Analysis of Some Optimization Techniques for Regularization of Inverse Problems.

Viateur IRAKARAMA

College of Science and Technology

School of Science

Department of Mathematics

Master of Science

2016

Analysis of Some Optimization Techniques for Regularization of Inverse Problems.

By

Viateur IRAKARAMA

PG 214003440

A dissertation submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In the College of Science and Technology

Supervisor: Dr. Japhet Niyobuhungiro

Huye, October 2016

Declaration

I declare that, this Dissertation contains my own work except where specifically acknowledged and it has not been previously submitted for any comparable academic award.

Viateur IRAKARAMA,

PG 214003440

Signed

Huye, October 9, 2016

Dedication

To my mother, brothers and sisters, all my family members, friends and classmates this thesis is dedicated.

Acknowledgments

First of all, I thank the Almighty God for his protection of my life. I am very grateful to my supervisor, Dr. Japhet Niyobuhungiro, for his excellent and patient guidance and support during my thesis work. I would like to express my sincere gratitude to the teaching staff in the master's program of Applied Mathematics. The realization of this work is a result of their combined efforts in one way or another. I wish to thank all staff of the University of Rwanda and the entire College of Science and Technology, and particularly the Department of Mathematics for the education that I have received. Sincerely, I would like to thank the Coordinator of Master's Program, Dr. Minani Froduald for all his efforts to ensure the smooth running of this program. I wish to thank my colleagues and classmates for their support and friendship which provided constant comfort. Finally, I express my deepest thanks to my beloved mother, brothers and sisters for their unconditional love and support during my entire life.

Viateur IRAKARAMA

Abstract

The main objective in inverse problems is to approximate some unknown parameters or attributes of interest, given some measurements that are only indirectly related to these parameters. This type of problem appears in many areas of science, engineering and industry. Examples can be found in medical computerized tomography, groundwater flow modeling, etc. In the process of solving these problems often appears an instability phenomenon known as ill-posedness which requires regularization. Ill-posedness is related to the fact that the presence of even a small amount of noise in the data can lead to enormous errors in the approximated solution. Different regularization techniques have been proposed in the literature. In this thesis our focus is put on Total Variation regularization. We study the total variation regularization for both image denoising and image deblurring problems.

Three algorithms for total variation regularization will be analysed, namely the split Bregman algorithms, the Alternating Direction Method of Multipliers and the Rudin Osher Fatemi denoising model on the graph. We experiment these algorithms for different implementation examples and compare their performance for denoising problems.

Our observation is that these algorithms are comparable in many cases, often times the Split Bregman algorithm is faster in the sense that it achieves a given number of iterations in a shorter running time, but at the same time even though the ROF model on the graph seems slower, it achieves a desired or a prescribed precision with fewer number of iterations.

Key words and phrases

Convex optimization algorithms; regularization of inverse problem; Signals and images; Total variation deconvolution; Total variation denoising.

List of symbols and acronyms

TV : Total Variation

ROF : Rudin, Osher and Fatemi

BV : Bounded Variation

SVD : Singular Value Decomposition

PSF : Point Spread Function

ADMM : Alternating Direction Method of Multipliers

SB : Split Bregman

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

In a wide variety of important applications in science, engineering and industry appears a so-called inverse problem. The goal in inverse problem is to estimate some unknown attributes of interest, given measurements that are only indirectly related to these attributes. Some examples are medical computerized tomography where the goal is to image structures within the body from measurements of X-rays that have passed through the body, also in some engineering applications like groundwater flow modeling, one wishes to estimate the material parameters of an aquifer from some measurements of for example pressure of a fluid that immerses the aquifer. Unfortunately however, the presence of even a small amount of noise in the data can lead to enormous errors in the estimates. In the literature, this instability phenomenon is known as ill-posedness and to address it, it is used a Mathematical technique called regularization. Several regularization techniques have been proposed in literature. In this thesis we analyze three optimization algorithms for total variation regularization. The definitions and concepts used in this chapter are given in Chapter 2.

Problem Formulation

We consider some application problems which can be written as the constrained optimization problem posed as follows

$$\inf_u E(u) \text{ subject to } H(u) = 0, \tag{1.1}$$

where, $E(u)$ is the convex regularizing functional and $H(u)$ is a convex differentiable functional.
The above problem is usually approached by converting it into unconstrained optimization problem with penalty function as follows

$$\inf_u \Big(E(u) + \lambda H(u) \Big),$$

where $\lambda > 0$ is a penalty parameter (also often referred to as regularization parameter).
We consider a theoretical analysis of this problem and consider three algorithms for its resolution for specific functionals E and H with applications to the problem of reconstruction of sparse signals and total variation denoising and deblurring.

Literature Survey

Regularization of inverse problems is now a very established and active area of research with a rich literature. Here we only aim at giving some of our main references. Curtis R. Vogel (2002) in [23] elaborated that this kind of problem is applied in many areas such as medical imaging, image processing, signal processing, in engineering and technology, etc. and studied, among others, the variational regularization methods for the solution of inverse problems. An important algorithms named Split Bregman to solve L1-Regularization problems applied to total variation denoising problem and compressed sensing problem. This method is very easy and fast to solve these kind of problems and this leads to be interesting method for total variation denoising and deconvolution for both Anisotropic and Isotropic total variations. It is presented in [11]. See also Yuying Shi et.al (2013) in [19] and Pascal Getreuer (2012) in [8] and references there in where the function considered is assumed to be smooth. Tom G. and Stanley O. (2009) in [11] uses the split Bregman method. We also consider the Alternating Direction Method of Multipliers whose details in [21] and a recent Rudin Osher Fatemi Denoising model on the graph proposed in [13].

1.2 Objectives

The main objectives of this Master's project are the following:

1. Study the three specific application problems, namely: The reconstruction of the sparse signals, the total variation denoising and the total variation deblurring. In addition, present them in the context of optimization.

2. Study, implement and experiment three algorithms: The Split Bregman, ADMM and the ROF model on the graph algorithms.

3. Through experiments, analyze these methods and attempt to have an insight of how they are compared in performance.

1.3 Limitations

The thesis is focusing on the study of inverse problems, methods of regularization and applications. The inverse problem and its applications is a very large field of study and in this thesis we limited the study to three optimization algorithms, two of which, to our best knowledge are very famous and widely applied. In applications, we only focus on the reconstruction of sparse signals, total variation deconvolution and denoising of images.

1.4 Approach

During the progress of this thesis, the books for various authors and the documentation via internet are used for the collection of theories and definitions that are used to identify the problem of interest to be solved and to accomplish the tasks. The Matlab software is used for numerically implementing and experimenting the algorithm of sparse signal reconstruction, TV denoising and TV deconvolution for the selected examples of applications. We experiment several times for the results and it has been observed that the target of discovering the original signal and image for the selected examples is achieved. Finally this thesis is organized and written using the LaTeX software for reporting.

1.5 Structure of the thesis

This thesis is composed of five chapters and one appendix. In Chapter 1, we give a general introduction and background of the thesis. Chapter 2 provides preliminary theory, Chapter 3 presents our algorithms of interest and some experiments. In Chapter 4 the algorithms are analyzed and compared and some final remarks are presented in Chapter 5. In the appendix A, we presents some of the source codes used in experiments.

Chapter 2

Preliminaries.

2.1 Convex analysis and Optimization.

In this section, we focus on analysis and optimization problems in finite dimension space mostly in Euclidean space $\mathbb{R}^n$. In this context, the inner product $\langle x, y \rangle$ is given by the expression $\sum_{i=1}^{n} x_i y_i$ and for a space, the dual space is the same because $\mathbb{R}^n$ is a Hilbert space.

2.1.1 Convex sets.

A subset C of $\mathbb{R}^n$ is called convex if $\forall x, y \in C$, the line segment lies in C. That is $\alpha x + (1-\alpha) y \in C, \alpha \in [0,1]$ For construction of convex sets, below are a few standard operations that yield convex sets:

(a) Intersection: Let $\left\{C_i\right\}_{i \in I}$ be an arbitrary collection of convex sets. Then, their intersection $C = \bigcap_{i \in I} C_i$ is also convex.

(b) Cartesian product: Let $\left\{C_i\right\}_{i \in I}$ be an arbitrary collection of convex sets. Then, their cartesian product $C = \prod_{i \in I} C_i$ is also a convex, for more detail see [22], [2].

The following result holds for convexity in a vector space as shown in [2].

Proposition 2.1. *Let $A : \mathbb{R}^n \to \mathbb{R}^m$ be an affine mapping and $C \subset \mathbb{R}^n$ be convex. Then the image $A(C) = \{A(x) : x \in \mathbb{R}^n\} \subset \mathbb{R}^m$ is also convex. Conversely, if $D \subset \mathbb{R}^m$ is convex, then the inverse image $A^{-1}(D) = \{x \in \mathbb{R}^n : A(x) \in D\} \subset \mathbb{R}^n$ is also convex.*

Consequently, we have the following corollary.

Corollary 2.1.

(i) If C is convex, then λC is also convex for all $\lambda \in \mathbb{R}$.

(ii) Let C_1 and C_2 be convex. Then, their "Minkowski sum" $C_1 + C_2 = \{x + y : x \in C_1, y \in C_2\}$ is also convex.

(iii) The projection of convex set onto some of its coordinate is convex. That means that, if $C \subset \mathbb{R}^n \times \mathbb{R}^m$ is convex, then $C_1 = \{x_1 \in \mathbb{R}^n, (x_1, x_2) \in C \ \texttt{for some} \ x_2 \in \mathbb{R}^m\}$ is convex.

2.1.2 Convex functions.

Let C be a convex subset of $\mathbb{R}^n$. A function $f : C \longrightarrow \mathbb{R}$ is called convex if: $\forall x, y \in C, \alpha \in [0,1]$, then

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y).$$

Then, from the books of the authors in [22] and [2], we have the following two propositions for a differentiable function f.

Proposition 2.2. *Let C be a convex subset of $\mathbb{R}^n$ and let $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ be a differentiable function over $\mathbb{R}^n$.*

(a) *f is convex over C if and only if $f(y) \geq f(x) + (y-x)^T \nabla f(x)$, $\forall x, y \in C$.*

(b) *f is strictly convex over C if and only if the above inequality is strict for all $x \neq y$.*

Proposition 2.3. *Let C be a convex subset of $\mathbb{R}^n$ and let $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ be a twice continuously differentiable convex function over $\mathbb{R}^n$.*

(a) *If the Hessian matrix $(\nabla^2 f(x))$ is positive semi-definite $\forall x \in C$, then f is convex over C.*

(b) *If the Hessian matrix $(\nabla^2 f(x))$ is positive definite $\forall x \in C$, then f is strictly convex over C.*

(c) *If C is open and f is convex over C, then the Hessian matrix $(\nabla^2 f(x))$ is positive semi-definite $\forall x \in C$.*

But also, the convex function can be defined by its epigraph as shown in [22], then we have the following definition and Theorem.

Definition 2.1. *Let $X \subset \mathbb{R}^n$ be a nonempty set. Let $f : X \subset \mathbb{R} \longrightarrow \mathbb{R}$. The epigraph of $f \Longleftrightarrow epi f$ is a subset of $\mathbb{R}^{n+1}$ defined by $epi f = \left\{(x, \mu) : f(x) \leq \mu, x \in X, \mu \in \mathbb{R}\right\}$.*

Theorem 2.2. *Let $X \subset \mathbb{R}^n$ be a nonempty convex set. Let $f : X \in \mathbb{R}^n \longrightarrow \mathbb{R}$. Then f is convex if and only if $epi f$ is a convex set.*

For more detail and proof of the Theorem, see [22].

2.1.3 Subgradients and Subdifferential.

Let $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ be a convex function with $X \subset \mathbb{R}^n$ a Banach Space. We say that a vector $d \in \mathbb{R}^n$ is a *subgradient* of f at a point $x_0 \in X$ if

$$f(x) \geq f(x_0) + (x - x_0)^T d, \ \forall x \in X.$$

The set of all subgradients of a convex function f at x_0 is called the *Subdifferential* of f at x_0, and is denoted by $\partial f(x_0)$.

The subdifferential is always a closed convex set. The subdifferential $\partial f(x_0)$ is taken as the value at x_0 of the multifunction or set-valued map:

$\partial f : X \longrightarrow X^*$.

Its domain of definition is: $Dom\partial f = \{x_0 \in X : \partial f(x_0) \neq \emptyset\}$. For more detail, see [2].

2.1.4 Conjugate functions

Let $f : \mathbb{R}^n \longrightarrow \mathbb{R} \cup \{+\infty\}$. The *conjugate* function $f^* : \mathbb{R}^n \longrightarrow \mathbb{R}$ of the function f is defined by

$$f^*(y) = \sup_{x \in \mathbb{R}^n} \Big(y^T x - f(x) \Big), \tag{2.1}$$

where, f^* is closed, this is to mean that its epigraph is a closed set.

The domain of the conjugate function consists of $y \in \mathbb{R}^n$ for which the supremum is achieved or finite, that is to mean equivalently,

$$f^*(y) = \sup_{x \in \mathbb{R}^n} \Big(y^T x - f(x) \Big) = \max_{x \in \mathbb{R}^n} \Big(y^T x - f(x) \Big).$$

The conjugate function f^* is convex function as it is pointwise supremum of a family of linear and convex functions of y and this does not depend on whether f is convex or not, see the detail in [2], [3], [12].

If f is differentiable and convex, the supremum in x from (2.1) is found by differentiating with respect to x and we have

$$y = \nabla f(x), \tag{2.2}$$

where, $x^* = f^*(y)$ is a solution and it is the maximum if f is convex. If the solution x^* of (2.1) is unique, then from (2.1) and (2.2) we have

$$f^*(y) + f(x^*) = \nabla f(x^*)^T x^*,$$

where, the pair $(x^*, y) = (x^*, \nabla f(x^*))$ is said to be the Legendre conjugate pair, see [12].

2.1.5 Optimization problem

The general form of optimization problem is:

$$\begin{aligned} &\min_x f(x) \\ &s.t \quad x \in X, \end{aligned} \tag{2.3}$$

where $x \in \mathbb{R}^n$ is a decision variable, $f(x)$ an objective function and $X \subset \mathbb{R}^n$ a constraint set or feasible region.

Particularly, if the constraint set $X = \mathbb{R}^n$, the optimization problem (2.3) is called an *unconstrained optimization problem* which has the form:

$$\min_{x \in \mathbb{R}^n} f(x). \tag{2.4}$$

The general constrained optimization can be written as follows:

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} f(x) \\ &s.t \quad c_i(x) = 0, i \in E \\ &\qquad c_i(x) \geq 0, i \in I, \end{aligned} \tag{2.5}$$

where, E and I are the index set of equality and inequality constraints respectively and $c_i(x), (i \in E \cup I)$ are constraint functions.

If both objective function and constraint functions are *linear functions*, the problem is called *linear programming problem*. If at least one of the constraints or objective function is not linear, then it is called *nonlinear programming problem*.

The optimization problem (2.3) is said to be *Convex optimization problem* if the constraint $X \in \mathbb{R}^n$ and the objective function $f(x)$ are both convex, otherwise the problem is said to be *non-convex optimization problem*, see in [22],[2], [3].

2.1.6 Optimality conditions for unconstrained optimization problems

Consider the unconstrained optimization problem (2.4). It is given by

$$\min_{x\in\mathbb{R}^n} f(x).$$

We distinguish between two types of minimizers as shown in [22]. Namely *local minimizer* and *global minimizer* are defined below:

(1) A point x^* is called a *local minimizer* if there exists $\delta > 0$ such that $f(x^*) \leq f(x)$ for all $x \in \mathbb{R}^n$ satisfying $\|x - x^*\| < \delta$.

A point x^* is called a *strict local minimizer* if there exists $\delta > 0$ such that $f(x^*) < f(x)$ for all $x \in \mathbb{R}^n$ with $x \neq x^*$ and $\|x - x^*\| < \delta$, where $\|x - x^*\|$ is a norm in $\mathbb{R}^n$.

(2) A point x^* is called a *global minimizer* if $f(x^*) \leq f(x)$ for all $x \in \mathbb{R}^n$.

A point x^* is called a *strict global minimizer* if $f(x^*) < f(x)$ for all $x \in \mathbb{R}^n$ with $x \neq x^*$.

Theorem 2.3. *(First-order Necessary condition)*
Let $f : D \subset \mathbb{R}^n \longrightarrow \mathbb{R}$ be continuously differentiable on an open set D. If $x^* \in D$ is a local minimizer of (2.4), then $\nabla f(x^*) = 0$ see in [22].

Proof. Let x^* be a local minimizer and take any direction $d \in \mathbb{R}^n$ then $x^* + \alpha \in D$ for $\alpha > 0$ small enough. By the Tailor's expansion we have,

$$f(x^* + \alpha d) = f(x^*) + \nabla f(x^*)^T \alpha d + o(\alpha), \tag{2.6}$$

where $\nabla f = (\frac{\partial f}{\partial x_1}, \cdots, \frac{\partial f}{\partial x_n})$ and $\lim_{\alpha\to 0} \frac{o(\alpha)}{\alpha} = 0$. From (2.6), we have

$$f(x^* + \alpha d) - f(x^*) = \nabla f(x^*)^T \alpha d + o(\alpha) \geq 0,$$

dividing both sides by α and taking limit as $\alpha \to 0$, it follows that

$$\nabla f(x^*)^T d \geq 0.$$

Since this is true for any d, then replacing d by $-d$ we get:

$$\nabla f(x^*)^T d \leq 0$$

and therefore, we conclude that

$$\nabla f(x^*)^T d = 0$$

and since d is arbitrary, then $\nabla f(x^*) = 0$. □

Theorem 2.4. *(Second-order Necessary condition)*
Let $f : D \subset \mathbb{R}^n \longrightarrow R$ be twice continuously differentiable on an open set D. If $x^* \in D$ is a local minimizer of (2.4), then $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive semi-definite, see in [22].

Proof. Let x^* be a local minimizer and take any direction $d \in \mathbb{R}^n$ then $x^* + \alpha \in D$ for $\alpha > 0$ small enough. Since $f \in C^2$ and $\nabla f(x^*) = 0$, then by Taylor's expansion we have,

$$f(x^* + \alpha d) = f(x^*) + \frac{1}{2} d^T \nabla^2 f(x^*) d\alpha^2 + o(\alpha^2).$$

This equation leads equivalently:

$$f(x^* + \alpha d) = f(x^*) + \frac{1}{2} d^T \nabla^2 f(x^*) d\alpha^2 + 0(\alpha^2) \geq 0,$$

dividing both sides by α^2 and taking limit as $\alpha \to 0$ and $\lim_{\to 0} \frac{o(\alpha^2)}{\alpha^2}$, it follows that $\frac{1}{2} d^T \nabla^2 f(x^*) d\alpha^2 \geq 0$. As d is arbitrary, we have that $\nabla^2 f(x^*) \geq 0$ and then $\nabla^2 f(x^*)$ is positive semi definite. □

Theorem 2.5. *(Second-order Sufficient condition)*
Let $f : D \subset \mathbb{R}^n \longrightarrow \mathbb{R}$ be twice continuously differentiable on an open set D. If $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive definite,then $x^* \in D$ is a strict local minimizer of (2.4) , see in [22].

2.1.7 Duality

Consider the optimization problem:

$$\begin{aligned} &\min_{x \in R^n} f(x) \\ &s.t \quad c_i(x) \leq 0, i = 1, ..., m \\ &\qquad d_i(x) = 0, i = 1, ..., k \end{aligned} \tag{2.7}$$

As shown by Stephen Boyd and Lieven Vandenberghe (2009) in [3], the Lagrangian function associated to this problem is defined by:

$$\begin{aligned} L: \quad & \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^k \longrightarrow \mathbb{R} \\ & (x, \lambda, \alpha) \longrightarrow L(x, \lambda, \alpha) = f(x) + \sum_{i=1}^{m} \lambda_i c_i(x) + \sum_{i=1}^{k} \alpha_i d_i(x), \end{aligned} \tag{2.8}$$

where, λ_i and α_i are respectively the Lagrange multipliers associated with i^{th} inequality constraints $c_i(x) \leq 0$ and i^{th} equality constraints $d_i(x) = 0$.

The vectors $\lambda = (\lambda_1, \lambda_2, ..., \lambda_m)^T \in \mathbb{R}^m$ and $\alpha = (\alpha_1, \alpha_2, ..., \alpha_m)^T \in \mathbb{R}^k$ are called the dual variables or the Lagrange multiplier vectors.

The Lagrange dual function or simply dual function is defined by the minimum value of the Lagrangian function over x. That is

$$\begin{aligned} h: \quad & \mathbb{R}^m \times \mathbb{R}^k \longrightarrow \mathbb{R} \\ & (\lambda, \alpha) \rightarrow h(\lambda, \alpha) = \inf_{x \in D} \quad L(x, \lambda, \alpha), \end{aligned}$$

where, $D = \bigcap_{i=1}^{m} dom\, c_i \cap \bigcap_{i=1}^{k} dom\, d_i$.

The dual function takes on the value $-\infty$ when the Lagrangian is unbounded below in x.

Let $f(x^*)$ be the optimal value of the problem (2.7). For any $\lambda \geq 0$ and any $\alpha \geq 0$, we have

$$h(\lambda, \alpha) \leq f(x^*), \tag{2.9}$$

where, x^* is the feasible point that solves the problem (2.7).

To verify the (2.9) with a feasible point x^* such that $c_i(x) \leq 0$ and $d_i(x) = 0$, we have

$$\sum_{i=1}^{m} \lambda_i c_i(x) + \sum_{i=1}^{k} \alpha_i d_i(x) \leq 0.$$

This follows that

$$L(x^*, \lambda, \alpha) = f(x^*) + \sum_{i=1}^{m} \lambda_i c_i(x^*) + \sum_{i=1}^{k} \alpha_i d_i(x^*) \leq f(x^*).$$

Hence,

$$h(\lambda, \alpha) = \inf_{x \in D} \quad L(x, \lambda, \alpha) \leq L(x^*, \lambda, \alpha) \leq f(x^*.$$

Therefore, it follows that the inequality (2.9) holds for every feasible point x^* that solves (2.7) and the pair (λ, α) is referred as a dual feasible.

2.2 Function of Bounded Variation(BV)

In this section, we provide an introduction to the concept of total variation.

2.2.1 Discrete total variation case

The discrete one-dimensional total variation(TV) of a function f on the interval $[0,1]$ as defined in analysis is by definition:

$$TV(f) = \sup \sum_{i=1}^{n} \left| f_i - f_{i-1} \right|, \tag{2.10}$$

where, $f_i = f(x_i)$ and the supremum is taken over all partitions $0 = x_0 < x_1 < ... < x_n = 1$ of the interval, see[23] and [14].

2.2.2 General total variation case

If f is piecewise constant with a finite number of jump discontinuities, then $TV(f)$ gives the sum of magnitudes of the jumps.

If the function f is smooth(a function that has continuous derivatives of all orders everywhere in its Domain), we have

$$TV(f) = \sum_{i=1}^{n} \left| f_i - f_{i-1} \right| = \sum_{i=1}^{n} \left| \frac{f_i - f_{i-1}}{D_i} \right| D_i, \tag{2.11}$$

where $D_i = x_i - x_{i-1}, i = 1,2,...n$ and $f_i - f_{i-1} = f(x_i) - f(x_{i-1})$.

For the smooth function f, the sum in (2.11) approximates the L_1 norm of the derivative as a non quadratic function of f by taking the limit as the $D_i \to 0$ to obtain:

$$TV(f) = \int_0^1 \left| \frac{df}{dx} \right| dx. \tag{2.12}$$

In 2 dimensional space, the total variation of f is given by the generalization of (2.12):

$$TV(f) = \int_0^1 \int_0^1 |\nabla f| dx dy, \tag{2.13}$$

where, $\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)$ is the gradient and $|\nabla f| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$.
An extension for any function f (smooth or not) is the following:

$$TV(f) = \sup_{\overrightarrow{v} \in V} \int_0^1 \int_0^1 f(x,y) div\, \overrightarrow{v}\, dxdy, \tag{2.14}$$

where, $V = C_0^1([0,1] \times [0,1])$ consists of vector-valued functions $\overrightarrow{v} = \left(v_1(x,y), v_2(x,y)\right)$ with v_1 and v_2 are continuously differentiable and $div\, \overrightarrow{v} = \frac{\partial v_1}{\partial x} + \frac{\partial v_2}{\partial x}$ is the divergence of $\overrightarrow{v}$ and $|v| \leq 1$, see in [23], [14].

In $n-$ dimensional space, let an open set $\Omega \subset \mathbb{R}^n$. The total variation (or BV Semi-norm) $TV(f) \in [0, \infty[$ of $f \in L^1\Omega$is defined by

$$TV(f) = \|f\|_{TV(\Omega)} = \int_\Omega |\nabla f| dx = \sup_{\overrightarrow{v} \in V} \int_\Omega f\, div\, \overrightarrow{v}\, dx,$$

where, $V = \left\{ \overrightarrow{v} \in C_0^1(\Omega; \mathbb{R}^n) : |\overrightarrow{v}(x)| \leq 1,\ \forall x \in \Omega \right\}$.
The Space of functions of bounded variation on Ω is defined by:

$$BV(\Omega) = \left\{ f \in L^1(\Omega) : TV(f) < \infty \right\},$$

and therefore, the function f is said to be a *function of bounded variation*($f \in BV(\Omega)$) if and only if $TV(f) < \infty$ to mean that $TV(f)$ is finite. For more details, see [23] and [25]

Lemma 2.1. *Let the function $u : [a,b] \longrightarrow \mathbb{R}$ be piecewise, linear and continuous. Then*

$$\|u\|_{BV} = \sup_{g \in C_c^1((a,b)), \|g\|_\infty \leq 1} \int ug' = \sum_{i=1}^n \left| u(x_i) - u(x_{i-1}) \right|,$$

where $a = x_0 < x_1 < \cdots < x_n = b$ partitions $[a,b]$ into sub-intervals $[x_i, x_{i+1}]$ such that u is linear on each sub-interval.

In the following two subsections, we define two types of total variation in two dimensional space referred to as the Anisotropic and Isotropic total variations.

2.2.3 Anisotropic Total Variation

The *Anisotropic total variation* for a smooth function f in two dimensional space is defined in [11, 19, 8] by:

$$\begin{aligned}\|f\|_{TV} &= \int_\Omega |\nabla f| dxdy = \int_\Omega (|\nabla_x f| + |\nabla_y f|) dxdy \\ &\approx \sum_{i,j} \Big(|(\nabla_x f)_{i,j}| + |(\nabla_y f)_{i,j}| \Big),\end{aligned} \tag{2.15}$$

where, $(\nabla_x f)_{i,j} = f_{i+1,j} - f_{i,j}$ and $(\nabla_y f)_{i,j} = f_{i,j+1} - f_{i,j}$ are discretizations of the horizontal and vertical derivatives from the two dimensional discrete gradient of $f_{i,j}, (i,j) \in \mathbb{Z}^2$ applying ∂ separately along the x and y dimensions.

The usual TV is invariant to rotation of the domain, but the anisotropic TV is not however, it allows for other approaches that do not apply with the usual TV and a variation that is sometimes used as it may be easier to minimize, is an anisotropic version.

From the matrix point of view as shown by Jie Yan (2011) in [24], the anisotropic total variation of a matrix X in l_1 denoted by TV(X) or $\|X\|_{TV}$ is:

$$\begin{aligned}TV(X) = &\sum_{i=1}^{M-1} \sum_{j=1}^{N-1} \Big(|X_{i,j} - X_{i+1,j}| + |X_{i,j} - X_{i,j+1}| \Big) \\ &+ \sum_{i=1}^{M-1} |X_{i,N} - X_{i+1,N}| + \sum_{j=1}^{N-1} |X_{M,j} - X_{M,j+1}|.\end{aligned} \tag{2.16}$$

2.2.4 Isotropic Total Variation

The *isotropic total variation* for a smooth function f in two dimensional space is defined in [11, 19, 8] by:

$$\begin{aligned}\|f\|_{TV} &= \int_\Omega |\nabla f| dxdy = \int_\Omega \|\nabla_x f, \nabla_y f\|_2 dxdy = \int_\Omega \sqrt{(\nabla_x f)^2 + (\nabla_y f)^2} dxdy \\ &\approx \sum_{i,j} \sqrt{(\nabla_x f)_{i,j}^2 + (\nabla_y f)_{i,j}^2},\end{aligned} \tag{2.17}$$

where, $(\nabla_x f)_{i,j}$ and $(\nabla_y f)_{i,j}$ are defined above.

A difficult with TV is that it has a derivative singularity when u is locally constant and to avoid this, some algorithms regularize TV by introducing a small parameter $\epsilon > 0$ within the square root;

$$\sum_{i,j} \sqrt{\epsilon^2 + (\nabla_x f)_{i,j}^2 + (\nabla_y f)_{i,j}^2}.$$

From the matrix point of view as shown also by Jie Yan (2011) in [24], the anisotropic total variation of a matrix X

$$\sum_{i,j} \sqrt{(D_h X)_{i,j}^2 + (D_v X)_{i,j}^2}, \tag{2.18}$$

where, $D_h X = X_{i,j} - X_{i+1,j}$ and $D_v X = X_{i,j} - X_{i,j+1}$ are respectively, the horizontal (rows) variation in X and vertical (columns) variation in X.

2.3 Inverse problem

An inverse problem in science is the process of calculating from a set of observations the causal factors that produced them, see [20].

Inverse problems are the opposites of *direct problems*. In a direct problem, one finds an effect from a cause and in an inverse problem one is given the effect and want to recover the cause.

As presented in [20] for example in image processing, define the direct problem as finding out how the given sharp image will be if the camera was incorrectly focused and the inverse problem is to find the sharp image from a given blurry image. The cause is the sharp image and the effect is the blurred image.

Suppose that we have a direct problem $y = Ax + \eta$ where, $x \in \mathbb{R}^n$ is a discrete representation of the unknown object(cause) and $y \in \mathbb{R}^m$ is the data provided by a device(effect), $A \in \mathbb{R}^{m \times n}$ is a matrix modeling the process and η is random error, then the inverse problem is to find a best approximation of x (cause) given y (effect) such that $x \approx A^{-1} y$.

As defined by Samuli Siltanen (2009) in [20], an inverse problem is said to be *well-posed* if the following conditions are satisfied:

(1) Existance: A solution exists.

(2) Uniqueness: The solution must be at most one.

(3) Stability: The solution's behavior changes continuously with the initial conditions (Input data).

If at least one of the above conditions is not satisfied, the inverse problem is said to be *ill-posed problem* and it is needed to be regularized and there are several methods of regularization such as: Tikhonov regularization, Filter function regularization, Truncated Singular Value Decomposition(SVD), Total Variation regularization,etc...,see[20].

In this work, we pay special attention to the Total Variation (TV) regularization, which is a technique that was originally developed for additive white Gaussian noise image denoising by Rudin, Osher and Fatemi (1992) in [16] and for image deconvolution, it was introduced by Rudin and Osher (1994) in [15].

2.4 Signal and Image processing

2.4.1 Sparse signals processing

The resolution of the signal problem needs the notions of the l_0, l_1, l_2 and l_p norms in general for $p > 0$. The definition of l_0 -norm is given by taking the limit of l_p norm in general for $p \longrightarrow 0$ and it is introduced in [7] as follows:

$$\begin{aligned} \|x\|_0 &= \lim_{p \longrightarrow 0} \|x\|_p^p \\ &= \lim_{p \longrightarrow 0} \sum_{k=1}^{m} |x|^p, \ where \ \forall i, \ x_i \neq 0. \end{aligned}$$

This is a very simple and intuitive measure of sparsity of a vector x, counting the number of nonzero entries in it.

The problem of sparse representation is to find an unknown sparse $\overline{x} \in \mathbb{R}^{m \times 1}$ such that $y = Ax$ and $\|x\|_0$ is minimized with $y \in \mathbb{R}^n$ representing a set of n linear projections of $\overline{x}$. That is

$$\begin{aligned} &\overline{x} = \arg\min_x \|x\|_0 \\ &s.t \quad y = Ax, \end{aligned} \tag{2.19}$$

where $\|x\|_0$ is defined as the number of nonzero components of x.

As $\|x\|_0$ is non-convex and combinatorial, the problem (2.19) is impractical for real applications.

A practical alternative is to solve the l_1-norm optimization problem:

$$\begin{aligned} &\min_x \|x\|_1 \\ &s.t \quad y = Ax, \end{aligned} \tag{2.20}$$

or

$$\min_x \|x\|_1 + \frac{1}{2\rho}\|y - Ax\|_2^2,$$

and letting $\lambda = \frac{1}{2\rho}$ this last equation becomes equivalently

$$\min_x \|x\|_1 + \frac{\lambda}{2}\|y - Ax\|_2^2 \tag{2.21}$$

where (2.20) is used when y contains little or no noise and (2.21) is used with a proper parameter $\rho > 0$, the detail see [7].

Let y denote the vector of signal (image) values to be represented, let A be the matrix whose columns are the elements of the different bases to be used in the representation. Providing a single representation among the many possible ones for y in the problem:

$$\min_x J(x)$$
$$s.t \quad y = Ax,$$

where, $J(x)$ is a convex functional and when using the l_2-norm measure, the outcome is a linear operator A that is used to compute x.

The problem (2.19)posed offers literally the sparsest representation of the signal content, see[7].

The use of sparsity in signal processing leads to solve the minimization problem:

$$\arg\min_x \frac{1}{2}\|y - Ax\|_2^2 + \lambda\|x\|_1 \tag{2.22}$$

where, λ is a regularization parameter that balances the trade-off between reconstruction error and sparsity. The signal $y \in \mathbb{R}^m$ is assumed to be generated by the model

$$y = Ax + \eta$$

where, $x \in \mathbb{R}^{n\times 1}$, $A \in \mathbb{R}^{m\times n}$ and η is the white Gaussian noise,

$$\|x\|_2^2 = \sum_{i=1}^{n} |x_i|^2 \quad \text{and} \quad \|x\|_1 = \sum_{i=1}^{n} |x_i|$$

denote the energy of x and the l_1-norm of x respectively.

In sparse signal processing, the general problem is to find the value of x that satisfies

$$\arg\min_x \frac{1}{2}\|y - Ax\|_2^2 + \sum_{i=1}^{n} \phi(x_i), \tag{2.23}$$

where, the function $\phi(.)$ is a penalty function or regularization function. If this function is set to be $\phi(x) = \lambda|x|$ which is a convex function, then the problem (2.23) becomes automatically the problem (2.22) for which we need to solve in sparse signal processing, see [18].

The convex functions are preferable for regularization since they are more reliable to be minimized than the non-convex functions.

2.4.2 Denoising problem

By Pascal Getreuer (2012) in [8], Denoising problem is the problem of removing noise from an image and the most commonly studied case is with additive white noise η, where the observed noisy image f is related to the underlying true image u by

$$f = u + \eta, \tag{2.24}$$

and η is at each point in space independently and identically distributed as a zero-mean Gaussian random variable and standard deviation σ. If Ω is the image domain dealing with the regularization and restoration problem, Rudin, Osher and Fatemi (1992) in [16] proposed to estimate the denoised image u by minimizing the following functional

$$\int_\Omega |Du| + \frac{\lambda}{2}\int_\Omega |u(x) - f(x)|^2;\ u \in BV(\Omega), \tag{2.25}$$

where Ω is an open set in $\mathbb{R}^n$, λ is a positive regularization parameter, $\int_\Omega |Du| = \|u\|_{TV(\Omega)}$, Du is the gradient of u and $f \in L^2(\Omega)$ is the observed image. Here we consider the case when u is smooth and in that case, $Du(x) = \nabla u(x)$, see detail in [5] and [1].

Denoising is performed as an infinite-dimensional minimization problem, where the search space is all bounded variation (BV) images and it corresponds to $K = I$ from the constraint

$$f = K * u + \eta \tag{2.26}$$

to obtain the above denoising model in (2.24), where K is a convolution operator whose Kernel represents the Point Spread Function of the optimal system.

Now, the denoising problem is to find the denoised image u that minimize the problem (2.25). That is:

$$\arg\min_{u \in BV(\Omega)} \Big\{ \int_\Omega |Du| + \frac{1}{2\lambda}\int_\Omega (u-f)^2 dx \Big\}, \tag{2.27}$$

and this can be written equivalently as

$$\arg\min_u \|u\|_{TV(\Omega)} + \frac{\lambda}{2}\|u-f\|_2^2. \tag{2.28}$$

2.4.3 Noise models

A noise is usually considered as undesired perturbation in an image.

As shown by Alka Pandey and Singh (2015) in [1], the sources of noise in image processing are mostly: the environmental conditions, low light and sensor temperature, dust particles in the scanner, interference in transmission channel, etc...

In [17], it is described that the simplest model for intensity error is *additive noise*. Let u be a discrete image and $\delta = (\delta_{ij})_{ij}$ be a matrix of realizations of independently identically distributed random variables. If recorded data are

$$u^\delta = u + \delta,$$

then, it is said to be *additive intensity errors* in the image data.

In addition to the above additive model, we have also the model of *multiplicative noise* defined by

$$u^\delta = u\delta$$

and the multiplication is understood pointwise, that is,

$$u^\delta_{ij} = u_{ij}\delta_{ij},$$

and it is said to be *multiplicative intensity errors*.

Some of the noise models commonly used as defined by Alka Pandey, Dr. K.K.Singh (2015) in [1] are the following:

1. Gaussian noise

This kind of noise is caused by natural sources such as thermal vibration of atoms and discrete nature of radiation of warm objects. The Gaussian noise model in one dimension with mean μ and variance σ^2 is defined by the probability density function of the form:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}. \tag{2.29}$$

2. Poisson noise or Photon noise

This noise is observed mostly in electromagnetic waves such as X-rays, visible light and gamma rays which are applied in medical imaging system and its model is of the form of the Poisson probability density function with mean $\mu = \lambda$ and variance $\sigma^2 = \lambda$:

$$p(x) = \frac{\lambda^x e^{-\lambda}}{x!}. \tag{2.30}$$

3. Exponential noise

This kind of noise has the probability density function of the form:

$$p(x) = \begin{cases} ae^{-ax}, & \text{if} \quad x \geq 0 \\ 0, & \text{if} \quad x < 0. \end{cases} \tag{2.31}$$

4. Uniform noise

This kind of noise is also called the quantization noise and it is caused by quantizing the pixels of image to a number of distinct levels, and this noise is used to degrade images for the evaluation of image restoration algorithms. The uniform noise model is given by:

$$p(x) = \begin{cases} \frac{1}{h}, & \text{if} \quad a \leq t \leq b \\ 0, & otherwise, \end{cases} \tag{2.32}$$

where, $h = b - a$ with mean $\mu = \frac{a+b}{2}$ and variance $\sigma^2 = \frac{h^2}{12}$.

Beyond these described noise models above, there exist others as presented in [1] such as salt and pepper noise, periodic noise, gamma noise , Rayleigh noise etc...

2.4.4 Deblurring

Convolution

Convolution is a mathematical operation which combines two time domain functions to get a third function.

Let $u(t)$ be the input functions, $f(t)$ the output function and $h(t)$ is the response of the system and is called a point spread function (PSF) of the imaging system or the operator or filter function.

The linear system is defined as:

$$f(t) = u(t) * h(t) = \int_{-\infty}^{+\infty} u(s)h(t-s)ds. \tag{2.33}$$

The function $f(t)$ is said to be the convolution of functions $u(t)$ and $h(t)$ and the integral is called the convolution integral, where $*$ denotes convolution operator, see [6].

In digital image processing, the general discrete model for a linear degradation caused by blurring and additive noise is given by:

$$f(t) = (u * h)(t) + \eta(t) = \sum_{s\in\mathbb{R}^2} h(t-s)u(t) + \eta(t), \tag{2.34}$$

where u, f, h, η represent the original image, observed image, blur or PSF and the observation noise, respectively and $t, s \in \mathbb{R}^2$, see [6].

Deblurring problem

Deconvolution (Deblurring) is a mathematical operation used in image restoration to recover an object from an image that is degraded by blurring and noise. This means that, image deconvolution is the ill-posed problem of recovering a sharp image given a blurry one generated by a convolution. The main feature of image deconvolution is *ill-posedness* and the practical implication of this property is that the solution of a convolution equation is completely corrupted by an excess of noise propagation.

Let u be a blurred image by convolution with h as a point spread function and corrupted with additive Gaussian noise η. Then the deconvolution problem is to find the estimate image of u from the observed image

$$f = h * u + \eta, \tag{2.35}$$

and this is equivalent to estimate u from the linear deblurred model

$$f = Au + \eta, \tag{2.36}$$

and this is an image degradation model of (2.35) in matrix-vector formulation, where A is the linear operator, see [9].

Blind and Non-blind deconvolution

A ***blind deconvolution problem*** is a deconvolution problem, with the *unknown blur* known as a point spread function(PSF).

The blind deconvolution is the problem of recovering a signal and degradation kernel (PSF) from their noisy convolution, which means that the general objective of blind restoration is to estimate u and h from the model in (2.35), see [4].

A ***non-blind deconvolution*** is a deconvolution problem with *known* Point Spread Function(PSF). In a non-blind deconvolution, both blurry image f and PSF h are given(known) and the problem is to find from (2.35), the approximate image of the original image u, see [4].

Total Variation Deconvolution problem

Solving the model (2.36) for u is an ill-posed inverse problem. The Rudin et al(1992) proposed to recover an image u from the functional model

$$\begin{aligned} &\arg\min_u \|u\|_{TV(\Omega)} + \frac{\lambda}{2}\|f - Au\|_2^2 \\ \Leftrightarrow &\arg\min_u \int_\Omega |\nabla u| + \frac{\lambda}{2}\int_\Omega |f - Au|^2 dx, \end{aligned} \tag{2.37}$$

where, λ is a positive parameter, $u \in BV(\Omega)$ is smooth, $f \in L^2(\Omega)$ and $\Omega \subseteq \mathbb{R}^2$, see [9]. Then, the above model (2.37) is the total variation deconvolution problem.

Chapter 3

Algorithms implementations and Applications.

3.1 Split Bregman methods.

In this section, we describe the Split Bregman algorithms that are in details described in [11] and these algorithms are experimented numerically in three applications. Consider the generalized constrained optimization problem:

$$\begin{aligned} &\min_u E(u) \\ &s.t\ Au = b, \end{aligned} \tag{3.1}$$

where, E is a convex functional and $A : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ a linear function. The corresponding unconstrained optimization problem is:

$$\min_u E(u) + \frac{\lambda}{2}\|Au - b\|_2^2, \tag{3.2}$$

where, λ is a Penalty function weight.

3.1.1 Bregman and Split Bregman iterations.

Bregman iterations.

Bregman iteration is a concept that originated in functional analysis for finding extrema of the convex functionals. The conventional solution to this problem is to let $\lambda \longrightarrow \infty$ in the unconstrained optimization problem (3.2).

Let define a "Bregman distance" associated with a convex functional E at the point v:

$$D_E^p(u,v) = E(u) - E(v) - \langle p, u - v\rangle,$$

where p is in the sub-gradient of E at v. Clearly, this is not a distance in the usual sense because it is not in general symmetric. However, it does measure the closeness in the sense that

$$D_E^p(u,v) \geq 0$$

equivalently we have

$$E(u) \geq E(v) - \langle p, u - v\rangle \text{ and } D_E^p(u,v) \geq D_E^p(w,v),$$

for w on the line segment between u and v.
Consider two energy functional, E and H, defined over $\mathbb{R}^n$ with

$$\min_{u\in\mathbb{R}^n} Hu = 0$$

and the associated unconstrained minimization problem is:

$$\min_u E(u) + \lambda H(u). \tag{3.3}$$

Solving this problem iteratively, we have a Bregman suggested iteration:

$$\begin{aligned} u^{k+1} &= \min D_E^p(u, u^k) + \lambda H(u) \\ &= \min_u E(u) - \langle p^k, u - u^k\rangle + \lambda H(u), \end{aligned} \tag{3.4}$$

where, H is assumed to be differentiable functional. Here, we have that $0 \in \left(D_E^p(u,u^k) + \lambda H(u)\right)$ where this subdifferential is evaluated at u^{k+1}. Since $p^{k+1} \in \partial(u^{k+1})$ at this location, we have that

$$p^{k+1} = p^k - \nabla H(u^{k+1}).$$

To solve the constrained problem (3.1) with the corresponding non-constrained problem (3.2), we apply the Bregman iteration in (3.4) to have:

$$\begin{aligned} u^{k+1} &= \min D_E^p(u, u^k) + \frac{\lambda}{2}\|Au - b^k\|_2^2 \\ &= \min_u E(u) - \langle p^k, u - u^k\rangle + \frac{\lambda}{2}\|Au - b\|_2^2, \end{aligned} \tag{3.5}$$

$$p^{k+1} = p^k - \lambda A^T(Au^{k+1} - b). \tag{3.6}$$

When A is linear, we have a simplified Bregman iteration:

$$\begin{aligned} u^{k+1} &= \min_u E(u) + \frac{\lambda}{2}\|Au - b^k\|_2^2 \\ b^{k+1} &= b^k + b - Au^k, \end{aligned} \tag{3.7}$$

and we have the convergence in the 2-norm sense:

$$\lim_{k\to\infty} Au^k = b.$$

This is to mean that for large k, the iterates u^k satisfy the constraint condition to an arbitrarily high degree of accuracy.

Theorem 3.1. *Let $H : \mathbb{R}^n \longrightarrow \mathbb{R}$ be convex. Let $A : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ be linear. Consider the algorithm in (3.7). Suppose that some iterate, u^* satisfies $Au^* = b$, then u^* is a solution to the constrained optimization problem (3.1).*

Proof. Let u^* and b^* be such that $Au^* = b$ and

$$u^* = \min_u E(u) + \frac{\lambda}{2}\|Au - b^*\|_2^2. \tag{3.8}$$

Let $\widehat{u}$ be a true solution to (3.1). Then $Au^* = b = A\widehat{u}$, which implies that

$$\|Au^* - b^*\|_2^2 = \|A\widehat{u} - b^*\|_2^2.$$

Because u^* satisfies (3.8), we have

$$E(u^*) + \frac{\lambda}{2}\|Au^* - b^*\|_2^2 \leq E(\widehat{u}) + \frac{\lambda}{2}\|A\widehat{u} - b^*\|_2$$

and this implies that

$$E(u^*) \leq E(\widehat{u}).$$

As $\widehat{u}$ satisfies the original optimization problem, this inequality can be transformed into an equality which shows that u^* solves (3.1). □

Split Bregman iterations.

Consider the general L_1-regularized optimization problem:

$$\min_u |\Phi(u)| + H(u), \tag{3.9}$$

where $|.|$ denotes the L_1-norm and both $|\Phi(.)|$ and $H(.)$ are convex functions and assume that $|\Phi(.)|$ is differentiable. Consider also the problem:

$$\begin{aligned} &\min_{u,d} |d| + H(u) \\ &s.t\ d = \Phi(u). \end{aligned} \tag{3.10}$$

Its corresponding unconstrained problem is:

$$\min_{u,d} |d| + H(u) + \frac{\lambda}{2}\|d - \Phi(u)\|_2^2.$$

Let $E(u,d) = |d| + H(u)$ and define $A(u,d) = d - \Phi(u)$, then with the above Bregman formulation in (3.7) we get the *Split Bregman iteration*:

$$\begin{aligned} \left(u^{k+1}, d^{k+1}\right) &= \min_{u,d} D_E^p(u, u^k, d, d^k) + \frac{\lambda}{2}\|d - \Phi(u)\|_2^2, \\ &= \min_{u,d} E(u,d) - \langle p_u^k, u - u^k\rangle - \langle p_d^k, d - d^k\rangle + \frac{\lambda}{2}\|d - \Phi(u)\|_2^2, \\ p_u^{k+1} &= p_u^k - \lambda(\nabla\Phi)^T(\Phi u^{k+1} - d^{k+1}), \\ p_u^{k+1} &= p_d^k - \lambda(d^{k+1} - \Phi u^{k+1}). \end{aligned} \tag{3.11}$$

When we apply the simplification presented in (3.7) , we get the following elegant two-phase algorithm:

$$\begin{aligned}\left(u^{k+1}, d^{k+1}\right) &= \min_{u,d} |d| + H(u) + \frac{\lambda}{2}\|d - \Phi(u) - b^k\|_2^2, \\ b^{k+1} &= b^k + \left(\Phi(u^{k+1}) - d^{k+1}\right).\end{aligned} \tag{3.12}$$

This is a reduced sequence of unconstrained optimization problems and Bregman updates of the L_1-regularized problem (3.9). Solving the first equation of above algorithm by iteratively minimizing with respect to u and d separately decoupling L_1 and L_2 components:

Step 1: $u^{k+1} = \min_u H(u) + \frac{\lambda}{2}\|d^k - \Phi(u) - b^k\|_2^2$,

Step 2: $d^{k+1} = \min_d |d| + \frac{\lambda}{2}\|d - \Phi(u^{k+1}) - b^k\|_2^2$.

To solve the Step 1, it is possible to use any convenient optimization technique such as Fourier transform method. In Step 2, it is possible to find d explicitly using Shrinkage operator:

$$d_j^{k+1} = Shrink\left(\Phi(u)_j + b_j^k, \frac{1}{\lambda}\right),$$

where, $Shrink(x,\gamma) = \frac{x}{|x|} * max\left(|x| - \gamma, 0\right)$.
To implement the algorithm above in (3.12), we have the following algorithm as shown in [11]:

Algorithm 1: Generalized Split Bregman Algorithm.

Input : f, d^0 and b^0.
Output: u
while $\|u^k - u^{k-1}\|_2 > tol$ **do**
  $u^{k+1} = \min_u H(u) + \frac{\lambda}{2}\|d^k - \Phi(u) - b^k\|_2^2$;
  $d^{k+1} = \min_d |d| + \frac{\lambda}{2}\|d - \Phi(u^{k+1}) - b^k\|_2^2$;
  $b^{k+1} = b^k + \left(\Phi(u^{k+1}) - d^{k+1}\right)$.
end

3.1.2 Split Bregman algorithm for Sparse Signal Reconstruction.

Algorithm.

As described in [10], the sparse signal x is recovered from the observed signal $y = Ax + \eta$ with a known linear operator A and this can be achieved by solving the minimization problem in (2.21) to get

$$\hat{x} = \arg\min \|x\|_1 + \frac{\lambda}{2}\|Ax - y\|_2^2.$$

By the split Bregman iteration and solving for $x^{k+1}, d^{k+1}, b^{k+1}$ respectively we have

$$\begin{aligned} x^{k+1} &= arg\min \frac{\lambda}{2}\|Ax - y\|_2^2 + \frac{\alpha}{2}\|d^k - x - b^k\|_2^2 \\ &\Rightarrow \lambda A^T(Ax^{k+1} - y) - \alpha(d^k - x^{k+1} - b^k) = 0, \end{aligned}$$

then solving we get

$$x^{k+1} = (\lambda A^T A + \alpha I)^{-1}(\lambda A^T y + \alpha(d^k - b^k)),$$

where A^T is a transpose matrix of A.

$$d^{k+1} = arg\min \frac{\lambda}{2}\|d\|_1 + \frac{\alpha}{2}\|d - x^{k+1} - b^k\|_2^2,$$

using the Shrinkage formula we get

$$d^{k+1} = Shrink\Big(x^{k+1} + b^k, \frac{1}{\alpha}\Big),$$

and finally we have

$$b^{k+1} = b^k + x^{k+1} - d^{k+1}.$$

Therefore, the Split Bregman algorithm to implement the results is established to be:

Algorithm 2: Split Bregman Sparse Signal recovering.

Input : $d^0 = 0$ and $b^0 = 0$.
Output: x
while $\|x^k - x^{k-1}\|_2 > tol$ **do**
  $x^{k+1} = (\lambda A^T A + \alpha I)^{-1}(\lambda A^T y + \alpha(d^k - b^k))$;
  $d^{k+1} = Shrink\Big(x^{k+1} + b^k, \frac{1}{\alpha}\Big)$;
  $b^{k+1} = b^k + x^{k+1} - d^{k+1}$.
end

Numerical experiment.

A numerical experiment for sparse signal reconstruction is made by presenting an example of an original sparse signal to be reconstructed by creating the spike signal as the original signal with length $N = 150$ and it is presented in the first plot of the Figure 3.1. The 5-point impulse response or point spread func-

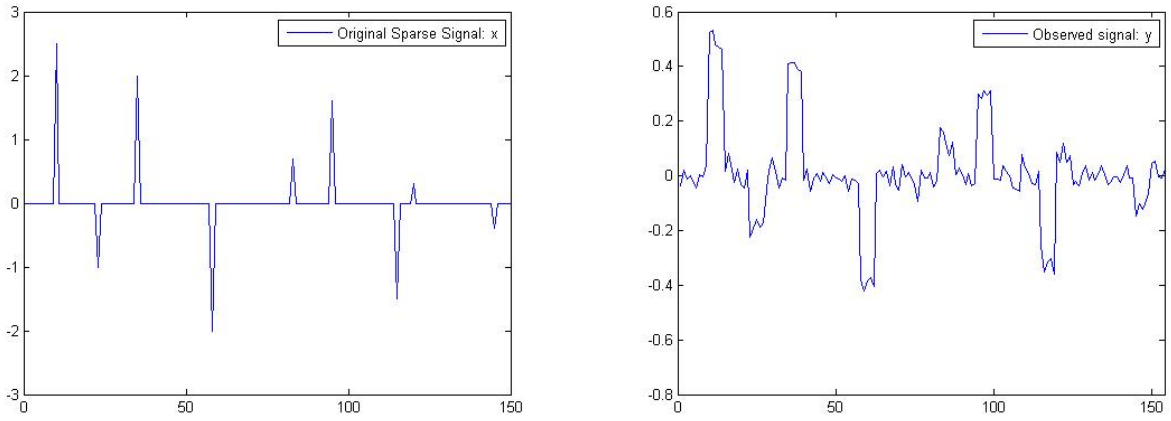

Figure 3.1: Original and Observed signals.
In these graphs, the first plot shows the original sparse signal and the second plot shows the observed sparse signal.

tion (PSF) f is chosen to be a uniform noise model in (2.32) of the form

$$f(t) = \begin{cases} \frac{1}{L}, & \text{if} \quad \frac{-L}{2} \leq t \leq \frac{L}{2} \\ 0, & otherwise \end{cases}$$

with the length of this PSF to be used $L = 5$. The additive zero-mean $\mu = 0$ Gaussian noise is created with standard deviation $\sigma = 0.03$ and it is added to the convolution of original signal x and impulse response to get the observed signal y and it is presented in the second plot of the Figure 3.1. The regularization parameter λ has been varied for different values: $\lambda = 5, \lambda = 15, \lambda = 20$, and $\lambda = 30$ and it is observed that for $\lambda = 30$ with splitting regularization parameter $\alpha = 1$, the reconstructed sparse signal is quite similar to the original signal as required and 20 iterations are enough for a good reconstruction of the signal with tolerance error 0.0021.

The first plot in figure 3.2 shows the reconstruction of the signal with different values of λ elaborated above and the second plot in figure 3.2 shows the comparison or fitness of the original and reconstructed signals for $\lambda = 30$.

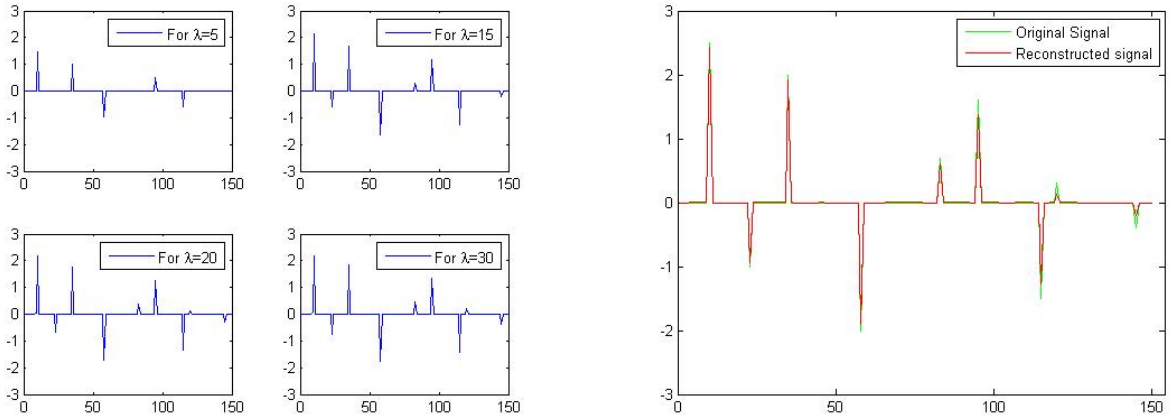

Figure 3.2: Reconstructed signals and Comparison of original and reconstructed signals.
In these figures, the first plot shows the reconstructed sparse signal for different values of λ and the signal for $\lambda = 30$ has the good similarity to the original signal and the second plot shows the fitness of the original sparse signal and the reconstructed sparse signal for $\lambda = 30$.

3.1.3 Split Bregman algorithm for Total Variation Deconvolution.

Algorithm.

In this section, we consider the non-blind deconvolution case for the application of regularization of an inverse problem.

Let us consider the Rudin et al (1992) proposed model for total variation deconvolution in (2.37) and considering the anisotropic problem for a linear operator A, we have

$$\min_u |\nabla_x u| + |\nabla_y u| + \frac{\lambda}{2}\|Au - f\|_2^2.$$

Replacing $\nabla_x u$ by d_x and $\nabla_y u$ by d_y, then we have

$$\min_u |d_x| + |d_y| + \frac{\lambda}{2}\|Au - f\|_2^2$$
$$s.t\ d_x = \nabla_x u$$
$$d_y = \nabla_y u.$$

The corresponding unconstrained problem with penalty function terms is:

$$\min_{d_x,d_y,u} |d_x| + |d_y| + \frac{\lambda}{2}\|Au - f\|_2^2 + \frac{\alpha}{2}\|d_x - \nabla_x u\|_2^2 + \frac{\alpha}{2}\|d_y - \nabla_y u\|_2^2,$$

and by the Split Bregman iteration (3.12) we get

$$\min_{d_x,d_y,u} |d_x| + |d_y| + \frac{\lambda}{2}\|Au - f\|_2^2 + \frac{\alpha}{2}\|d_x - \nabla_x u - b_x^k\|_2^2 + \frac{\alpha}{2}\|d_y - \nabla_y u - b_y^k\|_2^2. \tag{3.13}$$

By solving this equation (3.13) for u we have:

$$\lambda A^T(Au - f) - \alpha\nabla_x^T(d_x^k - \nabla_x u - b_x^k - \alpha\nabla_x^T(d_y^k - \nabla_y u - b_y^k) = 0,$$

and this gives

$$(\lambda A^T A - \alpha\triangle)u = \lambda A^T f - \alpha div(d^k - b^k).$$

Dividing by α and using the Fourier transform as shown in [10] we have:

$$\Big(\frac{\lambda}{\alpha}|\hat{A}|^2 - \Re(\hat{\triangle})\Big)\hat{U} = \frac{\lambda}{\alpha}\bar{\hat{A}}\hat{F} - \widehat{div(d^k - b^k)},$$

which leads to the Fourier transform:

$$\hat{U} = \Big(\frac{\lambda}{\alpha}|\hat{A}|^2 - \Re(\hat{\triangle})\Big)^{-1}\Big(\frac{\lambda}{\alpha}\bar{\hat{A}}\hat{F} - \widehat{div(d^k - b^k)}\Big) \tag{3.14}$$

where $\Re$ is the real parts and multiplication are pointwise and by the inverse Fourier transform we have the updated u to be u^{k+1}.

Solving for d from the equation (3.13) and using the Shrinkage formula we have:

$$\begin{aligned} d_x^{k+1} &= \min_{d_x} |d_x| + \frac{\alpha}{2}\|d_x - \nabla_x u^{k+1} - b_x^k\|_2^2 \\ &= Shrink\Big(\nabla_x u^{k+1} + b_x^k, \frac{1}{\alpha}\Big) \\ &= \max\Big(\|\nabla_x u^{k+1} + b_x^k\|_2 - \frac{1}{\alpha}, 0\Big)\frac{\nabla_x u^{k+1} + b_x^k}{\|\nabla_x u^{k+1} + b_x^k\|_2}, \end{aligned} \tag{3.15}$$

and

$$\begin{aligned} d_y^{k+1} &= \min_{d_y} |d_y| + \frac{\alpha}{2}\|d_y - \nabla_y u^{k+1} - b_y^k\|_2^2 \\ &= Shrink\Big(\nabla_y u^{k+1} + b_y^k, \frac{1}{\alpha}\Big) \\ &= \max\Big(\|\nabla_y u^{k+1} + b_y^k\|_2 - \frac{1}{\alpha}, 0\Big)\frac{\nabla_y u^{k+1} + b_y^k}{\|\nabla_y u^{k+1} + b_y^k\|_2}, \end{aligned} \tag{3.16}$$

with the update formula for b:

$$b_x^{k+1} = b_x^k + (\nabla_x u^{k+1} - d_x^{k+1}),$$
$$b_y^{k+1} = b_y^k + (\nabla_y u^{k+1} - d_y^{k+1}).$$

Therefore, we have the following Split Bregman algorithm 3 for TV Deconvolution as:

Algorithm 3: Split Bregman TV Deconvolution

Initialize: $u^0 = f$, $d_x^0 = d_y^0 = b_x^0 = b_y^0 = 0$.
Output : u.
while $\|u^k - u^{k-1}\|_2 > tol$ **do**
  $Update\, u^{k+1}\, by\, inverse\, Fourier\, transform\, of\, \hat{U}\, in$(3.14);
  $d_x^{k+1} = Shrink\left(\nabla_x u^{k+1} + b_x^k, \frac{1}{\alpha}\right)$;
  $d_y^{k+1} = Shrink\left(\nabla_y u^{k+1} + b_y^k, \frac{1}{\alpha}\right)$;
  $b_x^{k+1} = b_x^k + (\nabla_x u^{k+1} - d_x^{k+1})$;
  $b_y^{k+1} = b_y^k + (\nabla_y u^{k+1} - d_y^{k+1})$.
end

Numerical experiment.

A numerical experiment for Split Bregman TV-Deconvolution algorithm is made by using the standard test image (Lena) of size 512×512 as an original image. The Gaussian noise with mean $\mu = 30$ and variance $\sigma^2 = 30$ is applied to the original image and we obtain the blurred observed image and the algorithm 3 is experimented and applied to recover a deblurred image from the observed blurred image. It is well observed that, for a fixed splitting regularization parameter $\alpha = 0.1$ and as the regularization parameter λ is varying increasingly from the small to the highest chosen values, the original image is more well recovered.

The four different regularization parameters $\lambda = 10$, $\lambda = 100$, $\lambda = 10000$ and $\lambda = 100,000,000$ are chosen for experimenting as shown in Figure 3.3. It is observed that for the previous regularization parameters λ, a recovered image is not well observed but as λ is increasing to be large for instance as $\lambda = 100,000,000$, more a blur image is deblurred and recovered image is quite close and similar to the original image as show in Figure 3.3. This does not need more iterations, only 10 iterations are enough to recover the original image. The image results are shown in Figure 3.3.

Note that the Anisotropic and Isotropic TV Deconvolution give the similar results of deconvolution.

Figure 3.3: SB TV-Deblurring for Lena test image.
The figure shows, the original image of Lena to be recovered by TV Deconvolution, an observed blurred image and others are the recovered images for different values of λ.

3.1.4 Split Bregman algorithm for Total Variation Denoising.

The main problem that we have to solve is to recover the original image u from the observed noised image f with the constraint (2.24) and the corresponding proposed model from equation (2.28) which is to solve

$$\arg\min_u \|u\|_{TV(\Omega)} + \frac{\lambda}{2}\|u-f\|_2^2. \tag{3.17}$$

Now, we consider two different cases of TV Denoising which are *Anisotropic* and *Isotropic TV Denoising*.

Anisotropic TV Denoising algorithm.

Considering the above proposed model (3.17), the anisotropic problem is defined by:

$$\min_u |\nabla_x u| + |\nabla_y u| + \frac{\lambda}{2}\|u-f\|_2^2.$$

Replacing $\nabla_x u$ by d_x and $\nabla_y u$ by d_y, then we have

$$\begin{aligned}\min_u |d_x| + |d_y| + \frac{\lambda}{2}\|u-f\|_2^2 \\ s.t\ d_x = \nabla_x u \\ d_y = \nabla_y u.\end{aligned}$$

The corresponding unconstrained problem with penalty function terms is:

$$\min_{d_x,d_y,u} |d_x| + |d_y| + \frac{\lambda}{2}\|u-f\|_2^2 + \frac{\alpha}{2}\|d_x - \nabla_x u\|_2^2 + \frac{\alpha}{2}\|d_y - \nabla_y u\|_2^2,$$

and by the Split Bregman iteration (3.12) we get

$$\min_{d_x,d_y,u} |d_x| + |d_y| + \frac{\lambda}{2}\|u-f\|_2^2 + \frac{\alpha}{2}\|d_x - \nabla_x u - b_x^k\|_2^2 + \frac{\alpha}{2}\|d_y - \nabla_y u - b_y^k\|_2^2. \tag{3.18}$$

By solving this equation (3.18) for u, we have

$$\lambda(u-f) - \alpha\nabla_x^T(d_x^k - \nabla_x u - b_x^k - \alpha\nabla_x^T(d_y^k - \nabla_y u - b_y^k) = 0,$$

and this gives

$$(\lambda I - \alpha\triangle)u = \lambda f - \alpha div(d^k - b^k)$$

Dividing by α and using the Fourier transform as shown in [10] we have:

$$\left(\frac{\lambda}{\alpha}\hat{I} - \Re(\hat{\triangle})\right)\hat{U} = \frac{\lambda}{\alpha}\hat{F} - \widehat{div(d^k - b^k)},$$

which leads to the Fourier transform:

$$\hat{U} = \left(\frac{\lambda}{\alpha}\hat{I} - \Re(\hat{\triangle})\right)^{-1}\left(\frac{\lambda}{\alpha}\hat{F} - \widehat{div(d^k - b^k)}\right) \tag{3.19}$$

where $\Re$ is the real parts and multiplication are pointwise and by the inverse Fourier transform we have the updated u to be u^{k+1}.

Solving for d from the equation (3.18) and using the Shrinkage operator we have:

$$\begin{aligned} d_x^{k+1} &= \min_{d_x} |d_x| + \frac{\alpha}{2}\|d_x - \nabla_x u^{k+1} - b_x^k\|_2^2 \\ &= Shrink\Big(\nabla_x u^{k+1} + b_x^k, \frac{1}{\alpha}\Big) \\ &= \max\Big(\|\nabla_x u^{k+1} + b_x^k\|_2 - \frac{1}{\alpha}, 0\Big)\frac{\nabla_x u^{k+1} + b_x^k}{\|\nabla_x u^{k+1} + b_x^k\|_2}, \end{aligned} \tag{3.20}$$

and

$$\begin{aligned} d_y^{k+1} &= \min_{d_y} |d_y| + \frac{\alpha}{2}\|d_y - \nabla_y u^{k+1} - b_y^k\|_2^2 \\ &= Shrink\Big(\nabla_y u^{k+1} + b_y^k, \frac{1}{\alpha}\Big) \\ &= \max\Big(\|\nabla_y u^{k+1} + b_y^k\|_2 - \frac{1}{\alpha}, 0\Big)\frac{\nabla_y u^{k+1} + b_y^k}{\|\nabla_y u^{k+1} + b_y^k\|_2}. \end{aligned} \tag{3.21}$$

with the update formula for b:

$$\begin{aligned} b_x^{k+1} &= b_x^k + (\nabla_x u^{k+1} - d_x^{k+1}), \\ b_y^{k+1} &= b_y^k + (\nabla_y u^{k+1} - d_y^{k+1}). \end{aligned}$$

Therefore, we have the following Split Bregman algorithm 4:

Algorithm 4: Split Bregman Anisotropic TV Denoising

Initialize: $u^0 = f$, $d_x^0 = d_y^0 = b_x^0 = b_y^0 = 0$.
Output : u.
while $\|u^k - u^{k-1}\|_2 > tol$ **do**
  $Update\, u^{k+1}\, by\, inverse\, Fourier\, transform\, of\, \hat{U}\, in$(3.19);
  $d_x^{k+1} = Shrink\Big(\nabla_x u^{k+1} + b_x^k, \frac{1}{\alpha}\Big)$;
  $d_y^{k+1} = Shrink\Big(\nabla_y u^{k+1} + b_y^k, \frac{1}{\alpha}\Big)$;
  $b_x^{k+1} = b_x^k + (\nabla_x u^{k+1} - d_x^{k+1})$;
  $b_y^{k+1} = b_y^k + (\nabla_y u^{k+1} - d_y^{k+1})$.
end

Isotropic TV Denoising algorithm.

Considering also the proposed model (3.17), the isotropic problem is defined by:

$$\min_u \sum_i \sqrt{(\nabla_x u)_i^2 + (\nabla_y u)_i^2} + \frac{\lambda}{2}\|u-f\|_2^2.$$

Letting $d_x \approx \nabla_x u$ and $d_y \approx \nabla_y u$ and splitting L_1 and L_2 components by split Bregman formulation, we have:

$$\min_{d_x,d_y,u} \|(d_x,d_y)\|_2 + \frac{\lambda}{2}\|u-f\|_2^2 + \frac{\alpha}{2}\|d_x-\nabla_x u-b_x\|_2^2 + \frac{\alpha}{2}\|d_y-\nabla_y u-b_y\|_2^2,$$

where, $\|(d_x,d_y)\|_2 = \sqrt{d_{x,i,j}^2 + d_{x,i,j}^2}$. As in the anisotropic TV case, the resolution for u with its update u^{k+1} is given in 3.19 and solving for d_x and d_y we have by the Shrinkage formula:

$$d_x^{k+1} = \max\left(m^k - \frac{1}{\alpha}, 0\right)\frac{\nabla_x u^k + b_x^k}{m^k},$$
$$d_y^{k+1} = \max\left(m^k - \frac{1}{\alpha}, 0\right)\frac{\nabla_y u^k + b_y^k}{m^k},$$

where, $m^k = \sqrt{|\nabla_x u^k + b_x^k|^2 + |\nabla_y u^k + b_y^k|^2}$, and therefore the Split Bregman algoritm 5 is given by:

Algorithm 5: Split Bregman Isotropic TV Denoising

Initialize: $u^0 = f, d_x^0 = d_y^0 = b_x^0 = b_y^0 = 0$.
Output : u.
while $\|u^k - u^{k-1}\|_2 > tol$ **do**
  $Update\, u^{k+1}\, by\, inverse\, Fourier\, transform\, of\, \hat{U}\, in$(3.19);
  $d_x^{k+1} = \max\left(m^k - \frac{1}{\alpha}, 0\right)\frac{\nabla_x u^k + b_x^k}{m^k}$;
  $d_y^{k+1} = \max\left(m^k - \frac{1}{\alpha}, 0\right)\frac{\nabla_y u^k + b_y^k}{m^k}$;
  $b_x^{k+1} = b_x^k + (\nabla_x u^{k+1} - d_x^{k+1})$;
  $b_y^{k+1} = b_y^k + (\nabla_y u^{k+1} - d_y^{k+1})$.
end

Numerical experiment.

A numerical experiment for Split Bregman TV-Denoising algorithm is made by using the standard test image (Lena) of size 512×512 as an original image. The observed noised image is created by adding the additive white noise η created to the original image. The Total variation denoising algorithm is experimented to recover the denoised image from the observed image and this is achieved by varying the values

of regularization parameter λ and we observe that when this value is large, the denoising goes slowly and once λ is small, the denoised image becomes well recovered. The figure 3.4 shows the images results for different values of regularization parameter. It is observed that for the regularization parameters $\lambda = 3$,

Figure 3.4: SB TV-Denoising for Lena test image.
The figure shows the original image (Lena), an observed noised image and others are the recovered images for different values of regularization parameter λ.

$\lambda = 1.5$ and $\lambda = 0.5$ the noisy image is not well denoised but as the regularization parameters are decreasing to $\lambda = 0.1$ the observed noisy image is denoised and the image that is recovered, is quite close to an original image. Therefore, we observe that by varying the regularization parameter λ, the largest and the smallest values are not consistent for denoising and for the fixed split regularization parameter $\alpha = 1$, the regularization parameter $\lambda = 0.1$ gives the recovered image which is quite similar to the original image. This does not need many iterations, only 10 iterations are enough to recover the original image.

Note that the Anisotropic and the Isotropic TV denoising gives the similar result of denoising.

3.2 Alternating Direction Method of Multipliers

3.2.1 Description and Convergence

Description

The details of the Alternating Direction Method of Multipliers, abbreviated as ADMM can be found in [21]. ADMM solves the problem of the form

$$\begin{aligned} &\min f(x) + g(z) \\ &s.t\ Ax + Bz = c, \end{aligned} \tag{3.22}$$

with variables $x \in \mathbb{R}^n, z \in \mathbb{R}^m$, where $A \in \mathbb{R}^{p\times n}, B \in \mathbb{R}^{p\times m}$ and $c \in \mathbb{R}^p$.
In this problem, it is assumed that f and g are convex functions. The optimal value of (3.22) is denoted by

$$p^* = \inf\{f(x) + g(x) :\ Ax + Bz = c\}.$$

The corresponding augmented Lagrangian is

$$L_\alpha(x, z, y) = f(x) + g(x) + y^T(Ax + Bz - c) + \frac{\alpha}{2}\|Ax + Bz - c\|_2^2,$$

then ADMM consists of the following iterations:

$$x^{k+1} = \arg\min_x L_\alpha(x, z^k, y^k), \tag{3.23}$$

$$z^{k+1} = \arg\min_z L_\alpha(x^{k+1}, z, y^k), \tag{3.24}$$

$$y^{k+1} = y^k + \alpha(Ax^{k+1} + Bz^{k+1} - c), \tag{3.25}$$

where, $\alpha > 0$. In ADMM, x and z are updated in alternating or sequential fashion, which accounts for term alternating direction. The algorithm in ADMM consists of z^k and y^k. This is to mean that (z^{k+1}, y^{k+1}) is a function of (z^k, y^k) and the variable x^k is an intermediate result computed from the previous state (z^{k-1}, y^{k-1}).
ADMM can be formulated in a scaled form and by combining the linear and quadratic terms in the

augmented Lagrangian and scaling the dual variable to simplify ADMM iterations. Let the residual $r = Ax + Bz - c$ and the scaled dual variable $u = \frac{y}{\alpha}$, then we have

$$y^T r + \frac{\alpha}{2}\|r\|_2^2 = \frac{\alpha}{2}\left\|r + \frac{y}{\alpha}\right\|_2^2 - \frac{1}{2\alpha}\|y\|_2^2 = \frac{\alpha}{2}\|r+u\|_2^2 - \frac{\alpha}{2}\|u\|_2^2.$$

ADMM iterations (3.23), (3.24) and (3.25) become

$$x^{k+1} = \arg\min_x \left(f(x) + \frac{\alpha}{2}\left\|Ax + Bz^k - c + u^k\right\|_2^2\right), \tag{3.26}$$

$$z^{k+1} = \arg\min_z \left(g(z) + \frac{\alpha}{2}\left\|Ax^{k+1} + Bz - c + u^k\right\|_2^2\right), \tag{3.27}$$

$$u^{k+1} = u^k + Ax^{k+1} + Bz^{k+1} - c. \tag{3.28}$$

The residual at iteration k is: $r^k = Ax^k + Bz^k - c$, then we have $u^k = u^0 + \sum_{j=1}^k r^j$, and this is called the running sum of the residuals.

Convergence

The convergence is studied by considering two assumptions, one about the functions f and g and other about the problem (3.22).

1^{st} **assumption**: The functions $f : \mathbb{R}^n \longrightarrow \mathbb{R} \cup \{+\infty\}$ are closed, proper and convex. This assumption implies that $epif = \{(x,t) \in \mathbb{R}^n \times \mathbb{R} : f(x) \leq t\}$ is closed nonempty convex set and that the equation (3.23) and (3.24) are solvable. Also, this assumption allows f and g to be non differentiable and assume the value $+\infty$.

2^{nd} **assumption**: The unugmented Lagrangian L_0 has a saddle point. Explicitly, there exist (x^*, z^*, y^*), not necessarily unique, for which

$$L_0(x^*, z^*, y) \leq L_0(x^*, z^*, y^*) \leq L_0(x, z, y^*)$$

holds for all x, y and z.

By the first assumption, it follows that $L_0(x^*, z^*, y^*) < \infty$ for any saddle point (x^*, z^*, y^*) and this shows that (x^*, z^*) is a solution of (3.22), $f(x^*) < \infty$ and $g(x^*) < \infty$.

According to these two assumptions, the convergence of ADMM leads to:

* Residual convergence: $r^k \longrightarrow 0$ as $k \longrightarrow \infty$.

* Objective convergence: $f(x^k) + g(z^k) \longrightarrow p^*$ as $k \longrightarrow \infty$.

∗ Dual variable convergence: $y^k \longrightarrow y^*$ as $k \longrightarrow \infty$, where x^* is a dual optimal point.

Note that x^k and z^k need not to converge to optimal values.

Optimality conditions and Stopping criterion.

The necessary and sufficient optimality conditions for the ADMM problem (3.22) are:

$$Ax^* + Bz^* - c = 0, \tag{3.29}$$

$$0 \in \partial f(x^*) + A^T y^*, \tag{3.30}$$

$$0 \in \partial g(z^*) + B^T y^*, \tag{3.31}$$

where, ∂ denotes the sub-differential operator and 1^{st} condition is called the primal feasibility while the last two are the dual feasibility.
The last condition (3.31) holds for $(x^{k+1}, z^{k+1}, y^{k+1})$; the residuals for (3.29) and (3.30) are the primal

$$r^{k+1} = Ax^{k+1} + Bz^{k+1} - c \text{ and dual } s^{k+1} = \alpha A^T B(z^{k+1} - z^k)$$

and these residuals converge to 0 as ADMM proceeds.

Stopping criteria.

When r^k and s^k are small, the objective sub-optimality $f(x^k) + g(z^k) - p^*$ must be small and we have

$$f(x^k) + g(z^k) - p^* \leq -(y^k)^T r^k + (x^k - x^*)^T s^k.$$

If it is estimated that $\left\| x^k - x^* \right\|_2 \leq d$ since x^* is not know, we have

$$f(x^k) + g(z^k) - p^* - (y^k)^T r^k + d \left\| s^k \right\|_2 \leq \left\| y^k \right\|_2 \left\| r^k \right\|_2 d \left\| s^k \right\|_2.$$

It is suggested that a reasonable termination criterion is that the residuals r^k and s^k must be small. That is $\left\| r^k \right\|_2 \leq \epsilon_{pr}$ and $\left\| r^k \right\|_2 \leq \epsilon_{du}$, where, $\epsilon_{pr} > 0$ and $\epsilon_{du} > 0$ are feasible tolerances for the primal and dual

feasibility conditions (3.29) and (3.29) respectively and they are determined by:

$$\epsilon_{pr} = \sqrt{p}\epsilon_{ab} + \epsilon_{re} \max\left\{ \left\| Ax^k \right\|_2, \left\| Bz^k \right\|_2, \|c\|_2 \right\} \tag{3.32}$$

$$\epsilon_{du} = \sqrt{n}\epsilon_{ab} + \epsilon_{re} \left\| A^k y^k \right\|_2, \tag{3.33}$$

where, $\epsilon_{ab} > 0$ and $\epsilon_{re} > 0$ and absolute tolerance and relative tolerance, $\sqrt{p}$ and $\sqrt{n}$ are from that l_2 norms are in $\mathbb{R}^p$ and $\mathbb{R}^n$ respectively and the reasonable value for relative stopping criterion might be $\epsilon_{re} = 10^{-3}$ or $\epsilon_{re} = 10^{-4}$, depending on the application.

The choice of absolute stopping criterion depending on the scale of the typical variable values. More details about ADMM method, see in [21].

3.2.2 ADMM for Total variation denoising.

In this section, it is needed to solve

$$\min \frac{1}{2} \|Ax - b\|_2^2 + \lambda \|x\|_1, \tag{3.34}$$

where, $\lambda > 0$ is a scalar regularization parameter and for TV Denoising, the matrix A is assumed to be unitary.

In ADMM form, a general case of (3.34) can be written as

$$\min \frac{1}{2} \|Ax - b\|_2^2 + \lambda \|Fx\|_1, \tag{3.35}$$

where, F is an arbitrary linear transformation.

A special case is when $F \in \mathbb{R}^{(n-1)\times n}$ is the difference matrix,

$$F_{ij} = \begin{cases} 1, & \text{if} \quad j = i+1 \\ -1, & \text{if} \quad j = 1 \\ 0, & otherwise, \end{cases}$$

and $A = I$ in which case, the generalization reduces to

$$\min \frac{1}{2} \|x - b\|_2^2 + \lambda \sum_{i=1}^{n-1} |Fx|, \tag{3.36}$$

where, the second term is the total variation of x and this problem is called the total variation denoising. In ADMM form, the problem (3.35) is written as:

$$\min \frac{1}{2}\|Ax-b\|_2^2+\lambda\|z\|_1$$
$$s.t\ \ Fx-z=0,$$

and the corresponding ADMM iterations are:

$$x^{k+1}=(A^TA+\alpha F^TF)^{-1}(A^Tb+\alpha F^T(z^k-u^k)),$$
$$z^{k+1}=S_{\frac{\lambda}{\alpha}}(Fx^{k+1}+u^k),$$
$$u^{k+1}=u^k+Fx^{k+1}-z^{k+1}.$$

For the special case of total variation denoising $A^TA+\alpha F^TF$ is tridiagonal. So, the $x-$update can carried out in $O(n)$. The details of this method is found in [21].

3.2.3 Numerical experiment.

A numerical experiment for ADMM algorithm is made by using the standard test image (Lena) of size 512×512. An observed noisy image is created with additive Gaussian white noise from the original image and different regularization parameters are used for observation and comparison with the considered termination tolerance $\epsilon_{ab}=10^{-4}$ and $\epsilon_{rel}=10^{-2}$. The regularization parameters that are used are $\lambda=45,\lambda=60,\lambda=90$, and $\lambda=115$. It is observed that for $\lambda=115$ the approximate image is recovered comparatively to the original image and the stopping criterion is achieved at 140 iterations. The image results are shown in Figure 3.5.

3.3 The Rudin-Osher-Fatemi denoising model on the graph (ROF-Graph).

3.3.1 Description of the algorithm.

The Rudin-Osher-Fatemi denoising model on the graph, abbreviated here as ROF-Graph, is described in [13]. This method takes advantage of the fact that an image can be considered as a grid of points (pixels) and constructs an optimal way of recovering an estimate to the original image $f^*\in BV$ from the observed

Figure 3.5: ADMM TV-Denoising for Lena test image.
The figure shows the Original image, Observed noisy image and Denoised image by ADMM for two different values of regularization parameter λ.

image $f_{ob} = f^* + \eta$, where $\eta \in L^2$ is a noise. This is achieved by solving the ROF model

$$L_{2,1}(t, f_{ob}; L^2, BV) = \inf_{g \in BV} \left\{ \frac{1}{2} \left\| f_{ob} - g \right\|_{L^2}^2 + t \left\| g \right\|_{BV} \right\}, \tag{3.37}$$

for some optimal regularization parameter t. The problem that is formulated on a general graph is as follows: Let $G = (V, E)$ be a finite directed and connected graph with N vertices $V = \{v_1, \cdots, v_N\}$ and M directed edges $E = \{e_1, \cdots, e_M\}$, where, each edge is determined by a pair of two vertices $e = (v_i, v_j)$ for some $i, j \in \{1, 2, \cdots, N\}$.

Let $S_V = \{f : f : V \longrightarrow \mathbb{R}\}$ denote the set of real-valued functions defined on the edges. Then the analogue of ROF model on the graph G can be formulated as follows:

Problem 3.1. *Suppose that $f_{ob} \in S_V$ is known. For given $t > 0$, find exact minimizer of the functional*

$$L_{2,1}(t, f_{ob}; l^2(S_V), BV(S_V)) = \inf_{g \in BV(S_V)} \left\{ \frac{1}{2} \left\| f_{ob} - g \right\|^2_{l^2(S_V)} + t \left\| g \right\|_{BV(S_V)} \right\}, \tag{3.38}$$

where,

$$\left\| f \right\|^2_{l^2(S_V)} = \Big(\sum_{v \in V} (f(v))^2 \Big)^{\frac{1}{2}}; \ \left\| f \right\|_{BV(S_V)} = \left\| grad\, f \right\|_{l^1(S_E)}; \ \left\| h \right\|_{l^1(S_E)} = \sum_{l \in E} |h(e)|, \tag{3.39}$$

and the operator $grad : S_V \longrightarrow S_E$ is defined by the formula

$$(grad\, f)(e) = f(v_j) - f(v_i) \ \ if\ e = (v_i, v_j).$$

3.3.2 Algorithm construction.

With the $L_{2,1}$-functional defined above in (3.38), the exact minimizer $f_{opt,t}$ is given by

$$f_{opt,t} = f_{ob} - \widetilde{h},$$

where, $\widetilde{h}$ is the nearest elements to f_{ob} such that

$$\inf_{h \in tB_{BV^*(S_V)}} \left\| f_{ob} - h \right\|_{l^2(S_V)} = \left\| f_{ob} - \widetilde{h} \right\|_{l^2(S_V)}, \tag{3.40}$$

knowing that $tB_{BV^*(S_V)} = tdiv(B_{l^\infty(S_E)})$ for $t > 0$.

The proposed algorithm constructs $\widetilde{h}$ through a sequence of elements $g_n \in tB_{l^\infty(S_E)}$ such that $div(g_n) \longrightarrow \widetilde{h}$ as $n \longrightarrow +\infty$ in the metric of $l^2(S_V)$.

Algorithm.

Let $f_{ob} \in S_V$ on $G = (V = \{v_1, \cdots, v_N\}, E = \{e_1, \cdots, e_M\})$, a regularization parameter t and a minimum number of iterations N_{iter} be given. Set $e_k = (v_1, v_j) \in E, k = 1, 2, \cdots, M$; for some $i, j \in \{1, 2, \cdots, N\}$.

Define the operator $T = T_M T_{M-1} T_{M-2} \cdots T_2 T_1$, then $T_k : tB_{l^\infty(S_E)} \longrightarrow tB_{l^\infty(S_E)}$ is defined as:

$$\Big(T_k g\Big)(e) = \begin{cases} \psi(e), & \text{if} \quad e = e_k \\ g(e), & \text{if} \quad e \neq e_k, \end{cases}$$

where,

$$\psi(e) = \begin{cases} Kg(e_k), & \text{if} \quad Kg(e_k) \in [-t, +t] \\ -t, & \text{if} \quad Kg(e_k) < -t \\ +t, & \text{if} \quad Kg(e_k) > +t, \end{cases}$$

and

$$Kg(e_k) = \frac{1}{2}\Big\{[f_{ob}(v_j) - (div g)(v_j) - g(e_k)] - [f_{ob}(v_i) - (div g)(v_i) - g(e_k)]\Big\}.$$

Now, the steps of algorithm are the following:

Step 1. Take $g_0 = 0$, or choose any $g \in tB_{l^\infty(S_E)}$.

Step 2. Calculate $g = Tg_0$ i.e $(Tg_0)(e_k)$ for $k = 1, 2, \cdots, M$. If $g = g_0$ then take $\widetilde{h} = div(g_0)$, otherwise go to **step 3**.

Step 3. Put $g_0 = g$ and go to **step 2**.

These steps continue the process applying operator T to new element $g \in tB_{l^\infty}(S_E)$ generating $g_0, g_1 = Tg_0, g_2 = Tg_1, \cdots, g_n = Tg_{n-1}$, with $g_n \in tB_{l^\infty}(S_E), n = 0, 1, 2, \cdots$ until the maximum number of iterations N_{iter} is reached.

3.3.3 Convergence.

To study the convergence of the above algorithm, we have the following proposition and theorem.

Proposition 3.1. *Let $\widetilde{h}$ be the minimizer defined by The operator T is continuous and satisfies the following two conditions:*

(a) For any $g \in tB_{l^\infty(S_E)}$, $div g = \widetilde{h}$ if and only if $Tg = g$;

(b) For any $g \in tB_{l^\infty(S_E)}$, if $div g \neq \widetilde{h}$,

$$\left\| f_{ob} - div(Tg) \right\|_{l^2(S_V)} < \left\| f_{ob} - div g \right\|_{l^2(S_V)}.$$

Theorem 3.2. *Let $\widetilde{h}$ be the minimizer defined by $g \in tB_{l^\infty(S_E)}$ and let T be the operator constructed previously. Then $div(T^n g) \longrightarrow \widetilde{h}$ as $n \longrightarrow \infty$ in the metric of $l^2(S_V)$.*

See the proofs of this proposition and theorem in [13] and the details of ROF-Graph are found in [13].

3.3.4 Numerical experiment.

A numerical experiment for ROF-Graph algorithm is made by using the standard test image (Lena) of size 512×512 and an observed noisy image is created with additive Gaussian white noise from the original image. The different regularization parameters are used for observation and comparison with the considered termination tolerance $\epsilon = 10^{-3}$. The regularization parameters that are used are $t = 15$, $t = 20$, $t = 32$ and $t = 35$ and it is observed that for $t = 35$ the recovered image is well observed. The image results are shown in Figure 3.5.

Figure 3.6: ROF-Graph TV-Denoising for Lena test image.
The figure shows the Original image, Observed noisy image and Denoised image by ROF-Graph for two different values of regularization parameter t.

Chapter 4

Analysis and Discussions

In this chapter, we run some experiments with the three algorithms described in Chapter 3 on different types of real images namely Split Bregman TV-Denoising, ADMM TV-Denoising and ROF-Graph TV-Denoising, where three different images: standard test image (Lena), standard test image (House) and high attitude aerial test image (Pentagone) are tested for comparison. The images used in this thesis are the standard test images that were retrieved online from freely available image databases. By observation, the three algorithms experimented to three different images tried to recover the original image differently. For all tested images, the ROF-Graph TV-Denoising algorithm gives a good and clear recovered image comparing to the original image as shown in the Figures 4.1, 4.2 and 4.3 than the remaining two algorithms.The SB TV-Denoising algorithm gives a good approximate recovered image than the ADMM TV-Denoising algorithm.

According to the convergence plots in Figure 4.4, it is observed that the ROF-Graph algorithm converges well comparatively to others as a relative error is decreasing. The SB algorithm comes to be the second in convergence but first in fast convergence as it needs few iterations to achieve the target for all tested images in Figure 4.4 and lastly the ADMM algorithm converges slowly to achieve target as shown also by the relative errors in Figure 4.4.

Test 1

In this section, the standard test image (Lena) of size 512×512 is tested as an original image and the observed noisy image with the additive white noise is tested by applying three different algorithms to recover the original image and the results are shown in Figure 4.1.

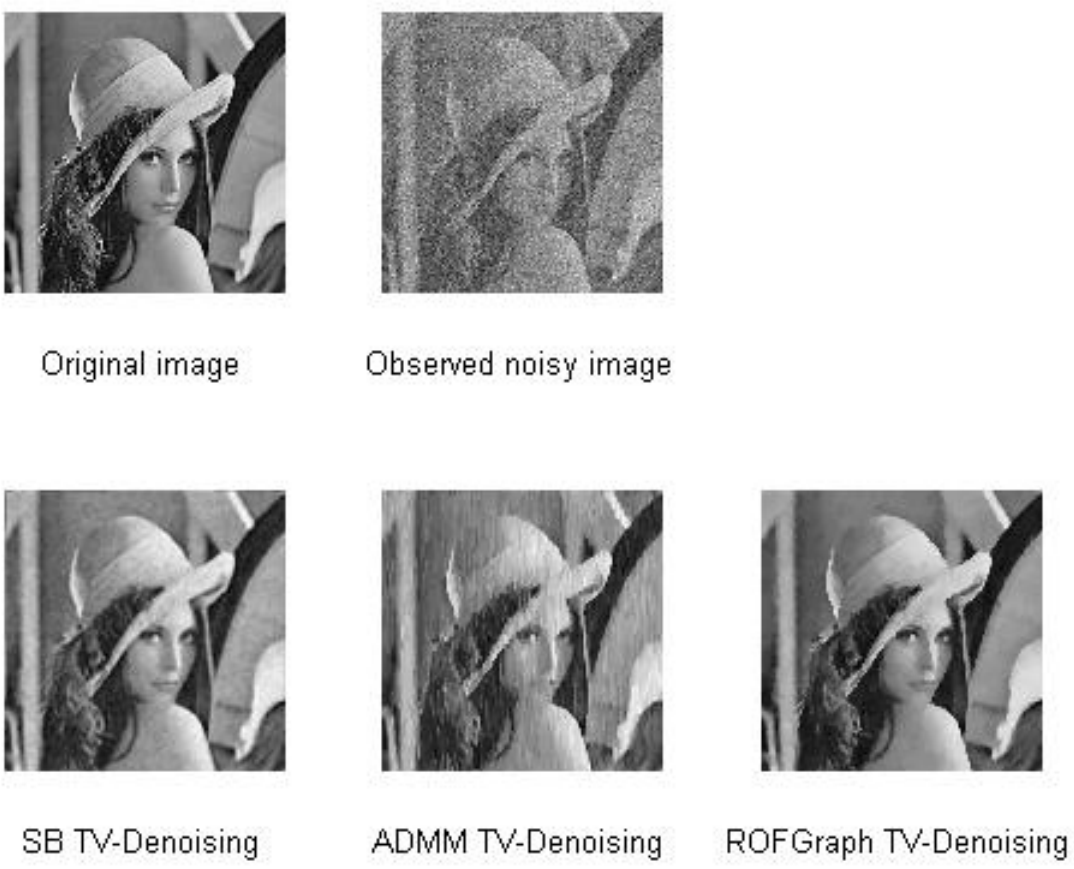

Figure 4.1: Comparison of algorithms for Lena test image.
The figure shows the Original, Observed noisy image and Denoised images of Lena test image by SB TV-Denoising, ADMM TV-Denoising and ROF-Graph TV-Denoising.

Test 2

In this section, the standard test image (House) of size 512×512 is tested as an original image and the observed noisy image with the additive white noise is tested by applying three different algorithms of TV-Denoising to recover the original image and the results are shown in Figure 4.2.

Figure 4.2: Comparison of algorithms for a House test image.
The figure shows the Original image, Observed noisy image and Denoised images of a picture of house by SB TV-Denoising, ADMM TV-Denoising and ROF-Graph TV-Denoising.

Test 3

In this section, the high attitude aerial image of Pentagone of size 512×512 is tested as an original image and the observed noisy image with the additive white noise is tested by applying three different algorithms of TV-Denoising to recover the original image and the results are shown in Figure 4.3.

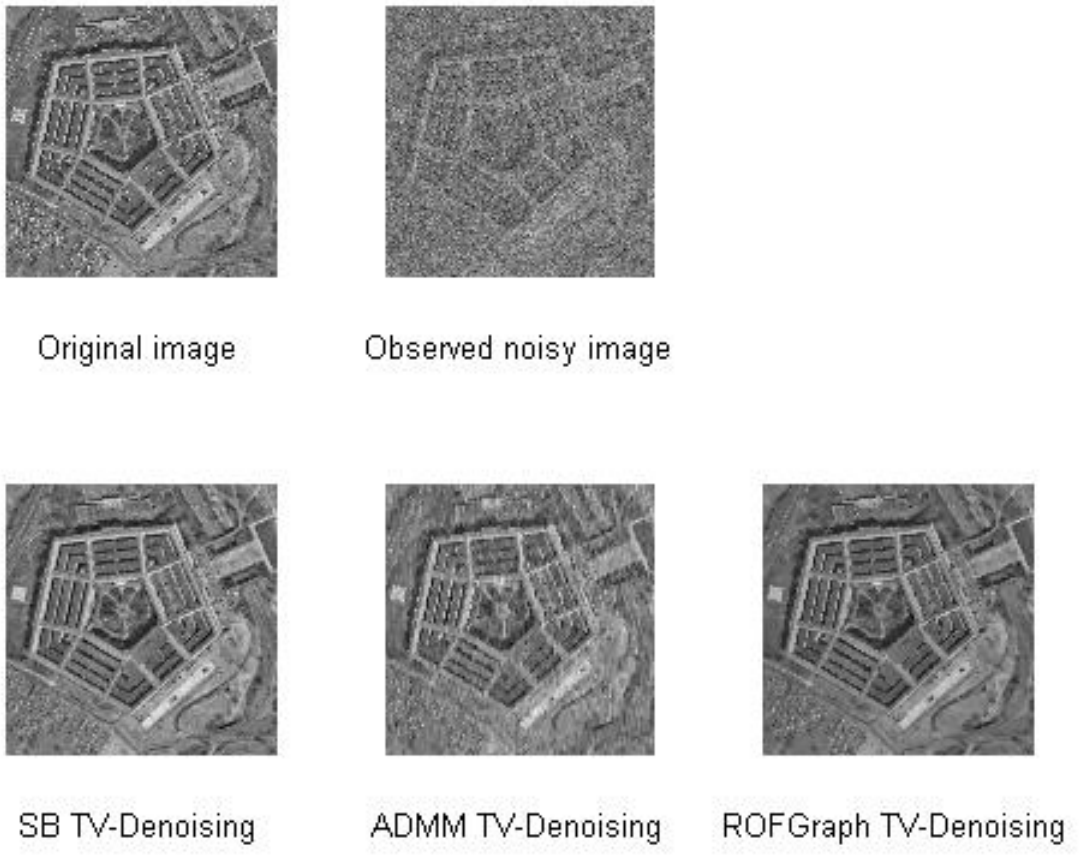

Figure 4.3: Comparison of algorithms for high attitude aerial test image.
The figure shows the Original image, Observed noisy image and Denoised images of high attitude aerial image of Pentagone by SB TV-Denoising, ADMM TV-Denoising and ROF-Graph TV-Denoising.

Comparison in convergence

This section shows the comparison of three algorithms in terms of convergence applied to three images with 100 iterations as shown in the figures below. The first plot describes a convergence test for Lena test image with relative errors 2.0601×10^{-6} for ROF-Graph, 8.45849×10^{-6} for SB and 1.55788×10^{-4} for ADMM. The second plot describes a convergence test for House test image with relative errors 2.57415×10^{-6} for ROF-Graph, 6.43004×10^{-6} for SB and 2.64989×10^{-4} for ADMM. The last plot describes a convergence test for Pentagone aerial test image with relative errors 1.37305×10^{-6} for ROF-Graph, 6.73468×10^{-6} for SB and 1.36008×10^{-4} for ADMM. In addition to this, SB algorithm converges quickly to recover the original image with relative errors 6.3×10^{-6} for Lena test image at 9 iteration, 3.8×10^{-6} for House test image at 10 iteration and 5.1×10^{-6} for Pentagone aerial test image at 9 iteration but ROF-Graph algorithm converges well to recover image comparing with two others.

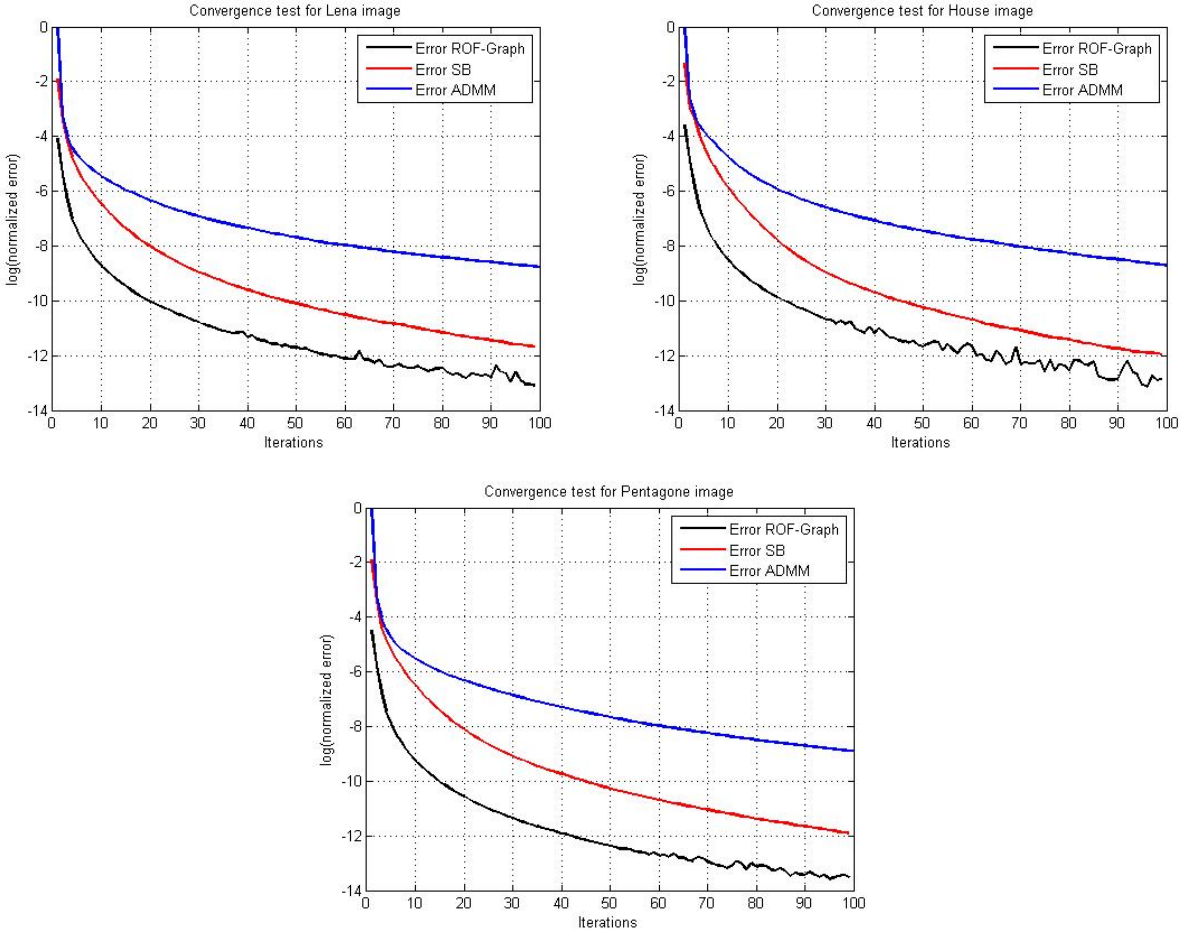

Figure 4.4: Comparison of three algorithms in convergence.

Chapter 5

Final remarks

In this chapter we give a small summary, further discussions and some final remarks. In the thesis, we present the regularization of the inverse problem specifically within the applications to Sparse signal reconstruction, total variation deblurring and total variation denoising. All of these applications are experimented using the Split Bregman algorithms. For TV Denoising, all three algorithms are tested, analysed and compared. The main output of the thesis is the analysis and experimental comparison of these methods.

For the sparse signal reconstruction, the split Bregman algorithm is developed and experimented to solve the ROF model for one dimensional and it is well observed that the original signal is reconstructed for a required regularization parameter λ with 20 iterations. For the total variation deconvolution, the split Bregman algorithm is developed and experimented to solve the proposed model for 2-dimension with the linear operator in the model of observed image. The observed blurred image is deblurred to recover the original image with a required regularization parameter λ which tends to have a good recovered image when it is large and this needs only few iterations.

Finally, for the total variation denoising, the split Bregman algorithm is experimented to solve the ROF model. An observed noised image with additive Gaussian white noise is denoised to recover the original image with a required regularization parameter λ which tends to have a good recovered image when it is small but not too small and also this does not need many iterations for denoising.

Comparing the Split Bregman algorithms for TV-Denoising with the ADMM TV-Denoising algorithm and ROF-Graph TV-Denoising algorithm, it is well observed that the Split Bregman algorithm for TV Denoising is very fast to converge than these last two algorithms (ADMM and ROF-Graph) as shown in Chapter 3, Split Bregman followed by ADMM TV-Denoising needs shorter running time for the same number of iterations than the ROF-Graph TV-Denoising. However, it is also observed that the ROF-Graph TV-Denoising

recovers well the original image within a smaller number of iterations than others.

In the future, a more theoretical analysis should be made, which takes into account the exact rates of convergence. It is also recommended that analysis and comparison with other methods could be made. It appears further, that various ℓ_1- regularized problems, other than signal and image reconstruction problems, can be carried out using these algorithms.

Bibliography

[1] Alka Pandey, D. K. S. (2015). Analysis of noise models in digital image processing.

[2] Bertsekas, D., Nedi, A., Ozdaglar, A., et al. (2003). Convex analysis and optimization.

[3] Boyd, S. and Vandenberghe, L. (2009). *Convex Optimization, seventh printing with corrections ed.* Cambridge University Press.

[4] Campisi, P. and Egiazarian, K. (2007). *Blind image deconvolution: theory and applications.* CRC press.

[5] Caselles, V., Chambolle, A., Novaga, M., et al. (2011). Regularity for solutions of the total variation denoising problem. *Revista Matemática Iberoamericana*, 27.

[6] Dimri, V. (2013). *Deconvolution and inverse theory: application to geophysical problems.* Elsevier.

[7] Elad, M. (2010). *Sparse and Redundant Representations, From Theory to Applications in Signal and Image Processing.* Springer Science+Business Media, LLC.

[8] Getreuer, P. (2012a). Rudin-osher-fatemi total variation denoising using split bregman. *Image Processing On Line*.

[9] Getreuer, P. (2012b). Total variation deconvolution using split bregman. *Image Processing On Line*.

[10] Gilles, J. (2011). The bregman cookbook.

[11] Goldstein, T. and Osher, S. (2009). The split bregman method for l1-regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343.

[12] Meila, M. Conjugate function, bregman divergences, exponential family models, stat 538 lecture 7.

[13] Niyobuhungiro, J. and Setterqvist, E. (2014). A new reiterative algorithm for the rudin-osher-fatemi denoising model on the graph. In *The 2nd International Conference on Intelligent Systems and Image Processing 2014, ICISIP2014, Kitakyushu, Japan, September 26-29, 2014*.

[14] Royden, H. L. (1968). Real analysis.

[15] Rudin, L. I. and Osher, S. (1994). Total variation based image restoration with free local constraints. In *Image Processing, 1994. Proceedings. ICIP-94., IEEE International Conference*, volume 1.

[16] Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60.

[17] Scherzer, O., Grasmair, M., Grossauer, H., Haltmeier, M., and Lenzen, F. (2009). *Variational methods in imaging*, volume 320. Springer.

[18] Selesnick, I. (2012). Penalty and shrinkage functions for sparse signal processing. *Connexions*.

[19] Shi, Y. and Chang, Q. (2013). Efficient algorithm for isotropic and anisotropic total variation deblurring and denoising. *Journal of Applied Mathematics*, 2013.

[20] Siltanen, S. (2009). Mat-52506 inverse problems. *Lecture Note, Version*, 10.

[21] Stephen, B., Neal, P., Eric, C., Borja, P., and Jonathan, E. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn*, 3.

[22] Sun, W. and Yuan, Y.-X. (2006). *Optimization theory and methods: nonlinear programming*, volume 1. Springer Science & Business Media.

[23] Vogel, C. R. (2002). *Computational methods for inverse problems*, volume 23. Siam.

[24] Yan, J. (2011). Isotropic and anisotropic total variation from a matrix point of view.

[25] ZIEMER, W. (1989). Weakly differentiable functions, sobolev spaces and functions of bounded variation. *Grad. Texts in Math.*, 120.

Appendix A

Appendix

Appendix I: Anisotropic total variation matlab code.

The following are the Matlab codes to execute the Anisotropic total variation in (2.16) of a matrix F defined randomly and they are edited from [24].

```
function aTV_F = Anisotropic_total_variation_of_F
% Compute the anisotropic total variation of matrix F
%%
% Example:
clear all; close all; clc
N=5;
F=randn(N)
%%
% Anisotropic_total_variation_of_F
[M,N] = size(F);
DxF = -diff(F,[],1);
DxF = [DxF;zeros(1,N)] %Variation of F in x direction
DyF = -diff(F,[],2);
DyF = [DyF zeros(M,1)] %Variation of F in y direction
%%
aTV_F = sum(sum(abs(DxF)+abs(DyF)))
```

Appendix II: Isotropic total variation matlab code.

The following are the matlab codes to execute the Anisotropic total variation in (2.18) of a matrix F defined randomly and they are edited from [24].

```
function iTV_F = Isotropic_total_variation_of_F
% Compute the isotropic total variation of matrix F
%%
% Example:
clear all; close all; clc
N=5;
F=rand(N)
%%
% Isotropic_total_variation_of_F
[M,N] = size(F);
DxF = diff(F,[],1);
DxF= [DxF;zeros(1,N)] %Variation of F in x direction
DyF = diff(F,[],2);
DyF = [DyF zeros(M,1)] %Variation of F in y direction
%%
iTV_F = sum(sum(sqrt(DxF.^2+DyF.^2)))
```

Appendix III: Sparse Signal reconstruction matlab codes.

I. The following matlab codes are used as the main program (function) to execute the algorithm 2 and they are edited from [10].

```
%% MAIN PROGRAM FOR SPARSE SIGMAL RECONSTRUCTION BY L1 Split Bregman Iteration
function x=L1_SplitBregmanIteration(y,A,lambda,alpha,Niter)
% This is compute the solution: u = arg min ||x||_1+0.5*lambda||Ax-y||_2^2
% by using the Split Bregman Iteration with:
% A: square matrix and a linear operator
% y: measured data or observed signal
%% lambda: regularization coefficient
```

```
% alpha: "spliting" regularization coefficient
% Niter: maximum number of iteration
%===============================================
N1=size(y,1)-4;   %This 4 due to lenght of signal M and L
d=zeros(N1,1); b=zeros(N1,1); x=zeros(N1,1); Z=zeros(N1,1);
Ft=lambda.*A'*y; B1=lambda*(A'*A); B2=alpha*eye(N1);
B=inv(B1+B2);       %B=inv(mu*(A'*A)+lambda*eye(N));
err=norm(y,2); tol=1e-3*err; K=0;
while ((err>tol) && (K<Niter)),
    K=K+1; xp=x; x=B*(Ft+alpha*(d-b)); tmp=x+b;
    d=sign(tmp).*max(Z,abs(tmp)-1/alpha); b=tmp-d; err=norm(x-xp,2);
end
```

II. The following matlab codes are used to create the original signal, observed signal and to evaluate the reconstructed Sparse signal.

```
%% SPARSE SIGNAL RECONSRUCTION BY Split Bregman iteration
%% ======================================================
clc; close all; clear all
%% Create spike signal (ORIGINAL SIGNAL)
N = 150;                    % N : length of signal
x= zeros(N,1);
k = [10 23 35 58 83 95 115 120 145];
a = [2.5 -1 2 -2 0.7 1.6 -1.5 0.3 -0.4];  % a : spike amplitudes
x(k) = a;                                 % x :Original signal
figure(1)
plot(x); legend('Original Sparse Signal: x'); ylim1 = [-3 3]; ylim(ylim1)
%% Creation of Observed signal
% The simulated observed signal is obtained by convolving
L = 5;        % L : length of impulse response
sigma = 0.03; % sigma : standard deviation of AWGN
h = ones(L,1)/L;        % h=1/L : 5-point impulse response (PSF)
```

```
M = N+L-1;              % M : length of observed signal
eta = sigma * randn(M,1); % eta : additive zero-mean Gaussian noise (AWGN)
y = conv(h,x) + eta;      % y : observed data
figure(2)
plot(y); xlim([0 M]); legend('Observed signal: y');
%% Create convolution matrix A such that y=Ax+noise
A = sparse(M,N);
e = ones(N,1);
for i = 0:L-1
    A = A + spdiags(h(i+1)*e, -i, M, N);   % A : convolution matrix (sparse)
end
%% L1 minimization
Niter=20;  %20 iterations are enough for reconstruction
alpha=1; err=norm(y,2); tol=1e-3*err
%% Reconstructed Sparse Signal for different values of lambda
lambda1=5; lambda2=15; lambda3=20; lambda4=30;
RecX1=L1_SplitBregmanIteration(y,A,lambda1,alpha,Niter);
RecX2=L1_SplitBregmanIteration(y,A,lambda2,alpha,Niter);
RecX3=L1_SplitBregmanIteration(y,A,lambda3,alpha,Niter);
RecX4=L1_SplitBregmanIteration(y,A,lambda4,alpha,Niter);
%% Plots
figure(3)
subplot(2,2,1); plot(RecX1); ylim(ylim1); legend('For \lambda=5')
subplot(2,2,2); plot(RecX2); ylim(ylim1); legend('For \lambda=15')
subplot(2,2,3); plot(RecX3); ylim(ylim1); legend('For \lambda=20')
subplot(2,2,4); plot(RecX4); ylim(ylim1);legend('For \lambda=30')
%% Comparison of Observed (y) and Reconstructed (x) SIgnals
figure(4)
plot(y,'g'); hold on; plot(RecX4, 'm'); xlim([0 M]); ylim(ylim1)
legend('Observed signal','Reconstructed signal')
```

Appendix IV: TV Deconvolution matlab codes.

I. The following are the main matlab codes of the Total Variation deconvolution established in algorithm 3 and they are edited from [10].

```
function u=ATV_NB_Deconvolution(f,A,lambda,alp,Niter,typeKernel)
%=========================================================================
%% This function performs the minimization of
% u=arg min |D_x u|_1+|D_y u|_1+0.5*lambda*||Au-f||_2^2 (Anisotropic case)
% f = blurry image
% A = convolution kernel
% lambda = regularization parameter
% alpha =split regularization modelity parameter
% Niter = maximum number of iterations
%% =========================================================================
[M,N]=size(f);
%constants initialization
f=double(f); dx=zeros(M,N); dy=zeros(M,N); bx=zeros(M,N); by=zeros(M,N);
u=f; Z=zeros(M,N);
%% Fourier constants initialization
K=0;
if typeKernel==0
    A_init=zeros(M,N); [H,L]=size(A);
    A_init([end+1-floor(H/2):end,1:ceil(H/2)],
    [end+1-floor(L/2):end,1:ceil(L/2)]) = A;
    A_tansf=fft2(A_init);
else
    A_tansf = A;
end
%% Fourier transform initialization
D = zeros(M,N);
D([end,1,2],[end,1,2]) = [0,1,0;1,-4,1;0,1,0];
FD=fft2(D);
```

```
%%
FW=((lambda/alp)*abs(A_tansf).^2-real(FD)).^-1;
FF=(lambda/alp)*conj(A_tansf).*fft2(f);
%% Bregman iterations
err=norm(f(:),2); tol=1e-3*err;
while ((err>tol) && (K<Niter)),
    K=K+1; tx=dx-bx; ty=dy-by; up=u;
    %Update u
    u=real(ifft2(FW.*(FF-fft2(tx-tx(:,[1,1:N-1])+ty-ty([1,1:M-1],:)))));
    ux=u-u(:,[1,1:N-1]); uy=u-u([1,1:M-1],:);
    %Update dx
    tmpx=ux+bx; dx=sign(tmpx).*max(Z,abs(tmpx)-1/alp);
    %Update dy
    tmpy=uy+by; dy=sign(tmpy).*max(Z,abs(tmpy)-1/alp);
    %Update bx and by
    bx=tmpx-dx; by=tmpy-dy; err=sum(sum((up-u).^2));
end
```

II. The following are the matlab codes to implement the Anisotropic Total Variation Deconvolution and they are edited from [10].

```
%% APPLICATION TO TV DECONVOLUTION (Anisotropic case)
%==========================================================
clear all;close all; clc
%% Original image
x=imread('Lena512.png');
%% DEBLURRING EXPERIMENTS
W0=size(x,2); H0=size(x,1); W1=20; H1=20;
P=[x(H1:-1:1,W1:-1:1) x(H1:-1:1,:) x(H1:-1:1,end:-1:end-W1+1);
   x(:,W1:-1:1) x x(:,end:-1:end-W1+1) ;
   x(end:-1:end-H1+1,W1:-1:1) x(end:-1:end-H1+1,:) x(end:-1:end-H1+1,end:-1:end-W1+1)];
%% Gaussian noise with mean=30 and variance=30.
h=fspecial('gaussian',30,30);
```

```
y=conv2(double(P),double(h));
%% PROCESS OF DEBLURRING
lambda1=10; lambda2=50; lambda3=100000000; alp=0.1;
Niter=5;    %Max iterations
%% Deblurred images for different values of lambda
x1=ATV_NB_Deconvolution(y,h,lambda1,alp,Niter,0);
x2=ATV_NB_Deconvolution(y,h,lambda2,alp,Niter,0);
x3=ATV_NB_Deconvolution(y,h,lambda3,alp,Niter,0);
%% Blurred (y)
y=y(H1+1:H1+H0,W1+1:W1+W0);
%% Plots of Original image
figure(1)
imshow(x,[]);xlabel('ORIGINAL IMAGE');
%% Plots of deblurred images for different lambda
figure(2)
subplot(2,2,1); imshow(abs(y),[]);xlabel('Blur image');
subplot(2,2,2); imshow(abs(x1),[]);xlabel('Deblurred image for \lambda=10');
subplot(2,2,3); imshow(abs(x2),[]);xlabel('Deblurred image for \lambda=50');
subplot(2,2,4); imshow(abs(x3),[]);
xlabel('Deblurred image for \lambda=100000000');
```

Appendix V: TV Denoising matlab codes.

I. The following are the main matlab codes of the Total Variation denoising established in algorithm (4) and they are edited from [10].

```
function u=ATV_Denoising(f,lambda,alpha,Niter)
%% ======================================================================
%% This function performs the minimization of
% u=arg min |D_x u|_1+|D_y u|_1+0.5*lambda*||u-f||_2^2 (Anisotropic case)
%y= noisy image
%lambda = regularization parameter
%alpha = regularization for the Bregman variable
```

```
%Niter = maximum number of iterations
%% =====================================================================
[M,N]=size(f);
%constants initialization
f=double(f); dx=zeros(M,N); dy=zeros(M,N); bx=zeros(M,N); by=zeros(M,N);
u=f; Z=zeros(M,N);
%% Fourier constants initialization
A_init=zeros(M,N); A_init(1,1) = 1; A_tansf=fft2(A_init);
%% Fourier transform initialization
D = zeros(M,N);
D([end,1,2],[end,1,2]) = [0,1,0;1,-4,1;0,1,0];
FD=fft2(D);
%%
FW=((lambda/alpha)*abs(A_tansf).^2-real(FD)).^-1;
FF=(lambda/alpha)*conj(A_tansf).*fft2(f); K=1;
%% Bregman iterations
err=norm(f(:),2); tol=1e-3*norm(f(:),2);
while ((err>tol) && (K<Niter)),
    K=K+1; tx=dx-bx; ty=dy-by; up=u;
    %Update u
    u=real(ifft2(FW.*(FF-fft2(tx-tx(:,[1,1:N-1])+ty-ty([1,1:M-1],:)))));
    ux=u-u(:,[1,1:N-1]); uy=u-u([1,1:M-1],:);
    %Update dx
    tmpx=ux+bx; dx=sign(tmpx).*max(Z,abs(tmpx)-1/alpha);
    %Update dy
    tmpy=uy+by; dy=sign(tmpy).*max(Z,abs(tmpy)-1/alpha);
    %Update bx and by
    bx=tmpx-dx; by=tmpy-dy; err=sum(sum((up-u).^2));
end
```

II. The following are the matlab codes to implement the Anisotropic Total Variation Denoising and they are edited from [10].

```
%% APPLICATION TO TV DENOISING (Anisotropic case)
%============================
clc; close all; clear all
%% Original image
x = double(imread('Lena512','png'));
%======================
%% DENOISING EXPERIMENTS
%======================
%% Noisy image
[N,M] = size(x);
eta=40*randn(N,M); %Additive white noise
y = x+ eta;  %Noisy image with additive white noise
%% PROCESS OF DENOISING
lambda1=3; lambda2=0.5; lambda3=0.1;alpha = 1;
Niter = 5; %Maximum iterations
%% Denoisy image for different values of lambda
y1=ATV_Denoising(y,lambda1,alpha,Niter);
y2=ATV_Denoising(y,lambda2,alpha,Niter);
y3=ATV_Denoising(y,lambda3,alpha,Niter);
%% Plots of denoisy image for different lambda
subplot(2,2,1);imshow(y,[]);xlabel('Noisy image');
subplot(2,2,2);imshow(y1,[]);xlabel('Denoised image for \lambda=3');
subplot(2,2,3);imshow(y2,[]);xlabel('Denoised image for \lambda=0.5');
subplot(2,2,4);imshow(y3,[]);xlabel('Denoised image for \lambda=0.1')
```