NTNU

Norwegian University of
Science and Technology

An algorithmic Framework for Multiresolution based non-parametric Regression

Anette Fossum Morken

Master of Science in Physics and Mathematics
Submission date: June 2017
Supervisor: Markus Grasmair, IMF

Norwegian University of Science and Technology
Department of Mathematical Sciences

Preface

This thesis is the final work for my Master of Science Degree in Applied Physics and Mathematics at the Norwegian University of Science and Technology (NTNU). The thesis was written during the spring of 2017 at the Department of Mathematical Sciences.

First, I want to give a great thank you to my supervisor, Markus Grasmair at the Department of Mathematical Science, for excellent guidance and discussions during my work with the master thesis and the specialization project. Second, I want to thank all my friends and family for all the support during the rollercoaster a master degree is. A special thank goes to all the amazing people I have met during my years in Trondheim, it would not have been the same without you.

To my nephew Tobias: Nå er tante ferdig på skolen i Trondheim!

Anette Fossum Morken

Trondheim, June 2017

Abstract

We study a method to solve non-parametric regression problems in one and two dimensions with statistical multiresolution estimation. We present the non-parametric regression problem, then introduce the multiresolution norm and use it to formulate the optimization problem. We will discuss two different regularization terms, a quadratic regularization term and a total variation term. We will solve the quadratic problem in both one and two dimensions. In order the solve this problem in one dimension, we use the ADMM (alternating direction method of multipliers) and Dykstra's projection method. For the two dimensional case, we use the Douglas-Rachford method. We will consider the total variation problem only in two dimensions. To solve the total variation problem we use the Douglas-Rachford method and Chambolle's projection method. Towards the end, we will verify and test the algorithms numerically.

Sammendrag

Vi vil studere en metode for å løse ikke-parametriske problemer i en og to dimensjoner med statistisk multiresolusjonsestimering. Vi starter med å presentere det ikke-parametriske problemet og multiresolusjonsnormen, disse vil deretter bli brukt til å formulere optimeringsproblemet. To forskjellige regulariseringsledd, et kvadratiskledd og et totalvariasjonsledd, vil bli diskutert. Det kvadratiske problemet vil bli løst i både en og to dimensjoner, mens totalvariasjonsproblemet bare vil bli løst i to dimensjoner. For å løse problemet i en dimensjon vil vi bruke ADMM (alternating direction method of multipliers) og Dykstras projeksjonsmetode. For å løse de to-dimensjonale problemene vil vi bruke Douglas-Rachfords metode. I tillegg til Douglas-Rachfords metode vil vi for totalvariasjonsproblemet bruke Chambolles projeksjonsmetode. Mot slutten vil vi numerisk verifisere og teste algoritmene.

Table of Contents

Chapter 1

Introduction

Noise reduction, or denoising, is a well-known field within signal processing. A signal can be many different things. In one dimension, the signal can be audio, electromagnetic waves or radio signals. In two dimensions, the signal can be an image. In three dimensions, the signal can be a three dimensional images such as the result of a CT scan or a film, which is a time dependent sequence of images. We will work with one and two dimensional signals, but the denoising method we present also would work in higher dimensions.

Many factors can influence a signal from when the signal appears until it is received and recorded. If a signal is traveling over a large distance, the signal may lose energy. This phenomenon is called dissipation. The signal can also absorb energy or information from other signals or energy sources. In this scenario, the signal has became noisy. Often, we divide the observed data Y, which is the received signal, into two terms the original signal f and the noise ε. The observed data can then be described with the equation

$$Y = f + \varepsilon.$$

The noise is often divided into different groups, one of them being white noise. White noise is in [4] defined as a discrete signal with mean equal to zero and constant variance σ^2. If the distribution of the noise is Gaussian, then the noise is called Gaussian noise. This distribution is a widely used model for noise. It will be assumed in this thesis that the noise can be described as Gaussian noise.

We are interested in the original function f. Therefore, we want to find a reconstruction u such that $u \approx Y$. To find the reconstruction u, we need to remove the noise ε from the data points [6]. Many different methods have been developed to reconstruct the original signal f. We can for instance apply different filters, use wavelet transforms or statistical methods. A widely used method over the past decades years has been regularization. This method uses an energy minimization approach. The approach often consists in minimizing a functional with two terms, one of which models how the observed signal is derived from the original signal and the other contains information about the original signal. Such a method is linear regression, where we assume the solution to be a linear function.

In many situations, the solution is not a linear function, so linear regression is not the best method for us. However, we could assume that the solution is smooth. Then, we can formulate the problem as a constrained optimization problem with a regularization term that is the L_2-norm of the gradient of the reconstruction. If we then solves the problem such that the L_2-norm of the residual, $u - Y$, is less than a constant $\delta > 0$, we have the minimization problem

$$\min \frac{1}{2}\|\nabla u\|_2^2 \quad \text{such that} \quad \|u - Y\|_2 \leq \delta. \tag{1.1}$$

However, with this constraint We have none local evaluating of the noise and the noise is not independent. Therefore, we will not use this method either. We will formulate the problem as a constrained optimization problem, but as constraint, we will use a statistical multiresolution norm introduced by Nemirovskiy in [13]. For the regularization term, we will study two different types.

In this thesis, we will study and develop algorithms to denoise signals in both one and two dimensions. In one dimension, we will study a quadratic regularization problem. To solve the problem, we will follow the approach in [7] and solve the problem with the alternating direction method of multipliers [5] and Dykstra's projection method [2].

In two dimensions, we will study the same regularization term as in one dimension. In addition, we will study an approach where the regularization term is the total variation norm, which was introduced as a method to reduce noise by Rudin, Osher and Fatemi in [16]. For both approaches, we will develop an algorithm to solve the minimization problem based on the Douglas-Rachford splitting algorithm [5]. When we have the total variation norm as regularization term, we will use Chambolle's projection method to solve the total variation problem [3]. At the end, we will implement the algorithms in Matlab and test the methods numerically.

Chapter 2

Problem formulation

We want to remove noise from a discrete signal Y. We will start to formulate the problem as non-parametric problem. Then, we will study a method to solve the problem.

We are given the data points $Y \in \mathbb{R}^{N\times M}$, for $N \leq M$, on the two-dimensional grid

$$\Gamma = \{(x_i, y_j) \in \mathbb{R}^2 \mid x_i = ih, y_j = jh, i = 1, \ldots, N, j = 1, \ldots, M\} \subset [0, a] \times [0, b]\,,$$

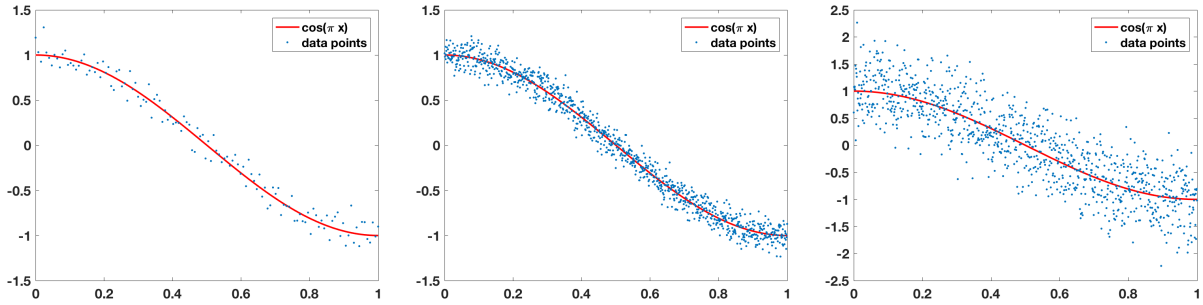
where $h = \frac{a}{N+1} = \frac{b}{M+1}$. We then assume the data points can be described in the form

$$Y_{i,j} = f(x_i, y_j) + \varepsilon_{i,j},$$

where $f : [0, a] \times [0, b] \to \mathbb{R}$ is an unknown function which we want to reconstruct. f is assumed to be a continuous function. The noise, denoted as ε, is assumed to be independent and identically distributed (i.i.d.) with Gaussian distribution, mean μ equal to zero and some variance $\sigma^2 > 0$, hence the noise is i.i.d. Gaussian. We assume that the variance of the noise is known, but the method to reconstruct f to be discussed here can also be applied in the case of unknown variance.

Three examples illustrating the data Y are plotted in figure 2.1. The examples are one dimensional, but the same principles applies to two dimensional problems. All three plots show the data points $Y = f(x) + \varepsilon$, where the original function $f(x) = \cos(\pi x)$. In subplot 2.1a we have 128 data points and the variance is 0.1, in subplot 2.1b, we have 1024 data points and the variance is 0.1 and in subplot 2.1c we have 1024 data points and the variance is 0.4. By studying the three different plots, we can see how the number of data points and the variance of the noise influence the shape of the data set.

First, we consider the influence of the number of data points. Comparing subplot 2.1a and subplot 2.1b, we see that they have the same variance, but subplot 2.1a has fewer data points than subplot 2.1b. We can also see that subplot 2.1b resembles a cosine function. However, while subplot 2.1a, might look like a cosine, it could also include additional oscillations since the distance between each point is large.



(a) $N = 128$ and $\sigma = 0.1$ **(b)** $N = 1024$ and $\sigma = 0.1$ **(c)** $N = 1024$ and $\sigma = 0.4$

Figure 2.1: The plots show the data points Y with noise as dots and the function $f(x) = \cos(\pi x)$ for $x \in (0, 1)$. f is the function we want to reconstruct from Y as a curve. In the subplots, we can see how the different number of data points and the variance impact the shape of the data points.

Second, to see how the variance can impact the shape the data points form, we compare subplot 2.1b and subplot 2.1c. In both figures we have $N = 1024$, but the functions have different variances, subplot 2.1b has variance $\sigma = 0.1$ and subplot 2.1c has variance $\sigma = 0.4$. We see that the data points are very scattered in subplot 2.1c. It is hard to see whether the data points either form a linear shape or a cosine shape. While in subplot 2.1b, the shape of the function f is more retained. Hence, the shape of the cosine function is easier to detect.

In summary, both the number of data points and the size of the variance impacts the appearance of the set of data points. The shape of the original function is better retained in the set of data points for large N and small variance, than for small N and large variance.

The aim is to find a reconstruction u of f from Y. A method to reconstruct f from Y is for instance the least squares method. This method assumes f to be affine, which is very restrictive. We want to solve as many different problems as possible, not only problems that are affine. For that reason, the least squares method is not the best method for us and therefore must consider other methods.

A better method to reconstruct u might be to use constrained optimization. A constrained method called the residual method is discussed in [8]. The optimization problem there is of the form

$$\min_u J(u) \quad \text{such that} \quad V(u, Y) \leq \gamma,$$

where J is the regularization term and V connects the observed data points and the reconstructed data. The purpose of the regularization term J is to impose some regularity properties to the reconstruction u. Which properties we want to regularize can vary, so at this moment we define it as a general (convex) functional $J(u)$. This functional will be discussed in more detail in chapter 2.2.

In the residual method, V is chosen such that the residual has the same properties as the noise. Then, we minimize $J(u)$ subject to the constraint V. To evaluate the residual it is necessary to evaluate u at the grid points. Therefore, we introduce the point evaluation

$$S_\Gamma : C^0 \to \mathbb{R}^{N \times M},$$

which samples any function u on the regular grid Γ, that is, if $u : [0, a] \times [0, b] \to \mathbb{R}$ then $(S_\Gamma u)_{i,j} = u(x_i, y_j)$. Using this notation we can express the residual as

$$r = S_\Gamma u - Y.$$

We know that Y consist of f and Gaussian noise ε. If we in addition assume the variance known, the variance of the residual should be equal or smaller than the variance of the noise. Which can be formulated as

$$\frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} (r_{i,j})^2 \leq \sigma^2 \tag{2.1}$$

and used as constraint.

We cant to formulate a method that do not require the variance of the noise known, so we will not use (2.1) as constraint. Instead of knowledge about the variance, we will use the residual r and the assumption that ε is i.i.d. Gaussian with mean $\mu = 0$ and compare the behavior of these two. The behaviour is descibed by the multiresolution norm [13].

2.1 Multiresolution norm

In this section, we will formulate first the multiresolution norm [13], and then the constraint for the optimization problem. To formulate the multiresolution norm, we will use some properties of a Gaussian random variable. The probability density of a Gaussian random variable is

$$k(y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}},$$

where σ^2 is the variance. Further, the sum of n independent Gaussians again is Gaussian distributed with variance $n\sigma^2$.

To describe the noise, we want to estimate the probability for the noise to lie with a certain distance from zero. For example, in figure 2.2, we have 100 data points which are i.i.d. Gaussian with mean $\mu = 0$ and variance $\sigma = 0.5$. Three distances from zero are also plotted in the figure, $d_1 = 0.5$, $d_2 = 1.0$ and $d_3 = 1.5$. As we can see, there are many points with distance larger than d_1 from zero, only a few with distance larger than d_2 from zero and none at all are further away from zero than d_3.

To describe this mathematically, we start with one grid point x_i, for some $i \in \{1, \ldots, N\}$ and $j \in \{1, \ldots, M\}$. The probability for the noise at one grid point $x_{i,j}$ to have larger distance from

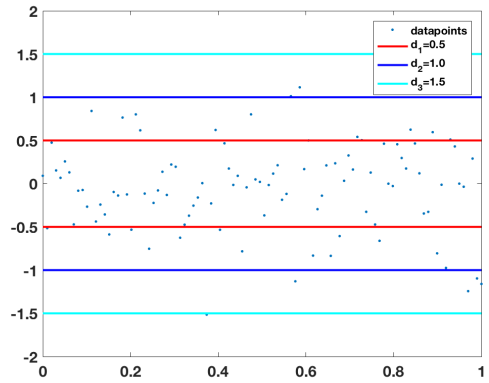


Figure 2.2: The plot shows how a set with 100 i.i.d. Gaussian random variables with mean $\mu = 0$ and variance $\sigma = 0.5$ are distributed around 0. The lines mark the distances $d_1 = 0.5$, $d_2 = 1.0$ and $d_3 = 1.5$ from 0, colored in red, blue and turquoise respectively. The sample at the point x_ν, where $\nu \in \{1, \dots, 100\}$ and $x \in (0, 1)$, has the distance $|\varepsilon_\nu|$ from 0.

zero than $C\sigma$, where $C > 0$, can be estimated as

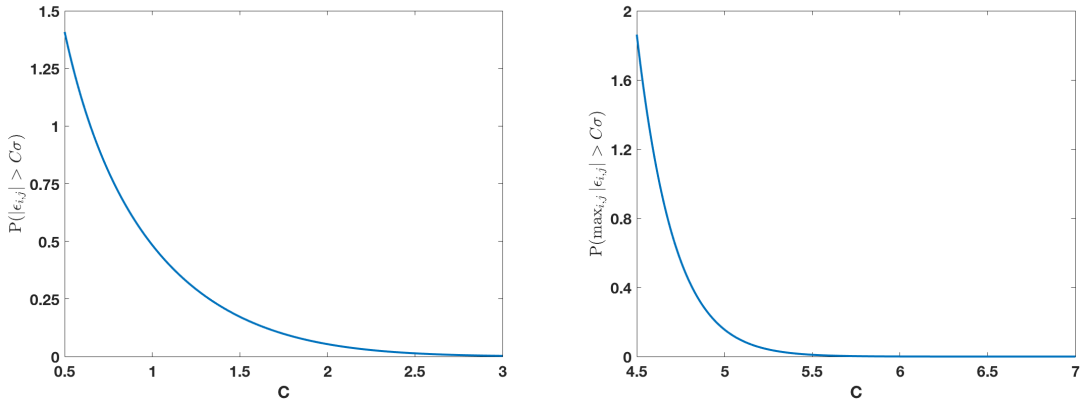
$$\begin{aligned}
\mathbb{P}(|\varepsilon_{i,j}| > C\sigma) &= \int_{-\infty}^{-C\sigma} k(y)dy + \int_{C\sigma}^{\infty} k(y)dy \\
&= 2\int_{C\sigma}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}} dy \\
&\leq 2\int_{C\sigma}^{\infty} \frac{1}{\sqrt{2\pi}} \frac{y}{C\sigma^2} e^{-\frac{y^2}{2\sigma^2}} dy \\
&= \sqrt{\frac{2}{\pi}} \frac{1}{C} e^{-\frac{C^2}{2}}.
\end{aligned} \tag{2.2}$$

This is illustrated in subplot 2.3a. As we can see, the probability for some $\varepsilon_{i,j}$ to be further away from zero than $C\sigma$ decreases fast as C grows. For instance, the probability is less than 1 when $C > 0.64$ and when $C > 2.5$ is the probability almost zero.

Now, we want to look at the largest sample of $|\varepsilon|$. We then get the probability for any $\varepsilon_{i,j}$ to be further away from zero than $C\sigma$. To find this, we find the probability for the ε with the largest distance to zero to be further away from zero than $C\sigma$, which can be estimated as

$$\begin{aligned}
\mathbb{P}(\max_{i,j} |\varepsilon_{i,j}| > C\sigma) &\leq \sum_{i=1}^{N} \sum_{j=1}^{M} \mathbb{P}(|\varepsilon_{i,j}| > C\sigma) \\
&\leq \frac{NM}{C} \sqrt{\frac{2}{\pi}} e^{-\frac{C^2}{2}}.
\end{aligned} \tag{2.3}$$

This probability is plotted for $N = M = 512$ in subplot 2.3b. As we can see, the probability behaves similar to (2.2), the further away from zero the $C\sigma$ is from zero, the lower is the probability to find any ε further from zero than $C\sigma$. The probability is less than 1 when $C > 4.62$ and when



(a) (2.2) **(b)** (2.3)

Figure 2.3: In the figure to the left, (2.2) is plotted and in the figure to the right, (2.3) is plotted, where $N = M = 512$. We see that in the plot to the left the probability is less than 1 when $C > 0.64$ and when $C > 2.5$ the probability is zero and in the plot to the right the probability is less than 1 when $C > 4.62$ and when $C > 5.5$ the probability is almost zero.

$C > 5.5$ the probability is almost zero. By comparing the two plots in figure 2.3, it is clear that the point where the probability is zero, the value of C is lower for (2.2) than for (2.3). We can then conclude that, the probability for a sample $\varepsilon_{i,j}$ to be further away from zero than $C\sigma$ is zero at a smaller C than the value of C where the probability to find the sample with largest distance from zero further away from zero than $C\sigma$ is zero.

Now, we apply the same estimates at a sum of the noise at four adjacent grid points that form a square with side lengths of two grid points. To that end, we assume $i \in \{1, \dots, N-1\}$ and $j \in \{1, \dots, M-1\}$ and consider the sum $\varepsilon_{i,j} + \varepsilon_{i+1,j} + \varepsilon_{i,j+1} + \varepsilon_{i+1,j+1}$. Since the noise is i.i.d. Gaussian, the sum is also Gaussian distributed, with variance $(2\sigma)^2$. Thus, the probability for the absolute value of this sum to be larger than $2C\sigma$ is

$$\mathbb{P}\left(|\varepsilon_{i,j} + \varepsilon_{i+1,j} + \varepsilon_{i,j+1} + \varepsilon_{i+1,j+1}| > 2C\sigma\right) \leq \sqrt{\frac{2}{\pi}}\frac{1}{C}e^{-\frac{C^2}{2}}.$$

We divide the sum by 2 and rewrite the estimate as

$$\mathbb{P}\left(\frac{1}{2}|\varepsilon_{i,j} + \varepsilon_{i+1,j} + \varepsilon_{i,j+1} + \varepsilon_{i+1,j+1}| > C\sigma\right) \leq \sqrt{\frac{2}{\pi}}\frac{1}{C}e^{-\frac{C^2}{2}}.$$

Then, taking the maximum over all possible sums of squares, we obtain

$$\begin{aligned}\mathbb{P}\left(\max_{i,j}\frac{1}{2}|\varepsilon_{i,j} + \varepsilon_{i+1,j} + \varepsilon_{i,j+1} + \varepsilon_{i+1,j+1}| > C\sigma\right) &\leq \sum_{i=1}^{N-1}\sum_{j=1}^{M-1}\mathbb{P}\left(|\varepsilon_{i,j}| > C\sigma\right) \\ &\leq \frac{(N-1)(M-1)}{C}\sqrt{\frac{2}{\pi}}e^{-\frac{C^2}{2}}.\end{aligned}$$

This result is similar to the result for one sample.

Finally, we assume side lengths of the square to be $k \in \{1, \dots, N\}$ grid points and the grid point number to be $i \in \{1, \dots, N-k+1\}$ and $j \in \{1, \dots, M-k+1\}$. We then consider the sum $\sum_{p=i}^{N-k+1} \sum_{q=j}^{M-k+1} \varepsilon_{p,q}$. Because of the properties of i.i.d Gaussians, the sum is also Gaussian and the variance is $k\sigma$. Using the same argumentation as earlier, we obtain that

$$\mathbb{P}\left(\frac{1}{k}\Big|\sum_{p=i}^{i+k-1}\sum_{q=j}^{j+k-1}\varepsilon_{p,q}\Big| > C\sigma\right) \leq \sqrt{\frac{2}{\pi}}\frac{1}{C}e^{-\frac{C^2}{2}}$$

and

$$\begin{aligned}\mathbb{P}\left(\max_{i,j}\frac{1}{k}\Big|\sum_{p=i}^{i+k-1}\sum_{q=j}^{j+k-1}\varepsilon_{p,q}\Big| > C\sigma\right) &\leq \sum_{i=1}^{N-k+1}\sum_{j=1}^{M-k+1}\mathbb{P}(|\varepsilon_{i,j}| > C\sigma) \\ &\leq \frac{(N-k+1)(M-k+1)}{C}\sqrt{\frac{2}{\pi}}e^{-\frac{C^2}{2}}. \end{aligned} \tag{2.4}$$

We want to estimate the largest value of (2.4). Therefore, we take the maximum of all side lengths and grid point numbers to obtain the estimate

$$\begin{aligned}\mathbb{P}\left(\max_{k,i,j}\frac{1}{k}\Big|\sum_{p=i}^{N-k+1}\sum_{q=j}^{M-k+1}\varepsilon_{p,q}\Big| > C\sigma\right) &\leq \sum_{k=1}^{N}\mathbb{P}\left(\max_{i,j}\frac{1}{k}\Big|\sum_{p=i}^{N-k+1}\sum_{q=j}^{M-k+1}\varepsilon_{p,q}\Big| > C\sigma\right) \\ &\leq \sum_{k=1}^{N}\frac{(N-k+1)(M-k+1)}{C}\sqrt{\frac{2}{\pi}}e^{-\frac{C^2}{2}} \\ &\leq \frac{(3MN+N-N^2)(N+1)}{6C}\sqrt{\frac{2}{\pi}}e^{-\frac{C^2}{2}} \\ &\leq \frac{MN^2}{C}\sqrt{\frac{2}{\pi}}e^{-\frac{C^2}{2}}. \end{aligned} \tag{2.5}$$

By introducing the index set

$$\hat{I} = \{(k,i,j) : k = 1, \dots, N,\ i = 1, \dots, N-k+1,\ j = 1, \dots, M-k+1\},$$

the inequality (2.5) can be rewritten to

$$\mathbb{P}\left(\max_{(k,i,j)\in\hat{I}}\frac{1}{k}\Big|\sum_{p=i}^{N-k+1}\sum_{q=j}^{M-k+1}\varepsilon_{p,q}\Big| > C\sigma\right) \leq \frac{MN^2}{C}\sqrt{\frac{2}{\pi}}e^{-\frac{C^2}{2}}. \tag{2.6}$$

From [7], we have that the term inside $\mathbb{P}$ is known as the multiresolution norm $\|\cdot\|_B$.

Definition 1. *Given a subset $I \subset \hat{I}$, we define the multiresolution norm of $v \in \mathbb{R}^{N\times M}$ as*

$$\|v\|_B := \max_{(k,i,j)\in I} \frac{1}{k} \Big| \sum_{p=i}^{N-k+1} \sum_{q=j}^{M-k+1} v_{p,q} \Big|.$$

With the multiresolution norm, we can describe the behavior of both the noise and the residual. Therefore, it is suitable to use as constraint for the optimization problem. There is one problem with the multiresolution norm, in the way the index set is defined the number of equations to solve increases fast when the number of grid points increases. Hence, time used to solve the problem increases. To reduce solution time we will use only the necessary quadrants. In particular, if $N = M = 2^m$, where $m \in \mathbb{N}$, we can use the dyadic index sets. Which is defined such that only some of the subsets of $\hat{I}$ is used. We can describe the dyadic index set as a union

$$I = \bigcup_{s=0}^{m} I_s$$

of the index sets

$$I_s = \{(k,i,j) : k = 2^s, i = p2^s + 1, j = q2^s + 1 \text{ with } p,q = 0,\dots,2^{m-s} - 1, s = 0,1,\dots,m\}.$$

More details about the dyadic subset can be found in appendix B.

Lemma 1. *If $\varepsilon \in \mathbb{R}^{N\times M}$ is a realization of an independent and identically distributed Gaussian random variable and $C \geq d\log(NM)$, where $d > 0$ is a constant, then*

$$\mathbb{P}\left(\|\varepsilon\|_B > C\sigma\right) \leq \frac{a^2 b}{h^3 d\log(\frac{ab}{h^2})} \left(\frac{ab}{h^2}\right)^{-d^2\log(\frac{\sqrt{ab}}{h})},$$

where $h = \frac{a}{N} = \frac{b}{M}$. In particular, we have

$$\mathbb{P}\left(\|\varepsilon\|_B > \sigma d\log(NM)\right) \underset{h\to 0}{\to} 0.$$

Proof. Inserting $C = d\log NM$ in (2.5), we obtain

$$\mathbb{P}\left(\|\varepsilon\|_B > \sigma d\log NM\right) \leq \frac{MN^2}{d\log(NM)} e^{-\frac{(d\log(NM))^2}{2}}.$$

Now, we insert $h = \frac{a}{N} = \frac{b}{M}$

$$\begin{aligned}
\mathbb{P}\left(\|\varepsilon\|_B > \sigma d\log NM\right) &\leq \frac{a^2 b}{h^3 d\log(\frac{ab}{h^2})} e^{-\frac{(d\log(\frac{ab}{h^2}))^2}{2}} \\
&= \frac{a^2 b}{h^3 d\log(\frac{ab}{h^2})} e^{-d^2\log(\frac{ab}{h^2})\log((\frac{ab}{h^2})^{\frac{1}{2}})} \\
&= \frac{a^2 b}{h^3 d\log(\frac{ab}{h^2})} \left(\frac{ab}{h^2}\right)^{-d^2\log(\frac{\sqrt{ab}}{h})},
\end{aligned}$$

which converges to zero when $h \to 0$. □

From lemma 1, we have that if the number of grid points on a fixed rectangle is increased, the probability for $\|\varepsilon\|_B$ to be larger than $d\sigma \log(NM)$ will go towards zero and, for large enough number of grid points, the probability will be negligible.

Lemma 2. *Let $w : [0, a] \times [0, b] \to \mathbb{R}$ be a continuous function such that $w \neq 0$. Then there exists some $c > 0$ such that*

$$\|S_\Gamma w\|_B \geq \frac{c}{h},$$

if h is sufficiently small.

Proof. There exists some $\bar{x} \in (0, a)$ and $\bar{y} \in (0, b)$ such that $|w(\bar{x}, \bar{y})| \neq 0$. Assume without loss of generality that $w(\bar{x}, \bar{y}) > 0$. Then there exist some $c > 0$ and $\delta > 0$ such that $w(x, y) > c$ for all $x \in [\bar{x} - \delta, \bar{x} + \delta]$ and $y \in [\bar{y} - \delta, \bar{y} + \delta]$. As a consequence

$$\begin{aligned}\|S_\Gamma w\|_B &= \max_{(k,i,j)\in I} \frac{1}{k} \Big| \sum_{p=i}^{N-k+1} \sum_{q=j}^{M-k+1} S_\Gamma w_{p,q} \Big| \\ &\geq \max_{\substack{(k,i,j)\in I \\ x_i \geq \bar{x}-\delta \\ x_{i+k-1} \leq \bar{x}+\delta \\ y_j \geq \bar{y}-\delta \\ y_{j+k-1} \leq \bar{y}+\delta}} \Big| \frac{1}{\sqrt{k^2}} \sum_{p=i}^{N-k+1} \sum_{q=j}^{M-k+1} w(x_{p,q}) \Big| \geq c \max_{\substack{(k,i,j)\in I \\ x_i \geq \bar{x}-\delta \\ x_{i+k-1} \leq \bar{x}+\delta \\ y_j \geq \bar{y}-\delta \\ y_{j+k-1} \leq \bar{y}+\delta}} |k|.\end{aligned}$$

Which is equivalent to maximizing the side lengths k of a square such that there exists some

$$i \in \{1, \ldots, N - k + 1\} \quad \text{with} \quad x_i \geq \bar{x} - \delta \quad \text{and} \quad x_{i+k-1} \leq \bar{x} + \delta, \tag{2.7}$$

and there exists some

$$j \in \{1, \ldots, M - k + 1\} \quad \text{with} \quad y_j \geq \bar{y} - \delta \quad \text{and} \quad y_{j+k-1} \leq \bar{y} + \delta. \tag{2.8}$$

In order to estimate the maximum, we want to find a lower bound for the side length k of squares in the square $[\bar{x} - \delta, \bar{x} + \delta] \times [\bar{y} - \delta, \bar{y} + \delta]$. For this purpose we use the two conditions (2.7) and (2.8). We start with (2.7), which is the x-direction. The interval in x-direction overlays the grid $[0, a]$ as illustrated in figure 2.4. The line on the top in the figure is a part of the grid Γ with N grid points with distance $h = \frac{a}{N+1}$ and the line at the bottom is the interval $[\bar{x} - \delta, \bar{x} + \delta]$ with length 2δ. Then k is the number of grid points from the top line that fits into the interval $[\bar{x} - \delta, \bar{x} + \delta]$. To find k, we find the difference between the index of the first grid point x_i in $[\bar{x} - \delta, \bar{x} + \delta]$, which is the point x_i with

$$i = \min\{l : x_l \in [\bar{x} - \delta, \bar{x} + \delta]\},$$

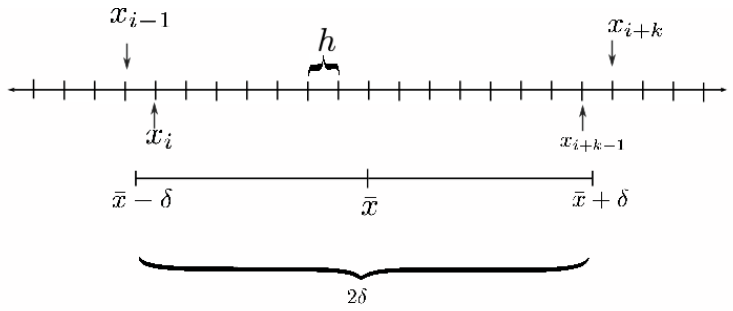


Figure 2.4: The upper line in the figure is a part of the grid Γ with grid size h. The bottom line is the interval $[\bar{x}-\delta, \bar{x}+\delta]$ with size 2δ. The grid point x_i is the grid point at most to the left in the interval $[\bar{x}-\delta, \bar{x}+\delta]$ and the grid point x_{i+k-1} is the grid point at most to the right in the interval $[\bar{x}-\delta, \bar{x}+\delta]$.

and the last grid point x_{i+k-1}, i.e. the point x_{i+k-1} with

$$i+k-1 = \max\{l : x_l \in [\bar{x}-\delta, \bar{x}+\delta]\}.$$

Then, the distance between $x_{i-1} < \bar{x}-\delta$, $x_{i+k} > \bar{x}+\delta$ is

$$|x_{i+k} - x_{i-1}| = (k+1)h > 2\delta$$

Thus, the number of grid points on the interval $[\bar{x}-\delta, \bar{x}+\delta]$ is

$$k > \frac{2\delta}{h} - 1.$$

For the second condition we apply the same method and the obtain the same result

$$k > \frac{2\delta}{h} - 1.$$

The result for k are the same in both directions are and both conditions (2.7) and (2.8) are satisfied. The result for the maximum is then

$$\max_{\substack{(k,i,j)\in I \\ x_i \geq \bar{x}-\delta \\ x_{i+k-1} \leq \bar{x}+\delta \\ y_j \geq \bar{y}-\delta \\ y_{j+k-1} \leq \bar{y}+\delta}} |k| \geq \frac{2\delta}{h} - 1 \geq \frac{\delta}{h} \quad \text{if } h \leq \delta.$$

Then if the multiresolution norm contains any continuous parts the norm becomes

$$\|S_\Gamma w\|_B \geq c \max_{\substack{(i,j,k)\in I \\ x_i \geq \bar{x}-\delta \\ x_{i+k-1} \leq \bar{x}+\delta \\ y_j \geq \bar{y}-\delta \\ y_{j+k-1} \leq \bar{y}+\delta}} |k| \geq \frac{c\delta}{h} \quad \text{if } h \leq \delta.$$

□

From lemma 2, we obtain that the multiresolution norm will behave linearly with the number of grid points in each direction if the samples consist of any continuous functions. Therefore, if the residual $r = S_\Gamma u - Y$ contains any continuous parts, we can expect that $\|r\|_B \gtrsim \frac{c}{h}$, where $c = \|r\|_\infty$. On the other hand, from lemma 1, we obtain that the multiresolution norm will behave logarithmic when the number of grid points increases if the samples do not consist of any continuous functions. For very small continuous functions, the norm struggles with separating the continuous functions and the noise. Hence, if $\|r\|_B \leq \log(\frac{ab}{h^2})$, we can expect that most of the continuous parts are removed from r. Since we aim to remove as many continuous functions as possible from r, we cam use lemma 1 and lemma 2 to formulate a constraint for the optimization problem. We formulate this constraint as

$$\|r\|_B \leq \gamma_h,$$

where γ_h is the multiresolution bound and chosen such that lemma 1 and lemma 2 are satisfied. The lemmas are satisfied when

$$\log(\frac{ab}{h^2}) \leq \gamma_h \leq \frac{c}{h}. \tag{2.9}$$

Now, we should be able to separate the noise and the original function such that the residual almost not contain any continuous functions.

2.2 Regularization

The purpose of the regularization term J is to impose some regularity properties to the reconstruction u. We will now study two different convex functionals to use as regularization terms, one where we take the L_2-norm of the gradient of u and one where we take the L_1-norm of the gradient of u, also known as the total variation norm of u [16].

Before we do so, we will discretize u and denote the discretized u as $u_{i,j}$, where $i = 1, \ldots, N$ and $j = 1, \ldots, M$, and the gradient $\nabla : \mathbb{R}^\Gamma \to \mathbb{R}^\Gamma \times \mathbb{R}^\Gamma$ is defined as $(\nabla u)_{i,j} = (\nabla u^1_{i,j}, \nabla u^2_{i,j})$, where

$$\nabla u^1_{i,j} = \begin{cases} u_{i+1,j} - u_{i,j} & \text{if } i < N \\ 0 & \text{if } i = N \end{cases} \quad \text{and} \quad \nabla u^2_{i,j} = \begin{cases} u_{i,j+1} - u_{i,j} & \text{if } j < M \\ 0 & \text{if } j = M \end{cases}.$$

We ignore the term $\frac{1}{h}$ in the definition of the gradient because it is only a constant and we will use it to solve a constrained optimization problem. Henceforward, we will use the discretized form of u.

First, we study quadratic regularization and the convex functional is then

$$J(u) = \frac{1}{2}\|\nabla u\|_2^2 = \sum_{i,j=1}^{N,M} \left((\nabla u^1_{i,j})^2 + (\nabla u^2_{i,j})^2 \right). \tag{2.10}$$

This regularization term is a very tempting functional, since it is both continuous and differentiable, and even quadratic, therefore often easy to minimize. The problem with this functional is that it

does not manage discontinuities very well. If the original function f has any discontinuities, these will be smoothed out in the reconstruction u.

The second regularization term we will discuss is the L_1-norm of the gradient of u, also called the total variation norm. As a tool in denoising, this norm was introduced by Rudin, Osher and Fatemi in [16]. The total variation norm is defined as

$$\|\nabla u\|_1 = \sum_{i,j=1}^{N,M} |\nabla u_{i,j}| = \sum_{i,j=1}^{N,M} \sqrt{(\nabla u_{i,j}^1)^2 + (\nabla u_{i,j}^2)^2}. \tag{2.11}$$

The regularization term then becomes

$$J(u) = \|\nabla u\|_1. \tag{2.12}$$

This functional is often harder to minimize, since it is not differentiable when u is zero. On the other hand, it manages discontinuities and edges, and partly smooth functions will be better reconstructed than by the quadratic regularization term. None of the convex functionals manage oscillations, so areas with oscillations in the function will give us problems.

The difference between these two functionals is the L_2 and L_1-norm. The L_2-norm is defined as

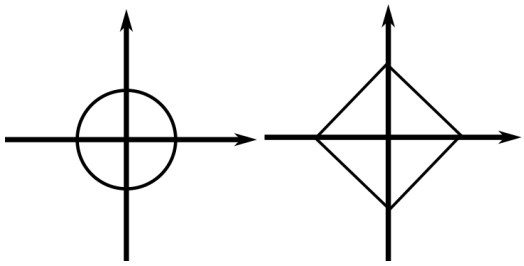
$$\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$$

and the L_1-norm is defined as

$$\|x\|_1 = |x_1| + |x_2| + \ldots + |x_n|.$$

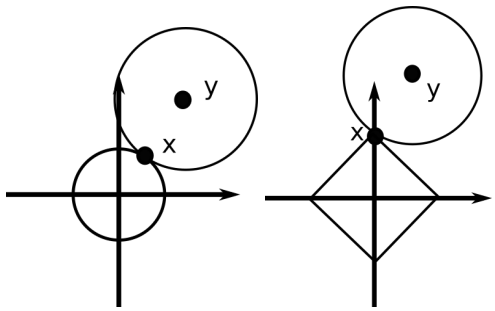
We can illustrate these two norms with their isosurfaces, which are shown in figure 2.5. As we can see in the figures, the isosurface for the L_2-norm is a circle and the the isosurface for the L_1-norm is a square with the corners at the axis. In regularization we often want to find the the shortest distance between a point $y = (y_1, y_2)$ and a $x = (x_1, x_2)$ on the isosurface. We can formulate this as a minimization problem. For the L_2-norm it is often formulated as

$$\min_x \frac{\alpha}{2}\|x\|_2^2 + \|x - y\|_2^2 \tag{2.13}$$



(a) L_2 **(b)** L_1

Figure 2.5: The figures show the isosurface for the L_2 and L_1 norm.



(a) L_2 **(b)** L_1

Figure 2.6: The figures show the shortest distance between the point y and the isosurface for the L_2 and L_1 norm.

and for the L_1-norm it is often formulated as

$$\min_x \alpha\|x\|_1 + \|x - y\|_2^2.$$

The solution of these two minimization problems is illustrated in figure 2.6. As we can see for (2.13), none of the components of the point x can be zero unless the point y lies at one of the axes. On the other hand, for (2.2), shortest distance between the isosurface and the point y can often be such that one of the components of x is zero, without y lying on one of the axes. This gives the L_1 norm the opportunity to be zero at all other places than these have something important happens.

Another difference between the L_2-norm and the L_1-norm is that large values for x_i is harder punished by the L_2-norm, than by the L_1-norm. The reason is that the L_2-norm grows faster than the L_1-norm. So, when the L_2-norm reconstructs a discontinuity, it is cheaper to have a small gradient over a larger area, than have a large gradient at the discontinuity. For the L_1-norm, it is the opposite, it is cheaper a large gradient at the discontinuity, than have a small gradient over a larger area. This property is the reason for the L_1-norm to favor discontinuities.

To illustrate the difference in the solution for the regularization terms (2.10) and (2.12) we formulate the minimization problems as

$$\min_u \frac{\alpha}{2}\|\nabla u\|_2^2 + \frac{1}{2}\|u - Y\|_2^2, \tag{2.14}$$

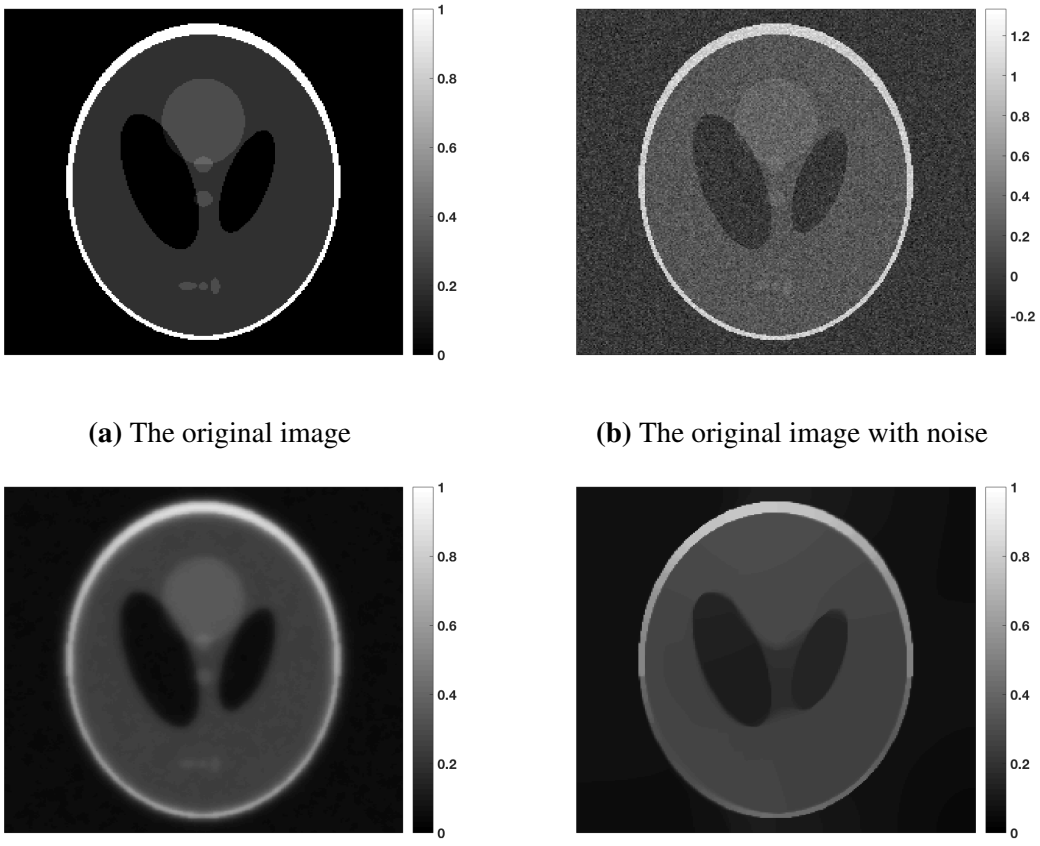
where Y is a noisy image and $\alpha > 0$ is a weighting parameter, for the regularization term (2.10) and

$$\min_u \mu\|\nabla u\|_1 + \frac{1}{2}\|u - Y\|_2^2, \tag{2.15}$$

where $\mu > 0$ is a weighting parameter, for the regularization term (2.12). The solution of (2.14) is easy to find because the equation is differentiable with respect to u. The solution is then found where the derivative solved for u is zero. We then have

$$(-\alpha\Delta + I)u = Y,$$

where I is the identity matrix, which can be solved directly. The solution of (2.15) is harder to find since it is not differentiable at $\nabla u = 0$. So, to find the solution, we must use iterative methods. We will use Chambolle's projection method with $\mu = 1$, which will be presented in chapter 3.5. Figure 2.7 shows the reconstructions and the original image with and without noise, with variance $\sigma = 0.1$. The solutions of (2.14) and (2.15) are shown in subfigure 2.7c and subfigure 2.7d, respectively. As we can see in subfigure 2.7c, the discontinuities are blurred when (2.14) is minimized. In the solution for (2.15), the discontinuities are sharper, but some of the details are missing.



(a) The original image

(b) The original image with noise

(c) Reconstructed with quadratic regularization and $\alpha = 10^{-7}$.

(d) Reconstructed with total variation.

Figure 2.7: The images in the figure shows a image called phantom. The two images at the top are the original image with and without noise, with variance $\sigma = 0.1$, is applied. The two images at the bottom is reconstructions. They are reconstructed with the optimization problem defined in the next section, where the regularization term are quadratic regularization (to the left) and total variation (to the right). As we can see, the reconstruction with quadratic regularization is blurred and the reconstruction with total variation has clear edges. Note that the original image and the reconstructions are plotted with the colorbar scaled between 0 and 1.

2.3 Optimization problem

In the two previous sections two different convex functionals $J(u)$ and the constraint are presented. Now, we will define the optimization problem.

Assume the grid Γ to be fixed and choose the dyadic index set I to define the multiresolution norm $\|\cdot\|_B$. The optimization problem then becomes

$$\min_{u\in\mathbb{R}^\Gamma} J(u) \quad \text{such that} \quad \|u-Y\|_B \leq \gamma_h.$$

In order to simplify the notation, γ is used instead of γ_h if the emphasis of h is not necessary.

The multiresolution norm is complicated to work with, so we want to reformulate the constraint such that we can use another norm. To that end, we introduce a linear operator $F : \mathbb{R}^\Gamma \to \mathbb{R}^I$,

$$v \mapsto \left(\frac{1}{\sqrt{\#Q_i \cap \Gamma}} \sum_{Q_i \cap \Gamma} v(x) \right)_{i\in I},$$

where Q_i is the ith square in the index set I, and write the multiresolution norm as

$$\|v\|_B = \max_{(k,i,j)\in I} |(Fv)_{k,i,j}| = \|Fv\|_\infty.$$

We now use the infinity norm instead of the multiresolution norm. The optimization problem then becomes

$$\min_{u\in\mathbb{R}^\Gamma} J(u) \quad \text{such that} \quad \|F(u-Y)\|_\infty \leq \gamma. \tag{2.16}$$

The coupling between the objective function and the constraint is complicated. Therefore, we want to rewrite optimization problem so we get a simpler coupling. We start with introducing

$$v = F(u-Y)$$

and inserting it into (2.16). The optimization problem then becomes

$$\min_{u\in\mathbb{R}^\Gamma, v\in\mathbb{R}^I} J(u) \text{ such that } \|v\|_\infty \leq \gamma \text{ and } F(u-Y) = v.$$

Instead of formulating the problem as a constraint problem, it can be reformulated as an equivalent unconstrained problem by replacing the constraints with indicator functions [11]. First, we formulate the indicator function for the inequality constraint, which is $i_{\mathcal{C}}(v) : \mathbb{R}^I \to \mathbb{R} \cup \{+\infty\}$ on the feasible region $\mathcal{C} = \{u \in \mathbb{R}^\Gamma : \|u-Y\|_B \leq \gamma\}$. The indicator function then becomes

$$i_{\mathcal{C}}(v) = \begin{cases} 0 & \text{if } \|v\|_\infty \leq \gamma, \\ +\infty & \text{otherwise.} \end{cases}$$

Next, we formulate the indicator function for the equality constraint, which is $i_{\mathcal{F}} : \mathbb{R}^{\Gamma} \times \mathbb{R}^{I} \to \mathbb{R} \cup \{+\infty\}$ on the feasible region $\mathcal{F} = \{(u, v) \in \mathbb{R}^{\Gamma} \times \mathbb{R}^{I} : F(u - Y) - v = 0\}$. This indicator function is then defined as

$$i_{\mathcal{F}}(u, v) = \begin{cases} 0 & \text{if } F(u - Y) = v, \\ +\infty & \text{otherwise.} \end{cases}$$

Now, we replace the constraints with the indicator functions and the final optimization problem becomes

$$\min_{u \in \mathbb{R}^{\Gamma}, v \in \mathbb{R}^{I}} J(u) + i_{\mathcal{C}}(v) + i_{\mathcal{F}}(u, v). \tag{2.17}$$

This probelm is a convex optimization problem and to solve it, we will use some splitting algorithms that are presented in [5]. Which algorithms will be used depends on the convex functional $J(u)$ and the dimension of the problem.

Chapter 3

Algorithms

In this chapter, the algorithms use to solve the optimization problem (2.17) are presented. Before we do so, some theory from convex analysis and convex optimization is explained.

3.1 Convex analysis

A general optimization problem might have the form

$$\min_x f(x) \quad \text{such that} \quad \begin{aligned} c_i(x) &= 0, i \in \mathcal{E} \\ c_i(x) &\geq 0, i \in \mathcal{I}, \end{aligned}$$

where $f : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ in the objective function, $\mathcal{E}$ and $\mathcal{I}$ are the index sets for the equality and inequality constraints. The domain of f is defined as $\operatorname{dom} f = \{x : x \in \mathbb{R} \cup \{+\infty\}\}$ and a definition of the relative interior ri can be found in [11]. Convex optimization problems is a special group of optimization problems. In these problems the objective function f is convex, the equality constraints are linear and the inequality constraints are concave.

A function is convex f if

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

for all $0 \leq \alpha \leq 1$ and for all $x, y \in \mathbb{R}^d$. A concave function is the opposite of a convex function, which means that a function f is concave if

$$f(\alpha x + (1-\alpha)y) \geq \alpha f(x) - (1-\alpha)f(y),$$

for all $0 \leq \alpha \leq 1$. This definition of a concave function is the same as to say that f is concave if $-f$ is convex. We also have that a sum of convex functions is convex. A linear function is both convex and concave, because both the conditions are satisfied. Since a linear function is convex,

the sum of a convex function and a linear function is also convex. For more details about convex functions, see [11].

We assume that $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^m \to \mathbb{R}$ are convex. The minimization program

$$\min_x f(x) + g(Ax), \tag{3.1}$$

where $A \in \mathbb{R}^{n\times m}$, is then a convex program. We have a solution of (3.1) if and only if

$$0 \in \partial f(x^*) + A^* \partial g(Ax^*), \tag{3.2}$$

where x^* is the solution, A^* is the conjugate of the operator A and ∂f is the sub differential of f. One definition of the sub differential is

$$\partial f = \{s \in \mathbb{R}^n : \langle s, d\rangle \leq f'(x, d) \quad \text{for all} \quad d \in \mathbb{R}^n\}.$$

For the convex program we have the dual program

$$\min_s f^*(-As) + g^*(s), \tag{3.3}$$

where f^* is the conjugate of f and defined by

$$f^*(s) = \sup\{\langle s, x\rangle - f(x) : x \in \mathbb{R}^n\},$$

and g^* is the conjugate of g and defined by

$$g^*(s) = \sup\{\langle s, x\rangle - g(x) : x \in \mathbb{R}^m\}.$$

The functions f^* ad g^* are convex, so the dual program is also convex. To solve (3.1), we will use a reformulation of (3.2). To find a more suitable form of (3.2) we use that

$$s \in \partial f(x),$$

if and only if

$$x \in \partial f^*(s).$$

If x^* solves (3.1), then there exists

$$s^* \in \partial g(Ax^*),$$

with

$$-A^*s^* \in \partial f(x^*).$$

As a consequence, we get that

$$Ax^* \in \partial g^*(s^*),$$

and

$$x^* \in \partial f^*(-A^*s^*).$$

This gives us that s^* solves the dual problem (3.3). Also, x^* solves (3.1) and s^* solves (3.3) if and only if

$$s^* \in \partial g(Ax^*),$$

and

$$-A^*s^* \in \partial f(x^*),$$

or, alternatively, if

$$Ax^* \in g^*(s^*), \tag{3.4}$$

and

$$x^* \in \partial f^*(-As). \tag{3.5}$$

Then to solve (3.1), we can solve (3.3) and use (3.4) and (3.5) to get x^*. For further readings see [12, Chapter III, Remark 4.2].

Since (2.17) is a convex optimization problem and the object function is a sum of convex functions, since we assume $J(u)$ to be convex, $i_{\mathcal{C}}$ is convex and $i_{\mathcal{F}}$ is linear, we can use some of the algorithms presented in [5] to solve it. The methods presented here are splitting algorithms that solves minimization problems of the form

$$\min_{x \in \mathbb{R}^N} f_1(x) + \ldots + f_m(x),$$

where $f_1, \ldots, f_m : \mathbb{R}^N \to]-\infty, \infty]$ are convex functions. Some of the functions can be seen as indicator functions [11] for a nonempty sets $C_i \in \mathbb{R}^n$. A natural method to minimize a indicator function is to project it onto C_i. The projection can then be formulated as

$$\min_{y \in \mathbb{R}^N} f_j(y) + \frac{1}{2}\|x - y\|_2^2,$$

where $x \in \mathbb{R}^N$ and $1 \leq j \leq m$. This minimization problem also makes sense when f is not an indicator function. From [11], we have that for every $x \in \mathbb{R}^N$ the minimization problem admits a unique solution. The unique solution is denoted as the proximity operator, which is formulated as

$$\text{prox}_{\ \mu_j f_j}(x) = \arg\min_{y \in \mathbb{R}^N} \mu_j f_j(y) + \frac{1}{2}\|x - y\|_2^2, \tag{3.6}$$

where μ_j is a constant.

The proximity operator is a weighted minimization of the distance between x and y at the same time is f minimized. When f is an indicator function this is, as mentioned, a simple projection. If f is not a indicator function, the problem is minimized iteratively.

3.2 Alternating direction method of multipliers

The alternating direction method of multipliers (ADMM) is a variant of the augmented Lagrangian method [15, chapter 17], where the objective function is a sum of convex functions depending on different variables. The objective function with two different variables should be of the form

$$f_1(x) + f_2(y),$$

and the optimization problem of the form

$$\min_{x\in\mathbb{R}^N, y\in\mathbb{R}^M, Lx=y} f_1(x) + f_2(y),$$

where L is such that L^TL is invertible and

$$(\operatorname{ri}\operatorname{dom} f_2) \cap \operatorname{ri} L(\operatorname{dom} f_1) \neq \emptyset.$$

As the augmented Lagrangian method, ADMM uses the augmented Langrangian $\mathcal{L}_A : \mathbb{R}^N \times \mathbb{R}^M \times \mathbb{R}^M \to]-\infty, +\infty[$ and finds the solution at the saddle point of the Lagrangian. The difference in the methods is how the saddle point is found. The augmented Lagrangian method minimizes with respect to the primal variables, x and y, simultaneously, while ADMM minimizes with respect to one at a time. For fixed λ and the Lagrange multiplier $p_0 \in \mathbb{R}^M$, we then obtain the algorithm

$$\begin{aligned} x_n &\in \arg\min_{v\in\mathbb{R}^N} \mathcal{L}_A(x, y_{n-1}; p_{n-1}), \\ y_n &\in \arg\min_{u\in\mathbf{H}} \mathcal{L}_A(x_n, y; p_{n-1}), \\ p_n &= p_{n-1} + \lambda(S_N u_n - v_n), \end{aligned} \tag{3.7}$$

where $n = 1, \ldots$. Alternatively, using the proximity operators the algorithm becomes

$$\begin{aligned} x_n &= \operatorname{prox}_{L\mu_1 f_1}(y_{n-1} - p_{n-1}), \\ y_n &= \operatorname{prox}_{\mu_2 f_2}(Lx_n + p_{n-1}), \\ p_n &= p_{n-1} + Lx_n - y_n, \end{aligned}$$

where $n = 1, \ldots$. The proximity operator for f_1 is defined as

$$\begin{aligned} \operatorname{prox}_{L\mu f_1}(x) &= \arg\min_{x\in\mathbb{R}^N} \mathcal{L}_A(x, y_{n-1}; p_{n-1}) \\ &= \arg\min \mu_1 f_1(x) - \frac{1}{2}\|Lx - y\|_2^2, \end{aligned}$$

and the proximity operator for f_2 is defined as

$$\operatorname{prox}_{\mu_2 f_2}(x) = \arg\min_{y\in\mathbb{R}^N} \mu_2 f_2(y) + \frac{1}{2}\|x - y\|_2^2.$$

For further reading see [5] that includes more references and proof of convergence.

3.3 Douglas-Rachfords splitting algorithm

A problem of the form

$$\min_{x\in\mathbb{R}^N} f_1(x) + f_2(x),$$

where f_1 and f_2 maps from $\mathbb{R}^N$ to $]-\infty, +\infty[$ and both are lower semi continuous convex functions such that

$$(\operatorname{ri}\operatorname{dom} f_i) \cap (\operatorname{ri}\operatorname{dom} f_2) \neq \emptyset$$

and

$$f_1(x) + f_2(x) \to +\infty \quad \text{as} \quad \|x\|_2 \to +\infty$$

can be solved with a method called Douglas-Rachford splitting from [5]. Douglas-Rachfords splitting algorithm is similar to the forward-backward algorithm [5] that solves the same minimization problem, but either f_1 or f_2 must be differentiable with a β-Lipschitz continuous gradient such that for all $(x, y) \in \mathbb{R}^N \times \mathbb{R}^N$

$$\|\nabla f_2(x) - \nabla f_1(x)\|_2 \leq \beta\|x - y\|_2,$$

where $\beta \in]0, +\infty[$.

The Douglas-Rachford algorithm uses the proximity operator (3.6) for f_1 and f_2 and for fixed $\epsilon \in]0, 1[$ and $\mu > 0$, and $y_0 \in \mathbb{R}^N$ is

$$\begin{aligned} x_n &= \operatorname{prox}_{\mu f_2}(y_n), \\ \lambda_n &\in [\epsilon, 2.\epsilon], \\ y_{n+1} &= y_n + \lambda_n(\operatorname{prox}_{\mu f_1}(2x_n - y_n) - x_n). \end{aligned}$$

From proposition 4.3 in [5], we have convergence of the Douglas-Rachford algorithm.

3.4 Dykstra's projection algorithm

Dykstra's projection algorithm is a method for computing the projection of a vector $h \in \mathbb{R}^N$ onto the intersection of a family of convex sets $D_1, \ldots, D_M \subset \mathbb{R}^N$. It projects h first onto D_1, then onto D_2 and so on until D_M. What distinguishes Dykstra's projection algorithm from many other iterative projection methods is that the change q_j of h from the last projection onto D_j is removed from h before the projection onto D_j. The algorithm continues until h lies in the intersection $\cap_i D_i$ and the projection steps do not change any more. Dykstra's projection algorithm is a more complicated method than other alternating projection methods. Unlike other methods, Dykstra's projection method finds the projection of x_0. Many other methods only produce some element in the intersection, not necessary the closest to h.

For a general problem, where we project h_0 onto $D_1, \ldots, D_M \subset \mathbb{R}^N$, the algorithm can be summarized in the following two loops. First,

$$\begin{aligned} h_n &= \operatorname{proj}_{D_n}(h_{n-1}), \\ q_n &= h_n - h_{n-1}, \end{aligned}$$

where $n = 1, \ldots, M$, is performed and set $h_0 = h_M$. Then

$$h_n = \text{proj}_{D_n}(h_{n-1} - q_n)$$
$$q_n = h_n - h_{n-1},$$

where $n = 1, \ldots, M$, is repeated until convergence. After each repetition we set $h_0 = h_M$. For more details and proof of convergence see [2].

A simple projection problem and the mechanism of the algorithm is illustrated in figure 3.1. Here we have two sets A and B in red and blue respectively, and their intersection marked in purple. We want to project the point x_{start} onto the intersection $A \cap B$, the resulting projection being the point x_{end} in the illustration. Both these points are colored blue. The arrows in the figure indicate projections, and the dotted arrows the change when we remove the last projection step.

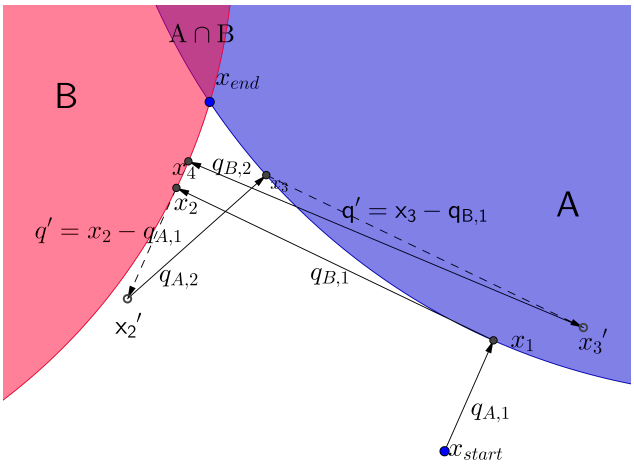


Figure 3.1: Illustration of Dykstra's projection algorithm. We want to project x onto the intersection of the two sets A and B. The startposition for x is x_{start}. The new position for x after an projection is marked as an filled dot, and the projection steps $q_{A,j}$ and $q_{B,j}$ are marked as lines. When the projection steps are subtracted from x, the new positions are marked with circles. The final solution of the projection is marked as x_{end}.

The problem in figure 3.1 is solved with Dykstra's projection algorithm. In the first iteration, we project x first onto A then onto B and obtain iterates x_1 and x_2. We denote the change in the first update of x by $q_{A,1} = x_1 - x_{start}$ and the second update by $q_{B,1} = x_2 - x_1$. The next iterations are different from the first. As in the first iteration, we first project x onto A, but before doing so we subtract $q_{A,1}$. That is, instead of projecting x_2, we rather project $x_2' = x_2 - q_{A,1}$ obtaining a new iterate x_3 and a new iteration step $q_{A,2} = x_3 - (x_2 - q_{A,1})$. Further, we project x_3 onto B by the same procedure with $x_3' = x_3 - q_{B,1}$ and obtain the new iterate x_4 and a new iteration step $q_{B,2} = x_4 - (x_3 - q_{B,1})$. Then, we check if x equals the projection x_{end} by checking if x now lies in the intersection $A \cap B$ and if the projection steps satisfy $q_{A,1} \approx q_{A,2}$ and $q_{B,1} \approx q_{B,2}$. If either of these conditions is not satisfied we continue with iterations.

3.5 Chambolle's projection algorithm

Chambolle's projection algorithm [3] is a method to minimize total variation. The problem must be of the form

$$\min_u |\nabla u| + \frac{1}{2\mu}\|u_0 - u\|_2^2.$$

This minimization problem can also be written of the form

$$\min_u R(Au) + S(u), \tag{3.8}$$

where $R(v) = \|v\|_1$, $S(u) = \frac{1}{2\mu}\|u_0 - u\|_2^2$ and $A = \nabla$. To derive the algorithm, we follow [3], [1] and [4, chapter 4]. Chambolle's projection algorithm uses the dual form. So if u^* solves (3.8) then, p^* solves the dual problem

$$\min_p R^*(-A^*p) + S^*(p). \tag{3.9}$$

R^* is the conjugate of r and defined as

$$R^*(p) = \sup\{\langle p, u\rangle + \|u\|_1 : u \in \mathbb{R}^{N\times M}\},$$

which is the same as the indicator function of the infinity ball, so

$$R^*(p) = \begin{cases} 0 & \text{if } \|p\|_\infty \leq 1, \\ +\infty & \text{if } \|p\|_\infty > 1. \end{cases}$$

The conjugate of S, S^*, is defined as

$$S^*(\zeta) = \sup\{\langle \zeta, u\rangle - \frac{1}{2\mu}\|u_0 - u\|_2^2 : u \in \mathbb{R}^{N\times M}\}, \tag{3.10}$$

where $\zeta = -A^*p$. The solution of the supremum can be found by differentiating the terms inside the supremum with respect to u, set the result equal to zero and solve for u. We then get

$$u = u_0 + \mu\zeta.$$

We then insert u into (3.10) and get

$$\begin{aligned} S^*(u) &= \langle \zeta, u_0 + \mu\zeta\rangle + \frac{1}{2\mu}\|\mu\zeta - u\|_2^2 \\ &= \frac{\mu}{2}\left(\|\zeta + \frac{u_0}{\mu}\|^2 - \|\frac{u_0}{\mu}\|_2^2\right). \end{aligned}$$

Since ζ is the only argument of S^*, the last term of S^* is a constant and can be ignored when deriving the minimization. The conjugate of S is then

$$S^*(-A^*p) \sim \frac{\mu}{2}\|u_0 - \zeta\|_2^2.$$

From the equation

$$\int \nabla u \cdot p = -\int u \text{div} p,$$

we have that

$$\nabla^* = -\text{div}.$$

Now, we can rewrite the minimization problem as

$$\min_p R^*(\text{div} p) + S^*(p).$$

From the theory of dual convex problems in chapter 3.1, we have that u^* solves (3.8) and p^* solves (3.9) if and only if

$$-A^* p^* \in \partial S(u^*) = \frac{1}{\mu}(u_0 - u).$$

From this condition, we have the minimum of (3.8), which is

$$u^* = u_0 + \mu \text{div} p. \tag{3.11}$$

We still have not found p^*, so we do not have the final solution of (3.8). To find p, we minimize

$$\min_p \frac{1}{2}\|u_0 + \mu \text{div} p\|^2 \quad \text{such that} \quad \|p\|_\infty \leq 1.$$

This minimization problem can be minimized with a projected gradient method, where we project $u_0 + \mu \text{div} p$ onto the infinity ball B_∞. So, we choose p_0, and then iterate

$$\begin{aligned} p_{k+1} &= \text{proj}_{B_\infty}\left[p_k + \lambda \nabla(u_0 + \mu \text{div} p_k)\right] \\ &= \frac{p_k + \lambda \nabla(u_0 + \mu \text{div} p_k)}{\max(1, |p_k + \lambda \nabla(u_0 + \mu \text{div} p_k|)}, \end{aligned} \tag{3.12}$$

where λ is the step length. By introducing the variables

$$\begin{aligned} w &= \frac{u}{\mu} \\ \tau &= \frac{\mu}{\lambda}, \end{aligned}$$

we can rewrite (3.11) and get

$$w = \frac{u_0}{\mu} + \text{div} p. \tag{3.13}$$

With (3.13), (3.12) can be written as

$$p_{k+1} = \frac{p_k + \tau \nabla w}{\max(1, |p_k + \tau \nabla w|)}. \tag{3.14}$$

For each iteration of (3.14), we find w with (3.13). The algorithm can be summarized as

$$w^m = \frac{u_0}{\mu} + \text{div}p^m$$
$$p_{i,j}^{m+1} = \frac{p_{i,j}^m + \tau(\nabla w^m)_{i,j}}{\max\{1, |p_{i,j}^n + \tau(\nabla w^m)_{i,j}|\}},$$

where

$$(\text{div}p)_{i,j} = \begin{cases} p_{i,j}^1 - p_{i-1,j}^1 & \text{if } 1 < i < N \\ p_{i,j}^1 & \text{if } i = 1 \\ -p_{i-1,j}^1 & \text{if } i = N \end{cases} + \begin{cases} p_{i,j}^2 - p_{i,j-1}^2 & \text{if } 1 < j < M \\ p_{i,j}^2 & \text{if } j = 1 \\ -p_{i,j-1}^2 & \text{if } j = M \end{cases}.$$

The final u is found by multiply w by μ after the last iteration. From theorem 3.1 in [3] the method will converge if we choose $\tau \leq 1/8$. For more details and proof of convergence, see [3].

Chapter 4

Algorithmic approach

We will now formulate the algorithms to minimize different variations of (2.17). First, we will formulate the algorithm for a problem in one dimension and when $J(u) = \frac{1}{2}\|u'\|_2^2$. Second, we will formulate the algorithms for the two dimensional cases. We will start with $J(u) = \frac{1}{2}\|\nabla u\|_2^2$ and continue wirh $J(u) = \|\nabla u\|_1$.

4.1 Quadratic regularization in one dimension

In one dimension, we are given the data points $Y \in \mathbb{R}^N$ on the one-dimensional grid

$$\Gamma = \{x_i \in \mathbb{R} \mid x_i = \frac{i-1}{N-1}, i = 1, \ldots, N\} \subset [0,1]\,.$$

We then assume that the data points can be described in the form

$$Y_i = f(x_i) + \varepsilon_i.$$

In one dimension, the dyadic index set is defined as

$$I = \bigcup_{s=0}^{m} I_s,$$

where

$$I_s = \{(k,i) : k = 2^s, i = p2^s + 1 \text{ with } p = 0, \ldots, 2^{m-s} - 1\}.$$

The dyadic index sets is discussed in appendix B. With the dyadic index set, the multiresolution norm is defined as

$$\|v\|_B := \max_{(k,i)\in I} \frac{1}{\sqrt{k}} |\sum_{p=i}^{i+k-1} v_p|.$$

In one dimension, we use the quadratic regularization term defined as

$$J(u) = \frac{1}{2}\|u'\|_2^2,$$

and then have a quadratic regularization problem. We discretise u' on the grid Γ using forward differences, and the term $\|u'\|_2^2$ becomes $\|Du\|_2^2$.

We now formulate the optimization problem as

$$\min_{u\in\mathbb{R}^N, v\in\mathbb{R}^I} \frac{1}{2}\|Du\|_2^2 + i_{\mathcal{C}}(v) \text{ such that } v + u - Y = 0$$

where $v = Y - u$ and $i_{\mathcal{C}}(u) : \mathbb{R}^N \to \mathbb{R} \cup \{+\infty\}$ is the characteristic function on the feasible region $\mathcal{C} = \{u \in \mathbb{R}^N : \|Y - u\|_B \leq \gamma\}$ and defined as

$$i_{\mathcal{C}}(v) = \begin{cases} 0 & \text{if } \|v\|_B \leq \gamma \\ +\infty & \text{otherwise.} \end{cases}$$

The objective function is a convex function since it is a sum of two convex functions. Further, the optimization problem consists of two primal variables, u and v. It would be possible to apply the augmented Lagrangian method presented in [15], but we will rather follow the approch of [7] and apply the ADMM.

ADMM in general is presented in the chapter 3.2 and now we will rewrite it for our problem. Before doing so, we formulate the augmented Lagrangian $\mathcal{L}_A : \mathbb{R}^N \times \mathbb{R}^I \times \mathbb{R}^N \to \mathbb{R} \cup \{+\infty\}$ for an positive parameter λ as

$$\begin{aligned} \mathcal{L}_A(u, v, p; \lambda) =& \frac{1}{2}\|Du\|_2^2 + i_{\mathcal{C}}(v) + \frac{1}{2\lambda}\|u + v - Y\|^2 - \langle p, u + v - Y\rangle \\ =& \frac{1}{2}\|Du\|_2^2 + i_{\mathcal{C}}(v) + \frac{1}{2\lambda}\|u + v - Y + \lambda p\|^2 - \frac{1}{2\lambda}\|\lambda p\|_2^2. \end{aligned} \tag{4.1}$$

Since we follow the algorithm presented in (3.7) the Lagrangian is minimized with respect to to u and v. We then note that the last term, $\frac{1}{2\lambda}\|\lambda p\|_2^2$, is a constant when $\mathcal{L}_A$ is minimized with respect to u and v. We can therefore ignore the last term.

Now that we have the Lagrangian, we can formulate the algorithm for our problem. To do so, we look at the two optimization problems in (3.7) in more detail. First, we look at the minimum with respect to u, here we have a quadratic optimization problem. Here, v is fixed and on the feasible set $\mathcal{C}$ such that $i_{\mathcal{C}}(v) = 0$. Hence, we have

$$\begin{aligned} u_n &\in \arg\min_{u\in\mathbb{R}^N} \mathcal{L}_A(v_n, u; p_{n-1}) \\ &= \arg\min_{u\in\mathbb{R}^N} \frac{1}{2}\|Du\|_2^2 + \frac{1}{2\lambda}\|u + v_n - Y + \lambda p_{n-1}\|_2^2. \end{aligned}$$

The minimum is found where the derivative with respect to u is zero. Then, we have

$$(2\lambda D^*D + I)u = (Y + \lambda p_{n-1} - v_n),$$

where I is the identity matrix. This linear system is tridiagonal and can be solved directly.

Next, we look at the minimum with respect to v. Now u is fixed, so the term$\frac{1}{2}\|Du\|_2^2$ is a constant which can be removed in (4.1). Thus, the first step of (3.7) becomes

$$\begin{aligned} v_n &\in \arg\min_{v\in\mathbb{R}^I} \mathcal{L}_A(v, u_{n-1}; p_{n-1}) \\ &= \arg\min_{v\in\mathbb{R}^I} i_{\mathcal{C}}(v) + \frac{1}{2\lambda}\|u_{n-1} + v - Y + \lambda p_{n-1}\|^2. \end{aligned}$$

In words, v_n minimizes the distance to $u - Y + \lambda p_{n-1}$ subject to the constraint such that $v_n \in \mathcal{C}$. Thus v_n can be found by projecting $u_{n-1} - Y + \lambda p_{n-1}$ onto $\mathcal{C}$, which can be written as

$$v_n = \text{proj}_{\mathcal{C}}(u_{n-1} - Y + \lambda p_{n-1}). \tag{4.2}$$

The ADMM for our problem is summarized in algorithm 1.

Algorithm 1 Alternating Direction Method of Multipliers

Require: $Y \in \mathbb{R}^N$ (data), $\lambda > 0$(step size, $\tau > 0$ (tolerance)
Ensure: (u, v) is an approximate solution of (4.1).
$u \leftarrow \mathbf{0}_N$ and $v = p \leftarrow \mathbf{0}_N$
$r \leftarrow \|u + v - Y\|$
while $r > \tau$ **do**
$v \leftarrow \text{proj}_{\mathcal{C}}(u + v - Yp)$
$u_{old} \leftarrow u$
$u \leftarrow (2\lambda D^*D + I)^{-1}(Y + \lambda p - v)$
$p \leftarrow p - \frac{u+v-Y}{\lambda}$
$r \leftarrow \max(\|u + v - Y\|_2, \|u - u_{old}\|_2)$
end while

As discussed in [7], the projection (4.2) can be computed with Dykstra's projection method. Dykstra's projection algorithm is in general presented in chapter 3.4, and we will now apply this algorithm to (4.2) for computing the projection of $Y - u + \lambda p$ onto $\mathcal{C}$. We write $\mathcal{C}$ as the intersection of the sets

$$\mathcal{C}_{k,i} = \{h : |s_{k,i}(h)| \leq \gamma\sqrt{k}\} \subset \mathbb{R}^N, \tag{4.3}$$

where

$$s_{k,i}(h) = \sum_{j=i}^{i+k-1} h_j,$$

and $(k,i) \in I$. When we project a vector $h \in \mathbb{R}^N$ onto $\mathcal{C}_{k,i}$ one of two situations will occur: Either $\mathcal{C}_{k,i}$ contains h or $\mathcal{C}_{k,i}$ does not contain h. If $\mathcal{C}_{k,i}$ contains h, we do nothing and the projection step q_j equals zero. In this case we have

$$\pi_{\mathcal{C}_{k,i}}(h) = h.$$

On the other hand, if $\mathcal{C}_{k,i}$ does not contain h, we project h onto $\mathcal{C}_{k,i}$ as illustrated in figure 4.1. As we can see, the projection can be carried out by subtracting the distance between h and $\mathcal{C}_{k,i}$ from h. We will now discuss a mathematical description for this projection.

We start by discussing the situation when $s_{k,i}(h) > \gamma\sqrt{k}$. The projection can then be obtained by subtracting the distance between h and $\mathcal{C}_{k,i}$ times the normalized normal vector $e_{k,i}$ from h. The normal vector to the boundary of $\mathcal{C}_{k,i}$ is defined as

$$\begin{array}{cc} i & i+k-1 \\ \downarrow & \downarrow \\ e_{k,i} = (0,\ldots,0,1,\ldots,1,0,\ldots,0). \end{array}$$

To find the distance we need to rewrite (4.3) so we have

$$\begin{aligned} \mathcal{C}_{k,i} &= \{h : |s_{k,i}(h)| \leq \gamma\sqrt{k}\} \\ &= \{h : |\langle e_{k,i}, h\rangle| \leq \gamma\sqrt{k}\} \\ &= \left\{h : \left|\left\langle \frac{e_{k,i}}{\sqrt{k}}, h \right\rangle\right| \leq \gamma\right\}. \end{aligned}$$

The distance between h and $\mathcal{C}_{k,i}$ is then

$$\text{dist}(h, \mathcal{C}_{k,i}) = \frac{1}{\sqrt{k}} |\langle e_{k,i}, h\rangle| - \gamma$$

and the projection step become

$$q_{\mathcal{C}_{k,i}}(h) = -\frac{1}{k}\left(|\langle e_{k,i}, h\rangle| - \gamma\sqrt{k}\right) e_{k,i}.$$

Thus, we see that for $s_{k,i}(h) > \gamma\sqrt{k}$ is

$$\pi_{\mathcal{C}_{k,i}}(h) = h + q_{\mathcal{C}_{k,i}}(h).$$

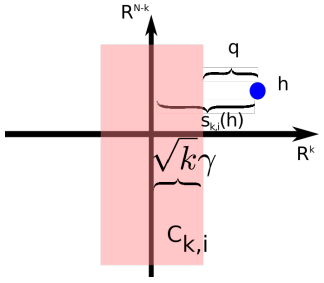


Figure 4.1: The figure shows how h is projected onto $\mathcal{C}_{k,i}$ when h is on the outside. Here q is the projection step when h is projected onto $\mathcal{C}_{k,i}$.

Now, for $s_{k,i}(h) < -\gamma\sqrt{k}$, we will follow the discussion above analogously and obtain the projection step

$$q_{\mathcal{C}_{k,i}}(h) = \frac{1}{k}\left(|\langle e_{k,i}, h\rangle| - \gamma\sqrt{k}\right) e_{k,i}.$$

Then we have

$$\pi_{\mathcal{C}_{k,i}}(h) = h + q_j(h).$$

The projection of h onto $\mathcal{C}_{k,i}$ can be collected into one function

$$\pi_{\mathcal{C}_{k,i}}(h) = \begin{cases} h & \text{if } h \in \mathcal{C}_{k,i}, \\ h + q_{\mathcal{C}_{k,i}} e_{k,i} & \text{otherwise} \end{cases}$$

where the projection step $q_{\mathcal{C}_{k,i}}$ is

$$q_{\mathcal{C}_{k,i}}(h) = -\frac{1}{k}\text{sign}(s_{k,i}(h))\left(|s_{k,i}(h)| - \sqrt{k}\gamma\right).$$

Dykstra's projection algorithm for (4.2) is summarized in algorithm 2.

Algorithm 2 Dykstra's algorithm

Require: $h \in \mathbb{R}^N$ (data), $\mathcal{C}_{k,i} \subset \mathbb{R}^N$, $(k,i) \in I$, where I is a index set (closed and convex sets), $\tau > 0$ (tolerance)

Ensure: A sequence $\{h_k\}_{k\in\mathbb{N}}$ that converges to $\pi_{\mathcal{C}}(h)$ where $\mathcal{C} = \cap_{(k,i)\in I}\mathcal{C}_{k,i}$

```
Q_{k,i} ← 0_N for all (k,i) ∈ I
Δq_max ← 0
while ρ ≥ τ do
    h_old ← h
    for (k,i) ∈ I do
        q_old ← Q_{k,i}
        h ← π_{C_{k,i}}(h − Q_{k,i}) and Q_{k,i} ← q_{C_{k,i}}(h − Q_{k,i})
        Δq_max = max{Δq_max, ‖q_old − Q_{k,i}‖_∞}
    end for
    ρ ← max{‖h − h_old‖_2, Δq_max}
end while
```

Dykstra's projection algorithm is expensive for large N because each subset performs an iteration in the inner loop of the algorithm. If we are employing all the consecutive subsets of $\mathbb{R}^N$, the number of iterations in the inner loop is $\frac{N(N-1)}{2} = O(N^2)$. If we reduce the subsets to only the dyadic subsets, we reduce the number of iterations to $2N - 1 = O(N)$. With the dyadic index set, we can reduce the iterations even more. For a given level s the sets are disjoint, hence the projections are independent. Therefore, the results on each level are independent of the order the projections are performed in. The independence in each level give us the opportunity to do the projections at one

level simultaneously. We can now rewrite the sets Dykstra's projection algorithm in algorithm 2 projects onto as

$$D_k = \bigcap_{i:(k,i)\in I} D_{k,i} = \{v : |\sum_{p=i}^{i+k-1} v_p| \leq \gamma\sqrt{k}\}.$$

The consequence of computing the projections simultaneously is that the number of iterations in the inner loop of algorithm 2 is reduced to $O(\log(N))$.

4.2 Quadratic regularization in two dimensions

Now, we will study (2.17) in two dimensions with the grid Γ fixed. We use the convex functional defined as

$$J(u) = \frac{1}{2}\|\nabla u\|_2^2$$

and the optimization problem is then defined as

$$\min_{u\in\mathbb{R}^\Gamma, v\in\mathbb{R}^I} \frac{1}{2}\|\nabla u\|_2^2 + i_{\mathcal{C}}(v) + i_{\mathcal{F}}(u,v). \tag{4.4}$$

This optimization problem is a sum of three convex functions and therefore convex. Since this optimization problem is an unconstrained convex optimization problem and the object function is a sum of convex functions, will we use one of the splitting methods presented in chapter 3.1. We will not use ADMM and Dykstra's projection algorithm, because Dykstra's projection algorithm will be too slow and instead of using ADMM with another algorithm for Dykstra's projection algorithm. We will therefore use Douglas-Rachford algorithm, which is presented in chapter 3.3.

We choose to split (4.4) into two functions, $g(u,v)$ and $h(v)$, where

$$\begin{aligned} g(u,v) &= \frac{1}{2}\|\nabla u\|_2^2 + i_{\mathcal{F}}(u,v), \\ h(v) &= i_{\mathcal{C}}(v). \end{aligned}$$

To use Douglas-Rachfords splitting algorithm we need the proximity operators from (3.6). For $g(u,v)$ and $h(v)$ the proximity operators become

$$\text{prox}_{\mu_g,g}(u,v) = \arg\min_{\eta,\zeta} \mu_g g(\eta,\zeta) + \frac{1}{2}\|u-\eta\|^2 + \frac{1}{2}\|v-\zeta\|^2 \tag{4.5}$$

$$\text{prox}_{\mu_h,h}(u,v) = \arg\min_{\eta,\zeta} \mu_h h(\zeta) + \frac{1}{2}\|u-\eta\|^2 + \frac{1}{2}\|v-\zeta\|^2. \tag{4.6}$$

We start by studying the proximity operator for h. This proximity operator is only a projection of η onto the infinity unit ball. The infinity unit ball is a hypercube and since v shall be less than γ, the side length of the square is 2γ. The projection is similar to the projections described in chapter 4.1: If v already lies inside the unit ball nothing will happen, if it lies outside of the ball the new v

will lie at the boundary of the unit ball. The sign of the new v depends on whether the original v was on the positive or the negative side of the infinity unit ball. We can then rewrite (4.6) as

$$\text{prox }_h(u,v) = \min\{\gamma, \max\{\zeta, -\gamma\}\}. \tag{4.7}$$

Next, we study the proximity operator for g. We can rewrite (4.5) such that it is a convex and quadratic optimization problem with linear constraints,

$$\text{prox }_{\mu_g g}(u,v) = \arg\min_{\eta,\zeta} \mu_g \frac{1}{2}\|\nabla\eta\|_2^2 + \frac{1}{2}\|u-\eta\|^2 + \frac{1}{2}\|v-\zeta\|^2 \quad \text{such that} \quad F(\eta - Y) = \zeta. \tag{4.8}$$

In appendix A, it is proven that there is necessary and sufficient that the solution satisfies the KKT conditions. Therefore, we will use the KKT conditions to find the solution of the proximity operator. To formulate the KKT conditions, we must formulate the Lagrangian for (4.8). With the Lagrange multiplicator λ the Lagrangian can be written as

$$\mathcal{L}(\eta,\zeta;\lambda) = \mu_g \frac{1}{2}\|\nabla\eta\|_2^2 + \frac{1}{2}\|u-\eta\|^2 + \frac{1}{2}\|v-\zeta\|^2 - \lambda^T(F(\eta - Y) - \zeta).$$

The KKT conditions then become

$$\nabla\mathcal{L}(\eta,\zeta;\lambda) = \begin{bmatrix}\partial_\eta\mathcal{L} \\ \partial_\zeta\mathcal{L}\end{bmatrix} = \begin{bmatrix}\eta - u - \mu_g\Delta\eta - F^*\lambda \\ \zeta - v + \lambda\end{bmatrix} = 0$$
$$F(\eta - Y) = \zeta,$$

which is the same as

$$\begin{aligned}(Id_\eta - \mu_g\Delta)\eta - F^*\lambda &= u,\\ \zeta + \lambda &= v,\\ -F\eta + \zeta &= -FY,\end{aligned}$$

where Id_η is a identity matrix of the size of η. This equations are a system of linear equalities and can be written in the matrix form $Ax = b$, where

$$A = \begin{bmatrix}(Id - \mu_g\Delta) & 0 & -F^T \\ 0 & Id_\zeta & Id_\lambda \\ -F & Id_\zeta & 0\end{bmatrix}, \quad x = \begin{bmatrix}\eta \\ \zeta \\ \lambda\end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix}u \\ v \\ -FY\end{bmatrix}.$$

Id_ζ and Id_λ are identity matrices on the same size as ζ and λ, respectively. If

$$\begin{bmatrix}\eta \\ \zeta \\ \lambda\end{bmatrix} = A^{-1}b,$$

we can define the proximity operator as

$$\text{prox }_g(u,v) = \begin{bmatrix}\eta \\ \zeta\end{bmatrix}. \tag{4.9}$$

For the convex functional $J(u) = \frac{1}{2}\|\nabla u\|_2^2$ with the proximity operators (4.7) and (4.9) the Douglas-Rachford algorithm for (2.17) can be summarized as in algorithm 3.

Algorithm 3 Douglas-Rachford algorithm when $J(u) = \frac{1}{2}\|\nabla u\|_2^2$.

Require: $u \in \mathbb{R}^{N\times M}$, $\lambda \in]0, 2[$, $\gamma > 0$(step size), tol > 0 (stopping criteria)
Ensure: (w, v_{old}) is an approximate solution of (4.4).
$r \leftarrow \mathbb{R}^I$
Set $r >$ tol
while $r >$ tol **do**
$w_{old} \leftarrow w$
$v_{old} \leftarrow v$
$b = (u, v, -FY)^T$
$w \leftarrow \min\{\gamma, \max\{v, -\gamma\}\}$
$(u, v, \lambda)^T \leftarrow (u, y, 0) + \lambda \left(A^{-1}\left[2b - (u, w, -FY)^T\right] - (u, y, 0)\right)$
$r \leftarrow \|w - w_{old}\|_2$
end while

4.3 Total variation regularization in two dimensions

We continue to study a two dimensional problem, but we change the convex functional to

$$J(u) = \|\nabla u\|_1.$$

Then the optimization problem is defined as

$$\min_{u\in\mathbb{R}^\Gamma, v\in\mathbb{R}^I} \|\nabla u\|_1 + i_{\mathcal{C}}(v) + i_{\mathcal{F}}(u, u). \tag{4.10}$$

We will also here use the Douglas-Rachford splitting algorithm, but we can not use the same method as used when $J(u) = \frac{1}{2}\|\nabla u\|_2^2$ because $\|\nabla u\|_1$ is not differentiable. Now, we divide the optimization problem into the two functions, $g(u, v)$ and $h(v)$, where

$$g(u, v) = \frac{1}{2}\|\nabla u\|_1 + i_{\mathcal{C}}(v),$$
$$h(u, v) = i_{\mathcal{F}}(u, v).$$

The proximity operators are

$$\text{prox}_{\mu_g, g}(u, v) = \arg\min_{\eta,\zeta} \mu_g g(\eta, \zeta) + \frac{1}{2}\|u - \eta\|^2 + \frac{1}{2}\|v - \zeta\|^2, \tag{4.11}$$

$$\text{prox}_{\mu_h, h}(u, v) = \arg\min_{\eta,\zeta} \mu_h h(\eta, \zeta) + \frac{1}{2}\|u - \eta\|^2 + \frac{1}{2}\|v - \zeta\|^2. \tag{4.12}$$

The first proximity operator consists of $J(u)$ and $i_{\mathcal{C}}(v)$, which are independent and can be solved separately. The proximity operator can then be written of the form

$$\text{prox }_{\mu g}(u,v) = \begin{pmatrix} \arg\min_{\eta} \frac{1}{2}\|\eta - u\|_1^2 + \mu\|\nabla\eta\|_1 \\ \arg\min_{\zeta} \frac{1}{2}\|\zeta - u\|_1^2 + i_{\mathcal{C}}(v) \end{pmatrix}.$$

The lower part of the proximity operator can be solved with projection onto the infinity ball. This projection is explained in the previous section, and the proximity operator can be written as

$$\text{prox }_{\mu g}(u,v) = \begin{pmatrix} \arg\min_{\eta} \frac{1}{2}\|\eta - u\|_1^2 + \mu\|\nabla\eta\|_1 \\ \min\{\gamma, \max\{\zeta, -\gamma\}\} \end{pmatrix}. \tag{4.13}$$

The upper part of the proximity operator can be minimized with Chambolle's projection algorithm, which is explained in chapter 3.5.

The minimization problem can be written as

$$\min_u R(A\eta) + S(\eta),$$

where $R(\zeta) = \|\zeta\|_1$, $S(\eta) = \frac{1}{2}(\eta - u)^2$, and $A = \nabla$. This minimization problem is the same as (3.8), so we follow the approach in chapter 3.5. We now have the convex dual problem

$$\min_p R^*(-A^*p) + S^*(p),$$

where

$$R^*(p) = \begin{cases} 0 & \text{if } \|p\|_\infty \leq 1, \\ +\infty & \text{if } \|p\|_\infty > 1, \end{cases}$$

and

$$S^*(-A^*p) = \frac{\mu}{2}\|u - A * p\|^2.$$

From the dual problem we get that

$$\eta^* = u + \mu \text{div} p, \tag{4.14}$$

$$w = \frac{\eta^*}{\mu},$$

and

$$p_{k+1} = \frac{p + \tau\nabla w}{\max(1, |p + \tau\nabla w|)} \tag{4.15}$$

where τ is the step length.

With (4.14) and (4.15), Chambolle's projection algorithm for the first part of (4.13) is summarized in algorithm 4. The proximity operator for g then become

$$\text{prox }_{\mu g}(u,v) = \begin{pmatrix} \text{Algorithm 4} \\ \min\{\gamma, \max\{\zeta, -\gamma\}\} \end{pmatrix}. \tag{4.16}$$

Algorithm 4 CHAMBOLLE's projection algorithm.

Require: $u \in \mathbb{R}^{NM}$, $\tau < 1/8$(step size), tol > 0(stopping criteria) , $\mu > 0$
Ensure: η is an approximate solution of (4.3).

$p \in \mathbb{R}^{2\times NM}$
$w \leftarrow \frac{u}{\mu} + \text{div}p$
$r \leftarrow \|w - u\|_2$
while $r >$ tol **do**
  $w_{old} \leftarrow w$
  $temp_p \leftarrow p + \tau\nabla w$
  $p \leftarrow \frac{temp_p}{max(1,|temp_p|)}$
  $w \leftarrow \frac{u}{\mu} + \text{div}p$
  $r \leftarrow \|w - w_{old}\|_2$
end while
$\eta \leftarrow w\mu$

Now, we insert h into (4.12) and the proximity operator becomes

$$\text{prox }_{h}(u,v) = \arg\min_{\eta,\zeta} \left[\frac{1}{2}\|\eta - u\|_2^2 + \frac{1}{2}\|\zeta - v\|_2^2 + i_{\mathcal{F}}(u,v)\right].$$

Here, we have a convex optimization problem, which can be written as

$$prox_h(u,v) = \arg\min_{\eta,\zeta} \left[\frac{1}{2}\|\eta - u\|_2^2 + \frac{1}{2}\|\zeta - v\|_2^2\right] \quad \text{such that} \quad F\eta - \zeta = FY.$$

In appendix A, it is proven that it is necessary and sufficient that the solution satisfies the KKT conditions. Therefore, we will use the KKT conditions to find the solution of the proximity operator. We start with formulating the Lagrangian

$$\mathcal{L}(u,v;\lambda) = \frac{1}{2}\|\eta - u\|_2^2 + \frac{1}{2}\|\zeta - v\|_2^2 - \lambda^T(F\eta - \zeta - FY),$$

where λ is the Lagrange multiplicator. The KKT-conditions then become

$$\nabla\mathcal{L}(u,v;\lambda) = \begin{bmatrix} \nabla_\eta\mathcal{L}(u,v;\lambda) \\ \nabla_\zeta\mathcal{L}(u,v;\lambda) \end{bmatrix} = \begin{bmatrix} \eta - u - F^T\lambda \\ \zeta - v - \lambda \end{bmatrix} = 0,$$
$$F\eta - \zeta = FY.$$

From $\nabla_\zeta \mathcal{L}$, we get

$$\lambda = \zeta - v.$$

Inserted into $\nabla_\eta \mathcal{L}$, we have

$$F^T\zeta - \eta = F^T v + u.$$

This formulation together with $F\eta - \zeta = FY$ form a set of linear equations that can be written as $Ax = b$, where

$$A = \begin{bmatrix} F & -Id \\ Id & -F^T \end{bmatrix}, x = \begin{bmatrix} \eta \\ \zeta \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} FY \\ F^T v + u \end{bmatrix}.$$

The equations can be solved directly. The proximity operator then becomes

$$\text{prox }_h(u, v) = A^{-1}b. \tag{4.17}$$

The Douglas-Rachford algorithm to minimize (4.10) with the convex functional is $J(u) = \frac{1}{2}\|\nabla u\|_1$ and the proximity operators (4.16) and (4.17) is summarized in algorithm 5.

Algorithm 5 Douglas-Rachford algorithm when $J(u) = \frac{1}{2}\|\nabla u\|_2^2$.

Require: $u \in \mathbb{R}^{N\times M}$, $\lambda \in]0, 2[$, $\gamma > 0$(step size), $\text{tol}_C > 0$ (stopping criteria)
Ensure: (η, ζ) is an approximate solution of (4.10).

```
Set r > tol
η ← u and ζ ← v
while r > tol_C do
    ζ ← ζ_old
    η ← η_old
    b = (F^T Y, F^T v + u)^T
    (η, ζ)^T ← A^{-1} b
    temp_η ← CHAMBOLLE(2η − u)
    temp_ζ ← min{γ, max{2ζ − v, −γ}}
    u = u + λ(temp_η − η)
    v = v + λ(temp_ζ − ζ)
    r ← ‖η − η_old‖_2
end while
```

Chapter 5

Numerical experiments in 1D

We will now study the results of the numerical experiments in one dimension. We have implemented algorithm 1 and algorithm 2 in Matlab, which can be seen in appendix C.1. Dykstra's projection algorithm is implemented in two versions. One version is a general version that project onto all structures of subsets. The second version is specified for the dyadic subsets, where the projections for each level are performed simultaneous. We perform the projections simultaneously by vectorizing the implementation. This implementation will be talked about as the dyadic version. The vectorized subsets were explained at the end of chapter 4.1. The general implementation of Dykstra's projection algorithm is called dykstra and the vectorized implementation is called dykstra_dyadic.

We will use the implementations in one dimension to compare the results when the index set is $\hat{I}$ and I and to study how the variance of the noise and the multiresolution bound γ affect the result. First, we will compare the result when $\hat{I}$ and I are used as index sets by study the error for the reconstructions and by studying the numerical efficiency of the two different implementations of algorithm 2. Second, we will study how the variance of the noise and the multiresolution bound γ affect the result of the algorithms. Finally, we will also test the algorithms on four other functions.

The multiresolution bound γ is often given as a constant C times the variance. To find the constant C we use the condition

$$\frac{N^2 e^{-\frac{C^2}{2}}}{C} < 0.1. \tag{5.1}$$

By means of (2.6), (5.1) implies

$$\mathbb{P}(\|\varepsilon\|_B > C\sigma) < 0.1. \tag{5.2}$$

5.1 Test functions

To verify and test the implementations we will primarily use the test function defined as

$$f(x) = \cos(\pi x).$$

This function is plotted in subfigure 5.1a.

In chapter 5.4, we will test the dyadic implementation on the functions heavisine, doppler, blocks and bumps. They are presented in [6]. They all have discontinuities, so theoretically the algorithm will not manage to solve them, but it is interesting to see how well or how bad the implementation manages them.

We start with the heavisine function, which is defined as

$$f(x) = \sin(4\pi x) - \text{sign}(x - 0.3) - \text{sign}(0.72 - x). \tag{5.3}$$

The function is normalized and plotted in subfigure 5.1b. The test function is normalized by dividing the function by its largest value. As the plot shows, the heavisine function has jumps at $x = 0.3$ and $x = 0.72$. These jumps are are discontinuities and not differentiable.

The doppler function function is defined as

$$f = \sqrt{x(1-x)}\sin\left(\frac{3\pi}{x + 0.5}\right). \tag{5.4}$$

Doppler is a sine function and as we can see in figure 5.1c, where the function is normalized, both the wave length and the amplitude are small at $x = 0$ and grow slowly until $x = 1$.

The block function is defined as

$$f(x) = \sum_{j=1}^{11} h_j K(x - t_j), \tag{5.5}$$

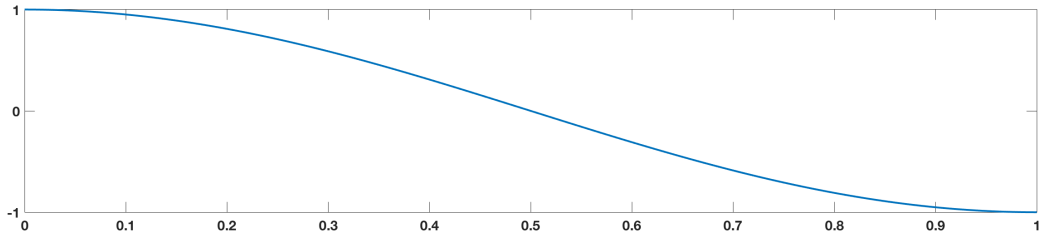
where

$$\begin{aligned} K(x) &= \frac{1 + \text{sign}(x)}{2}, \\ t &= \{0.1, 0.13, 0.15, 0.23, 0.25, 0.40, 0.65, 0.76, 0.78, 0.82\}, \\ h &= \{4, -5, 3, -4, -4.2, 2.1, 4.3 - 3.1, 2.1, -4.2\}. \end{aligned}$$

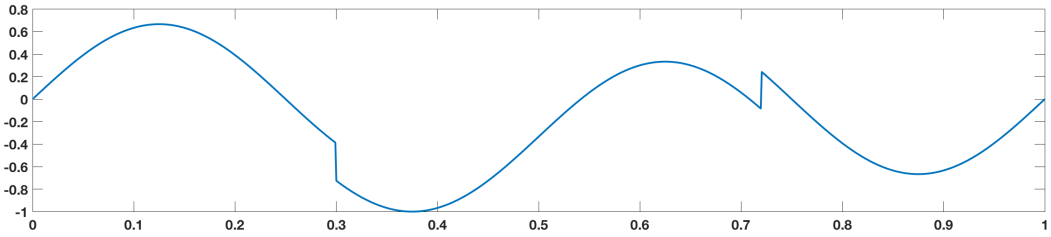
Subfigure 5.1d shows a normalized version of the block function. As we can see the function contains many jumps and between the jumps is the function constant.

The last function is the bump function

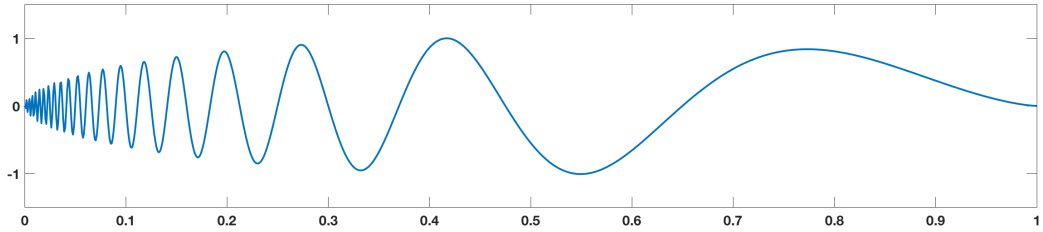
$$f(x) = \sum_{j=1}^{11} h_j K\left(\frac{x - t_j}{w_j}\right), \tag{5.6}$$



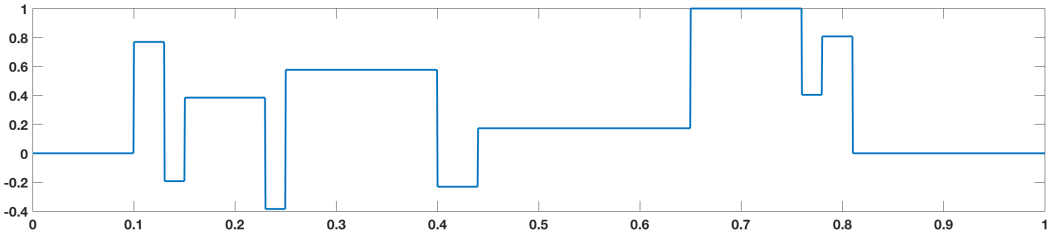
(a) The cosine function on the interval $x \in (0, 1)$.



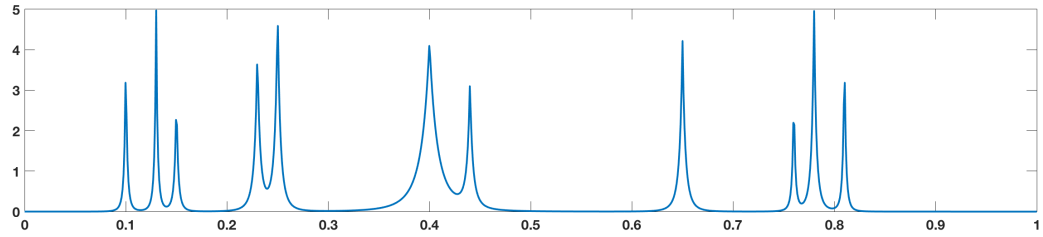
(b) The normalized heavisine function on the interval $x \in (0, 1)$.



(c) The normalized doppler function on the interval $x \in (0, 1)$.



(d) The normalized block function on the interval $x \in (0, 1)$.



(e) Bumps on the interval $x \in (0, 1)$

Figure 5.1: Test functions for the one dimensional implementations.

where

$$
\begin{aligned}
K(x) &= (1+|x|)^{-4}, \\
h &= \{4, -5, 3, -4, -4.2, 2.1, 4.3 - 3.1, 2.1, -4.2\}, \\
w &= \{0.005, 0.005, 0.006, 0.01, 0.03, 0.01, 0.01, 0.005, 0.008, 0.005\}, \\
t &= \{0.1, 0.13, 0.15, 0.23, 0.25, 0.40, 0.44, 0.65, 0.76, 0.78, 0.81\}.
\end{aligned}
$$

As we can see in subfigure 5.1e, the function is most of the time constant at zero, but there are some peaks.

5.2 Comparison of $\hat{I}$ and I

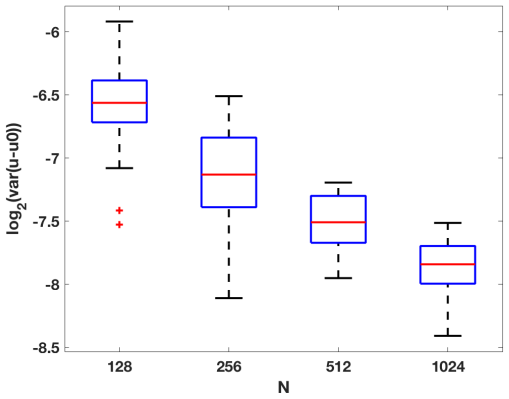
We will now compare the results for using all the consecutive index sets $\hat{I}$ and only the dyadic index sets I. For the index set $\hat{I}$, the general implementation is used and for the dyadic index set I, the dyadic implementation is used. To compare the index sets, we will study the behavior of the error and the numerical efficiency for the implementations. We choose 128, 256, 512 and 1024 number of data points, the variance $\sigma = 0.1$, C is defined as in (5.1) and the stop criteria is set to be $0.1\log(N)$. For each number of data points, we have performed 30 reconstructions of $f = \cos(\pi x)$ with different noise ε. For each reconstruction, we have then computed the variance of the error

$$\text{VAR}\left[f - u\right].$$

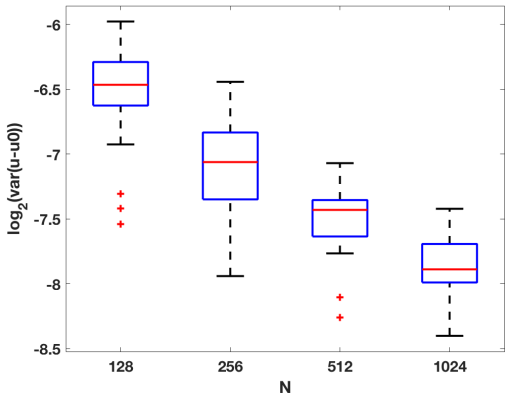
The logarithm of the variances is plotted in a box plot, with a box for each number of data points. We will start with studying the differences for the general implementation for a varying number of data points, in particular the behavior of the error when the number of data points increase.

Figure 5.2 shows the logarithmic box plots of the variance of the error. In subfigure 5.2a, we have used the complete index set $\hat{I}$ and in subfigure 5.2b, we have used the dyadic index set I. Comparing the two plots, we see that the variance of the error is smaller when we use the index set $\hat{I}$, which is as expected because some information can get lost with the dyadic subsets. Some information can get lost since the dyadic subsets are not as strongly coupled. From the scatter plots we have the slope of the regression lines. For the general implementation, the slope is -0.4198 and its confidence interval at level 0.95 is $[-0.9859, 0.1463]$, and for the dyadic implementation is the slope -0.4368 and its confidence interval at level 0.95 is $[-1.0436, 0.1701]$. We see that the slopes are pretty similar and around -0.4. From the slopes, we have that the variance of the error for both methods have polynomial dependence of the number of grid points N, with rate around $N^{-0.4}$. We would expect the error to behave like $N^{-\frac{4}{5}}$ for $\gamma_N \sim \log N$ [9]. Note that size of the confidence interval for the slope is so large that the result is not reliable. To obtain better results a larger number of test will be needed.

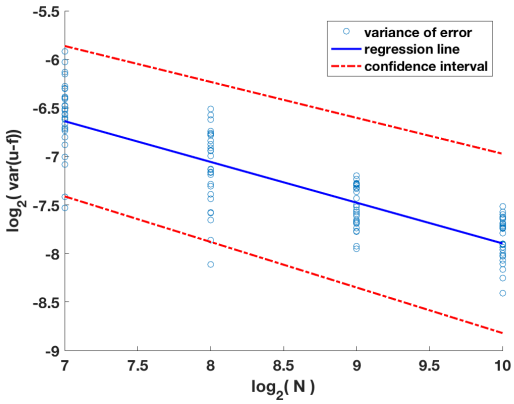
Next, we will study the difference in numerical efficiency. We will look at the number of iterations in algorithm 1, which is the number of times algorithm 2 is used, and the mean number of iterations in the outer loop of algorithm 2 in one simulation. We start with the number of iterations in



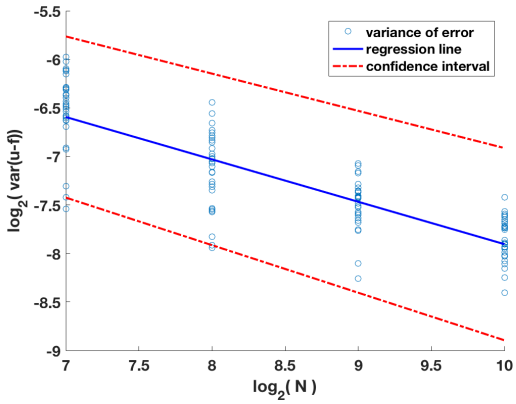
(a) Box plot of the general implementation with all consecutive index sets



(b) Box plot of the dyadic implementation with only the dyadic index sets.



(c) Scatter plot of the general implementation with all consecutive index sets. The slope of the regression line is -0.4198 and its confidence interval at level 0.95 is $[-0.9859, 0.1463]$.



(d) Scatter plot of the dyadic implementation with only the dyadic index sets. The slope of the regression line is -0.4368 and its confidence interval at level 0.95 is $[-1.0436, 0.1701]$.

Figure 5.2: The figure show a box plot and a scatter plot of the logarithm of base 2 of the variance of the error for the algorithm 1 when all the consecutive subsets of R^N is used, solved with the general implementation (to the left) and only the dyadic subsets is used, solved with the dyadic implementation (to the right). Each box in the box plots show the error for one of the number of data points $N = 128, 256, 512, 1024$. In the scatter plots are the logarithm of base 2 of the variance of the error plotted against the logarithm of base 2 of N. In the scatter plot also the regression line and its confidence interval at level 0.95 are plotted. The simulations are solved with $\sigma = 0.1$ and C as in (5.1).

Table 5.1: The table shows the mean number of times algorithm 2 runs for each of the number of data points, which means the number of iterations in algorithm 1. $\hat{I}$ shows the number of times when h is projected onto all the consecutive subsets of R^N with the general implementation and I shows the number of times when h is projected onto the dyadic subsets with the dyadic implementation.

N	$\hat{I}$	I
128	27.5	27.4
256	28.0	28.2
512	28.0	28.0
1024	28.4	28.4

algorithm 2, which in all cases was 2. The number of iterations in algorithm 2 is so low because of the stopping criteria. With lower tolerance, we would expect a higher number of iterations. The same can apply for the number of iterations in algorithm 1.

Now, we study the number of iterations in algorithm 1. The number of iterations was counted at the same time as the error was measured and for each number N the mean number of iterations in algorithm 1 is represented in table 5.1. We can see in table 5.1 that the number of iterations does not vary much. It is therefore difficult to draw an exact conclusion about the number of iterations. However, we know from the short discussion at the end of chapter 4.1 that each run of algorithm 2 should be significantly faster when we use the dyadic index set I than when we use the index set $\hat{I}$, and even faster for I here since we are performing the projections simultaneously. This behavior is because with the index set $\hat{I}$ are there then performed $O(N^2)$ iterations in the inner loop. While for the index set I the number of subsets is fewer and for each level s the projections are performed simultaneous, which reduces the number of iterations in the inner loop to $O(\log(N))$. We can conclude that the implementation is faster when we choose I as index set.

We can conclude that the implementation with either of the subsets will give a reasonably good approximation for f for large enough N. The error will not be so much better if we choose all the consecutive subsets if compared to the time we gain by using the dyadic subsets. We will use the dyadic subsets for the rest of the numerical experiments.

5.3 The effect of variance and multiresolution bound

We will now study the effect of different values of γ and different variances on the approximations u. We will first study how the multiresolution bound γ alone affects u, then the variance σ alone, and finally we look at the situation where we vary variances and let γ vary with the variance. For all the three simulations we use the same data set Y, with the number of data points $N = 1024$, which we obtain by adding $f(x) = \cos(\pi x)$ and the noise ε together.

We start by studying how the multiresolution bound γ affect the approximation. We approximate for the different values of γ, $\gamma = 0.01$, $\gamma = 0.1$, $\gamma = 0.5$, $\gamma = 1.0$, $\gamma = 2.0$ and $\gamma = 4.0$, and always use the same variance $\sigma = 0.1$. We can see in figure 5.3 how the different values of γ affect the

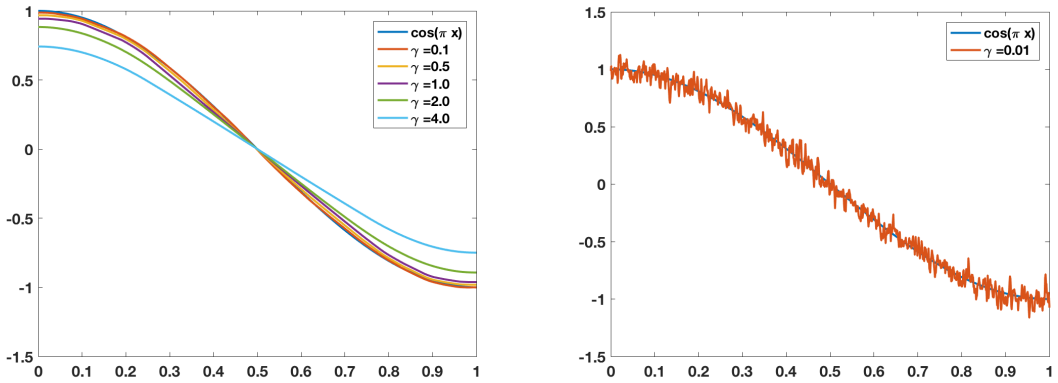


Figure 5.3: The plot to the left shows the results when γ varies while the variance remains constant at $\sigma = 0.1$. We have solved the problem with 1024 data points and $\gamma = 0.1$, $\gamma = 0.5$, $\gamma = 1.0$, $\gamma = 2.0$ and $\gamma = 4.0$. The plot to the right shows an example of overfitting, where $\gamma = 0.01$.

result. In both plots, the original function $f(x) = \cos(\pi x)$ is plotted as a reference. In the plot to the right is the result for $\gamma = 0.01$. As we can see the result is overfitted, which means that it fits too well to the data points and the noise is not removed. Therefore, we can conclude that $\gamma = 0.01$ is too small. In the plot to the left, the results have same form as f, but the amplitude varies. We can see that the approximation is very close to f when $\gamma = 0.1$ and the result for $\gamma = 4.0$ is far from f. The amplitude is reduced because $\|\nabla u\|_2$ minimizes the energy in the function. Therefore, it will reduce the amplitude as much as possible. Larger γ will allow $\|\nabla u\|_2$ to reduce the amplitude more because, as discussed in chapter 2.1, the larger the multiresolution bound is, the more continuous functions is removed as noise and since. We should try to use a γ as small as possible to retain as much energy as possible, but not too small so that the result is overfitting.

Next, we study how the variance affects the reconstruction u when $N = 1024$ and $\gamma = 0.1\log(N)$. We will study affects by looking at the results for $\sigma = 0.1$, $\sigma = 0.3$ and $\sigma = 0.5$.The larger the variance, the more scattered the data points. This scattering is discussed in chapter 2 and illustrated in figure 2.1. We compare the approximations to the original function $f(x) = \cos(\pi x)$ and see that the results for $\sigma = 0.1$ and $\sigma = 0.3$ seem to follow the original function well and for $\sigma = 0.5$, the result is overfitting. The reconstruction for $\sigma = 0.3$ has larger amplitudes than reconstruction for $\sigma = 0.1$, hence there is reason to think that $\gamma = 0.1\log(N)$ is more suitable for $\sigma = 0.3$. . If we investigate the shape of the reconstructions, we see that the reconstruction for $\sigma = 0.1$ has a cosine-shape, while the reconstruction for $\sigma = 0.3$, the cosine-shape is not completely reconstructed since the amplitudes has different high. The reason for the cosine-shape is not completely reconstructed for $\sigma = 0.3$, might be because the noise, for $x \in (0.8, 1)$ is clustered below -1 and therefore pulls the reconstruction downwards.

A method to improve the results in figure 5.4 might be to vary γ with the variance σ. Now, we define $\gamma = C\sigma$, where the constant C is defined as in (5.1) such that γ is dependent of the variance. We can see in figure 5.5 that the results improve. It is easiest to see the improvements by investigating the result for the data set with largest variance. In figure 5.4 the result is overfitting, but in figure

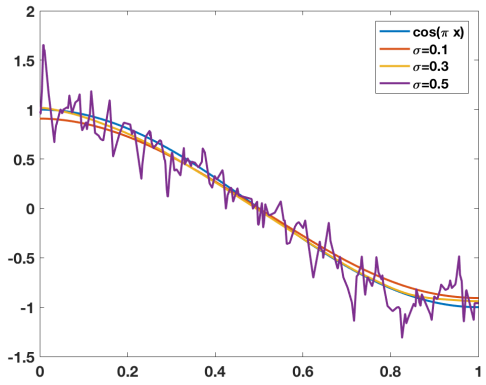


Figure 5.4: The plot shows the results when the variance are varied while the $\gamma = 0.1 \log N$. We have solved the problem with 1024 data points and the different variances are $\sigma = 0.1$, $\sigma = 0.3$ and $\sigma = 0.5$.

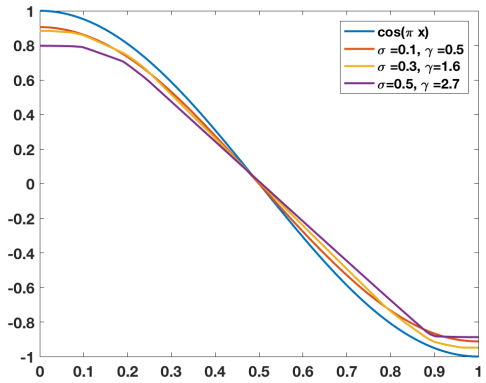


Figure 5.5: The plot shows the results when the variance are varied and $\gamma = C\sigma$, where the C is as in (5.1). We have solved the problem with 1024 data points and the different variances are $\sigma = 0.1$, $\sigma = 0.3$ and $\sigma = 0.5$.

5.5 the noise is removed and the function is a cosine function. The amplitude is significant smaller than f and, as discussed for the varying multiresolution bound, there are reasons to think that γ is too large.

In summary, the variance and the multiresolution bound affect the result. If we have a small multiresolution bound, the result can be overfitted, but a too large multiresolution bound will reduce the energy in the function. If the variance is large, we will need a larger multiresolution bound to prevent overfitting. We can conclude that, if we know the size of the variance, the multiresolution bound should be scaled with the variance for a better result.

5.4 Further numerical tests

We will now test the algorithm on some other test functions, heavisine, doppler, blocks and bumps [6]. Our method is not constructed to solve these kind of functions, but it is still interesting to see how well the method manages to reconstruct these. For all these functions, the vectorized implementation of algorithm 2 is used. The data sets have size $N = 1024$ and the noise is generated with the variance $\sigma = 0.1$. For all the test functions, the multiresolution bound $\gamma = C\sigma$ is found such that C satisfies (5.1).

We start with the normalized heavisine function (5.3). The heavisine function has jumps, which are discontinuities and not differentiable, at $x = 0.3$ and $x = 0.72$. Both the original function f, the result u and the data points are plotted figure 5.6, as a blue function, a red function and dots, respectively. We can see that in the reconstruction u, the discontinuities has been smoothed out. The discontinuity at $x = 0.3$ is not detectable at all in the reconstruction. The discontinuity at $x = 0.72$ is shown as a bump. Even though the discontinuities are not detectable in the reconstruction, the main sinus-shape is still clear. The amplitudes are a bit smaller in u than in f, so we could have tried with a smaller γ. Even the discontinuities are smoothed out and the amplitudes are lower, the local extremum is detectable.

Now, we will look at the normalized doppler function (5.4). This function is a sine function where both the wave length and the amplitude are small at $x = 0$ and grow slowly until $x = 1$. The doppler function is plotted together with the data points and the reconstruction in figure 5.7. In subfigure 5.7a, the function is approximated with 512 data points and in subfigure 5.7b, the function is approximated with 1024 data points. We can see that the method do not detect the shortest waves, but as the frequency increases the method detects the waves. We expect the reconstruction to behave this way because there are few data points around each local extremum for short wave lengths, and they are therefore harder to detect. When we compare the reconstruction for $N = 512$ and $N = 1024$, we see that the reconstruction for $N = 1024$ detects the waves earlier. This behavior is as expected because we will have enough data points at smaller wave lengths for a larger number of data points. We can therefore expect the result to be even better for larger N.

Now, we will study the reconstruction of the normalized block function (5.5). As we can see in figure 5.8, which shows f, the reconstruction u and the data points, the block function contains

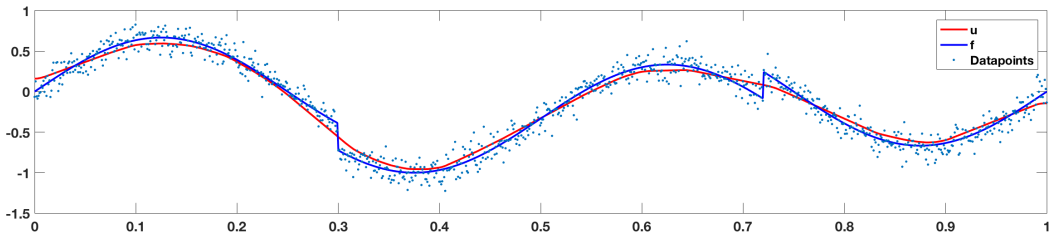
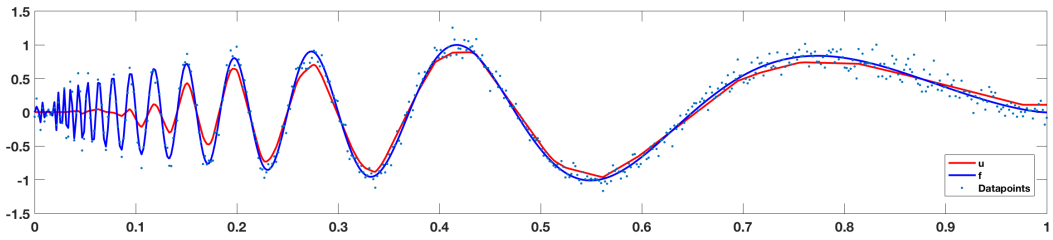
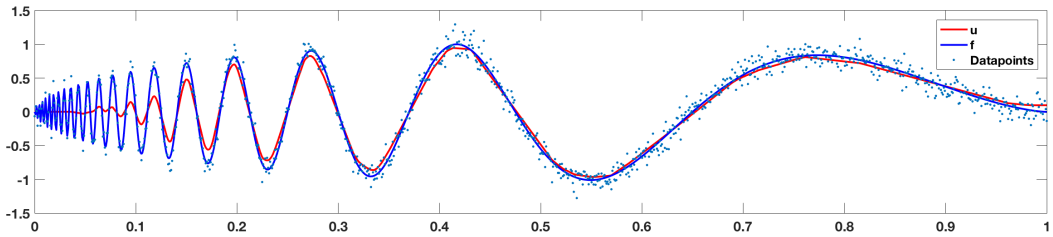


Figure 5.6: The plot shows the result when the normalized heavisine function is used as test function. We have $N = 1024$ and $\sigma = 0.1$. The blue function is the test function f, the red function is the reconstruction u and the dots are the data points Y.



(a) $N = 512$



(b) $N = 1024$

Figure 5.7: The plot shows the reconstruction when the normalized doppler function is used as test function. In the upper plot, we have $N = 512$ and $\sigma = 0.1$ and in the lower plot, we have $N = 1024$ and $\sigma = 0.1$. In both figures, the blue function is the test function f, the red function is the reconstruction u and the dots are the data points Y.

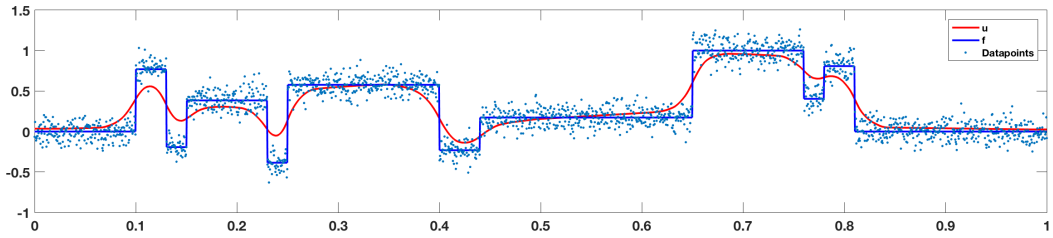
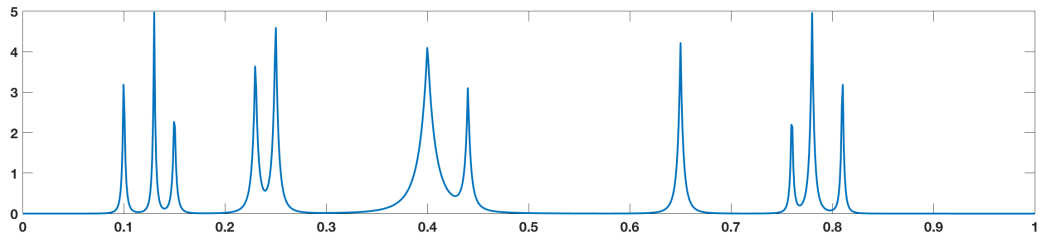


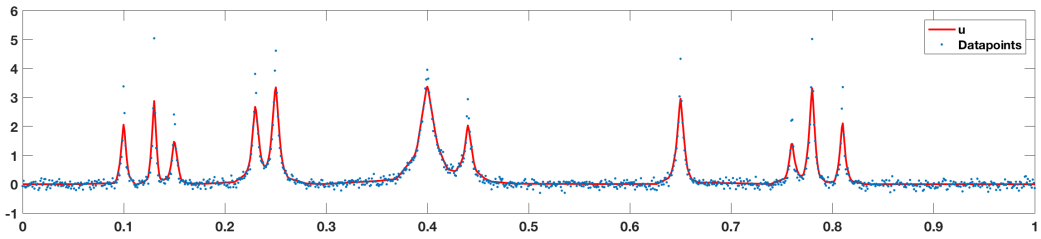
Figure 5.8: The plot shows the result when the normalized block function is used as test function. We have $N = 1024$ and $\sigma = 0.1$. The blue function is the test function f, the red function is the reconstruction u and the dots are the data points Y.

many jumps and is not differentiable in these points. We can see that these discontinuities are problematic and are smoothed out in the reconstruction. Even if the blocks are smoothed out, the modes are detectable. If we only want to know approximately where the modes are, this method can be used.

The last function we will study is a function with bumps (5.6). As we can see in subfigure 5.9b, the function is such that most of the time $f = 0$, but there are some peaks. The reconstruction and the data points are plotted together in in subfigure 5.9b. As we can see in the figure, all the peaks are in reconstructed, but the peaks are not that high as in f. The reason for the peaks to be lower in the reconstruction might be the sudden change is interpreted as noise and the method tries to smooth it out. The method can thus be used to detect the peaks, but we can not reconstruct them



(a) The test function bumps.



(b) The data points and the reconstruction

Figure 5.9: The plots shows the result when the bumps function is used as test function. We use $N = 1024$ and $\sigma = 0.1$. The upper plot shows the test function f and lower plot shows the reconstruction u and the data points Y as dots.

exactly

We can conclude that algorithm 1 to a certain extent manages to reconstruct functions that are not continuous and differetiable, but the discontinuities are smoothed out. Too sudden change and small oscillations are either smoothed out or not detected at all.

Chapter 6

Numerical experiments in 2D

In two dimensions, we have studied two different solution methods, one where the convex functional is $J(u) = \frac{1}{2}\|\nabla\|_2^2$ and one where the convex functional is the total variation norm $\|\nabla\|_1$. For these two, we have developed two different algorithms. For the quadratic regularization problem we have used Douglas-Rachford algorithm as presented in chapter 4.2 and for the total variation norm, we have used Douglas-Rachford algorithm with Chambolle's projection algorithm as presented in chapter 4.3. We have implemented both algorithms in Matlab, and we will now study how the algorithms minimize the given problems.

Before we can start with the numerical experiments, we will present the test functions and we need to set the value for the multiresolution bound. We will start to study the quadratic regularization problem and then study the total variation problem. In both cases, we will use test functions customized to the problems. At the end, we will compare the two implementations and try them on different images. For all experiments, the tests are performed on a quadratic grid such that $N = M$.

6.1 Test functions

The test functions which are used to test the implementations in two dimensions are shown in figure 6.1. To verify and test the implementations, we will use the image phantom, shown in subfigure 6.1a, and a test function is a sum of multivariate Gaussian distributions, shown in subfigure 6.1b, defined as

$$\sum_j c_j \exp(-(x-\mu_j)^T \Sigma_j (x-\mu_j)), \tag{6.1}$$

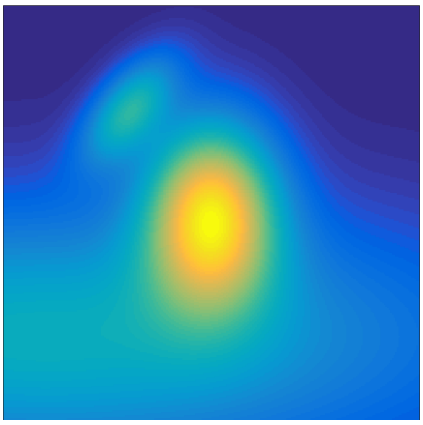
where

$$
\begin{aligned}
&c = \{0.5, 1.0, 0.5\}, \\
&\mu = \{(0.1, 0.8), (0.5, 0.5), (0.3, 0.25)\}, \\
&\Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix}, \quad \Sigma_2 = \begin{bmatrix} 20 & 0 \\ 0 & 10 \end{bmatrix} \quad \Sigma_3 = \begin{bmatrix} 40 & 20 \\ 20 & 30 \end{bmatrix}.
\end{aligned}
$$

In addition, we will test the implementations on the well-known test images Barbara and Lena. Barbara and Lena are plotted in subfigure 6.1c and subfigure 6.1d, respectively. They are both natural images, which contain discontinuities, areas with oscillations and small details. For all these test functions, we will always normalize them. We normalize them by dividing the function by its largest value.

To test the implementation for the quadratic regularization problem, we will primarily use subfigure 6.1b. The reason for this is that the L_2-norm does not manage discontinuities well, and this function is smooth. For the total variation problem, we will primarily use subfigure 6.1a. This image contains discontinuities, but not any oscillations. We will also be able to study when details

(a) Phantom



(b) Sum of Gaussians.

(c) Barbara

(d) Lena

Figure 6.1: Test functions and test images used in the two dimensional cases.

disappear from the reconstruction because of the details of different sizes.

6.2 Choice of the multiresolution bound

In chapter 2.1 and in the numerical experiments in one dimension in chapter 5 we discuss the importance of choosing the multiresolution bound γ wisely. The aim with the multiresolution norm is to separate the continuous functions and the noise. As discussed in chapter 2.1, the multiresolution norm cannot separate the noise and small continuous functions. If we choose a large multiresolution bound γ, larger continuous functions are interpreted as noise. The reconstruction will then be more and more flattened out, the larger γ is. On the other hand, by using a too small γ, the chance for overfitting is high. So, we want a γ such that the residual contains as few continuous parts as possible, but it cannot be so small that the reconstructed function is too close to the observed data.

We have assumed the residual to behave as the noise when we have a good reconstruction. Then, we can define γ by studying how the multiresolution norm of the noise is distributed. We find the distribution by estimating

$$\mathbb{P}(\|\varepsilon\|_B \leq \gamma) \geq \alpha, \tag{6.2}$$

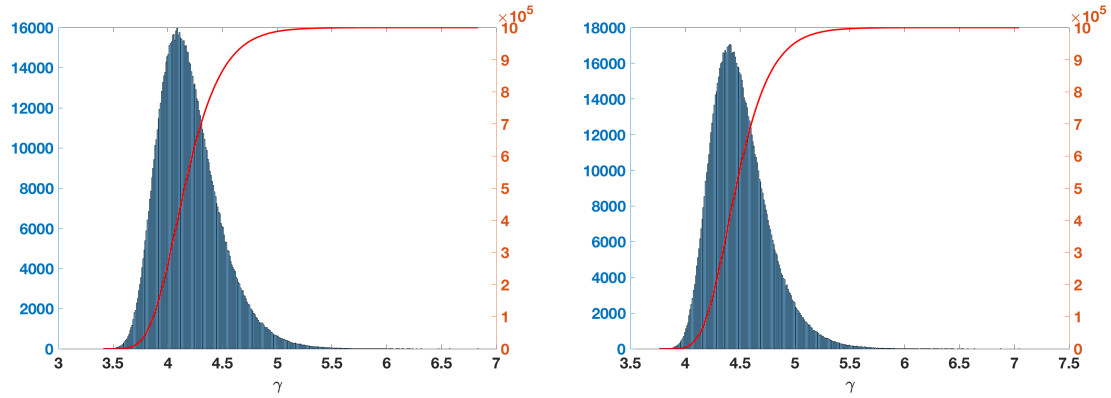
where $\varepsilon \in \mathbb{R}^{N\times N}$ is i.i.d. Gaussian with variance $\sigma = 1$ and $0 \leq \alpha \leq 1$ is some percentile. The condition (6.2) means that the probability for $\|\varepsilon\|_B \leq \gamma$ is equal or larger than α, which means that the condition $\|\varepsilon\|_B \leq \gamma$ is satisfied a fraction α of the incidents. Thus, if the variance of ε is $\sigma = 1$, we prescribe a value of α and choose γ so the condition (6.2) is satisfied. The variance is only scaling each sample $\varepsilon_{i,j}$ with σ, so it does not affect the shape of the distribution. Therefore, if the variance of ε is $\sigma \neq 1$, we can first find a parameter γ corresponding to a variance $\sigma = 1$, and then multiply it with σ.

We find the distribution of the multiresolution norm by estimating

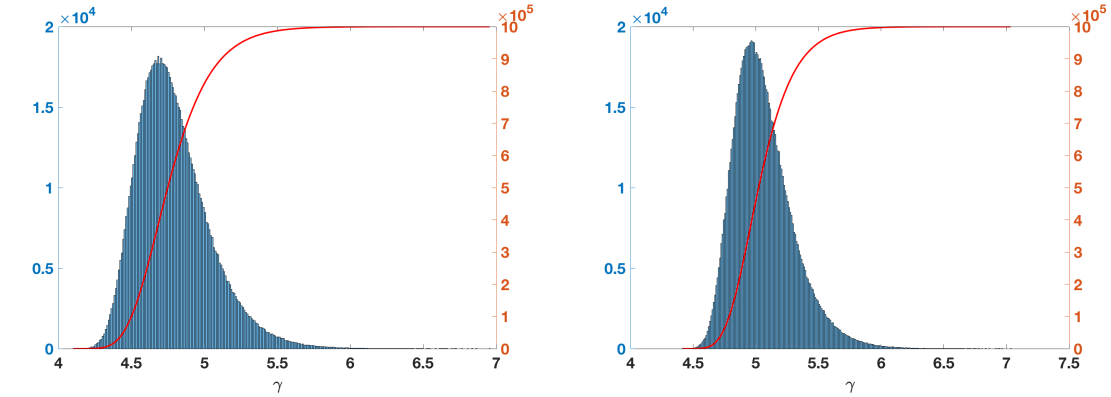
$$\gamma \mapsto \mathbb{P}(\|\varepsilon\|_B \leq \gamma).$$

The probability distribution is estimated by calculating $\|\varepsilon\|_B$ for many different realizations of ε. We then plot the results in a histogram. Together with the histogram the cumulative sum of the number of results from small to large γ is plotted as a graph. The value of γ where (6.2) is at the point where the cumulative sum is the fraction α of the total number of realizations.

In figure 6.2, the results of the estimation of the probability distribution for $N = 128$, $N = 256$, $N = 512$ and $N = 1204$ are plotted as histograms. From the figures, the values for γ are found for $\alpha = 0.90$ $\alpha = 0.95$ and $\alpha = 0.99$ and shown in table 6.1.



(a) $N = 128$ **(b)** $N = 256$



(c) $N = 512$ **(d)** $N = 1024$

Figure 6.2: The plots shows the probability distribution of the multiresolution norm when $\varepsilon \in \mathbb{R}^{N \times N}$ is i.i.d. Gaussian as histogram and the cumulative sum of the results from small to large γ plotted as a graph.

Table 6.1: The table shows the values of γ for the percentile $\alpha = 0.90$ $\alpha = 0.95$ and $\alpha = 0.99$ for $N = 128$, $N = 256$, $N = 512$ and $N = 1204$.

$\alpha \setminus N$	128	256	512	1024
0.90	4.57	4.85	5.12	5.37
0.95	4.70	4.99	5.25	5.50
0.99	5.03	5.30	5.54	5.78

6.3 Numerical experiments for the quadratic regularization problem

The optimization problem from chapter 2.3 with the quadratic regularization term (2.10) is defined as

$$\min_{u \in \mathbb{R}^\Gamma, v \in \mathbb{R}^I} \frac{1}{2}\|\nabla u\|_2^2 + i_{\mathcal{C}}(v) + i_{\mathcal{F}}(u, v).$$

Table 6.2: The table shows the tolerance in the stopping criteria used for the implementation of Douglas-Rachford algorithm for the quadratic regularization problem.

N	128	256	512
tol	e^{-5}	$e^{-4.5}$	e^{-4}

To solve this problem we have implemented the Douglas-Rachford algorithm presented in algorithm 3 in Matlab. The implementation can be seen in appendix C.2. As test function, we will use the Gaussian test function shown in figure 6.1b. This test function is smooth, which is what this optimization problem is constructed to minimize.

To ensure that the reconstruction is good enough and to prevent an unnecessarily long run time and possible overfitting, we must choose the stopping criteria wisely. All theoretical stop criteria need strict convexity and our method is only convex so we cannot use the theoretical stop criteria. Therefore, the stopping criterion for our Douglas-Rachford algorithm is chosen heuristically and based on numerical experiments. A solution is good enough when the constraint with the multiresolution norm is satisfied and J and the error is small , where we define the error after each iteration as

$$\text{error} = \frac{\|u_n - f\|_2}{\sqrt{NM}}.$$

During the reconstruction, we will not have the original function. We cannot calculate the error and the constraint is not that useful to help us choosing the stopping criteria. Therefore, we choose to use the step length as a stopping criteria and define the stopping criteria as

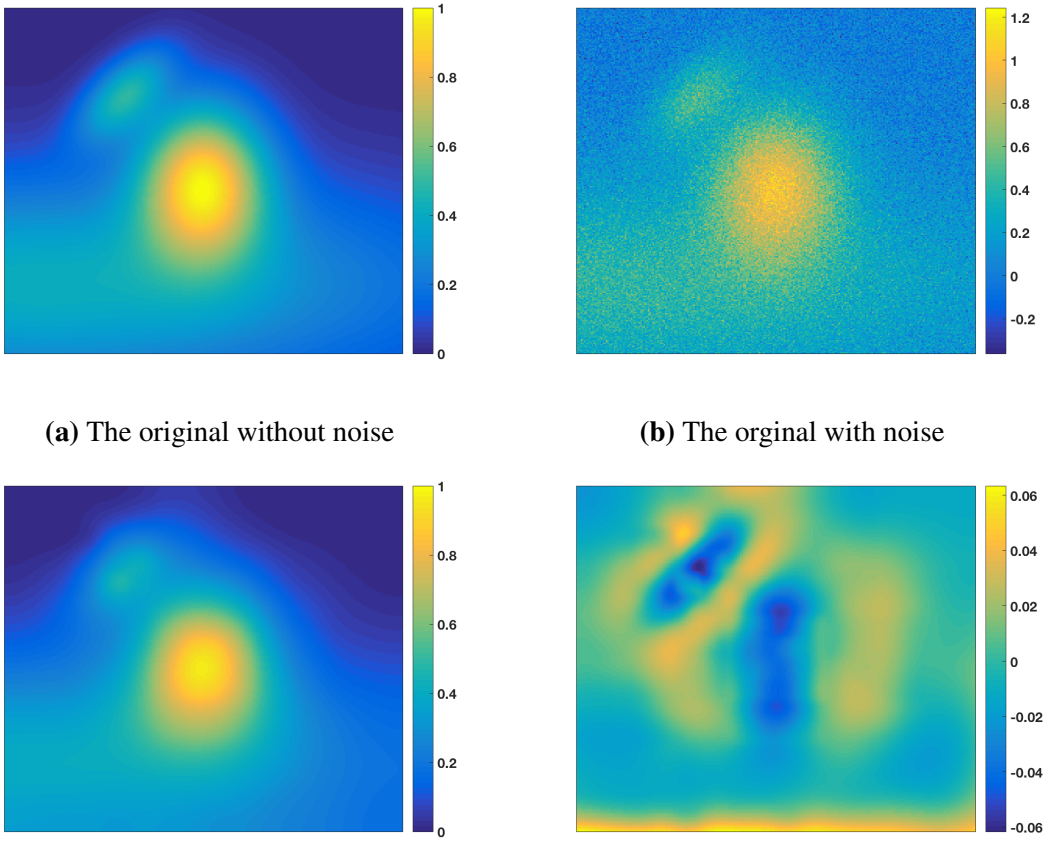
$$\|u_n - u_{n-1}\|_2 \leq \text{tol},$$

where $\text{tol} > 0$ is some tolerance. After a certain number of iterations, the error and the value of J remains constant. So, the value of tol is chosen such that the algorithm stops when the value of the error and J are stabilized. The values of the tolerance for $N = 128$, $N = 256$ and $N = 512$ are shown in table 6.2.

The result of running algorithm 3 on the test function, where $N = 256$ and the variance of the noise is $\sigma = 0.1$, is shown in figure 6.3.

Subfigure 6.3a and subfigure 6.3b show the test function without and with noise, respectively, subfigure 6.3c shows the reconstruction and subfigure 6.3d shows the residual. As we can see, the noise makes a lot of changes in the original image, but the reconstruction after we have used algorithm 3 is very close to the original function. The plot of the residual shows that the largest error in one pixel is around ± 0.06, and occurs close to the boundary and the maxima.

The algorithm seems to behave correctly, but to be sure we will do some further tests. We will perform some statistical test of the error and study how the error behaves with different variances. The results of these will be presented and discussed in the next sections.



(a) The original without noise

(b) The orginal with noise

(c) Result

(d) Residual

Figure 6.3: This figure shows the original image with and without noise, the result after denoising and the residual. The grid size is $N = 256$, the noise has variance $\sigma = 0.1$ and the multiresolution bound $\gamma = 4.85$. The colorbar for subfigure 6.3a and subfigure 6.3c is scaled between 0 and 1 and the error lies between -0.06 and 0.06.

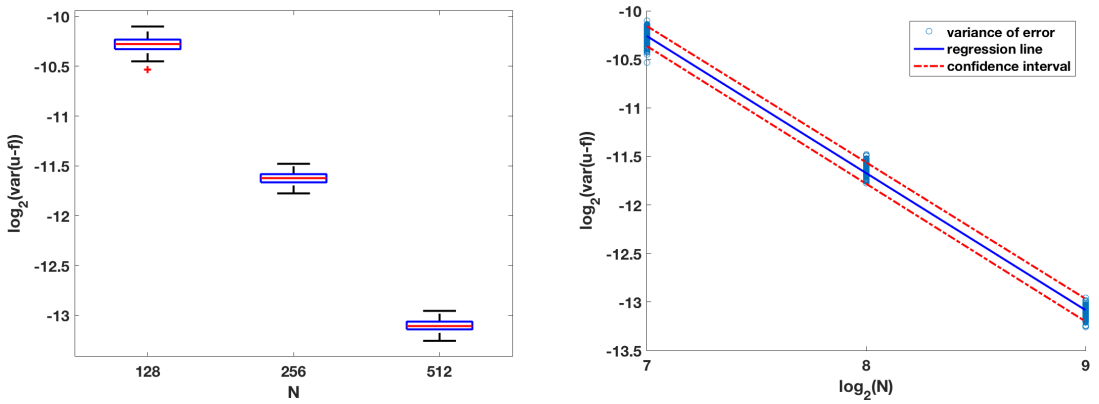
6.3.1 Error

We will now study the error of the algorithm. To study the error we have two approaches; The behavior of the error for different grid sizes is studied and the connection between the error and the multiresolution norm of the noise.

We start with studying the behavior of the error for different grid sizes by observing how the variance of the error changes for varying grid size. The variance of the error is found by calculating the error in each pixel, then calculating the variance of this error. The variance of the error is then

$$\text{error} = \text{VAR}(f - u). \tag{6.3}$$

The variance of the error is calculated for 200 experiments for the grid sizes $N = 128$, $N = 256$ and $N = 512$ with the variance of the noise $\sigma = 0.1$ and the multiresolution bound for $\alpha = 0.90$.



(a) box plot **(b)** Scatter plot with regression line

Figure 6.4: The subplots show the variance of the error for the quadratic regularization problem. The logarithm of base 2 of the variance of the error is plotted against the side lengths in a box plot and in a scatter plot. Each box in the box plot shows variance of the error for $N = 128$ $N = 256$ or $N = 512$ number of data points in each direction. In the scatter plot is also a regression line and the confidence interval plotted. The regression line has slope -1.4 and its confidence interval at level 0.95 is $[-1.51, -1.30]$. The simulations are performed with $\sigma = 0.1$ and the multiresolution bound for $\alpha = 0.90$. From the plot, we can see that the variance of the error is dependent of the number of grid points.

We then plot the logarithm to the base 2 of variances against the number of grid points in each direction in a box plot and a scatter plot; Both are shown in figure 6.4. From the box plot in subfigure 6.4a, we can see that the variance of the error is not varying a lot, hence the method is quite consistent.

Subplot 6.4b shows the scatter plot of the logarithm of base 2 of the variance of error against the number of grid points in each direction. We have also performed a linear regression on the errors, the regression line and the confidence interval at level 0.95 are plotted as lines in the scatter plot. As we can see, the logarithm of base 2 of the variance of the error is dependent of the logarithm of base 2 of the number of grid points in each direction, then the error is polynomial dependent of grid size. The slope of the regression line is -1.4 and its confidence interval at level 0.95 is $[-1.51, -1.30]$. The rate for this method is then $CN^{-1,4}$ and there is a polynomial dependence. From [14, theorem 2.2.1] we have that there is at least one function in $\mathbf{H}^2$ which have $N^{-\frac{4}{3}}$ as the best possible rate. This rate is worse that our rate, but the theoretical rate lies within our confidence interval, so if we performed a higher number experiments and on larger grids, we must expect our rate to become closer to the theoretical rate. The same might apply if we uses another function, because the our function is very smooth.

Next, we study the connection between the total error and the multiresolution norm of the noise, where the total error is defined as

$$\text{error} = \frac{\|f - u\|_2}{\sqrt{NM}}. \tag{6.4}$$

This is studied by plotting the logarithm of total error against the multiresolution norm of the noise

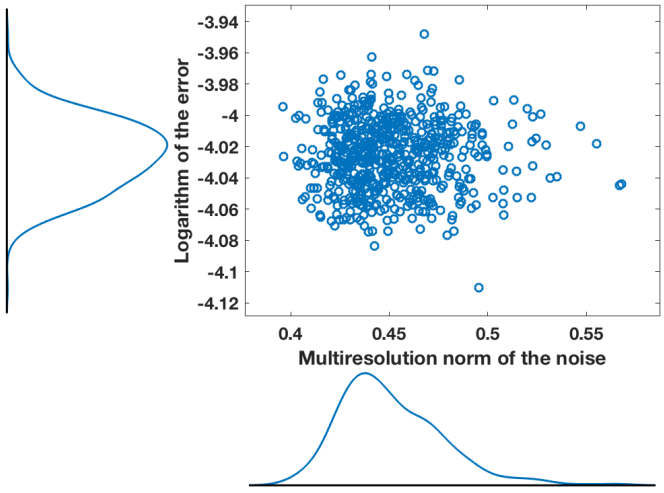


Figure 6.5: Scatter plot of the natural logarithm of the error and $\|\varepsilon\|_B$ for the quadratic regularization problem. Together with the scatter plot the marginals are plotted. The plot shows that the error is consistent and independent of $\|\varepsilon\|_B$.

in a scatter plot. For figure 6.5, this procedure is performed 600 times on the test function with grid size $N = 256$ and the variance of the noise $\sigma = 0.1$ and the multiresolution bound is chosen to be $\gamma = 0.485$. Together with the scatter plot the marginals are plotted. As we can see in figure 6.5, the logarithm of the error is almost always between -4.08 and -3.96 and is consistent regardless of the value of the multiresolution norm of the noise. The value of the multiresolution norm of the noise is almost always between 0.4 and 0.53. From this we have that the error of algorithm 5 is independent of the multiresolution norm of the noise.

To summarize, we have seen in this section that the error is independent of the multiresolution norm of the noise and that the variance of the error is polynomially dependent of the grid size.

6.3.2 Variance

We will now study how the variance of the noise affects the result. First, we will look at how the error varies with the variance in a scatter plot. Second, we will look at some reconstructions for different variances.

We start with looking at the scatter plot in figure 6.6. Here, the logarithm of the error is plotted against the logarithm of the variance of the noise. The regression line and its confidence interval are also plotted in figure 6.6. The data is found by performing the following experiment 100 times. The noise is generated, then scaled with the different values of variances between 0.01 and 1.0, then applied to the test function. Next, the test functions are reconstructed with the multiresolution bound $\gamma = 4.85\sigma$. From figure 6.6, we see that there is a dependence. The rate of the regression line is $CN^{0.72}$ and its confidence interval at level 0.95 is $[N^{0.71}, N^{0.71}]$. This rate is probably

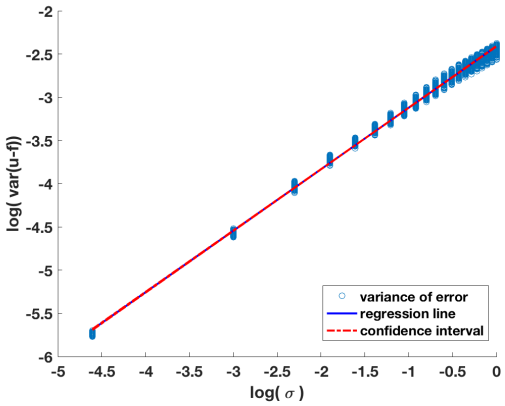
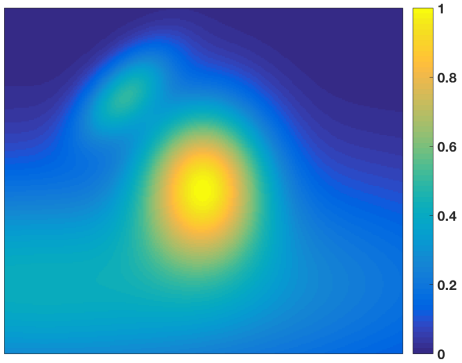


Figure 6.6: The figure shows the natural logarithm of the error plotted against the natural logarithm of the variance of the noise, for the quadratic regularization problem. The regression line and the confidence interval at level 0.95 are also plotted. The reconstruction is performed for a test function with grid size $N = 256$ and the multiresolution bound $\gamma = 4.85\sigma$. The slope of the regression line is 0.72 and its confidence interval at level 0.95 is $[0.71, 0.71]$. We can see that the error increases for larger variance of the noise and that there is an dependence between the error and the variance of the noise.

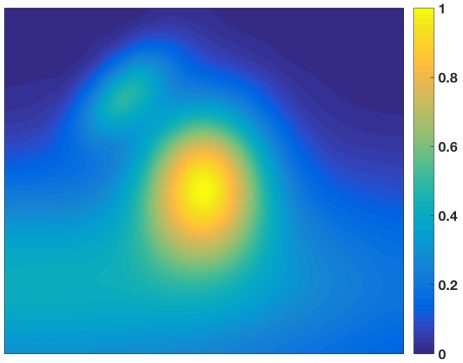
a under estimate because; First, the number of different variances between $\log(\sigma) = -4.5$ and $\log(\sigma) = -2$ is only three, therefore we have not much information about the behavior here. Second, the regression line is a straight line, while the data set seems to be slightly concave, at least after $\log(\sigma) = -1$. The concaveness of the data set pulls the regression line downwards and reduces the slope of the regression line. Because of the concaveness of the data set, the dependence between the error and the variance of the noise is not polynomial, but more complicated.

The reason for the error to increase when the variance is increasing is that the multiresolution bound is scaled with the variance of the noise, and for larger variance, the multiresolution bound is larger. Algorithm 3 is such that the reconstruction is a smooth function and the multiresolution norm of the residual is smaller than the multiresolution bound. As discussed in chapter 2.1, with a small enough multiresolution norm we can separate almost all the continuous functions and the noise, but the method can not remove all the small continuous functions from the residual. So, for larger multiresolution bound, more continuous functions are allowed in the residual. The reconstruction will for large enough variance converge to a constant function that is the average of the the original function f.

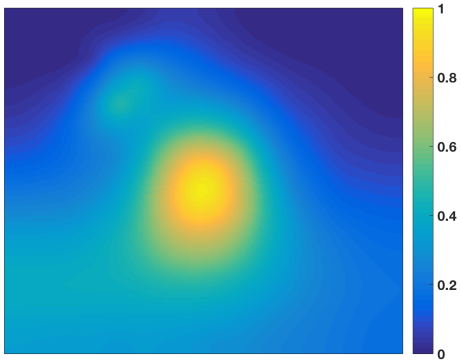
The consequence of larger variance for the reconstruction is easier to see in figure 6.7. Here the test function with side length $N = 256$ and the noise scaled with the different variances $\sigma = 0.01$, $\sigma = 0.1$, $\sigma = 0.2$, $\sigma = 0.5$ and $\sigma = 1.0$ is plotted, the original test function is plotted in subfigure 6.7a as a reference. The colorbar in the plots are equally scaled, so there is easier to detect the changes. As we can see, the reconstruction when $\sigma = 0.01$ is very close to the original. Both maxima are presented and seem to have the same values as the maxima in the original test function, for the rest of the function the reconstruction is also similar to the original. On the other hand, the reconstruction when $\sigma = 1$ is not very close to the original. The smallest maximum has totally disappeared and the value of the largest maximum is smaller than for the original. For the rest of



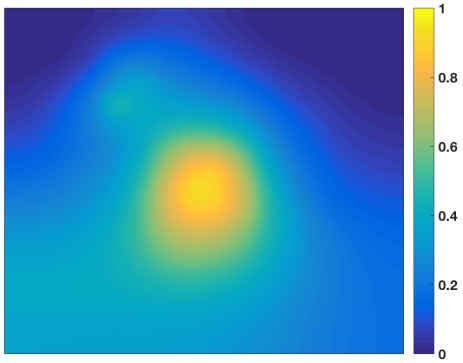
(a) The Gaussian test function



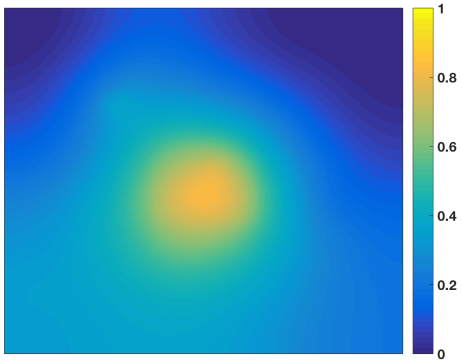
(b) The reconstruction when $\sigma = 0.01$.



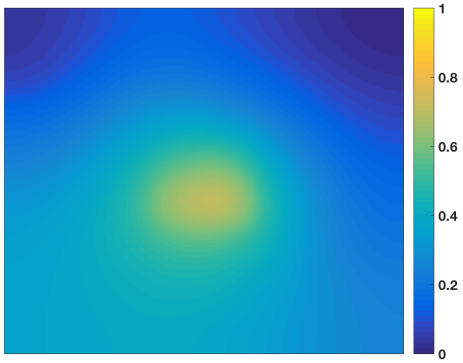
(c) The reconstruction when $\sigma = 0.1$.



(d) The reconstruction when $\sigma = 0.2$.



(e) The reconstruction when $\sigma = 0.5$.



(f) The reconstruction when $\sigma = 1$.

Figure 6.7: The figure shows reconstructions of the Gaussian test function with grid size $N = 256$ for different variances and the original function as a reference. As we can see, for the quadratic regularization problem the reconstructions is flattened more and more out the larger the variance is. The reconstructions are performed with the multiresolution bound $\gamma = 4.85\sigma$. The colorbars are scaled between 0 and 1 for all the subfigures.

the function, the reconstruction is more flattened out. For the reconstructions with the variances $\sigma = 0.1$, $\sigma = 0.2$ and $\sigma = 0.5$, the functions are more and more flattened out when the variance is increasing. The reason for the reconstruction to be more and more flattened out for larger variances is, as discussed earlier, the larger the variance is, the larger the multiresolution bound must be, and a larger number of continuous functions are allowed in the residual. In the reconstruction for the problem with $\sigma = 1.0$, we can see that the reconstruction begins to be close a constant function. For larger variances, the reconstruction will be even closer to a constant function, and the value of the function will be the average of the test function.

To summarize, the variance has a impact on the result. The larger the variance is, the larger the multiresolution bound must be. The consequence is then that a larger number of continuous functions are allowed in the residual and the reconstruction is then flattened out and the error increases.

6.4 Numerical experiments for the total variation problem

The minimization problem from chapter 2.3 with the total variation norm (2.11) as the regularization term is defined as

$$\min_{u \in \mathbb{R}^\Gamma, v \in \mathbb{R}^I} \|\nabla u\|_1 + i_{\mathcal{C}}(v) + i_{\mathcal{F}}(u, v).$$

To solve the problem, we have implemented the Douglas-Rachford algorithm and Chambolle's projection algorithm, as presented in chapter 4.3, in Matlab. The implementations can be seen in appendix C.3. As test function, we will use the image shown in figure 6.1a, which is piecewise continuous and contains discontinuities.

For the implementation of the Douglas-Rachford algorithm the stopping criterion is chosen heuristically by the same method as for the quadratic regularization problem in two dimensions. The tolerance for the stopping criteria are shown in table 6.3. The stopping criteria for the implementation of Chambolle's projection algorithm is chosen to be such that the algorithm stops when

$$\|w - w_{old}\|_2 \leq \text{tol}_C,$$

where tol_C is the tolerance for Chambolle's projection algorithm. This value is also chosen heuristically, and is

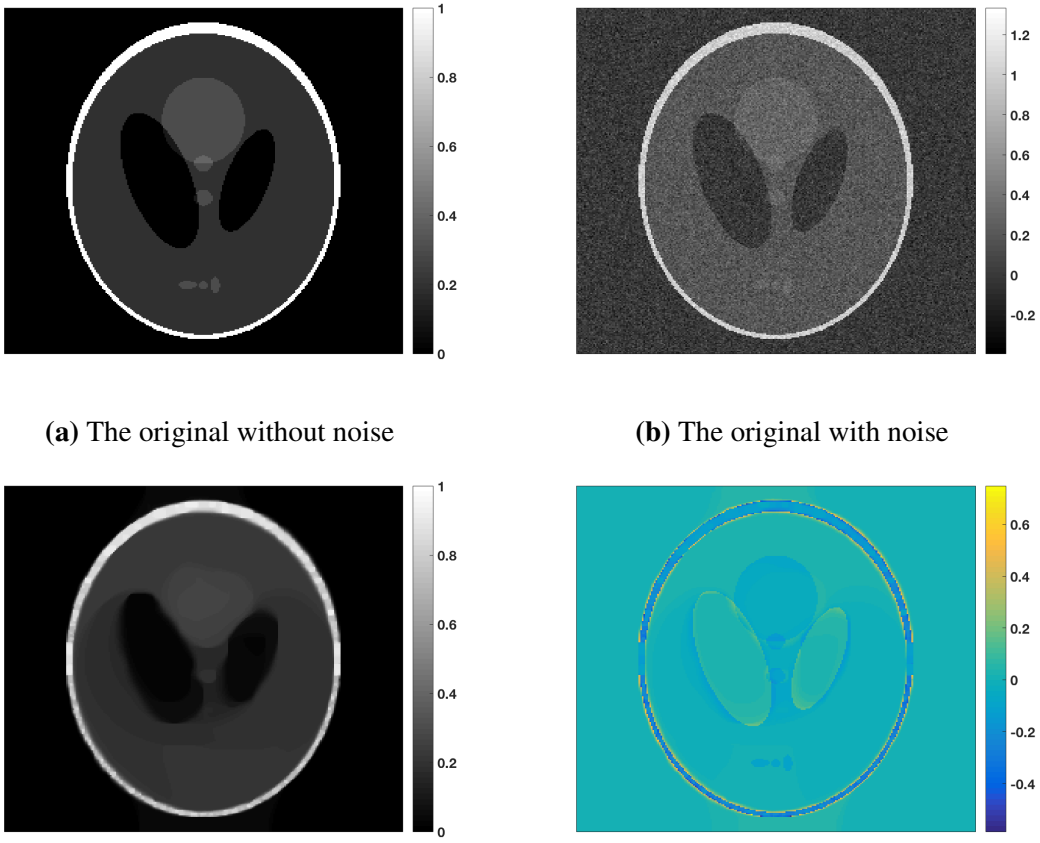
$$\text{tol}_C = NM10^{-10}.$$

For all the experiments, we will use $\mu = 1$.

The result of running the algorithm on this test function, where $N = M = 256$ and the variance of the noise $\sigma = 0.1$, is shown in figure 6.8.

Table 6.3: The table shows the step length when Douglas-Rachford algorithm for the total variation problem will stop.

N	128	256	512
tol	$e^{-2.8}$	$e^{-2.5}$	$e^{-1.9}$



(a) The original without noise

(b) The original with noise

(c) The reconstruction

(d) Residual

Figure 6.8: In this figure the original image with and without noise, the result after denoising and the residual are plotted. The grid size is $N = 256$, the noise has variance $\sigma = 0.1$ and the multiresolution bound $\gamma = 4.85$. Note that the colorbars for the original without noise and the reconstruction are scaled between 0 and 1 and the error is between -0.6 and 0.8.

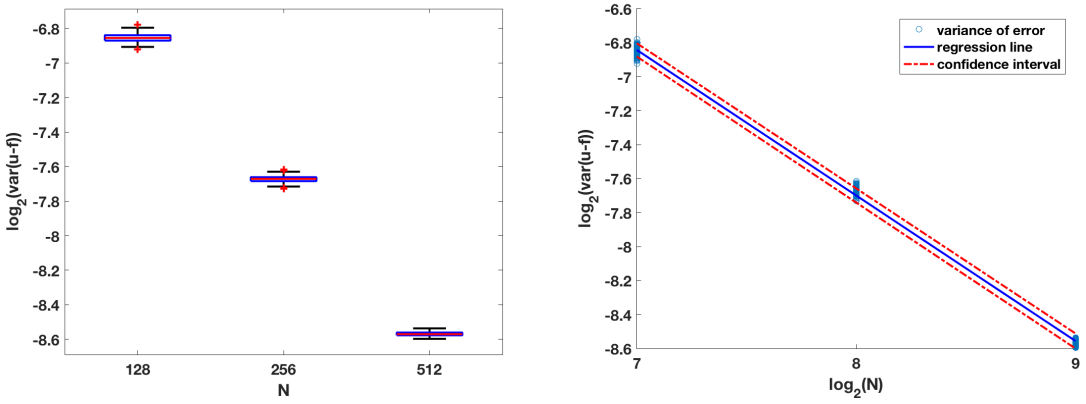
In subfigure 6.8a and subfigure 6.8b, we can see the test function without and with noise, respectively. In subfigure 6.8c, the reconstruction is shown and in subfigure 6.8d is the residual shown. As we can see, the noise makes a lot of changes in the original image, but the result after the denoising is very close to the original function. In the plot of the residual, we can see that the error in one pixel lies between -0.6 and 0.8. This error is located near the discontinuities, such as by the large white circle. We can also see that some of the small circles have disappeared, which is expected since the total variation norm either detects them or not. So it seems that the algorithm behaves at it shall, but we must run some more tests to be sure. To test the algorithm, we have performed some statistical tests of the error and how the error behaves with different variances. We will also study where in the image the constrains affect the reconstruction, and which subsets in the index set that affect the results.

6.4.1 Error

We will, as for the quadratic regularization problem in two dimensions, study the variance of the error and the connection between the error and the multiresolution norm of the noise.

We start with the variance of the error for the grid sizes $N = 128$, $N = 256$ and $N = 512$ with the variance of the noise $\sigma = 0.1$ and the multiresolution bound for $\alpha = 0.90$. To find the variance of the error, we will follow the method used for the quadratic regularization problem in two dimensions. For each number of data points, 200 experiments are performed, and the logarithm of base 2 of the variances of the errors is plotted against the number of grid points in each direction in a box plot and a scatter plot. The variance of the error is calculated as in (6.3). Figure 6.4 shows the results. Subplot 6.9a shows the box plot of the logarithm of base 2 of the variance of the error for $N = 128$, $N = 256$ and $N = 512$ grid points in each direction. From the box plot, we can see that the error is quite consistent and that the error decreases when the number of grid points increases.

Subplot 6.9b shows the scatter plot of the logarithm of base 2 of the error against the number of grid points in each direction. We have also performed a linear regression and the regression line ant its confidence interval are plotted as a lines in the scatter plot. As we can see, the logarithm of base 2 of the error is dependent of the logarithm of base 2 of the number of grid points in each direction. slope of the regression line is -0.86 and its confidence interval at level 0.95 is $[-0.90, -0.82]$. The rate of the method is then $CN^{-0.86}$ and there is a polynomial dependence. To the best of my knowledge, there is no known theoretical rate for the total variation method. If we compare this



(a) Box plot **(b)** Scatter plot with regression line

Figure 6.9: The subplots shows the variance of the error for the total variation problem. The logarithm of base 2 of the variance of the error is plotted against the side lengths in a box plot and in a scatter plot. Each box in the box plot shows variance of the error for $N = 128$ $N = 256$ or $N = 512$ number of data points in each direction. In the scatter plot is also a regression line and its confidence interval plotted. The regression line has slope -0.86 and its confidence interval at level 0.95 is $[-0.90, -0.82]$. The simulations are solved with $\sigma = 0.1$ and the multiresolution bound for $\alpha = 0.90$. From the plot, we can see that the variance of the error is dependent of the number of grid points.

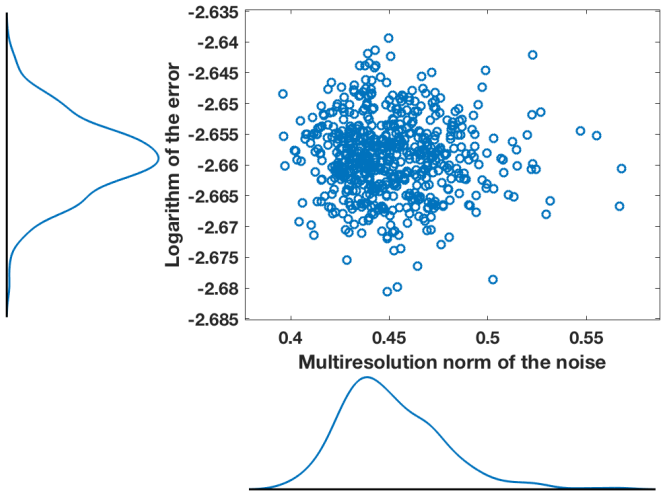


Figure 6.10: Scatter plot of the logarithm of the error and $\|\varepsilon\|_B$ for the quadratic regularization problem. Together with the scatter plot the marginals are plotted. The plot shows that the error is consistent and independent of $\|\varepsilon\|_B$.

result with the result for the quadratic regularization problem in two dimensions, we observe that the rate for the quadratic regularization problem is larger than for the total variation problem. This is as expected, because the rate decreases for less smooth functions, and the functions the total variation problem solves is not very smooth.

Now, we will study the the connection between the error and the multiresolution norm of the noise, by plotting the error of the reconstruction against the multiresolution norm of the noise in a scatter plot, which is plotted in figure 6.10. For this experiment the variance of the noise is $\sigma = 0.1$ and the error is calculated with (6.4). Together with the scatter plot the marginals are plotted. From the marginals, we can see that the logarithm of the error is distributed around -2.66. The multiresolution norm of the noise is distributed between 0.4 and 0.5. We can see that the error is consistent and that there is no dependence between the error and $\|\varepsilon\|_B$.

We can then conclude that there is a dependence between the error and the number of grid points and that the rate of the dependence is less that for the quadratic regularization problem in two dimensions. On the other hand, the error is independent of the multiresolution norm of the noise.

6.4.2 Variance

We will now study how the variance of the noise affects the result. To study this, we will denoise the test image phantom with the noise scaled with different variances.

How the size of the variance is affecting the reconstruction is easiest to see in the images of the reconstructions in figure 6.11. To make it easier to detect the changes, the colorbars are scaled

between 0 and 1. Here, the result of reconstructions of the test image phantom when the noise has the variance $\sigma = 0.01$, $\sigma = 0.05$, $\sigma = 0.1$, $\sigma = 0.5$ and $\sigma = 1$ are plotted. For the variance $\sigma = 0.01$, the reconstruction is very close to the original. On the other hand, for the variance $\sigma = 1$, the details are not reconstructed, so the reconstruction is not similar to the original image at all. For the reconstruction in between these two, the reconstruction gradually is getting worse. When $\sigma = 0.01$ all the details are reconstructed. When the variance is increased to $\sigma = 0.05$ the three small gray circles down at the middle are not reconstructed. When the variance increased to $\sigma = 0.5$, more or less no details are detectable and only some parts of the white circle are reconstructed.

The reason for these details to disappear for larger variances is, as discussed earlier, the size of the multiresolution bound. The details can be explained as small continuous functions and the smaller the detail is, the smaller the continuous function is. As we can see in figure 6.11, the smallest details are those that disappear first and the largest details disappear last. This is as expected because the consequence of increasing the multiresolution bound is, as discussed earlier, that more continuous functions are interpreted as noise and hence removed. For instance, when the variance is increased from $\sigma = 0.01$ to $\sigma = 0.05$ the multiresolution bound is also increased, since we have defined it as $\gamma = 4.85\sigma$. As we can see, the three small details down in the middle are not reconstructed for $\sigma = 0.05$.

We will now study how the variance of error behaves when the variance of the noise is varied between 0.01 and 1.0. The method is the same as for the quadratic regularization problem in two dimensions and the result for 100 experiments is plotted as a scatter plot in figure 6.12. In the scatter plot, the logarithm of the error is plotted against the logarithm of the variances. The regression line and its confidence interval at level 0.95 are also plotted in figure 6.12. The slope of the regression line is 1.07 and its confidence interval at level 0.95 is $[1.0713, 1.0743]$. For $-2.5 \leq \log(\sigma) \leq 0$, we have that the data points fits to the regression line. For $\log(\sigma) \leq -3$ the error is almost constant and the data points do not fit the regression line. The reason for the error to be constant at $\log(\sigma) \leq -3$ might be the reconstruction of the small details. In figure 6.11, where the reconstructions with different variances are plotted, we can see that from $\log(\sigma) = -3$ to $\log(\sigma) = -4.6$ the small details change from being not reconstructed at all to well reconstructed. In the reconstructions in between, the details are blurred and merged together. The effect is stronger for a variance closer to $\log(\sigma) = -3$. For $\log(\sigma) > -3$ none of the small details are reconstructed and therefore only the large details are having an impact in the change in the error. Here, we can see that the error of the reconstruction is polynomially dependent of the variance. This is because, as discussed earlier, we have defined the multiresolution bound such that it is scaled with σ and for larger multiresolution bound, the more continuous functions are interpreted as noise.

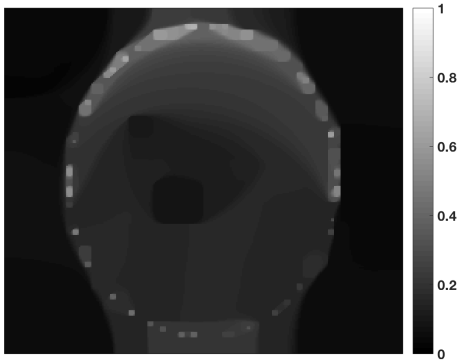
We can conclude that the variance is dependent of the variance. As discussed, the multiresolution bound is increased when the variance is increased, the consequence of that is that small continuous functions are removed as noise, as we have shown in figure 6.11.

(a) The original phantom

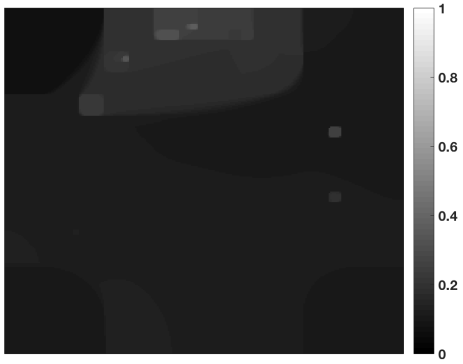
(b) The reconstruction when $\sigma = 0.01$ and $\log(\sigma) = -4.6$.

(c) The reconstruction when $\sigma = 0.02$ and $\log(\sigma) = -3.9$.

(d) The reconstruction when $\sigma = 0.05$ and $\log(\sigma) = -3.0$.



(e) The reconstruction when $\sigma = 0.5$ and $\log(\sigma) = -0.7$.



(f) The reconstruction when $\sigma = 1$ and $\log(\sigma) = 0$.

Figure 6.11: The figure shows reconstructions of the test function with grid size $N = 256$ for different variances and the original function as a reference. As we can see the reconstructions is flatten more and more out and more and more details are missing, the larger the variance is. The reconstructions are performed with the multiresolution bound $\gamma = 4.85\sigma$. The colorbars are scaled between 0 and 1 for all of the images.

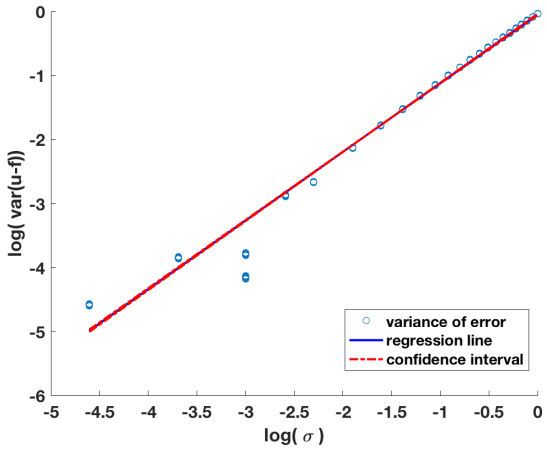


Figure 6.12: The figure shows a scatter plot of the logarithm of the error and the logarithm of variance. We have performed linear regression an the data set and the regression line and its confidence interval at level 0.95 are also plotted. The slope of the regression line is 1.07 and its confidence interval at level 0.95 is $[1.0713, 1.0743]$. The reconstruction is performed for a test function with grid size $N = 256$ and the multiresolution bound $\gamma = 4.85\sigma$.

6.4.3 Active sets

In this section, we will study which quadrants in the dyadic index sets that is dominant in the result. First, recall that for a grid with $N = 2^m$ grid points in each direction, the dyadic index set is defined as a union

$$I = \bigcup_{s=0}^{l} I_s$$

of the index sets

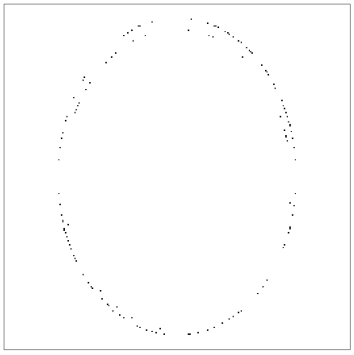
$$I_s = \{(k, i, j) : k = 2^s, i = p2^s + 1, j = q2^s + 1 \text{ with } p, q = 0, \ldots, 2^{m-s} - 1, s = 0, 1, \ldots, m\},$$

where s is referred to as the level of the index set. Then, a quadrant is one of the squares the multiresolution norm sums over.

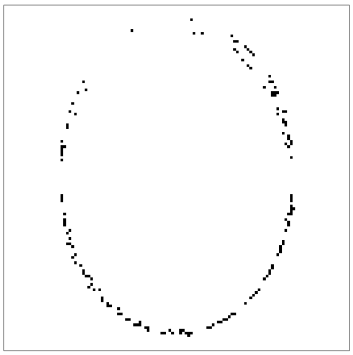
To study this, we use test image phantom with the variance of the noise was $\sigma = 0.1$. We start with reconstructing the image with the multiresolution bound $\gamma = 4.85\sigma$, then we find the value of each quadrant of the multiresolution norm. To find the value of the quadrants in a level s in the index set I, we use the sums in the multiresolution. The quadrants that are active, and hence dominant, are those that lie close to the boundary of the constraint. Therefore, we are interested in the quadrants where the values are close to γ. In figure 6.13, we have detected which quadrants that have value very close to γ; These are the squares that are black. The original image of phantom is plotted in figure 6.13 as a reference.

If we compare the plots of the quadrants in figure 6.13 and the original image in subfigure 6.13a, we see that where there are black squares in the quadrants, there are discontinuities in the original

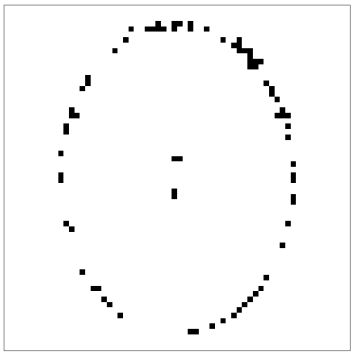
(a) The original phantom



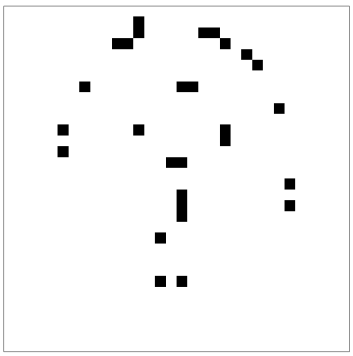
(b) $s = 0$



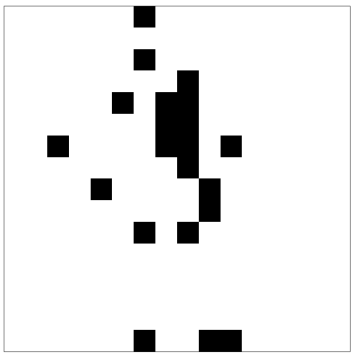
(c) $s = 1$



(d) $s = 2$



(e) $s = 3$



(f) $s = 4$

Figure 6.13: The plots shows the quadrants for each level s on the dyadic index set. In the plots, the quadrants that lies on the boundary of the constraint is marked as black. These quadrants lies where the function has discontinuities. The original image phantom is plotted as a reference.

image. The reason for the quadrants to lie on the discontinuities is that at the discontinuities the minimal of the total variation norm does not satisfy the constraint. Then the constraint control the value of u such that the constraint is satisfied, and when the constraint is satisfied is the value of the quadrant equal to γ. The areas where the function is continuous are the gradients zero, hence the value of the quadrants are zero. We then have that the quadrants that are dominant for the solution are those that lies on discontinuities.

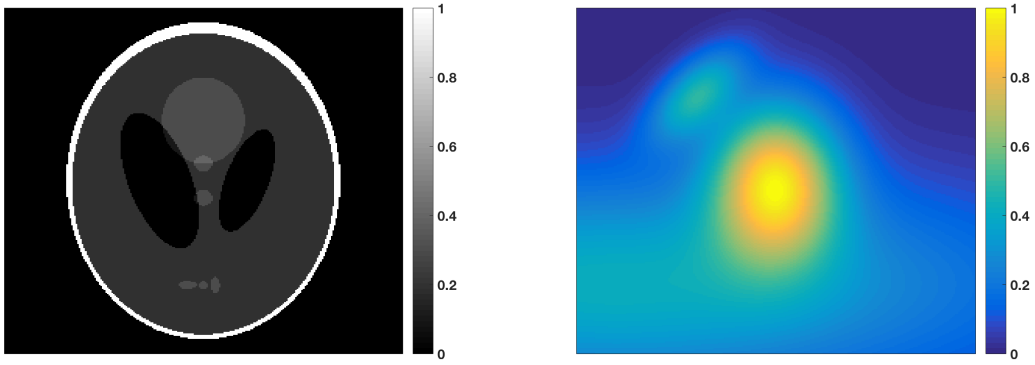
From this we can think that there is at these areas, where the quadrants lies at the boundary of the constraints, that is the most important areas. Most of the areas where the quadrants do not lie on the boundary of the constraint is the function constant. A possible improvement of the solution method could then be to first find the discontinuities in the image, then reconstruct these areas and the areas without discontinuities separately.

6.5 Comparison of the quadratic regularization problem and the total variation problem

As mentioned in chapter 2.2, the quadratic regularization problem and the total variation problem manages different type of functions. The quadratic regularization problem manages smooth functions well, but theoretically it does not manage discontinuities. On the other hand, the total variation problem manages discontinuities. As we have seen in the two previous sections, for both methods the error is dependent on the grid size and the variance of the noise, but independent of $\|\varepsilon\|_B$.

We will now study the differences in the methods. Until now, both problems has been tested on functions they are expected to manage. To study the differences in the methods, we will compare how they reconstruct the Gaussian test function and the test image phantom, both are plotted as reference in figure 6.14. For all the experiments in this section the number of data points is 256 in both directions, the variance is $\sigma = 0.1$ and the multiresolution bound $\gamma = 4.85\sigma$.

We start with comparing how the methods manage to denoise the test image phantom. The function that forms this image contains discontinuities and between the discontinuities the function is constant. The results are plotted in figure 6.15, subfigure 6.15a and subfigure 6.15b show the reconstructions, subfigure 6.15c and subfigure 6.15d show the residuals, and subfigure 6.15e and subfigure 6.15f show a detail of the reconstructions. At first sight, we see that the total variation problem reconstructs the phantom much better than the quadratic regularization problem. If we look at subfigure 6.15e and subfigure 6.15f, where only a detail of the images is shown, we can see that the quadratic regularization problem is smoothing out the discontinuities, while the total variation problem shows the edges clear. The quadratic regularization problem smooths out the edges mainly because, as explained in chapter 2.2, it punishes large gradients, so it is cheaper to have a long distance where the gradients are small, while the total variation problem is exactly opposite, hence it reconstruct discontinuities. Even though the quadratic regularization problem minimizes the gradient, the gradient will almost never go to zero, which means that the reconstruction will almost never be a constant function. If most of the noise at some area is larger than zero,



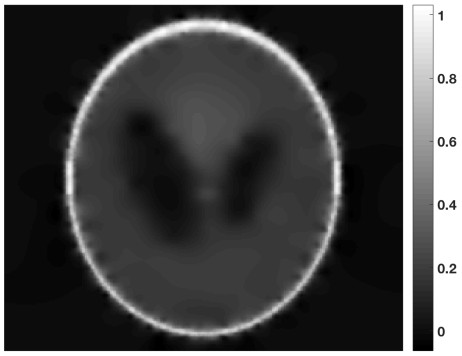
(a) Phantom **(b)** The Gaussian test function

Figure 6.14: The test images.

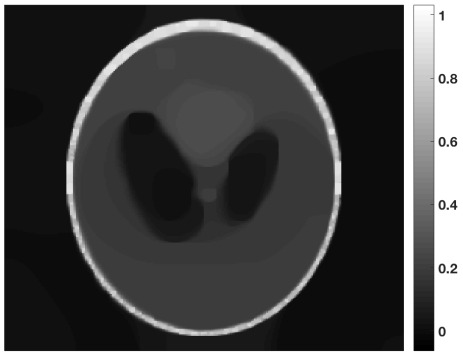
the quadratic regularization problem will interpret this as that the original function is larger here. As a consequence, in some areas in the image that are constant, the reconstruction can have some oscillations.

We have now seen how the two methods manage discontinuities. Next we will study is how they manage smooth functions. To study this, we have used the two dimensional Gaussian test function. The results are plotted in figure 6.16, subfigure 6.16a and subfigure 6.16b show the reconstructions, subfigure 6.16c and subfigure 6.16d show the residuals, and subfigure 6.16e and subfigure 6.16f show a cross section in one dimension of the reconstructions and the original function. At first, it looks like they both manage the Gaussian test function pretty well, both have the maxima and seem to have the same the shape as the Gaussian test function. However, if we study the cross sections, we can see that the reconstruction with total variation is not as smooth as the reconstruction with quadratic regularization. The reason that the reconstruction with the total variation problem is not that smooth is, as explained in chapter 2.2, that the square shape of the isosurface of the L_1-norm has the effect that the gradient often is minimized to zero. A gradient with value zero gives a constant function, hence the reconstruction with the total variation gives often a constant function. For this test function, it looks like the total variation makes the gradient equal to zero where the original gradient is small and interpret large gradients as discontinuities. On the other hand, the quadratic regularization makes a smooth reconstruction, and follows the original function pretty good.

To sum up, when we choose to use the quadratic regularization problem, we also assume the function to be smooth, therefore this method works best on smooth functions. The total variation problem uses the L_1- norm that favors discontinuities, which makes this method great to reconstruct functions with discontinuities. A down side with the total variation problem is that it minimizes the gradient where there are no discontinuities to zero, hence the reconstruction consists of piecewise constant parts.



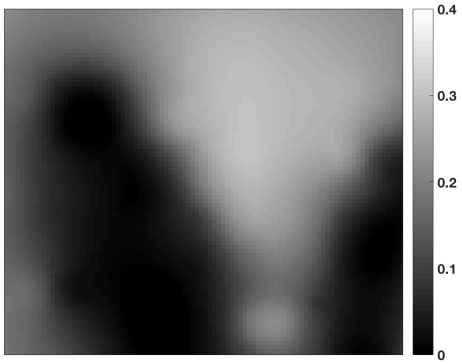
(a) Reconstruction with the quadratic regularization method.



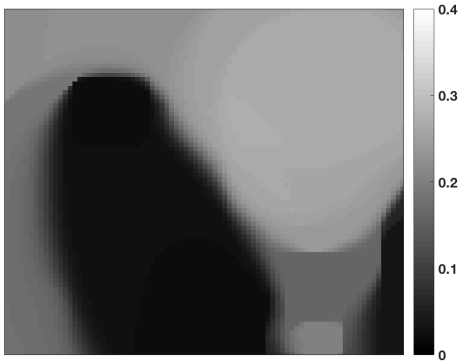
(b) Reconstruction with the total variation method.

(c) The residual of the reconstruction with the quadratic regularization method.

(d) The residual of the reconstruction with the total variation method.

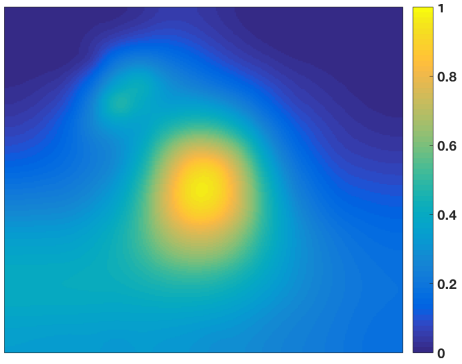


(e) A segment of the reconstruction with the total variation method.

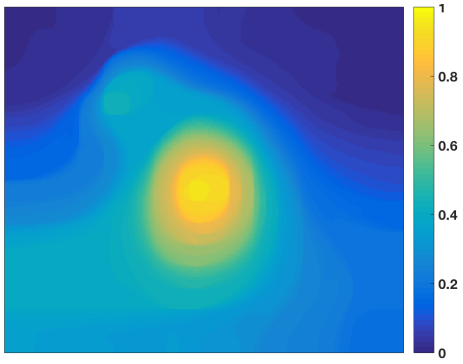


(f) A segment of the reconstruction with the quadratic regularization method.

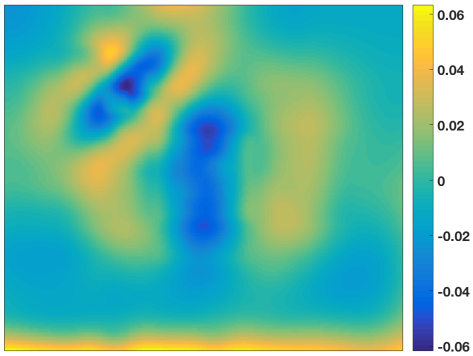
Figure 6.15: The figure shows the reconstructions, the residuals and a detail of the reconstructions of the test image phantom reconstructed with the quadratic regularization method and the quadratic regularization method. The colorbar for the reconstructions are scaled between 0 and 1. We see that for the reconstruction with the quadratic regularization method is the discontinuities blurred, while with the total variation method, the discontinuities are shown clearly.



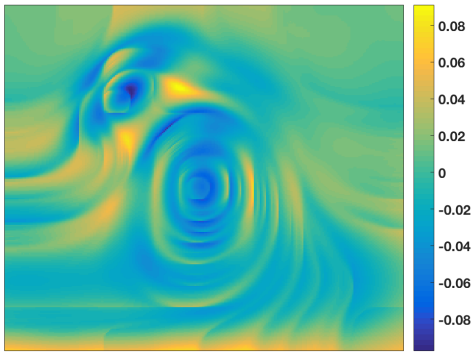
(a) Reconstruction with the quadratic regularization method.



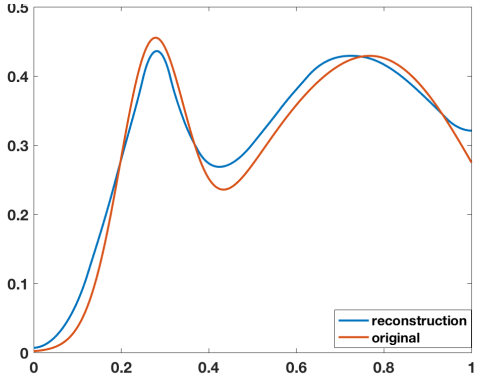
(b) Reconstruction with the total variation method.



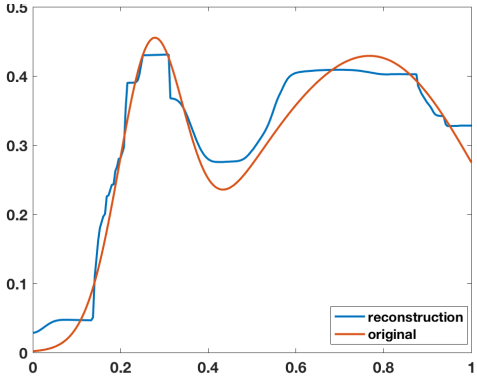
(c) The residual of the reconstruction with the quadratic regularization method.



(d) The residual of the reconstruction with the total variation method.



(e) A cross section of the reconstruction the quadratic regularization method and the test function



(f) A cross section of the reconstruction the total variation method and theGagussian test function

Figure 6.16: The figure shows the reconstructions, the residuals and a detail of the reconstructions of the Gaussian test function reconstructed with the quadratic regularization method and the quadratic regularization method. The colorbar for the reconstructions are scaled between 0 and 1. We see that for the reconstruction with the quadratic regularization method is smooth, while with the total variation method, the reconstruction is not smooth.

6.6 Natural images

We will now test the methods on some natural images. We will use the well-known images Lena and Barbara as test images. They both contain small oscillations, discontinuities and smooth functions. The test image Lena consists of large details with large areas where nothing changes. The test image Barbara consists of many small details as for example the books in the bookcase, the patterns on the cloth and the trousers.

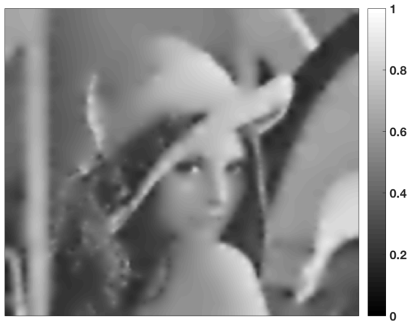
We can see that the reconstructions of both test images with the quadratic regularization method are blurred. All main details, like person, mirror, hat and feather in the reconstruction of Lena, shown in subfigure 6.17c, and person, table and bookcase in the reconstruction of Barbara, shown in subfigure 6.18c, are visible, but blurred. None of the small details or patterns are visible. These can be interpreted as noise and therefore not reconstructed. So, all places where there are discontinuities or patters are blurred out.

For the total variation method, we see that none of the images are blurred, but the reconstructions are cartoon-like. If we first look at Lena, which is shown in subfigure 6.17d, all larger details such as the frame of the mirror are clear, but smaller details such as the feathers have gotten a cartoon-like structure, the same with her face. The reconstruction of the image Barbara, shown in subfigure 6.18d, has an even more cartoon-like structure. No patterns are visible, the areas where the original has patterns is now a constant function. A reason for why the patterns are not reconstructed can be that they are very small oscillating functions that are interpreted as noise by the multiresolution norm. That all edges are clear and the patterns are not reconstructed gives the reconstruction a cartoon-like structure.

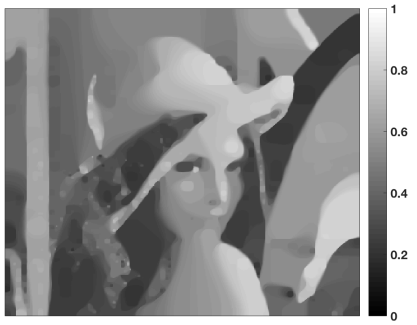
As we have seen, none of the two methods in two dimensions are very good at reconstruct natural images. The quadratic regularization method blurs out the images and the total variation method gives a cartoon-like structure on the image. What both methods manage is to give a reconstruction that shows where the large details are and what they are, but important information such as patterns are not reconstructed.

(a) Lena.

(b) Lena with noise with variance $\sigma = 0.1$.

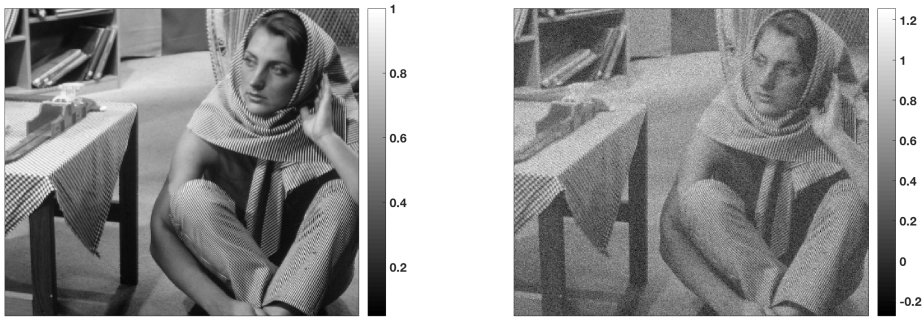


(c) Reconstruction by quadratic regularization.



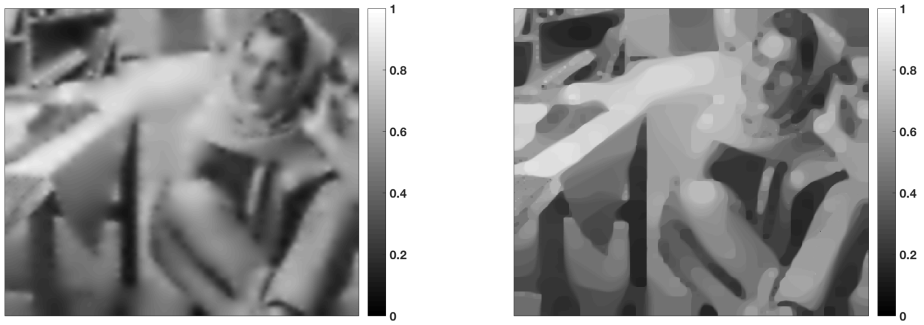
(d) Reconstruction by total variation

Figure 6.17: The test image Lena with and without noise and the reconstructions. By comparing the reconstructions with the original image, we can see that in both reconstructions, most of the small details and patterns are not reconstructed. We can also see that the reconstruction with by the quadratic recularization problem is blurred and the reconstruction with the total variation problem, some areas has gotten cartoon-like structure. Note that the colorbar for original image and the reconstructions are scaled between 0 and 1.



(a) Barbara

(b) Barbara with noise with variance $\sigma = 0.1$.



(c) Reconstruction by quadratic regularization.

(d) Reconstruction by total variation

Figure 6.18: The test image Barbara with and without noise and the reconstructions. By comparing the reconstructions with the original image, we can see that in both reconstructions, most of the small details and patterns are not reconstructed. We can also see that the reconstruction with by the quadratic recularization problem is blurred and the reconstruction with the total variation problem has cartoon-like structure. Note that the colorbar for original image and the reconstructions are scaled between 0 and 1.

Chapter 7

Summary and further development

We started with formulating our problem. We assumed the data points to be of the form

$$Y = f + \varepsilon,$$

where f is the continuous function we want to reconstruct and ε is i.i.d. Gaussian noise. To remove the noise we formulated the constrained optimization problem. As constraint for the optimization problem, we introduced and discussed the multiresolution norm. We then obtained the optimization problem

$$\min_u J(u) \quad \text{such that} \quad \|u - Y\|_B \leq \gamma,$$

where J is the regularization term. For the regularization term J, we have discussed two different functionals, the quadratic regularization term

$$J(u) = \frac{1}{2}\|\nabla u\|_2^2$$

and the total variation norm

$$J(u) = \|\nabla u\|_1.$$

We then developed and implemented in Matlab an algorithm for solving the problem for the quadratic regularization term in one dimension. Here we used ADMM and Dykstra's projection algorithm. Next, we developed and implemented in Matlab algorithms to solve the quadratic problem in two dimensions and the total variation problem in two dimensions. For the quadratic problem, we used the Douglas-Rachford algorithm and for the total variation problem we used the Douglas-Rachford algorithm and Chambolle's projection algorithm. In the last part of the thesis, we tested the method and implementation.

For the experiments in one dimensions, we compared the two different index sets $\hat{I}$ and I, where $\hat{I}$ consists of all the consecutive subsets and I is the dyadic index sets $\hat{I}$. We compared both the error and the numerical efficiency of the algorithm. We saw that the difference in the error for the two index sets is small. We also saw that the numerical efficiency is much better with I than with $\hat{I}$. In

the numerical tests in one dimension we also studied how the variance σ and the multiresolution bound γ affect the reconstruction. We saw that for large variances the reconstruction loses energy and the amplitudes get smaller. For the multiresolution bound we saw that for too small γ the reconstruction is overfitted and for too large γ the reconstruction loses energy and the amplitudes get smaller. Further, the method has been tested on test functions that not always are continuous and differentiable and we have seen that to some extent we can solve these with this method.

For the quadratic regularization problem in two dimensions, we have studied the error and the variance. We have obtained that there is a dependence between the variance of the error and the number of grid points and that the variance of the error decreases with the rate $CN^{-1,4}$ for increasing number of grid points, hence the dependence is polynomial. The error is independent of the multiresolution norm of the noise, but it is dependent on the variance of the noise. For larger variance the error increases because the multiresolution norm increases, hence the multiresolution norm removes more of the small continuous functions in f as noise and the reconstruction u become more and more like a constant function.

The last method we tested was the total variation problem in two dimensions. Here, we achieved similar results as for the the quadratic regularization problem in two dimensions. The rate of the dependence between the variance of the error and the number of grid points in each direction is $CN^{-0.86}$, which is smaller than the rate for the quadratic regularization problem. For this problem we also studied which quadrants in the dyadic index set affect the reconstruction. The quadrants that affected the reconstruction most are those that lie at discontinuities. This is because the total variation norm minimizes the gradient of u such that the gradient is almost zero everywhere, except at the discontinuities and here u is such that the constraint is not satisfied. So at the discontinuities, the multiresolution norm controls the reconstruction.

At the end, both methods in two dimensions were tested on some natural test images. None of the methods are constructed to solve them, but both methods were able to reconstruct some of the larger details. No patterns and small details were reconstructed.

Further development

There are some things that would be interesting to study further. One thing that would have been necessary is to develop a better method to decide when to stop the algorithms. We know that the regularization term J, the constraint and the error are almost constant and the step lengths do not change much after some iterations. Therefore, it could be a good idea to stop the iterations when this occurs. Since the original image is not available, we cannot use the error, but a possible method is to use the regularization term J and the step length. In order to find where J is almost constant we can estimate its first derivative and when the derivative is around zero we might want to stop the iteration. For the step length, we can find where the rate of the change is smallest by using the second derivative and stop when the second derivative is small. As stopping criteria we could then use that the iterations should stop when the first derivative of J is around zero and the second derivative of the step length small.

In chapter 6.4.3, we discussed where the constraint in the total variation problem affects the reconstruction. We concluded there that the constraint affects the results only at the discontinuities. Therefore, it would been interesting to study the possibility to solve the continuous areas and the discontinuous areas separately for the total variation problem. To solve the problem with this decomposition could possibly increase the numerical efficiency and reduce the run time of the implementation.

For the quadratic regularization problem in two dimensions, the structure of the proximity operators is such that the algorithm might be implemented in parallel. The proximity operator for h consists of only projections which are all independent of each other. The proximity operator for g uses the matrix A, which is defined as

$$A = \begin{bmatrix} (Id - \mu_g \Delta) & 0 & -F^T \\ 0 & Id_\zeta & Id_\lambda \\ -F & Id_\zeta & 0 \end{bmatrix}.$$

Both $Id - \mu_g \Delta$ and F have such structures that A can be solved in parallel with shared memory. It could be interesting to investigate if this parallelization might improve the run time of the implementation.

Appendix A

Karush-Kuhn-Tucker conditions

Consider the convex program

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{such that} \quad x \in C, \tag{A.1}$$

where f is a convex function and $C \subset \mathbb{R}^n$, $C \neq \emptyset$ is defined by the conditions

$$\begin{aligned} c_i(x) &\geq 0, \quad i \in \mathcal{I}, \\ Ax - b &= 0, \end{aligned}$$

where $\mathcal{I}$ is the set of inequality constrains, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and the concave functions $c_i : \mathbb{R}^n \to \mathbb{R}$. For this program, the Lagrangian $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^I \to \mathbb{R} \cup \{+\infty\}$ is

$$\mathcal{L}(x; \lambda, \mu) = f(x) - \lambda^T (Ax - b) - \sum_{i \in \mathcal{I}} \mu_i c_i(x).$$

For a solution of (A.1) to be optimal is it necessary that the solution satisfies the KKT conditions. In general the KKT conditions are not sufficient to characterize an optimal solution except if the program is convex, in which case the KKT conditions are sufficient. If the problem also satisfies Slater's conditions, see definition 4, then the KKT conditions are necessarily. [15]

Definition 2. *The Karush-Kuhn-Tucker (KKT) conditions for* (A.1) *are defined as*

$$\nabla_x \mathcal{L}(x^*; \lambda^*, \mu^*) = \nabla f(x^*) - A^T \lambda - \sum_{i \in \mathcal{I}} \mu_i^* \nabla c_i(x^*) = 0, \tag{A.2}$$

$$c_i(x^*) \geq 0 \quad \textit{for all} \quad i \in \mathcal{I}, \tag{A.3}$$

$$Ax^* - b = 0, \tag{A.4}$$

$$\mu_i^* \geq 0 \quad \textit{for all} \quad i \in \mathcal{I}, \tag{A.5}$$

$$\mu_i^* c_i(x^*) = 0 \quad \textit{for all} \quad i \in \mathcal{I}. \tag{A.6}$$

Another characteristic for an optimal solution of (A.1) is that the optimal solution is a saddle point of the Lagrangian $\mathcal{L}$.

Definition 3. *A point (x^*, λ^*, μ^*) with $\mu^* \geq 0$ is a saddle point of the Lagrangian $\mathcal{L}$, if*

$$\mathcal{L}(x^*; \lambda, \mu) \leq \mathcal{L}(x^*; \lambda^*, \mu^*) \leq \mathcal{L}(x; \lambda^*, \mu^*) \quad \textit{for all} \quad (x, \lambda, \mu), \mu \geq 0. \tag{A.7}$$

The following theorem give us a connection between the KKT conditions and saddle points.

Theorem 1. *Let $x^* \in \mathbb{R}^n$ and μ^*, λ^* and $\mu^* \geq 0$, then the point (x^*, λ^*, μ^*) is a saddle point if and only if it satisfies the KKT conditions.*

Proof. Suppose (x^*, λ^*, μ^*) is a saddle point, then (A.7) holds. First, we use the first inequality in (A.7), then we have

$$\mathcal{L}(x^*; \lambda, \mu^*) \leq \mathcal{L}(x^*; \lambda^*, \mu^*) \quad \text{for all} \quad \lambda \tag{A.8}$$

and

$$\mathcal{L}(x^*; \lambda^*, \mu) \leq \mathcal{L}(x^*; \lambda^*, \mu^*) \quad \text{for all} \quad \mu \geq 0. \tag{A.9}$$

We use (A.8) and have that

$$\mathcal{L}(x^*, \lambda^*, \mu) - \mathcal{L}(x^*, \lambda, \mu) = (\lambda - \lambda^*)^T (Ax^* - b) \geq 0 \quad \text{for all} \quad \lambda,$$

which gives

$$Ax^* - b = 0$$

which is the same as (A.4) from the KKT conditions.

Now, we use (A.9) and have that

$$\mathcal{L}(x^*; \lambda^*, \mu^*) - \mathcal{L}(x^*; \lambda^*, \mu) = -\sum_{i \in \mathcal{I}} (\mu_i^* - \mu_i) c_i(x^*) \geq 0.$$

We choose $i \in \mathcal{I}$ and set $\mu_j = \mu_j^*$ for all $j \neq i$, which gives

$$-(\mu_i^* - \mu_i) c_i(x^*) \geq 0 \quad \text{for all} \quad \mu_i \geq 0.$$

Then if $\mu_i^* = 0$,

$$\mu_i c_i(x^*) \geq 0 \quad \text{for all} \quad \mu_i \geq 0$$

and thus

$$c_i(x^*) \geq 0.$$

Further, if $\mu_i^* > 0$ and we choose $\mu_i = 0$, then

$$-\mu_i^* c_i(x^*) \geq 0$$

and thus

$$c_i(x^*) \leq 0.$$

If we rather choose $\mu_i = 2\mu_i^*$, then

$$\mu_i^* c_i(x^*) \geq 0$$

and

$$c_i(x^*) \geq 0.$$

Since $c_i(x^*)$ does not depend on μ_i, it follows that $c_i(x^*) = 0$ whenever $\mu^* > 0$. To summarize, we either have that $\mu^* = 0$ and $c_i \geq 0$ or that $\mu^* > 0$ and $c_i = 0$. In both situations

$$\mu_i^* c_i(x^*) = 0,$$

which is the same as (A.6) from the KKT conditions.

Now, we use the second inequality in definition 3 so that

$$\mathcal{L}(x^*; \lambda^*, \mu^*) \leq \mathcal{L}(x; \lambda^*, \mu^*) \quad \text{for all} \quad x.$$

This inequality is the same as minimizing the Lagrangian with respect to x and can be written in the form

$$x^* \quad \text{minimizes} \quad x \mapsto \mathcal{L}(x; \lambda^*, \mu^*).$$

Since the Lagrangian is convex, x^* is the mininmum of the Lagrangian if and only if

$$\nabla_x \mathcal{L}(x^*; \lambda^*, \mu^*) = 0.$$

We use this condition and get

$$\nabla f(x) - A^T \lambda^* - \sum_{i \in \mathcal{I}} \mu_i^* \nabla c_i(x) = 0,$$

which is the same as (A.2). Here, we can also go in the other direction, from (A.2) and get the minimum x^*.

Now, suppose the KKT conditions are satisfied such that $c_i(x^*) \geq 0$, $Ax^* = b$, $\mu^* c_i(x^*) = 0$. Then we have that

$$\mathcal{L}(x^*; \lambda^*, \mu^*) - \mathcal{L}(x^*; \lambda, \mu) = (\lambda^* - \lambda)^T (Ax^* - b) - \sum_{i \in \mathcal{I}} (\mu_i^* - \mu_i) c_i(x^*).$$

The term $(\lambda^* - \lambda)^T (Ax^* - b) = 0$ because $Ax^* = b$, and for the term $\sum_{i \in \mathcal{I}} (\mu_i^* - \mu_i) c_i(x^*)$ we have that $\mu_i^* c_i(x^*) = 0$ so $-\mu_i c_i(x^*) \leq 0$. Then we have

$$\mathcal{L}(x^*; \lambda^*, \mu^*) - \mathcal{L}(x^*; \lambda, \mu) \geq 0.$$

□

From theorem 1 we have that a point is a saddle point only if it satisfies the KKT conditions. The next theorem gives us that if we have a saddle point (x^*, λ^*, μ^*) then x^* solves (A.1).

Theorem 2. *Let $x^* \in \mathbb{R}^n$ and μ^*, λ^* and $\mu^* \geq 0$. If the point (x^*, λ^*, μ^*) is a saddle point, then x^* solves* (A.1).

Proof. From the first part of the proof of theorem 1 we obtain that

$$\mathcal{L}(x^*; \lambda, \mu) \leq \mathcal{L}(x^*; \lambda^*, \mu^*)$$

implies that

$$Ax^* = b, \tag{A.10}$$
$$c_i(x^*) \geq 0, \tag{A.11}$$
$$\mu^* c_i(x^*) = 0. \tag{A.12}$$

Now, (A.10) and (A.11) together are equivalent to the assumption $x^* \in C$ and (A.10) and (A.12) give

$$\mathcal{L}(x^*; \lambda^*, \mu^*) = f(x^*) - \lambda^T(Ax^* - b) - \sum_{i \in \mathcal{I}} \mu_i^* c_i(x^*) = f(x^*).$$

If $x \in C$, then

$$\mathcal{L}(x; \lambda^*, \mu^*) = f(x) - \lambda^T(Ax - b) - \sum_{i \in \mathcal{I}} \mu_i^* c_i(x) \leq f(x),$$

since $Ax - b = 0$, $c_i(x) \geq 0$ and $\mu_i^* \geq 0$. We then have that

$$f(x^*) = \mathcal{L}(x^*; \lambda^*, \mu^*) \leq \mathcal{L}(x; \lambda^*, \mu^*) \leq f(x) \quad \text{for all} \quad x \in C$$

so $f(x^*) \leq f(x)$ and therefore x^* solves (A.1). □

Now, we have that if we have a saddle point, then is it a solution for (A.1). We also have that if a point is a saddle point then the KKT conditions are satisfied.

Definition 4. *Slater's condition is satisfied, if there exist some x^* such that*

$$c_i(x) > 0, \quad i \in \mathcal{I}$$

and

$$Ax - b = 0.$$

Theorem 3. *Assume* (A.1) *has a solution and that Slater's condition is satisfied, then there exists a saddle point of the Lagrangian* $\mathcal{L}$*.*

Theorem 3 is a special case of theorem 11.15 in [10, chapter 11] (specially the last part of the theorem) and is proved there. The differences are that Güler [10, chapter 11] uses $C \subseteq \mathbb{R}^n$ while we use $\mathbb{R}^n$ as C, Güler [10, chapter 11] use convex constraints and we use concave constraints and Güler [10, chapter 11] has affine inequalities as one constraint and we have affine equality constraint.

From theorem 3, we have that when (A.1) have a solution and Slater's condition is satisfied, there exists a saddle point. From theorem 1, the KKT conditions are now satisfied.

All our problems are coercive, so where the constraints are satisfied, we do have a solution. So, we have a solution x^* and this point is a saddle point that satisfies the KKT conditions. Therefore is the KKT conditions both necessary and sufficient.

Appendix B

Index sets

In chapter 2.1 the multiresolution norm is formulated. In the two dimensional case, we formulate it with the index set

$$\hat{I} = \{(k, i, j) : k = 1, \ldots, N,\ i = 1, \ldots, N - k + 1,\ j = 1, \ldots, M - k + 1\}$$

and in the one dimensional case, we formulate it with the index set

$$\hat{I} = \{(k, i) : k = 1, \ldots, N,\ i = 1, \ldots, N - k + 1\}.$$

To use all the subsets in $\hat{I}$ will make the computations slow. As discussed at the end of chapter 4.1, for a one dimensional grid the number of constraints is $\frac{N(N-1)}{2} = O(N^2)$ for the index set $\hat{I}$. Therefore, we introduce the dyadic index set, which only uses a few of the sets in $\hat{I}$. By using the dyadic index set, the constraints for a one dimensional grid reduce to $2N - 1 = O(N)$. In the next two sections, the dyadic subset is explained, first for a one dimensional grid, then for a two dimensional grid.

B.1 Dyadic index set for a one dimensional grid

For a one dimensional grid, assume the number of data points is a power of two, for instance $N = 2^m$, where $m \in \mathbb{N}$. We then divide for every $0 \leq s \leq m$ the data points into disjoint sets of size $k = 2^s$. For each level s, the data points are divided into the sets

$$\{1, \ldots, 2^s\}, \quad \{2^s + 1, \ldots 2 \cdot 2^s\}, \ldots, \{(m-1)2^s + 1, \ldots, 2^m\}.$$

Figure B.1 is an example of how a grid with 8 grid points is divided into the dyadic sets. As we can see, the in level $s = 0$ consists of sets of one point. In level $s = 1$ each set consists of 2 grid points, which in the figure are joined with a line. In level $s = 2$ each set consists of 4 grid point

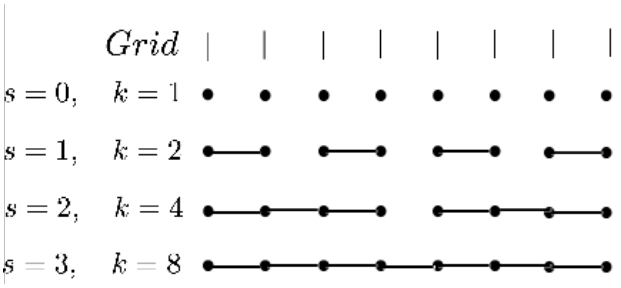


Figure B.1: The figure shows how the dyadic subsets of a grid of size $N = 8$ are constructed. For each level s are there $k = 2^s$ elements in each set. The dots mark the elements in that level, the line connects the elements into sets.

and in level $s = 3$ there is one set that consists of all the points. In the figure, we can see that for each level every point is represented only once.

More generally, the subsets for each level can be described with the index set

$$I_s = \{(k, i) : k = 2^s, i = p2^s + 1 \text{ with } p = 0, \ldots, 2^{m-s} - 1\}.$$

The index set for the dyadic sets is then the union of all the index sets for all the levels, that is,

$$I = \bigcup_{s=0}^{m} I_s.$$

B.2 Dyadic index set for a two dimensional grid

For a two dimensional grid, we follow the same technique as for a one dimensional grid, but instead of dividing into one dimensional sub sets we divide into squares. Assume the number of data points in each direction is a power of two, for instance $M = N = 2^m$, where $m \in \mathbb{N}$. Then, divide for every $0 \leq s \leq m$ the data points into disjoint squares with side length $k = 2^s$. For each level s, the data points (i, j), where $i, j \in \{1, \ldots 2^m\}$, are divided into squares. Each level can be described with the general formulation

$$I_s = \{(k, i, j) : k = 2^s, i = p2^s + 1, j = q2^s + 1 \text{ with } p, q = 0, \ldots, 2^{m-s} - 1,\ s = 0, 1, \ldots, m\}.$$

The whole dyadic index set is then

$$I = \bigcup_{s=0}^{m} I_s$$

Figure B.2 is an example of how a two dimensional grid with $N = M = 8$ grid points in each direction is divided into the dyadic sets. For level $s = 0$ the squares only consists of one grid point. In the figure they are marked with yellow squares. In the level $s = 1$, the side lengths of the squares are two grid points. So the subsets consists of four dots, these subsets are marked in blue. For the

level $s = 2$, the side lengths are four grid points and the subsets is squares with 16 dots. Here, the subsets are marked in red. For the last level, level $s = 3$, the side lengths are eight grid points, this is the whole set and the square is market in green. In figure B.2, we can see that for each level every point is embedded in only one square.

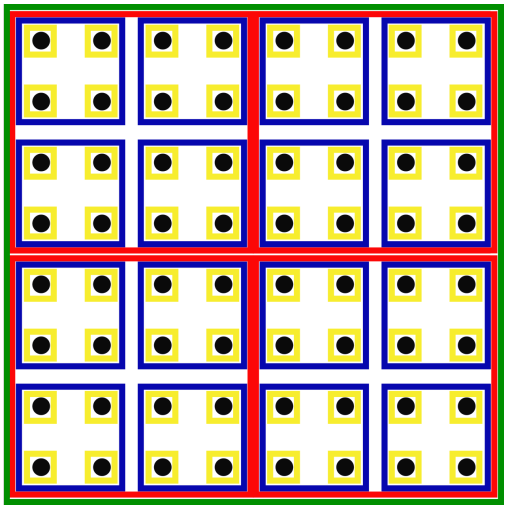


Figure B.2: The figure shows how the dyadic subsets of a two dimensional grid of size $N = M = 8$ are constructed. For each level s, the side lengths of each square is $k = 2^s$. The dots mark the elements in that level and the squares the subsets and the subsets are marked with colored squares, level $s = 0$ is marked with yellow squares, level $s = 1$ with blue, level $s = 2$ with red and level $s = 3$ with green.

Appendix C

Implementations

This chapter presents the implementation used to produce the results in chapter 5 and 6. First, the code for the one dimensional case is presented, then the code for the quadratic regrularization problem in two dimensions and at the end, the implementation of the total variation in two dimensions. All code is written in matlab.

C.1 Quadratic regularization in one dimension

Here are the implementations of ADMM and Dykstra's projection algorithm as they are presented as algorithms in chapter 4.1. First, is ADMM presented, then two implementations of Dykstra's projection algorithm. One version is a general version that project onto all structures of subsets. The second version is specific to the dyadic subsets, and the projections for each level are performed simultaneously. After these three implementations are other functions these three need to run.

```
function u = ADMM(Y, e, eps, gamma, lambda, dykstra)
% This function finds u as a smooth function from the datapoints Y=S_N u +
    niose, where noise is gaussian distributed, reformulated as a optimization
     problem on the form min J(v)+G(u) s.t. Y-u=v with augmented Larange and
    ADMM, and dykstra for projection of v and direct solver for u.
%
% INPUT: Y- datapoints with gaussian noise
%        e - eigenmatrix for C_I if dykstra needs it.
%        eps - stoping criteria for ADMM and dykstra
%        gamma - multiresolution norm
%        lambda - Lagrange multiplier
%        dykstra - function for minimize v (needs input: h, e, gamma, eps)
% OUTPUT: u- the reconstruction of Y

        % The regularization term
        J= @(u) 0.5*norm(forwardDifferences(n)*u, 2)*norm(forwardDifferences(n
            )*u, 2);
```

```
14
15          % Initialize: Generate u0 and v0.
16          n=length(Y);
17          u=zeros(n, 1);
18          v=zeros(n, 1);
19          p=zeros(n, 1);
20          I=speye(n,n);
21          D=laplace(n)
22          r= norm(u+v-Y,2);
23
24          while r>eps
25                  % Find v
26                  [v]=dykstra(Y+lambda*p-u, e, gamma, eps);
27
28                  % Find u
29                  u_old=u;
30                  A=2*lambda*D+I;
31                  b=(Y+lambda*p-v);
32                  u=A\b;
33
34                  % Find p
35                  p=p-(u+v-Y)/lambda;
36
37                  r= max(norm(u+v-Y,2), norm((u-u_old),2));
38          end
39  end
```

```
1  function h=dykstra(h, e, gamma, eps)
2  % This function projects h onto the intersection of the subsets C_I defined by
       the matrix with all the eigenvectors e, by using Dykstras projection
      algorithm. This implementation manages all types of subsets.
3  % INPUT: h- vector
4  %        e- matrix with eigenvectors to the subsets C_I
5  %        gamma- a constant,  multiresolution bound
6  %        eps- stopping criteria
7  % OUTPUT h- vector with h projected onto the intersection of the subsets.
8
9          % Initializations
10     [len M]=size(e);
11     Q=zeros(len,M);
12     Ij=sum(e,1);
13
14     % cycle 1
15     h_old = h;
16     for j=1:M
17          c=dot(h, e(:,j));
18          lambda=c/sqrt(Ij(j));
19          if abs(c)<gamma*sqrt(Ij(j)) % If h is in C_I, then do nothing
20              Q(:,j) = zeros(size(Q(:,j)));
21          else %project h onto C_I
22              h=h-sign(c)*(abs(c)-gamma*sqrt(Ij(j)))*e(:,j)/Ij(j);
23              Q(:,j)=-sign(c)*(abs(c)-gamma*sqrt(Ij(j)))*e(:,j)/Ij(j);
24          end
```

```
    end

    % cycle 2
    k=1;
    r=norm(h-h_old,2);
    while r>eps
        k=k+1;
        h_old = h;
        rho=0;

        for j=1:M
            c=dot(h-Q(:,j), e(:,j));
            lambda=c/sqrt(Ij(j));
            q_old=Q(:,j);

            if abs(c)<gamma*sqrt(Ij(j)) % If h is in C_I, then do nothing
                h=h-Q(:,j);
                Q(:,j) = zeros(size(Q(:,j)));
            else %project h onto C_I
                h=h-Q(:,j)-sign(c)*(abs(c)-gamma*sqrt(Ij(j)))*e(:,j)/Ij(j);
                Q(:,j)=-sign(c)*(abs(c)-gamma*sqrt(Ij(j)))*e(:,j)/Ij(j);
            end

            rho=max(rho, max(abs(Q(:,j)-q_old)));
        end
        r=max(norm(h-h_old,2), rho);
    end
end
```

```
function h=dykstra_dyadic(h, e , gamma, eps)
% This function projects h onto the intersection of the subsets C_I defined by
    the matrix with all the eigenvectors e, by using Dykstras projection
    algorithm. This implementation manages only dyadic subsets.
% INPUT: h- vector
%        e- matrix with eigenvectors to the subsets C_I
%        gamma- constant,  multiresolution bound
%        eps- stopping criteria
% OUTPUT h- vector with h

% Initializations
    n=length(h);
    J=log2(n)+1;
    Q=zeros(n, J);
    j=1;
    h_old=h;

    % First cycle
    for k=1:J
        e=ones(j,1);
        % Performs all the projection on one level at once
        t=sum(reshape(h,j,n/j),1);
        s=t-min(gamma*sqrt(j), max(-gamma*sqrt(j), t));
        h=h-reshape(e*s,n,1)/j;
```

```
        Q(:,k)=-reshape(e*s,n,1)/j;
        j=2^k;
    end

        % Second cycle
    r= norm(h-h_old,2);
    while r>eps
        j=1;
        h_old=h;
        rho=0;
        for k=1:J
                q_old=Q(:,k);
                e=ones(j,1);

                % Performs all the projection on one level at once
                t=sum(reshape(h-Q(:,k),j,n/j),1);
                s=t-min(gamma*sqrt(j), max(-gamma*sqrt(j), t));
                q=reshape(e*s,n,1)/j;
                h=h-Q(:,k)-q;
                Q(:,k)=-q;

                rho=max(rho, max(abs(Q(:,k)-q_old)));
                j=2^k;
        end
        r=max(norm(h-h_old,2), rho);
    end
end
```

```
function D=forwardDifferences(n)
% This function discretizes the nabla operator with forward differences in one
    dimension
%
% INPUT n- number of grid points
% OUTPUT D- matrix

        D=n*sparse(diag(-ones(n,1))+diag(ones(n-1,1),1));
end
```

```
function D=laplace(n)
% This function discretizes the laplace operator with forward differences in
   one dimension
%
% INPUT n- number of grid points
% OUTPUT D- matrix

        D=(n*n)*(diag(2*ones(n,1)) - diag(ones(n-1,1),1) - diag(ones(n-1,1)
            ,-1));
        D(1,1)=1*(n*n);
        D(end,end)=1*(n*n);
end
```

C.2 Quadratic regulerization in two dimensions

In this section, the implementation of the Douglas-Rachford algorithm from chapter 4.2 is presented. First, the implementation of Douglas-Rachford algorithm is presented, then the functions that Douglas-Rachford algorithm need to run.

```
function [p, q]=DouglasRachford(Y, F, gamma, mu, tol)
% This function removes noise from a two dimensional signal by quadratic
    regularization where the constraint is the multiresolution norm. The
    algorithm used is Douglas-Rachford splitting algorithm.
%
% INPUT Y - a two dimensional signal on matrix form.
%       F - a matrix with the indeces for the index sets the multiresolution
    norm need. (Each column is the indeces for one square)
%       gamma - a constant, the multiresolution bound
%       mu - step length
%       eps - stopping criteria
% OUTPUT p - the denoised signal
%       q- the removed noise

        % Initialization
        [N, M]=size(Y);
        u=reshape(Y, N*M,1); % reshape the Y so it is a vector
        FY=-F*u;
        A=defineA(F, N, M, mu) ;
        d=forwardDiff(N, M);
        v=FY;
        sl=100;
        p=u;

        while sl>tol
                p_old=p;

                % prox_h
                w=min(gamma, max(v, -gamma));

                % prox_g
                qp=flip(A,2)\[2*u-u;2*w-v;FY];
                pq=flipud(qp);
                p=pq(1:length(u));
                q=pq(length(u)+1:length(u)+length(FY));

                % calculates u_n+1 and v_n+1
                y=[u;v]+mu*(pq(1:(length(u)+length(FY)))-[u;w]);
                u=y(1:length(u));
                v=y(length(u)+1:end);

                % Calculates the change in p
                sl=norm((p-p_old),2);
    end
    % Reshape the result p and q to a matrix
    p=reshape(p, N, M);
```

```
    q=reshape(q, N, M);
end
```

```
function A=defineA(F, N, M, mu)
% This function defines the matrix A in prox_h used by the function
    DouglasRachford.
%
% INPUT F - a matrix with the indeces for the index sets the multiresolution
    norm need. (Each column is the indeces for one square)
%       N - number of grid points horizontally
%       M - number of grid points vertically
%       mu -a constant, the multiresolution bound
% OUTPUT A - a matrix

    laplace=laplaceOperator(N, M);
    [row, col]=size(F);
    A=sparse(col+row+row,col+row+row );
    A(1:col,1:col)=speye(col)-mu*laplace; %(I-mu*laplace)p
    A(1:col, col+row+1:end)=-F'; %-Ftrans*lambda
    A(col+1:col+row, col+1:col+row)=speye(row); %q
    A(col+row+1:col+row+row, col+1:col+row)=speye(row); %lambda
    A(col+row+1:end, 1:col)=-F; % -F*p
    A(col+1:col+row, col+row+1:col+row+row)=speye(row); %q
end
```

```
function laplace=laplaceOperator(N,M)
% This function discretizes the  laplace operator with forward differences in
    two dimensions
%
% INPUT N - number of grid points horizontally
%       M - number of grid points vertically
% OUTPUT laplace- matrix

    e=ones(N*M,1);
    laplace = spdiags([e -4*e e], -1:1, N*M, N*M)+spdiags([e], -N, N*M, N*M)+
        spdiags([e], N, N*M, N*M);
    e=ones(M,1);
    laplace(1:M,1:M)=spdiags([e -3*e e], -1:1, M, M);
    laplace(N*M-M+1 : N*M, N*M-M+1 : N*M)=spdiags([e -3*e e], -1:1, M, M);
    for i=N:N:N*M-1
        laplace(i, i+1)=0;
        laplace(i+1, i)=0;
        laplace(i, i) = -3;
         laplace(i+1 , i+1) = -3;
    end
    laplace(1,1)=-2;
    laplace(M,M)=-2;
    laplace(N*M-M+1,N*M-M+1)=-2;
    laplace(N*M, N*M)=-2;
    laplace=N*M*laplace;
end
```

```
function d=forwardDiff(N, M)
```

```
% This function discretizes the nabla operator with forward differences in two
    dimensions
%
% INPUT N - number of grid points horizontally
%       M - number of grid points vertically
% OUTPUT d- matrix

    e=ones(N*M,1);
    d = spdiags([-2*e e], 0:1, N*M, N*M)+spdiags([e], N, N*M, N*M);
    e=ones(M,1);
    d(1:M,1:M)=spdiags([-e e], 0:1, M, M);
    d(N*M-M+1 : N*M, N*M-M+1 : N*M)=spdiags([-e e], 0:1, M, M);
    for i=N:N:N*M-1
        d(i, i+1)=0;
        d(i,i)=-1;
    end
    d(N*M, N*M)=0;
    d=N*d;
end
```

C.3 Total variation regularization in two dimensions

In this section, the implementations of the Douglas-Rachford algorithm and Chambolle's projection algorithm from chapter 4.3 is presented. First is the implementation of Douglas-Rachford algorithm presented and then Chambolle's projection algorithm. At the end are the functions Chambolle's projection algorithm uses. The functions Douglas-Rachford algorithm uses are the same as the implementations the Douglas-Rachford algorithm for the quadratic regularization problem uses and is presented in the previous section.

```
function [eta, zeta]=douglasRachford(Y, F, gamma, mu, tol, tol_chambolle,
    mu_chambolle, tau_chambolle)
% This function removes noise from a two dimensional signal by constrained
    optimization, where a total variation norm is the regularization term
    andthe constraint is the multiresolution norm. The algorithm used is
    Douglas-Rachford splitting algorithm. Douglas-Rachford splitting algorithm
    uses Chambolle's projection algorithm to minimize the total variation norm.
%
% INPUT Y - a two dimensional signal on matrix form.
%       F - a matrix with the indeces for the index sets the multiresolution
    norm need. (Each column is the indeces for one square)
%       gamma - a constant, the multiresolution bound
%       mu - step length for Douglas-Rachford algorithm
%       tol - stopping criteria for Douglas-Rachford algorithm
%       mu_chambolle - a step length in Chamolle's projection algorithm
%       tol_chambolle - stopping criteria for Chamolle's projection algorithm
% tau_chambolle- step length for the gradien projection algorithm in Chamolle'
    s projection algorithm
% OUTPUT eta - the denoised signal
%       zeta- the removed noise
```

```
14
15          % Initialization
16      [N, M]=size(Y);
17      u=reshape(Y, N*M, 1);% reshape the Y so it is a vector
18      FY=-F*Y;
19      v=FY;
20          cham=ones(N*M, 2);
21      d=forwardDiff(N, M);
22
23     % Define the matrix A used by prox_h
24      A1=[F, -speye(length(FY))];
25      A2=[speye(length(u)), F'];
26      A=[A1;A2];
27
28      % Define the div and nabla operator for Chambolle's projection algorithm
29      [divx, divy]=div(N, M);
30      [dx, dy]=dxdy(N, M);
31
32          eta=u;
33          sl=100;
34      while sl>tol
35          u_old=u;
36          eta_old=eta;
37
38                  % prox_h
39          b=[-FY; F'*v+u];
40          EtaZeta=A\b;
41
42          % prox_g
43          eta=(pq(1:N*M));
44          zeta=(pq(N*M+1:end));
45          [tempu, cham]=chambolle(2*eta-u,cham, tol_chambolle, mu_chambolle,
                tau_chambolle, N, M, divx, divy, dx, dy);
46           tempv=min(gamma, max(2*zeta-v, -gamma));
47
48          % Calculates u_n+1 and v_n+1
49          u=u+lambda*(tempu-eta);
50          v=v+lambda*(tempv-zeta);
51
52                  % Calculates the change in p
53          sl=norm((p-p_old),2);
54      end
55      % Reshape the result p and q to a matrix
56      eta=reshape(p, N, M);
57      zeta=reshape(q, N, M);
58  end

1   function [u, p]=chambolle(u, p, tol_C, mu, tau, N, M, divx, divy, dx, dy)
2   % This function uses Chambolle's projection algorithm to minimize total
        variation in two dimensions.
3   %
4   % INPUT u- the signal two dimensional signal minimize on vector form
5   %        p- the result p form Chambolle from the previous iteration of Douglas-
```

```
    Rachford
% tol- stopping criteria
%       mu- a step length
%       tau - step length of the projected gradient method used
% N, M- number of grid points, horizontally and vertically, respectively
%       divx, divy - x and y component of the div operator,( found by the
    function div(N,M))
%       dx, dy - dicretization of the nabla operator (found by the function
    dxdy(N,M))
% OUTPUT u- minimized u
%       p- a temp variable returned to use as initial variable for p in the
    next run of Chambolle in the next iteration of Douglas-Rachford. (Shortens
    the running time of Chambolle)

    normp=@(x) (x(:,1).^2+x(:,2).^2).^(1/2);

        % Initialization
    w=zeros(N*M, 1);
    sl=100;

        % First iteration
    w=u/mu+divx*p(:,1)+divy*p(:,2);

    sl=max(abs(v-u));
        while sl>tol
        w_old=w;
        tempP=[p(:, 1)+tau.*dx*w, p(:, 2)+tau.*dy*w];
        tempMax=max(1,normp(tempP));
        p(:,1)=tempP(:,1)./tempMax;
        p(:,2)=tempP(:,2)./tempMax;
        w=u/mu+divx*p(:,1)+divy*p(:,2);

        % Finds the change in v
        sl=max(abs(w-w_old));
    end
    % Final result of u
    u=w*mu;
end
```

```
function [divx, divy]=div(N,M)
% Dicretize the div operator used in Chambolle's projection algorithm.
%
% INPUT N, M- number of gridpoints, horizonlaly and vertically, respectively.
% OUTPUT divx, divy - x and y component of the div operator, respectively.

        e=ones(N*M,1);
    divy= spdiags([-e e], -1:0, N*M, N*M);
    divx=sparse(N*M, N*M);
    divx(1:N*M-N, 1:N*M-N)= speye(N*M-N);
    divx(1:N*M, 1:N*M)=divx(:,:)-spdiags([e], -N, N*M, N*M);

    for i=N:N:N*M-1
        divy(i,i)= 0;
```

```
        divy(i+1,i)=0;
    end
    divy(N*M, N*M)=0;
end
```

```
function [dx, dy]=dxdy(N, M)
% Dicretize the nabla operator in the total variation, used in Chambolle's
    projection algorithm.
%
% INPUT N, M- number of gridpoints, horizonlaly and vertically, respectively.
%        OUTPUT dx, dy - x and y component of the nabla operator, respectively.

    e=ones(N*M,1);
    dy= spdiags([-e e], 0:1, N*M, N*M);
    dx= -speye(N*M)+spdiags([e], N, N*M, N*M);
    for i=N:N:N*M
        dy(i,:)=zeros(1, N*M);
    end
    dx((N*M-N+1):N*M, :)=zeros(N, N*M);
end
```

Bibliography

[1] Jean-François Aujol. Some first-order algorithms for total variation based image restoration. *J. Math. Imaging Vision*, 34(3):307–327, 2009.

[2] James P. Boyle and Richard L. Dykstra. A method for finding projections onto the intersection of convex sets in Hilbert spaces. In *Advances in order restricted statistical inference (Iowa City, Iowa, 1985)*, volume 37 of *Lect. Notes Stat.*, pages 28–47. Springer, Berlin, 1986.

[3] Antonin Chambolle. An algorithm for total variation minimization and applications. *J. Math. Imaging Vision*, 20(1):89–97, 2004. Special issue on mathematics and image analysis.

[4] Tony F. Chan and Jianhong Shen. *Image processing and analysis*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2005. Variational, PDE, wavelet, and stochastic methods.

[5] Patrick L. Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, volume 49 of *Springer Optim. Appl.*, pages 185–212. Springer, New York, 2011.

[6] David L. Donoho and Iain M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, 1994.

[7] Klaus Frick, Philipp Marnitz, and Axel Munk. Statistical multiresolution Dantzig estimation in imaging: fundamental concepts and algorithmic framework. *Electron. J. Stat.*, 6:231–268, 2012.

[8] Markus Grasmair, Markus Haltmeier, and Otmar Scherzer. The residual method for regularizing ill-posed problems. *Applied Mathematics and Computation*, 218(6):2693 – 2710, 2011.

[9] Markus Grasmair, Housen Li, and Axel Munk. Variational multiscale nonparametric regression: Smooth functions. *To appear in: Ann. Inst. Henri Poincaré Probab. Stat.*

[10] Osman Güler. *Foundations of optimization*, volume 258 of *Graduate Texts in Mathematics*. Springer, New York, 2010.

[11] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Fundamentals of convex analysis*. Grundlehren Text Editions. Springer-Verlag, Berlin, 2001. Abridged version of ıt Convex analysis and minimization algorithms. I [Springer, Berlin, 1993; MR1261420 (95m:90001)] and ıt II [ibid.; MR1295240 (95m:90002)].

[12] Roger Temam Ivar Ekeland. *Convex Analysis and Variational Problems*. Elsevier, 1976.

[13] Arkadi Nemirovski. Nonparametric estimation of smooth regression functions. *Izv. Akad. Nauk. SSR Teckhn. Kibernet. (in Russian)*, 3:50–60, 1985. *J. Comput. System Sci.*, 23:1–11, 1986 (in English).

[14] Arkadi Nemirovski. Topics in non-parametric statistics. In *Lectures on probability theory and statistics (Saint-Flour, 1998)*, volume 1738 of *Lecture Notes in Math.*, pages 85–277. Springer, Berlin, 2000.

[15] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, second edition, 2006.

[16] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259 – 268, 1992.