{x}_{LS} = \arg\min_{x} \|Ax - b\|_F^2, \tag{3.1}$$

gives a reasonable solution. This approach doesn't provide any new information for the denoising problem, and for image deblurring problems $A$ is often ill-conditioned, so the solution is contaminated by round-off error and amplified noise .

In order to stabilize the solution, a variety of regularizers can be used which exploit features of the true image such as smoothness or sparsity in a wavelet domain. This results in the problem formulation

$$\hat{x} = \arg\min_{x} \|Ax - b\|_F^2 + \lambda R(x), \tag{3.2}$$

where the parameter $\lambda > 0$ is chosen to balance the tradeoff between fidelity to the model and the assumed feature. In this project we compare two choices for the regularization term: $l_1$ wavelet regularization with $R(x) = \|Wx\|_1$, and total variation regularization with $R(x) = TV(x)$.

## 3.1 Problem Formulation

The general approach to the image deblurring and denoising problem can now be stated as a minimization problem of the form

$$\min_{x} f(Ax - b) + \lambda R(x), \tag{3.3}$$

where $Ax$ is a discrete convolution of the true image with a blur kernel, the *fidelity term* $f(Ax - b)$ measures how well the recovered image complies with the linear model, and $R(x)$ is our chosen *regularization.* Furthermore, if we assume that the pixels of our image satisfy $x_{i,j} \in [0, 1]$, we can add an additional term to the objective function

$$L_b(x) = f(Ax - b) + \lambda R(x) + \delta(x|[0, 1]), \tag{3.4}$$

where $\delta(x|[0, 1])$ is the indicator function for the set $[0, 1]$.

## 3.2 $l_1$ Wavelet Regularization

The wavelet regularization deblurring model, as seen in **FISTA**, can be written in the form of (3.3) as

$$\min_x f(Ax - b) + \lambda W x_1, \tag{3.5}$$

where $W$ corresponds to a given wavelet transform and $\|Wx\|_1$ is the sum of absolute values of all entries of the matrix $Wx$.

### 3.2.1 Proximal of $l_1$ Wavelet Regularization

$R(x) = \|Wx\|_1$ is the soft-thresholding operation:

$$\begin{aligned} prox_{\alpha^{-1}\lambda\|W\cdot\|_1}(x) &= \arg\min_y \frac{1}{2}\|y - x\|_2^2 + \alpha^{-1}\lambda\|Wy\|_1 \\ &= \arg\min_y \frac{1}{2}\|Wy - Wx\|_2^2 + \alpha^{-1}\lambda\|Wy\|_1 \\ &= W^T \arg\min_{Wy} \frac{1}{2}\|Wy - Wx\|_2^2 + \alpha^{-1}\lambda\|Wy\|_1 \\ &= W^T prox_{\alpha^{-1}\lambda\|\cdot\|_1}(Wx) \\ &= W^T \operatorname{sgn}(Wx)\max(0, |Wx| - \alpha^{-1}\lambda) \end{aligned} \tag{3.6}$$

## 3.3 Total Variation Regularization

The total variation deblurring model, can be written in the form of (3.3) as

$$\min_x f(Ax - b) + \lambda \mathrm{TV}(x) \tag{3.7}$$

where $\mathrm{TV}(x)$ is the total variation semi-norm. And $f(Ax - b) = \frac{1}{2}||A(x) - b||_F^2$
Two choices exist for the TV-norm: the isotropic version and the $l_1$-based, anisotropic version. The $l_1$-based TV-norm, defined as

$$TV_{l_1}(x) = \sum_{i=1}^{m-1}\sum_{j=1}^{n-1}(x_{i,j} - x_{i+1,j} + x_{i,j} - x_{i,j+1}) + \sum_{i=1}^{m-1} x_{i,n} - x_{i+1,n} + \sum_{j=1}^{n-1} x_{m,j} - x_{m,j+1},$$

for $x \in^{m\times n}$, where reflexive boundary conditions

$$\begin{aligned} x_{m+1,j} - x_{m,j} &= 0, \ \text{for all } j \\ x_{i,n+1} - x_{i,n} &= 0, \ \text{for all } i \end{aligned}$$

are assumed.

Isotropic TV defined as:

$$\|X\|_{TV} = \sum_{i=1}^{n-1}\sum_{j=1}^{n-1}\sqrt{(X_{i,j} - X_{i+1,j})^2 + (X_{i,j} - X_{i,j+1})^2} or\ \|X\|_{TV} = \sum_{i=1}^{n-1}\sum_{j=1}^{n-1}|x_{l,j} - X_{i+1,j}| + |X_{i,j} - X_{i,j+1}|$$

### 3.3.1 First Attempt to Solve this Optimization

As the objective function is the sum of two convex functions (one smooth and one with a convenient proximal representation), we can employ the proximal gradient method. For each iteration we take a gradient step in $f$ and apply the proximal operator of $R$,

$$\begin{aligned} x_{k+1} &= prox_{\alpha^{-1}\lambda R}\big(x_k - \alpha^{-1}A^T\nabla f(Ax_k - b)\big) \\ &= \min_y \frac{1}{2}\Big\| y - \big(x_k - \alpha^{-1}A^T\nabla f(Ax_k - b)\big)\Big\|_2^2 + \alpha^{-1}\lambda R(y) \end{aligned} \tag{3.8}$$

Further we can use FISTA here, the Iterates are:

$$\begin{aligned} x_k &= prox_{\alpha^{-1}\lambda R}\big(y_k - \alpha^{-1}A^T\nabla f(Ay_k - b)\big) \\ t_{k+1} &= \frac{1+\sqrt{1+4t_k^2}}{2} \\ y_{k+1} &= x_k + \left(\frac{t_k - 1}{t_{k+1}}\right)(x_k - x_{k-1}) \end{aligned} \tag{3.9}$$

### 3.3.2 Total Variation Regularization Solver

Optimization of the objective function (3.7) is based on the proximal gradient algorithm. The proximal gradient step is given as follows,

$$\begin{aligned} x_{k+1} &= prox_{\alpha^{-1}(\lambda \mathrm{TV}(\cdot))}\big(\underbrace{x_k - \alpha^{-1}A^T\nabla f(Ax_k - b)}_{u_k}\big) \\ &= \arg\min_z \big(\|u_k - z\|_F^2 + \alpha^{-1}\lambda TV(z)\big)\big) \end{aligned}$$

Here the learning rate $\alpha$ is taken to be upper bounded by multiplicative inverse of the Lipschitz constant of $\nabla f$.
To evaluate this expression we must solve a denoising problem with input $u_k = x_k - \alpha^{-1}A^T\nabla f(Ax_k - b)$.

The resulting denoising problem with Frobenius norm fidelity function can be solved with speed. The $TV$ function is shown to have a dual representation as a trace and the relationship between trace and Frobenius norm is used to develop a dual formulation of the denoising problem. The method is:

$$\begin{aligned} &\mathcal{P} = \{(p,q) \in^{(m-1)\times n} \times^{m\times(n-1)} : p_{i,j} \le 1, p_{i,j} \le 1\} \\ &\mathcal{L} :^{(m-1)\times n} \times^{m\times(n-1)} \to^{m\times n} \text{ such that } \mathcal{L}(p,q)_{i,j} = p_{i,j} + q_{i,j} - p_{i-1,j} - q_{i,j-1} \\ &\quad \text{for } i = 1,\dots,m,\ j = 1,\dots,n \text{ and } p_{0,j} = p_{m,j} = q_{i,0} = q_{i,n} = 0 \end{aligned}$$

The total variation functional can be written as

$$TV(x) = \max_{(p,q)\in\mathcal{P}} (\mathcal{L}(p,q)^T x), \tag{3.10}$$

and we may now use the relationship between trace and Frobenius norm to obtain the dual problem for Frobenius denoising using total variation regularization.

$$\begin{aligned}
\min_{x\in[0,1]} x - b_F^2 + 2\lambda \mathrm{TV}(x) &= \min_{x\in[0,1]} \max_{(p,q)\in\mathcal{P}} x - b_F^2 + 2\lambda(\mathcal{L}(p,q)^T x) \\
&= \max_{(p,q)\in\mathcal{P}} \min_{x\in[0,1]} x - b_F^2 + 2\lambda(\mathcal{L}(p,q)^T x) \\
&= \max_{(p,q)\in\mathcal{P}} \min_{x\in[0,1]} x_F^2 + b_F^2 - 2(x^T b) + 2\lambda(\mathcal{L}(p,q)^T x) \\
&= \max_{(p,q)\in\mathcal{P}} \min_{x\in[0,1]} x_F^2 + b_F^2 - 2(x^T(b - \lambda\mathcal{L}(p,q))) \\
&= \max_{(p,q)\in\mathcal{P}} \min_{x\in[0,1]} \underbrace{x_F^2 + b - \lambda\mathcal{L}(p,q)_F^2 - 2(x^T(b - \lambda\mathcal{L}(p,q)))}_{=x-(b-\lambda\mathcal{L}(p,q))_F^2} - b - \lambda\mathcal{L}(p,q)_F^2 + b_F^2 \\
&= \max_{(p,q)\in\mathcal{P}} \min_{x\in[0,1]} x - (b - \lambda\mathcal{L}(p,q))_F^2 - b - \lambda\mathcal{L}(p,q)_F^2 + b_F^2
\end{aligned}$$

The interchanging of min and max operators is permitted due to the convexity of the objective in $x$ and concavity in $p$ and $q$ .The problem of maximizing over $x$ is the projection of each index of $b - \lambda\mathcal{L}(p,q)$ onto the set $[0,1]$. This gives the optimality condition for $x$ in terms of $p$ and $q$ which we put back in to obtain the dual problem.

$$\begin{aligned}
(p^*, q^*) &= \arg \max_{(p,q)\in\mathcal{P}} P_{[0,1]}(b - \lambda\mathcal{L}(p,q)) - (b - \lambda\mathcal{L}(p,q))_F^2 - b - \lambda\mathcal{L}(p,q)_F^2 + b_F^2 \\
x &= P_{[0,1]}(b - \lambda\mathcal{L}(p^*, q^*))
\end{aligned}$$

The dual problem has the objective to be Lipschitz differentiable. The objective is smooth and we can use projected gradient algorithm.

## 3.4 Algorithms

### 3.4.1 FGP

Total variation has also been identified as a sparsity promoting penalty term. The motivation behind this regularization is because it has been noted to be appropriate for denoising and deblurring . The objective function can then be summarized as:

F(x) = $|| \; Ax - b \; ||^2 \; + 2\lambda TV(x)$

The first step towards solving this objective by improved means is a FAST Gradient Projection method. Now we have a TV term in the objective of our function and that is difficult to minimize.
Chambolle introduced the idea of solving it by solving the dual of the problem.
**The dual in unconstrained case is convex quadratic programming ( (maximization of a concave quadratic function subject to linear constraints)**
Beck and Teboulle's solution to the total variation regularization problem involves segments of pseudocode. The first component, fast gradient projection (FGP), is for solving the pure denoising case of the problem where $A \equiv I$. The following are definitions required for the FGP algorithm:

When $A \equiv I$ we consider constrained minimization problems of the form

$$\min_{x \in C} ||x - b||^2 + 2\lambda(x) \tag{3.11}$$

with an extra constraint for $x$ to lie in closed convex set $C \subset^{m \times n}$ with $C = B_{l,u} = \{x : l \leq x_{ij} \leq u, \; \forall i, j\}$ where $l = 0, \; u = 1$. This is specially the case of images that they have to lie between 0-255 or 0-1. Thus $B_{l,u}$ is cube.

To construct a dual of the constrained problem, we introduce the dual variables $(p, q)$ as elements of $\mathcal{P}$ which is a set of matrix-pairs $(p, q)$ with $p \in^{(m-1)\times n}$ and $q \in^{m \times (n-1)}$ satisfying

$$p_{i,j}^2 + q_{i,j}^2 \leq 1 \quad i = 1, \ldots, m-1; \; j = 1, \ldots, n-1 \tag{3.12}$$
$$p_{i,j} \leq 1 \quad i = 1, \ldots, m-1 \tag{3.13}$$
$$q_{i,j} \leq 1 \quad j = 1, \ldots, n-1 \tag{3.14}$$

while the entries are given by

$$p_{i,j} = x_{i,j} - x_{i+1,j}, \quad i = 1, \ldots, m-1; \; j = 1, \ldots, n \tag{3.15}$$
$$q_{i,j} = x_{i,j} - x_{i,j+1}, \quad i = 1, \ldots, m; \; j = 1, \ldots, n-1 \tag{3.16}$$

We have to introduce 4 operators.
**A. $\mathcal{L}$:**
The linear operator $\mathcal{L} : R^{(m-1)\times n} \times R^{m \times (n-1)} \mapsto R^{m \times n}$ is defined by

$$\mathcal{L} : (p, q)_{i,j} = p_{i,j} + q_{i,j} - p_{i-1,j} - q_{i,j-1} \; \forall i = 1, \ldots, m; \; j = 1, \ldots, n \tag{3.17}$$

$\boldsymbol{\mathcal{L}^T}$:

**B.**$\mathcal{L}^T : R^{m\times n} \mapsto R^{(m-1)\times n} \times R^{m\times(n-1)}$ represents the discrete gradient operator $G = h$ applying the first order forward differences in both dimensions to the stacked coloumns vector $x$. The mapping is given by

$$\mathcal{L}^T(x) = (p, q) \tag{3.18}$$

In the code $p$ corresponds to $px$, $q$ coresponds to $py$ and $(p, q)$ corresponds to $p1$. $px$, $py$ and $p1$ are each vectors of stacked coloumns of the image.

**C.** $\boldsymbol{P_C}$ : $R^{m\times n} \mapsto R^{m\times n}$:

Next we consider the orthogonal projection operator $P_C$ on the set $C = B_{l,u}$

$$P_C(x)_{ij} = P_{B_{l,u}}(x)_{ij} = \begin{cases} l & x_{ij} < l \\ x_{ij} & l \leq x_{ij} \leq u \\ u & x_{ij} > u \end{cases} \tag{3.19}$$

The nonsmooth convex optimization model $\min_x f(x)+g(x)$ with $g$ being a proper closed convex function and $f$ a continously Lipschitz differentiable function.
The standard smooth convex constrained optimization problem $\min_{x\in C} f(x)$ is obtained by choosing $g(x) \equiv \delta_C$ with $C = B_{l,u} =\subset^{m\times n}$ being some closed convex set.
The proximal mapping of $g = \delta_C$ is then given by $prox_t(g) = prox_t(\delta_C) = (I+t\partial\delta_C)^{-1} = P_C = P_{B_{l,u}}$ .
The optimality condition for $x^*$ solving the convex minimization problem is given by $0 \in tf(x) + t\partial g(x^*)$ that is equivalent to $x^* = (I + t\partial g)^{-1}(I - tf)(x^*)$.
This gives the GP: $x_k = P_C(x_{k-1} - t_k f(x_{k-1}))$.

Our main goal is to solve for the objective $\min_{x\in C} ||x - b||^2 + 2\lambda(x)$.
**D.** $\boldsymbol{P_p}$ : $R^{(m-1)\times n}, R^{(m)\times(n-1)} \mapsto R^{(m-1)\times n}, R^{(m)\times(n-1)}$:
**For the projection step is given by:** There is another projection operator defined $P_{\mathcal{P}}$ on $\mathcal{P}$ given by

$$P_{\mathcal{P}}(p, q) = (r, s) \tag{3.20}$$

where $r \in R^{(m-1)\times n}$ and $s \in R^{m\times(n-1)}$ are given by

$$r_{ij} = \begin{cases} \frac{p_{ij}}{\max\{1,\sqrt{p_{ij}^2+q_{ij}^2}\}} & i = 1,\dots,m-1;\ j = 1,\dots,n-1 \\ \frac{p_{in}}{\max\{1,p_{in}\}} & i = 1,\dots,m-1 \end{cases} \tag{3.21}$$

$$s_{ij} = \begin{cases} \frac{q_{ij}}{\max\{1,\sqrt{p_{ij}^2+q_{ij}^2}\}} & i = 1,\dots,m-1;\ j = 1,\dots,n-1 \\ \frac{q_{mj}}{\max\{1,q_{mj}\}} & j = 1,\dots,n-1 \end{cases} \tag{3.22}$$

The operators were implemented as $\textbf{operator}_L.m, operator_{LA}djoint.m, operator_projection_{o}n_{s}et_C.mandoperator$

**The dual problem that is convex and contains a continously differentiable objective and the gradient of the objective is given by $\vec{x}^* = P_C\left[b - \lambda\mathcal{L}(p_N, q_N)\right]$. The Fast Gradient Projection algorithm FGP$(b, \lambda, N)$ as:**

```
FGP$(b, \lambda, N)$
Input :
b - observed image
$\lambda$ - regularization parameter
N - Number of Iterations
Output :
  $(r^1_{ij}, s^1_{ij}) = (p^0_{ij}, q^0_{ij}) = 0;\ t_1 = 1$
  for $k = 1 : N$ do
      $(p_k, q_k) = P_{\mathcal{P}}\left((r_k, s_k) - \frac{\mathcal{L}^T P_{[0,1]}(b - \lambda\mathcal{L}(r_k, s_k))}{8\lambda}\right)$
      $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$
      $(r_k, s_k) = (p_k, q_k) + \frac{t_k - 1}{t_{k+1}}(p_k - p_{k-1}, q_k - q_{k-1})$
  end for
  return Set x$^*$ =$\mathbf{P}_{[0,1]}(b - \lambda\mathcal{L}(p_N, q_N))$
```

**As the denoising problem has been defined above, the solution to the deblurring problem where the operator$H$ is no longer the identity matrix but our forward operator can be then defined as the following MFISTA problem**

### 3.4.2 MFISTA

**The optimization is implemented by using a variant of FISTA called Monotone FISTA or MFISTA. In FISTA, the objective function values are not guaranteed to be nonincreasing, so MFISTA requires the evaluation of the objective function to check for monotonicity. The sped-up projected gradient algorithm is implemened using the same momentum technique as in FISTA.**

```
**MFISTA**$(b, f, \lambda)$
    $y_1 = x_0 = b$; $t_1 = 1$
    $\alpha \geq Lip(\nabla f)$
    for $k = 1 : N$ do
        $u_k = y^k - \frac{A^T \nabla f(Ay_k - b)}{\alpha}$
        $z_k = FGP(u_k, \frac{\lambda}{2\alpha})$
        $x_k = \underset{x \in \{x_{k-1}, z_k\}}{\textbf{argmin}}\ F(x)$
        $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$
        $y_{k+1} = x^k + \frac{t_k}{t_{k+1}}(z_k - x_k)$
                $+ \frac{t_{k-1}}{t_{k+1}}(z_k - x_k)$
    end for
    return $x_N$
```

**An iteration number of 10 was chosen for the FGP algorithm (pure denoising component) as Beck and Teboulle suggested in their coding of the algorithm. n was chosen to be 2 in the FISTA algorithm as n has be larger than 1, and doubling the constant, $L_p$, seemed like a reasonable method to have relatively fast fulfillment of the inequality in substep 1. $L_{p0}$ was chosen to be 0.005 through trial and error. Different values were chosen until the first iteration of MFISTA had a $F(p_{Lp_0}(y_k)) > Q_{Lp}(p_{Lp_0}(y_k), y_k)$, which results in an increase in L per the algorithm's $L_p = \eta^{i_k} L_{p_{k-1}}$ update rule. This ensured that $L_p$ for all iterations are appropriate in size as $L_{p_1}$ used in the first iteration was calculated from an $L_{p_0}$ that did not satisfy the inequality. $\lambda$ was chosen as 0.1 . Note that there is a dot product term in the definition of $Q_{LP}(x, y)$.This dot product term in image reconstruction literature usually involves vectorized forms of both x and y. However, in this case, the sum of the vector output of the dot product function in MatLab with these matrices as the input, sum(dot(x,y)), also works, as the dot product function in MatLab calculates the inner products of the column vectors if the inputs are matrices.**

# 4 Experimentation

## 4.1 FISTA vs ISTA

The paper does transform the Images using wavelet transforms (possibly to a sparse representation) and also performs operations like Blurring using Suitable Transformations (DCT) and thus few benefits of that approach are:

- The computation becomes fast, for example convolution is just a multiplication.
- L1 regularization in the problem statement prefers Sparse Signals (images) and thus using suitable wavelet transform favours the L1 norm in objective function.

But there are a few disadvantages too:

- The Matrices in the problem statement are not explicit as we are using the blurring operation and not just a matrix operation
- Applying vanilla algorithms (FISTA/ISTA) and getting the intuition gets hindered as several other factors like the parameters if transformations, etc gets to play a role
- Several spurious effects are introduced by having optimized operations in Transformed domain (like ringing effect based on type of Boundary Condition in Blurring)

Thus we decided to not use transformations like DCT or the wavelets. And have Convolution as Matrix-Vector Multiplication.

Matrix-vector operation based on lexicographic notation is inefficient in terms of memory usage and speed. If the size of 2D image is (N,N), and that of blur kernel is (K,K), then the vectorized image will be of $N^2x1$, and the matrix used as Blur Matrix (A) of $N^2, N^2$.
Inverse of the matrix will take forever for any normal size images. Fortunately, convolution operation of image data and blur kernel is equivalent to the multiplication of the matrix and the vector in **lexicographic manner**.
The convolution operation doesn't require large memory space (only $N^2$, much less than $N^4$) and faster (since $M \ll K$).

ISTA is a first-order method which is gradient-based so it is simple and efficient. However, its convergence is slow - O(1/k). A fast ISTA (FISTA) is developed for faster convergence, which gives an improved complexity, $\mathcal{O}(\frac{1}{(k^2)})$.

### 4.1.1 The Objective Function

$$\hat{x} = argmin_x ||Ax - b||^2 + \lambda ||x||_1 \tag{4.1}$$

In the paper the Degradation Model, i.e. the matrix **A** is chosen as $\mathbf{A} = \mathbf{RW}$ where **R** is the blurring matrix and **W** contains a wavelet basis (i.e., multiplying by **W** corresponds to performing inverse wavelet transform).
We assumed the degradation model as :

$$b = Ax = RWx = Rx$$

because we asssume $\mathbf{A} = \mathbf{RW}$, the matrix corresponding to wavelet transform is $\mathbf{W} = \mathbf{I}$ in our assumed case.
Thus the Deconvolution problem becomes as in the equation 4.1.

#### Deformation to Image

The image went through a Gaussian blur of size 9  9 and standard deviation 4 (applied by the MATLAB function fspecial) followed by an additive zero-mean white Gaussian noise with standard deviation $10^3$ .

We create the matrix **A** corresponding to the blur convolution.

we know the degraded image, the blur kernel as big matrix we are solving for the unknown image, $x$. (Note this image is now vectorized).

### 4.1.2 Parameters for Iterative Algorithms

The value of $\lambda$, the regularization parameter $= 10^{-2}$
In the implementation of the FISTA and ISTA in code, there is a provision for backtracking when we don't know the Lipschitz constant. But in these examples we can directly get the lipschitz constant from the objective function.

So, here we just use FISTA and ISTA when Lipschitz constant is known.
We first run Both the Algorithms for 100 iterations. And then again Both for 1000 iterations.
We note :

- Cost Function
- Root Mean Square Error
- PSNR

in each iteration and we plot to observe the algorithm.

## 4.2 Observations

We take a small cameraman image

- First thing to observe in that the cost function in each successive iteration the cost function is always non-increasing in both the algorithms FISTA and ISTA.
  This means that each new point in domain found by the iterative algorithms is better than any of the previous point.
  **But it turns out for specific example, in general ISTA is guaranteed to have function values as non-increasing but FISTA does not guarantee this property.**
- Time taken in each iteration of both the algorithms are nearly same,but FISTA is slightly slower than ISTA.
- Plotting the Restored images, it is observed that in same number of iteration, FISTA restored image is better looking than ISTA.
- Further the PSNR of FISTA is always greater than ISTA.
- **Cost Function:** In the plot of cost function, we can observe that the graph of FISTA is always below the graph of ISTA.
  And also there is a sharp dip initially in the graph of FISTA, signifying that it attains the minimum value of cost function very fast.
  For this image, not much difference is observed in cost function graph due to increase in number of iterations as after a point both graphs co-incide. Signifying the minimal value of objective function is same in both the algorithms.
- **PSNR :** PSNR of FISTA iterates are significantly larger than PSNR of ISTA iterates, signifying the faster converge of FISTA.
  **Also, it shold be noted that PSNR of FISTA after 100 iterations is already greater than PSNR of ISTA after 1000 iterations.**
- **RMSE :** RMSE shows almost similar trend so as to PSNR.
  **Again RMSE error that FISTA attains after only 100 iterations is better than RMSE of ISTA after 1000 iterations.**
  The rate of decrease in RMSE is much greater in FISTA curve signifying the faster rate of convergence of it's iterates.

### 4.2.1 Numerical Values

**ISTA**

- Data after 100 iterations
  - **Time for 100 iterations** = 3.25s
  - **PSNR after 100 iterations** = 22.93
- Data after 1000 iterations
  - **Time for 100 iterations** = 32.15s
  - **PSNR after 100 iterations** = 25.81

**FISTA**

- Data after 100 iterations
  - **Time for 100 iterations** = 31.73
  - **PSNR after 100 iterations** = 33.94

## 4.3 GP vs FGP

TV L1 is the TV operator that is implemented in the cod section.

We take the cameraman image and apply noise : **zero mean, standard deviation = 0.1** to it. We have $\lambda = 0.1$ with both GP and FGP methods. The finding of the results are shown in the figure.

- Figure 5 shows the comparison of applying the FGP and GP methods and we can observe that both the methods on visual inspection looks similar. This may give us assumption that there is not much separation on how the two algorithms progress. That is, the difference in the progress of two algorithms is similar in this setting.
- **Function value :** Figure 6 shows the graph of function value at each iteration of FGP and GP. There are several most interesting observations there
  - FGP curve is below GP curve saying that FGP converges faster than GP in simulations.
  - The non-smppthness in both the curves occur at similar points, suggesting both are encountering the same behaviour of points along the iteration.
- $f(x_k) - f(x_convergence)$ shown in Figure 7:

  - It shows that FGP is faster than GP
  - **Most important observation here is the increase seen in FGP curve at iteration after 5. Thus FGP is not monotone. Thus, convergence is faster but not that successive iterates are better.**

- **Relative difference at each iterate :**$\frac{f(x_k)-f(x_{k-1}}{f(x_{k-1})}$ Figure 8 also confirms better convergence of FGP than GP.
- **PSNR and RMSE at each iterate :** Figure 9, 10 shows PSNR and RMSE resp. of both FGP and GP along with iterations. This follows same trens of FGP superiority.

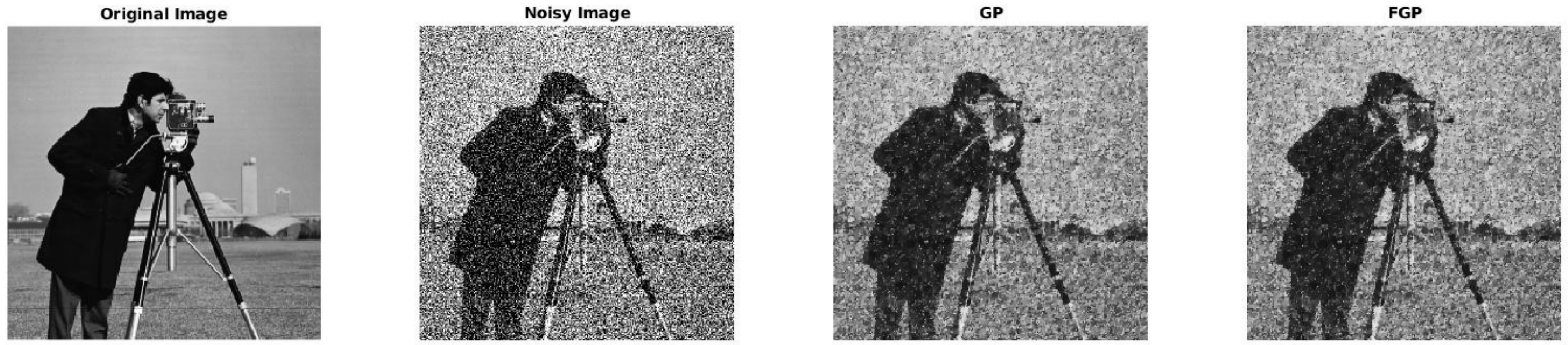

Figure 5: Images after denoising by FGP and GP

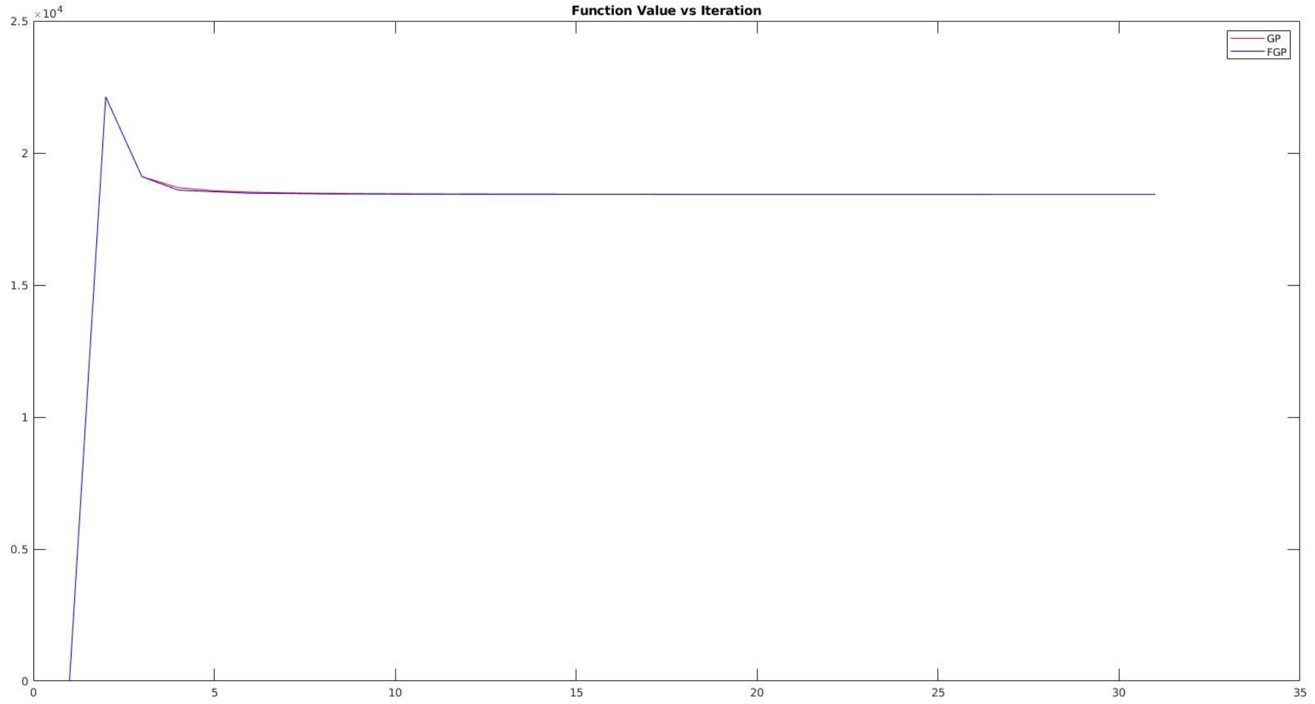

Figure 6: Function value per iteration by FGP and GP

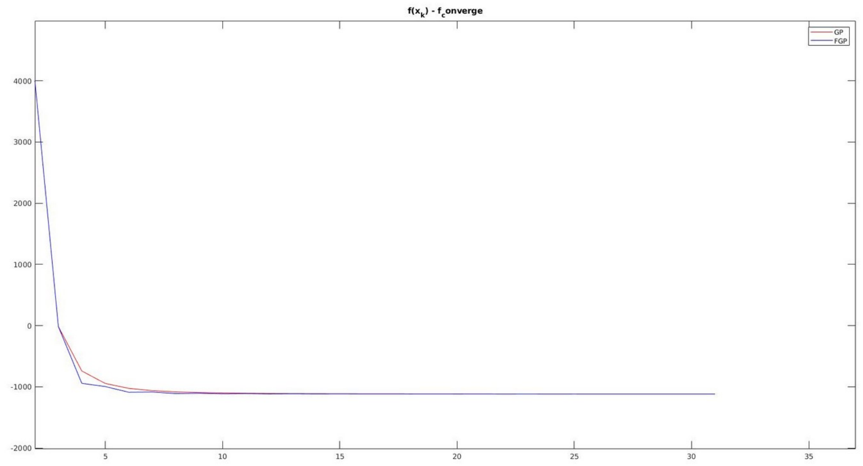

Figure 7: f(x_k)-f(convergence) per iteration by FGP and GP

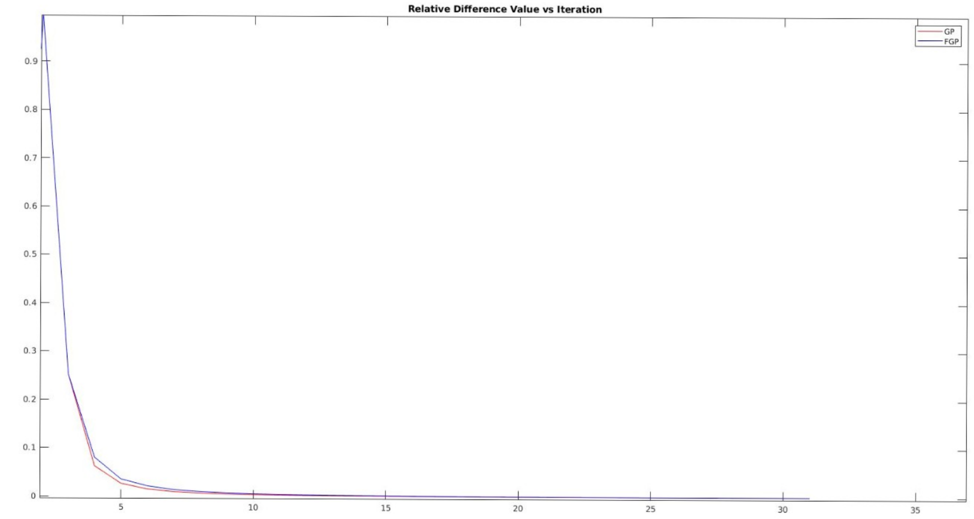

Figure 8: f(x_k)-f(x_k-1)/f(x_k-1) per iteration by FGP and GP

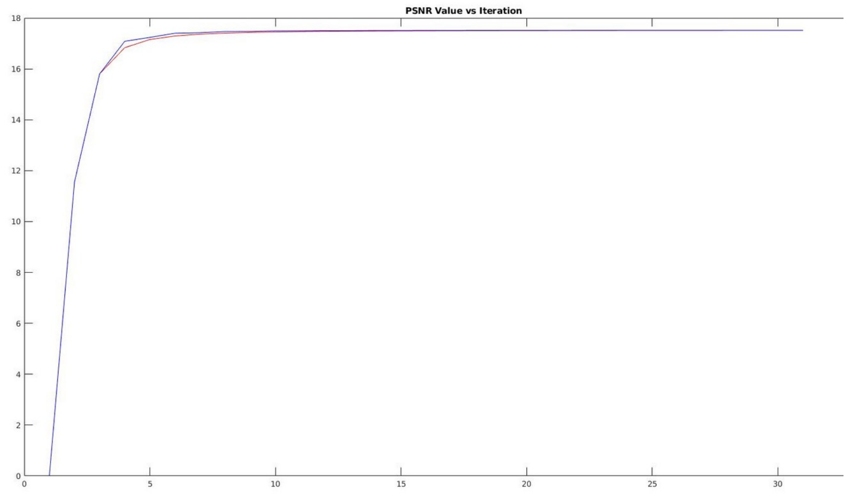

Figure 9: PSNR per iteration by FGP and GP

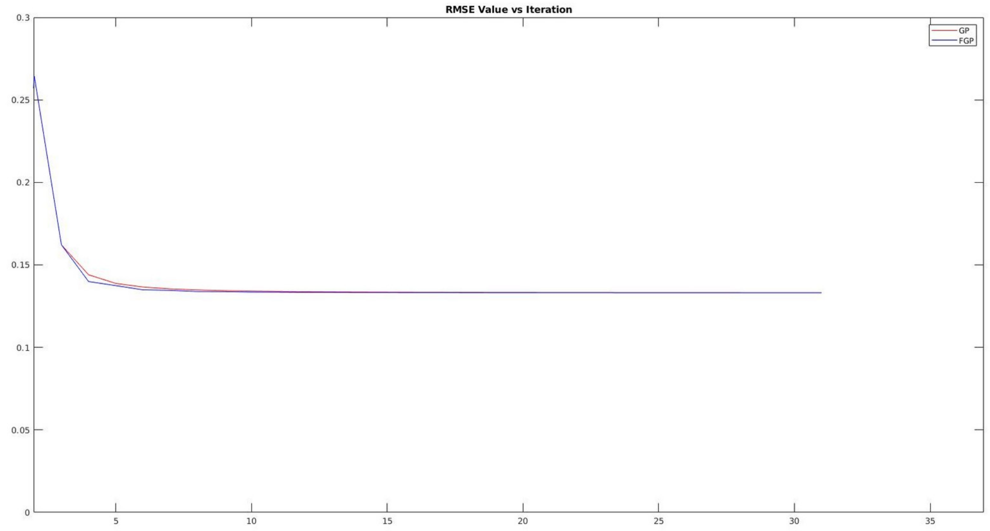

Figure 10: f(x_k)-f(x_k-1)/f(x_k-1) per iteration by FGP and GP